National Technical University of Athens

School of Electrical and Computer Engineering
division of signals, control and robotics

Multi-task learning for Action and Gesture Recognition

Diploma Thesis

of

Konstantinos Spathis

Supervisor: Petros Maragos
Professor, NTUA

computer vision, speech communication and signal processing group
Athens, July 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Computer Vision, Speech Communication and Signal Processing Group

Multi-task learning for Action and Gesture Recognition

Diploma Thesis

of

Konstantinos Spathis

Supervisor: Petros Maragos
Professor NTUA

Approved by the Examining Committee on the 18th July, 2024.

.....	.....	.....
Petros Maragos	Athanasios Rontogiannis	Ioannis Kordonis
Professor NTUA	Associate Professor NTUA	Assistant Professor NTUA

Athens, July 2024

.....

Konstantinos Spathis
Electrical and Computer Engineering Graduate, NTUA

Copyright © – All rights reserved Konstantinos Spathis, 2024.
All rights reserved.

This work is copyright and may not be reproduced, stored nor distributed in whole or in part for commercial purposes. Permission is hereby granted to reproduce, store and distribute this work for non-profit, educational and research purposes, provided that the source is acknowledged and the present copyright message is retained. Enquiries regarding use for profit should be directed to the author.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the National Technical University of Athens.

to my family

Περίληψη

Με την ανάπτυξη των αρχιτεκτονικών βαθιάς μάθησης έχουν επιτευχθεί αξιοσημείωτες επιδόσεις σε διάφορα προβλήματα/εργασίες της όρασης υπολογιστών, συμπεριλαμβανομένης της αναγνώρισης δράσεων και χειρονομιών. Στόχος αυτών των προβλημάτων είναι να εξάγουν σημασιολογικά χρήσιμη πληροφορία από οπτικά δεδομένα. Οι μεθοδολογίες, που προτείνονται, για την αναγνώριση δράσεων και χειρονομιών επικεντρώνονται στην εφαρμογή νέων αρχιτεκτονικών βαθιάς μάθησης για την επίτευξη καλύτερων επιδόσεων, ενώ αντιμετωπίζουν αυτά τα πρόβλημα ξεχωριστά. Όμως, αυτά τα προβλήματα βρίσκουν εφαρμογή σε διάφορους τομείς όπου απαιτείται η αναγνώριση τόσο των δράσεων όσο και των χειρονομιών, όπως για παράδειγμα στους ρομποτικούς υποβοηθούς, σε συστήματα επιτήρησης ή στην αυτόνομη οδήγηση όπου επιδιώκεται ο εντοπισμός και η ανίχνευση αντικειμένων/ανθρώπων ταυτόχρονα. Επομένως, η αλληλοεπικάλυψη, που έχουν αυτά τα προβλήματα δημιουργεί την ανάγκη για την απο κοινού επίλυσή τους, με τη δημιουργία αλγορίθμων και αρχιτεκτονικών, που τα επιλύουν ταυτόχρονα.

Επιπλέον, η ανάπτυξη μοντέλων βαθιάς μάθησης απαιτεί τη συλλογή μεγάλου όγκου δεδομένων, το οποίο είναι συχνά δύσκολο και χρονοβόρο. Για αυτό έχουν προταθεί εναλλακτικές που αξιοποιούν την πληροφορία από πολλαπλές εργασίες για τη βελτίωση της απόδοσης των μοντέλων βαθιάς μάθησης. Μία από αυτές είναι η "μάθηση πολλαπλών εργασιών", όπου πολλαπλές εργασίες/προβλήματα μαθαίνονται να επιλύονται από κοινού, μοιράζοντας πληροφορία μεταξύ τους. Η μάθηση πολλαπλών εργασιών έχει εφαρμοστεί με επιτυχία σε προβλήματα με βίντεο, που απεικονίζουν ανθρώπινες κινήσεις, όπως η αναγνώριση δράσεων και η εκτίμηση πόζας, καθώς και σε προβλήματα με δεδομένα χειρονομιών, τα οποία συνδυάζουν την αναγνώριση και εντοπισμό χειρονομιών.

Σε αυτή την εργασία, στοχεύουμε να δείξουμε ότι οι εργασίες αναγνώρισης δράσεων και χειρονομιών μπορούν να αντιμετωπιστούν με τη χρήση μιας εννιαίας αρχιτεκτονικής. Κατασκευάζουμε διαφορετικά μοντέλα μάθησης πολλαπλών εργασιών, όπου οι εργασίες αναγνώρισης δράσεων και χειρονομιών μαθαίνονται από κοινού. Αξιολογούμε την απόδοση των προτεινόμενων μοντέλων σε σύγκριση με τα αντίστοιχα μοντέλα μονής εργασίας. Τα αποτελέσματα δείχνουν ότι τα προτεινόμενα μοντέλα επιτυγχάνουν καλύτερη απόδοση σε σύγκριση με τα μοντέλα μονής εργασίας, δείχνοντας τα οφέλη της μάθησης πολλαπλών εργασιών για την ταυτόχρονη επίλυση των προβλημάτων της αναγνώρισης δράσεων και χειρονομιών. Επιπλέον, επεκτείνουμε αυτή τη μέθοδο για να αναπτύξουμε ένα πολυτροπικό μοντέλο μάθησης πολλαπλών εργασιών, όπου διαφορετικού τύπου δεδομένα, συγκεκριμένα δεδομένα από έγχρωμες κάμερες και αισθητήρες βάθους, μπορούν να εκπαιδεύονται από κοινού στο ίδιο μοντέλο, για να επιτύχουν καλύτερη απόδοση σε σύγκριση με τα μοντέλα μονής εργασίας και τις αντίστοιχες πολυτροπικές προσεγγίσεις τους.

Λέξεις Κλειδιά — βαθιά μάθηση, όραση υπολογιστών, αναγνώριση δράσεων, αναγνώριση χειρονομιών, μάθηση πολλαπλών εργασιών

Abstract

The recent advances in deep learning have revolutionized the field of computer vision. Deep learning models have achieved state-of-the-art performance in various tasks, including action and gesture recognition. These two human-centric tasks involve the recognition of human actions and gestures in videos, aiming to mathematically model the human perception of actions and gestures. The current state-of-the-art models for action and gesture recognition focus on applying novel deep learning architectures to achieve better performance, while handling each task separately. However, these tasks find application in various fields where the recognition of both actions and gestures is required, as it arises for example with robotic assistants, surveillance systems, or autonomous driving, where object/human detection and recognition are required simultaneously. Therefore, these problems show great overlap, requiring common algorithms that address both of them at the same time.

Recently, alternative approaches of learning methods have been proposed to improve the performance of deep learning models, without requiring the development of novel architectures or the collection of more data. One of these approaches is "multi-task learning", where multiple tasks are learned jointly, sharing information between them. Multi-task learning has been successfully applied to various computer vision tasks. Tasks including actions, such as action recognition and pose estimation have been shown to benefit from multi-task learning. While in the field of gesture recognition, multi-task learning has also been applied to tasks such as hand gesture recognition and segmentation, achiving remarkable results.

In this thesis, we aim to show that the tasks of action and gesture recognition can be learned jointly, benefiting from each other. We constuct different multi-task learning models, where the tasks of action and gesture recognition are learned jointly. We evaluate the performance of the proposed models on the respective single-task learning models for action and gesture recognition. The results show that the proposed models achieve better performance compared to the single task models, demonstrating the benefits of multi-task learning in action and gesture recognition. Moreover, we extent this method to develop a multimodal multi-task learning model, where different modalities, specifically rgb and depth data, can be learnt jointly in the same framework, to achieve better performance in comparison to single task models and multimodal approaches.

Keywords — deep learning, computer vision, action recognition, gesture recognition, multi-task learning

Acknowledgments

First of all, I would like to thank my supervisor, Prof. Petros Maragos, and co-supervisor, Nikolaos Kardaris, for their guidance and support. I am grateful for the collaboration and the knowledge I gained from them. Also, I would like to thank the lab members of the "Computer Vision, Speech Communication and Signal Processing Group" for letting me be part of their scientific research

Last but not least, I would like to thank my family for their support and encouragement throughout my life. All these years they have been there for me, giving everything they could to help me chase my dreams. All the sacrifices they made, will never be forgotten. May this thesis be a small token for even greater things to come to show my gratitude, because ...

there is no feeling like other, than doing what you love!

Konstantinos Spathis
July 2024

Contents

List of Figures

List of Tables

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Εισαγωγή

1.1.1 Κίνητρο Διπλωματικής Εργασίας

Τα τελευταία χρόνια, το πεδίο της Βαθιάς Μάθησης έχει δει μια σημαντική ανάπτυξη και σημαντικά αποτελέσματα σε διάφορα προβλήματα. Στον τομέα της όρασης υπολογιστών, προβλήματα όπως η αναγνώριση δράσεων και η αναγνώριση χειρονομιών έχουν ενισχυθεί από την εφαρμογή συγχρόνων τεχνικών, οδηγώντας σε πληθώρα εφαρμογών σε διάφορες πτυχές της καθημερινής μας ζωής. Η αυτόνομη οδήγηση δεν θα είχε προσελκύσει το ίδιο ενδιαφέρον αν δεν είχαν αναπτυχθεί αλγόριθμοι αναγνώρισης δράσεων, ενώ οι ρομποτικοί βοηθοί δεν θα μπορούσαν να καταλάβουν τις εντολές των ανθρώπων αν δεν είχαν αναπτυχθεί αλγόριθμοι αναγνώρισης χειρονομιών. Αυτά τα δύο προβλήματα είναι ανθρωποκεντρικά καθώς περιλαμβάνουν την αναγνώριση ανθρώπινων κινήσεων και χειρονομιών, αλλά ακόμα και σήμερα θεωρούνται ως ξεχωριστά προβλήματα κατά την εκπαίδευση ενός νευρωνικού δικτύου.

Γενικά, οι αλγόριθμοι μάθησης μηχανής δημιουργούνται για να αντικατοπτρίζουν τον τρόπο σκέψης των ανθρώπων. Όταν ένα άτομο αντιμετωπίζει ένα πρόβλημα, χρησιμοποιεί γνώση από όλες τις προηγούμενες σχετικές καταστάσεις στις οποίες έχει βρεθεί. Για να επιτευχθεί αυτή η συμπεριφορά στη μηχανική μάθηση, έχουν αναπτυχθεί διάφορες τεχνικές διαμοιρασμού πληροφοριών μεταξύ διαφορετικών προβλημάτων. Μια δημοφιλής μέθοδος για τον διαμοιρασμό πληροφοριών είναι η μάθηση πολλαπλών εργασιών. Στη μάθηση πολλαπλών εργασιών, σχετικά προβλήματα εκπαιδεύονται από κοινού, βοηθώντας το ένα το άλλο να επιτύχουν καλύτερα αποτελέσματα και να συγκλίνουν πιο γρήγορα σε βέλτιστες λύσεις.

Το γεγονός ότι σε προβλήματα αναγνώρισης δράσεων οι δράσεις περιλαμβάνουν συχνά χειρονομίες και ομοίως σε προβλήματα αναγνώρισης χειρονομιών οι χειρονομίες περιλαμβάνουν συχνά δράσεις, μας οδήγησε στο ερώτημα αν μπορούν να εκπαιδευτούν ταυτόχρονα σε ένα νευρωνικό δίκτυο.

1.1.2 Συνεισφορά Διπλωματικής Εργασίας

Στόχος αυτής της διπλωματικής εργασίας είναι να αξιολογηθεί η αποτελεσματικότητα των αλγορίθμων μάθησης πολλαπλών εργασιών στα προβλήματα αναγνώρισης δράσεων και αναγνώρισης χειρονομιών. Για να επιτευχθεί αυτός ο στόχος, πραγματοποιείται μια εκτενής αξιολόγηση διαφορετικών αρχιτεκτονικών μάθησης πολλαπλών εργασιών, καθώς και οι επιπτώσεις διαφορετικών μεθόδων υπολογισμού της συνάρτησης σφάλματος πολλαπλών εργασιών εξετάζονται. Επιπλέον, η διερεύνηση αυτή επεκτείνεται για να αξιολογηθεί πώς μπορούν να χρησιμοποιηθούν διαφορετικοί τύποι δεδομένων σε μια αρχιτεκτονική μάθησης πολλαπλών εργασιών. Πιο συγκεκριμένα, προτείνεται ένα μοντέλο μάθησης πολλαπλών εργασιών και επεξεργασίας πολλαπλών τύπων δεδομένων για δεδομένα RGB και βάθους με τρόπο, που να εκπαιδεύονται από κοινού και να χρησιμοποιούνται και οι δύο τύποι δεδομένων κατά τη διαδικασία αξιολόγησης. Συνοπτικά, οι κύριες συνεισφορές αυτής της εργασίας περιγράφονται παρακάτω.

- Αξιολογούμε αν η μάθηση πολλαπλών εργασιών μπορεί να χρησιμοποιηθεί για την κοινή εκπαίδευση των εργασιών αναγνώρισης δράσεων και χειρονομιών.
- Αξιολογούμε τις επιπτώσεις διαφορετικών αρχιτεκτονικών μάθησης πολλαπλών εργασιών στο πρόβλημα κοινής εκπαίδευσης των εργασιών αναγνώρισης δράσεων και χειρονομιών.
- Αξιολογούμε τις επιπτώσεις διαφορετικών μεθόδων υπολογισμού της συνάρτησης σφάλματος πολλαπλών εργασιών στο πρόβλημα κοινής εκπαίδευσης των εργασιών αναγνώρισης δράσεων και χειρονομιών.
- Προτείνουμε ένα μοντέλο μάθησης πολλαπλών εργασιών και πολλαπλών τύπων δεδομένων για να εκπαιδευτούν από κοινού δεδομένα RGB και βάθους, και να χρησιμοποιούνται και οι δύο τύποι δεδομένων για τη διαδικασία της αξιολόγησης.

1.2 Θεωρητικό Υπόβαθρο

1.2.1 Όραση Υπολογιστών

Η Όραση Υπολογιστών (Computer Vision) είναι η επιστήμη, που ασχολείται με την ικανότητα των υπολογιστών να βλέπουν. Στόχος αυτής της επιστήμης είναι να προσομοίωσει τις ικανότητες της όρασης των βιολογικών οντοτήτων. Συγκεκριμένα, ο υπολογιστής καλείται με χρήση αισθητήρων που ποσοτικοποιούν το προσπίπτον φως και με μαθηματικούς αλγορίθμους να αντιληφθεί τα αντικείμενα γύρω του, όπως ένας οργανισμός μπορεί να διαχωρίσει τα αντικείμενα στο περιβάλλον του από το φως, που αντανακλάται πάνω στα αντικείμενα και εισέρχεται στα όργανα όρασης του οργανισμού αυτού.

Εικόνες και Βίντεο

Μια εικόνα είναι μια οπτική αναπαράσταση μιας έννοιας του κόσμου, που δημιουργείται μέσω της καταγραφής και αναπαραγωγής του φωτός. Η διαδικασία δημιουργίας μιας εικόνας περιλαμβάνει τη χρήση φωτοευαίσθητων αισθητήρων, που συλλέγουν φως από το περιβάλλον και τα μετατρέπουν σε ψηφιακά σήματα για την αναπαραγωγή τους. Για τους υπολογιστές, μια εικόνα ερμηνεύεται ως ένας πίνακας, που αποτελείται από έναν πεπερασμένο αριθμό στοιχείων, τα οποία αντιστοιχίζονται σε θέσεις σε ένα καρτεσιανό επίπεδο με συντεταγμένες x, y και η τιμή τους αντιστοιχεί στην ένταση τους χρώματος της εικόνας σε αυτό το σημείο. Για παράδειγμα, για ασπρόμαυρες εικόνες, κάθε στοιχείο του πίνακα αντιστοιχεί σε μια τιμή έντασης της γκρίζας απόχρωσης της εικόνας με την τιμή 0 να αντιστοιχεί στο μαύρο και την τιμή 255 να αντιστοιχεί στο λευκό. Ομοίως, για έγχρωμες εικόνες κάθε στοιχείο του πίνακα αντιστοιχεί σε μια τριάδα τιμών, που υποδεικνύει την ένταση του χρώματος κάθε χρωματικού καναλιού. Οι έγχρωμες εικόνες αναπαρίστανται συνήθως με τριάδας καναλιών, δηλαδή πινάκων που αντιστοιχούν σε τιμές χρώματος, όπως το RGB, όπου κάθε κανάλι αντιστοιχεί στην ένταση του κόκκινου, του πράσινου και του μπλε χρώματος αντίστοιχα, καθώς με αυτή την τριάδα μπορεί να αναπαρασταθεί κάθε απόχρωση από το φάσμα του χρώματος. Συνεπώς, οι εικόνες περιγράφονται μαθηματικά ως πίνακες διαστάσεων (C, H, W), όπου C δηλώνει τον αριθμό των καναλιών που χρησιμοποιούνται για την απεικόνιση του χρώματος, H δηλώνει τις γραμμές και W δηλώνει τις στήλες του πίνακα. Αυτή η κατανόηση επεκτείνεται στην ερμηνεία των βίντεο, όπου γίνονται αντιληπτοί ως μια ακολουθία εικόνων και περιγράφονται μαθηματικά ως πίνακες διαστάσεων (D, C, H, W), όπου η επιπλέον διάσταση D δηλώνει τον αριθμό των εικόνων που χρησιμοποιούνται για την κατασκευή του βίντεο.

Προβλήματα και Εφαρμογές

Η Όραση Υπολογιστών εστιάζει στην επίλυση προβλημάτων, με χρήση μαθηματικών αλγορίθμων, που επεξεργάζονται εικόνες και βίντεο. Κάποια από τα πιο δημοφιλή προβλήματα, που αντιμετωπίζονται στον τομέα αυτό παρουσιάζονται ακολούθως.

- **Η Ταξινόμηση Εικόνων και Βίντεο**
  είναι ίσως το πιο δημοφιλές πρόβλημα στην Όραση Υπολογιστών, καθώς και αυτό στο οποίο έχει πραγματοποιηθεί η περισσότερη έρευνα. Στόχος αυτού του προβλήματος είναι ο αλγόριθμος να ταξινομήσει μια εικόνα ή ένα βίντεο σε μια από τις προκαθορισμένες κατηγορίες, που έχουν οριστεί.
- **Η Ανίχνευση Αντικειμένων**
  στοχεύει στην ανίχνευση της συγκεκριμένων αντικειμένων σε μια εικόνα ή ένα βίντεο, καθώς και στην παροχή πλαισίων γύρω από τα αντικείμενα αυτά, όταν αυτά εντοπίζονται.
- **0 Εντοπισμός Σημαντικής Πληροφορίας**
  στοχεύει στην ανίχνευση της σημαντικής πληροφορίας σε μια εικόνα ή ένα βίντεο, δηλαδή η εύρεση των pixel που ανήκουν σε ανθρώπινες μορφές ή σε αντικείμενα ενδιαφέροντος, καθορίζοντας με αυτό τον τρόπο τα σύνορά τους.

Η Όραση Υπολογιστών εφαρμόζεται σε πολλούς τομείς της καθημερινής ζωής, κάποιοι από τους οποίους παρουσιάζονται παρακάτω.

- **Η αλληλεπίδραση ανθρώπου - υπολογιστή**
  εστιάζει στο σχεδιασμό διεπαφών, που επιτρέπουν την αποτελεσματική και φιλική προς τον χρήστη επικοινωνία μεταξύ ανθρώπου και υπολογιστή. Με τη χρήση αλγορίθμων όρασης υπολογιστών αυτές οι διεπαφές παρέχουν περισσότερες λειτουργίες και διευκολύνουν ακόμα περισσότερη τη χρήση των υπολογιστών.

- **Η Επαυξημένη και Εικονική Πραγματικότητα**
  είναι τεχνολογίες, που επεκτείνουν την αντίληψη του χρήστη, προσθέτοντας ψηφιακές πληροφορίες, όπως εικόνες, κείμενο ή 3D μοντέλα, στον πραγματικό κόσμο. Η Επαυξημένη Πραγματικότητα προσθέτει ψηφιακές πληροφορίες στον πραγματικό κόσμο, ενώ η Εικονική Πραγματικότητα δημιουργεί ένα εντελώς ψηφιακό περιβάλλον, στο οποίο ο χρήστης μπορεί να αλληλεπιδρά με το περιβάλλον αυτό.

- **Τα Αυτόνομα Οχήματα**
  είναι οχήματα, που μπορούν να πλοηγηθούν και να λειτουργήσουν χωρίς την ανθρώπινη παρέμβαση. Τα αυτόνομα οχήματα χρησιμοποιούν αλγόριθμους όρασης υπολογιστών για την ανίχνευση των αντικειμένων γύρω τους και την πλοήγηση τους στο περιβάλλον τους.

- **Η Επιτήρηση και Ασφάλεια**
  περιλαμβάνουν συστήματα, που αποσκοπούν στην προστασία των ανθρώπινων και ενισχύονται με τεχνικές όρασης υπολογιστών, αφού καθιστούν όλο και λιγότερο αναγκαίο τον ανθρώπινο παράγοντα για την επιτήρηση των περιβαλλόντων.

Είδη Δεδομένων

Στην Όραση Υπολογιστών η επεξεργασία εικόνων και βίντεο γίνεται επάνω σε δεδομένα διαφόρων ειδών. Τα πιο δημοφιλή είδη δεδομένων παρουσιάζονται παρακάτω [1].

- **Τα RGB δεδομένα**
  η αλλιώς Red Green Blue είναι εικόνες ή βίντεο, που έχουν καταγραφεί από RGB κάμερες, με σκοπό να αναπαραστήσουν αυτό που βλέπουν τα ανθρώπινα μάτια. Αυτά τα δεδομένα συλλέγονται εύκολα, όμως η χρήση αλγορίθμων πάνω σε αυτά είναι απαιτητική λόγω των διακυμάνσεων των φόντων, των γωνιών θέασης, των κλιμάκων των ανθρώπων και των συνθηκών φωτισμού.

- **Τα δεδομένα Βάθους (Depth)**
  είναι εικόνες, που οι τιμές των pixel αντιπροσωπεύουν την απόσταση των σημείων της σκηνής από την κάμερα. Τα δεδομένα βάθους παρέχουν αξιόπιστες πληροφορίες για την 3D δομή και το γεωμετρικό σχήμα της σκηνής. Η ουσία της κατασκευής ενός χάρτη βάθους είναι να μετατραπούν τα 3D δεδομένα σε μια 2D αναπαράσταση εικόνας. Τα δεδομένα βάθους μπορούν να αποκτηθούν με τη χρήση καμερών βάθους ή με την εφαρμογή αλγορίθμων εκτίμησης βάθους σε RGB δεδομένα.

- **Τα δεδομένα Σκελετού**
  κωδικοποιούν τις τροχιές των αρθρώσεων του ανθρώπινου σώματος. Τα δεδομένα αυτά μπορούν να αποκτηθούν με συστήματα καταγραφής κίνησης ή με την εφαρμογή αλγορίθμων εκτίμησης θέσης σε RGB ή Depth εικόνες.

- **Τα Infrared δεδομένα**
  είναι εικόνες, που καταγράφονται από κάμερες θερμικής όρασης. Τα δεδομένα αυτά παρέχουν πληροφορίες για την θερμοκρασία των αντικειμένων της σκηνής και είναι ανθεκτικά στις αλλαγές φωτισμού [2].

- **Τα δεδομένα Οπτικής Ροής**
  αναπαριστούν το πρότυπο της φαινομενικής κίνησης των αντικειμένων, των επιφανειών και των ακμών σε μια οπτική σκηνή, που προκαλείται από τη σχετική κίνηση μεταξύ ενός παρατηρητή και μιας σκηνής. Η οπτική ροή είναι ένα δισδιάστατο διανυσματικό πεδίο, όπου κάθε διάνυσμα είναι ένα διάνυσμα μετατόπισης, που δείχνει την κίνηση των σημείων από την πρώτη εικόνα στη δεύτερη. Η οπτική ροή έχει πολλές εφαρμογές σε πεδία όπως η ανίχνευση και η παρακολούθηση αντικειμένων, η σταθεροποίηση βίντεο και η ανίχνευση κίνησης [3].

Πολλά προβλήματα Όρασης Υπολογιστών χρησιμοποιούν πολλαπλά είδη δεδομένων για να είναι πιο αποτελεσματικά. Η διαδικασία συνδυασμούν διαφορετικών ειδών δεδομένων ονομάζεται Πολυτροπική Συνένωση (Multimodal Fusion). Οι πιο δημοφιλείς μέθοδοι συνένωσης είναι η Αρχική, η Μεσαία και η Τελική Συνένωση.

- **Η Αρχική Συνένωση (Early Fusion)**
  αναφέρεται στη διαδικασία συνένωσης των ειδών δεδομένων στην αρχή του αλγορίθμου. Με αυτό τον τρόπο δίνεται έμφαση στην εκμάθηση χαρακτηριστικών από διάφορες πηγές δεδομένων. Ωστόσο, αυτή η προσέγγιση μπορεί να μην επιτρέψει σε κάθε είδος δεδομένων να μάθει στο μέγιστο χαρακτηριστικά από την είσοδό του.

- **Η Μεσαία Συνένωση (Mid-Level Fusion)**
  αναφέρεται στη διαδικασία συνένωσης των ειδών δεδομένων σε κάποιο ενδιάμεσο σημείο του αλγορίθμου. Αυτή η προσέγγιση ενώ επιτρέπει σε κάθε είδος δεδομένων να μάθει χαρακτηριστικά από την είσοδό του, αλλά μπορεί να είναι δύσκολο να προσδιοριστεί το βέλτιστο σημείο για τη συνένωση.
- **Η Τελική Συνένωση (Late Fusion)**
  αναφέρεται στη διαδικασία συνένωσης των ειδών δεδομένων στο τέλος του αλγορίθμου. Αυτή η προσέγγιση επιτρέπει σε κάθε είδος δεδομένων να μάθει χαρακτηριστικά από την είσοδό του, αλλά μπορεί να μην αποτυπώνει τις αλληλεπιδράσεις μεταξύ των διαφορετικών πηγών δεδομένων.

Η επιλογή της κατάλληλης μεθόδου συνένωσης εξαρτάται από τις απαιτήσεις της εργασίας και τη φύση των δεδομένων εισόδου.

1.2.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning) είναι ένας τομέας της Τεχνητής Νοημοσύνης (Artificial Intelligence), που ασχολείται με την ανάπτυξη αλγορίθμων και τεχνικών, που επιτρέπουν σε έναν υπολογιστή να χρησιμοποιήσει δεδομένα για να μάθει να παίρνει αποφάσεις, όπως σκέφτεται ένας άνθρωπος. Ανάλογα με τον τρόπο, που οι αλγόριθμοι διαχειρίζονται τα δεδομένα, καθώς και τη φύση των δεδομένων, που δίνονται, η μηχανική μάθηση μπορεί να χωριστεί στις τρεις ακόλουθες κατηγορίες [4].

- **Η επιβλεπόμενη μάθηση (supervised learning)** εκπαιδεύει μοντέλα σε δεδομένα, όπου φέρουν ετικέτες, που χαρακτηρίζουν το είδος του δεδομένου. Στόχος ενός αλγορίθμου αυτής της κατηγορίας είναι να μάθει να αντιστοιχίζει τα δεδομένα στις ετικέτες τους.
- **Η μη επιβλεπόμενη μάθηση (unsupervised learning)** ασχολείται με την εύρεση μοτίβων και σχέσεων σε δεδομένα, μη ονοματοποιημένα.
- **Η ενισχυτική μάθηση (reinforcement learning)** εκπαιδεύει μοντέλα σε δεδομένα μη ονοματοποιημένα, με έναν τρόπο ανταμοιβής-ποινής. Στόχος ενός αλγορίθμου αυτής της κατηγορίας είναι να μάθει να παίρνει αποφάσεις, που θα του επιφέρουν τη μεγαλύτερη δυνατή ανταμοιβή.

Η επιβλεπόμενη μάθηση είναι η πιο διαδεδομένη κατηγορία της μηχανικής μάθησης, καθώς οι περισσότερες εφαρμογές της μηχανικής μάθησης απαιτούν την ύπαρξη ετικετών στα δεδομένα. Η παρούσα διπλωματική εργασία, ασχολείται με προβλήματα, που υπάγονται στην επιβλεπόμενη μάθηση, όποτε θα πρέπει να ερμηνευθούν οι ακόλουθοι όροι.

Βάση Δεδομένων (Dataset)

Σε κάθε πρόβλημα μηχανικής μάθησης, οι αλγόριθμοι χρειάζονται δεδομένα για να μπορέσουν να ανταπεξέλθουν στο πρόβλημα. Αυτά τα δεδομένα μπορεί να είναι διαφορετικού τύπου, όπως αριθμητικά ή κατηγορικά. Το σύνολο των δεδομένων, που χρησιμοποιείται για την εκπαίδευση ενός αλγορίθμου, ονομάζεται βάση δεδομένων (dataset). Η βάση δεδομένων, χωρίζεται σε τρία υποσύνολα, τα οποία δεν επικαλύπτονται. Αυτή η διαίρεση χρησιμοποιείται για να διασφαλιστεί, ότι ο αλγόριθμος δεν θα μάθει απλά τα δεδομένα της εκπαίδευσης αλλά θα μάθει να γενικεύει τις λύσεις του σε νέα δεδομένα. Η βάση δεδομένων, χωρίζεται στα ακόλουθα τρία υποσύνολα δεδομένων.

- **Το σύνολο εκπαίδευσης (train set)**, χρησιμοποιείται για την εκπαίδευση του μοντέλου και είναι το μεγαλύτερο υποσύνολο της βάσης δεδομένων.
- **Το σύνολο επικύρωσης (validation set)**, χρησιμοποιείται κατά τη διάρκεια της διαδικασίας εκπαίδευσης και στοχεύει στην καθορισμό των υπερπαραμέτρων του μοντέλου - επιπλέον παράμετροι, που επηρεάζουν τη διαδικασία εκπαίδευσης.
- **Το σύνολο ελέγχου (test set)**, χρησιμοποιείται για την αξιολόγηση του μοντέλου, παρουσιάζοντάς του νέα δείγματα, για τα οποία πρέπει να προβλέψει τα επιθυμητά αποτελέσματα.

Συνάρτηση Κόστος (Loss Function)

Σε ένα πρόβλημα μηχανικής μάθησης, για την εκπαίδευση ενός μοντέλου χρησιμοποιείται μια συνάρτηση, που συγκρίνει την πρόβλεψη του μοντέλου με την πραγματική τιμή του δείγματος. Αυτή η συνάρτηση ονομάζεται συνάρτηση κόστους (loss function) και ο στόχος της είναι να ελαχιστοποιηθεί, καθώς η έξοδός της δείχνει

το σφάλμα στην πρόβλεψη του μοντέλου. Η ελαχιστοποίηση της συνάρτησης κόστους, επιτυγχάνεται με την προσαρμογή των παραμέτρων του μοντέλου. Η επιλογή της συνάρτησης κόστους, εξαρτάται από τον τύπο του προβλήματος, που προσπαθεί να λύσει το μοντέλο. Ορισμένες από τις πιο διαδεδομένες συναρτήσεις κόστους, που χρησιμοποιούνται, παρουσιάζονται παρακάτω.

- **Η συνάρτηση μέσου τετραγωνικού σφάλματος (mean squared error - MSE)**
  χρησιμοποιείται σε προβλήματα παλινδρόμησης (regression), όπου η έξοδος του μοντέλου είναι μια συνεχής τιμή. Είναι μια μετρική, που δίνει βαρύτητα στην ακρίβεια των προβλέψεων του μοντέλου. Όσο πιο πολύ απέχει μια πρόβλεψη, τόσο μεγαλύτερο θα είναι το σφάλμα.
- **Η συνάρτηση μέσης απόλυτης τιμής (mean absolute error - MAE)**
  χρησιμοποιείται σε προβλήματα παλινδρόμησης, όπου η έξοδος του μοντέλου είναι μια συνεχής τιμή. Παρέχει μια μέτρηση της μέσης απόλυτης διαφοράς μεταξύ των προβλεπόμενων και των πραγματικών τιμών y. Το MAE είναι λιγότερο ευαίσθητο στα ακραία σημεία από το MSE, καθώς δεν περιλαμβάνει το τετράγωνο των σφαλμάτων.
- **Η συνάρτηση λογαριθμικού σφάλματος (log loss)**
  χρησιμοποιείται σε προβλήματα ταξινόμησης (classification), όπου η έξοδος του μοντέλου είναι δυαδική κατηγορική μεταβλητή.
- **Η συνάρτηση απώλειας εντροπίας (cross-entropy loss)**
  χρησιμοποιείται σε προβλήματα ταξινόμησης, όπου η έξοδος του μοντέλου είναι παραπάνω από δύο κατηγορίες.

Αλγόριθμοι Βελτιστοποίησης (Optimization Algorithms)

Για την επίλυση, συνεπώς, ενός προβλήματος μηχανικής μάθησης, αφού καθοριστεί το είδος του προβλήματος και επιλεχθεί μια κατάλληλη συνάρτηση κόστους, που θα ελαχιστοποιηθεί το κόστος της, σύμφωνα με τις προβλέψεις του μοντέλου και τα δείγματα, που παρέχονται από τη βάση δεδομένων, χρειάζεται να οριστεί και ένας αλγόριθμος βελτιστοποίησης. Ο αλγόριθμοι βελτιστοποίησης ρυθμίζει τις παραμέτρους του μοντέλου, ώστε να ελαχιστοποιηθεί η συνάρτηση κόστους, δηλαδή αλλάζει παλινδρομικά τις τιμές των παραμέτρων του μοντέλου, προς την κατεύθυνση της ελάχιστης τιμής της συνάρτησης κόστους. Οι αλγόριθμοι βελτιστοποίησης, που χρησιμοποιούνται για την εύρεση των βέλτιστων παραμέτρων του μοντέλου, είναι οι ακόλουθοι.

- **Οι Αλγόριθμοι Βελτιστοποίησης Πρώτου Βαθμού**
  χρειάζονται μόνο τις πρώτες παραγώγους των παραμέτρων του μοντέλου, για να καθορίσουν την κατεύθυνση της ενημέρωσης τους. Ο πιο διαδεδομένος αλγόριθμος βελτιστοποίησης πρώτου βαθμού είναι ο αλγόριθμος κατάβασης του κλίσης (gradient descent).
- **Οι Αλγόριθμοι Βελτιστοποίησης Δεύτερου Βαθμού**
  χρησιμοποιούν την πληροφορία της δεύτερης παραγώγου των παραμέτρων του μοντέλου, για να καθορίσουν την κατεύθυνση της ενημέρωσης τους. Συγκεκριμένα, χρησιμοποιούν τον πίνακα Hessian, που αντιπροσωπεύει τις δεύτερες παραγώγους της συνάρτησης κόστους ως προς τις παραμέτρους του μοντέλου.
- **Οι Αλγόριθμοι Βελτιστοποίησης Χωρίς Παραγώγους**
  χρησιμοποιούν την πληροφορία της συνάρτησης κόστους, χωρίς να απαιτούν την ύπαρξη παραγώγων της.

Ο πιο διαδεδομένος αλγόριθμος βελτιστοποίησης πρώτου βαθμού είναι ο αλγόριθμος κλίσης καθόδου (gradient descent). Ο αλγόριθμος αυτός, ενημερώνει τις παραμέτρους του μοντέλου, με την κατεύθυνση της αντίθετης της κλίσης της συνάρτησης κόστους, όπως περιγράφεται και από την ακόλουθη συνάρτηση

$$\theta^{(\tau)} = \theta^{(\tau-1)} - \eta \nabla_\theta L(f(x_i; \theta^{(\tau-1)}), y_i) \quad , \tag{1.2.1}$$

όπου το τ δηλώνει την χρονική στιγμή της επανάληψης, το $\theta^{(\tau)}$ δηλώνει το διάνυσμα των παραμέτρων, που ενημερώνεται κατά την επανάληψη τ, το η δηλώνει το ρυθμό μάθησης στον αλγόριθμο κλίσης καθόδου, το $\nabla_\theta L(f(x_i; \theta^{(\tau-1)}), y_i)$ δηλώνει την παράγωγο της συνάρτησης κόστους, ως προς τις παραμέτρους του μοντέλου, με τιμές $\theta^{(\tau-1)}$, που προκύπτουν από την προηγούμενη επανάληψη, το $f(x_i; \theta^{(\tau-1)})$ δηλώνει την πρόβλεψη του μοντέλου για το δείγμα x_i, με τιμές παραμέτρων $\theta^{(\tau-1)}$ και το y_i δηλώνει την πραγματική τιμή.

Μετρικές Αξιολόγησης (Evaluation Metrics)

Οι μετρικές αξιολόγησης είναι ποσοτικά μέτρα, που χρησιμοποιούνται για να αξιολογήσουν την απόδοση ενός μοντέλου σε ένα συγκεκριμένο πρόβλημα. Αυτές οι μετρικές βοηθούν στο να ποσοτικοποιηθεί πόσο καλά γενικεύει ένα μοντέλο σε άγνωστα δεδομένα ενός προβλήματος, στο οποίο εκπαιδεύτηκε να λύσει. Η επιλογή της συνάρτησης που χρησιμοποιείται για την αξιολόγηση εξαρτάται από τον τύπο του προβλήματος μηχανικής μάθησης (π.χ., ταξινόμηση, παλινδρόμηση) και τους στόχους του μοντέλου. Για παράδειγμα, μερικές από τις πιο διαδεδομένες μετρικές αξιολόγησης σε προβλήματα ταξινόμησης παρουσιάζονται παρακάτω.

- **Η Ποσοστιαία Επίδοση (Accuracy)**
  μετρά το ποσοστό των σωστών προβλέψεων σε σχέση με το σύνολο των προβλέψεων.
- **Η Ακρίβεια (Precision)**
  μετρά το ποσοστό των σωστών θετικών προβλέψεων σε σχέση με το σύνολο όλων των προβλέψεων, που κατανεμήθηκαν στην κατηγορία των θετικών προβλέψεων.
- **Η Ανάκληση (Recall)**
  μετρά το ποσοστό των σωστών θετικών προβλέψεων σε σχέση με το σύνολο όλων των πραγματικών θετικών παραδειγμάτων.
- **Το F1-score**
  συνδυάζει την ακρίβεια και την ανάκληση σε ένα μόνο μέτρο. Είναι η αρμονική μέση της ακρίβειας και της ανάκλησης, παρέχοντας μια ισορροπία μεταξύ αυτών των μετρικών. Το F1-score είναι ιδιαίτερα χρήσιμο σε καταστάσεις, όπου υπάρχει ανισορροπία μεταξύ των κλάσεων.

Η Εκπαίδευση Μοντέλων Μηχανικής Μάθησης

Οι αλγόριθμοι Μηχανικής Μάθησης σχεδιάζονται σύμφωνα με τα ακόλουθα βήματα.

1. **Η Συλλογή Δεδομένων**
   είναι η πρώτη διαδικασία, που απαιτείται για την εκπαίδευση ενός μοντέλου Μηχανικής Μάθησης. Η ποιότητα και η ποσότητα των δεδομένων επηρεάζουν σημαντικά την απόδοση του μοντέλου.
2. **Η Προεπεξεργασία Δεδομένων**
   περιλαμβάνει την καθαρισμό και την προετοιμασία των δεδομένων για την εκπαίδευση. Αυτό το βήμα περιλαμβάνει την αντιμετώπιση των απουσιάζουσων τιμών, των ακραίων τιμών και την μετατροπή των δεδομένων σε μια κατάλληλη μορφή για τον επιλεγμένο αλγόριθμο.
3. **Η Επιλογή Αλγορίθμου**
   είναι ένα κρίσιμο βήμα, καθώς ο αλγόριθμος που επιλέγεται επηρεάζει την απόδοση του μοντέλου. Οι αλγόριθμοι μπορούν να είναι διαφορετικοί, ανάλογα με τον τύπο του προβλήματος και τα χαρακτηριστικά των δεδομένων.
4. **Η Εκπαίδευση του Μοντέλου**
   περιλαμβάνει την προσαρμογή των παραμέτρων του μοντέλου στα δεδομένα εκπαίδευσης. Κατά τη διάρκεια της εκπαίδευσης, το μοντέλο μαθαίνει τα πρότυπα και τις σχέσεις μεταξύ των δεδομένων, προκειμένου να προβλέψει τις επιθυμητές τιμές.
5. **Η Αξιολόγηση του Μοντέλου**
   είναι το τελευταίο βήμα, που αξιολογεί την απόδοση του μοντέλου σε ένα σύνολο δεδομένων ελέγχου. Οι μετρικές αξιολόγησης χρησιμοποιούνται για να αξιολογήσουν την απόδοση του μοντέλου και να συγκρίνουν τις προβλέψεις του μοντέλου με τις πραγματικές τιμές.

1.2.3 Βαθιά Μάθηση

Η Βαθιά Μάθηση είναι ένα υποσύνολο της Μηχανικής Μάθησης, που χρησιμοποιεί μετασχηματισμούς και γράφους, προκειμένου να κατασκευάσει πολυεπίπεδα μοντέλα μάθησης, γνωστά ως βαθιά νευρωνικά δίκτυα (DNNs). Σε ένα τυπικό αλγόριθμο Μηχανικής Μάθησης, τα δεδομένα δίνονται σε μορφή διανυσμάτων μεταβλητών (ακέραιοι ή πραγματικοί αριθμοί). Έτσι, δεδομένα όπως ηχητικά σήματα ή εικόνες χρειάζονται κάποια προετοιμασία πριν δοθούν ως είσοδοι στο μοντέλο. Συγκεκριμένα, τα χαρακτηριστικά του ενδιαφέροντος θα πρέπει να εξαχθούν, προκειμένου να σχηματιστούν τα διανύσματα. Αυτή η προσέγγιση στοχεύει στην κατασκευή

χαρακτηριστικών από τα μη επεξεργασμένα δεδομένα, έξω από την αρχιτεκτονική του μοντέλου. Από την άλλη, οι αλγόριθμοι βαθιάς μάθησης επιτυγχάνουν την εξαγωγή χαρακτηριστικών με αυτόματο τρόπο μέσα στο μοντέλο τους. Αν και η Βαθιά Μάθηση χρειάζεται μεγάλο όγκο δεδομένων και υπολογιστικούς πόρους για την εκπαίδευση των μοντέλων, έχει επιδείξει εξαιρετική απόδοση σε πολλά προβλήματα, όπως η αναγνώριση φωνής, η αναγνώριση εικόνων και βίντεο.

Βαθιά Νευρωνικά Δίκτυα (DNNs)

Οι αλγόριθμοι της Βαθιάς Μάθησης, γνωστοί και ως Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANNs), αποτελούνται από πολλά επίπεδα οργανωμένων δομικών στοιχείων, που ονομάζονται νευρώνες. Ο νευρώνας ή αλλιώς perceptron, προτάθηκε από τον Rosenblatt [5]. Το perceptron λαμβάνει ένα ή περισσότερα σήματα εισόδου, τα οποία πολλαπλασιάζονται με τα βάρη τους και στη συνέχεια προστίθενται, για να προκύψει με αυτό τον τρόπο ένας γραμμικός συνδυασμός των εισόδων αυτών. Αυτός ο γραμμικός συνδυασμός των εισόδων περνά από μια συνάρτηση, που προσφέρει ένα μη γραμμικό μετασχηματισμό, για να παράγει την τελική έξοδο, όπως φάινεται και στο Σχήμα 1.2.1.

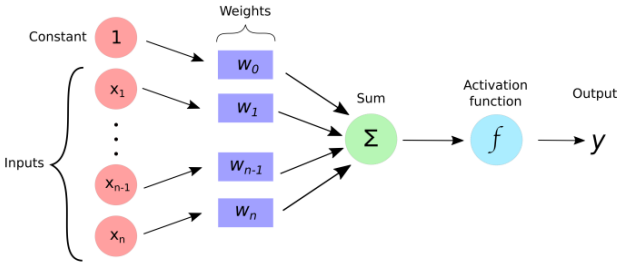


Figure 1.2.1: Νευρώνας ή Perceptron (από [6])

Τα επίπεδα, στα οποία οργανώνονται οι νευρώνες μπορεί να είναι τρία: το επίπεδο εισόδου, τα κρυφά επίπεδα και το επίπεδο εξόδου. Το επίπεδο εισόδου λαμβάνει τα δεδομένα εισόδου και τα μετατρέπει σε μορφή, που μπορεί να επεξεργαστεί το νευρωνικό δίκτυο. Τα κρυφά επίπεδα είναι εκείνα, που επεξεργάζονται τα δεδομένα και εξάγουν τα χαρακτηριστικά τους. Τέλος, το επίπεδο εξόδου παράγει την τελική έξοδο του μοντέλου.

Οι συναρτήσεις ενεργοποιήσεις (activation functions) χρησιμοποιούνται στη Βαθιά Μάθηση για να εισάγουν μη γραμμικότητα στο μοντέλο. έτσι ώστε δεδομένα, όπως εικόνες και ήχους, να μετατραπούν σε μορφή διαφορίσιμη ως προς τις παραμέτρους του μοντέλου [7].

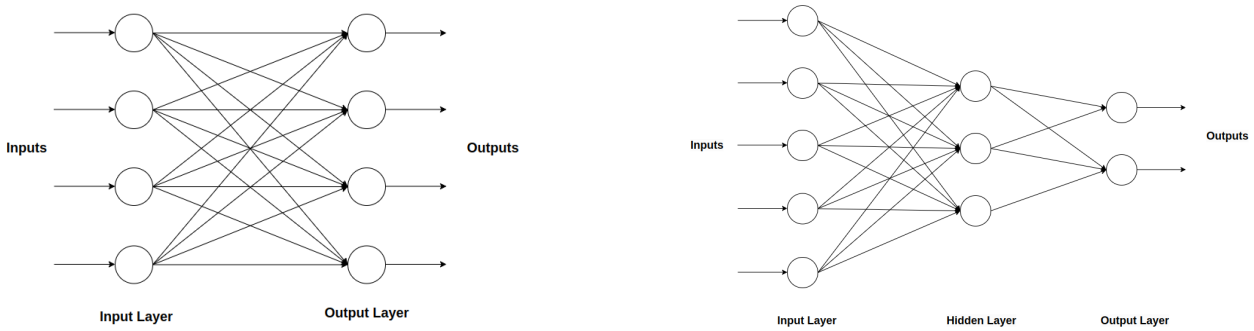
Κάποιες από τις πιο διαδεδομένες συναρτήσεις ενεργοποίησης [8] παρουσίαζονται ακολούθως.

- **Η Rectified Linear Unit (ReLU) συνάρτηση**
  χρησιμοποιείται για να αφαιρέσει τις αρνητικές τιμές, όπως αρνητικούς κλίσεις, όταν το κατώφλι είναι στο μηδέν.
- **Η Υπερβολική εφαπτομένη (tanh) συνάρτηση**
  χρησιμοποιείται για να μετατρέψει τις τιμές σε ένα εύρος μεταξύ -1 και $+1$. Αυτή η συνάρτηση ενεργοποίησης χρησιμοποιείται όταν η αρνητική κλίση είναι σημαντική.
- **Η Σιγμοειδής συνάρτηση**
  χρησιμοποιείται για να μετατρέψει τις τιμές σε ένα εύρος μεταξύ 0 και 1, κρατώντας με αυτό τον τρόπο την πληροφορία σε θετικές συνεχόμενες τιμές.
- **Η Softmax συνάρτηση**
  δημιουργεί μια διακριτή κατανομή πιθανοτήτων. Αυτή η συνάρτηση ενεργοποίησης χρησιμοποιείται όταν η έξοδος του μοντέλου χρειάζεται να είναι κατηγορική.

Τα Βαθιά Νευρωνικά Δίκτυα (DNNs) χωρίζονται σε τρεις κατηγορίες, ανάλογα με τον τρόπο με τον οποίο διαμοιράζεται η πληροφορία μέσα στα επίπεδα των νευρωνικών δικτύων: τα Νευρωνικά Δίκτυα Εμπρόσθιας Τροφοδότησης (Feedforward Neural Networks - FNNs), τα Συνελικτικά Δίκτυα (Convolutional Networks - CNNs) και τα Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNNs) [9].

Νευρωνικά Δίκτυα Εμπρόσθιας Τροφοδότησης (Feedforward Neural Networks - FNNs)

Τα Νευρωνικά Δίκτυα Εμπρόσθιας Τροφοδότησης (FNNs) είναι ένας τύπος νευρωνικών δικτύων, όπου η επεξεργασία των δεδομένων γίνεται μόνο προς τα εμπρός, από το επίπεδο εισόδου προς το επίπεδο εξόδου. Οι νευρώνες σε αυτό το είδος δικτύων οργανώνονται σε στρώματα και ανάλογα με τον αριθμό των στρωμάτων, τα FNNs χωρίζονται σε δύο κατηγορίες: τα Μονοεπίπεδα Νευρωνικά Δίκτυα (Single-Layer FNNs) και τα Πολυεπίπεδα Νευρωνικά Δίκτυα (Multi-Layer FNNs). Τα Μονοεπίπεδα Νευρωνικά Δίκτυα αποτελούνται από ένα επίπεδο εισόδου και ένα επίπεδο εξόδου, όπως φαίνεται στο Σχήμα 1.2.2a. Από την άλλη, τα Πολυεπίπεδα Νευρωνικά Δίκτυα αποτελούνται από περισσότερα από δύο επίπεδα, όπως φαίνεται στο Σχήμα 1.2.2b, με κάθε επίπεδο που να βρίσκεται ανάμεσα στο επίπεδο εισόδου και το επίπεδο εξόδου, να ονομάζεται κρυφό επίπεδο. Τα κρυφά επίπεδα χρησιμοποιούνται για την ανάλυση πιο πολύπλοκων αναπαραστάσεων των δεδομένων εισόδου.



(a) (b)

Figure 1.2.2: (a) Μονοεπίπεδα Νευρωνικά Δίκτυα. (b) Πολυεπίπεδα Νευρωνικά Δίκτυα (από [10])

Επιπλέον, τα FNNs μπορούν να χωριστούν σε άλλες δύο κατηγορίες, ανάλογα με τον τρόπο με τον οποίο οι νευρώνες συνδέονται μεταξύ τους: τα Πλήρως Συνδεδεμένα Νευρωνικά Δίκτυα (Fully-Connected Neural Networks), όπου όλοι οι νευρώνες σε ένα επίπεδο συνδέονται με όλους τους νευρώνες στο επόμενο επίπεδο, και τα Μερικώς Συνδεδεμένα Νευρωνικά Δίκτυα (Partially-Connected Neural Networks), όπου οι νευρώνες σε ένα επίπεδο συνδέονται με ένα υποσύνολο των νευρώνων στο επόμενο επίπεδο.

Συνελικτικά Νευρωνικά Δίκτυα (CNNs ή ConvNets)

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs ή ConvNets) είναι μια κατηγορία FNN, που είναι ειδικά σχεδιασμένα για την επεξεργασία δεδομένων πλέγματος, όπως εικόνες, καθώς έχουν τη δυνατότητα να εξάγουν χρήσιμα χαρακτηριστικά από αυτά τα δεδομένα, χωρίς να χρειάζεται κάποια προεπεξεργασία. Τα ConvNets αποτελούνται από μια σειρά επιπέδων, όπως φαίνεται στο Σχήμα 1.2.3, που είναι σε θέση να μετατρέψουν τα δεδομένα, επίπεδο με επίπεδο, σε κλάσεις για την ταξινόμηση ή σε άλλη χρήσιμη πληροφορία.

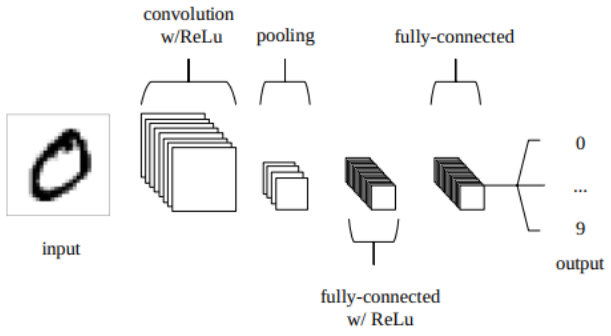


Figure 1.2.3: Μια απλή δομή Συνελικτικού Νευρωνικού Δικτύου αποτελούμενη από 5 επίπεδα (από [11])

Συνελκτικό Επίπεδο (Convolutional Layer) Το Συνελκτικό Επίπεδο (Convolutional Layer) χρησιμοποιείται για την εξαγωγή χαρακτηριστικών από τα δεδομένα. Το επίπεδο αυτό αποτελείται από ένα σύνολο φίλτρων, τα οποία είναι μικρότερα από την είσοδο και μετακινούνται πάνω σε αυτήν, πραγματοποιώντας συνελίξεις μεταξύ των φίλτρων και της εισόδου του επιπέδου. Το αποτέλεσμα είναι ένας χάρτης χαρακτηριστικών, που περιέχει χρήσιμη πληροφορία μαθηματικά αναπαριστώμενη για την είσοδο. Το Συνελκτικό Επίπεδο αποτελείται από τρία βασικά στοιχεία: την είσοδο, τα φίλτρα και το βήμα (stride), που καθορίζει την απόσταση μεταξύ των φίλτρων κατά την εφαρμογή τους στην είσοδο. Το Συνελκτικό Επίπεδο μπορεί να περιλαμβάνει και άλλες παραμέτρους, όπως το μέγεθος του φίλτρου, το μέγεθος του βήματος και η μέθοδος συμπλήρωσης (padding) της εισόδου, προκειμένου να διατηρηθεί η διάσταση της εισόδου στην έξοδο. Ανάλογα τις διαστάσεις των δεδομένων εισόδου, διαφορετικού τύπου φίλτρα μπορούν να χρησιμοποιηθούν, όπως δισδιάστατα (2D) ή τρισδιάστατα (3D) φίλτρα, για την εξαγωγή χωρικής ή χωροχρονικής πληροφορίας αντιστοίχως, όπως φαίνεται στο Σχήμα 1.2.4.

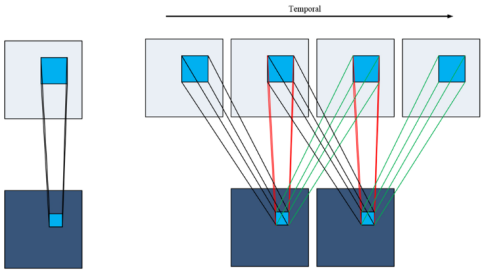


Figure 1.2.4: 2-D Συνελικτικό Επίπεδο (αριστερά). 3-D Συνελικτικό Επίπεδο (δεξιά). (από [12])

Για μια εικόνα με διαστάσεις (C, H, W), όπου C είναι ο αριθμός των καναλιών της εικόνας, H είναι το ύψος και W είναι το πλάτος, και ένα φίλτρο με διαστάσεις (F, F), όπου F είναι οι χωρικές διαστάσεις του φίλτρου, το αποτέλεσμα που παράγεται είναι η συνέλιξη του φίλτρου με την εικόνα εισόδου. Το φίλτρο ολισθαίνει πάνω στην είσοδο, και υπολογίζεται το εσωτερικό γινόμενο των βαρών του φίλτρου με τις χωρικές τιμές της εικόνας. Συνεπώς, δημιουργείται ένας χάρτης ενεργοποίησης, που αντιπροσωπεύει την ενεργοποίηση του φίλτρου σε κάθε σημείο της εικόνας. Στη συνέχεια, και άλλα φίλτρα εφαρμόζονται στην εικόνα, παράγοντας περισσότερους χάρτες ενεργοποίησης, οι οποίοι στο τέλος συνδυάζονται σε έναν τρισδιάστατο όγκο ενεργοποίησης, που αντιπροσωπεύει την εξόδο του συνελκτικού επιπέδου.

$$\begin{aligned} H_{out} &= \frac{H_{in} + 2 \cdot padding\_height - filter\_height}{stride\_height} + 1 \quad . \\ W_{out} &= \frac{W_{in} + 2 \cdot padding\_width - filter\_width}{stride\_width} + 1 \quad . \end{aligned} \tag{1.2.2}$$

Επομένως, η έξοδος θα έχει διαστάσεις (N, H_{out}, W_{out}), όπου το N είναι το πλήθος των φίλτρων, και H_{out}, W_{out} είναι οι χωρικές διαστάσεις, που υπολογίζονται σύμφωνα με την Εξίσωση 1.2.2.

Όσον αφορά ένα βίντεο με διαστάσεις (C, D, H, W), όπου το D αντιστοιχεί στο πλήθος των frames του βίντεο, κάθε κυβικό δείγμα του βίντεο συνελίσσεται με ένα τρισδιάστατο φίλτρο και το αποτέλεσμα προκύπτει από το άθροισμα των επιμέρους στοιχείων, όπως υποδεικνύεται από το Σχήμα 1.2.5.

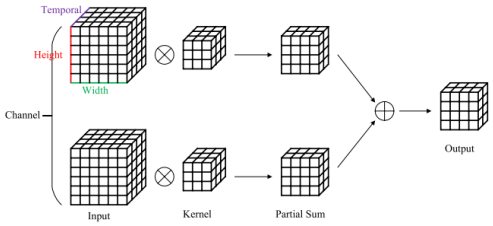


Figure 1.2.5: Εφαρμογή Συνελεκτικού Επιπέδου σε 3-D είσοδο (από [12])

Στη τρισδιάστατη συνέλιξη, λοιπόν, το φίλτρο ολισθαίνει κατά μήκος των frames του βίντεο, και όχι κατά μήκος των καναλιών της εικόνας, όπως στη δισδιάστατη συνέλιξη, κρατώντας με αυτό τον τρόπο την χρονική

πληροφορία. Η έξοδος θα έχει διαστάσεις $(N, D_{out}, H_{out}, W_{out})$, όπου το N ισούται με το πλήθος των φίλτρων και $D_{out}, H_{out}, W_{out}$ είναι οι χωροχρονικές διαστάσεις, που προκύπτουν σύμφωνα με την Εξίσωση 1.2.3

$$\begin{aligned} D_{out} &= \frac{D_{in} + 2 \cdot padding\_frames - filter\_frames}{stride\_frames} + 1 \quad , \\ H_{out} &= \frac{H_{in} + 2 \cdot padding\_height - filter\_height}{stride\_height} + 1 \quad , \\ W_{out} &= \frac{W_{in} + 2 \cdot padding\_width - filter\_width}{stride\_width} + 1 \quad , \end{aligned} \tag{1.2.3}$$

όπου το $filter\_size$ είναι οι διαστάσεις του κυβικού φίλτρου, το $stride$ είναι το βήμα με το οποίο ολισθαίνει το φίλτρο πάνω στο βίντεο και το $padding$ είναι επιπρόσθετες τιμές, συνήθως ίσες με το 0, κατά μήκος όλων των διαστάσεων της εισόδου για να κρατηθεί συγκεκριμένη επιθυμητή πληροφορία.

Επίπεδο Συγκέντρωσης (Pooling Layer) Σε ένα συνελκτικό νευρωνικό δίκτυο, ένα συνελκτικό επίπεδο συνήθως ακολουθείται από ένα επίπεδο συγκέντρωσης. Ένα επίπεδο συγκέντρωσης μετασχηματίζει την είσοδο με τέτοιο τρόπο ώστε να μειώσει τις διαστάσεις ενός χάρτη ενεργοποίησης, καθώς και τις παραμέτρους του δικτύου. Συγκεκριμένα, υποδειγματοληπτεί την είσοδο με τέτοιο τρόπο ώστε να αφαιρεθεί κάθε πλεονάζουσα πληροφορία. Για παράδειγμα σε ένα βίντεο με διαστάσεις $(C, D_{in}, H_{in}, W_{in})$, η έξοδος θα προκύψει σύμφωνα με την ακόλουθη Εξίσωση 1.2.4 :

$$\begin{aligned} D_{out} &= \frac{D_{in} - filter\_frames}{stride\_frames} + 1 \quad , \\ H_{out} &= \frac{H_{in} - filter\_height}{stride\_height} + 1 \quad , \\ W_{out} &= \frac{W_{in} - filter\_width}{stride\_width} + 1 \quad . \end{aligned} \tag{1.2.4}$$

Στην πράξη για μια μονοκαναλική εικόνα, δηλαδή ασπρόμαυρη (greyscale) με διαστάσεις $(1, 4, 4)$ και ένα επίπεδο συγκέντρωσης με πυρήνα 2×2 με $stride = 2$, ο πυρήνας θα εφαρμοστεί στις χρωματισμένες τετράδες, που φαίνονται στο Σχήμα 1.2.6, εξάγοντας μια τιμή για κάθε τετράδα, σύμφωνα με την τεχνική της συγκέντρωσης που χρησιμοποιείται. Οι πιο διαδεδομένες τεχνικές για επίπεδα συγκέντρωσης είναι η συγκέντρωση μέσου όρου (average pooling) και η συγκέντρωση μέγιστου όρου (max pooling), όπως περιγράφονται παρακάτω.

- **Η Συγκέντρωση Μέσου Όρου** υπολογίζει την μέση τιμή σε όλα τα στοιχεία του πυρήνα. Αυτή η μέθοδος λειτουργεί ως φίλτρο εξομάλυνσης, αλλά ταυτόχρονα θολώνει την είσοδο.
- **Η Συγκέντρωση Μέγιστου Όρου** υπολογίζει τη μέγιστη τιμή σε όλα τα στοιχεία του πυρήνα, τονίζοντας με αυτό το τρόπο τις πιο σημαντικές πληροφορίες.

(a) (b)

Figure 1.2.6: (a) Συγκέντρωση Μέσου Όρου. (b) Συγκέντρωση Μέγιστου Όρου. (από [13])

Τα επίπεδα συγκέντρωσης πέρα από την μείωση του υπολογιστικού κόστους του νευρωνικού δικτύου, χρησιμοποιούνται και για τον έλεγχο της υπερπροσαμογής (overfitting). Η υπερπροσαμογή είναι ένα πολύ συχνό πρόβλημα, που αντιμετωπίζουν τα μοντέλα βαθιάς μάθησης, κατά το οποίο ο αλγόριθμος ρυθμίζει τα βάρη του με τέτοιο τρόπο ώστε να μπορεί να δώσει σωστή απάντηση μόνο σε δείγματα του συνόλου εκπαίδευσης, χωρίς να μπορεί να γενικεύσει τις απαντήσεις του και σε άγνωστα δεδομένα.

Επίπεδο Κανονικοποίησης Πακέτων (Batch Normalization Layer) Όταν εκπαιδεύεται ένα βαθύ νευρωνικό μοντέλο, τα δείγματα των συνόλων δεδομένων (datasets) περνάνε από το μόντελο σε ομάδες, που καλούνται πακέτα ($mini - batches$). Με αυτό τον τρόπο το πακέτο αποτελεί μια μικρογραφία του συνόλου δεδομένων και ο υπολογισμός της παραγώγου του σφάλματος ως προς το πακέτο είναι πιο αντιπροσωπευτικός του συνόλου, με την ακρίβεια αυτής της αντιστοίχισης να αυξάνεται όσο αυξάνεται και το μέγεθος του πακέτου. Όμως, όταν η κατανομή των δειγμάτων δεν είναι ομοιόμορφη τότε αυτή η πρόσεγγιση μπορεί να οδηγήσει σε λύσεις τοπικού ελαχιστού και όχι ολικού. Λύση σε αυτό το πρόβλημα δίνει το επίπεδο κανονικοποίησης πακέτων [14], που κανονικοποιεί τα δεδομένα του πακέτου, κάνοντας τη μέση τιμή ίση με το 0 και την τυπική απόκλιση ίση με 1, όπως περιγράφεται από την ακόλουθη Εξίσωση 1.2.5.

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=0}^{m} x_i \quad , \\ \sigma &= \frac{1}{m}\sum_{i=0}^{m}(x_i-\mu)^2 \quad , \\ \hat{x_i} &= \frac{x_i-\mu}{\sqrt{\sigma^2+\epsilon}} \quad , \\ y_i &= \gamma x_i + \beta \quad , \end{aligned} \tag{1.2.5}$$

όπου x, y είναι η είσοδος και η έξοδος του επιπέδου κανονικοποίησης πακέτων για ένα συγκεκριμένο m mini-batch, μ είναι η μέση τιμή του mini-batch, σ είναι η τυπική απόκλιση του the mini-batch, β, γ είναι παράμετροι κλίμακας και μετατόπισης, που μαθαίνονται κατά την εκπαίδευση, και ϵ είναι μια μικρή σταθερά που προστίθεται στην τυπική απόκλιση για να αποφευχθεί η διαίρεση με το μηδέν [15].

Πλήρως Συνδεδεμένα Επίπεδα Τα τελευταία επίπεδα ενός νευρωνικού δικτύου Βαθιάς Μάθησης, που μετασχηματίζουν τους νευρώνες του προηγούμενου επιπέδου στο πλήθος των εξόδων του μοντέλου, καλούνται Πλήρως Συνδεδεμένα Επίπεδα. Αυτά τα επίπεδα δέχονται στην είσοδό τους μονοδιάστατους πίνακες και παράγουν στην έξοδό τους επίσης μονοδιάστατους πίνακες, όμως διαφορετικής διάστασης. Για αυτό το λόγω η έξοδος του προηγούμενου σταδίου πρέπει να μετατραπεί σε μια διάσταση. Αυτή η μετατροπή ονομάζεται flattening και οδηγεί στο πρώτο επίπεδο πλήρως συνδεδεμένων νευρώνων, όπως φαίνεται στο Σχήμα 1.2.7.

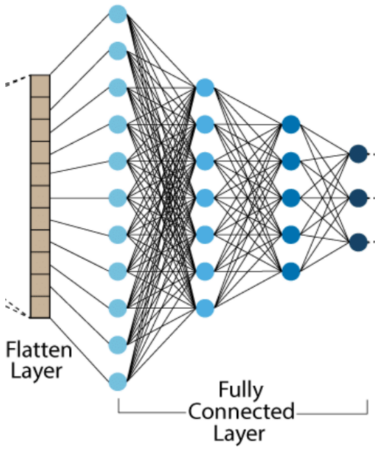


Figure 1.2.7: Πλήρως Συνδεδεμένα Επίπεδα (από [16])

Αναδρομικά Νευρωνικά Δίκτυα

Τα Νευρωνικά Δίκτυα στα οποία οι συνδέσεις μεταξύ των βαρών δημιουργούν βρόγχους ονομάζονται Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNNs). Αυτή η δομή των RNNs τους επιτρέπει να διατηρούν μνήμη των προηγούμενων εισόδων, κάνοντάς τα ικανά να χειρίζονται ακολουθίες δεδομένων, όπως σήματα ήχου, αναπαραστάσεις ακολουθιών λέξεων ή βίντεο, αναπαραστάσεις ακολουθιών εικόνων. Ωστόσο, απλές δομές RNNs

αντιμετωπίζουν το πρόβλημα της εξαφάνισης των παραγώγων, εννοόντας ότι οι τιμές των παραγώγων μπορεί να γίνουν πολύ μικρές κατά την εκπαίδευση του δικτύου, κάνοντας την εκμάθηση μακρών ακολουθιών δεδομένων προβληματική. Λύση σε αυτό το πρόβλημα προσφέρουν εναλλακτικές δομές αναδρομικών δικτύων, όπως τα μοντέλα Βραγχείας Μακράς Μνήμης (Long Short Term Memory - LSTM) και μοντέλα Μονάδων με Πύλες (Gated Recurrent Units - GRUs).

Απλή δομή Αναδρομικού Νευρωνικού Δικτύου (RNN) Μια απλή δομή RNN λαμβάνει ως είσοδο x_t μια ακολουθία δεδομένων και επιστρέφει μια ακολουθία κρυφών καταστάσεων h_t και εξόδων y_t. Η κρυφή κατάσταση h_t υπολογίζεται από την προηγούμενη κρυφή κατάσταση h_{t-1} και την τρέχουσα είσοδο x_t, ενώ η έξοδος y_t υπολογίζεται από την τρέχουσα κρυφή κατάσταση h_t. Η κρυφή κατάσταση h_t περνά στο επόμενο βήμα, όπου χρησιμοποιείται για τον υπολογισμό της επόμενης κρυφής κατάστασης h_{t+1}, και ούτως καθεξής. Με αυτό τον τρόπο τα RNNs κρατάνε την πληροφορία κατά την εκπαίδευση.

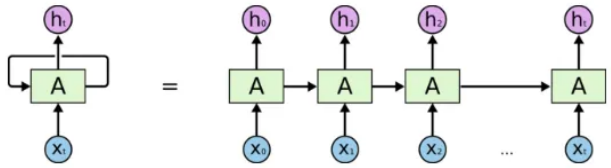


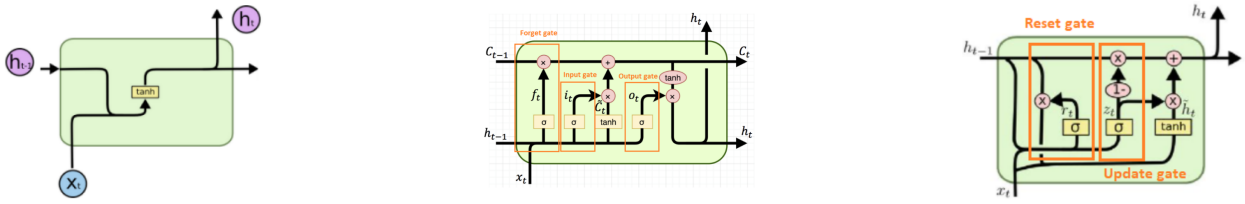
Figure 1.2.8: Απλή δομή Αναδρομικού Νευρωνικού Δικτύου ξεδιπλωμένη σε βάθος επαναλήψεων (από [17])

Από το Σχήμα 1.2.8 φαίνεται από τα δομικά στοιχεία των RNNs πως η είσοδος x_t περνά στην κρυφή κατάσταση h_t, η οποία περνά στην έξοδο y_t. Αυτός ο μηχανισμός μπορεί να διατυπωθεί με τις ακόλουθες εξισώσεις:

$$\begin{aligned} h_t &= tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \quad , \\ y_t &= W_{hy}h_t + b_y \quad , \end{aligned} \tag{1.2.6}$$

όπου το t δηλώνει την κατάσταση του μοντέλου, το W_{hh} δηλώνει το βάρος της προηγούμενης κρυφής κατάστασης, το W_{hx} δηλώνει το βάρος της τρέχουσας εισόδου, το W_{hy} δηλώνει το βάρος της έξοδου, και το $tanh(\cdot)$ είναι η συνάρτηση ενεργοποίησης, που εφαρμόζεται.

Μοντέλα Βραγχείας Μακράς Μνήμης (LSTM) Τα μοντέλα Βραγχείας Μακράς Μνήμης (LSTM) αποτελούν μια εξέλιξη των απλών RNNs, που προσθέτουν ένα δομικό στοιχείο, το οποία μπορεί να αποθηκεύσει δεδομένα για μεγαλύτερα χρονικά διαστήματα από τα απλά RNNs. Η δομή αυτή διαχειρίζεται από τρεις πύλες: την Πύλη Εισόδου, την Πύλη Λήθης και την Πύλη Εξόδου, όπως φαίνεται στο Σχήμα 1.2.9b.



(a) (b) (c)

Figure 1.2.9: (a) RNN δομικό στοιχείο. (b) LSTM δομικό στοιχείο. (c) GRU δομικό στοιχείο (από [18])

Η *Πύλη Εισόδου (Input Gate)* καθορίζει ποιες πληροφορίες θα εισέλθουν στο δομικό στοιχείο, για να τροποποιήσουν τη μνήμη. Συγκεκριμένα, η σιγμοειδής συνάρτηση αποφασίζει ποιες τιμές θα περάσουν, ενώ η συνάρτηση tanh δίνει βάρος στις τιμές που περνούν, αποφασίζοντας το επίπεδο σημασίας τους.

Η *Πύλη Λήθης (Forget Gate)* καθορίζει ποιες πληροφορίες θα ξεχαστούν από την μνήμη του δομικού στοιχείου και ποιες θα αποθηκευτούν. Η απόφαση λαμβάνεται από μια σιγμοειδή συνάρτηση, η οποία εξετάζει την προηγούμενη κατάσταση (h_{t-1}) και την είσοδο (x_t) και επιστρέφει έναν αριθμό μεταξύ 0 (παράλειψη) και 1 (διατήρηση) για κάθε αριθμό στην κατάσταση της μνήμης C_{t-1}.

Η *Πύλη Εξόδου (Output Gate)* καθορίζει ποιες πληροφορίες θα εξαχθούν από την μνήμη του δομικού στοιχείου και θα παραδοθούν στην έξοδο. Η απόφαση λαμβάνεται από μια σιγμοειδή συνάρτηση, η οποία αποφασίζει ποιες τιμές θα περάσουν, ενώ η συνάρτηση tanh καθοριζεί τη σημασία που θα δοθεί σε αυτή την πληροφορία.

Μοντέλα Μονάδων με Πύλες (GRUs) Τα μοντέλα Μονάδων με Πύλες (GRUs) αποτελούν μια εξέλιξη των μοντέλων LSTM, που προσφέρουν παρόμοια λειτουργικότητα με μικρότερο αριθμό παραμέτρων. Η δομή των GRUs, όπως φαίνεται στο Σχήμα 1.2.9c, αποτελείται από δύο πύλες: την Πύλη Ενημέρωσης και την Πύλη Επιλογής. Η Πύλη Ενημέρωσης αποφασίζει ποιες πληροφορίες θα διατηρηθούν στην κρυφή κατάσταση, ενώ η Πύλη Επιλογής αποφασίζει ποιες πληροφορίες θα περάσουν στην έξοδο. Με αυτό τον τρόπο τα GRUs μπορούν να διατηρήσουν μνήμη για μεγαλύτερα χρονικά διαστήματα από τα απλά RNNs, χωρίς να απαιτούν τόσο μεγάλο αριθμό παραμέτρων όπως τα LSTM [9].

Η Εκπαίδευση Βαθιών Μοντέλων Μάθησης

Σε ένα βαθύ μοντέλο μάθησης, οι παράμετροι του μοντέλου αναβαθμίζονται με παρόμοιο τρόπο με ένα μοντέλο Μηχανικής Μάθησης υπολογίζοντας την κλίση της συνάρτησης κόστους. Όπως αναφέρθηκε προηγουμένως, τα βαθιά μοντέλα μάθησης έχουν πολύ περισσότερες παραμέτρους από τα μοντέλα μηχανικής μάθησης, κάτι που καθιστά την εκπαίδευση τους πολύ πιο απαιτητική υπολογιστικά. Επομένως, η απλή εφαρμογή του αλγορίθμου κλίσης καθόδου (gradient descent) είναι ανεφάρμοστη για τα βαθιά μοντέλα μάθησης. Στην πράξη, η εκπαίδευση των βαθιών μοντέλων μάθησης γίνεται με τη χρήση πιο πολύπλοκων μεθόδων βελτιστοποίησης, οι οποίες αποτελούν μια παραλλαγή του απλού αλγορίθμου κλίσης καθόδου.

Στοχαστική Κλίση Καθόδου (Stochastic Gradient Descent - SGD)

Η Στοχαστική Κλίση Καθόδου (Stochastic Gradient Descent - SGD) είναι μια παραλλαγή του αλγορίθμου κλίσης καθόδου, που χρησιμοποιείται για την εκπαίδευση βαθιών μοντέλων μάθησης. Αυτή η παραλλαγή του αλγορίθμου κλίσης καθόδου αντί να χρησιμοποιεί ολόκληρο το σύνολο των δεδομένων για κάθε επανάληψη, επιλέγει τυχαία ένα υποσύνολο των δεδομένων, γνωστό ως πακέτο (batch), για τον υπολογισμό της κλίσης και την ενημέρωση των παραμέτρων του μοντέλου. Αυτή η τυχαιότητα που εισάγεται στη διαδικασία βελτιστοποίησης, εξηγεί τον όρο "στοχαστική" στον τίτλο του αλγορίθμου. Ορισμένα από τα πλεονεκτήματα της Στοχαστικής Κλίσης Καθόδου είναι η ταχύτητα και η αποδοτικότητά της. Από την άλλη, η SGD μπορεί να κάνει τη διαδικασία βελτιστοποίησης λιγότερο σταθερή και να οδηγήσει σε ταλαντώσεις γύρω από το ελάχιστο. Επίσης, μπορεί να χρειαστεί περισσότερες επαναλήψεις για να συγκλίνει στο ελάχιστο, καθώς ενημερώνει τις παραμέτρους για κάθε δείγμα εκπαίδευσης ξεχωριστά. Επιπλέον, η επιλογή του ρυθμού μάθησης μπορεί να είναι κρίσιμη στην SGD, καθώς η χρήση ενός υψηλού ρυθμού μάθησης μπορεί να οδηγήσει τον αλγόριθμο να υπερβεί το ελάχιστο, ενώ η χρήση ενός χαμηλού ρυθμού μάθησης μπορεί να καθυστερήσει τη σύγκλιση. Τέλος, λόγω των τυχαίων ενημερώσεων, η SGD μπορεί να μην συγκλίνει στο ακριβές ολικό ελάχιστο και να οδηγήσει σε μια υποβέλτιστη λύση, αν και αυτό μπορεί να αντιμετωπιστεί με τη χρήση τεχνικών, όπως η προγραμματισμένη ρύθμιση του ρυθμού μάθησης.

Adaptive Moment Estimation (Adam)

Ο αλγόριθμος Adam είναι ένας αλγόριθμος βελτιστοποίησης που μπορεί να χρησιμοποιηθεί αντί της στοχαστικής κλίσης καθόδου, καθώς η SGD διατηρεί ένα μόνο ρυθμό μάθησης για όλες τις ενημερώσεις βαρών και ο ρυθμός μάθησης δεν αλλάζει κατά τη διάρκεια της διαδικασίας εκπαίδευσης [19]. Συγκεκριμένα, ο Adam είναι ένας συνδυασμός δύο άλλων επεκτάσεων της στοχαστικής κλίσης καθόδου, της AdaGrad και της RMSProp, με σκοπό την εκμετάλλευση των οφελών των δύο αλγορίθμων, όπως περιγράφονται παρακάτω.

- **Ο Adaptive Gradient Algorithm (AdaGrad) αλγόριθμος** διατηρεί έναν ρυθμό μάθησης που προσαρμόζεται για κάθε παράμετρο του μοντέλου, βελτιώνοντας την απόδοση σε προβλήματα με αραιά δεδομένα.
- **Ο Root Mean Square Propagation (RMSProp) αλγόριθμος** διατηρεί ρυθμούς μάθησης που προσαρμόζονται για κάθε παράμετρο με βάση τον μέσο όρο των τελευταίων βαθμίδων των βαρών του μοντέλου του μοντέλου.

Αντίστροφη Διάδοση (Backpropagation)

Στην Βαθιά Μάθηση, η ουσία της διαδικασίας εκπαίδευσης είναι η βελτιστοποίηση των παραμέτρων του μοντέλου, ώστε η συνάρτηση κόστους να ελαχιστοποιηθεί. Αυτός ο στόχος επιτυγχάνεται με τη χρήση ενός αλγορίθμου βελτιστοποίησης, όπως ο SGD ή ο Adam, για την ενημέρωση των βαρών του μοντέλου με επαναλαμβανόμενο τρόπο. Για να ενημερωθούν τα βάρη, πρέπει να υπολογιστεί η κλίση της συνάρτησης κόστους ως προς τις παραμέτρους του μοντέλου, δηλαδή οι παράγωγοί της. Αυτό γίνεται με τη χρήση του αλγορίθμου αντίστροφης διάδοσης (backpropagation), ο οποίος αποτελείται από δύο στάδια, το στάδια της εμπρόσθιας διάδοσης και στάδιο της αντίστροφης διάδοσης πληροφορίας.

- **Στην Εμπρόσθια Διάδοση (Forward Pass)**
  η πληροφορία διαδίδεται μέσα στο δίκτυο με κατεύθυνση από το επίπεδο εισόδου προς το επίπεδο εξόδου. Κατά τη διάρκεια αυτής της διαδικασίας, υπολογίζεται η έξοδος κάθε επιπέδου, με βάση τα τρέχοντα βάρη του μοντέλου. Η έξοδος του τελευταίου επιπέδου είναι η έξοδος του δικτύου.
- **Στην Αντίστροφη Διάδοση (Backward Pass)**
  η πληροφορία ρέει μέσα στο δίκτυο με κατεύθυνση από το επίπεδο εξόδου προς το επίπεδο εισόδου. Κατά τη διάρκεια αυτής της διαδικασίας, υπολογίζεται η παράγωγος της συνάρτησης κόστους ως προς τα βάρη του μοντέλου. Με αυτή τη παράγωγο, όταν χρησιμοποιηθούν οι μαθηματικοί τύποι, που περιγράφουν κάθε επίπεδο του δικτύου, και με χρήση του κανόνα της αλυσίδας (chain rule), δημιουργούνται μαθηματικοί τύποι για την ενημέρωση των βαρών όλου του δικτύου.

1.3 Σχετική Βιβλιογραφία

1.3.1 Αναγνώριση Ανθρώπινων Δράσεων και Χειρονομιών

Αναγνώριση Ανθρώπινων Δράσεων

Η αναγνώριση των ανθρώπινων δράσεων είναι ένα πρόβλημα που απασχολεί εντατικά την ερευνητική κοινότητα τα τελευταία χρόνια. Μια δράση αναφέρεται σε μια συγκεκριμένη κίνηση που εκτελεί ένα υποκείμενο, συνήθως ένα άτομο, σε ένα βίντεο. Για παράδειγμα, δράσεις μπορεί να είναι το περπάτημα, το άλμα, το διάβασμα, το να παίζει κάποιο μουσικό όργανο κ.λπ. Κάθε μια από αυτές τις δράσεις περιλαμβάνει ένα συγκεκριμένο πρότυπο κίνησης με την πάροδο του χρόνου, το οποίο μπορεί να αναγνωριστεί με την ανάλυση της ακολουθίας των εικόνων του βίντεο.

Όπως και στην όραση υπολογιστών γενικότερα, η αναγνώριση δράσεων είναι χρήσιμη σε ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένων της επιτήρησης βίντεο και της αλληλεπίδρασης ανθρώπου-υπολογιστή. Η αναγνώριση δράσεων είναι ιδιαίτερα σημαντική για την κατανόηση της ανθρώπινης συμπεριφοράς σε βίντεο, και για την ενεργοποίηση των μηχανών να κατανοούν και να ανταποκρίνονται στις ανθρώπινες δράσεις.

Αναγνώριση Ανθρώπινων Χειρονομιών

Η αναγνώριση χειρονομιών είναι ένα ακόμα σημαντικό πρόβλημα στον τομέα της όρασης υπολογιστών. Οι χειρονομίες ορίζονται ως μια ειδική κατηγορία κινήσεων, που μπορούν να θεωρηθούν ως μια μορφή μη-γλωσσικής επικοινωνίας στην οποία ορατές σωματικές ενέργειες χρησιμοποιούνται για να επικοινωνήσουν συγκεκριμένα μηνύματα. Οι χειρονομίες μπορούν να χρησιμοποιηθούν για να μεταδώσουν ένα ευρύ φάσμα πληροφοριών, συμπεριλαμβανομένων συναισθημάτων, προθέσεων και εντολών. Στο πλαίσιο της όρασης υπολογιστών, η αναγνώριση χειρονομιών αναφέρεται στο πρόβλημα της αναγνώρισης και ερμηνείας των ανθρώπινων χειρονομιών από οπτικά δεδομένα. Για παράδειγμα, οι χειρονομίες μπορεί να περιλαμβάνουν κινήσεις των χεριών, προσωπικές εκφράσεις, στάσεις σώματος και άλλα μη-γλωσσικά σήματα. Αυτή η εργασία επικεντρώνεται στην αναγνώριση κινήσεων χεριών στα βίντεο. Παραδείγματα κινήσεων χεριών σε βίντεο μπορεί να περιλαμβάνουν το να χαιρετά, το να δείχνει, το χειροκρότημα και άλλες κινήσεις των χεριών.

Η αναγνώριση χειρονομιών έχει πολλές εφαρμογές, συμπεριλαμβανομένης της αλληλεπίδρασης ανθρώπου-υπολογιστή, της αναγνώρισης νοηματικής γλώσσας και της εικονικής πραγματικότητας.

Πρωταρχικές Μέθοδοι Επίλυσης Προβλημάτων Αναγνώρισης Δράσεων και Χειρονομιών

Οι μέθοδοι, που χρησιμοποιούνταν στα πρώιμα στάδια της όρασης υπολογιστών, ακολουθούσαν την παρακάτω διαδικασία για την επίλυση προβλημάτων, όπως η αναγνώριση δράσεων και χειρονομιών: αρχικά, έπρεπε να εντοπιστούν τα χαρακτηριστικά του ενδιαφέροντος στα οπτικά δεδομένα, στη συνέχεια αυτά τα χαρακτηριστικά αυτά εξάγονταν, μετά δημιουργούνταν ένας χάρτης μεταξύ των χαρακτηριστικών και των αναπαραστάσεων τους και τέλος χρησιμοποιούνταν ένας ταξινομητής για τη διάκριση τους μεταξύ διαφορετικών κλάσεων [20], όπως φαίνεται στο Σχήμα 1.3.1.

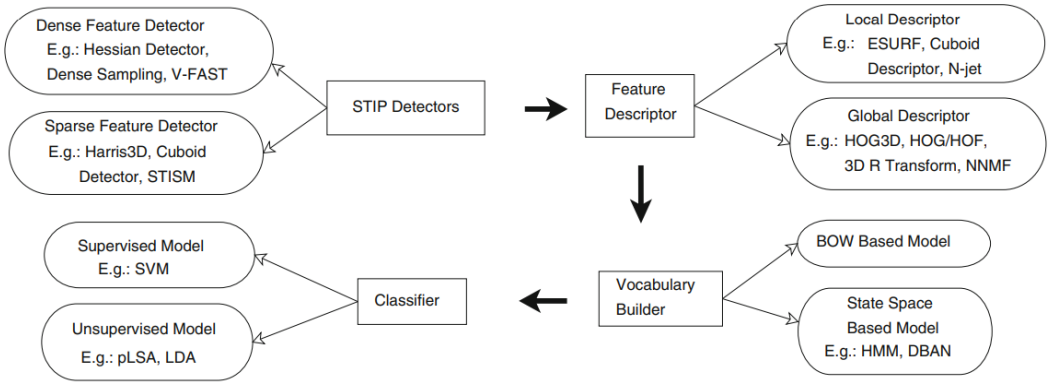


Figure 1.3.1: Διάγραμμα Μεθόδων Επίλυσης Προβλημάτων Αναγνώρισης Δράσεων και Χειρονομιών (από [20])

Ένα χαρακτηριστικό ενδιαφέροντος είναι μια περιοχή της εικόνας, που περιέχει πληροφορίες για το αντικείμενο ή τη δράση, που επιθυμούμε να αναγνωρίσουμε. Τα χαρακτηριστικά μπορούν να είναι γωνίες, σημεία έναρξης ή τερματισμού γραμμών, σημεία με μέγιστη ή ελάχιστη ένταση και άλλα. Για τα βίντεο, τα χαρακτηριστικά είναι επίσης γνωστά ως σημεία ενδιαφέροντος χωροχρονικών περιοχών (spatio-temporal interest points - STIP).

Για την ανίχνευση χαρακτηριστικών, χρησιμοποιούνται αλγόριθμοι STIP, που μπορούν να κατηγοριοποιηθούν σε δύο είδη, πυκνούς και αραιούς.

- **Οι Πυκνοί Ανιχνευτές Χαρακτηριστικών**
  εντοπίζουν χαρακτηριστικά σε μια εικόνα σε κάθε pixel ή σε έναν πυκνό πλέγμα αυτής. Με αυτό τον τρόπο εξασφαλίζεται ότι τα χαρακτηριστικά ανιχνεύονται ομοιόμορφα σε όλη την εικόνα, παρέχοντας μια περιεκτική αναπαράσταση του περιεχομένου που απεικονίζεται. Ένας από τους πιο δημοφιλείς πυκνούς ανιχνευτές χαρακτηριστικών είναι ο Hessian Ανιχνευτής [21], που βασίζεται στον πίνακα Hessian, ο οποίος είναι ένα τετραγωνικός πίνακας δεύτερης τάξης μερικών παραγώγων της συνάρτησης έντασης της εικόνας, παρέχοντας πληροφορίες για την καμπυλότητα της επιφάνειας έντασης της εικόνας, κάνοντας τον χρήσιμο για την ανίχνευση περιοχών με σημαντικές μεταβολές έντασης όπως blobs.

- **Οι Αραιοί Ανιχνευτές Χαρακτηριστικών**
  εντοπίζουν χαρακτηριστικά σε μια εικόνα σε συγκεκριμένες περιοχές. Αυτό το είδος ανιχνευτών χρησιμοποιείται όταν η εξαγωγή χαρακτηριστικών από όλη την εικόνα είναι υπερβολικά ακριβή και χρονοβόρα. Ένας από τους πιο δημοφιλής αραιούς ανιχνευτές χαρακτηριστικών είναι ο Harris Ανιχνευτής [22], όπου χρησιμοποιείται για την ανίχνευση γωνιών σε μια εικόνα, καθώς και η επέκτασή του σε βίντεο, ο Harris3D [23]. Οι γωνίες είναι σημεία σε μια εικόνα όπου η ένταση αλλάζει σημαντικά σε πολλές κατευθύνσεις. Έτσι, ο πίνακας κλίσης της συνάρτησης έντασης της εικόνας υπολογίζεται στις κατευθύνσεις x και y, από τον οποίο μπορεί να προκύψει η πιθανότητα ένα pixel να είναι μέρος μιας γωνίας.

Για την εξαγωγή χαρακτηριστικών, χρησιμοποιούνται περιγραφητές χαρακτηριστικών, οι οποίοι αναπαριστούν τα χαρακτηριστικά που εντοπίστηκαν στα οπτικά δεδομένα. Οι περιγραφητές χαρακτηριστικών μπορούν να χωριστούν σε δύο κατηγορίες, τους τοπικούς και τους γενικούς.

- **Οι Τοπικοί Περιγραφητές**
  αναπαριστούν μόνο τοπικές ή στατικές πληροφορίες, για παράδειγμα το χρώμα ή την υφή της εικόνας. Το σχήμα και οι ακμές των αντικειμένων χρησιμοποιούνται για την καθορισμό των τοπικών χαρακτηριστικών. Ορισμένοι από τους πιο δημοφιλείς τοπικούς περιγραφητές είναι ο Περιγραφητής Μετασχηματισμού Χαρακτηριστικών Ανεξάρτητος Κλίμακας (Scale Invariant Feature Transform - SIFT) [24] και ο Περιγραφητής Επιταγχυμένων Ανθεκτικών Χαρακτηριστικών (Speeded Up Robust Features - SURF) [25].

  1. Ο SIFT περιγραφητής μετασχηματίζει μια εικόνα σε μια συλλογή τοπικών διανυσματικών χαρακτηριστικών, τα οποία δεν επηρεάζονται από την κλιμάκωση ή την περιστροφή της, καθώς βασίζεται στην κατανομή της κλίσης της εικόνας σε τοπικές γειτονιές. Αυτά τα χαρακτηριστικά χρησιμοποιούνται για την αναγνώριση αντικειμένων σε διαφορετικές προβολές μιας σκηνής. Το κύριο πλεονέκτημα του SIFT είναι ότι είναι ανθεκτικός σε παραμορφώσεις, προσθήκη θορύβου και αλλαγές στον φωτισμό.

  2. Ο SURF περιγραφητής είναι μια εξέλιξη του SIFT, αλλά είναι ταχύτερος και πιο ανθεκτικός στις μετασχηματισμούς της εικόνας. Συγκεκριμένα, ο SURF χρησιμοποιεί ένα διαφορετικό τρόπο υπολογισμού των κλίσεων, καθώς επιλύει μια προσεγγιστική λύση του προβλήματος της κλίσης με τη χρήση του διακριτού μετασχηματισμού της εικόνας.

- **Οι Γενικοί Περιγραφητές**
  αναπαριστούν τη δυναμική πληροφορία των οπτικών δεδομένων, όπως αλλαγές κλίμακας, αλλαγές φωτισμού, μετατόπιση και περιστροφή. Κάποιοι από τους πιο δημοφιλείς γενικούς περιγραφητές είναι τα Ιστογράμματα Προσανατολισμένης Κλίσης (Histogram of Oriented Gradients - HOG) [26] και τα Ιστογράμματα Οπτικής Ροής (Histogram of Optical Flow - HOF) [27].

  1. Ο HOG περιγραφητής χρησιμοποιείται για την αναπαράσταση της δομής και της εμφάνισης των αντικειμένων σε μια εικόνα. Αυτός ο περιγραφητής χρησιμοποιεί την κατανομή των κλίσεων των γραμμών της εικόνας για την αναπαράσταση των αντικειμένων.

  2. Ο HOF περιγραφητής χρησιμοποιείται για την αναπαράσταση των κινητικών μοτίβων σε ένα βίντεο. Αυτός ο περιγραφητής χρησιμοποιεί την κατανομή των κατευθύνσεων και των μεγεθών της οπτικής

ροής για την αναπαράσταση των κινητικών μοτίβων.

Στη συνέχεια τα χαρακτηριστικά πρέπει να αντιστοιχηθούν με τις αναπαράστασεις τους. Αυτό μπορεί να γίνει είτε με τη χρήση ενός Μοντέλου Σάκου Λέξεων (Bag of Words - BoW) [28] είτε με τη χρήση ενός Μοντέλου Χώρου Καταστάσεων.

- **Ο Σάκος Λέξεων (Bag of Words - BoW)**
  είναι μια μέθοδος κωδικοποίησης χαρακτηριστικών που βασίζεται στην επεξεργασία φυσικής γλώσσας. Στο πλαίσιο της όρασης υπολογιστών, ο BoW αναφέρεται συχνά ως Ο Σάκος Οπτικών Λέξεων (Bag of Visual Words - BoVW). Αυτή η μέθοδος κωδικοποιεί μια εικόνα αντικαθιστώντας κάθε τοπικό χαρακτηριστικό με την πλησιέστερη οπτική λέξη από έναν προκαθορισμένο λεξιλόγιο. Η εικόνα αναπαρίσταται στη συνέχεια ως ιστόγραμμα της συχνότητας εμφανίσης των οπτικών λέξεων, για την οπτικοποίηση της κατανομής των χαρακτηριστικών.
- **Τα Μοντέλα Χώρου Καταστάσεων**
  είναι δυναμικά συστήματα, που αποτελούνται από μια σειρά καταστάσεων, που εξελίσσονται με τον χρόνο. Ένα ευρέως διαδεδομένο μοντέλο χώρου καταστάσεων είναι τα Κρυφά Μαρκοβιανά Μοντέλα (Hidden Markov Model - HMM) [29]. Ένα HMM είναι ένα στατιστικό μοντέλο που χρησιμοποιείται για την περιγραφή δυναμικών συστημάτων, όπου η κατάσταση του συστήματος εξελίσσεται με τον χρόνο και παρατηρείται μέσω μετρήσεων.

Τέλος, χρησιμοποιείται ένας ταξινομητής για τη διάκριση μεταξύ διαφορετικών κλάσεων. Ανάλογα με το πρόβλημα, γίνεται χρήση είτε μεθόδων επιβλεπόμενης μάθησης είτε μεθόδων μη - επιβλεπόμενης μάθησης.

- **Στην Επιβλεπόμενη Μάθηση**
  οι αλγόριθμοι εκπαιδεύονται στο σύνολο των χαρακτηριστικών που εξάγονται από τα προηγούμενα στάδια, για την κατανομή τους σε διαφορετικές κλάσεις. Ένας από τους πιο δημοφιλείς ταξινομητές για αυτό το σκοπό είναι ο Μηχανισμός Διανυσμάτων Υποστήριξης (Support Vector Machines - SVM) [30]. Το SVM κατασκευάζει ένα υπερεπίπεδο ή ένα σύνολο υπερεπιπέδων σε έναν χώρο υψηλής διάστασης για τη διάκριση διαφορετικών κλάσεων δεδομένων. Στόχος του SVM είναι να βρει το βέλτιστο υπερεπίπεδο που μεγιστοποιεί το περιθώριο μεταξύ των πιο κοντινών σημείων των κλάσεων, γνωστά ως διανύσματα υποστήριξης.
- **Στη Μη Επιβλεπόμενη Μάθηση**
  αλγόριθμοι όπως ο Γραμμικός Αναλυτής Διακριτών Συνιστωσών (Linear Discriminant Analysis - LDA) [31] και ο Αναλυτής Κύριων Συνιστωσών (Principal Component Analysis - PCA) [32] χρησιμοποιούνται για τη μείωση της διαστατικότητας του χώρου χαρακτηριστικών και για την εύρεση των πιο σημαντικών χαρακτηριστικών που μπορούν να χρησιμοποιηθούν για τη διάκριση μεταξύ των κλάσεων.

Στην πορεία αυτή η μέθοδος επίλυσης προβλημάτων όρασης υπολογιστών αντικαταστήθηκε από την Βαθιά Μάθηση, η οποία μπορεί να λάβει ως είσοδο δεδομένα, όπως εικόνες και βίντεο, και να εξάγει χαρακτηριστικά λόγω της πολυεπίπεδης δομής της. Συγκεκριμένα, όπως φαίνεται στο Σχήμα 1.3.2, τα πρώτα επίπεδα ενός CNN εξάγουν χαμηλού επιπέδου χαρακτηριστικά, όπως ακμές, γωνίες και blobs, ενώ τα μεσαία επίπεδα εξάγουν χαρακτηριστικά μεσαίου επιπέδου, όπως μάτια, μύτες και στόματα και τα τελευταία επίπεδα εξάγουν χαρακτηριστικά υψηλού επιπέδου, όπως δομές προσώπου.

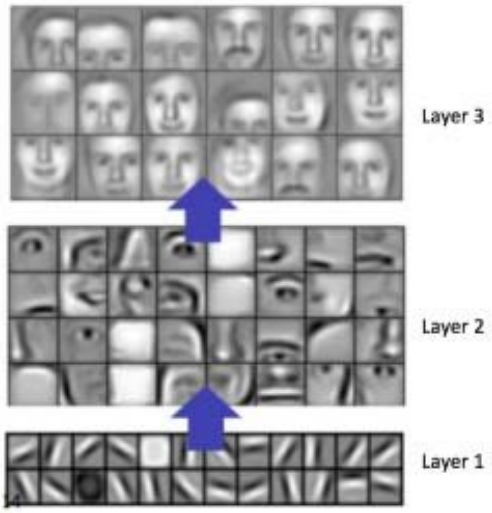


Figure 1.3.2: Τύποι Χαρακτηριστικών που εξάγονται από Διαφορετικά Επίπεδα ενός Συνελικτικού Νευρωνικού Δικτύου (από [33])

1.3.2 Εφαρμοσμένες Αρχιτεκτονικές Νευρωνικών Δικτύων

Τα μοντέλα Βαθιάς Μάθησης αντικατέστησαν τους εξωτερικούς περιγραφητές χαρακτηριστικών και χρησιμοποιήθηκαν ευρέως για την εξαγωγή χαρακτηριστικών από εικόνες, βίντεο και άλλους τύπους δεδομένων. Η τυπική δομή ενός βαθιού νευρωνικού δικτύου, που χρησιμοποιείται για την εξαγωγή χαρακτηριστικών από τα δεδομένα, ονομάζεται βασικό μοντέλο (backbone). Το backbone είναι συνήθως μια αρχιτεκτονική Συνελικτικού Νευρωνικού Δικτύου (CNN), η οποία μπορεί να τροποποιηθεί για να προσαρμοστεί στις ανάγκες της εργασίας ή ώστε να προταθούν νέες μεθοδολογίες. Η αναγνώριση δράσεων και χειρονομιών είναι δύο διαφορετικές εργασίες, που ανήκουν και οι δύο στην κατηγορία της ταξινόμησης βίντεο στην όραση υπολογιστών. Έτσι, τα βασικά νευρωνικά δίκτυα που χρησιμοποιούνται για κάθε εργασία είναι παρόμοια, αν και έχουν προταθεί αρκετές αρχιτεκτονικές για καλύτερη απόδοση σε κάθε εργασία. Συγκεκριμένα, στην αναγνώριση δράσεων οι αρχιτεκτονικές μπορούν να επιτύχουν καλή απόδοση εκμεταλλευόμενες τις πληροφορίες που παρέχονται από τα RGB καρέ, ακόμα και με δείγματα βίντεο χαμηλής ποιότητας, ενώ στην αναγνώριση χειρονομιών τα μοντέλα συχνά απαιτούν δεδομένα υψηλής ανάλυσης, καθώς και επιπλέον πληροφορίες που παρέχονται από δεδομένα βάθους ή σκελετού, λόγω της ομοιότητας μεταξύ των κλάσεων χειρονομιών. Σε αυτήν την ενότητα παρουσιάζονται μερικά από τα πιο δημοφιλή βασικά νευρωνικά δίκτυα που χρησιμοποιούνται για τις εργασίες ταξινόμησης βίντεο.

Τρισδιάτα Συνελικτικά Δίκτυα (3D Convolutional Network - C3D)

Τα τρισδιάτα συνελικτικά δίκτυα είναι μια αρχιτεκτονική που βασίζεται στην ιδέα ότι τα τρισδιάτα συνελικτικά επίπεδα μπορούν να χρησιμοποιηθούν για την εξαγωγή χωροχρονικών χαρακτηριστικών από βίντεο. Το τρισδιάτο συνελικτικό δίκτυο, ή C3D, προτάθηκε από τον Tran και άλλους στο [34], ως εναλλακτική λύση στα δισδιάτα συνελικτικά δίκτυα. Στα μοντέλα C3D, οι λειτουργίες συνέλιξης και συγκέντρωσης εκτελούνται χωροχρονικά, ενώ στα δισδιάτα συνελικτικά δίκτυα εκτελούνται μόνο χωρικά. Συγκεκριμένα, η συνέλιξη 2D που εφαρμόζεται σε μια εικόνα θα δώσει ως έξοδο μια εικόνα, η συνέλιξη 2D που εφαρμόζεται σε πολλές εικόνες θα δώσει επίσης μια εικόνα. Έτσι, τα δισδιάτα συνελικτικά δίκτυα χάνουν τη χρονική πληροφορία του εισόδου αμέσως μετά από κάθε λειτουργία συνέλιξης. Μόνο η τρισδιάτη συνέλιξη διατηρεί τη χρονική πληροφορία των εισόδων, δίνοντας ως έξοδο έναν όγκο, φαινόμενο που εφαρμόζεται επίσης στα pooling επίπεδα.

Στο Σχήμα 1.3.3 παρουσιάζεται η αρχιτεκτονική του C3D. Το C3D αποτελείται από 8 συνελικτικά επίπεδα, 5 επίπεδα συγκέντρωσης, 2 πλήρως συνδεδεμένα επίπεδα και ένα επίπεδο softmax loss για την πρόβλεψη των ετικετών δράσεων. Όλα τα τρισδιάτα συνελικτικά πυρήνες έχουν μέγεθος $3 \times 3 \times 3$ με βήμα 1 τόσο στις χωρικές όσο και στις χρονικές διαστάσεις. Όλα τα επίπεδα συγκέντρωσης έχουν μέγεθος $2 \times 2 \times 2$ με βήμα $2 \times 2 \times 2$, εκτός από το πρώτο που έχει μέγεθος πυρήνα $1 \times 2 \times 2$ και βήμα $1 \times 2 \times 2$ με την πρόθεση να διατηρήσει τη χρονική πληροφορία στην αρχή. Κάθε πλήρως συνδεδεμένο επίπεδο έχει 4096 μονάδες εξόδου.

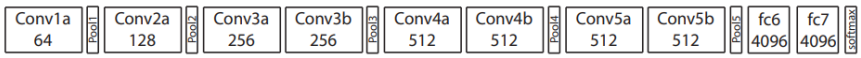


Figure 1.3.3: C3D αρχιτεκτονική (από [34])

Αυτό το συνελικτικό δίκτυο εφαρμόστηκε σε εργασίες τόσο αναγνώρισης δράσεων όσο και χειρονομιών, με επιτυχία. Συγκεκριμένα, στο [34] οι συγγραφείς εφάρμοσαν το C3D στο πρόβλημα της αναγνώρισης δράσεων και πέτυχαν αποτελέσματα που ξεπέρασαν τα τότε state-of-the-art αποτελέσματα σε δύο διαφορετικές βάσεις δεδομένων, το Sports-1M [35] και το UCF-101 [36]. Στο [37] οι συγγραφείς εφάρμοσαν το C3D στο πρόβλημα της αναγνώρισης χειρονομιών όπου και απέδειξαν την εφαρμοσιμότητα του μοντέλου και σε αυτό το πρόβλημα.

Δύο Ροών Δίκτυα (Two Stream Network - TSN)

Μιας και ένα βίντεο είνα μια ακολουθία από εικόνες, τότε μπορεί να αποδομηθεί στα χωρικά και χρονικά στοιχεία του. Το χωρικό μέρος, σε μορφή εμφάνισης των εικόνων, μεταφέρει πληροφορίες για τις σκηνές και τα αντικείμενα που απεικονίζονται στο βίντεο. Το χρονικό μέρος, σε μορφή κίνησης μεταξύ των εικόνων, μεταφέρει την κίνηση του παρατηρητή, της κάμερας και των αντικειμένων. Ακολουθώντας αυτήν την παρατήρηση, οι Simonyan και Zisserman στο [38] πρότειναν την αρχιτεκτονική των Δικτύων Δύο Ροών (TSN), η οποία αποτελείται από δύο ξεχωριστά δίκτυα, ένα για τις χωρικές και ένα για τις χρονικές πληροφορίες, όπως φαίνεται στο Σχήμα 1.3.4. Κάθε ροή υλοποιείται χρησιμοποιώντας ένα βαθύ ConvNet, ακολουθούμενο από μια συνάρτηση softmax, τα αποτελέσματα των οποίων συνδυάζονται με την τεχνική της τελικής συνένωσης (late fusion).

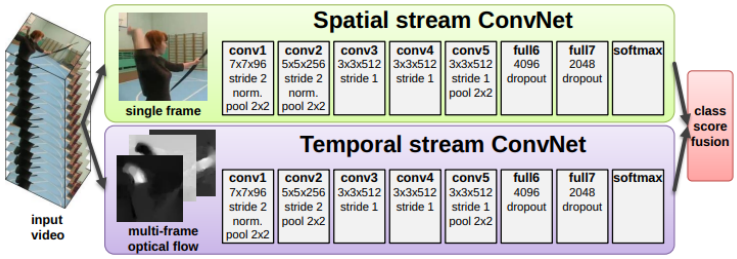


Figure 1.3.4: Δύο Ροών Δίκτυα (από [38])

Στο άρθρο με το οποίο εισήχθηκε το TSN, οι συγγραφείς εφάρμοσαν την αρχιτεκτονική στην αναγνώριση δράσεων σε βίντεο, χρησιμοποιώντας RGB εικόνες για την εξαγωγή χωρικών χαρακτηριστικών και την οπτική ροή για την εξαγωγή χρονικών χαρακτηριστικών. Συγκεκριμένα, στη χωρική ροή, οι ερευνητές εφάρμοσαν αναγνώριση δράσεων από στατικές εικόνες, σε αντίθεση με τη χρονική ροή, η οποία χρησιμοποίησε στοιβαγμένες οπτικές ροές μεταξύ αρκετών συνεχόμενων καρέ για την απόκτηση πληροφοριών κίνησης. Οι συγγραφείς αξιολόγησαν την προτεινόμενη αρχιτεκτονική σε δύο βάσεις δεδομένων, το UCF-101 και το HMDB-51, και πέτυχαν συγκρίσιμα αποτελέσματα σε σύγκριση με τα τότε state-of-the-art αποτελέσματα σε και τις δύο βάσεις δεδομένων.

Από την άλλη πλευρά, ο Chen και άλλοι στο [39] πρότειναν μια παραλλαγή της αρχιτεκτονικής TSN για την αναγνώριση χειρονομιών. Σε αυτήν την εργασία, οι ερευνητές χρησιμοποίησαν ένα διπλό κωδικοποιητή για την αντιμετώπιση των RGB δεδομένων και των χαρτών θερμοκρασίας. Ο διπλός κωδικοποιητής αποτελείται από δύο ξεχωριστούς κωδικοποιητές, ένα για τα RGB δεδομένα και ένα για τους χάρτες θερμοκρασίας, οι οποίοι συγχωνεύονται στο τέλος του δικτύου. Ο κωδικοποιητής είναι μια αρχιτεκτονική νευρωνικού δικτύου που λαμβάνει μια είσοδο και εξάγει μια χαρακτηριστική αναπαράσταση της εισόδου.

Residual Νευρωνικά Δίκτυα (Residual Network - ResNet)

Τα μοντέλα Βαθιάς Μάθησης έχουν την τάση να εξάγουν πιο πολύπλοκα χαρακτηριστικά στα πιο βαθιά επίπεδα του δικτύου, όπως εξηγήθηκε και νωρίτερα. Ωστόσο, η προσθήκη περισσότερων επιπέδων σε ένα βαθύ νευρωνικό δίκτυο δεν οδηγεί πάντα σε καλύτερη απόδοση. Αντίθετα, η απόδοση του δικτύου μπορεί να μειωθεί, φαινόμενο γνωστό ως πρόβλημα υποβάθμισης (degradation problem). Για να αντιμετωπιστεί το πρόβλημα υποβάθμισης, ο Kaiming He και άλλοι πρότειναν τη χρήση μιας δομής μάθησης, γνωστή ως Residual δομικό στοιχείο (Residual

Block) [40], όπως φαίνεται στο Σχήμα 1.3.5. Η δομή αυτή έχει τη δυνατότητα να μεταφέρει πληροφορία από τα εισερχόμενα στα εξερχόμενα επίπεδα, παραλλείπτοντας τα συνελικτικά επίπεδα ενός βαθιού νευρωνικού δικτύου, κρατώντας με αυτό τον τρόπο αναλλοίωτη την πληροφορία της εισόδου.

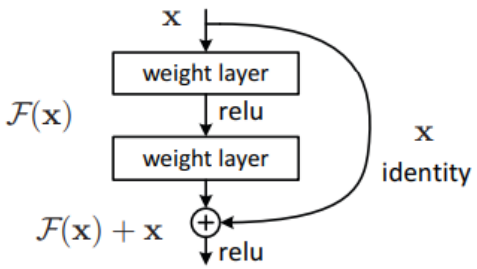


Figure 1.3.5: Residual Δομικό Στοιχείο (από [40])

Οι συγγραφείς πρότειναν τη χρήση των Residual Blocks για την εκπαίδευση βαθιών νευρωνικών δικτύων, επιτρέποντας την εκπαίδευση δικτύων με περισσότερα από 100 επίπεδα. Η προσθήκη περισσότερων επιπέδων σε ένα βαθύ νευρωνικό δίκτυο μπορεί να οδηγήσει σε καλύτερη αναπαράσταση των δεδομένων, αλλά μπορεί επίσης να οδηγήσει σε προβλήματα όπως το πρόβλημα υποβάθμισης. Το πρόβλημα υποβάθμισης προκύπτει από το γεγονός ότι η προσθήκη περισσότερων επιπέδων μπορεί να εκμηδενίσει προσεγγιστικά την παράγωγο της συνάρτησης σφάλματος με αποτέλεσμα να μην εκπαιδεύεται το δίκτυο. Το Residual δομικό στοιχείο λύνει αυτό το πρόβλημα εισάγοντας συνδέσεις παράλειψης, οι οποίες επιτρέπουν τη διάδοση της παραγώγου αναλλοίωτης σε προηγούμενα επίπεδα. Συγκεκριμένα, οι συνδέσεις παράλειψης αντιστοιχούν στην προσθήκη της εισόδου του επιπέδου στην έξοδο του επιπέδου, χωρίς να υποστεί κάποιο μετασχηματισμό.

Επομένως, το Residual νευρωνικό δίκτυο (ResNet) είναι ένα είδος συνελκτικού νευρωνικού δικτύου (CNN), που μπορεί να κατασκευάσει αρχιτεκτονικές μεγάλου βάθους, χωρίς να υποφέρει από τα προβλήματα υποβάθμισης και της εξαφάνισης της κλίσης. Ανάλογα με το βάθος του δικτύου, μπορούν να δημιουργηθούν διάφορα μοντέλα ResNet. Κάποια από τα πιο δημοφιλή μοντέλα ResNet είναι τα ResNet-18, ResNet-34, ResNet-50 και ResNet-101 με 18, 34, 50 και 101 επίπεδα αντίστοιχα. Για παράδειγμα, η αρχιτεκτονική του ResNet-18 αποτελείται από 4 στάδια, με κάθε στάδιο να περιέχει 2 Residual Blocks. Κάθε Residual Block αποτελείται από 2 συνελικτικά επίπεδα, τα οποία ακολουθούνται από ένα επίπεδο κανονικοποίησης πακέτων και ένα επίπεδο ενεργοποίησης ReLU, όπως φαίνεται στο Σχήμα 1.3.6.

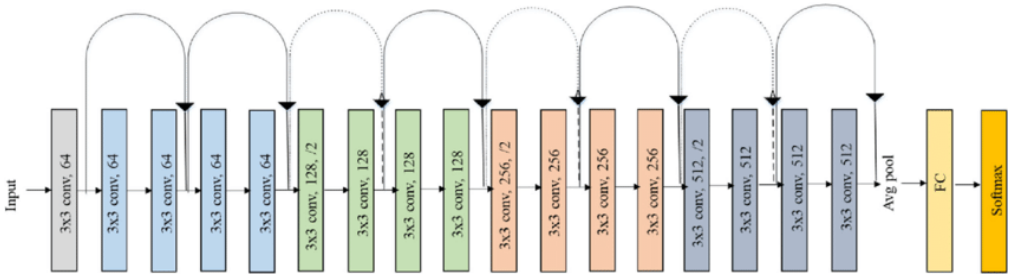


Figure 1.3.6: ResNet-18 αρχιτεκτονική (από [41])

Στο άρθρο στο οποίο προτάθηκε η αρχιτεκτονική του ResNet, οι συγγραφείς χρησιμοποίησαν το δίκτυο για την ταξινόμηση εικόνων στο ImageNet, μιας μεγάλης κλίμακας βάση δεδομένων με 1,2 εκατομμύρια εικόνες και 1.000 κλάσεις. Αργότερα, ο Kensho Hara και άλλοι πρότειναν μια 3D έκδοση του ResNet στο [42], για εργασίες ταξινόμησης βίντεο. Συγκεκριμένα, τα 2D συνελικτικά επίπεδα, καθώς και οποιεσδήποτε άλλες 2D λειτουργίες, αντικαταστάθηκαν από 3D συνελικτικά επίπεδα, ενώ η υπόλοιπη αρχιτεκτονική παρέμεινε ίδια.

Το ResNet έγινε γρήγορα δημοφιλές σε εργασίες ταξινόμησης βίντεο και μια ποικιλία μεθόδων και μοντέλων χρησιμοποίησαν το ResNet ως backbone. Στην αναγνώριση δράσεων, τα μοντέλα ResNet είναι τόσο δημοφιλή ώστε να έχουν χρησιμοποιηθεί σε πολλές εργασίες και να έχουν επιτύχει ανταγωνιστικά αποτελέσματα. Ομοίως στην αναγνώριση χειρονομιών, τα μοντέλα ResNet χρησιμοποιούνται συχνά, αλλά συνήθως συνδυάζονται με άλλες αρχιτεκτονικές για να επιτύχουν καλύτερη απόδοση. Για παράδειγμα, αρκετές παραλλαγές του ResNet

δοκιμάστηκαν στο σύνολο δεδομένων EgoGesture, ένα σύνολο δεδομένων με 83.000 βίντεο και 50 κλάσεις χειρονομιών, και πέτυχαν ανταγωνιστικά αποτελέσματα με τα state-of-the-art της εποχής, αποδεικνύοντας την αποτελεσματικότητα των ResNet μοντέλων σε εργασίες αναγνώρισης χειρονομιών.

1.3.3 Μεταφορά Γνώσης (Transfer Learning)

Η Μεταφορά Γνώσης είναι μια τεχνική μηχανικής μάθησης, όπου οι παράμετροι ενός μοντέλου, που έχει εκπαιδευτεί σε ένα πρόβλημα μεταφέρονται σε ένα άλλο μοντέλο, το οποίο θα επιλύσει ένα σχετικό πρόβλημα με αυτό στο οποίο εκπαιδεύτηκε το αρχικό. Αυτή η τεχνική λειτουργεί επειδή τα χαρακτηριστικά που μαθαίνονται από ένα πρόβλημα είναι συχνά χρήσιμα για ένα άλλο πρόβλημα. Έτσι, το να εκπαιδεύσουμε ένα μοντέλο σε ένα μεγάλο σύνολο δεδομένων και στη συνέχεια να χρησιμοποιήσουμε τα βάρη του μοντέλου για να εκπαιδεύσουμε ένα άλλο μοντέλο σε ένα μικρότερο σύνολο δεδομένων, συνήθως οδηγεί σε καλύτερη απόδοση, καθώς και σε ταχύτερη σύγκλιση, από το να εκπαιδεύσουμε το δεύτερο μοντέλο από την αρχή.

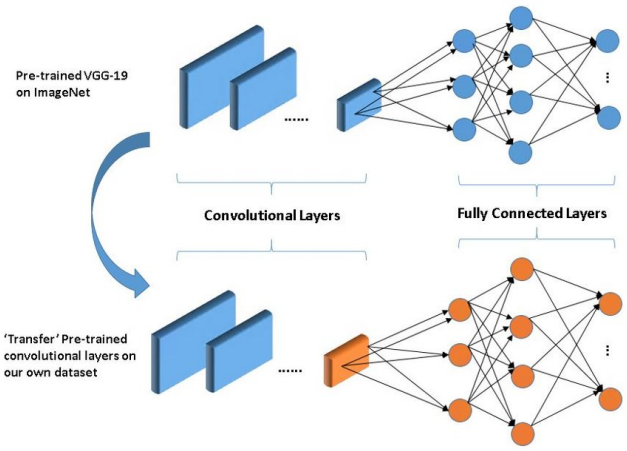


Figure 1.3.7: Αναπαράσταση Μεταφοράς Γνώσης (από [43])

Για την επιτυχή εφαρμογή της τεχνικής της Μεταφορά Γνώσης σε ένα πρόβλημα, είναι σημαντικό να επιλεγούν οι σωστές παράμετροι του μοντέλου, που θα μεταφερθούν. Στα αρχικά επίπεδα ενός βαθιού νευρωνικού δικτύου εξάγονται χαμηλού επιπέδου χαρακτηριστικά, όπως γραμμές και σχήματα, ενώ στα τελικά επίπεδα εξάγονται υψηλού επιπέδου χαρακτηριστικά, όπως αντικείμενα και σκηνές. Έτσι, αν το πρόβλημα στο οποίο εκπαιδεύτηκε το αρχικό μοντέλο είναι παρόμοιο με το πρόβλημα στο οποίο θα εκπαιδευτεί το δεύτερο μοντέλο, τότε είναι καλύτερο να μεταφερθούν περισσότερα επίπεδα του αρχικού μοντέλου. Αντίθετα, αν τα προβλήματα δεν είναι τόσο κοντά σημασιολογικά, τότε είναι καλύτερο να μεταφερθούν λιγότερα επίπεδα του αρχικού μοντέλου. Επιπλέον, είναι σημαντικό να επιλεγούν τα σωστά επίπεδα του αρχικού μοντέλου, που θα μεταφερθούν, καθώς η μεταφορά πολλών επιπέδων μπορεί να οδηγήσει σε αρνητικά αποτελέσματα. Συνεπώς, η επιλογή των επιπέδων που θα μεταφερθούν είναι κρίσιμη για την επιτυχή εφαρμογή της τεχνικής της Μεταφοράς Γνώσης και εξαρτάται άμεσα από την ομοιότητα των προβλημάτων.

Μεταφορά Γνώσης στην Αναγνώριση Ανθρώπινων Δράσεων

Ο Kensho Hara και άλλοι στο [44] παρουσίασαν τις επιδόσεις μεταφοράς γνώσης από το Kinetics σύνολο δεδομένων στα σύνολα δεδομένων UCF-101 και HMDB-51, για διάφορες παραλλαγές του ResNet. Συγκεκριμένα, οι συγγραφείς εκπαίδευσαν διάφορες παραλλαγές του ResNet στο σύνολο δεδομένων Kinetics και στη συνέχεια χρησιμοποίησαν τα εκπαιδευμένα βάρη για να εκπαιδεύσουν τα ίδια μοντέλα στα σύνολα δεδομένων UCF-101 και HMDB-51. Στα πειράματα που διεξήχθησαν, οι συγγραφείς εκπαίδευσαν μόνο το τελευταίο επίπεδο, το πλήρως συνδεδεμένο επίπεδο, των εκπαιδευμένων μοντέλων, μιας και δοκιμές, όπου επέτρεπαν την εκπαίδευση περισσότερων επιπέδων των μοντέλων, δεν έδωσαν καλύτερα αποτελέσματα. Τα αποτελέσματα έδειξαν ότι τα μοντέλα που εκπαιδεύτηκαν μεταφέροντας τα βάρη από το Kinetics σύνολο δεδομένων είχαν καλύτερη απόδοση από τα μοντέλα που εκπαιδεύτηκαν από την αρχή, όπως παρουσιάζεται στον Πίνακα 1.1. Αυτό αποδεικνύει την αποτελεσματικότητα της μεταφοράς γνώσης στην αναγνώριση δράσεων από σύνολα δεδομένων δράσεων και ότι τα εκπαιδευμένα μοντέλα μπορούν να χρησιμοποιηθούν ως μοντέλα βάσης για τη δοκιμή νέων αρχιτεκτονικών.

Μοντέλο	UCF-101	HMDB-51
ResNet-18 (from scratch)	42.4	17.1
ResNet-18	84.4	56.4
ResNet-34	87.7	59.1
ResNet-50	89.3	61.0
ResNet-100	88.9	61.7
ResNet-152	89.6	62.4
ResNet-200	89.6	63.5
DenseNet-121	87.6	59.6
ResNetXt-101	90.7	63.8

Table 1.1: Επιδόσεις προεκπαιδευμένων μοντέλων στις βάσεις δεδομένων UCF-101 και HMDB-51 (από [44])

Μεταφορά Γνώσης στην Αναγνώριση Ανθρώπινων Χειρονομιών

Η μεταφορά γνώσης έχει επίσης χρησιμοποιηθεί με επιτυχία στην αναγνώριση χειρονομιών. Στο [45] οι συγγραφείς χρησιμοποίησαν ένα προεκπαιδευμένο μοντέλο στο ImageNet σύνολο, σε ένα πρόβλημα αναγνώρισης χειρονομιών, με 5 κλάσεις. Συγκεκριμένα, οι συγγραφείς χρησιμοποίησαν το μοντέλο AlexNet, ένα δικτυακό μοντέλο με 5 συνελικτικά επίπεδα και 3 πλήρως συνδεδεμένα επίπεδα. Οι συγγραφείς εκπαίδευσαν το τελευταίο πλήρως συνδεδεμένο επίπεδο του μοντέλου στο σύνολο δεδομένων των χειρονομιών και πέτυχαν αποτελέσματα που ξεπέρασαν τα state-of-the-art της εποχής. Τα αποτελέσματα αυτά αποδεικνύουν την αποτελεσματικότητα της μεταφοράς γνώσης στην αναγνώριση χειρονομιών και την αξία της χρήσης προεκπαιδευμένων μοντέλων ως μοντέλα βάσης για την εκπαίδευση νέων μοντέλων.

Επιπλέον, οι Noha Sarhan και Simone Frintrop στο [46], έδειξαν ότι η μεταφορά γνώσης από την αναγνώριση δράσεων στην αναγνώριση χειρονομιών είναι επίσης αποτελεσματική. Συγκεκριμένα, οι συγγραφείς χρησιμοποίησαν ένα προεκπαιδευμένο μοντέλο στο Kinetics σύνολο δεδομένων, σε ένα πρόβλημα αναγνώρισης χειρονομιών νοηματικής γλώσσας. Το μοντέλο, που επιλέχθηκε, ήταν το InceptionV1, το οποίο εκπαιδεύτηκε στο Kinetics σύνολο δεδομένων και τα εκπαιδευμένα βάρη χρησιμοποιήθηκαν για την εκπαίδευση ενός μοντέλου στο ChaLearn IsoGD σύνολο δεδομένων, ένα σύνολο δεδομένων για την αναγνώριση χειρονομιών. Τα αποτελέσματα έδειξαν ότι το μοντέλο που εκπαιδεύτηκε με τα εκπαιδευμένα βάρη είχε καλύτερη απόδοση από τα state-of-the-art μοντέλα της εποχής, αποδεικνύοντας την αποτελεσματικότητα της μεταφοράς γνώσης από την αναγνώριση δράσεων στην αναγνώριση χειρονομιών.

1.3.4 Βάσεις Δεδομένων

Βάσεις Δεδομένων Αναγνώρισης Ανθρώπινων Δράσεων

Στην αναγνώριση ανθρώπινων δράσεων, υπάρχουν πολλές βάσεις δεδομένων, με διαφορετικά χαρακτηριστικά, όπως το μέγεθος, ο αριθμός των κλάσεων, το είδος των δράσεων, η ποικιλία των φόντων και άλλα. Είναι ένα πρόβλημα το οποίο διαθέτει πολλές βάσεις δεδομένων, με κάποιες από αυτές μάλιστα να έχουν πολύ μεγάλο πλήθος δεδομένων και πολλές κλάσεις. Αυτή η ποικιλία βάσεων δεδομένων έχει οδηγήσει στην ανάπτυξη πολλών μοντέλων και μεθόδων για την αναγνώριση ανθρώπινων δράσεων. Κάποια από τις πιο γνωστές βάσεις δεδομένων αναγνώρισης δράσεων είναι τα Kinetics, UCF-101, NTU-RGB+D, HMDB-51.

- **Η Kinetics [47] βάση δεδομένων**
  είναι μια μεγάλη βάση δεδομένων αναγνώρισης δράσεων, που περιέχει πολλές διαφορετικές κλάσεις, ενώ τα βίντεο έχουν προέλθει από το YouTube. Συνεπώς, τα βίντεο αυτά είναι ερασιτεχνικά και έχουν μεγάλες διακυμάνσεις σε φόντα, φωτισμό, στην ποιότητα και στις κινήσεις της κάμερας και άλλα χαρακτηριστικά. Η βάση αυτή έχει τρείς διαφορετικές εκδοχές, το Kinetics-400, το Kinetics-600 και το Kinetics-700.

  1. Η Kinetics-400 περιέχει περίπου 240.000 βίντεο χωρισμένα σε 400 κλάσεις, όπου κάθε κλάση περιέχει τουλάχιστον 400 βίντεο.
  2. Η Kinetics-600 περιέχει περίπου 360.000 βίντεο χωρισμένα σε 600 κλάσεις, όπου κάθε κλάση περιέχει τουλάχιστον 600 βίντεο.
  3. Η Kinetics-700 περιέχει περίπου 650.000 βίντεο χωρισμένα σε 700 κλάσεις, όπου κάθε κλάση περιέχει

τουλάχιστον 700 βίντεο.

- **Η UCF-101** [36] **βάση δεδομένων**
  αποτελείται από 13.320 βίντεο συλλεγμένα από το YouTube και χωρισμένα σε 101 κλάσεις, περιέχοντας δράσεις όπως κινήσεις σώματος, αλληλεπιδράσεις ανθρώπου-ανθρώπου, αλληλεπιδράσεις ανθρώπου-αντικειμένου, αθλήματα και μουσικά όργανα. Στα βίντεο αυτά δεν υπάρχουν περιορισμοί στο φόντο και στο φωτισμό, ενώ η ποιότητα των εικόνων είναι σε χαμηλό επίπεδο.

- **Η NTU-RGB+D** [48] **βάση δεδομένων**
  είναι μια μεγάλη βάση, που δημοσιεύθηκε σε δύο εκδοχές, την NTU-RGB+D 60 και την NTU-RGB+D 120.

  1. Η NTU-RGB+D 60 περιέχει 56.800 βίντεο χωρισμένα σε 60 κλάσεις, εκτελούμενες από 40 διαφορετικά άτομα.
  2. Η NTU-RGB+D 120 περιέχει 114.480 βίντεο χωρισμένα σε 120 κλάσεις, εκτελούμενες από 106 διαφορετικά άτομα. 600 βίντεο.

  Η NTU-RGB+D βάση περιλαμβάνει δράσεις όπως καθημερινές κινήσεις, κινήσεις σχετιζόμενες με την υγεία του ανθρώπου και αλληλεπιδράσεις ανθρώπου-ανθρώπου και ανθρώπου-αντικειμένου. Τα βίντεο λήφθηκαν μέσα σε εργαστήριο έχοντας σταθέρες συνθήκες φωτισμού και φόντου, ενώ χρησιμοποιήθηκαν τρεις διαφορετικές κάμερες για τη λήψη των βίντεο. Παράλληλα, τα βίντεο αυτά περιέχουν τόσο το RGB όσο και το βάθος των εικόνων, καθώς και παρέχονται δεδόμενα σκελετού.

- **Η HMDB-51** [49] **βάση δεδομένων**
  αποτελείται από 6.766 βίντεο χωρισμένα σε 51 κλάσεις, που περιέχουν δράσεις όπως αθλήματα, χοροί, κινήσεις σώματος και άλλες. Τα βίντεο αυτά προέρχονται από ταινίες και το διαδίκτυο και περιέχουν μεγάλες διακυμάνσεις σε φόντα, φωτισμό, στην ποιότητα και στις κινήσεις της κάμερας και άλλα χαρακτηριστικά.

Βάσεις Δεδομένων Αναγνώρισης Ανθρώπινων Χειρονομιών

Στην αναγνώριση ανθρώπινων χειρονομιών, το πλήθος των βάσεων δεδομένων δεν είναι τόσο εκτενές και μάλιστα ακόμα και η ποιότητα των δεδομένων δεν είναι πάντα ικανοποιητική, καθιστώντας το πρόβλημα αυτό πιο απαιτητικό από ότι ήδη ήταν από την φύση του. Παρόλα αυτά, υπάρχουν μερικές βάσεις δεδομένων, που χρησιμοποιούνται συχνά για την αναγνώριση χειρονομιών, όπως η ChaLearn IsoGD, η NVGesture, η EgoGesture και η Jester.

- **Η Chalearn IsoGD** [50] **βάση δεδομένων**
  είναι μια μεγάλη βάση δεδομένων για την αναγνώριση χειρονομιών, με δεδομένα RGB-D, rgb και βάθους. Η βάση αυτή περιέχει 47.933 βίντεο χωρισμένα σε 249 κλάσεις, που περιέχουν χειρονομίες από 21 διαφορετικά άτομα. Οι χειρονομίες αυτές προέρχονται από διάφορους τομείς, όπως η νοηματική γλώσσα, οι καθημερινές χειρονομίες και άλλες.

- **Η NVGesture** [51] **βάση δεδομένων**
  είναι μια βάση δεδομένων για την αναγνώριση χειρονομιών, που περιέχει δεδομένα από πολλαπλούς αισθητήρες, όπως χρώμα, βάθος και στερεο-IR. Η βάση αυτή περιέχει 1.532 δυναμικές χειρονομίες χωρισμένες σε 25 κλάσεις, που προέρχονται από 20 διαφορετικά άτομα. Για τη συλλογή των δεδομένων, οι άνθρωποι εκτελούσαν χειρονομίες με το δεξί τους χέρι, ενώ το αριστερό τους χέρι κρατούσε ένα τιμόνι.

- **Η EgoGesture** [37] **βάση δεδομένων**
  είναι μια μεγάλη βάση δεδομένων, όπου τα δείγματά της προέρχονται από την πρώτη προσωπική οπτική γωνία. Η βάση αυτή περιέχει 24.161 RGB-D δείγματα χωρισμένα σε 83 κλάσεις, που περιέχουν στατικές και δυναμικές χειρονομίες, ειδικά για την αλληλεπίδραση με φορετά συστήματα.

- **Η Jester** [52] **βάση δεδομένων**
  είναι μια μεγάλη βάση δεδομένων, αποτελούμενη από 148.092 βίντεο χωρισμένα σε 27 κλάσεις, που περιέχουν χειρονομίες από 25 διαφορετικές κλάσεις χειρονομιών και δύο κλάσεις που δεν περιέχουν χειρονομίες. Είναι ένα ρεαλιστικό σύνολο δεδομένων για την αναγνώριση χειρονομιών, που περιέχει δεδομένα που έχουν ληφθεί από την οθόνη υπολογιστή, προσομοιώνοντας την καθημερινή χρήση των χειρονομιών, με διαφοροποιήσεις στο φόντο, στο φωτισμό και στην κίνηση της κάμερας.

1.4 Μάθηση Πολλαπλών Εργασιών

1.4.1 Μάθηση Πολλαπλών Εργασιών (Multi-task learning - MTL)

Η Μάθηση Πολλαπλών Εργασιών (MTL) προτάθηκε από τον Caruana στο [53], κατά την οποία πολλαπλές εργασίες λύνονται ταυτόχρονα, εκμεταλλευόμενες τα κοινά στοιχεία και τις διαφορές μεταξύ τους. Η Μάθηση Πολλαπλών Εργασιών αντικατοπτρίζει τον τρόπο που οι άνθρωποι σκέφονται καλύτερα από τη μάθηση μίας μόνο εργασίας. Κάθε φορά που ένας άνθρωπος προσπαθεί να μάθει κάτι νέο, χρησιμοποιεί ένα τεράστιο όγκο προηγούμενων γνώσεων. Αντίστοιχα, τα νευρωνικά δίκτυα απαιτούν τέτοιον τεράστιο αριθμό παραδειγμάτων εκπαίδευσης και χρόνο υπολογισμού, όταν δεν χρησιμοποιείται καμία προηγούμενη γνώση. Έτσι, οι αρχιτεκτονικές MTL αυξάνουν την αποδοτικότητα των δεδομένων και μπορεί να οδηγήσουν σε ταχύτερη ταχύτητα μάθησης, βοηθώντας να μειωθεί η ανάγκη για μεγάλες απαιτήσεις δεδομένων και μεγάλους υπολογιστικούς πόρους.

Ορισμός Εργασίας (Task)

Μια εργασία είναι ένα συγκεκριμένο πρόβλημα μάθησης, με τη δική της συνάρτηση στόχου και το δικό της σύνολο δεδομένων, το οποίο πρέπει να λύσει ένα μοντέλο. Μια εργασία μπορεί να οριστεί ως κατηγορία προβλήματος μηχανικής μάθησης, όπως η ταξινόμηση ή η παλινδρόμηση. Για παράδειγμα, στις εργασίες όρασης υπολογιστών, οι εργασίες μπορεί να είναι η ταξινόμηση εικόνας βίντεο ή η ανίχνευση αντικειμένου. Επιπλέον, μια εργασία T^i συνήθως συνοδεύεται από ένα σύνολο εκπαίδευσης D^i που αποτελείται από n^i δείγματα εκπαίδευσης, δηλαδή $D^i = \{x^i_j, y^i_j\}$ για $j = 1, ..., n^i$, όπου $x^i_j \in R$ είναι το j-οστό δείγμα εκπαίδευσης στο D^i και y^i_j είναι η ετικέτα του. Ορίζουμε με X_i τον πίνακα δεδομένων εκπαίδευσης για την εργασία T^i, δηλαδή, $X^i = (x^i_1, ..., x^i_{n_i})$, όπως ορίζεται από τους Zhang και Yang στο [54]. Έτσι, οι εργασίες μπορούν επίσης να διακριθούν ανάλογα με τα σύνολα δεδομένων τους, καθώς και τον τύπο των δεδομένων που χρησιμοποιούν. Για παράδειγμα, διαφορετικά σύνολα δεδομένων μπορούν να υποδηλώνουν διαφορετικές εργασίες και διαφορετικές εργασίες μπορούν να χρησιμοποιούν διαφορετικές κατηγορίες δεδομένων, όπως rgb, βάθος.

Τύποι Μάθησης Πολλαπλών Εργασιών

Όπως αναφέρθηκε προηγουμένως, οι εργασίες μπορούν να διακριθούν ανάλογα με τα σύνολα δεδομένων τους, τον τύπο των δεδομένων που χρησιμοποιούν και τις συναρτήσεις στόχου τους. Έτσι, οι μέθοδοι μάθησης πολλαπλών εργασιών μπορούν να χωριστούν σε δύο τύπους, ανάλογα με τον τύπο των εργασιών που λύνουν, στις ομογενείς και στις ετερογενείς εργασίες.

- **Στην ομογενή μάθηση πολλαπλών εργασιών**
  όλες οι εργασίες είναι ίδιου τύπου προβλήματος, δηλαδή για παράδειγμα στην όραση υπολογιστών, όλες οι εργασίες μπορεί να είναι εργασίες ταξινόμησης εικόνας/βίντεο.

- **Στην ετερογενή μάθηση πολλαπλών εργασιών**
  οι εργασίες είναι διαφορετικού τύπου, δηλαδή για παράδειγμα στην όραση υπολογιστών, μια εργασία μπορεί να είναι εργασία ταξινόμησης εικόνας/βίντεο, ενώ μια άλλη εργασία μπορεί να είναι εργασία ανίχνευσης αντικειμένου.

Επιπλέον, οι μέθοδοι μάθησης πολλαπλών εργασιών μπορούν να διαχωριστούν σε δύο τύπους, ανάλογα με τον τύπο των δεδομένων που χρησιμοποιούν, σε ομογενή χαρακτηριστικών και ετερογενή χαρακτηριστικών μάθηση πολλαπλών εργασιών.

- **Στην ομογενών χαρακτηριστικών μάθηση πολλαπλών εργασιών**
  τα χαρακτηριστικά που χρησιμοποιούνται σε κάθε εργασία είναι του ίδιου τύπου, δηλαδή για δείγματα d_i από D_i και d_j από D_j, d_i ισούται με d_j για κάθε $i \neq j$. Για παράδειγμα στην όραση υπολογιστών, όλες οι εργασίες μπορεί να χρησιμοποιούν δεδομένα rgb.

- **Στην ετερογενών χαρακτηριστικών μάθηση πολλαπλών εργασιών**
  τα χαρακτηριστικά που χρησιμοποιούνται σε κάθε εργασία είναι διαφορετικού τύπου, δηλαδή για δείγματα d_i από D_i και d_j από D_j, d_i δεν ισούται με d_j για κάθε $i \neq j$. Για παράδειγμα στην όραση υπολογιστών, μια εργασία μπορεί να χρησιμοποιεί δεδομένα rgb, ενώ μια άλλη εργασία μπορεί να χρησιμοποιεί δεδομένα σκελετού.

Ακόμα, οι μέθοδοι μάθησης πολλαπλών εργασιών μπορούν να διαχωριστούν σε δύο τύπους, ανάλογα με τα σύνολα δεδομένων που χρησιμοποιούν, σε μάθηση πολλαπλών εργασιών με το ίδιο σύνολο δεδομένων και με διαφορετικό σύνολο δεδομένων.

- **Στην μάθηση πολλαπλών εργασιών με το ίδιο σύνολο δεδομένων**
  όλες οι εργασίες χρησιμοποιούν τα δείγματα του ίδιου συνόλου δεδομένων.
- **Στην μάθηση πολλαπλών εργασιών με διαφορετικά σύνολα δεδομένων**
  οι εργασίες χρησιμοποιούν δείγματα από διαφορετικά σύνολα δεδομένων.

Επιλογή Εργασιών για Κοινή Μάθηση

Αν και η μάθηση πολλαπλών εργασιών μπορεί να είναι ωφέλιμη σε πολλές περιπτώσεις, η μάθηση εννοιών για πολλαπλές εργασίες φέρνει δυσκολίες, τις οποίες η μάθηση μίας μόνο εργασίας δεν αντιμετωπίζει. Συγκεκριμένα, κάποιες εργασίες μπορεί να έχουν αντικρουόμενες ανάγκες. Σε αυτήν την περίπτωση, η αύξηση της απόδοσης ενός μοντέλου σε μία εργασία θα βλάψει την απόδοση σε μία εργασία με διαφορετικές ανάγκες, οδηγώντας συνεπώς σε χειρότερη συνολική απόδοση. Αυτό το φαινόμενο ονομάζεται *αρνητική μεταφορά (negative transfer)* ή *καταστροφική παρέμβαση (destructive inference)*. Η αρνητική μεταφορά μπορεί να συμβεί όταν οι εργασίες δεν είναι σχετικές μεταξύ τους, ή όταν οι εργασίες είναι σχετικές μεταξύ τους, αλλά η μέθοδος διαμοιρασμού βαρών που χρησιμοποιείται στο μοντέλο MTL δεν είναι κατάλληλη για τις εργασίες. Έτσι, προκύπτουν δύο ερωτήσεις.

- **Ποιες εργασίες πρέπει να μάθουν μαζί;**
  Οι εργασίες πρέπει να είναι σχετικές μεταξύ τους, ώστε το μοντέλο να μπορεί να εκμεταλλευτεί τα κοινά και τις διαφορές μεταξύ τους.
- **Πώς πρέπει να μάθουν μαζί οι εργασίες;**
  Η μέθοδος διαμοιρασμού βαρών πρέπει να είναι κατάλληλη για τις εργασίες, ώστε το μοντέλο να μπορεί να εκμεταλλευτεί τα κοινά και τις διαφορές μεταξύ τους.

Όμως η απάντηση σε αυτές τις ερωτήσεις δεν είναι δυαδική. Η απάντηση εξαρτάται από τις εργασίες, τα σύνολα δεδομένων, τη μέθοδο διαμοιρασμού βαρών που χρησιμοποιείται στο μοντέλο MTL, καθώς και τη συνολική αρχιτεκτονική και τις μεθόδους που χρησιμοποιούνται για τη βελτιστοποίηση των παραμέτρων του μοντέλου. Έτσι, αν δύο εργασίες πρέπει να μάθουν μαζί ή όχι, είναι μια ερώτηση που πρέπει να απαντηθεί ανά περίπτωση. Ακόμα και αν συμβεί αρνητική μεταφορά, δεν σημαίνει ότι το μοντέλο MTL δεν είναι χρήσιμο. Ένας τρόπος για να αντιμετωπιστεί η αρνητική μεταφορά είναι να σταθμιστούν απώλειες κάθε εργασίας, ώστε οι εργασίες να μην επικρατούν η μία της άλλη λόγω διαφορών κλίμακας απωλειών [55]. Επίσης, έχουν διεξαχθεί πολλές έρευνες σχετικά με τη σχετικότητα των εργασιών, όπως η έρευνα των Standley και άλλων στο [56], όπου πρότειναν ένα πλαίσιο για την καθορισμό της πιο ωφέλιμης ομαδοποίησης εργασιών για ένα σύνολο εργασιών.

1.4.2 Μέθοδοι Διαμοιρασμού Παραμέτρων

Κατά τη σχεδίαση μιας αρχιτεκτονικής μάθησης πολλαπλών εργασιών, υπάρχουν πολλοί διαφορετικοί παράγοντες που πρέπει να ληφθούν υπόψη, όπως το ποσοστό των παραμέτρων του μοντέλου που θα μοιραστούν μεταξύ των εργασιών, καθώς και τον τρόπο παραμετροποίησης και συνδυασμού των εργασιών. Πολλές από τις προτεινόμενες αρχιτεκτονικές για την MTL προσπαθούν να ισορροπήσουν το βαθμό πληροφοριών που μοιράζονται μεταξύ των εργασιών, καθώς πολύς διαμοιρασμός θα οδηγήσει σε αρνητική μεταφορά και μπορεί να προκαλέσει χειρότερη απόδοση των μοντέλων MTL από τα μοντέλα που εκπαιδεύονται ξεχωριστά για κάθε εργασία, ενώ πολύ λίγος διαμοιρασμός δεν επιτρέπει στο μοντέλο να εκμεταλλευτεί αποτελεσματικά τις πληροφορίες μεταξύ των εργασιών. Παρόλο που υπάρχουν πολλοί τρόποι για την κοινή χρήση πληροφοριών μεταξύ των εργασιών, οι πιο συνηθισμένοι είναι ο ολικός διαμοιρασμός παραμέτρων (hard parameter sharing), ο μερικός διαμοιρασμός παραμέτρων (soft parameter sharing) και η δρομολόγηση εργασιών (task routing).

Ολικός Διαμοιρασμός Παραμέτρων (Hard Parameter Sharing)

Ο ολικός διαμοιρασμός παραμέτρων, που προτάθηκε από τον Caruana στο [53], είναι η πιο συνηθισμένη μέθοδος για το διαμοιραμό πληροφορίας μεταξύ των εργασιών σε ένα μοντέλο MTL. Συνήθως εφαρμόζεται με τον κοινό διαμοιρασμό των κρυφών επιπέδων μεταξύ όλων των εργασιών, ενώ διατηρούνται διαφορετικά τα επίπεδα εξόδου για κάθε εργασία, όπως φαίνεται στο Σχήμα 1.4.1. Τα κοινά επίπεδα εκπαιδεύονται από κοινού σε όλες τις

εργασίες, ενώ τα επίπεδα εξόδου εκπαιδεύονται ανεξάρτητα για κάθε εργασία. Τα κοινά επίπεδα συνήθως είναι τα πρώτα επίπεδα του δικτύου, ενώ τα επίπεδα εξόδου είναι τα τελευταία επίπεδα του δικτύου. Η λογική πίσω από αυτήν την προσέγγιση είναι ότι τα κοινά επίπεδα θα μάθουν χαρακτηριστικά που είναι χρήσιμα για όλες τις εργασίες, καθώς τα πρώτα επίπεδα ενός δικτύου μάθησης μαθαίνουν πιο γενικά χαρακτηριστικά, ενώ τα επίπεδα εξόδου θα μάθουν χαρακτηριστικά που είναι χρήσιμα μόνο για την κάθε εργασία.

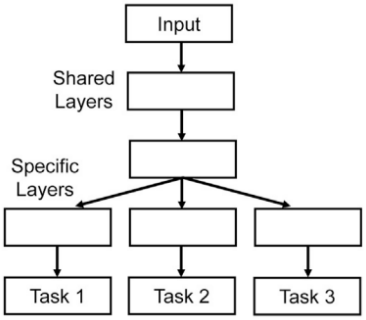


Figure 1.4.1: Αρχιτεκτονική Ολικού Διαμοιρασμού Παραμέτρων (από [57])

Μερικός Διαμοιρασμός Παραμέτρων (Soft Parameter Sharing)

Ο μερικός διαμοιρασμός παραμέτρων είναι μια πιο ευέλικτη προσέγγιση για το διαμοιρασμό πληροφορίας μεταξύ των εργασιών σε ένα μοντέλο MTL, μιας και επιτρέπει συγκεκριμένη πληροφορία να μοιραστεί μεταξύ των εργασιών, αλλά και περισσότερη πληροφορία να είναι ειδική για κάθε εργασία. Συγκεκριμένα, στον μερικό διαμοιρασμό παραμέτρων, κάθε εργασία έχει το δικό της μοντέλο με τις δικές της παραμέτρους, αλλά οι παράμετροι των μοντέλων των εργασιών είναι συνδεδεμένες μεταξύ τους μέσω ενός κοινού μηχανισμού κανονικοποίησης, όπως φαίνεται στο Σχήμα 1.4.2. Ο μηχανισμός κανονικοποίησης μπορεί να είναι η L_1 νόρμα, η L_2 νόρμα, η νόρμα του ίχνου (trace norm) ή άλλοι μηχανισμοί κανονικοποίησης. Ο μηχανισμός κανονικοποίησης είναι ρυθμισμένος από μια υπερπαράμετρο λ, που ελέγχει το πόση πληροφορία μοιράζεται μεταξύ των εργασιών. Όσο μεγαλύτερη είναι η τιμή της υπερπαραμέτρου λ, τόσο περισσότερη πληροφορία μοιράζεται μεταξύ των εργασιών, και τόσο λιγότερο είναι το ενδεχόμενο του overfitting στην αρχική εργασία.

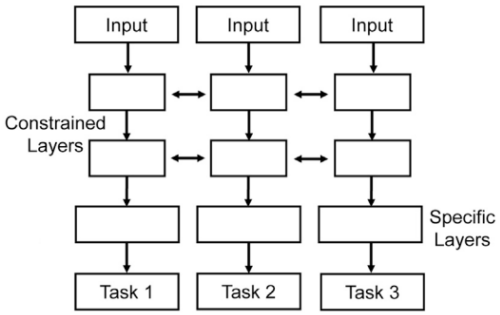


Figure 1.4.2: Αρχιτεκτονική Μερικού Διαμοιρασμού Παραμέτρων (από [57])

Μια από τις πιο διαδεδομένες μεθόδους μερικού διαμοιρασμού παραμέτρων είναι τα δικτύα Cross-Stitch, που προτάθηκαν από τον Misra και άλλους στο [58]. Τα δίκτυα Cross-Stitch αποτελούνται από ένα πλήθος μοντέλων, ένα για κάθε εργασία, που συνδέονται μεταξύ τους μέσω μονάδων Cross-Stitch, όπως φαίνεται στο Σχήμα 1.4.3. Οι μονάδες Cross-Stitch εκπαιδεύονται από κοινού με το υπόλοιπο δίκτυο, και η βέλτιστη συνδυαστική έξοδος των δύο δικτύων μαθαίνεται κατά την εκπαίδευση του μοντέλου. Τα επίπεδα, που συνδέονται με μονάδες Cross-Stitch, δεν είναι απαραίτητα τα ίδια για όλες τις εργασίες. Για παράδειγμα, στην αρχική εργασία των συγγραφέων, εφαρμόστηκαν μονάδες Cross-Stitch μετά από κάθε συνελκτικό επίπεδο και μετά από κάθε επίπεδο συγκέντρωσης, και βρέθηκε ότι οι μονάδες Cross-Stitch μετά από τα επίπεδα συγκέντρωσης είχαν καλύτερη απόδοση. Επιπλέον, ρυθμίζοντας τις παραμέτρους των μονάδων Cross-Stitch, μπορεί να ελεγχθεί το ποσοστό της πληροφορίας που μοιράζεται μεταξύ των εργασιών.

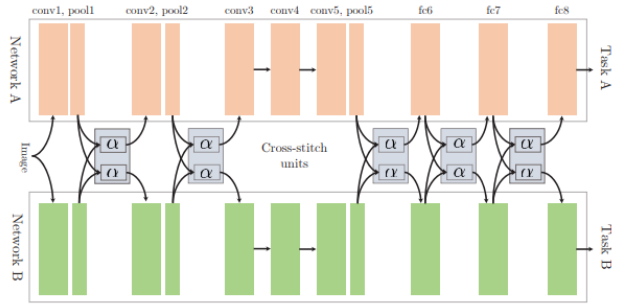


Figure 1.4.3: Μονάδες cross stitch στο AlexNet (ένα γνωστό συνελκτικό νευρωνικό δίκτυο) (από [58])

Δρομολόγηση Εργασιών (Task Routing)

Ο Strezoski και άλλοι στο [59] πρότειναν μια προσέγγιση για την κοινή μάθηση πολλαπλών εργασιών, που ονομάζεται Δίκτυα Δρομολόγησης Εργασιών (Task Routing Networks). Η προσέγγιση αυτή επιτρέπει το διαμοιρασμό παραμέτρων μεταξύ των εργασιών σε επίπεδο χαρακτηριστικών, αντί σε επίπεδο επιπέδων, όπως στον ολικό διαμοιρασμό παραμέτρων, και επιτρέπει την εκπαίδευση ενός υπο-δικτύου για κάθε εργασία, που μοιράζεται παραμέτρους με τα υπόλοιπα υπο-δίκτυα. Η δρομολόγηση των εργασιών επιτυγχάνεται μέσω της εφαρμογής ενός δικτύου δρομολόγησης εργασιών, που επιλέγει ποια χαρακτηριστικά θα μοιραστούν μεταξύ των εργασιών, όπως φαίνεται στο Σχήμα 1.4.4.

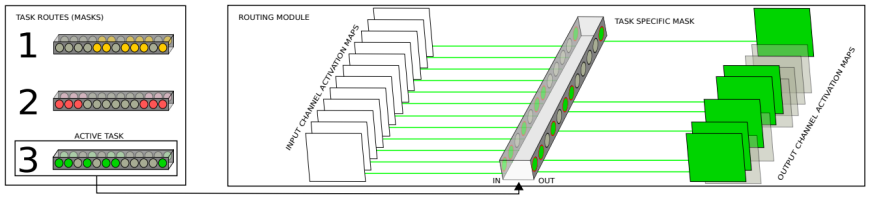


Figure 1.4.4: Ενεργοποίηση συγκεκριμένων συνελκτικών επιπέδων για τη δρομολόγηση εργασίας (από [59])

Αυτές οι μάσκες, που εφαρμόζονται στα επίπεδα, για να διαχωρίσουν τα χαρακτηριστικά σε εργασίες δεν εκπαιδεύονται, αλλά αρχικοποιούνται τυχαία στην αρχή της εκπαίδευσης και παραμένουν σταθερές. Αν και με αυτό τον τρόπο περιορίζεται ο σχεδιασμός διαμοιρασμού παραμέτρων μεταξύ των εργασιών, το μοντέλο έχει ακόμα τον έλεγχο του βαθμού διαμοιρασμού μεταξύ των εργασιών μέσω της χρήσης ενός υπερπαραμέτρου σ, γνωστής ως ποσοστό διαμοιρασμού. Η υπερπαράμετρος σ παίρνει τιμές μεταξύ 0 και 1, καθορίζοντας το ποσοστό των μονάδων σε κάθε επίπεδο, που είναι συγκεκριμένες για κάθε εργασία. Όσο μεγαλύτερη είναι η τιμή της υπερπαραμέτρου σ, τόσο περισσότερη πληροφορία μοιράζεται μεταξύ των εργασιών.

Προχωρημένες Μέθοδοι Διαμοιρασμού Παραμέτρων

Οι μέθοδοι διαμοιρασμού παραμέτρων, που περιγράφηκαν παραπάνω, έχουν κάποιους περιορισμούς, που εμποδίζουν τα μοντέλα να εκμεταλλευτούν αποτελεσματικά τις πληροφορίες μεταξύ των εργασιών. Για αυτό το λόγο έχουν προταθεί πιο προχωρημένες μέθοδοι διαμοιρασμού παραμέτρων, που προσπαθούν να αποφύγουν αυτούς τους περιορισμούς. Μία από αυτές είναι η μέθοδος Εκμάθησης Διαμοιρασμού Βαρών (Learned Weight Sharing - LWS), που προτάθηκε από τον Prellberg και άλλους στο [60]. Στόχος αυτής της μεθόδου είναι να μάθει ποιο σύνολο βαρών θα διαμοιράζεται για κάθε εργασία. Για παράδειγμα στο Σχήμα 1.4.5 απεικονίζονται διάφορες τεχνικές διαμοιρασμού παραμέτρων για την επίλυση ενός προβλήματος MTL με τρεις εργασίες. Κάθε κουτί απεικονίζει ένα επίπεδο δικτύου και τα γράμματα υποδηλώνουν διαφορετικά βάρη. Τα χρωματισμένα τμήματα σε κάθε κουτί εικονίζουν ποιες εργασίες μοιράζονται βάρη για αυτό το επίπεδο. Η LWS μαθαίνει, συνεπώς, μια ανάθεση βαρών συγκεκριμένη σε κάθε επίπεδο με βάση κάθε εργασία, κάνοντάς την πιο ευέλικτη από τις προηγούμενες μεθόδους, μιας και εξαρτάται από τις εργασίες, που εκπαιδεύονται από κοινού, και όχι από τα δεδομένα, που χρησιμοποιούνται.

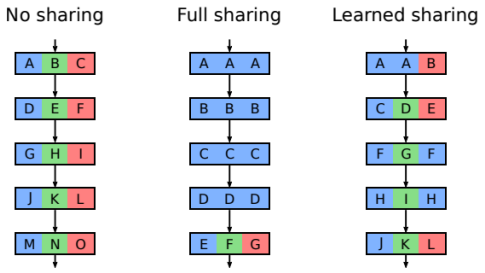


Figure 1.4.5: Διαφορετικές μέθοδοι διαμοιρασμού παραμέτρων μεταξύ μοντέλων ειδικών για κάθε εργασία (από [60])

Για τη δημιουργία ενός μοντέλου LWS πολλαπλών εργασιών, επιλέγεται οποιαδήποτε αρχιτεκτονική νευρωνικού δικτύου ως βασική αρχιτεκτονική και αντιγράφεται μία φορά για κάθε εργασία. Το τελευταίο επίπεδο κάθε μοντέλου τροποποιείται για να έχει τον κατάλληλο αριθμό εξόδων για την εργασία, στην οποία αντιστοιχεί. Για τα κοινά επίπεδα δημιουργείται ένα σύνολο K βαρών και όλα τα N επίπεδα των δικτύων εργασίας αντιστοιχίζονται σε ένα βάρος από το αντίστοιχο σύνολο K. Με το να αντιστοιχίζεται το ίδιο βάρος σε πολλά επίπεδα δικτύων εργασίας, επιτυγχάνεται ο διαμοιρασμός βαρών. Το πρόβλημα συνεπώς ανάγεται στην εύρεση καλών αντιστοιχίσεων μεταξύ των βαρών και των επιπέδων των δικτύων εργασίας και ταυτόχρονα να εκπαιδευτούν τα ίδια τα βάρη. Για να επιτευχθεί αυτό, οι συγγραφείς προτείνουν να χρησιμοποιηθούν δύο αλγόριθμοι βελτιστοποίησης, ένας για το πρόβλημα της αντιστοίχισης και ένας για τη βελτιστοποίηση των βαρών οι ίδιοι. Συγκεκριμένα, το πρόβλημα της ανάθεσης διαμοιρασμού των βαρών στις αντίστοιχες εργασίες λύνεται με τον αλγόριθμο Στρατηγικής Φυσικής Εξέλιξης (Natural Evolution Strategy - NES) και η βελτιστοποίηση των βαρών με τον αλγόριθμο Στοχαστικής Κλίσης Καθόδου (Stochastic Gradient Descent - SGD).

Ο αλγόριθμος NES αναφέρεται σε μια κλάση αλγορίθμων βελτιστοποίησης, που ενημερώνουν μια κατανομή πιθανότητας προς την κατεύθυνση της υψηλότερης αναμενόμενης απόδοσης, χρησιμοποιώντας τη φυσική κλίση (natural gradient) της αναμενόμενης απόδοσης. Η φυσική κλίση είναι μια παραλλαγή της κλασικής κλίσης, που λαμβάνει υπόψη την καμπυλότητα στο χώρο των παραμέτρων, δηλαδή τις δεύτερες παραγώγους της συνάρτησης απώλειας ως προς τις παραμέτρους, για να προσαρμόσει το ρυθμό μάθησης για κάθε παράμετρο.

Επόμενως, η αναζήτηση καλών αντιστοιχίσεων μεταξύ των βαρών και των επιπέδων των δικτύων κάθε εργασίας ισοδυναμεί με ένα πρόβλημα βελτιστοποίησης που περιγράφεται μαθηματικά ως εξής:

$$\min_{\theta,\alpha} f(\theta, \alpha), \tag{1.4.1}$$

όπου το $f : \Theta \times A \to R$ είναι η συνάρτηση απώλειας του μοντέλου, το $\theta \in \Theta$ είναι το σύνολο των βαρών του μοντέλου και το $\alpha \in A$ είναι το σύνολο των αντιστοιχίσεων μεταξύ των βαρών και των επιπέδων των δικτύων εργασίας. Η σύναρτηση σφάλματος f είναι διαφορίσιμη ως προς τα βάρη θ, όχι όμως και ως προς τις αντιστοιχίσεις α. Έτσι, οι συγγραφείς προτείνουν μια στοχαστική εκδοχή του προβλήματος, εισάγοντας μια κατανομή πιθανότητας π πάνω στο σύνολο όλων των πιθανών αντιστοιχίσεων βαρών στα επίπεδα των δικτύων εργασίας A, με πυκνότητα πιθανότητας $p(\alpha|\pi)$ - μια στατιστική έκφραση που ορίζει μια κατανομή πιθανότητας για μια συνεχή τυχαία μεταβλητή, που σημαίνει ότι η πιθανότητα της τυχαίας μεταβλητής να πέσει σε οποιοδήποτε συγκεκριμένο εύρος τιμών μπορεί να υπολογιστεί με την ολοκλήρωση της πυκνότητας πιθανότητας πάνω σε αυτό το εύρος.

Το βελτιστοποιητικό πρόβλημα περιγράφεται τότε ως εξής:

$$\min_{\theta,\pi} J(\theta, \pi) = E_{\pi}[f(\theta, \alpha)]. \tag{1.4.2}$$

Η στοχαστική διατύπωση του προβλήματος επιτρέπει τις αντιστοιχίσεις να βελτιστοποιηθούν μέσω της παραμέτρου π χρησιμοποιώντας τον αλγόριθμο NES, αλλά απαιτεί τη δειγματοληψία των αντιστοιχίσεων για τον υπολογισμό της κλίσης ως προς την παράμετρο θ. Με αυτό τον τρόπο το διακριτό, μη διαφορισιμό πρόβλημα των αντιστοιχίσεων α μετατρέπεται σε ένα συνεχές, διαφορισιμό πρόβλημα ως προς την παράμετρο π. Στην

πράξη, αντί να επιλέγονται απευθείας οι αντιστοιχίσεις, ο αλγόριθμος επιλέγει τις παραμέτρους της κατανομής π και οι αντιστοιχίσεις επιλέγονται στη συνέχεια με βάση την κατανομή πιθανότητας, που ορίζεται από αυτές τις παραμέτρους. Η βελτιστοποίηση του προβλήματος επιτυγχάνεται με την εναλλαγή μεταξύ των ακόλουθων βημάτων βελτιστοποίησης:

Βελτιστοποίηση Ανάθεσης Βαρών Για την βελτιστοποίηση της κατανομής πιθανότητας π ενώ τα βάρη θ παραμένουν σταθερά, επιλέγονται οι αντιστοιχίσεις των βαρών $a_1, ..., a_{\lambda_\pi}$ που κατανέμονται σύμφωνα με την πιθανότητα $p(\alpha|\pi)$ και υπολογίζονται οι τιμές της συνάρτησης απώλειας $l_i = f(\theta, \alpha_i)$ σε ένα batch δεδομένων εκπαίδευσης για όλες τις αντιστοιχίσεις. Οι συγγραφείς προτείνουν τη χρήση ενός εκτιμητή Μοντέ Κάρλο, με μέγεθος πληθυσμού λ, της κλίσης της συνάρτησης απώλειας ως προς την παράμετρο π ως εξής:

$$\nabla_\pi J(\theta, \pi) \approx \frac{1}{\lambda_\pi} \sum_{i=1}^{\lambda_\pi} u_i \nabla_\pi \log p(\alpha_i|\pi), \tag{1.4.3}$$

όπου το u_i δηλώνει τις τιμές χρησιμότητας, που δημιουργούνται για να καθίσταται ο αλγόριθμος ανεξάρτητος από την κλίμακα της συνάρτησης απώλειας. Οι τιμές απώλειας l_i μετατρέπονται σε τιμές χρησιμότητας χρησιμοποιώντας την ακόλουθη έκφραση:

$$u_i = 2 \cdot \frac{l_i - 1}{\lambda_\pi - 1} - 1. \tag{1.4.4}$$

Η κλίση χρησιμοποιείται στη συνέχεια για την ενημέρωση των παραμέτρων της κατανομής π με ρυθμό μάθησης η_π, σύμφωνα με το ακόλουθο βήμα στην κατεύθυνση της $\nabla_\pi J(\theta, \pi)$:

$$\pi + \eta_\pi \nabla_\pi J(\theta, \pi). \tag{1.4.5}$$

Βελτιστοποίηση Βαρών Για τη βελτιστοποίηση των βαρών θ ενώ οι τιμές της κατανομής πιθανότητας π παραμένουν σταθερές, οι συγγραφείς ακολουθούν μια Μόντε Κάρλο προσέγγιση. Συγκεκριμένα, οι αντιστοιχίσεις των βαρών στα επίπεδα των δικτύων εργασίας $a_1, ..., a_{\lambda_\theta}$ διανέμονται σύμφωνα με την πιθανότητα $p(\alpha|\pi)$ και εκτελείται ο αλγόριθμος αντίστροφης διάδοσης σφάλματος για κάθε δείγμα. Οι κλίσεις που προκύπτουν $\nabla_\theta f(\theta, \alpha_i)$ μέσω της διαδικασίας αυτής, είναι ο μέσος όρος όλων των αντιστοιχίσεων, ώστε η τελική κλίση να περιγράφεται από την ακόλουθη εξίσωση:

$$\nabla_\theta J(\theta, \pi) \approx \frac{1}{\lambda_\theta} \sum_{i=1}^{\lambda_\theta} \nabla_\theta f(\theta, \alpha_i). \tag{1.4.6}$$

Η κλίση χρησιμοποιείται στη συνέχεια για την ενημέρωση των παραμέτρων θ με ρυθμό μάθησης η_θ, σύμφωνα με το ακόλουθο βήμα:

$$\theta - \eta_\theta \nabla_\pi J(\theta, \pi). \tag{1.4.7}$$

1.4.3 Μέθοδοι Υπολογισμού Σφάλματος κατά τη Μάθηση Πολλαπλών Εργασιών

Ένας από τους πιο σημαντικούς παράγοντες που πρέπει να ληφθούν υπόψη κατά το σχεδιασμό μιας αρχιτεκτονικής μάθησης πολλαπλών εργασιών είναι ο τρόπος με τον οποίο υπολογίζεται το σφάλμα της μάθησης πολλαπλών εργασιών. Ο όρος σφάλμα πολλαπλών εργασιών αναφέρεται στο συνολικό σφάλμα που παράγεται από το μοντέλο για όλες τις εργασίες, το οποίο χρησιμοποιείται για την ενημέρωση των παραμέτρων του μοντέλου για όλες τις εργασίες. Οι πιο συνηθισμένες μέθοδοι για τον υπολογισμό του σφάλματος πολλαπλών εργασιών είναι ο υπολογισμός του μέσου όρου των σφαλμάτων (average loss), η αβεβαιότητα στην στάθμιση των σφαλμάτων (uncertainty in weighing losses) και ο δυναμικός μέσος όρος των βαρών (dynamic weight average).

Μέσος Όρος των Σφαλμάτων

Το σφάλμα πολλαπλών εργασιών μπορεί να υπολογιστεί ως ο μέσος όρος των σφαλμάτων όλων των εργασιών. Αυτή η μέθοδος είναι η πιο απλή και πιο απευθείας για τον υπολογισμό του σφάλματος πολλαπλών εργασιών, καθώς θεωρεί όλες τις εργασίες ισότιμες. Ωστόσο, δεν λαμβάνει υπόψη τη δυσκολία κάθε εργασίας, γεγονός που μπορεί να οδηγήσει σε μη βέλτιστα αποτελέσματα. Η μαθηματική διατύπωση του υπολογισμού του μέσου των σφαλμάτων είναι:

$$\mathcal{L}_{MTL} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_i, \tag{1.4.8}$$

όπου το N είναι το πλήθος των εργασιών και το $\mathcal{L}_i$ είναι το σφάλμα της i-οστής εργασίας.

Ένας επιπλέον τρόπος για τον υπολογισμό του σφάλματος πολλαπλών εργασιών είναι η χρήση ενός βάρους για κάθε εργασία, που αντιπροσωπεύει τη σημασία της κάθε εργασίας στο συνολικό πρόβλημα. Αν και αυτή η μέθοδος λαμβάνει υπόψη τη δυσκολία κάθε εργασίας, οι βαθμοί σημασίας καθορίζονται εκ των προτέρων και παραμένουν σταθεροί κατά τη διάρκεια της εκπαίδευσης του μοντέλου, γεγονός που καθιστά αυτή την προσέγγιση αρκετά αυθαίρετη και τυχαία για την εύρεση της βέλτιστης λύσης.

Αβεβαιότητα στη Στάθμιση των Σφαλμάτων

Για να ληφθεί υπόψιν και η δυσκολία κάθε εργασίας, πρέπει να ανατεθούν βάρη σε κάθε εργασία, σταθμίζοντας με αυτό τον τρόπο τη βαρύτητα κάθε εργασίας στο συνολικό πρόβλημα. Σε αυτή τη μέθοδο η απόδοση του μοντέλου εξαρτάται σημαντικά από την επιλογή των βαρών για κάθε εργασία, ενώ η αναζήτηση των βέλτιστων βαρών είναι χρονοβόρα και απαιτεί πολύ υπολογιστικό χρόνο.

Ο A. Kendall και άλλοι στο [61] πρότειναν μια μέθοδο για τον υπολογισμό του σφάλματος πολλαπλών εργασιών, βασισμένη στην αβεβαιότητα της στάθμισης των εργασιών, που σημαίνει ότι η γνώση για το ποια εργασία είναι πιο σημαντική για την επίτευξη της βέλτιστης λύσης δεν είναι γνωστή εκ των προτέρων και πρέπει να μάθεται κατά τη διάρκεια της εκπαίδευσης. Συγκεκριμένα, η αβεβαιότητα στη στάθμιση των σφαλμάτων ορίζεται ως η ανάθεση ενός βάρους σε κάθε εργασία, που μαθαίνεται κατά τη διάρκεια της εκπαίδευσης, βασισμένο στη ελαχιστοποίηση της συνολική συνάρτησης σφάλματος. Αυτή η προσέγγιση εξαρτάται από τις εργασίες και όχι από τα δεδομένα, που χρησιμοποιούνται, δηλαδή προσπαθεί να βελτιστοποίηση μια ποσότητα που παραμένει σταθερή για όλα τα δεδομένα εισόδου και διαφέρει μεταξύ διαφορετικών εργασιών. Η μαθηματική διατύπωση του υπολογισμού του σφάλματος πολλαπλών εργασιών, βασισμένη στην αβεβαιότητα στη στάθμιση των σφαλμάτων, είναι:

$$\mathcal{L}_{MTL}(w,\alpha_1,\alpha_2) = \frac{1}{\alpha_1^2}\mathcal{L}_1(w) + \frac{1}{\alpha_2^2}\mathcal{L}_2(w) + \log(\alpha_1) + \log(\alpha_2) \ , \tag{1.4.9}$$

όπου το α_1 και το α_2 είναι τα βάρη αβεβαιότητας για τις δύο εργασίες, είναι θετικοί αριθμοί, που ερμηνεύονται ως τη θερμοκρασία σε μια κατανομή Boltzmann, όπου η κλίμακα μεγέθους αυτών των παραμέτρων καθορίζει πόσο ομοιόμορφη (επίπεδη) είναι η διακριτή κατανομή και το w είναι οι παράμετροι του μοντέλου.

Κατά τη υλοποίηση αυτής της μαθηματικής έκφρασης, το δίκτυο προβλέπει το λογάριθμο της διασποράς, $\sigma_i := \log(\alpha_i^2)$, για να είναι πιο αριθμητικά σταθερό από το να προβλέπει τη διασπορά, καθώς με αυτό τον τρόπο αποφεύγονται διαιρέσεις με το μηδέν. Παρόλο, που παρέχει έναν τρόπο για την επιλογή καλύτερων βαρών για τις εργασίες, κατά την εκπαίδευση, η μαθηματική υλοποίηση δεν είναι βέλτιστη, καθώς οδηγεί σε αρνητικές τιμές για τη διασπορά, που δεν είναι αποδεκτές στο πλαίσιο της εκτίμησης της αβεβαιότητας. Λύση σε αυτό το πρόβλημα προτείνεται στο [62] από τον Liebel και άλλους, όπου οι συγγραφείς τροποποιούν τον υπολογισμό του σφάλματος αβεβαιότητας για να αποφευχθούν αρνητικές τιμές για τη διασπορά, μετατρέποντας τον όρο $\log(\alpha_i^2)$ σε $\log(1+\alpha_i^2)$.

Συνεπώς, η νέα μαθηματική διατύπωση του υπολογισμού του σφάλματος αβεβαιότητας για δύο εργασίες, αναφερόμενη ως αυτόματη στάθμιση σφαλμάτων είναι:

$$\mathcal{L}_{MTL}(w,\alpha_1,\alpha_2) = \frac{1}{\alpha_1^2}\mathcal{L}_1(w) + \frac{1}{\alpha_2^2}\mathcal{L}_2(w) + \log(1+\alpha_1^2) + \log(1+\alpha_2^2) \ . \tag{1.4.10}$$

Δυναμικός Μέσος Όρος Βαρών

Ο δυναμικός μέσος όρος των βαρών [63] είναι μια μέθοδος για τον υπολογισμό του σφάλματος πολλαπλών εργασιών, που αναθέτει ένα βάρος σε κάθε εργασία, βασιζόμενο στο ρυθμό μεταβολής του σφάλματος για κάθε εργασία κατά τις προηγούμενες επαναλήψεις. Είναι εύκολο να υλοποιηθεί, καθώς απαιτεί μόνο τις αριθμητικές τιμές των σφαλμάτων κάθε εργασίας στις τρέχουσες και προηγούμενες επαναλήψεις. Η μαθηματική διατύπωση του υπολογισμού του δυναμικού μέσου όρου βαρών είναι:

$$\begin{aligned} \mathcal{L}_{MTL} &= \sum_{i=1}^{k} \lambda_k(t)\mathcal{L}_k, \\ \lambda_k(t) &= \frac{K \exp(w_k(t-1)/T)}{\sum_i \exp(w_i(t-1)/T)}, \text{ with } w_k(t-1) = \frac{\mathcal{L}_k(t-1)}{\mathcal{L}_k(t-2)} \ , \end{aligned} \tag{1.4.11}$$

όπου το $\mathcal{L}_k$ είναι το σφάλμα της k-οστής εργασίας, το T είναι ο όρος θερμοκρασίας που ελέγχει την ομοιομορφία στη στάθμιση των εργασιών, με μεγάλες τιμές να οδηγούν σε πιο ομοιόμορφη κατανομή μεταξύ των εργασιών, το $w_k(t-1)$ είναι το βάρος της k-οστής εργασίας στο χρονικό σημείο $t-1$ και το $\lambda_k(t)$ είναι το βάρος της k-οστής εργασίας στο χρονικό σημείο t.

Στην υλοποίηση, η τιμή του σφάλματος $\mathcal{L}_k(t)$ υπολογίζεται ως ο μέσος όρος από αρκετές επαναλήψεις, μειώνοντας με αυτό τον τρόπο την αβεβαιότητα κατά τον υπολογισμό της στοχαστικής κλίσης καθόδου, καθώς και την τυχαία επιλογή δεδομένων εκπαίδευσης. Για $t = 1, 2$ τα βάρη $w_k(t)$ αρχικοποιούνται στο 1, αλλά οποιαδήποτε άλλη αρχικοποίηση βασισμένη σε προηγούμενες γνώσεις εκπαίδευσης του δικτύου θα μπορούσε επίσης να εισαχθεί.

Στην πράξη, η μέθοδος του δυναμικού μέσου όρου βαρών θα αναθέσει μεγαλύτερα βάρη σε εργασίες με χαμηλότερο ρυθμό σφάλματος, πράγμα που σημαίνει ότι το μοντέλο θα εστιάσει περισσότερο σε εργασίες που βελτιώνονται πιο αργά, προκειμένου να επιτύχει καλύτερη συνολική απόδοση.

1.4.4 Προηγούμενες Εφαρμογές της Μάθησης Πολλαπλών Εργασιών

Η μάθηση πολλαπλών εργασιών έχει εφαρμοστεί σε προηγούμενες εργασίας για την αναγνώριση ανθρώπινων δράσεων και χειρονομιών, με σκοπό τη βελτίωση της απόδοσης των αλγορίθμων. Σε αυτή την ενότητα παρουσιάζονται μερικές από τις πιο σημαντικές εφαρμογές της μάθησης πολλαπλών εργασιών, που έχουν χρησιμοποιηθεί στην αναγνώριση ανθρώπινων δράσεων και χειρονομιών.

Αναγνώριση Ανθρώπινων Δράσεων

Ένας τρόπος για να επωφεληθούμε από τη μάθηση πολλαπλών εργασιών είναι να εκπαιδεύσουμε ένα μοντέλο σε διαφορετικά σύνολα δεδομένων, που έχουν τον ίδιο τύπο δεδομένων. Οι Karen Simonyan και Andrew Zisserman στο άρθρο τους [38] πρότειναν τη δημοφιλή αρχιτεκτονική Two-Stream Convolutional Neural Network (TSN), την οποία και αξιολόγησαν σε ένα πρόβλημα αναγνώρισης ανθρώπινων δράσεων. Αυτό το μοντέλο απαιτούσε μεγάλο όγκο δεδομένων εκπαίδευσης για να επιτύχει καλή απόδοση, αλλά οι βάσεις δεδομένων της εποχής δεν ήταν αρκετά μεγάλες. Για να αντιμετωπίστει αυτό το ζήτημα, πρότειναν μια προσέγγιση μάθησης πολλαπλών εργασιών, ώστε το μοντέλο να μπορεί να εκπαιδευτεί σε δεδομένα από διαφορετικά σύνολα δεδομένων.

Συγκεκριμένα, το δίκτυο εκπαιδεύτηκε σε δύο σύνολα δεδομένων, τα UCF-101 και HMDB-51, τα οποία αποτελούνταν από βίντεο δράσεων, με κάποιες από αυτές να υπάρχουν και στα δύο σύνολα. Έτσι, μια απλή συνένωση των δύο συνόλων δεδομένων δεν ήταν εφικτή, λόγω του ότι υπήρχαν κατηγορίες δράσεων που επικαλύπτονταν. Αντί για αυτό, πρότειναν μια προσέγγιση μάθησης πολλαπλών εργασιών, όπου το μοντέλο μοιράζεται τις παραμέτρους των πρώτων επιπέδων του δικτύου, αλλά έχει ξεχωριστά τα τελευταία επίπεδα, για να αντιστοιχούν στα διαφορετικά σύνολα δεδομένων. Κάθε ένα από τα διαφορετικά επίπεδα έχει τη δική του συνάρτηση σφάλματος, που λειτουργεί μόνο στα βίντεο που προέρχονται από το αντίστοιχο σύνολο δεδομένων. Το συνολικό σφάλμα εκπαίδευσης υπολογίζεται ως άθροισμα των σφαλμάτων των διαφορετικών εργασιών και οι παράγωγοι των βαρών του δικτύου μπορούν να υπολογιστούν με τη μέθοδο της αντίστροφης διάδοσης (back-propagation). Τα αποτελέσματα έδειξαν ότι το δίκτυο μπορούσε να γενικεύσει καλύτερα σε καινούρια δεδομένα, σε σύγκριση με την περίπτωση που εκπαιδευόταν σε κάθε σύνολο δεδομένων ξεχωριστά, αποδεικνύοντας τα

οφέλη της μάθησης πολλαπλών εργασιών στην αναγνώριση ανθρώπινων δράσεων με την ίδια μορφή δεδομένων, αλλά με διαφορετικά σύνολα δεδομένων.

Ένας ακόμη τρόπος για να επωφεληθούμε από τη μάθηση πολλαπλών εργασιών είναι να εκπαιδεύσουμε ένα μοντέλο να εκτελεί διαφορετικές εργασίες ή να χρησιμοποιεί διαφορετικές μεθόδους. Στο άρθρο των [64], οι συγγραφείς πρότειναν ένα μοντέλο μάθησης πολλαπλών εργασιών για την εκτίμηση 2D ή 3D ανθρώπινων στάσεων και την ταξινόμηση δράσεων από ακολουθίες βίντεο. Σκοπός αυτής της μεθόδου είναι να αντιμετωπίσει την εκτίμηση στάσεων και την αναγνώριση δράσεων ως ένα κοινό πρόβλημα, ενώ παράλληλα χρησιμοποιεί τις εκτιμημένες πόζες στην αναγνώριση δράσεων, και επωφελείται κατά τη διάρκεια της εκπαίδευσης από το διαμοιρασμό των βαρών μεταξύ των δύο εργασιών.

Αναγνώριση Ανθρώπινων Χειρονομιών

Στην αναγνώριση χειρονομιών, η μάθηση πολλαπλών εργασιών έχει χρησιμοποιηθεί για την από κοινού εκπαίδευση διαφορετικών εργασιών. Συγκεκριμένα, στο άρθρο [65], οι συγγραφείς πρότειναν ένα μοντέλο μάθησης πολλαπλών εργασιών για την από κοινού διαχείριση της αναγνώρισης χειρονομιών (gesture recognition) και της τμηματοποίησης τοποθεσιών χειρονομιών (gesture segmentation). Ένα συνελκτικό νευρωνικό δίκτυο εκπαιδεύτηκε στην τμηματοποίηση τοποθεσιών χειρονομιών με βάση τα δεδομένα βάθους, που χρησιμοποιήθηκαν κατά την εκπαίδευση του μοντέλου, ενώ κατά τη διάρκεια της αξιολόγησης χρησιμοποιήθηκαν μόνο τα δεδομένα RGB. Το μοντέλο επέτυχε καλύτερα αποτελέσματα από τις τότε καλύτερες μεθόδους, αποδεικνύοντας τα οφέλη της μάθησης πολλαπλών εργασιών στην αναγνώριση χειρονομιών.

Συνδυασμός με τη Μεταφορά Γνώσης

Κατά τη μάθηση πολλαπλών εργασιών, οι εργασίες μαθαίνονται παράλληλα κατά τη διάρκεια της εκπαίδευσης, απαιτώντας τις παραμέτρους του μοντέλου να είναι εκπαιδεύσιμες κατά τη διάρκεια ολόκληρης της διαδικασίας εκπαίδευσης. Από την άλλη, στη μεταφορά γνώσης, ένα μοντέλο εκπαιδεύεται πρώτα σε μια εργασία και στη συνέχεια γίνεται επαναδιαμόρφωση σε μια δεύτερη εργασία, απαιτώντας κάποιες από τις παραμέτρους του μοντέλου να μην είναι εκπαιδεύσιμες κατά τη διάρκεια της εκπαίδευσης της δεύτερης εργασίας, για να κρατηθεί η πληροφορία, που μαθεύτηκε από την προηγούμενη εκπαίδευση και να μειωθεί ο χρόνος εκπαίδευσης της δεύτερης εργασίας.

Στην εργασία [66], οι συγγραφείς πρότειναν ένα μοντέλο μάθησης πολλαπλών εργασιών, που συνδυάζει τη μάθηση πολλαπλών εργασιών με τη μεταφορά γνώσης, για να βελτιώσει την απόδοση του μοντέλου. Συγκεκριμένα, το μοντέλο εκπαιδεύεται σε δύο φάσεις. Στην πρώτη φάση εκπαιδεύεται σε πολλαπλές εργασίες ταυτόχρονα, σύμφωνα με τη θεωρία της μάθησης πολλαπλών εργασιών, και στη συνέχεια αυτό το μοντέλο χρησιμοποιείται ως προ-εκπαιδευμένο μοντέλο για την εκπαίδευση μιας μόνο εργασίας, που χρησιμοποιήθηκε και στην πρώτη φάση, ακολουθώντας έτσι μια προσέγγιση μεταφοράς γνώσης. Τα αποτελέσματα έδειξαν ότι η προτεινόμενη μέθοδος επιτυγχάνει καλύτερη απόδοση από την εκπαίδευση μιας μόνο εργασίας, καθώς και από την εκπαίδευση πολλαπλών εργασιών, για τις εργασίες.

1.4.5 Προτεινόμενη Μεθοδολογία

Αυτή η διπλωματική εργασία εξετάζει την αποτελεσματικότητα της μάθησης πολλαπλών εργασιών στον συνδυασμό των εργασιών αναγνώρισης ανθρώπινων δράσεων και χειρονομιών. Καθώς η απάντηση στο ερώτημα εάν η μάθηση πολλαπλών εργασιών μπορεί να χρησιμοποιηθεί για την κοινή εκπαίδευση δύο εργασιών δεν είναι δυαδική, διάφορες αρχιτεκτονικές μάθησης πολλαπλών εργασιών υλοποιούνται και δοκιμάζονται, για να αξιολογηθεί η απόδοσή τους. Συγκεκριμένα, οι τρεις διαφορετικές αρχιτεκτονικές μάθησης πολλαπλών εργασιών, που υλοποιήθηκαν, ακολουθούν διαφορετικές μεθόδους διαμοιρασμού των παραμέτρων, και αυτές είναι ο ολικός διαμοιρασμός παραμέτρων, ο μερικός διαμοιρασμός παραμέτρων και μια πιο πολύπλοκη μέθοδος, που στοχεύει να εκμεταλλευτεί τα πλεονεκτήματα και των δύο μεθόδων, αντίστοιχα. Η προτεινόμενη προσέγγιση μάθησης πολλαπλών εργασιών εμπίπτει στην κατηγορία της μάθησης πολλαπλών εργασιών διαφορετικών συνόλων δεδομένων, καθώς οι δύο εργασίες εκπαιδεύονται σε διαφορετικά σύνολα δεδομένων. Επίσης, μπορεί να θεωρηθεί ως μία προσέγγιση μάθησης πολλαπλών εργασιών ομοιόμορφης μάθησης, καθώς οι δύο εργασίες είναι του ίδιου τύπου, ταξινόμησης, και συγκεκριμένα είναι εργασίες αναγνώρισης. Τέλος, η προτεινόμενη προσέγγιση μπορεί να κατηγοριοποιηθεί ως προσέγγιση μάθησης πολλαπλών εργασιών ομοιόμορφης μάθησης χαρακτηριστικών, καθώς οι δύο εργασίες χρησιμοποιούν δεδομένα RGB ως είσοδο.

Επιπλέον, η εξεταζόμενη ιδέα επεκτείνεται για να αξιολογήσει την επίδοση των μεθόδων πολλαπλής μάθησης όταν διαχειρίζονται δεδομένα διαφορετικού τύπου. Συγκεκριμένα, οι τρεις διαφορετικές αρχιτεκτονικές μάθησης πολλαπλών εργασιών, που υλοποιήθηκαν, αξιολογούνται στην αναγνώριση χειρονομιών, χρησιμοποιώντας δεδομένα βίντεο RGB και βίντεο βάθους. Αυτή η εναλλακτική προσέγγιση εμπίπτει στην κατηγορία της μάθησης πολλαπλών εργασιών διαφορετικών τύπων δεδομένων, καθώς οι δύο εργασίες εκπαιδεύονται σε διαφορετικούς τύπους δεδομένων, RGB και βάθους. Επιπλέον, μπορεί να θεωρηθεί ως μία προσέγγιση μάθησης πολλαπλών εργασιών ομοιόμορφης μάθησης, καθώς οι δύο εργασίες είναι του ίδιου τύπου, ταξινόμησης, και συγκεκριμένα είναι εργασίες αναγνώρισης. Τέλος, η προτεινόμενη προσέγγιση μπορεί να κατηγοριοποιηθεί ως προσέγγιση μάθησης πολλαπλών εργασιών ίδιου συνόλου δεδομένων, μιας και τα δεδομένα RGB και βάθους ανήκουν στο ίδιο σύνολο δεδομένων.

Επεξεργασία Βίντεο

Στην όραση υπολογιστών τα μοντέλα μάθησης βαθιάς μάθησης δέχονται στην είσοδό τους σταθερού μεγέθους δεδομένα, ενώ τα βίντεο μπορεί να έχουν μεταβλητό μήκος. Έτσι, τα βίντεο πρέπει να προεπεξεργαστούν ώστε να έχουν σταθερό μήκος. Ορισμένες από τις πιο συνηθισμένες μεθόδους προεπεξεργασίας βίντεο είναι να διαιρεθεί κάθε βίντεο σε κλιπ με σταθερό πλήθος εικόνων ή να δειγματοληπτηθεί ένας σταθερός αριθμός εικόνων από κάθε βίντεο.

- **Στη Διαίρεση Βίντεο σε Κλιπ**
  κάθε βίντεο διαιρείται σε κλιπ με σταθερό μήκος. Ένα στάνταρ μήκος κλιπ F επιλέγεται και στη συνέχεια το πλήθος των κλιπ ανά βίντεο υπολογίζεται διαιρώντας το συνολικό αριθμό εικόνων του βίντεο με το μήκος του κλιπ και στρογγυλοποιώντας προς το μεγαλύτερο ακέραιο. Στη συνέχεια τα κλιπ δημιουργούνται παίρνοντας τις πρώτες F εικόνες του βίντεο, στη συνέχεια τις επόμενες F εικόνες και ούτω καθεξής, μέχρι το τέλος του βίντεο. Το βήμα μεταξύ της αρχής ενός κλιπ και της αρχής του επόμενου κλιπ είναι δυνατό να μην είναι ίσο με το μήκος του κλιπ, οπότε δημιουργούνται επικαλυπτόμενα κλιπ. Ακολουθώντας αυτή τη μέθοδο δημιουργούνται πολλά δείγματα από κάθε βίντεο, τα οποία χρησιμοποιούνται ως δείγματα για το μοντέλο, οδηγώντας σε δύο μεθόδους αξιολόγησης της απόδοσης του μοντέλου, ανά βίντεο και ανά κλιπ.

  1. **Κατά την αξιολόγηση Ανά Βίντεο** το μοντέλο εκπαιδεύεται και αξιολογείται στα κλιπ κάθε βίντεο ξεχωριστά. Η πρόβλεψη του μοντέλου για κάθε βίντεο εξάγεται με τον μέσο όρο των προβλέψεων των κλιπ του.
  2. **Κατά την αξιολόγηση Ανά Κλιπ** το μοντέλο εκπαιδεύεται και αξιολογείται σε κάθε κλιπ ξεχωριστά. Η πρόβλεψη του μοντέλου για κάθε κλιπ ορίζει τη συνολική του απόδοση.

  Αυτή η μέθοδος έχει το πλεονέκτημα της χρήσης ολόκληρου του βίντεο ως εισόδου στο μοντέλο, αλλά έχει το μειονέκτημα της δημιουργίας μεγάλου αριθμού δειγμάτων, που μπορεί να οδηγήσει σε υπολογιστική ανεπάρκεια.

- **Στην Επιλογή Δείγματος Εικόνων**
  ένας σταθερός αριθμός εικόνων N δειγματοληπτείται από κάθε βίντεο. Αν το βίντεο έχει λιγότερες από N εικόνες, τότε το βίντεο συμπληρώνεται με την τελευταία εικόνα του βίντεο, δημιουργώντας ένα στατικό βίντεο μετά το βίντεο. Αυτή η μέθοδος έχει το πλεονέκτημα της δημιουργίας ενός σταθερού αριθμού δειγμάτων από κάθε βίντεο, αλλά έχει το μειονέκτημα της απώλειας πληροφορίας από το βίντεο.

Σε αυτή τη διπλωματική εργασία, η μέθοδος δειγματοληψίας εικόνων χρησιμοποιείται για την προεπεξεργασία των βίντεο, λόγω της υπολογιστικής απλότητας της μεθόδου και της επιτυχίας της σε προηγούμενες εργασίες χωρίς να επηρεάζει την απόδοση του μοντέλου. Το πλήθος των εικόνων N επιλέγεται να είναι 32, το οποίο είναι ένα συνηθισμένο πλήθος εικόνων που χρησιμοποιείται σε εργασίες ταξινόμησης βίντεο.

Βάση Δεδομένων Μάθησης Πολλαπλών Εργασιών (Multi-task Dataset)

Η βάση δεδομένων μάθησης πολλαπλών εργασίων, που χρησιμοποιείται σε αυτή τη διπλωματική εργασία, ακολουθεί την ίδια δομή με τη βάση δεδομένων μάθησης πολλαπλών εργασιών, που χρησιμοποιήθηκε στην εργασία [60]. Συγκεκριμένα, οι J. Prellberg και O. Kramer διαχωρίζουν τη βάση δεδομένων σε υποσύνολα, ώστε να δημιουργήσουν σύνολα ειδικά για κάθε εργασία και στη συνέχεια συνδυάζουν αυτά τα σύνολα σε ένα σύνολο δεδομένων, όπου κάθε δείγμα του συνόλου δεδομένων είναι ένα λεξικό, με κλειδιά που αντιπροσωπεύουν τις εργασίες και

τιμές που αντιπροσωπεύουν τα δείγματα των εργασιών, όπως φαίνεται για ένα παράδειγμα δύο εργασιών στο Σχήμα 1.4.6.

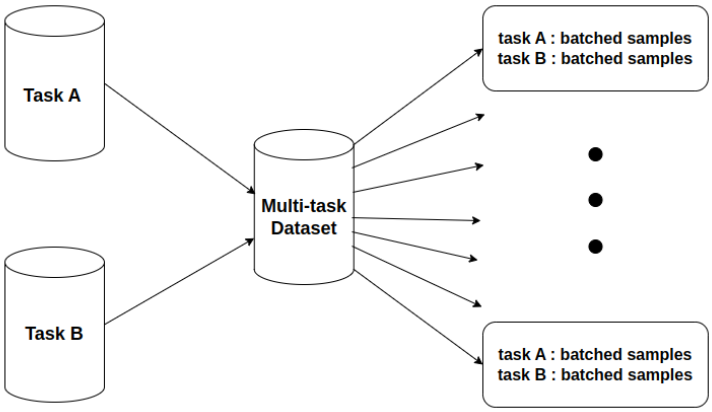


Figure 1.4.6: Βάση Δεδομένων Μάθησης Πολλαπλών Εργασιών

Νευρωνικά Δίκτυα Μάθησης Πολλαπλών Εργασιών

Οι μέθοδοι διαμοιρασμού των παραμέτρων, που χρησιμοποιούνται για την υλοποίηση μοντέλων μάθησης πολλαπλών εργασιών, μπορούν να εφαρμοστούν σε οποιαδήποτε αρχιτεκτονική βαθιάς μάθησης. Σε αυτή τη διπλωματική εργασία, το μοντέλο που χρησιμοποιείται ως βασικό μοντέλο (backbone) για την υλοποίηση των μοντέλων μάθησης πολλαπλών εργασιών είναι το ResNet-18 3D. Η αρχιτεκτονική ResNet-18 3D επιλέχθηκε γιατί είναι μια αρχιτεκτονική που χρησιμοποιείται ευρέως για εργασίες ταξινόμησης βίντεο και υπάρχουν πολλές παραλλαγές της που μπορούν να χρησιμοποιηθούν για καλύτερη προσαρμογή των δεδομένων. Επιπλέον, η αρχιτεκτονική ResNet-18 3D έχει δημόσια διαθέσιμα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400, τα οποία μπορούν να χρησιμοποιηθούν για καλύτερη απόδοση στις εργασίες.

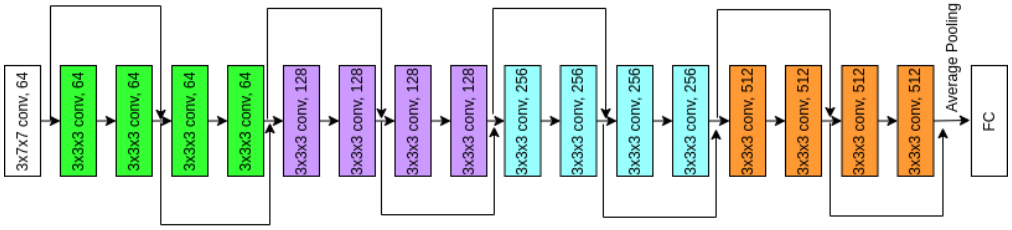


Figure 1.4.7: ResNet-18 3D αρχιτεκτονική

Το ResNet-18 3D δέχεται στην είσοδό του έναν 5D τανυστή (τένσορα - tensor), που αντιπροσωπεύει έναν 4D όγκο ή ένα βίντεο. Οι διαστάσεις περιλαμβάνουν το μέγεθος του batch, τον αριθμό των καναλιών, το πλήθος των εικόνων, το ύψος και το πλάτος των δεδομένων εισόδου. Όπως απεικονίζεται στο Σχήμα 1.4.7, το μοντέλο ResNet-18 3D είναι ένα συνελκτικό νευρωνικό δίκτυο, που αποτελείται από 18 επίπεδα, συμπεριλαμβανομένων 17 συνελκτικών επιπέδων και 1 πλήρως συνδεδεμένο επίπεδο. Κάθε ομάδα χρωματισμένων συνελκτικών επιπέδων στο σχήμα αντιπροσωπεύει ένα ResNet-Layer, το οποίο είναι ένα σύνολο Residual δομικών στοιχείων. Συνεπώς, κάθε Residual Block είναι ένα σύνολο συνελκτικών επιπέδων, που συνδέονται με μια σύνδεση παράκαμψης (skip connection). Κάθε συνελκτικό επίπεδο ακολουθείται από ένα επίπεδο κανονικοποίησης πακέτων (batch normalization) και μια συνάρτηση ενεργοποίησης ReLU, τα οποία δεν απεικονίζονται στο σχήμα για απλότητα. Οι συνδέσεις παράκαμψης χρησιμοποιούνται για να προστεθεί η είσοδος ενός επιπέδου στην έξοδο ενός επιπέδου πιο κάτω στο δίκτυο.

Επομένως, αυτή η αρχιτεκτονική τροποποιείται κατάλληλα με βάση τις διαφορετικές μεθόδους διαμοιρασμού των παραμέτρων για να δημιουργηθούν τα αντίστοιχα μοντέλα μάθησης πολλαπλών εργασιών. Σε όλες τις

παραλλαγές της μάθησης πολλαπλών εργασιών, τα πρώτα επίπεδα χρησιμοποιούν προ-εκπαιδευμένα βάρη από το σύνολο δεδομένων Kinetics-400, ενώ τα τελευταία επίπεδα εκπαιδεύονται από την αρχή, όπως αναπαρίσταται με κόκκινο χρώμα και πράσινο χρώμα αντίστοιχα. Αυτό γίνεται για να εκμεταλλευτούμε τα πλεονεκτήματα της μεταφοράς γνώσης, ενώ επιτρέπεται στο μοντέλο να επωφεληθεί από τη μάθηση πολλαπλών εργασιών.

Ολικός Διαμοιρασμός Παραμέτρων (HPS) Για την υλοποίηση της μεθόδου Ολικού Διαμοιρασμού Παραμέτρων (HPS), το μοντέλο ResNet-18 3D διαμοιράζει όλα τα κρυφά επίπεδά του σε όλες τις εργασίες, εκτός από το τελευταίο επίπεδο, το οποίο είναι διαφορετικό για κάθε εργασία, όπως φαίνεται στο Σχήμα 1.4.8.

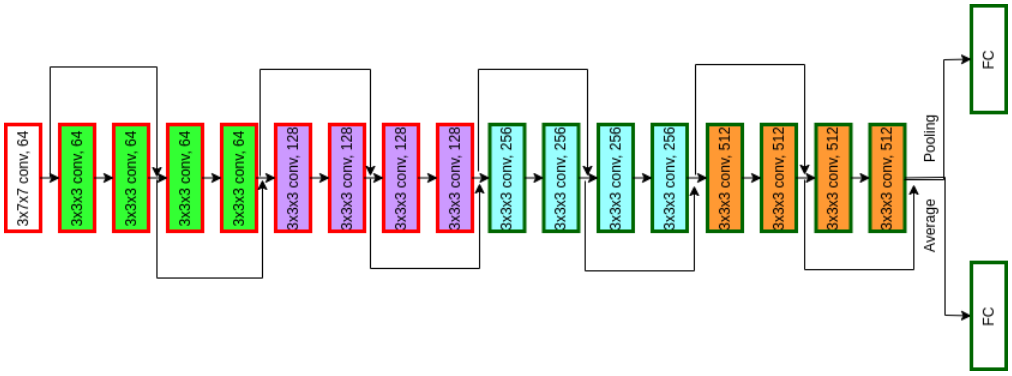


Figure 1.4.8: Ολικός Διαμοιρασμός Παραμέτρων (HPS) ResNet-18 3D

Μερικός Διαμοιρασμός Παραμέτρων (SPS) Για την υλοποίηση της μεθόδου Μερικού Διαμοιρασμού Παραμέτρων (SPS), δημιουργούνται τόσα δίκτυα ResNet 3D όσες είναι και οι εργασίες, τα οποία συνδέονται με μονάδες cross-stitch, όπως φαίνεται στο Σχήμα 1.4.9. Έτσι, τα δίκτυα μπορούν να μοιράζονται πληροφορίες μεταξύ τους, ενώ ταυτόχρονα εκπαιδεύονται ανεξάρτητα.

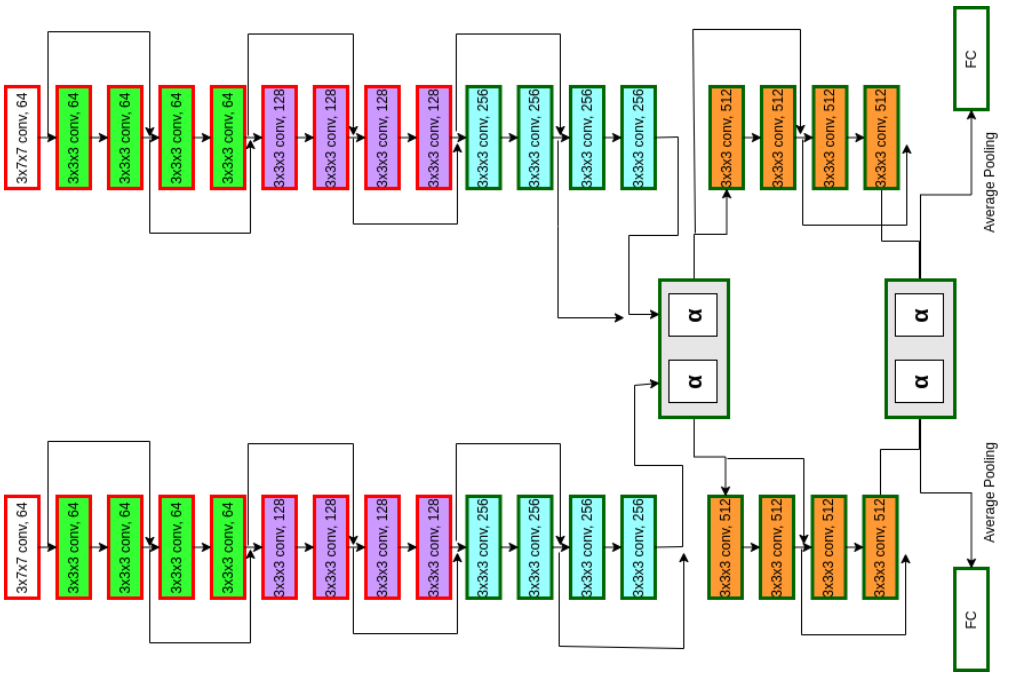


Figure 1.4.9: Μερικός Διαμοιρασμός Παραμέτρων (SPS) ResNet-18 3D

Εκμάθηση Διαμοιρασμού Βαρών (LWS) Για την υλοποίηση της μεθόδου Εκμάθησης Διαμοιρασμού Βαρών (LWS), δημιουργούνται διαφορετικά δίκτυα ResNet-18 3D για κάθε εργασία, τα οποία μοιράζονται κάποια

βάρη ανά επίπεδα μεταξύ τους, όπως φαίνεται στο Σχήμα 1.4.10. Με αυτό τον τρόπο τα δίκτυα μπορούν να βρουν την καλύτερη ανάθεση για τα βάρη τους, ενώ ταυτόχρονα να βελτιστοποίησουν και τις τιμές των βαρών τους με την εναλλαγή των βελτιστοποιητών NES και SGD, όπως αναλύθηκε προηγουμένως.

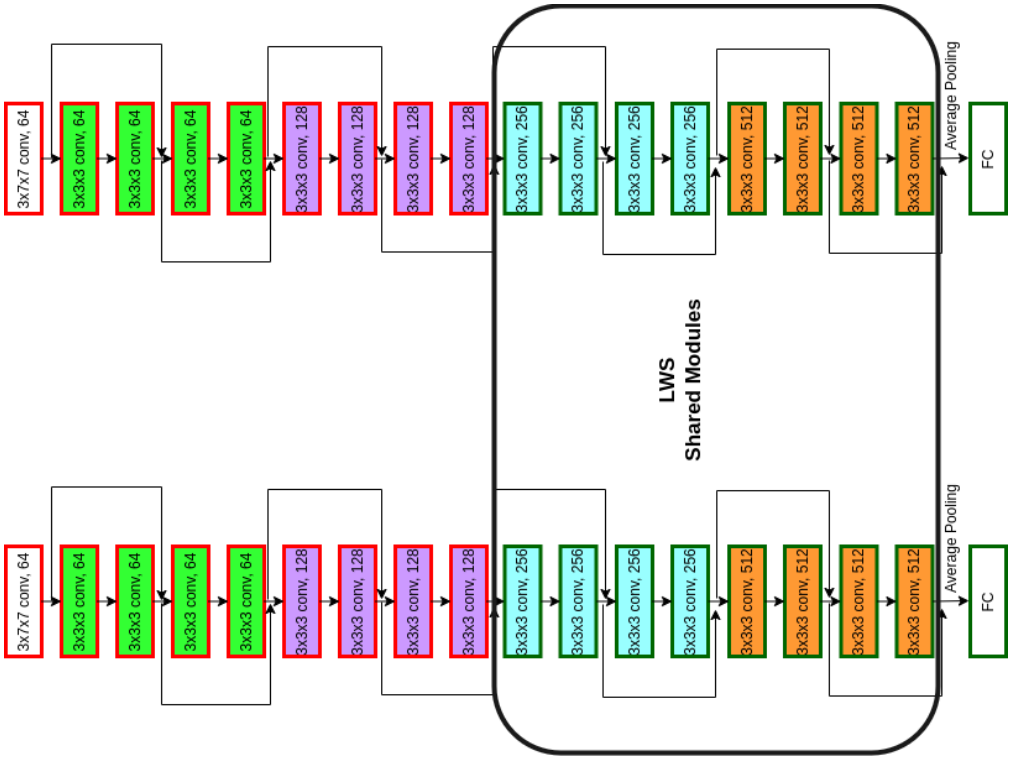


Figure 1.4.10: Εκμάθηση Διαμοιρασμού Βαρών (LWS) ResNet-18 3D

Εκπαίδευση Μοντέλων Μάθησης Πολλαπλών Εργασιών

Μια βάση δεδομένων για την εκπαίδευση ενός μοντέλου μιας εργασίας αποτελείται από πακέτα (batch) δεδομένων, τα οποία περνούν από το μοντέλο. Η διαδικασία εκπαίδευσης αυτού του μοντέλου θα περιλαμβάνει την ενημέρωση των βαρών του μοντέλου με βάση το πρόβλημα βελτιστοποίησης του μοντέλου, που είναι η ελαχιστοποίηση της απώλειας μεταξύ των προβλέψεων του μοντέλου και των πραγματικών τιμών των δεδομένων. Αυτή η διαδικασία όπου όλα τα πακέτα δειγμάτων της βάσης δεδομένων περνούν από το μοντέλο μια φορά ονομάζεται εποχή. Αυτή η διαδικασία επαναλαμβάνεται για έναν αριθμό εποχών, μέχρι το μοντέλο να συγκλίνει στη βέλτιστη λύση.

Μια βάση δεδομένων για την εκπαίδευση ενός μοντέλου μάθησης πολλαπλών εργασιών αποτελείται από πακέτα δεδομένων από διαφορετικά σύνολα δεδομένων, τα οποία αντιστοιχούν στις διαφορετικές εργασίες. Αυτά τα σύνολα δεδομένων μπορεί να αποτελούνται από διαφορετικό αριθμό δειγμάτων. Συνεπώς, αν ακολουθηθεί η διαδικασία εκπαίδευσης, που χρησιμοποιείται για τις μονές εργασίες, το μοντέλο θα εκπαιδευτεί σε διαφορετικό αριθμό δειγμάτων από κάθε εργασία, που μπορεί να οδηγήσει σε λανθασμένα αποτελέσματα στην αξιολόγηση του μοντέλου. Αυτό συμβαίνει γιατί τα μοντέλα μάθησης πολλαπλών εργασιών θα συγκριθούν με τα μοντέλα μάθησης μιας εργασίας. Αν η διαδικασία εκπαίδευσης που ακολουθείται είναι με εποχές, το μοντέλο θα διανύσει όλα τα δείγματα του μεγαλύτερου συνόλου δεδομένων μια φορά σε κάθε εποχή, ενώ για το μικρότερο σύνολο δεδομένων κάποια δείγματα θα διανυθούν περισσότερες φορές στην ίδια εποχή. Έτσι, αν θέλουμε να συγκρίνουμε την απόδοση των μοντέλων μάθησης πολλαπλών εργασιών με τα μοντέλα μάθησης μιας εργασίας, αυτή η προσέγγιση μπορεί να οδηγήσει σε λανθασμένα αποτελέσματα, δίνοντας την εντύπωση ότι τα μοντέλα μάθησης πολλαπλών εργασιών είναι καλύτερα από τα μοντέλα μάθησης μιας εργασίας, αλλά η αλήθεια θα είναι ότι τα μοντέλα θα έχουν δει περισσότερα δείγματα στην ίδια εποχή.

Για να αποφευχθεί αυτό το πρόβλημα, εφαρμόζεται μια εναλλακτική διαδικασία εκπαίδευσης, τόσο για τις μεθόδους μάθησης πολλαπλών εργασιών όσο και για τις μεθόδους μάθησης μιας εργασίας, όπου το μοντέλο εκπαιδεύεται για έναν αριθμό επαναλήψεων, αντί για εποχές. Σε κάθε επανάληψη, ένα πακέτο δεδομένων εξάγεται από τη

βάση δεδομένων μάθησης πολλαπλών εργασιών και δίνεται στο μοντέλο. Τα βάρη του μοντέλου ενημερώνονται για κάθε πακέτο, σύμφωνα με το πρόβλημα βελτιστοποίησης του μοντέλου. Αυτή η διαδικασία επαναλαμβάνεται για έναν αριθμό επαναλήψεων, μέχρι το μοντέλο να συγκλίνει σε μια βέλτιστη λύση. Αυτή η διαδικασία εκπαίδευσης επιτρέπει στο μοντέλο να δει το ίδιο πλήθος δειγμάτων σε κάθε επανάληψη, ώστε να μπορούμε να συγκρίνουμε την απόδοση των μεθόδων μάθησης πολλαπλών εργασιών με τις μεθόδους μάθησης μιας εργασίας, καθώς το μοντέλο θα έχει δει το ίδιο πλήθος δειγμάτων στον ίδιο αριθμό επαναλήψεων.

1.5 Πειράματα

1.5.1 Αποτελέσματα Πειραμάτων Μονής Εργασίας

Για τα πειράματα που διεξήχθησαν με σκοπό να αποδειχθεί η αποτελεσματικότητα των προτεινόμενων ιδεών, η απόδοση κάθε μοντέλου μάθησης πολλαπλών εργασιών (MTL) θα συγκριθεί με τα μοντέλα μάθησης μονής εργασίας (STL), που εκπαιδεύονται ξεχωριστά σε κάθε εργασία. Για την εργασία της αναγνώρισης δράσεων χρησιμοποιούνται δύο σύνολα δεδομένων, το UCF-101 και το NTU-RGB+D, ενώ για την εργασία της αναγνώρισης χειρονομιών χρησιμοποιούνται επίσης δύο σύνολα δεδομένων, το IsoGD και το NVGesture. Η απόδοση των μοντέλων θα αξιολογηθεί χρησιμοποιώντας τη μετρική της ακρίβειας, δηλαδή με το ποσοστό των σωστά ταξινομημένων δειγμάτων. Τόσο τα μοντέλα μάθησης μονής εργασίας όσο και τα μοντέλα μάθησης πολλαπλών εργασιών θα εκπαιδευτούν για 20.000 επαναλήψεις, για να υπάρχει μια κοινή βάση σύγκριση μεταξύ τους.

Το βασικό δίκτυο που επιλέγεται να χρησιμοποιηθεί στα πειράματα STL και MTL είναι η αρχιτεκτονική ResNet-18 3D. Η αρχιτεκτονική ResNet-18 3D επιλέγεται διότι είναι μια ευρέως χρησιμοποιούμενη αρχιτεκτονική στην ταξινόμηση βίντεο και υπάρχουν πολλές παραλλαγές της, που μπορούν να χρησιμοποιηθούν για καλύτερη προσαρμογή των δεδομένων. Επιπλέον, η αρχιτεκτονική ResNet-18 3D έχει διαθέσιμα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400, τα οποία μπορούν να χρησιμοποιηθούν για καλύτερη απόδοση στις εργασίες.

Συγκεκριμένα, αξιολογείται η απόδοση της αρχιτεκτονικής ResNet-18 3D στα σύνολα δεδομένων UCF-101, NTU-RGB+D, IsoGD και NVGesture για διαφορετική αρχικοποίηση βαρών για τη χρήση προ-εκπαιδευμένων βαρών στο σύνολο δεδομένων Kinetics-400, όπου διαφορετικός αριθμός παραμέτρων αφήνεται εκπαιδεύσιμος κατά τη διάρκεια της εκπαίδευσης. Κάθε πίνακας περιγράφει τον αριθμό των επιπέδων που εκπαιδεύονται της κάθε μοντέλου, τις εκπαιδεύσιμες παραμέτρους και την ακρίβεια του μοντέλου στο αντίστοιχο σύνολο δεδομένων. Στη στήλη εκπαιδεύσιμων επιπέδων, αναφερόμαστε στον αριθμό των επιπέδων Residual που αφήνονται εκπαιδεύσιμα, οπότε το layer 1 αναφέρεται στο πρώτο επίπεδο Residual, το οποίο περιλαμβάνει 4 συνελικτικά επίπεδα όπως εξηγήθηκε προηγουμένως, και έτσι συνεχίζεται το πρότυπο.

ResNet στη βάση UCF-101

Στον πίνακα που ακολουθεί, παρουσιάζεται η απόδοση της αρχιτεκτονικής ResNet-18 3D στο σύνολο δεδομένων UCF-101 για την πρώτη διαίρεση του συνόλου δεδομένων (split-1). Από τα αποτελέσματα, παρατηρείται ότι η καλύτερη απόδοση επιτυγχάνεται όταν τα βάρη αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400. Συγκεκριμένα, η ακρίβεια του μοντέλου αυξάνεται όσο περισσότερα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά. Η καλύτερη απόδοση επιτυγχάνεται όταν όλα τα επίπεδα εκτός από το τελευταίο πλήρως συνδεδεμένο επίπεδο αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά, ώστε να μην ενημερώνονται κατά τη διάρκεια της εκπαίδευσης. Αυτή η προσέγγιση επιτυγχάνει μια υψηλή ακρίβεια του 85.33% στην πρώτη διαίρεση του συνόλου δεδομένων UCF-101, λόγω του γεγονότος ότι τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400 είναι κοντά στα βέλτιστα βάρη για το σύνολο δεδομένων UCF-101.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33,218,085	23.82
Kinetics-400	ALL	33,218,085	24.12
Kinetics-400	layers 1, 2, 3, 4 + fc	33,189,733	32.45
Kinetics-400	layers 2, 3, 4 + fc	32,746,853	41.27
Kinetics-400	**layers 3, 4 + fc**	**31,189,093**	**58.26**
Kinetics-400	layers 4 + fc	24,960,613	77.32
Kinetics-400	fc	51,813	85.33

Table 1.2: Αποτελέσματα προεκπαιδευμένου ResNet-18 στη βάση UCF-101.

ResNet στη βάση NTU-RGB+D

Από την εκπαίδευση του μοντέλο ResNet-18 3D στο σύνολο δεδομένων NTU-RGB+D φαίνεται ότι η αρχικοποίηση των βαρών με τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400 επωφελεί την απόδοση του μοντέλου, όπως εξακριβώνεται και από τα αποτελέσματα του παρακάτω πίνακα. Τα δύο σύνολα δεδομένων

είναι σε συγκρίσιμο μέγεθος και τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400, αν και κοντά στα βέλτιστα βάρη για το σύνολο δεδομένων NTU-RGB+D, δεν είναι βέλτιστα. Έτσι, αυτή η αρχιτεκτονική χρειάζεται περισσότερα εκπαιδεύσιμα επίπεδα για να προσαρμοστεί στο σύνολο δεδομένων NTU-RGB+D. Η καλύτερη απόδοση επιτυγχάνεται όταν μόνο το τελευταίο επίπεδο Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά. Αυτή η προσέγγιση επιτυγχάνει μια ακρίβεια του 56.68% στη διαίρεση του συνόλου υποκειμένων NTU-RGB+D.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33,227,832	12.76
Kinetics-400	ALL	33,227,832	12.91
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	22.78
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	48.18
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**55.29**
Kinetics-400	layers 4 + fc	24,970,360	56.68
Kinetics-400	fc	61,560	23.45

Table 1.3: Αποτελέσματα προεκπαιδευμένου ResNet-18 στη βάση NTU-RGB+D για RGB δεδομένα.

ResNet στη βάση IsoGD

Στους ακόλουθους πίνακες αναγράφονται τα αποτελέσματα της εκπαίδευσης του μοντέλου ResNet-18 3D στο σύνολο δεδομένων IsoGD, για δεδομένα RGB και Βάθους, στον πρώτο και στο δεύτερο πίνακα αντιστοίχως. Τα αποτελέσματα είναι παρόμοια, καθώς και τα δύο μοντέλα επωφελούνται από την πληροφορία που μεταφέρεται από τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400. Η καλύτερη απόδοση επιτυγχάνεται όταν το τελευταίο επίπεδο Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά. Αυτή η προσέγγιση επιτυγχάνει μια ακρίβεια του 26.92% στο σύνολο δεδομένων IsoGD για τα δεδομένα RGB και μια ακρίβεια του 29.8% για τα δεδομένα Βάθους. Αυτή η συμπεριφορά οφείλεται στο γεγονός ότι το σύνολο δεδομένων IsoGD είναι ένα μεγάλο σύνολο δεδομένων κλάσεων χειρονομιών και το σύνολο δεδομένων Kinetics-400, στο οποίο εκπαιδεύτηκε το μοντέλο ResNet, αποτελείται από κλάσεις δράσεων. Έτσι, το μοντέλο χρειάζεται περισσότερα εκπαιδεύσιμα επίπεδα για να προσαρμοστεί στο νέο σύνολο δεδομένων.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33.294.009	1.4
Kinetics-400	ALL	33.294.009	1.5
Kinetics-400	layers 1, 2, 3, 4 + fc	33.265.657	7.91
Kinetics-400	layers 2, 3, 4 + fc	32.822.777	18.42
Kinetics-400	**layers 3, 4 + fc**	**31.265.017**	**25.67**
Kinetics-400	layer 4 + fc	25.036.537	26.92
Kinetics-400	fc	127.737	11.75

Table 1.4: Αποτελέσματα προεκπαιδευμένου ResNet-18 στη βάση IsoGD για RGB δεδομένα.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33.294.009	2.2
Kinetics-400	ALL	33.294.009	2.6
Kinetics-400	layers 1, 2, 3, 4 + fc	33.265.657	14.64
Kinetics-400	layers 2, 3, 4 + fc	32.822.777	19.63
Kinetics-400	**layers 3, 4 + fc**	**31.265.017**	**23.19**
Kinetics-400	layer 4 + fc	25.036.537	29.8
Kinetics-400	fc	127.737	10.6

Table 1.5: Αποτελέσματα προεκπαιδευμένου ResNet-18 στη βάση IsoGD για δεδομένα Βάθους.

ResNet στη βάση NVGesture

Η επίδοση του μοντέλου ResNet-18 3D στη βάση δεδομένων NVGesture για τα δεδομένα RGB και Βάθους παρουσιάζονται στους παρακάτω πίνακες. Τα αποτελέσματα είναι παρόμοια με τα αποτελέσματα του συνόλου δεδομένων IsoGD, όπου η καλύτερη απόδοση επιτυγχάνεται όταν το τελευταίο επίπεδο Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά. Αυτή η προσέγγιση επιτυγχάνει μια ακρίβεια του 25.52% στο σύνολο δεδομένων NVGesture για τα δεδομένα RGB και μια ακρίβεια του 32.29% για τα δεδομένα Βάθους.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33.182.972	1.1
Kinetics-400	ALL	33.182.972	1.3
Kinetics-400	layers 1, 2, 3, 4 + fc	33.150.745	4.15
Kinetics-400	layers 2, 3, 4 + fc	32.707.865	18.26
Kinetics-400	**layers 3, 4 + fc**	**31.150.105**	**20.54**
Kinetics-400	layer 4 + fc	24.921.625	25.52
Kinetics-400	fc	12.825	8.3

Table 1.6: Αποτελέσματα προεκπαιδευμένου ResNet-18 στη βάση NVGesture για RGB δεδομένα.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33.182.972	3.01
Kinetics-400	ALL	33.182.972	3.24
Kinetics-400	layers 1, 2, 3, 4 + fc	33.150.745	7.02
Kinetics-400	layers 2, 3, 4 + fc	32.707.865	23.12
Kinetics-400	**layers 3, 4 + fc**	**31.150.105**	**28.71**
Kinetics-400	layer 4 + fc	24.921.625	32.29
Kinetics-400	fc	12.825	14.52

Table 1.7: Αποτελέσματα προεκπαιδευμένου ResNet-18 στη βάση NVGesture για δεδομένα Βάθους.

Διαίρεση της βάσης NTU-RGB+D σε σύνολα δεδομένων δράσεων και χειρονομιών

Η βάση NTU-RGB+D είναι μια μεγάλη βάση δεδομένων, που περιέχει ένα μεγάλο αριθμό κλάσεων, συμπεριλαμβανομένων τόσο κλάσεων δράσεων όσο και χειρονομιών. Προκειμένου να αξιολογηθούν τα προτεινόμενα μοντέλα MTL, τα οποία εξετάζουν εάν οι εργασίες της αναγνώρισης δράσεων και χειρονομιών μπορούν να εκπαιδευτούν ταυτόχρονα, η βάση NTU-RGB+D διαιρείται σε δύο μέρη, ένα που περιέχει μόνο τις κλάσεις δράσεων και ένα που περιέχει μόνο τις κλάσεις χειρονομιών.

Η διαίρεση αυτή γίνεται χειροκίνητα επιλέγοντας τις κλάσεις που θεωρούνται κλάσεις δράσεων και τις κλάσεις που θεωρούνται κλάσεις χειρονομιών. Αυτή η αυθαίρετη διαίρεση μπορεί να διαφέρει καθώς κάποιες κλάσεις μπορεί να θεωρηθούν και ως κλάσεις δράσεων και ως κλάσεις χειρονομιών. Σε αυτή την εργασία, οι κλάσεις χειρονομιών που επιλέγονται από τη βάση NTU-RGB+D είναι οι εξής: *clapping (A10), hand waving (A23), pointing to something with finger (A31), rub two hands together (A34), salute (A38), put the palms together (A39), cross hands in front (say stop) (A40), use a fan (with hand or paper)/feeling warm (A49), point finger at the other person (A54), handshaking (A58), hush (quite) (A67), thumb up (A69), thumb down (A70), make ok sign (A71), make victory sign (A72), snapping fingers (A77), apply cream on hand back (A86), shake fist (A93), hands up (both hands) (A95), cross arms (A96), arm circles (A97), arm swings (A98), high-five (A112), finger-guessing game (playing rock-paper-scissors) (A120).*

Συνεπώς, για να αξιολογηθούν τα προτεινόμενα μοντέλα MTL, το μοντέλο ResNet εκπαιδεύεται στη βάση NTU-RGB+D για τη διαίρεση που περιέχει μόνο τις κλάσεις δράσεων (**NTU_ar**) και τη διαίρεση που περιέχει μόνο τις κλάσεις χειρονομιών, η οποία συνδυάζεται με τα σύνολα χειρονομιών IsoGD (**NTU_gr_IsoGD**) και NVGesture (**NTU_gr_NVGesture**)αντίστοιχα. Όταν συνδυάζονται οι κλάσεις χειρονομιών από τη βάση NTU-RGB+D με τις βάσεις δεδομένων IsoGD και NVGesture, κάποιες κλάσεις είναι κοινές μεταξύ των βάσεων δεδομένων. Οι κοινές κλάσεις είναι για το συνδυασμό των NTU-RGB+D και IsoGD είναι οι *ok sign, victory*

sign ενώ για το συνδυασμό των NTU-RGB+D και NVGesture είναι οι *ok sign, thumb up*. Στα νέα σύνολα, που προκύπτουν, αυτές οι κλάσεις τροποποιούνται ώστε να περιέχουν δείγματα και από τη βάση NTU-RGB+D και από τις βάσεις δεδομένων IsoGD και NVGesture, αντίστοιχα.

Τα αποτελέσματα, που παρουσιάζονται στους παρακάτω πίνακες, για τις διαίρεσεις σε υποσύνολα δεδομένων χειρονομιών, δείχνουν παρόμοια συμπεριφορά με τα προηγούμενα πειράματα. Για το μοντέλο που εκπαιδεύεται στις κλάσεις δράσεων της βάσης NTU-RGB+D, η καλύτερη απόδοση επιτυγχάνεται όταν το τελευταίο επίπεδο Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά. Αυτή η προσέγγιση επιτυγχάνει μια ακρίβεια του 58.72% στη διαίρεση του συνόλου δεδομένων NTU-RGB+D για τις κλάσεις δράσεων. Για το μοντέλο που εκπαιδεύεται στις κλάσεις χειρονομιών της βάσης NTU-RGB+D σε συνδυασμό με τις βάσεις IsoGD και NVGesture, η καλύτερη απόδοση επιτυγχάνεται όταν τα τελευταία 2 επίπεδα Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, επιτυγχάνοντας ακρίβεια 42.68% και 67.23% αντιστοίχως.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33,227,832	16.88
Kinetics-400	ALL	33,227,832	18.12
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	29.93
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	51.33
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**58.72**
Kinetics-400	layers 4 + fc	24,970,360	60.94
Kinetics-400	fc	61,560	31.95

Table 1.8: Αποτελέσματα προεκπαιδευμένου ResNet-18 στο σύνολο δεδομένων δράσεων της βάσης NTU_ar για RGB δεδομένα.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33,227,832	4.82
Kinetics-400	ALL	33,227,832	6.33
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	9.38
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	39.72
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**42.68**
Kinetics-400	layers 4 + fc	24,970,360	42.6
Kinetics-400	fc	61,560	29.49

Table 1.9: Αποτελέσματα προεκπαιδευμένου ResNet-18 στο σύνολο δεδομένων χειρονομιών της βάσης NTU_gr_IsoGD.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33,227,832	24.23
Kinetics-400	ALL	33,227,832	29.67
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	41.07
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	56.55
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**67.23**
Kinetics-400	layers 4 + fc	24,970,360	65.96
Kinetics-400	fc	61,560	43.19

Table 1.10: Αποτελέσματα προεκπαιδευμένου ResNet-18 στο σύνολο δεδομένων χειρονομιών της βάσης NTU_gr_NVGesture.

Σύνοψη αποτελεσμάτων ResNet-18 3D

Σε αυτή την ενότητα, αξιολογήθηκε η απόδοση της αρχιτεκτονικής ResNet-18 3D στα σύνολα δεδομένων UCF-101, NTU-RGB+D, IsoGD και NVGesture. Τα αποτελέσματα δείχνουν ότι η καλύτερη απόδοση επιτυγχάνεται

όταν κάποια από τα επίπεδα αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400 και κρατούνται σταθερά. Με αυτό τον τρόπο, για να αξιοποιηθούν τόσο τα οφέλη της μεταφορά γνώσης όσο και της μάθησης πολλαπλών εργασιών, τα προτεινόμενα μοντέλα MTL θα έχουν τα τελευταία 2 επίπεδα Residual και το πλήρως συνδεδεμένο επίπεδο εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά. Αυτή η δομή ακολουθείται ώστε η μέθοδος μάθησης πολλαπλών εργασιών να λειτουργήσει στα τελευταία εκπαιδεύσιμα επίπεδα, που μαθαίνουν πιο περίπλοκα χαρακτηριστικά που αφορούν την εργασία, ενώ τα πρώτα επίπεδα, που μαθαίνουν πιο γενικά χαρακτηριστικά των εργασιών, δεν θα ενημερωθούν κατά τη διάρκεια της διαδικασίας εκπαίδευσης, αλλά θα οριστούν σύμφωνα με τα προ-εκπαιδευμένα βάρη στο σύνολο δεδομένων Kinetics-400.

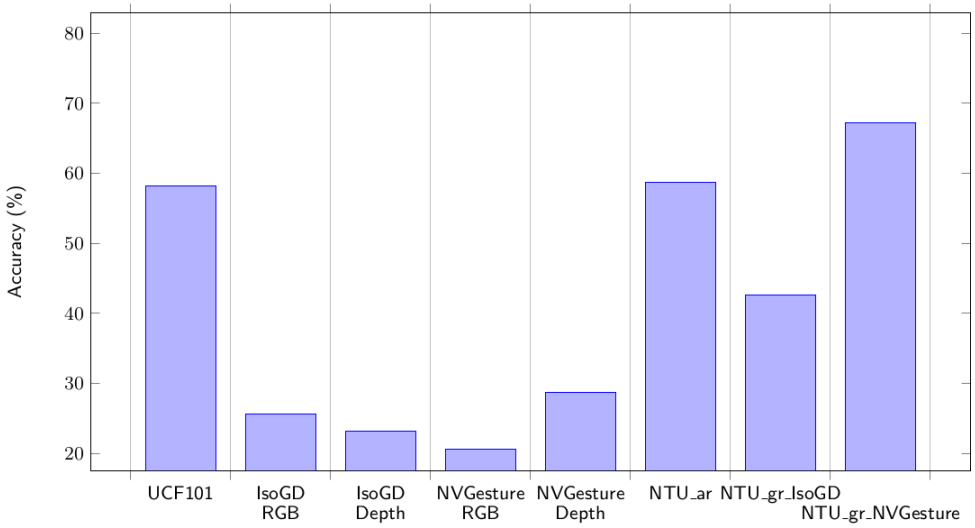


Figure 1.5.1: Επιδόσεις ResNet-18 3D για τα μοντέλα μονής εργασίας

1.5.2 Αποτελέσματα Πειραμάτων σε Μάθηση των Εργασιών της Αναγνώρισης Ανθρώπινων Δράσεων και των Χειρονομιών

Για να δείξουμε την αποτελεσματικότητα της προτεινόμενης μεθόδου MTL στην από κοινού εκπαίδευση των εργασιών της αναγνώρισης ανθρώπινων δράσεων και χειρονομιών, υλοποιούμε 3 διαφορετικές μεθόδους MTL και 3 διαφορετικούς τρόπους υπολογισμού της συνάρτησης κόστους της μάθησης πολλαπλών εργασιών. Έτσι, προκύπτουν 9 διαφορετικά πειράματα για κάθε συνδυασμό των συνόλων δεδομένων δράσεων και χειρονομιών. Για τις μεθόδους MTL, επιλέγονται οι μέθοδοι Ολικού Διαμοιρασμού Παραμέτρων, Μερικού Διαμοιρασμού Παραμέτρων και Εκμάθησης Διαμοιρασμού Βαρών. Όσον αφορά τους τρόπους υπολογισμού του συνολικού σφάλματος της μάθησης πολλαπλών εργασιών επιλέγονται οι μέθοδοι Μέσου Όρου, Αβεβαιότητας στη Στάθμιση των Σφαλμάτων και ο Δυναμικός Μέσος Όρος Βαρών.

Οι επιδόσεις του μοντέλου ResNet-18 3D με μονή μάθηση εργασιών, κρατώντας τα τελευταία 2 επίπεδα Residual και το πλήρως συνδεδεμένο επίπεδο εκπαιδεύσιμα, συνοψίζονται ακολούθως.

- UCF-101: 58.26%
- IsoGD: 25.67%
- NVGesture: 20.54%
- NTU_ar (NTU-RGB+D action classes): 58.72%
- NTU_gr_IsoGD (NTU-RGB+D gesture classes + IsoGD): 42.68%
- NTU_gr_NVGesture (NTU-RGB+D gesture classes + NVGesture): 67.23%

Επίσης είναι σημαντικό να αναφερθεί ότι για αυτή τη δομή του ResNet-18 3D, το σύνολο των εκπαιδεύσιμων παραμέτρων είναι περίπου 35 εκατομμύρια. Αυτή η πληροφορία είναι σημαντική για τη σύγκριση με τα μοντέλα μάθησης πολλαπλών εργασιών που ακολουθούν, ώστε να μην οδηγηθούμε σε λανθασμένα αποτελέσματα ότι τα μοντέλα MTL είναι καλύτερα απλά επειδή έχουν περισσότερες εκπαιδεύσιμες παραμέτρους, αλλά να επικεντρωθεί μόνο στην αποτελεσματικότητα του να μοιράζονται γνώση μεταξύ των εργασιών που εκπαιδεύονται.

Ολικός Διαμοιρασμός Παραμέτρων (Hard Parameter Sharing - HPS)

Τα μοντέλα Ολικού Διαμοιρασμού Παραμέτρων είναι απλές παραλλαγές των μοντέλων μάθησης μίας εργασίας, καθώς διαφέρουν μόνο στα τελευταία επίπεδα που αφορούν την εκάστοτε εργασία. Έτσι, τα παρακάτω μοντέλα έχουν λίγο περισσότερες παραμέτρους από τα μοντέλα μάθησης μίας εργασίας.

Όταν η μέθοδος Ολικού Διαμοιρασμού Παραμέτρων χρησιμοποιείται στα σύνολα δεδομένων UCF-101 και IsoGD, το μοντέλο πολλαπλών εργασιών επιτυγχάνει μια ακρίβεια του 57.36% και 25.45% αντίστοιχα, ως το καλύτερο αποτέλεσμα. Αυτή η απόδοση είναι περίπου ίδια με το μοντέλο μάθησης μίας εργασίας, που σημαίνει ότι αυτή η μέθοδος δεν είναι η καλύτερη επιλογή για αυτό το σύνολο δεδομένων.

Σφάλμα	Βάση Δεδομένων	
	UCF-101	IsoGD
Μάθηση Μονής Εργασίας		
-	58.26	25.67
Ολικός Διαμοιρασμός Παραμέτρων		
average	54.16	20.49
dwa	57.36	25.45
uncertainty	60.53	20.02

Table 1.11: Αποτελέσματα Ολικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στις βάσεις UCF-101 και IsoGD.

Από την άλλη, όταν η μέθοδος Ολικού Διαμοιρασμού Παραμέτρων χρησιμοποιείται στα σύνολα δεδομένων UCF-101 και NVGesture, το μοντέλο πολλαπλών εργασιών επιτυγχάνει μια ακρίβεια του 66.24% και 33.68% αντίστοιχα, ως το καλύτερο αποτέλεσμα, με χρήση της αβεβαιότητας στη στάθμιση σφαλμάτων ως μέθοδος υπολογισμού του λάθους πολλαπλών εργασιών. Αυτή η απόδοση είναι καλύτερη από τα μοντέλα μάθησης μίας εργασίας και για τα δύο σύνολα δεδομένων.

Σφάλμα	Βάση Δεδομένων	
	UCF-101	NVGesture
Μάθηση Μονής Εργασίας		
-	58.26	20.54
Ολικός Διαμοιρασμός Παραμέτρων		
average	64.34	27.86
dwa	58.6	25.57
uncertainty	66.24	33.68

Table 1.12: Αποτελέσματα Ολικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στις βάσεις UCF-101 και NVGesture.

Για την εφαρμογή αυτής της μεθόδου στα σύνολα δεδομένων NTU-RGB+D και IsoGD, το μοντέλο πολλαπλών εργασιών επιτυγχάνει μια ακρίβεια του 70.65% και 60.25% αντίστοιχα, όταν υπολογίζει το συνολικό σφάλμα ως το μέσο όρο των σφαλμάτων των εργασιών, επιτυγχάνοντας καλύτερα αποτελέσματα από τα μοντέλα μάθησης μίας εργασίας και για τα δύο σύνολα δεδομένων.

Σφάλμα	Βάση Δεδομένων	
	NTU_ar	NTU_gr_IsoGD
Μάθηση Μονής Εργασίας		
-	58.72	42.68
Ολικός Διαμοιρασμός Παραμέτρων		
average	70.65	60.25
dwa	64.25	58.87
uncertainty	49.42	56.57

Table 1.13: Αποτελέσματα Ολικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στα συνόλα δεδομένων NTU_ar και NTU_gr_IsoGD.

Ομοίως, η εφαρμογή της μεθόδου Ολικού Διαμοιρασμού Παραμέτρων στα σύνολα δεδομένων NTU-RGB+D και NVGesture, επιτυγχάνει μια ακρίβεια του 72.53% και 74.33% αντίστοιχα, όταν υπολογίζει το συνολικό σφάλμα ως το μέσο όρο των σφαλμάτων των εργασιών, επιτυγχάνοντας καλύτερα αποτελέσματα από τα μοντέλα μάθησης μίας εργασίας και για τα δύο σύνολα δεδομένων.

Σφάλμα	Βάση Δεδομένων	
	NTU_ar	NTU_gr_NVGesture
Μάθηση Μονής Εργασίας		
-	58.72	67.23
Ολικός Διαμοιρασμός Παραμέτρων		
average	71.91	74.33
dwa	72.53	73.85
uncertainty	72.51	72.80

Table 1.14: Αποτελέσματα Ολικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στα σύνολα δεδομένων NTU_ar και NTU_gr_NVGesture.

Συγκεντρωτικά, τα μοντέλα Ολικού Διαμοιρασμού Παραμέτρων, χρησιμοποιώντας διαφορετικούς συνδυασμούς συνόλων δεδομένων και μεθόδων υπολογισμού του σφάλματος πολλαπλών εργασιών, μπορούν να οδηγήσουν σε βέλτιστα αποτελέσματα, καθώς όλοι οι συνδυασμοί υπερτερούν των μοντέλων μάθησης μίας εργασίας, αποδεικνύοντας την αποτελεσματικότητα του διαμοιρασμού παραμέτρων. Αυτά τα μοντέλα έχουν λίγο περισσότερες εκπαιδεύσιμες παραμέτρους από τα μοντέλα μάθησης μίας εργασίας, αλλά εξακολουθούν να υπερτερούν, δίνοντας ισχυρές βάσεις για την πεποίθηση ότι ο διαμοιρασμός βαρών οφελεί τις εργασίες των ανθρώπινων δράσεων και των χειρονομιών. Μόνο ένα από τους συνδυασμούς συνόλων δεδομένων, το σύνολο δεδομένων UCF-101 και IsoGD, δεν μπόρεσε να πετύχει καλύτερα αποτελέσματα, καθώς το σύνολο δεδομένων των χειρονομιών ήταν αρκετά μεγαλύτερο από το σύνολο δεδομένων των δράσεων, με αποτέλεσμα το μοντέλο

να μην μπορεί να μάθει το σύνολο δεδομένων των δράσεων σωστά

Μερικός Διαμοιρασμός Παραμέτρων (Soft Parameter Sharing - SPS)

Για το Μερικό Διαμοιρασμό Παραμέτρων, επιλέγονται τα δίκτυα cross-stitch, τα οποία απαιτούν την υλοποίηση δικτύων για κάθε εργασία και στη συνέχεια τη συνένωσή τους χρησιμοποιώντας μια μονάδα cross-stitch. Αυτή η δομή είναι πιο πολύπλοκη από τα μοντέλα μάθησης μίας εργασίας, έτσι το πλήθος των εκπαιδεύσιμων παραμέτρων είναι υψηλότερο από τα μοντέλα μάθησης μίας εργασίας, περίπου διπλάσιο (70 εκατομμύρια), καθώς το μοντέλο θα μάθει δύο εργασίες. Επιπλέον, στη μονάδα cross-stitch, χρησιμοποιείται η παράμετρος σ για να ελέγξει την ποσότητα των πληροφοριών που μοιράζονται μεταξύ των εργασιών. Όπως αναφέρεται στο άρθρο τους, η επιλογή δεν μπορεί να είναι αυθαίρετη, αλλά πρέπει να επιλεγεί εμπειρικά για κάθε σύνολο δεδομένων. Έτσι, πειραματιζόμαστε με διαφορετικές τιμές του σ για να βρούμε τη βέλτιστη για κάθε σύνολο δεδομένων, επιλέγοντας μεταξύ 0.2 και 0.8.

Όταν τα δίκτυα cross-stitch χρησιμοποιούνται για το σύνολο δεδομένων UCF-101 και IsoGD, η καλύτερη απόδοση επιτυγχάνεται όταν η παράμετρος σ ορίζεται στο 0.8, με τη μέθοδο υπολογισμού του σφάλματος να είναι η αβεβαιότητα στη στάθμιση του σφάλματος, επιτυγχάνοντας ακρίβεια 70.24% και 35.79% αντίστοιχα. Αυτή η απόδοση είναι καλύτερη από τα μοντέλα μάθησης μίας εργασίας, μια συμπεριφορά που δεν παρατηρήθηκε για τη μέθοδο Ολικού Διαμοιρασμού Παραμέτρων.

Σφάλμα	Βάση Δεδομένων	
	UCF-101	IsoGD
Μάθηση Μονής Εργασίας		
-	58.26	25.67
Cross-Stitch (s=0.2)		
average	67.64	26.6
dwa	62.2	30.96
uncertainty	67.62	26.25
Cross-Stitch (s=0.8)		
average	64.22	26.78
dwa	63.1	32.03
uncertainty	70.24	35.79

Table 1.15: Αποτελέσματα Μερικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στις βάσεις UCF-101 και IsoGD.

Επιπλέον, όταν τα δίκτυα cross-stitch χρησιμοποιούνται για το σύνολο δεδομένων UCF-101 και NVGesture, η καλύτερη απόδοση επιτυγχάνεται όταν η παράμετρος σ ορίζεται στο 0.2, με τη μέθοδο υπολογισμού του σφάλματος να είναι η αβεβαιότητα στη στάθμιση του σφάλματος, επιτυγχάνοντας ακρίβεια 71.66% και 43.24% αντίστοιχα.

Σφάλμα	Βάσεις Δεδομένων	
	UCF-101	NVGesture
Μάθηση Μονής Εργασίας		
-	58.26	20.54
Cross-Stitch (s=0.2)		
average	69.94	36.17
dwa	71.66	37.42
uncertainty	71.66	43.24
Cross-Stitch (s=0.8)		
average	73.14	35.76
dwa	67.94	34.93
uncertainty	68.75	39.09

Table 1.16: Αποτελέσματα Μερικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στις βάσεις UCF-101 και NVGesture.

Για τα σύνολα δεδομένων NTU-RGB+D και IsoGD, η καλύτερη απόδοση επιτυγχάνεται όταν η παράμετρος σ ορίζεται στο 0.2, με τη μέθοδο υπολογισμού του σφάλματος να είναι η αβεβαιότητα στη στάθμιση του σφάλματος, επιτυγχάνοντας ακρίβεια 74.45% και 62.14% αντίστοιχα.

Σφάλμα	Βάση Δεδομένων	
	NTU_ar	NTU_gr_IsoGD
Μάθηση Μονής Εργασίας		
-	58.72	42.68
Cross-Stitch (s=0.2)		
average	72.27	60.13
dwa	71.84	60.91
uncertainty	74.45	62.14
Cross-Stitch (s=0.8)		
average	71.82	59.3
dwa	72.02	60.55
uncertainty	73.66	60.8

Table 1.17: Αποτελέσματα Μερικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στα σύνολα δεδομένων NTU_ar και NTU_gr_IsoGD.

Τέλος, για τα σύνολα δεδομένων NTU-RGB+D και NVGesture, η καλύτερη απόδοση επιτυγχάνεται όταν η παράμετρος σ ορίζεται στο 0.8, με τη μέθοδο υπολογισμού του σφάλματος να είναι η αβεβαιότητα στη στάθμιση του σφάλματος, επιτυγχάνοντας ακρίβεια 75.51% και 74.03% αντίστοιχα.

Σφάλμα	Βάση Δεδομένων	
	NTU_ar	NTU_gr_NVGesture
Μάθηση Μονής Εργασίας		
-	58.72	67.23
Cross-Stitch (s=0.2)		
average	74.2	74.49
dwa	73.51	74.59
uncertainty	74.17	73.9
Cross-Stitch (s=0.8)		
average	72.65	74.21
dwa	74.03	75.51
uncertainty	72.25	73.54

Table 1.18: Αποτελέσματα Μερικού Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στα σύνολα δεδομένων NTU_ar και NTU_gr_NVGesture.

Συνοψίζοντας, για τη μέθοδο Μερικού Διαμοιρασμού Παραμέτρων, όλοι οι συνδυασμοί συνόλων δεδομένων και μεθόδων υπολογισμού του σφάλματος πολλαπλών εργασιών υπερτερούν των μοντέλων μάθησης μίας εργασίας, δείχνοντας ότι η μέθοδος αυτή είναι αποτελεσματική για την εκπαίδευση του μοντέλου στις εργασίες της αναγνώρισης ανθρώπινων δράσεων και των χειρονομιών. Επιπλέον, η αβεβαιότητα στη στάθμιση του σφάλματος είναι η καλύτερη μέθοδος υπολογισμού του σφάλματος πολλαπλών εργασιών για τα σύνολα δεδομένων των ανθρώπινων δράσεων και των χειρονομιών, ενώ η τιμή της παραμέτρου σ ανάλογα με τις βάσεις δεδομένων.

Εκμάθηση Διαμοιρασμού Βαρών (Learned Weight Sharing - LWS)

Για τη δημιουργία ενός μοντέλου με εκμάθηση διαμοιρασμού βαρών, υλοποιούνται δίκτυα για κάθε εργασία, οδηγώντας σε ένα μοντέλο με έναν αριθμό εκπαιδεύσιμων παραμέτρων περίπου διπλάσιο (70 εκατομμύρια), καθώς το μοντέλο θα μάθει δύο εργασίες.

Όταν η μέθοδος Εκμάθησης Διαμοιρασμού Βαρών χρησιμοποιείται στα σύνολα δεδομένων UCF-101 και IsoGD, το μοντέλο πολλαπλών εργασιών επιτυγχάνει μια ακρίβεια του 70.58% και 34.9% αντίστοιχα, ως το καλύτερο αποτέλεσμα, με τη μέθοδο υπολογισμού του σφάλματος να είναι ο δυναμικός μέσος όρος των σφαλμάτων πολλαπλών εργασιών.

Σφάλμα	Βάση Δεδομένων	
	UCF-101	IsoGD
Μάθηση Μονής Εργασίας		
-	58.26	25.67
Εκμάθηση Διαμοιρασμού Παραμέτρων		
average	64.39	35.2
dwa	70.58	34.9
uncertainty	64.79	28.7

Table 1.19: Αποτελέσματα Εκμάθησης Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στις βάσεις UCF-101 και IsoGD.

Για τις βάσεις δεδομένων UCF-101 και NVGesture, το καλύτερο αποτέλεσμα επιτυγχάνεται όταν η μέθοδος υπολογισμού του σφάλματος είναι ο δυναμικός μέσος όρος των σφαλμάτων πολλαπλών εργασιών, με ακρίβεια 71.42% και 43.04% αντίστοιχα.

Σφάλμα	Βάσεις Δεδομένων	
	UCF-101	NVGesture
Μάθηση Μονής Εργασίας		
-	58.26	20.54
Εκμάθηση Διαμοιρασμού Παραμέτρων		
average	69.73	39.62
dwa	71.42	43.04
uncertainty	72.27	37.21

Table 1.20: Αποτελέσματα Εκμάθησης Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στις βάσεις UCF-101 και NVGesture.

Για τα σύνολα δεδομένων NTU-RGB+D και IsoGD, το καλύτερο αποτέλεσμα επιτυγχάνεται όταν η μέθοδος υπολογισμού του σφάλματος είναι ο μέσος όρος των σφαλμάτων, με ακρίβεια 73.03% και 56.27% αντίστοιχα.

Σφάλμα	Βάση Δεδομένων	
	NTU_ar	NTU_gr_IsoGD
Μάθηση Μονής Εργασίας		
-	58.72	42.68
Εκμάθηση Διαμοιρασμού Παραμέτρων		
average	73.03	56.27
dwa	72.6	53.64
uncertainty	72.48	53.07

Table 1.21: Αποτελέσματα Εκμάθησης Διαμοιρασμού Βαρών προεκπαιδευμένου ResNet-18 στα σύνολα δεδομένων NTU_ar και NTU_gr_IsoGD.

Τέλος, για τα σύνολα δεδομένων NTU-RGB+D και NVGesture, το καλύτερο αποτέλεσμα επιτυγχάνεται όταν η μέθοδος υπολογισμού του σφάλματος είναι ο δυναμικός μέσος όρος των σφαλμάτων πολλαπλών εργασιών, με ακρίβεια 73.41% και 75.47% αντίστοιχα.

Σφάλμα	Βάση Δεδομένων	
	NTU_ar	NTU_gr_NVGesture
Μάθηση Μονής Εργασίας		
-	58.72	67.23
Εκμάθηση Διαμοιρασμού Παραμέτρων		
average	73.45	74.43
dwa	73.41	75.47
uncertainty	72.96	76.4

Table 1.22: Αποτελέσματα Εκμάθησης Διαμοιρασμού Παραμέτρων προεκπαιδευμένου ResNet-18 στα σύνολα δεδομένων NTU_ar και NTU_gr_NVGesture.

Επομένως, για τη μέθοδο Εκμάθησης Διαμοιρασμού Βαρών, όλοι οι συνδυασμοί συνόλων δεδομένων και μεθόδων υπολογισμού του σφάλματος πολλαπλών εργασιών υπερτερούν των μοντέλων μάθησης μίας εργασίας, αποδεικνύοντας την αποτελεσματικότητα της μεθόδου αυτής για την εκπαίδευση του μοντέλου στις εργασίες της αναγνώρισης ανθρώπινων δράσεων και των χειρονομιών.

Σύνοψη Αποτελεσμάτων Μεθόδων Μάθησης Πολλαπλών Εργασιών

Οι καλύτερες επιδόσεις για κάθε συνδυασμό συνόλων δεδομένων και μεθόδων υπολογισμού του σφάλματος πολλαπλών εργασιών παρουσιάζονται στο Σχήμα 1.5.2.

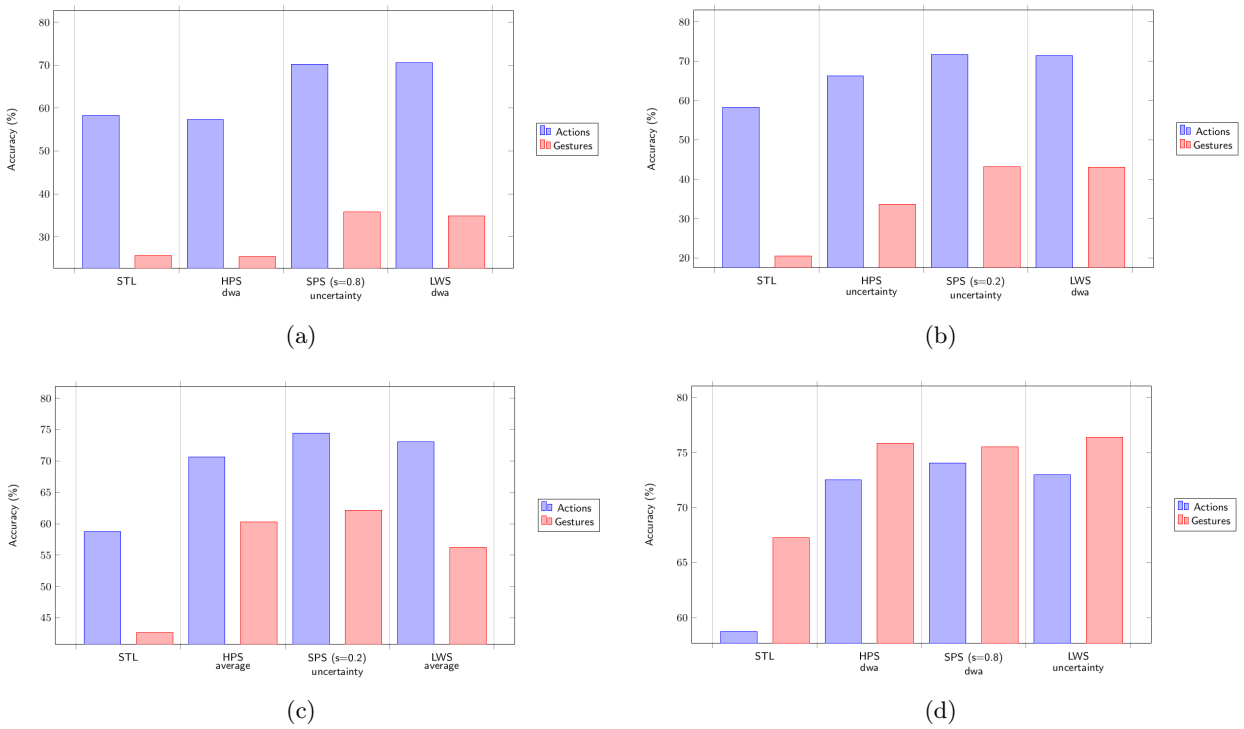


Figure 1.5.2: Αποτελέσματα Μάθησης Πολλαπλών Εργασιών: (a) UCF101-IsoGD, (b) UCF101-NVGesture, (c) NTU_ar-NTU_gr_IsoGD, (d) NTU_ar-NTU_gr_NVGesture.

Όπως φαίνεται από αυτά τα διαγράμματα, η επιλογή της μεθόδου υπολογισμού του σφάλματος πολλαπλών εργασιών δεν είναι δεδομένη, αλλά φαίνεται ότι πιο πολύπλοκες μέθοδοι, όπως η αβεβαιότητα στη στάθμιση του σφάλματος και ο δυναμικός μέσος όρος των σφαλμάτων πολλαπλών εργασιών, είναι πιο αποτελεσματικές από τη μέθοδο του μέσου όρου των σφαλμάτων πολλαπλών εργασιών. Επιπλέον, η επιλογή της μεθόδου μάθησης πολλαπλών εργασιών είναι σημαντική. Παρατηρείται ότι η μέθοδος Μερικού Διαμοιρασμού Παραμέτρων είναι πιο αποτελεσματική από τις άλλες μεθόδους, με τη μέθοδο Εκμάθησης Διαμοιρασμού Βαρών να φέρνει συγκρίσιμα αποτελέσματα με αυτή. Από την άλλη, η μέθοδος Ολικού Διαμοιρασμού Παραμέτρων αν και είναι η λιγότερο αποτελεσματική, εξακολουθεί να είναι καλύτερη από τα περισσότερα μοντέλα μάθησης μονής εργασίας, με εξαίρεση την περίπτωση που το σύνολο δεδομένων των χειρονομιών είναι μεγαλύτερο από το σύνολο δεδομένων των δράσεων. Συνολικά, οι μέθοδοι μάθησης πολλαπλών εργασιών είναι αποτελεσματικές για την εκπαίδευση του μοντέλου στα σύνολα δεδομένων των δράσεων και των χειρονομιών, καθώς υπερτερούν των μοντέλων μάθησης μονής εργασίας.

1.5.3 Αποτελέσματα Πειραμάτων σε Μάθηση Μοντέλων Διαχείρισης Διαφορετικών Δεδομένων για Δεδομένα RGB και Depth

Το βασικό μοντέλο ResNet-18 3D τροποποιείται για να δέχεται διαφορετικού τύπου δεδομένα ως είσοδο, δημιουργώντας δύο κλάδους διαφορετικών επιπέδων Residual, ένα για τα δεδομένα RGB και ένα για τα δεδομένα βάθους. Η έξοδος αυτών των κλάδων συγχωνεύεται χρησιμοποιώντας μια μέθοδο συγχώνευσης μέσης τιμής στο τέλος του μοντέλου, δημιουργώντας έτσι την τελική πρόβλεψη. Αυτό το μοντέλο εκπαιδεύεται στα σύνολα δεδομένων IsoGD και NVGesture, όπου τα δεδομένα RGB και βάθους είναι διαθέσιμα. Το μοντέλο εκπαιδεύεται για 20.000 επαναλήψεις, για να μπορεί να συγκριθεί με το μοντέλο μάθησης πολλαπλών εργασιών. Όπως και στα προηγούμενα πειράματα, η απόδοση του μοντέλου αξιολογείται χρησιμοποιώντας τη μετρική ακρίβειας, ενώ τα προ-εκπαιδευμένα βάρη από το σύνολο δεδομένων Kinetics-400 χρησιμοποιούνται για την αρχικοποίηση.

ResNet στη βάση IsoGD

Για την περίπτωση της διαχείρισης διαφορετικών τύπων δεδομένων της βάσης IsoGD, το μοντέλο ResNet-18 3D παρουσιάζει παρόμοια συμπεριφορά με τη μάθηση ενός μόνο τύπου δεδομένων, όπως φαίνεται στον παρακάτω πίνακα. Επιπλέον, όπως αναμενόταν από τη βιβλιογραφία, το μοντέλο αυτό παρουσιάζει βελτιωμένη απόδοση σε σχέση με τα μοντέλα με έναν τύπο δεδομένων, καθώς το μοντέλο μπορεί να μάθει από τους δύο τύπους δεδομένων και να συγχωνεύσει τις πληροφορίες σε μια μόνο πρόβλεψη. Η καλύτερη απόδοση επιτυγχάνεται όταν τα τελευταία δύο επίπεδα Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, επιτυγχάνοντας ακρίβεια 30.61% στη βάση δεδομένων IsoGD.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33.294.009	8.76
Kinetics-400	ALL	33.294.009	9.87
Kinetics-400	layers 1, 2, 3, 4 + fc	33.265.657	14.26
Kinetics-400	layers 2, 3, 4 + fc	32.822.777	25.85
Kinetics-400	**layers 3, 4 + fc**	**31.265.017**	**30.61**
Kinetics-400	layer 4 + fc	25.036.537	28.91
Kinetics-400	fc	127.737	15.34

Table 1.23: Αποτελέσματα ResNet-18 Μάθησης Διαφορετικών Τύπων Δεδομένων στη βάση IsoGD για RGB δεδομένα και δεδομένα Βάθους.

ResNet στη βάση NVGesture

Το μοντέλο ResNet-18 3D που εκπαιδεύτηκε στη βάση δεδομένων NVGesture για την περίπτωση της διαχείρισης διαφορετικών τύπων δεδομένων παρουσιάζει την ίδια συμπεριφορά με τη βάση δεδομένων IsoGD, όπως φαίνεται στον παρακάτω πίνακα, με την καλύτερη δομή να είναι αυτή που αφήνει το τελευταίο επίπεδο Residual και το πλήρως συνδεδεμένο επίπεδο εκπαιδεύσιμα, επιτυγχάνοντας ακρίβεια 46.49%.

Προεκπαίδευση	Εκπαιδεύσιμα Επίπεδα	Παράμετροι	Απόδοση
from scratch	-	33.182.972	5.19
Kinetics-400	ALL	33.182.972	5.22
Kinetics-400	layers 1, 2, 3, 4 + fc	33.150.745	9.95
Kinetics-400	layers 2, 3, 4 + fc	32.707.865	28.82
Kinetics-400	**layers 3, 4 + fc**	**31.150.105**	**32.57**
Kinetics-400	layer 4 + fc	24.921.625	46.49
Kinetics-400	fc	12.825	24.69

Table 1.24: Αποτελέσματα ResNet-18 Μάθησης Διαφορετικών Τύπων Δεδομένων στη βάση IsoGD για RGB δεδομένα και δεδομένα Βάθους.

Σύνοψη αποτελεσμάτων πολυτροπικών ResNet

Σε αυτή τη ενότητα, αξιολογήθηκε η απόδοση του μοντέλου ResNet-18 3D στα σύνολα δεδομένων IsoGD και NVGesture, για τη διαχείριση διαφορετικού τύπου δεδομένων στο ίδιο νευρωνικό δίκτυο. Τα αποτελέσματα δείχνουν βελτίωση σε σχέση με τα μοντέλα με έναν τύπο δεδομένων. Για την αξιολόγηση του προτεινόμενου μοντέλου μάθησης πολλαπλών εργασιών με διαφορετικούς τύπους δεδομένων, η δομή του μοντέλου επιλέχθηκε να είναι η ίδια με τα προηγούμενα μοντέλα, όπου τα τελευταία 2 επίπεδα Residual και το πλήρως συνδεδεμένο επίπεδο αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά.

1.5.4 Αποτελέσματα Πειραμάτων σε Μάθηση Πολλαπλών Εργασιών για Δεδομένα RGB και Depth

Σε αυτή την ενότητα, παρουσιάζονται τα αποτελέσματα της μάθησης πολλαπλών εργασιών για τα σύνολα δεδομένων IsoGD και NVGesture, χρησιμοποιώντας τα δεδομένα RGB και βάθους. Συγκεκριμένα, υλοποιούνται 3 διαφορετικές μεθόδους μάθησης πολλαπλών εργασιών και 3 διαφορετικές μεθόδους υπολογισμού του σφάλματος πολλαπλών εργασιών. Έτσι, προκύπτουν 9 διαφορετικά πειράματα για κάθε σύνολο δεδομένων. Για τις μεθόδους μάθησης πολλαπλών εργασιών, επιλέγονται οι μέθοδοι ολικού διαμοιρασμού παραμέτρων, μερικού διαμοιρασμού παραμέτρων και εκμάθησης διαμοιρασμού βαρών, ενώ για τις μεθόδους υπολογισμού του σφάλματος πολλαπλών εργασιών, χρησιμοποιούνται ο μέσος όρος, η αβεβαιότητα στη στάθμιση του σφάλματος και ο δυναμικός μέσος όρος.

Οι επιδόσεις των μοντέλων μονής εργασίας ResNet, με τα τελευταία 2 επίπεδα Residual και το πλήρως συνδεδεμένο επίπεδο να αφήνονται εκπαιδεύσιμα, ενώ τα υπόλοιπα επίπεδα αρχικοποιούνται με τα προ-εκπαιδευμένα βάρη και κρατούνται σταθερά, παρουσιάζονται στη συνέχεια.

- IsoGD (RGB): 25.67%
- IsoGD (Depth): 23.19%
- IsoGD (RGB + Depth): 30.61%
- NVGesture (RGB): 20.54%
- NVGesture (Depth): 28.71%
- NVGesture (RGB + Depth): 32.57%

Πολυτροπική Μάθηση Πολλαπλών Εργασιών για Δεδομένα RGB και Depth στη βάση IsoGD

Τα πειράματα πολυτροπικής μάθησης πολλαπλών εργασιών στο σύνολο δεδομένων IsoGD παρουσιάζονται σε αυτή την ενότητα για τα δεδομένα RGB και βάθους. Τα αποτελέσματα που παρουσιάζονται στον παρακάτω πίνακα δείχνουν ότι για όλα τα πειράματα, το προτεινόμενο πλαίσιο μάθησης πολλαπλών εργασιών υπερτερεί του μοντέλου μάθησης μίας εργασίας και του πολυτροπικού μοντέλου. Η καλύτερη απόδοση επιτυγχάνεται όταν η μέθοδος διαμοιρασμού παραμέτρων επιλέγεται να είναι η μέθοδος μερικού διαμοιρασμού παραμέτρων, με την υλοποίηση cross stitch για $\sigma = 0.8$ και τη μέθοδο υπολογισμού του σφάλματος να είναι ο δυναμικός μέσος όρος, με ακρίβεια 48.91%, η οποία είναι περίπου 15% καλύτερη από το μοντέλο μάθησης μίας εργασίας και το πολυτροπικό μοντέλο.

Μάθηση Πολλαπλών Εργασιών	Σφάλμα	Απόδοση
Ολικός Διαμοιρασμός Παραμέτρων	average	43.82
Ολικός Διαμοιρασμός Παραμέτρων	dwa	44.33
Ολικός Διαμοιρασμός Παραμέτρων	uncertainty	46.09
Cross-Stitch (s=0.2)	average	46.95
Cross-Stitch (s=0.2)	dwa	43.66
Cross-Stitch (s=0.2)	uncertainty	45.72
Cross-Stitch (s=0.8)	average	43.45
Cross-Stitch (s=0.8)	dwa	48.91
Cross-Stitch (s=0.8)	uncertainty	45.34
Εκμάθηση Διαμοιρασμού Βαρών	average	46.42
Εκμάθηση Διαμοιρασμού Βαρών	dwa	47.3
Εκμάθηση Διαμοιρασμού Βαρών	uncertainty	46.04

Table 1.25: Αποτελέσματα ResNet-18 Μάθησης Πολλαπλών Εργασιών και Μάθησης Διαφορετικών Τύπων Δεδομένων στη βάση IsoGD για RGB δεδομένα και δεδομένα Βάθους.

Πολυτροπική Μάθηση Πολλαπλών Εργασιών για Δεδομένα RGB και Depth στη βάση NVGesture

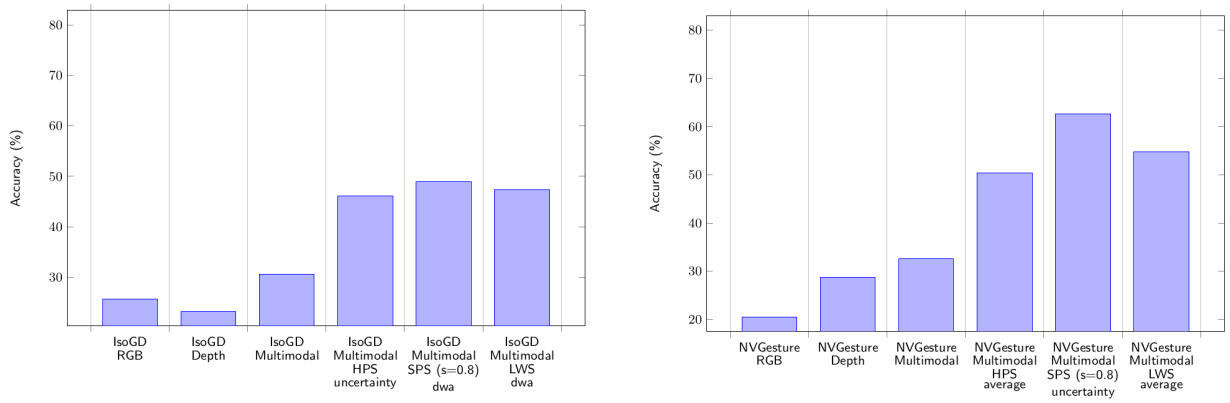
Τα πειράματα πολυτροπικής μάθησης πολλαπλών εργασιών στο σύνολο δεδομένων NVGesture παρουσιάζονται σε αυτή την ενότητα για τα δεδομένα RGB και βάθους. Τα αποτελέσματα που παρουσιάζονται στον παρακάτω πίνακα δείχνουν ότι για όλα τα πειράματα, το προτεινόμενο πλαίσιο μάθησης πολλαπλών εργασιών υπερτερεί του μοντέλου μάθησης μίας εργασίας και του πολυτροπικού μοντέλου. Η καλύτερη απόδοση επιτυγχάνεται όταν η μέθοδος διαμοιρασμού παραμέτρων επιλέγεται να είναι η μέθοδος μερικού διαμοιρασμού παραμέτρων, με την υλοποίηση cross stitch για $\sigma = 0.8$ και τη μέθοδο υπολογισμού του σφάλματος να είναι η αβεβαιότητα στη στάθμιση του σφάλματος, με ακρίβεια 75.51%, η οποία είναι περίπου 10% καλύτερη από το μοντέλο μάθησης μίας εργασίας και το πολυτροπικό μοντέλο.

Μάθηση Πολλαπλών Εργασιών	Σφάλμα	Απόδοση
Ολικός Διαμοιρασμός Παραμέτρων	average	50.41
Ολικός Διαμοιρασμός Παραμέτρων	dwa	46.47
Ολικός Διαμοιρασμός Παραμέτρων	uncertainty	45.23
Cross-Stitch (s=0.2)	average	52,7
Cross-Stitch (s=0.2)	dwa	54.98
Cross-Stitch (s=0.2)	uncertainty	51.87
Cross-Stitch (s=0.8)	average	51.66
Cross-Stitch (s=0.8)	dwa	55.6
Cross-Stitch (s=0.8)	uncertainty	62.66
Εκμάθηση Διαμοιρασμού Βαρών	average	54.77
Εκμάθηση Διαμοιρασμού Βαρών	dwa	50.41
Εκμάθηση Διαμοιρασμού Βαρών	uncertainty	51.24

Table 1.26: Αποτελέσματα ResNet-18 Μάθησης Πολλαπλών Εργασιών και Μάθησης Διαφορετικών Τύπων Δεδομένων στη βάση NVGesture για RGB δεδομένα και δεδομένα Βάθους.

Σύνοψη Αποτελεσμάτων του Μοντέλου Πολυτροπικής Μάθησης Πολλαπλών Εργασιών για Δεδομένα RGB και Depth

Συνοψίζοντας, το προτεινόμενο πλαίσιο μάθησης πολλαπλών εργασιών είναι αποτελεσματικό για τα σύνολα δεδομένων IsoGD και NVGesture, καθώς το μοντέλο μπορεί να μάθει από τους δύο τύπους δεδομένων και να συγχωνεύσει τις πληροφορίες σε μια μόνο πρόβλεψη. Η καλύτερη απόδοση επιτυγχάνεται όταν η μέθοδος διαμοιρασμού παραμέτρων επιλέγεται να είναι η μέθοδος μερικού διαμοιρασμού παραμέτρων, με την υλοποίηση cross stitch για $\sigma = 0.8$ και τη μέθοδο υπολογισμού του σφάλματος να είναι ο δυναμικός μέσος όρος και η αβεβαιότητα στη στάθμιση του σφάλματος για τα σύνολα δεδομένων IsoGD και NVGesture, αντίστοιχα, όπως φαίνεται από το Σχήμα 1.5.3.



(a) (b)

Figure 1.5.3: Πολυτροπική Μάθηση Πολλαπλών Εργασιών για δεδομένα RGB και Βάθους: (a) IsoGD (b) NVGesture

1.6 Σύνοψη και Μελλοντική Εργασία

1.6.1 Συνόψη

Σε αυτή τη διπλωματική εργασία, αξιολογείται η αποτελεσματικότητα της προτεινόμενης μάθησης πολλαπλών εργασιών για την αναγνώριση δράσεων και χειρονομιών. Τα σύνολα δεδομένων που χρησιμοποιούνται για αυτές τα προβλήματα είναι ανθρωποκεντρικά, εστιάζοντας στην αναγνώριση ανθρώπινων δράσεων και χειρονομιών σε βίντεο. Ενώ πολλές εφαρμογές στην πραγματική ζωή απαιτούν την αναγνώριση και των δύο εργασιών, η ερευνητική κοινότητα αντιμετωπίζει αυτά τα προβλήματα ξεχωριστά. Το προτεινόμενο πλαίσιο μάθησης πολλαπλών εργασιών στοχεύει στο να δίνει τη δυνατότητα να αντιμετωπιστούν και τα δύο προβλήματα ταυτόχρονα από μια μόνο αρχιτεκτονική.

Το προτεινόμενο μοντέλο μάθησης πολλαπλών εργασιών αξιολογείται σε διάφορα πειράματα, χρησιμοποιώντας διαφορετικά σύνολα δεδομένων και μεθόδους μάθησης. Η ποικιλία σε διαφορετικά σύνολα δεδομένων δράσεων και χειρονομιών, καθώς και οι διάφορες μέθοδοι διαμοιρασμού παραμέτρων που υλοποιήθηκαν, σε συνδυασμό με τις διάφορες μεθόδους υπολογισμού του συνολικού σφάλματος, παρέχουν μια συνολική αξιολόγηση του προτεινόμενου πλαισίου μάθησης πολλαπλών εργασιών. Τα πειράματα, που πραγματοποιήθηκαν δείχνουν ότι το προτεινόμενο μοντέλο επιτυγχάνει καλύτερη απόδοση σε σύγκριση με τα μοντέλα μονής εργασίας, δείχνοντας τα οφέλη της μάθησης πολλαπλών εργασιών στην αναγνώριση δράσεων και χειρονομιών.

Επιπλέον, αναπτύχθηκε ένα πολυτροπικό μοντέλο μάθησης πολλαπλών εργασιών, όπου τα δεδομένα RGB και Depth μαθαίνονται από κοινού. Ένα μεγάλο πλήθος προβλημάτων στην όραση υπολογιστών χρησιμοποιεί πολυτροπικά δεδομένα, καθώς παρέχουν συμπληρωματικές πληροφορίες. Αν και αυτά τα δεδομένα συνδυάζονται για την τελική πρόβλεψη, οι περισσότερες προσεγγίσεις δεν χρησιμοποιούν ένα κοινό μοντέλο εκπαίδευσης για τα δύο δεδομένα. Το προτεινόμενο μοντέλο στοχεύει στο να δείξει ότι η μάθηση και των δύο δεδομένων από κοινού μπορεί να βελτιώσει την απόδοση του μοντέλου. Αυτό το δίκτυο αξιολογείται στα αντίστοιχα μοντέλα μάθησης μονής εργασίας καθώς και στο πολυτροπικό μοντέλο χρησιμοποιώντας όλες τις μεθόδους διαμοιρασμού παραμέτρων και υπολογισμού σφάλματος που χρησιμοποιήθηκαν για τα πειράματα μάθησης πολλαπλών εργασιών. Τα αποτελέσματα δείχνουν ότι το προτεινόμενο πολυτροπικό μοντέλο μάθησης πολλαπλών εργασιών είναι αποτελεσματικό για τα δεδομένα RGB και Depth, καθώς μπορεί να εκπαιδευτεί ταυτόχρονα για διαφορετικού τύπου δεδομένα και να συγχωνεύσει τις πληροφορίες για να κάνει μια καλύτερη πρόβλεψη.

1.6.2 Μελλοντική Εργασία

Η μάθηση πολλαπλών εργασιών είναι ένα πεδίο της μηχανικής μάθησης με πολλές εναλλακτικές προσεγγίσεις και μεθόδους. Σε αυτή τη διπλωματική εργασία, πραγματοποιήθηκε μια εξερεύνηση των βασικών της μάθησης πολλαπλών εργασιών στο πλαίσιο της αναγνώρισης δράσεων και χειρονομιών. Ωστόσο, υπάρχουν πολλές άλλες πτυχές της μάθησης πολλαπλών εργασιών που θα μπορούσαν να εξερευνηθούν σε μελλοντική εργασία.

Ένα πιθανό μονοπάτι για μελλοντική εργασία είναι να εξερευνηθεί ο συνδυασμός της μάθησης πολλαπλών εργασιών στο πλαίσιο άλλων σχετικών εργασιών, όπως η εντοπισμός δράσεων, η ανίχνευση δράσεων και η τμηματοποίηση δράσεων. Επιπλέον, θα μπορούσαν να χρησιμοποιηθούν και άλλες αρχιτεκτονικές, όπως τα 3D CNNs, για να εξερευνηθούν και οι επιπτώσεις σε διαφορετικές αρχιτεκτονικές δικτύων.

Ένα άλλο πιθανό μονοπάτι για μελλοντική εργασία είναι να εξερευνηθεί η δυνατότητα παράλληλης εκπαίδευσης διαφορετικού τύπου εργασιών καθώς και τύπου δεδομένων. Αυτή η υπόθεση οδηγεί στην ιδέα δημιουργίας ενός μοντέλου μάθησης πολλαπλών εργασιών και πολλαπλών τύπων δεδομένων που μπορεί να μάθει από πολλαπλά σύνολα δεδομένων και πολλαπλές εργασίες ταυτόχρονα.

Επιπλέον, αυτά τα μοντέλα μάθησης πολλαπλών εργασιών χρησιμοποιούν ως είσοδο στην εκτίμηση την εργασία του δείγματος που πρόκειται να προβλεφθεί. Αυτό θα μπορούσε να είναι το επόμενο βήμα στην εξέλιξη αυτών των μοντέλων, καθώς το μοντέλο θα μπορούσε να μάθει την εργασία του δείγματος και να χρησιμοποιήσει αυτές τις πληροφορίες για να κάνει μια καλύτερη πρόβλεψη.

Τέλος, μπορεί να διεξαχθούν περισσότερες έρευνες στο πλαίσιο του πλαισίου μάθησης πολλαπλών εργασιών και πολλαπλών τύπων δεδομένων, εξερευνώντας τους διαφορετικούς μηχανισμούς συγχώνευσης.

Chapter 2

Introduction

2.1 Motivation

Lately, the field of Deep Learning has seen a significant growth in the development of algorithms, as well as remarkable results in various tasks. In computer vision, tasks such as action recognition and gesture recognition have been revolutionized by applying modern techniques, consequently leading to a wide range of applications in various aspects of our daily lives. Surveillance and autonomous driving would not have received the same attention if it was not for the advancements in action recognition, while robot assistants would not have been able to understand human commands if it was not for the growth in gesture recognition algorithms. These two tasks are human-centric as they involve the recognition of human actions and gestures, but still to this day they are treated as separate tasks when training a neural network.

In general, machine learning algorithms are inspired by the way humans think. When a person tackles a problem, knowledge from all previous related situations is used. To achieve this behavior in machine learning, different information sharing techniques between different problems have been developed. One popular method to share information between different tasks is the field of Multi-task Learning. In Multi-task Learning, related tasks are trained jointly, helping each other to perform better and converge faster to optimal solutions.

Since the tasks of action and gesture recognition are related, the question whether they can benefit from a simultaneously trained model is generated. For example, the task of action recognition can benefit from the task of gesture recognition, as gestures are often used to convey information about actions. This observation motivated this work to evaluate the effectiveness of multi-task learning algorithms in the tasks of action recognition and gesture recognition, by training a single model to solve both tasks simultaneously.

2.2 Contributions

In this thesis, the main goal is to evaluate the effectiveness of multi-task learning algorithms in the tasks of action recognition and gesture recognition. To achieve this goal, an extensive evaluation of different multi-task learning architectures is performed, as well as the effects of different multi-task loss calculation methods are tested. This comprehensive evaluation is performed to determine whether the tasks of action recognition and gesture recognition can be trained jointly, and if so, how this can be achieved. Moreover, the hypothesis is extented to evaluate how different modalities can be used in a multi-task learning framework. To be more precise, a multi-task multi-modal learning framework is proposed to handle the modalities of rgb and depth data, in a way that share information during training, while using both modalities for the inference process. In summary, the key contributions of this paper are summarized bellow.

- We evaluate whether multi-task learning can be used to jointly learn the tasks of action recognition and gesture recognition.
- We evaluate the effects of different multi-task learning architectures on the joint training problem of action and gesture recognition tasks.
- We evaluate the effects of different multi-task loss calculation methods on the joint training problem of action and gesture recognition tasks.
- We propose a multi-task multi-modal learning framework to share knowledge between different modalities during training, while using both modalities for the inference process.

2.3 Thesis Outline

The rest of the diploma thesis is structured as follows.

- In Chapter 3 the theoretical background of computer vision, as well as machine and deep learning, is explained.
- In Chapter 4 the field of action and gesture recognition is analyzed, the popular datasets and architectures are presented and related methods are discussed.

- In Chapter 5 the field of multi-task learning is analyzed, the knowledge sharing methods and multi-task loss calculation methods are explained. Moreover, related work on multi-task learning for action and gesture recognition is presented. Lastly, the the proposed idea is stated.
- In Chapter 6 the experiments for this thesis are presented and the results are analyzed.
- In Chapter 7 the conclusions of this thesis are presented, as well as the future work that can be done.

Chapter 3

Theoretical Background

3.1 Computer Vision

Computer vision is the science of endowing computers or other machines with the ability to see. But what exactly does it mean for a computer to see? Seeing is more than the process of recording light in a form that can be played back, like the recording of a video camera. For a biological creature vision is a way to make inferences about the world from the light impinging upon it. So, this field of artificial intelligence aims to mathematically model the processes of visual perception in living beings and to generate algorithms that allow the simulation of these visual abilities using the capacity of computers.

3.1.1 Images and Videos

An image is a visual representation of a concept of the world, that is created through the capture and reproduction of light. The process typically involves the use of imaging devices, such as cameras or sensors, which gather light information from the environment. The capture of light is crucial in forming an image, and various technologies are employed to achieve this. In traditional photography, for example, light-sensitive chemicals on film react to light exposure to create a latent image, which is then developed into a visible photograph. In digital imaging, sensors convert light signals into digital data, which can be processed and stored electronically.

For computers, an image is interpreted as a matrix composed of a finite number of elements, each of which has a position on a Cartesian plane with x,y coordinates, and a value associated with the intensity of the colour of the image at that point. These elements are called elemental points of the image or pixels. Each pixel contains information about colour, intensity, and brightness. For grayscale images, each pixel is represented by a single value, while for coloured images, each pixel is represented by a combination of values, for example for the RGB representation three values are used, which encode its colour as a combination of the amount of red, green and blue, as shown in Fig. 3.1.1.

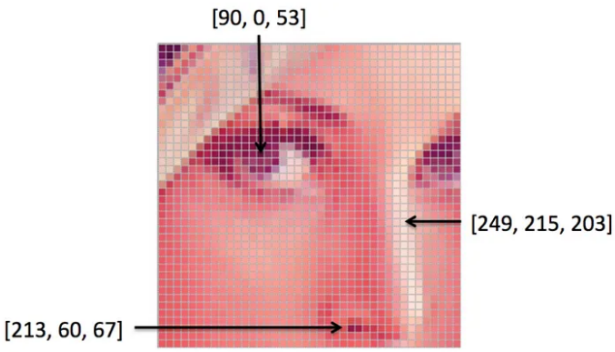


Figure 3.1.1: How computers see an RGB-image (from Lena)

So, computers see an image as a matrix of dimensions (C, H, W), where C denotes the channels used to depict the colour, H denotes the rows and W denotes the column of the vector. This understanding is expanded to the interpretation of videos. To be more precise, videos are a sequence of images. So computers see a video as a matrix of dimensions (D, C, H, W), where the extra dimension D denotes the numbers of images used to construct the video.

3.1.2 Tasks and Applications

Computer vision focuses on solving tasks, with mathematical algorithms, that process visual data. Some of the most common tasks for computer vision are presented below.

- **Image and Video Classification**
  is perhaps the most well-known and widely researched area of computer vision. Image classification is the process of categorizing an entire image into a specific class. The goal is to teach a computer algorithm to recognize and differentiate between different objects or scenes within an image, as shown in

Fig. 3.1.2. Video classification on the other hand, extends the concept of image classification, in order to assign labels to videos, indicating the main content or activity captured in the video. Some of the most popular tasks falling into the category of Video Classification are Action and Gesture Recognition.

- **Object detection**
  aims to detect instances of semantic objects of a certain category, such as humans, buildings, or cars, in images and videos. This task aims to provide information about the presence and location of multiple objects within the scene. The primary goal is to draw bounding boxes around each object and assign a corresponding class label to indicate the type of object, as shown in Fig. 3.1.2.

- **Semantic Segmentation**
  is the partition of the image pixels into subgroups called image segments. Practically it means to delineate the boundaries of the desired area. In the case of instance segmentation, each of the objects is drawn independently, as shown in Fig. 3.1.2. During semantic segmentation, the algorithm segments the images based on the pixels, so a semantic label is assigned to each of the pixels. Thus, this method does not only identify the object itself but marks its boundaries as well.

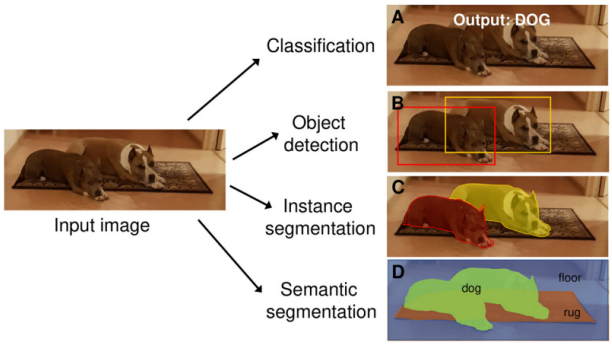


Figure 3.1.2: Comparison of image classification, object detection, instance and semantic segmentation (from [67])

Computer vision applies in many aspects in real life, some of which are described below.

- **Human-computer interaction (HCI)**
  focuses on the design and interaction between humans and computers. The goal of HCI is to create systems and interfaces that facilitate effective and user-friendly communication between humans and computers. Computer vision plays a significant role in enhancing HCI by enabling systems to perceive and interpret visual information, facilitating more natural and intuitive interactions between humans and computers.

- **Augmented and Virtual Reality**
  overlays digital information, such as images, text, or 3D models, onto the real-world environment in real-time. AR enhances the user's perception of the physical world by adding contextual and interactive elements to it. On the other hand, Virtual Reality creates a completely immersive, computer-generated environment that users can interact with. VR typically involves the use of a headset to block out the real world and replace it with a simulated, three-dimensional environment. Computer vision plays a crucial role in both AR and VR by enabling systems to understand and interact with the surrounding environment, track user movements, and enhance the overall user experience.

- **Autonomous vehicles**
  also known as self-driving vehicles, are vehicles that can navigate and operate without human intervention. These vehicles use a combination of sensors, actuators, and advanced algorithms, including computer vision, to perceive their environment, make decisions, and navigate safely.

- **Surveillance and security**
  involve the monitoring, analysis, and protection of people, property, and assets to prevent and respond to potential threats or incidents, tasks that without the help of computer vision would be unfeasible.

3.1.3 Modalities

In computer vision, the term modality refers to the type of data used to represent the input. The most common modalities used in computer vision are listed below [1].

- **RGB**
  or Red Green Blue modality generally refers to images or videos captured by RGB cameras, which aim to recreate what human eyes see. RGB data are usually easy to collect and contain rich appearance information of the captured scene context. However, the use of RGB data is often challenging, owing to the variations of backgrounds, viewpoints, scales of humans, and illumination conditions. Besides, RGB videos have generally large data sizes, which can be beneficial to handling a variety of tasks but also leads to high computational costs when modeling the spatio-temporal context for video data.

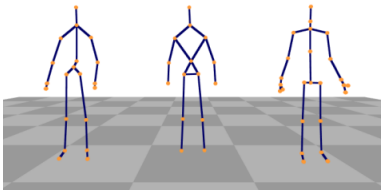
- **Skeleton**
  modalities encode the trajectories of human body joints, which characterize informative human motions, as shown in Fig. 3.1.3b. The skeleton data can be acquired with motion capture systems or by applying pose estimation algorithms on RGB or Depth maps. Human pose estimation, in general, is sensitive to viewpoint variations, while motion capture systems that are insensitive to view and lighting can provide reliable skeleton data.

- **Depth**
  maps refer to images where the pixel values represent the distance information from a given viewpoint to the points in the scene. The depth modality, which is often insensitive to variations of color and texture, provides reliable 3D structural and geometric shape information of human subjects. The essence of constructing a depth map is to convert the 3D data into a 2D image representation. The depth modality can be obtained by using depth cameras or by applying depth estimation algorithms on RGB videos.

- **Infrared**
  modality comes from the collection of RGB data with cameras demand proper illumination to function effectively. Thermal infrared (IR) cameras generate images based on the heat radiated by the body, thereby do not need to rely on external ambient light. This feature of thermal IR cameras makes them particularly suitable for monitoring all day, including night time. Moreover, thermal IR cameras are robust to illumination changes and can capture the heat information of human subjects. However, these type of cameras can be affected by the changes in the temperature and the captured images are usually of low resolution and low contrast [2].

- **Optical Flow**
  is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Optical flow can be estimated by using optical flow estimation algorithms on RGB or Depth maps. Optical flow is a 2D vector field, where each vector is a displacement vector showing the movement of points from first frame to second. Optical flow has many applications in areas like object detection and tracking, video stabilization, and motion detection [3].

(a)



(b)

(c)

(d)

Figure 3.1.3: Modalities Samples. (a) RGB (from [68]) (b) Skeleton ((from [69])) (c) Depth (from [70]) (d) Infrared (from [2])

In computer vision tasks, usually more than one modalities are used, in order to achieve better performance. This process is called Multimodal Fusion. The term Fusion, in general, refers to the process of combining information from multiple modalities or sources, such as outcomes from different classifiers, to improve the overall performance of a system. The most commonly used fusion methods are Early, Mid-Level, and Late Fusion.

- **Early Fusion**
  refers to the process of combining the modalities at the beginning of the algorithm. This allows the network to learn features that capture interactions between the modalities. However, this approach may not allow each modality to independently learn features from its input.
- **Mid-Level Fusion**
  refers to the process of combining the modalities at some intermediate point in the algorithm. This approach allows each modality to independently learn features from its input and capture interactions between the modalities. However, it may be difficult to determine the optimal point in the network to perform the fusion.
- **Late Fusion**
  refers to the process of combining the modalities at the end of the algorithm. This allows each modality to independently learn features from its input, but it may not capture interactions between the modalities.

The choice of the fusion method depends on the specific requirements of the task and the nature of the input data.

3.2 Machine Learning

3.2.1 Definition

In the latest decades, the field of Machine Learning (ML) has engendered remarkable advancements, revolutionizing scientific researches as well as reshaping the way complex problems across various domains are approached. To be more precise, Machine Learning is a branch of artificial intelligence (AI) and computer science, which focuses on the use of data and algorithms to imitate the way that humans learn. It is considered easier to explain the nature of an object or a concept by showing examples, rather than trying to formulate explicit rules that define it. Machine learning builds upon this statement creating algorithms, also known as neural networks or neural models or simply models, that leverage the information given from the data formulating rules to make predictions or decisions [4].

3.2.2 Machine Learning Types

Machine learning can be broadly categorized into three types according to the nature of the data provide and the way algorithms handle this information: supervised learning, unsupervised learning and reinforcement learning. In supervised learning, models are trained on labeled data, learning to map data to their corresponding labels. Unsupervised learning involves discovering patterns and relationships in unlabeled data. Reinforcement learning, while working on unlabeled data, focuses on training models to make sequences of decisions by rewarding desired behaviors.

Supervised Learning

In supervised learning, algorithms are trained on labeled data as input variables, often referred to as features. Specifically, a dataset is provided, that contains input-output pairs, where the input is the data, for example vectors of integers or real numbers, and the output is the corresponding label, for example a type of an animal. The goal of supervised learning is to learn a mapping from inputs to outputs, allowing the algorithm to make predictions or decisions on unseen data, once the model has been successfully trained.

Moreover, according to the type of the output variables of the machine learning algorithm, supervised learning can further be divided into two major categories of problems, classification and regression problems [71].

- **Classification** problems have ouputs, which are a categorical class and can be furthermore distinguished into two categories:
  - binary classification, where the output can be represented only by two classes,
  - multi-label classification, where the output can be represented by multiple classes.
- **Regression** problems have ouputs, which consist of continuous variables and can be similarly distinguished into two categories:
  - univariable regression, where the output is a single continuous variable,
  - multivariable regression, where the output is a set of continuous variables.

Unsupervised Learning

In unsupervised learning, the training data consist of a set of input vectors without any corresponding target labels. Algorithms are trained to discover patterns in these unseen data without any labels or specifications. Unsupervised learning can further be divided into three major categories, according to the problem that the algorithms handle, which are clustering, dimensionality reduction and density estimation.

- **Clustering** aims to find structural information of interest, such as groups of elements that share similar properties.
- **Dimensionality Reduction** aims to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.
- **Density Estimation** aims to determine the distribution of data within the input space.

Reinforcement Learning

In reinforcement learning, algorithms aim to find suitable actions to take in a given situation in order to maximize a reward. Here the learning algorithm is not given examples of optimal outputs, in contrast to supervised learning, but must instead discover them by a process of trial and error. Typically, the problem to be solved in reinforcement learning (RL) is defined as a Markov Decision Process (MDP), meaning there is a sequence of states and actions in which the learning algorithm is interacting with its environment. In many cases, the current action not only affects the immediate reward but also has an impact on the reward at all subsequent time steps. An RL problem includes four elements such as Agent, Environment, Rewards, and Policy. This type of machine learning algorithms can be further splited into Model-based and Model-free techniques, according to the existence of the policy network.

- **Model-based** is the process of inferring optimal behavior from a model of the environment by performing actions and observing the results, which include the next state and the immediate reward.
- **Model-free** techniques does not use the distribution of the transition probability and the reward function associated with MDP.

3.2.3 Machine Learning Terminology

This diploma thesis focuses on supervised learning problems, where the machine learning algorithms can be formulated as an optimization problem. This approach typically involves minimizing a loss function L, that measures the discrepancy between the predicted output $f(x)$ and the true output y. The choice of the loss function depends on the specific task (e.g., regression, classification) and the desired properties of the model. The optimization problem can be stated as follows [72]:

$$J(\theta) = \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i; \theta), y_i) \quad , \tag{3.2.1}$$

where x denotes the input data, y denotes the output data, θ denotes the parameters of the model, $f(\cdot)$ denotes the output predicted by the model, and L denotes the loss function.

So, in order to solve a machine learning problem the following aspects will be used.

Dataset

In Machine Learning, the algorithms perform different calculations, either mapping, searching for similarities or aiming towards a goal. All these calculations require data, that will lead the algorithms take the correct steps, in order to find the desired output. These data could be a variety of different types, for instance numerical or categorical. In order to correctly tune its parameters and perform better on unseen samples, machine learning algorithms require big amount of samples of data. The collection of all the data samples is called dataset. The dataset is further divided into 3 subsets, that do not overlap, with each set used for a different purpose in the algorithm. This division is used to ensure that the model does not simply memorize the exemplar data but learns to generalize to unseen examples. To be more precise, the dataset is divided into the 3 sets of data, the train set, the validation set and the test set.

- **Train set** is used for the model's training and it is the largest subset, as it is used to optimize the model's parameters through optimization methods, that aim to minimize the difference between the predicted and the actual outcomes.
- **Validation set** is used during the training procedure and aims to determine the model's hyperparameters, which are additional parameters that affect the training procedure, as it provides an unbiased evaluation of the model during this phase of the algorithm.
- **Test set**, this set of data is used to evaluate the model, as it simulates a real-world scenario, where the model encounters new instances for which it needs to use to make the desired outcomes.

Loss Functions

As described in the general representation of a machine learning problem in 3.2.1, the purpose is to minimize a loss function, because by this way the learning algorithm adjusts its parameters so that improves the model's performance. The loss function is a scalar function differentiable w.r.t. the model's parameters, that calculates the loss of the model on a given dataset according to the distance between the labels y and the model's outputs $\widehat{y}$. According to the type of machine learning task and the data available, a different loss function can be used. Some of the most common loss functions used are described as follows.

Regression Problems

- **Mean Squared Error (MSE)**
  This loss function calculates the average squared difference of the labels y and the model's outputs $\widehat{y}$. It provides a quantitative measure of how well a regression model is performing. The lower the MSE, the better the model's predictions align with the true values. MSE penalizes larger errors more heavily due to the squaring operation, making it sensitive to outliers. The mathematical expression of MSE is described as follows:

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad . \tag{3.2.2}$$

- **Mean Absolute Error (MAE)**
  This loss function calculates the average absolute difference between the predicted values and the actual values in a regression problem. It provides a measure of the average magnitude of errors without considering their direction (i.e., whether the predictions are too high or too low). A lower MAE indicates better alignment between the predicted $\widehat{y}$ and true values y. MAE is less sensitive to outliers than MSE since it does not involve squaring the errors. The mathematical expression of MAE is described as follows:

$$L_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad . \tag{3.2.3}$$

Classification Problems

- **Binary Cross-Entropy (Log Loss)**
  This loss function calculates the dissimilarity between the predicted probabilities for the positive class and the true binary labels. This loss function is suitable for models that output probabilities and is especially used in logistic regression and neural networks for binary classification. The formula comprises two terms. The first term penalizes predictions when the true label is 1, and the second term penalizes predictions when the true label is 0. The negative sign ensures that the overall loss is minimized. The mathematical expression of BCE is described as follows:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad . \tag{3.2.4}$$

- **Categorical Cross-Entropy**
  This loss function calculates the the dissimilarity between the predicted class probabilities and the true class labels. Categorical Cross-Entropy is an extension of Binary Cross-Entropy for situations where there are more than two classes. The formula calculates the negative log likelihood of the true class, penalizing predictions based on how confident they are in the correct class. The mathematical expression of CCE is described as follows:

$$L_{CCE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \quad . \tag{3.2.5}$$

Optimization Methods

After the selection of a certain machine learning algorithm to perform a desired task in a dataset, this model needs to tune its parameters so that it fits the dataset. In order to fit the weights of the model in the given dataset, the outcome of the loss function needs to be minimized. In machine learning, the way of minimizing the loss function is by iteratively give cetrain values in the model's parameters and update them towards the minimum loss. The methods used to iteratively update the weights of the model are called optimization methods. From the perspective of the gradient information, the optimization methods can be divided into the three categories explained below [73].

- **First-order optimization methods** only require first-order derivatives of the model's parameters.
- **second-order optimization methods** use second-order information, specifically the Hessian matrix, to guide the optimization process. The Hessian matrix represents the second-order partial derivatives of the objective function with respect to the parameters being optimized.
- **heuristic derivative-free optimization methods** seek to overcome the computational challenges involved in dealing with the Hessian matrix, as it is computationally expensive.

The most commonly used first-order optimization methods are mainly based on gradient descent. Gradient descent is an iterative optimization algorithm used to minimize a function by moving in the direction of the steepest descent. The algorithm is based on the observation that if the function is defined and differentiable in a neighborhood of a point, then the function decreases fastest if one goes from that point in the direction of the negative gradient of the function at that point. The gradient descent algorithm is formulated as follows 3.2.6

$$\theta^{(\tau)} = \theta^{(\tau-1)} - \eta \nabla_\theta L(f(x_i; \theta^{(\tau-1)}), y_i) \quad , \tag{3.2.6}$$

where τ denotes the updating iteration, $\theta^{(\tau)}$ denotes the model variable vector θ updated via iteration τ (at $\tau = 0$, vector θ denotes the initial model variable vector), $\nabla_\theta L(f(x_i; \theta^{(\tau-1)}))$ denotes the derivative of the loss function regarding the variable θ based on the complete training set and the variable vector value $\theta^{(\tau-1)}$, obtained from the previous iteration, and η denotes the learning rate in the gradient descent algorithm, which is usually a hyper-parameter with a small value.

Such an iterative updating process continues until convergence, and the variable vector θ achieved at convergence will be outputted as the (globally or locally) optimal variable for the machine learning model.

Evaluation Metrics

Evaluation metrics in machine learning are quantitative measures used to assess the performance of a model on a given task. These metrics help to quantify how well a model generalizes to unseen data of a problem trained to solve. The thing that distinguishes evaluation metrics from loss functions is that evaluation metrics are not used to optimize the model, so the function chosen does not require to be differentiable w.r.t. the model's parameters. The choice of the function used for evaluation depends on the type of machine learning task (e.g., classification, regression) and the specific goals of the model. For instance, some of the most widely used metrics in classification problems are presented underneath.

- **Accuracy**
  measures the proportion of correctly classified instances out of the total instances. Mathematically, accuracy is expressed as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad . \tag{3.2.7}$$

- **Precision**
  measures the accuracy of positive predictions. It is defined as the fraction of correctly predicted positive instances out of all instances predicted as positive. Precision is particularly relevant in situations

where false positives (incorrectly predicted positive instances) are costly or need to be minimized. Mathematically, precision is defined as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad . \tag{3.2.8}$$

- **Recall**
  also known as Sensitivity or True Positive Rate, measures the ability of a model to capture all the positive instances. It is defined as the fraction of correctly predicted positive instances out of all actual positive instances. Mathematically, recall is expressed as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad . \tag{3.2.9}$$

- **F1-score**
  combines precision and recall into a single value. It is the harmonic mean of precision and recall, providing a balance between these two metrics. The f1-score is particularly useful in situations where there is an imbalance between classes. Mathematically, the f1-score is defined as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad . \tag{3.2.10}$$

3.2.4 How Machine Learning Works

The process of developing Machine Learning algorithms may be decomposed into the following five steps.

1. **Data Collection**
   acquiring the relevant data needed for the machine learning task is the first step. The quality and quantity of the data significantly impact the model's performance.
2. **Data Preprocessing**
   cleaning and preparing the data for training. This step involves handling missing values, addressing outliers, and transforming the data into a suitable format for the chosen algorithm. Feature scaling, normalization, and encoding categorical variables are common preprocessing tasks.
3. **Algorithm Selection**
   choosing an appropriate machine learning algorithm based on the nature of the problem (classification, regression) and the characteristics of the data. Different algorithms may be suitable for different types of tasks and data distributions.
4. **Algorithm Training**
   training the selected algorithm on the prepared dataset. During training, the model learns patterns and relationships within the data, adjusting its parameters to minimize the difference between its predictions and the actual target values.
5. **Algorithm Evaluation**
   assess the performance of the trained model on a separate dataset not used during training. The evaluation metrics depend on the nature of the problem (accuracy, precision, recall, F1 score for classification; mean squared error for regression, etc.). The goal is to ensure the model generalizes well to new, unseen data.

3.3 Deep Learning

3.3.1 Definition

Deep learning is a subset of machine learning, which employs transformations and graph technologies simultaneously in order to build up multi-layer learning models, known as deep neural networks (DNNs). In a traditional machine learning algorithm the data are given in the form of vectors of variables (integers or real numbers). So, data like audio signals or images will need some preparation before given as inputs to the model. Specifically, features of interest will need to be extracted in order to form the vectors, that are passed through the model. This approach aims at constructing features from raw data, outside architecture of the model. On the other hand, deep learning algorithms feature extraction is achieved in an automatic way throughout the deep learning algorithms, as illustrated in Fig. 3.3.1. For example, in image processing with deep learning, lower layers of the model may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces, while in traditional machine learning these features will need to be extracted "by hand" using external algorithms.

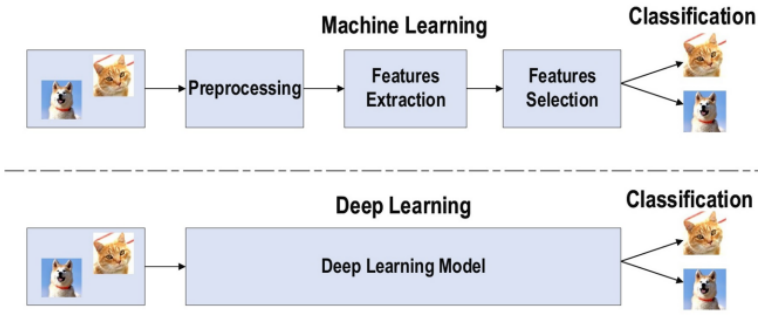


Figure 3.3.1: Traditional Machine Learning and Deep Learning Comparison (from [74])

Even though computationally demanding, Deep Learning has obtained outstanding performances across a variety of challenging tasks, including audio and speech recognition, natural language processing, image and video processing and recognition.

3.3.2 Deep Learning Neural Networks

In Deep Learning, the algorithms used are commonly known as Artificial Neural Networks (ANNs). ANNs are computational models inspired by the structure and function of the human brain, specifically designed to process information and perform tasks such as pattern recognition, classification and regression. The basic building block of ANNs is a processing unit called artificial neuron or perceptron. The perceptron, first introduced by Rosenblatt [5], receives one or more inputs, processes them and produces an output. The output is determined by a mathematical function applied to the weighted sum of inputs, that is passed through an activation function, as illustrated in Fig. 3.3.2.

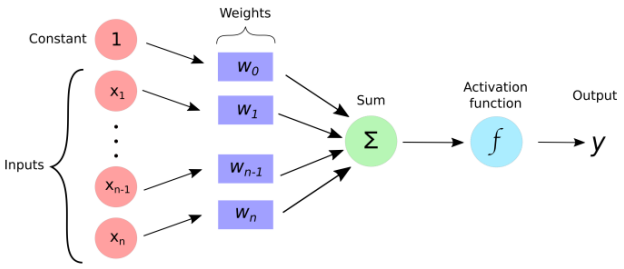


Figure 3.3.2: Neuron or Perceptron (from [6])

The mathematical expression that describes how a neuron calculates its output is defined as follows:

$$y = \phi(\sum_{i=0}^{N}(w_i x_i + b)) \quad , \tag{3.3.1}$$

where x denotes input of the neuron, w denotes weights per inputs, ϕ denotes activation function and y denotes output of the neuron.

In every artificial neuron of the output layer of a Deep Learning model, a function is applied, providing non-linear transformations, which is called **Activation Function**. The output of the nodes in the last layer of a neural network has come from simple linear transformations, thus it is just a polynomial of degree one. So, if an activation function were not to be applied on this polynomial, the neural network will resemble a Linear Regression model with simple yet limited performance. Deep Learning architectures are designed to handle more complicated types of data, such as images, videos or audio signals. For this activation functions are needed so that these neural networks are capable of differentiate complex data, that can not be classified linearly [7]. Moreover, acrivation functions can be classified into saturated, where the output of the activation layer ranges between finite boundaries, and non-saturated, where the output tends to infinite. The non-saturated activation functions have many advantages compared to saturated activation layers. For instance, the non-saturated layers can significantly help in the exploding/vanishing gradient problem of the backpropagation algorithm, which is one of the main problems when training a Deep Learning neural network.

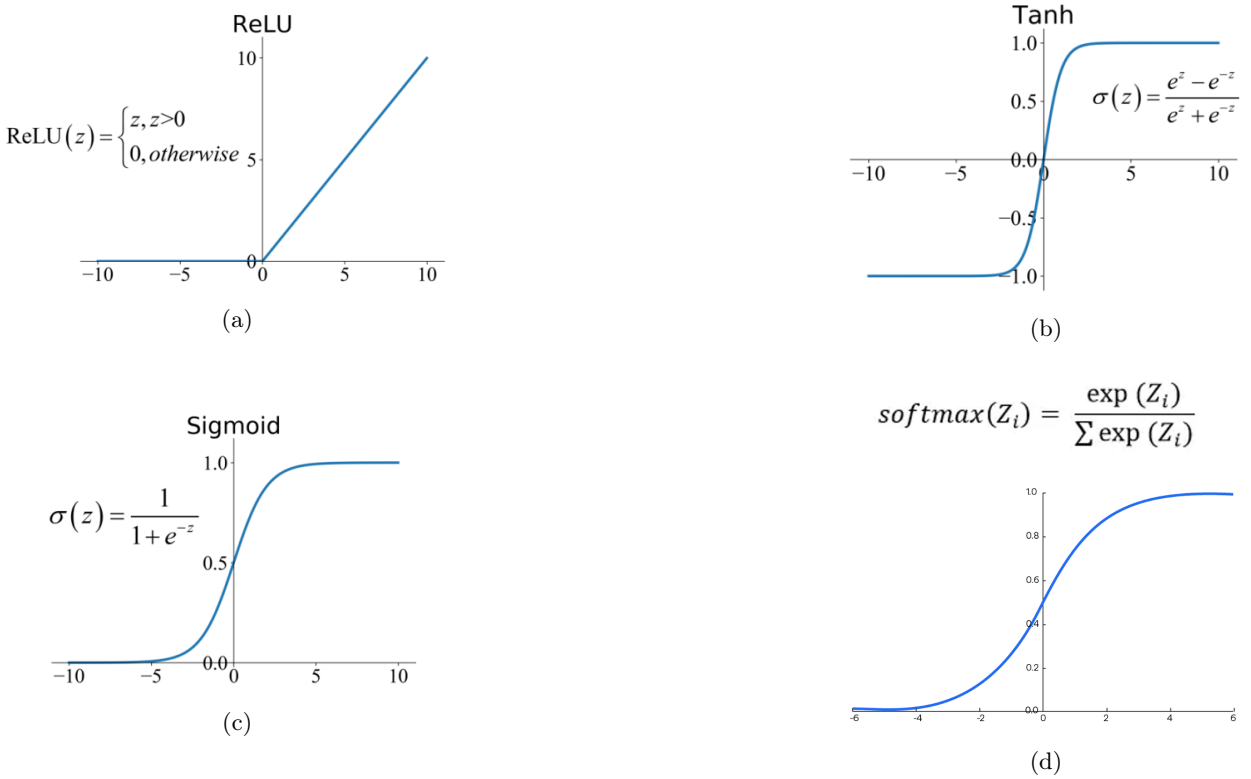


Figure 3.3.3: Activation Functions: (a) ReLU (b) Tanh (c) Sigmoid (d) Softmax (from [75])

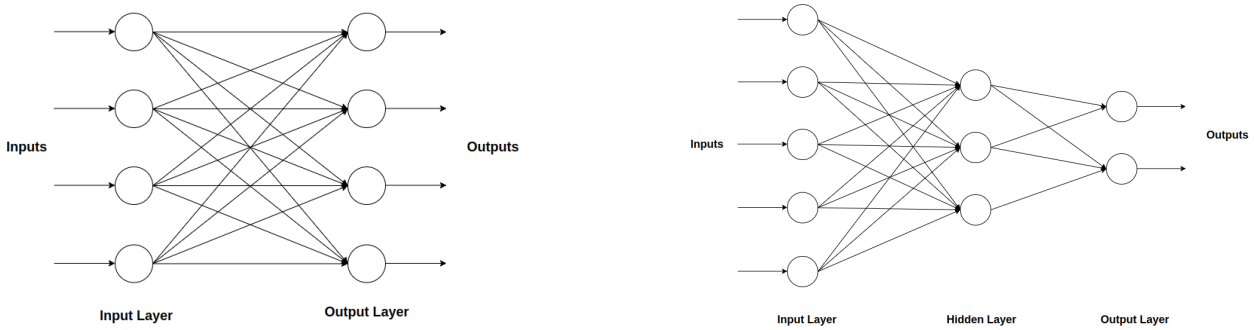
Some of the most commonly used activation functions [8], as shown in Fig. 3.3.3, are explained beneath.

- **Rectified Linear Unit (ReLU)** is a non-saturated activation function that is mainly used to remove any negative values, for example negative gradients when the threshold is at zero.
- **Hyperbolic Tangent (tanh)** is a saturated activation function that is commonly used when a negative gradient is important. It outputs a number in the range of $[-1,+1]$.
- **Sigmoid** is a saturated activation function, where the input is a real number and the output is a number in the range of [0,1]. This function is used when the output of the neural network needs to be continuous.
- **Softmax** produces a discrete probability distribution vector. Though similar to the sigmoid activation function, this function is used when the output of the neural network needs to be categorical.

Artificial Neural Networks consist of a number of neurons connected with each other through synaptic weights or weights in short. So, ANNs are divided into 3 categories according to way the information flows through the layers of neurons: Feedforward Neural Networks (FNNs), Convolutional Networks (CNNs) and Recurrent Neural Networks (RNNs) [9].

Feedforward Neural Networks (FNNs)

In Artificial Neural Networks, where information travels in one direction, from the input layer to the output layer, without forming cycles or loops, are called Feedforward Neural Networks. The neurons inside of a FNN are arranged in the form of layers. Depending on the number of layers, these model can be divided into 2 categories: Single-Layer and Multi-Layer. Single-Layer FNNs, as depticted in Fig. 3.3.4a, consist only of 2 layers of neurons, including the input layer. In this architecture computations are only performed in the output layer. On the other hand, Multi-Layer FFNs, as depticted in Fig. 3.3.4b, consist of more than 2 layers. Specifically, every layer between the input and the output layers is called hidden layer and the neurons in these layers hidden neurons. The purpose of hidden layers is to help the model decompose more complex representations of the input data.



(a) (b)

Figure 3.3.4: (a) Single-Layer Feedforward Neural Network. (b) Multi-Layer Feedforward Neural Network (from [10])

In the exemplar Fig. 3.3.4 above, all neurons in one layer are connected to the next layer and so on for the next layer, thus forming a Fully-Connected Neural Network, often referred to as Multi-Layer Perceptron. If some of the synaptic connections were missing, the network would be called as Partially-Connected Neural Network.

Convolutional Neural Networks (CNNs or ConvNets)

Convolutional Neural Networks, although falling into the category of Feedforward Neural Networks, where information travels only in one direction, without creating cycles, these models are specifically designed for tasks involving grid-like structures efficiently, such as images. The capability of automatically discovering

essential features from the input without the need for external feature extractors makes them more powerful than a traditional network. Moreover, as shown in Fig. 3.3.5 ConvNets consist of a series of layers, as shown in 3.3.5, that makes them capable of converting the original input layer by layer into class scores for classification and other purposes.

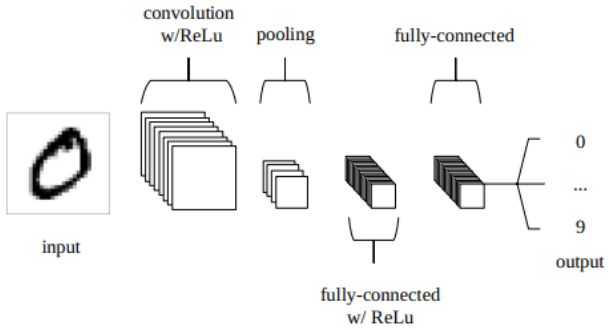


Figure 3.3.5: A simple CNN architecture, comprised of just 5 layers (from [11])

Convolutional Layer A Convolutional Layer is used to extract features in convolutional neural networks. In CNNs, different types of dimensions are used, depending on the type of data the model is trained on. For example, 2-D Convolutional Layers are used to capture spatial information, while 3-D Convolutional Layers can be applied on videos (sequence of images), to capture spatiotemporal information as shown in Fig. 3.3.6.

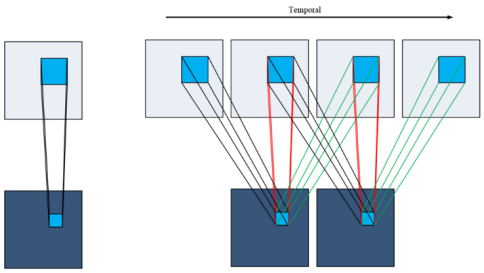


Figure 3.3.6: 2-D Convolutional Layer (left). 3-D Convolutional Layer (right). (from [12])

Given an image with dimensions (C, H, W), where C denotes the channels of the image, H denotes the height and W denotes the width of the image, and a 2-D filter with dimensions (F, F), where F denotes the spatial dimensions of the filter, the output produced will be the result of the convolution operation between the input and the filter. The filter slides through the input, and the dot product between the filter's weights and the input's spatial values is calculated, resulting into a 2-D output, called activation map. Then more filters are applied to the input, producing more activation maps. These activation maps are stacked to produce a 3-D output.

$$\begin{aligned} H_{out} &= \frac{H_{in} + 2 \cdot padding\_height - filter\_height}{stride\_height} + 1 \quad . \\ W_{out} &= \frac{W_{in} + 2 \cdot padding\_width - filter\_width}{stride\_width} + 1 \quad . \end{aligned} \tag{3.3.2}$$

So the output produced will have dimensions (N, H_{out}, W_{out}), where N equals to the number of filters applied and the spatial dimensions are calculated as described in Eq. 3.3.2.

In the case of videos, with dimension (C, D, H, W), where D denotes the number of frames of the video, each sampled cube is convoluted by a 3-D filter with dimensions (F, F, F) and the convolutional result is obtained by summing the total partial sum, as illustrated in Fig. 3.3.7. In 3-D convolution, each feature map in a cube is connected to multiple adjacent consecutive frames, so temporal information can be extracted.

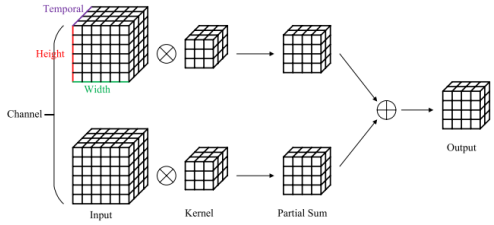


Figure 3.3.7: Convolutional Layer applied on 3-D input (from [12])

The output produced will have dimensions $(N, D_{out}, H_{out}, W_{out})$, where N equals to the number of filters applied and the spatiotemporal dimensions are calculated as described in Eq. 3.3.3.

$$\begin{aligned} D_{out} &= \frac{D_{in} + 2 \cdot padding\_frames - filter\_frames}{stride\_frames} + 1 \quad , \\ H_{out} &= \frac{H_{in} + 2 \cdot padding\_height - filter\_height}{stride\_height} + 1 \quad , \\ W_{out} &= \frac{W_{in} + 2 \cdot padding\_width - filter\_width}{stride\_width} + 1 \quad , \end{aligned} \tag{3.3.3}$$

where $filter\_size$ (representing the cubic filter) is the size of the window of the layer, *stride* is the step the window takes to perform the next transformations and *padding* is an addition of values, most usually zeros, in all dimensions of the input.

Pooling Layer In most cases, the convolutional layer is followed by a pooling layer. This type of CNN component performs transformations in order to reduce the dimensionality of the feature maps, produced from the previous step, as well as the network's parameters. To be more precise, it downsamples values of the vector, in spatial dimensions for images and in spatiotemporal dimensions for videos, leading to dimensionality reduction, as shown in Fig. 3.3.8.

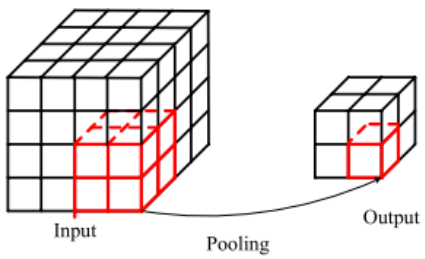


Figure 3.3.8: Pooling Layer applied on 3-D input (from [12])

For example, for a video input to the model with dimensions $(C, D_{in}, H_{in}, W_{in})$, with C representing Channels, D_{in} the input's Depth, meaning the total number of frames and H_{in}, W_{in} representing the spatial dimensions Height and Width, then the output's dimensions will be :

$$\begin{aligned} D_{out} &= \frac{D_{in} - filter\_frames}{stride\_frames} + 1 \quad , \\ H_{out} &= \frac{H_{in} - filter\_height}{stride\_height} + 1 \quad , \\ W_{out} &= \frac{W_{in} - filter\_width}{stride\_width} + 1 \quad . \end{aligned} \tag{3.3.4}$$

In practice, given a greyscale image, with dimensions of (1, 4, 4) and a 2x2 pooling kernel with stride 2, the kernel will perform the downsampling across the coloured blocks in Fig. 3.3.9. Some of the most popular pooling operations are average pooling and max pooling, as described below.

- **Average Pooling** calculates the average value across all elements in the selected window. This method operates as a smoothing filter, but simultaneously blurs the input.
- **Max Pooling** keeps the max value across all elements in the selected window. This method emphasizes the most relevant information.

(a) (b)

Figure 3.3.9: (a) Average Pooling Operation. (b) Max Pooling Operation. (from [13])

Pooling layers besides reducing the computational cost of the network aim to control overfitting, a very common problem faced in Deep Learning, where the model's weights are tuned to handle samples only from the training set and cannot generalize at unseen data will a good performance.

Batch Normalization Layer When training a Deep Learning neural network samples of the dataset are passed through the model in groups, called mini-batches, instead of passing them one at a time. This approach is more practical in the training process, as the gradient of the loss over a mini-batch is an estimate of the gradient over the whole training set, which quality of estimate improves as the batch size increases, and moreover computation over a batch can be much more efficient, due to the parallelism afforded by the modern computing platforms. Even though this approach might be useful, if the distribution of non-linearity inputs is not stable then it is possible for the optimizer to get stuck in the saturated regime and the training would slowed down. Solution to this problem provides the Batch Normalization algorithm [14] described in Eq. 3.3.5, which normalizes the input by keeping zero mean and unit standard deviation. To be more precise,

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=0}^{m} x_i \quad , \\ \sigma &= \frac{1}{m}\sum_{i=0}^{m}(x_i-\mu)^2 \quad , \\ \hat{x_i} &= \frac{x_i-\mu}{\sqrt{\sigma^2+\epsilon}} \quad , \\ y_i &= \gamma x_i + \beta \quad , \end{aligned} \tag{3.3.5}$$

where x, y are input and output of batch normalization layer for a specific m mini-batch, μ is the mean of the mini-batch, σ is the variance of the mini-batch, β, γ are scaling and shifting transformations variables, which are learned during training and ϵ is a small constant to avoid division by zero [15].

Fully Connected Layer In a Deep Learning architecture the last layers, also known as head of the model, are transformations which map the neurons of the previous step with the number of the outputs of the model and are called Fully Connected layers. These layers take as input 1-D vectors and produce 1-D vectors in the output. So the output of the previous step, the last hidden layer of the model, has to be converted into a 1-D vector. For instance, the output of the last hidden layer of a model that takes images as inputs will have to convert a 3-D vector to 1-D, where a model that takes videos as inputs will have to convert a 4-D vector to 1-D. This transformation is called flattening and it leads the first fully connected layer of the model, as illustrated in the Fig. 3.3.10.

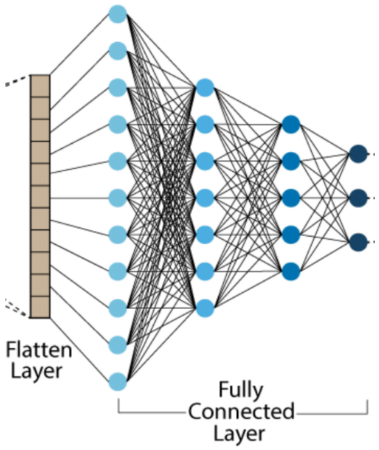


Figure 3.3.10: Fully Connected Layer (from [16])

Recurrent Neural Networks

In Artificial Neural Networks, where the connections between the weights create loops are called Recurrent Neural Networks (RNNs). This design of the RNNs allows them to maintain a memory of previous inputs, making them capable of handling sequential data, like audio signals, representing sequences of words, or videos, representing sequences of images. However, standard recurrent networks are dealing with the issue of vanishing gradients, where the gradients of the loss function become very small, as they are backpropagated through the network, and exploding gradients, where the gradients of the loss function become very large, as they are backpropagated through the network, making the learning of long data sequences challenging. Solution to this problem provide RNN varients like Long Short Term Memory (LSTM) models and Gated Reccurent Units (GRUs).

RNNs Vanilla RNNs take as input x_t a sequence of data and returns a sequence of hidden states h_t and outputs y_t. The hidden state h_t is calculated by the previous hidden state h_{t-1} and the current input x_t, while the output y_t is calculated by the current hidden state h_t. The hidden state h_t is passed to the next step, where it is used to calculate the next hidden state h_{t+1}, and so on. This way, RNNs remember the context while training.

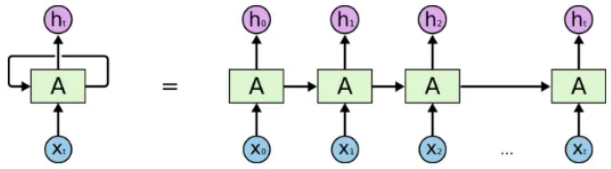


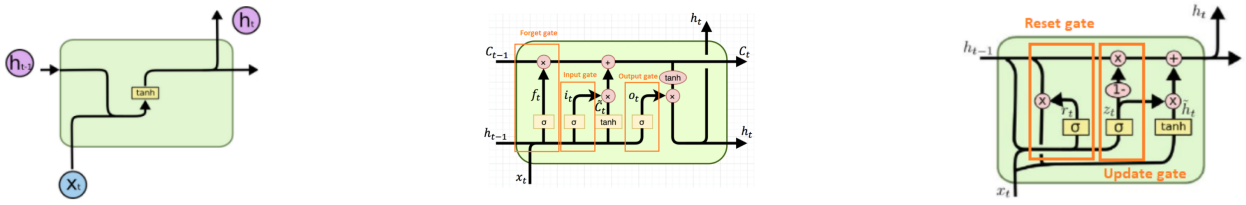
Figure 3.3.11: RNNs neurons unfolded per iterations (from [17])

As shown in Fig. 3.3.11, the RNNs cells unfolded according to the number of iterations, show how the input x_t is passed to the hidden state h_t, which is then passed to the output y_t. This mechanism can be formulated with the following expressions:

$$\begin{aligned} h_t &= tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \quad , \\ y_t &= W_{hy}h_t + b_y \quad , \end{aligned} \tag{3.3.6}$$

where t denotes the state of the model, W_{hh} denotes the weight at previous hidden state, W_{hx} denotes the weight at current input state, W_{hy} denotes the weight at the output state and $tanh(\cdot)$ denotes the activation function, that implements a non-linearity that squashes the activations to the range[-1.1].

Long Short Term Memory (LSTM) Long Short Term Memory (LSTM) models consists of a memory cell, that can store data for longer periods than vanilla RNNs, while the flow of information in and out of this cell is managed by different gates: Input Gate, Forget Gate and Output Gate, as shown in Fig. 3.3.12b.



(a) (b) (c)

Figure 3.3.12: (a) RNN cell. (b) LSTM cell. (c) GRU cell (from [18])

The *Input Gate* determines which information should enter the cell state, to modify the memory. Specifically, the sigmoid function decides which values to let through 0, 1 , while tanh function gives weightage to the values which are passed deciding their level of importance ranging from -1 to 1. The calculates performed in the input gate can be mathematically expressed as follows:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad , \\ \widetilde{C}_t &= tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad , \end{aligned} \tag{3.3.7}$$

where i_t denotes the input gate, $\widetilde{C}_t$ denotes the candidate value to be added to the cell state, W_i, W_C denote the weights of the input gate and the candidate value, h_{t-1}, x_t denote the previous state and the content input and b_i, b_C denote the biases of the input gate and the candidate value.

The *Forget Gate* determines what information from the previous state cell will be memorized and what information will be removed, as it is no longer useful. To make the decision a sigmoid function is used, which looks at the previous state (h_{t-1}) and the content input (x_t) and outputs a number between 0 (omit this) and 1 (keep this) for each number in the cell state C_{t-1}. The calculates performed in the forget gate can be mathematically expressed as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad , \tag{3.3.8}$$

where f_t denotes the forget gate, W_f denotes the weights of the forget gate, h_{t-1}, x_t denote the previous state and the content input and b_f denotes the biases of the forget gate.

Regulates the The *Output Gate* controls the amount of information from the cell state that is exposed to the hidden state and subsequently to the output. To be more precise, the sigmoid function decides which values to let through 0,1 , while tanh function gives weightage to the values which are passed deciding their level of importance ranging from -1 to 1 and multiplied with output of sigmoid. The calculates performed in the output gate can be mathematically expressed as follows:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad , \\ h_t &= o_t * tanh(C_t) \quad , \end{aligned} \tag{3.3.9}$$

where o_t denotes the output gate, W_o denotes the weights of the output gate, h_{t-1}, x_t denote the previous state and the content input, b_o denotes the biases of the output gate, h_t denotes the output of the LSTM cell and C_t denotes the cell state.

Gated Reccurent Units (GRUs) Another RNN variant that handles the vanish gradient problem is the Gated Reccurent Units (GRUs), which uses gating methods to control and manage information flow between cells in the neural network. The GRU's structure, as shown in Fig. 3.3.12c, enables it to capture dependencies from large sequences of data in an adaptive manner, without discarding information from earlier parts of the sequence. Thus GRU is a slightly more streamlined variant that often offers comparable performance and is significantly faster to compute [9].

3.3.3 How Deep Learning Works

In Deep Learning, the parameters of the models are update with similar approach as a traditional Machine Learning model, with the help of gradient descent. As mentioned before, in machine learning the gradient descent algorithms iters through the entire trainset in order to update the model's weights. In deep learning, networks can have a tremendous number of parameter, for example hundreds of millions, as the depth of the model increases, as well as the complexity of the data given as input becomes more advanced. So, the standard gradient descent algorithm is computationally unfeasible for a deep learning network. Solutions to this problem provide complex optimization methods, which alter the standard gradient descent method.

Stochastic Gradient Descent (SGD)

This variant of Gradient Descent instead of using the entire dataset for each iteration, only a single random training example, also known as a small batch, is selected to calculate the gradient and update the model parameters. This random selection introduces randomness into the optimization process, hence the term "stochastic". Some of the advantages of Stochastic Gradient Descent are its speed and memory efficiency. On the other hand, SGD might make the optimization process less stable and lead to oscillations around the minimum. Also, it may require more iterations to converge to the minimum, since it updates the parameters for each training example one at a time. Moreover, the choice of learning rate can be critical in SGD since using a high learning rate can cause the algorithm to overshoot the minimum, when the algorithm passes the minimum and starts to move away from it, while a low learning rate can make the algorithm converge slowly. Finally, due to the random updates, SGD may not converge to the exact global minimum and can result in a suboptimal solution, although this can be mitigated by using techniques such as learning rate scheduling and momentum-based updates.

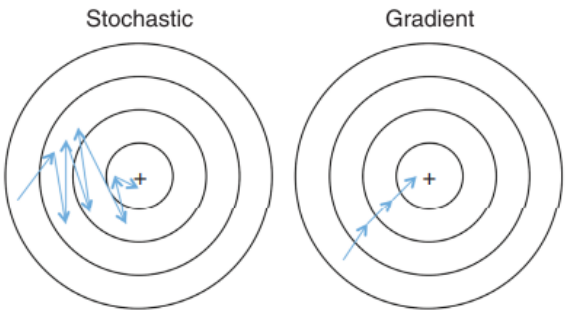


Figure 3.3.13: Comparison of Stochastic Gradient Descent (left) with Gradient Descent(right) (from [76])

Adaptive Moment Estimation (Adam)

Adam is an optimization algorithm that can be used instead of the stochastic gradient descent, as SGD maintains a single learning rate for all weight updates and the learning rate does not change during the training procedure [19]. Specifically, Adam is a combination of two other extensions of stochastic gradient descent, AdaGrad and RMSProp, in order to leverage the benefits of both algorithms, as described below.

- **Adaptive Gradient Algorithm (AdaGrad)**
  maintains a per-parameter learning rate that improves performance on problems with sparse gradients.

- **Root Mean Square Propagation (RMSProp)**
  maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight.

Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

Backpropagation

In Deep Learning, the essence of the training process is the optimization of the model's parameters, so that the loss function is minimized. This goal is achieved by using a complex optimization algorithm, such as SGD or Adam, to update the weights of the model iteratively. In order to update the weights, the gradient of the loss function with respect to the model's parameters needs to be calculated. This is done by using the backpropagation algorithm, which consists of two stages, the forward pass and the backward pass.

- **In Forward Pass**
  data flows through the network in the forward direction, from the input layer to the output layer. During this process, the output of each layer is calculated, based on the current weights of the model. The output of the last layer is the output of the network.
- **In Backward Pass**
  data flows through the network in the backward direction, from the output layer to the input layer. During this process, the error of each layer is calculated, based on the output of the previous layer. The error of the first layer is the error of the network. The error of each layer is then used to calculate the gradient of the loss function with respect to the weights of the layer. This gradient is then used to update the weights of the layer. This gradient can be used with the chain rule as well as with the mathematical expressions that describe each layer of the model, to formulate the partial derivatives of the loss function with respect to the model's parameters, in order to update them.

Chapter 4

Related Work

4.1 Action and Gesture Recognition

4.1.1 Action Recognition

Recognition of human actions in videos is a challenging task which has received a significant amount of attention in the research community. An action refers to a specific behavior or activity performed by a subject, usually a person in a video. For example, actions could include walking, jumping, drinking, reading, playing an instrument etc. Each of these actions involves a specific pattern of movement over time, which can be recognized by analyzing the sequence of video frames.

As in computer vision in general, action recognition is useful in a wide range of applications, including video surveillance, human-computer interaction, and content-based video retrieval. Action recognition is particularly important for understanding human behavior in videos, and for enabling machines to understand and respond to human actions.

4.1.2 Gesture Recognition

Alongside action recognition, gesture recognition is another important task in the field of computer vision. Gestures are defined as a form of non-verbal communication in which visible bodily actions are used to communicate particular messages. Gestures can be used to convey a wide range of information, including emotions, intentions, and commands. In the context of computer vision, gesture recognition refers to the task of identifying and interpreting human gestures in video data. For example, gestures could include hand movements, facial expressions, body postures, and other non-verbal cues. This thesis focuses on the recognition of hand gestures in videos. Hand gestures examples could include waving, pointing, clapping, and other hand movements.

Gesture recognition has a wide range of applications, including human-computer interaction, sign language recognition, and virtual reality.

4.1.3 Traditional Learning for Action and Gesture Recognition tasks

In the early stages of computer vision, where Deep Learning architectures were not yet introduced, the process of solving problems, such as image video classification, included first the detection of features of interest in the visual data, after that these features were extracted, then a map between the features and representations was created and finally a classifier was used to discriminate them between different classes [20], as shown in Fig. 4.1.1.

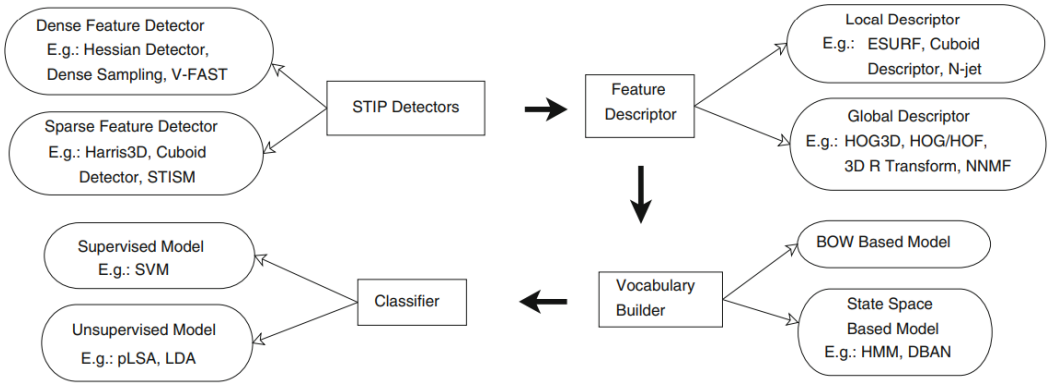


Figure 4.1.1: Flowchart of traditional recognition of actions and gestures (from [20])

To begin with, a feature of interest may be a specific structure in the image such as a corner, an isolated point (where intensity is maximum or minimum), an end point of a line, point of a curve (where curvature is maximum), or blobs (regions that are either brighter or darker than the surrounding area). For videos the features are also referred to as spatio-temporal interest points (STIP).

For the detection of features, (STIP) detectors are used, which can be broadly categorized into two types, dense and sparse.

- **Dense Feature Detectors**
  are algorithms that identify features from an image at every pixel or in a dense grid across the image. This approach ensures that features are detected uniformly across the entire image, providing a comprehensive representation of the content visualized. One of the most popular dense feature detectors is Hessian Detector [21] which relies on the Hessian matrix, which is a square matrix of second-order partial derivatives of the image intensity function, providing information about the curvature of the image intensity surface, thus making it useful for detecting regions with significant intensity variations such as blobs.

- **Sparse Feature Detectors**
  are algorithms that identify features from an image at specific locations. This approach is more computationally efficient than dense feature detection, as it only processes a subset of the image. One of the most popular sparse feature detectors is Harris Detector [22], where it is used to detect corners in an image, as well as its extension to videos, the Harris3D [23]. Corners are points in an image where the intensity changes significantly in multiple directions. So, the gradient of the image intensity function is calculated in both the x and y directions, that can be used to describe the likelihood that a pixel is part of a corner. Pixels with large gradient magnitudes in both directions are good candidates for being corners.

For the extraction of features, feature descriptors are used which represent vectors that provide a detailed characterization of the local appearance around a detected feature point. They encode the essential information about the surroundings of the feature, so that it is robust to changes in scale, rotation, illumination, and viewpoint. Feature descriptors can be divided into two categories, local and global.

- **Local Descriptors**
  capture only local or static information, for example the color or the texture of the image. The shape or edge relevant data of objects are used for determining the local features. Some of the most popular local descriptors are the Scale Invariant Feature Transform (SIFT) [24] and the Speeded Up Robust Features (SURF) [25].

  1. The SIFT descriptor transforms an image into a collection of local feature vectors, each of which is invariant to any scaling or rotation, because it relies on the distribution of the image gradient in local neighbourhoods. These features are used to identify matching objects in different views of a scene. The main advantage of SIFT is that it is robust against distortion, addition of noise and change in illumination.

  2. The SURF descriptor is based on similar properties with SIFT but is faster and more robust to image transformations. Specifically, SURF uses a different method to calculate the gradients, as it fixes as reproducible orientation based on information from a circular region around the point of interest. Then, it constructs a square region aligned to the selected orientation and extracts the features from it.

- **Global Descriptors**
  capture the dynamic information of the visual data, for example scale changes, illumination changes, speed variation or phase variation. The global information points towards the description of flow or motion of a video. Some of the most popular global descriptors are the Histogram of Oriented Gradients (HOG) [26] and the Histogram of Optical Flow (HOF) [27].

  1. The HOG captures the structure and appearance of objects by computing the distribution of gradient orientations in localized portions of an image. It provides a compact and informative representation that is particularly useful for identifying and recognizing objects within images.

  2. The HOF represents the distribution of motion directions and magnitudes within localized regions of a video frame, providing a compact summary of the motion patterns.

After the features are extracted, they are used to create a map between the features and their representations. This can be done other by using a Bag of Words (BoW) [28] or a state space model.

- **Bag of Words (BoW)**
  is a feature encoding method inspired by natural language processing. In the context of computer vision, BoW is often referred to as the Bag of Visual Words (BoVW). This method encodes an image by replacing each local feature with the nearest visual word from a predefined vocabulary. The image is then represented as a histogram of visual word occurrences, capturing the distribution of features in a simple yet efficient manner.

- **State Space Model**
  is mathematical frameworks used to model dynamic systems, where the system's state evolves over time and is observed through measurements. One of the most commonly used state space models is the Hidden Markov Model (HMM) [29]. An HMM is a statistical model used to describe a system where the state of the system is hidden and cannot be directly observed, but can be inferred from observable outputs. HMMs are particularly useful for modeling sequential data where the underlying states evolve over time according to probabilistic transitions, and each state generates observable outputs with certain probabilities.

Finally, a classifier is used to discriminate between different classes. According to the problem that these features will be used, the algorithms can be divided into two categories, supervised and unsupervised.

- **In Supervised Learning**
  algorithms are trained on the set of features extracted in the previous steps, and in order to distribute the features into different classes. One of most commonly used classifier in the traditional computer vision is the Support Vector Machines (SVM) [30]. SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space to separate different classes of data points. The goal of SVM is to find the optimal hyperplane that maximizes the margin between the closest points of the classes, known as support vectors.

- **In Unsupervised Learning**
  algorithms like the Linear Discriminant Analysis (LDA) [31] and the Principal Component Analysis (PCA) [32] are used to reduce the dimensionality of the feature space and to find the most important features that can be used to discriminate between the classes.

Later, these hand-crafted mathematical models for feature extraction were replaced by Deep Learning architectures, which could take as input raw data, like images and videos, and extract features due to their multi-layer structure, while found to be more powerful and achieved better performance. To be more precise, as depticted in Fig. 4.1.2, the first layers of a CNN extract low-level features, like edges, corners, and blobs, while layers in the middle extact mid level features, like eyes, noses, and mouths and the last layers extract high-level features, like facial structure. So, the deeper the layer, the more abstract the features extracted.

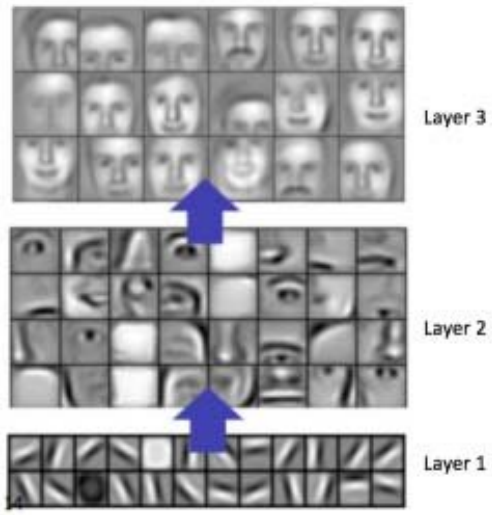


Figure 4.1.2: Features extracted in each layer of a CNN. The input is the 3-D vector representation of an image (from [33])

4.2 Baseline Architectures

Deep Networks replaced handcrafted feature descriptors and were used widely to extract features from images, videos and other types of data. The standard structure of a deep neural network, that is used to extract features from the data, is called backbone. The backbone is usually a Convolutional Neural Network (CNN) architecture, which structure can be altered to fit the needs of the task or so that new methdologies can be proposed. Action and Gesture Recognition are two different tasks, both falling into video classification category of computer vision tasks. So, the baseline neural networks used for each task are similar, though several architectures have been proposed for better performance in each task. Specifically, in action recognition architectures can perform well leveraging the information provided by RGB frames, even with low quality video samples, while in gesture recognition models often require high resolution data, as well as additional information, given from depth or skeleton data, due to the similarity between gesture classes. In this section, some of the most popular baseline neural networks used for video classification tasks are presented.

4.2.1 3D Convolutional Network (C3D)

This architecture is based on the idea that 3D convolutional layers can be used to extract spatio-temporal features from video data. 3D Convolutional Network, or C3D, was proposed by Tran et al. in [34], as an alternative to 2D Convolutional Networks. In C3D models, convolution and pooling operations are performed spatio-temporally while in 2D ConvNets they are done only spatially. To be more precise, 2D convolution applied on an image will output an image, 2D convolution applied on multiple images also results in an image. Hence, 2D ConvNets lose temporal information of the input signal right after every convolution operation. Only 3D convolution preserves the temporal information of the input signals resulting in an output volume, a phenomenon also applicable for pooling operations.

As depticted in Fig. 4.2.1, the proposed C3D has 8 convolution layers, 5 pooling layers, 2 fully-connected layers and a softmax loss layer to predict action labels. All 3D convolution kernels are of size $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. All pooling layers are of size $2 \times 2 \times 2$ with stride $2 \times 2 \times 2$, except for the first one which has kernel size $1 \times 2 \times 2$ and stride $1 \times 2 \times 2$ with the intention of preserving the temporal information in the early phase. Each fully connected layer has 4096 output units.

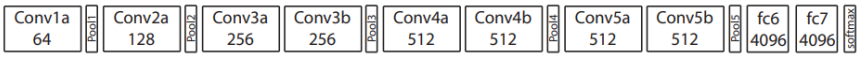


Figure 4.2.1: C3D architecture (from [34])

In the original paper, the authors evaluated the proposed architecture on two datasets, Sports-1M [35] and UCF-101 [36]. Sports-1M is a large-scale dataset with 1.1 million videos and 487 classes of sports videos, while UCF-101 is a smaller dataset with 101 classes, as mentioned before. The authors used the first 1 million videos of Sports-1M for training and the rest for testing. For UCF-101, they used the around 9,000 videos for training and the remaining videos for testing. The classification model consisted of the C3D architecture, used to extract features from the videos, and a linear SVM classifier, used to classify the videos. The model achieved state-of-the-art accuracies on both datasets, at the time, showing its ability to capture spatio-temporal features from videos.

This architecture quickly became popular in action recognition tasks and a variety of methods and state-of-the-art models used it as a backbone. The popularity of this model led Zhang et al. in [37] to evaluate it on the EgoGesture dataset, a dataset with 83,000 videos of 50 gesture classes, proving its applicability in gesture recognition tasks.

4.2.2 Two Stream Network (TSN)

A video is a sequence of frames, thus it can be decomposed into spatial and temporal components. The spatial part, in the form of individual frame appearance, carries information about scenes and objects depicted in the video. The temporal part, in the form of motion across the frames, conveys the movement of the observer, the camera, and the objects. Following this hypothesis, Simonyan and Zisserman in [38] proposed the Two-Stream Network (TSN) architecture, which consists of two separate streams, one for spatial and one for

temporal information, as shown in Fig. 4.2.2. Each stream is implemented using a deep ConvNet, followed by a softmax activation, the outputs of which are combined by late fusion.

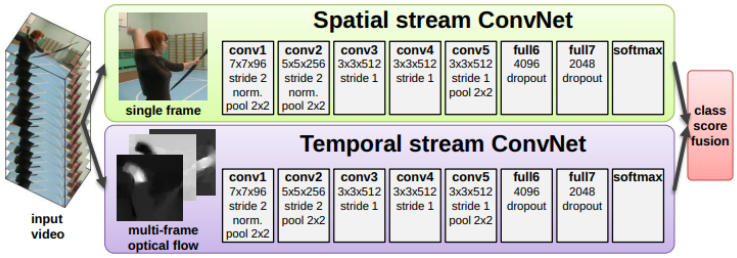


Figure 4.2.2: Two-Stream architecture (from [38])

In the original paper, the authors implemented the TSN architecture using RGB frames, for the spatial stream, and optical flow frames, for the temporal stream. Specifically, in the spatial stream, the authors performed action recognition from still images, in contrast to the temporal stream, which used stacked optical flow displacement fields between several consecutive frames to capture motion information. The authors evaluated the proposed architecture on two datasets, UCF-101 and HMDB-51, and achieved promising results in comparison with the state-of-the-art results, at the time, on both datasets.

Moreover, Chen et al. in [39] proposed a variant of the TSN architecture for gesture recognition tasks. In this work, the authors used a dual encoder to handle RGB data and keypoint heatmaps. The dual encoder consists of two separate encoders, one for RGB data and one for keypoint heatmaps, which are fused at the end of the network. An encoder is a neural network architecture that takes an input and outputs a feature representation of the input.

4.2.3 Residual Network (ResNet)

It was observed from the early stages of machine learning, that Deep neural networks extract more complex features in the later layers, as mentioned before. The question that naturally generated from this observation was whether the levels of features can be enriched by the number of stacked layers, improving the performance of the network as more layers are added. However, it was observed that the performance of the network saturates or even degrades as the number of layers increases. This phenomenon is known as degradation problem. Unexpectedly, this problem is not caused due to overfitting and adding more layers to a suitably deep model leads to higher training error. To address the degradation problem Kaiming He et al. proposed the use of a deep residual learning framework, known as Residual Block in [40], as shown in Fig. 4.2.3. The residual block is able to transfer the data from one layer to another, without any transformation, using skip connections, thus keeping the information intact.

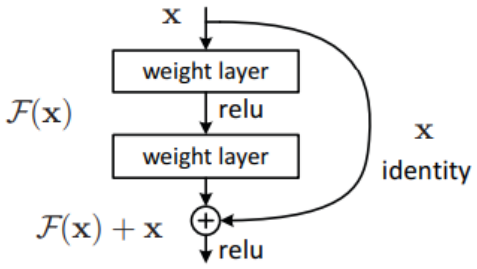


Figure 4.2.3: Residual building block (from [40])

The authors proposed the Residual Block to solve the degradation problem, allowing deeper networks to be trained, simultaneously solving another problem, the vanishing gradient problem. The vanishing gradient problem is caused by the fact that the gradient of the loss function with respect to the input of a layer is the product of the gradients of all the layers that follow it. As the number of layers increases, the gradient of the loss function with respect to the input of the first layer becomes exponentially smaller, causing the network to

train very slowly or not train at all. The Residual Block solves this problem by introducing skip connections, which allow the gradient to be directly backpropagated to earlier layers. To be more precise, skip connections reffer to the connections that skip one or more layers, connecting the input of a layer to the output of a later layer, using an identity mapping, convolutional layers that do not perform any transformations.

So, a Residual Neural Network (ResNet) is a convolutional neural network (CNN) that uses skip connections to allow the network to be deeper without suffering from the degradation problem and vanishing gradient problem. According to depth of the network, various of ResNet models can be created. Some of the most popular ResNet models are ResNet-18, ResNet-34, ResNet-50 and ResNet-101 with 18, 34, 50 and 101 layers respectively. For instance, the ResNet-18 architecture consists of 4 stages, with each stage containing 2 residual blocks. Each residual block consists of 2 convolutional layers, each followed by a batch normalization layer and a ReLU activation layer, as illustrated in Fig. 4.2.4.

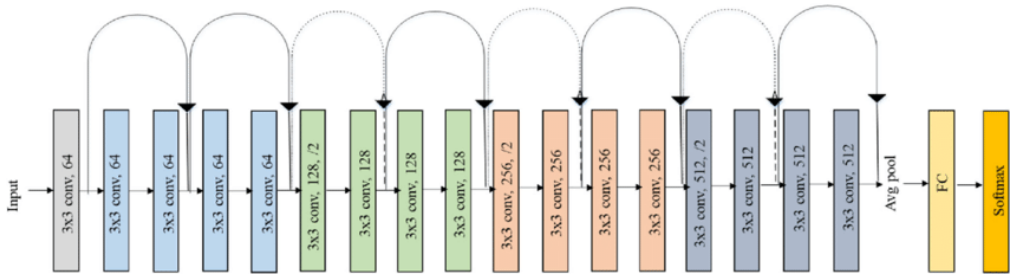


Figure 4.2.4: ResNet-18 architecture (from [41])

In the original paper, the authors proposed a neural network architecture for image classification, ResNet-18, and evaluated it on the ImageNet, a large-scale dataset with 1.2 million images and 1,000 classes. Later, Kensho Hara et al. proposed a 3D version of ResNet in [42], for video classification tasks. Specifically, the 2D convolutional layers, as well as any other 2D operations, were replaced by 3D convolutional layers, while the rest of the architecture remained the same.

This architecture quickly became popular in video classification tasks and a variety of methods and state-of-the-art models used it as a backbone. In action recognition specifically, ResNet variants are so popular due to their ability to extract spatio-temporal features from RGB frames, without the need of additional information, such as depth or skeleton data. In gesture recognition tasks, ResNet variants are also used, but they are often combined with other architectures, such as ConvLSTM, to achieve better performance. For instance, several ResNet variants were evaluated on the EgoGesture dataset, a dataset with 83,000 videos of 50 gesture classes, and achieved competitive results with state-of-the-art, at the time, on the dataset, proving the effectiveness of ResNet variants in gesture recognition tasks.

4.3 Transfer Learning

In machine learning, transfer learning is a technique where a model trained on one task is re-purposed on a second related task. This technique works due to the fact that features learned from one task are often useful for another task. So, training a model on a large dataset and then using the weights of the model to train another model on a smaller dataset, often leads to better performance, as well as faster convergence, than training the second model from scratch. Another term related to transfer learning is fine-tuning, which refers to the process of using a pretrained model and training only the some of the last layers of the model, instead of training the whole model from scratch.

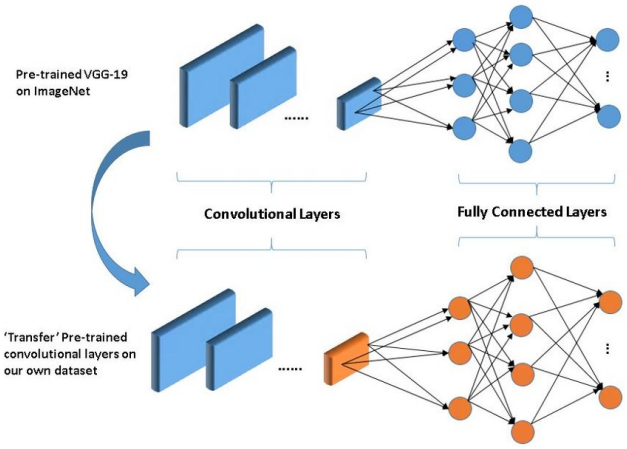


Figure 4.3.1: Transfer Learning illustration (from [43])

It should be noted that in transfer learning, the decision of which layers to fine-tune is crucial. As mentioned before, the first layers of a neural network extract low-level features while the last layers extract high-level features. So, if the task that the model was pretrained on is similar to the task that the model is used for, then it is better to fine-tune only the last layers of the model, as the low-level features are already useful for the new task. On the other hand, if the tasks are not similar, then it is better to fine-tune more layers, as the low-level features are not useful for the new task. For instance, if a model is pretrained on ImageNet, a dataset with images of objects, and is used for action recognition, then it is better to fine-tune only the last layers, as the low-level features of the pretrained model are useful for the new task. On the other hand, if a model is pretrained on Kinetics, a dataset with videos of actions, and is used for gesture recognition, then it is better to fine-tune more layers, as the low-level features of the pretrained model are not useful for the new task. All in all, the decision of which layers to fine-tune depends on the similarity of the tasks that the model was pretrained on and the task that the model is used for.

4.3.1 Transfer Learning on Action Recognition

Kensho Hara et al. in [44] presented the results of transfer learning from the Kinetics dataset, to the UCF-101 and HMDB-51 datasets, for 3-D ResNet variants. To be more precise, the authors trained several 3-D ResNet variants on the Kinetics dataset and then used the pretrained weights to train the same models on the UCF-101 and HMDB-51 datasets, two popular datasets for action recognition. In the expirements conducted, the authors fine-tuned only the last layer, the fully connected layer, of the pretrained models, since the experiments that allowed the fine-tuning of more layers show worse results. The results showed that the pretrained models outperformed the models trained from scratch, as presented in table 4.1, proving the effectiveness of transfer learning in action recognition from action datasets and that the pretrained models can be used as backbone models for testing new architectures.

Model	UCF-101	HMDB-51
ResNet-18 (from scratch)	42.4	17.1
ResNet-18	84.4	56.4
ResNet-34	87.7	59.1
ResNet-50	89.3	61.0
ResNet-100	88.9	61.7
ResNet-152	89.6	62.4
ResNet-200	89.6	63.5
DenseNet-121	87.6	59.6
ResNetXt-101	90.7	63.8

Table 4.1: Accuracy of pretrained models on UCF-101 and HMDB-51 datasets (from [44])

4.3.2 Transfer Learning on Gesture Recognition

When it comes to gesture recognition, transfer learning can also be used to improve the performance of models. In [45] the authors used a pretrained model on the ImageNet dataset, a benchmark dataset for image recognition consisting of 1.2 million images with 1000 classes, to train the model on a small set of gestures, consisting of images categorized into 5 classes. The pretrained model was the AlexNet model [77], a popular deep convolutional neural network consisting of 5 convolutional layers and 3 fully connected layers. The authors fine-tuned the last layer of the AlexNet model on the small set of gestures and achieved remarkable results, showing that transfer learning can be used to improve the performance of convolutional neural networks on gesture recognition tasks.

Noha Sarhan and Simone Frintrop in [46], showed that transferring spatiotemporal features from action recognition datasets is highly valuable to gesture recognition tasks, such as sign language recognition (SLR). To be more precise, the authors used the InceptionV1 model [47], a popular deep convolutional neural network, to extract spatiotemporal features from the Kinetics dataset. The authors then used the extracted features to train a model on the ChaLearn IsoGD dataset, a dataset for gesture recognition consisting of videos of people performing gestures. The results showed that the model trained on the extracted features outperformed the at the time state-of-the-art models, proving the effectiveness of transferring spatiotemporal features from action in gesture recognition tasks.

4.4 Benchmark Datasets

4.4.1 Action Recognition

Kinetics

The Kinetics dataset [47] is a large-scale, high-quality dataset designed for human action recognition in videos, as it covers a diverse range of human actions. Some of the action classes that are included in the dataset are:

- **Person Actions**, for example drawing, drinking, laughing
- **Person-Person Actions**, for example hugging, kissing, shaking hands
- **Person-Object Actions**, for example opening present, mowing lawn, washing dishes

The clips are sourced from YouTube videos. Consequently, for the most part, they are not professionally filmed or edited, and thus exhibit considerable variation in camera motion, illumination, cluttered background and appearance of different actors and objects, as shown in Fig. 4.4.1.

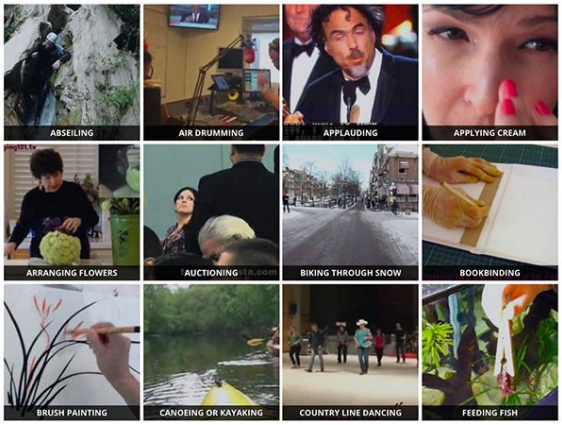


Figure 4.4.1: Sample frames for some action classes of Kinetics-700 (from kinetics-700 homepage)

The dataset has three versions:

- **Kinetics-400**
  which consists of around 240,000 video clips covering 400 human action classes with at least 400 video clips for each action class.
- **Kinetics-600**
  which consists of around 360,000 video clips covering 600 human action classes with at least 600 video clips for each action class.
- **Kinetics-700**
  which consists of around 650,000 video clips covering 700 human action classes with at least 700 video clips for each action class.

UCF-101

The UCF-101 dataset [36] is used for action recognition consisting of videos with realistic actions, as the data are collected from YouTube. This dataset has 101 action categories and contains 13,320 videos and is an extension of previously released datasets UCF-50 and UCF-101 dataset, which have 50 with and 101 action categories respectively. The 101 categories of the UCF-101 dataset can be divide into 5 action types:

- **Body motion**, including actions like pull ups, push ups, swing
- **Human-human interactions**, including actions like shaking hands, hugging, kissing
- **Human-object interactions**, including actions like apply eye makeup, drying hair, pizza tossing

- **Playing musical instruments**, including actions like playing guitar, playing piano, playing violin
- **Sports**, including actions like baseball pitch, basketball shooting, bench press

As mentioned before the dataset is composed of web videos, which are recorded in unconstrained environments, leading to videos exhibit camera motion, cluttered background, various lighting conditions, partial occlusion, as well as low quality frames, as shown in Fig. 4.4.2.

Figure 4.4.2: Sample frames for some action classes of UCF-101 (from [36])

In the original paper, the authors suggested three different train-test splits to evaluate the performance of action recognition models. Each split has its own training and test sets, and are provided in the official website of the dataset.

NTU-RGB+D

The NTU-RGB+D dataset [48] is a large-scale dataset for RGB - D human action recognition. The original dataset was introduced in the paper [78] and it was composed of 56,880 samples of 60 action classes collected from 40 subjects. Later, the dataset was extended to the NTU-RGB+D 120 dataset, which contains 114,480 samples of 120 action classes performed by 106 subjects. The actions can be generally divided into three categories:

- **Daily actions**, which contain 82 classes, including actions such as eating, writing, sitting down
- **Health-related actions**, which contain 12 classes, including actions such as blowing nose, staggering, falling down
- **Mutual actions**, which contain 26 classes, including actions such as handshaking, pushing, hugging

Some examples of the action classes can be seen in Fig. 4.4.3.

Figure 4.4.3: Sample frames for some action classes of NTU-RGB+D (from [79])

The actions were captured using three cameras with different horizontal imaging viewpoints, namely -45°, 0°, and +45°, while the vertical heights of the cameras, the distances of the cameras to the subjects, the locations and the background all vary, leading to 32 different collection setups. Also, the dataset provides multi-modality information for action characterization, including depth maps, 3D skeleton joint position, RGB frames, and infrared sequences.

Moreover, the authors proposed a benchmark for the evaluation of action recognition models, which includes a cross-subject and a cross-view evaluation.

- **Cross-subject evaluation**, where the 106 subjects are split into training and testing groups, according to the authors with each group consisting of 53 subjects
- **Cross-view evaluation**, where the samples with even collection setup IDs are used for training and those with odd setup IDs are used for testing

HMDB-51

The HMDB-51 dataset [49] is a large collection of realistic videos from various sources, including movies and web videos. The dataset is composed of 6,766 video clips divided into 51 action categories. These categories can be grouped in five action types:

- **General facial actions**, including actions like smile, laugh, talk
- **Facial actions with object manipulation**, including actions like smoke, eat, drink
- **General body movements**, including actions like cartwheel, clap hands, climb
- **Body movements with object interaction**, including actions like brush hair, catch, draw sword
- **Body movements for human interaction**, including actions like fencing, hug, kiss

The HMDB-51 dataset contains actions, from daily acitvities as shown in Fig. 4.4.4 which vary in motion rather than static poses and can thus be seen as a valid contribution for the evaluation of action recognition systems as well as for the study of relative contributions of motion and static appearance information.

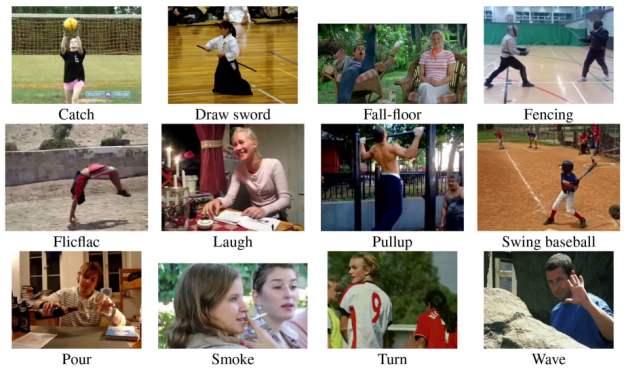


Figure 4.4.4: Sample frames for some action classes of HMDB-51 (from [80])

4.4.2 Gesture Recognition

Chalearn IsoGD

The Chalearn IsoGD dataset [50] is a large-scale dataset for RGB-D gesture recognition. The dataset was derived from the ChaLearn Gesture Dataset [81] by isolating the gestures in order to have one gesture per video. The dataset contains 47,933 RGB-D gesture videos, with 249 gesture labels performed by 21 different individuals. This variety of gestures is drawn from different domains, such as sign language for the deaf, underwater sign language, helicopter and traffic signal, pantomimes and symbolic gestures, Italian gestures, and body language, some examples of which can be seen in Fig. 4.4.5. This dataset provides a challenging benchmark for gesture recognition due to the large number of gesture classes, as well as the presence of rgb and depth information.

Figure 4.4.5: Sample frames for some gesture classes of Chalearn IsoGD (from [82])

NVGesture

The NVGesture dataset [51] is a large multi-modal dynamic hand gesture dataset captured with color, depth and stereo-IR sensors. This dataset is composed of 1,532 dynamic gestures performed by 20 subjects, categorized into 25 classes, intended for human-computer interfaces and recorded by multiple sensors.

For the collection of data, the subjects performed several types of gestures indoors in a car simulator with both bright and dim artificial lighting. The subjects performed gestures with their right hand, while observing the simulator's display and controlling the steering wheel with their left hand, as can be seen in Fig. 4.4.6.

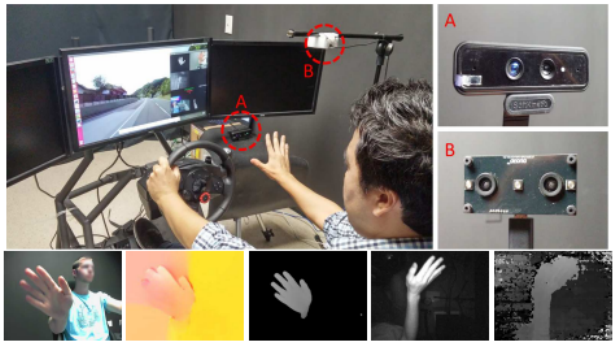


Figure 4.4.6: Sample frames for some gesture classes of NVGesture (from [51])

EgoGesture

The EgoGesture dataset [37] is a large-scale dataset for the task of egocentric gesture recognition. Specifically, the dataset is designed to address the challenges of recognizing gestures in a first-person perspective, in contrast to the more commonly studied third-person perspective. This dataset contains 24,161 RGB-D video samples frames from 50 distinct subjects, categorized in 83 classes of static or dynamic gestures specifically for interaction with wearable devices.

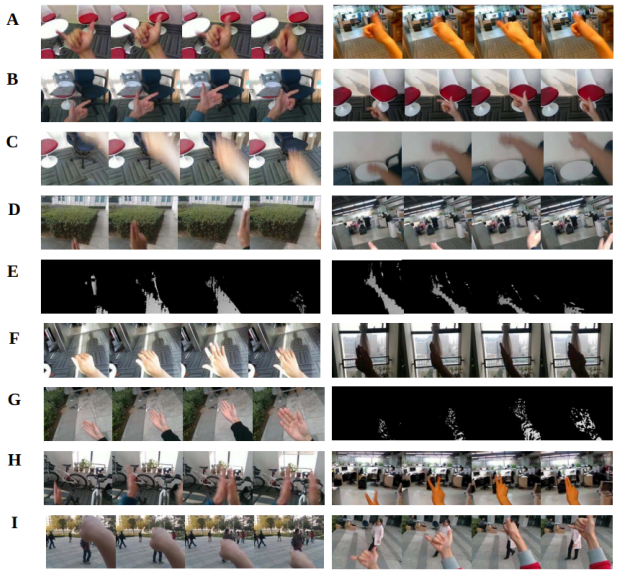


Figure 4.4.7: Sample frames for some gesture classes of EgoGesture (from [37])

Jester

The Jester dataset [52] is a large-scale, real-world dataset for dynamic gesture recognition. The data have been collected with the help of 1,376 different subjects performing gestures in front of a laptop camera or a webcam in their unconstrained environments, thus providing a significant variation in the background and appearance among actors, as shown in Fig. 4.4.8.

Figure 4.4.8: Sample frames for some gesture classes of Jester (from [83])

This dataset consists of 148,092 video clips in total categorized into 27 classes, which include 25 gesture classes and two classes, that should not be recognized as any particular movement, one "No Gesture" and one "Doing Other Things" class. From the 25 gesture classes, 5 classes are considered as static gestures, for instance "Thumb Up", "Thumb Down", while the remaining classes require distinguishing between fine-grained visual details, for example "Zooming Out With Two Hands", "Zooming In With Full Hand". The "No Gesture" category presents a video of a person sitting or standing still, while the "Doing Other Things" category is a collection of various activities, that a user of a human-computer interface might perform without intending to communicate with the system. This approach makes the system more robust to false positives, as it is less likely to confuse a gesture with a non-gesture class.

Chapter 5

Multitask Learning on Action and Gesture Recognition

5.1 Overview

Multi-task learning (MTL) is a subfield of machine learning introduced by Caruana in [53], in which multiple tasks are solved simultaneously, while exploiting commonalities and differences across tasks. Multi-task learning reflects the learning process of human beings more accurately than single task learning (STL). Any time that a human attempts to learn something new, tremendous amount of prior knowledge is brought to the table. Similar to humans, neural networks require such numerous training examples and computation time, when none prior knowledge is used. So, MTL architectures increase data efficiency and can potentially yield faster learning speed, helping to reduce the need for large-scale data requirements and large computational resources.

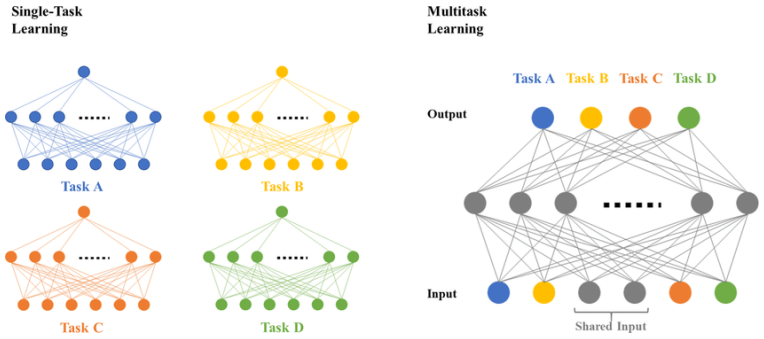


Figure 5.1.1: Comparison of single task learning (left) with multitask learning (right) (from [84])

5.1.1 Task Definition

Before diving deeper into the details of MTL, a definition of what a task is essential. A task is a specific learning problem, with its own objective function and its own set of data, that a model needs to solve. A task can be defined as a category of machine learning problem, such as classification or regression. For example, in computer vision tasks could be image/video classification, object detection or semantic segmentation. Moreover, a task T^i is usually accompanied by a training dataset D^i consisting of n^i training samples, i.e., $D^i = \{x_j^i, y_j^i\}$, for $j = 1, ..., n^i$, where $x_j^i \in R$ is the jth training instance in D^i and y_j^i is its label. We denote by X^i the training data matrix for T^i, i.e., $X^i = (x_1^i, ..., x_{n_i}^i)$, as defined by Zhang and Yang in [54]. So, tasks can be also distinguished by their datasets, as well as the type of data they use. For example, different dataset can denote different tasks and different tasks can use different modalities, such as rgb, depth, skeleton.

5.1.2 Multi-task Learning Types

As mentioned before, tasks can be distinguished by their datasets, types of data they use and their objective functions. So, multi-task learning methods can be split into two types, according to the type of tasks they solve, in homogeneous and heterogeneous MTL.

- **In Homogeneous MTL**
  or same-domain MTL, all tasks are of the same type. For example in computer vision, all tasks could be image/video classification tasks.
- **In Heterogeneous MTL**
  or cross-domain MTL, different tasks are of the different type. For example in computer vision, one task could be an image/video classification task, while another task could be an object detection task.

Also, multi-task learning methods can be divided into two types, according to the type of data they use, in homogeneous-feature and heterogeneous-feature MTL.

- **In Homogeneous - feature MTL**
  or same-modality MTL, the features, used in each task, are of the same type, meaning that for samples of d_i from D_i and d_j from D_j, d_i equals d_j for any $i \neq j$. For example in computer vision, both tasks could use rgb features.

- **In Heterogeneous - feature MTL**
  or cross-modality MTL, the features, used in each task, are of the different type, meaning that for samples of d_i from D_i and d_j from D_j, d_i does not equal d_j for any $i \neq j$. For example in computer vision, one task could use rgb features, while another task could use skeleton data.

Moreover, multi-task learning methods can be further categorized into two types, according to the datasets they use, in same-dataset and cross-dataset MTL.

- **In Same-Dataset MTL**
  all tasks use the samples of the same dataset.
- **Cross-Dataset MTL**
  different tasks could use samples from different datasets.

5.1.3 What tasks to share?

Though multi-task learning may be beneficial in many cases, learning concepts for multiple tasks does bring difficulties, which single task learning is not dealing with. In particular, some tasks may have conflicting needs. In this case, increasing the performance of a model on one task will hurt performance on a task with different needs, leading consequently to inferior overall performance. This phenomenon is called *negative transfer* or *destructive interference*. Negative transfer can occur when the tasks are not related to each other, or when the tasks are related to each other, but the weight sharing method used in the MTL model is not suitable for the tasks. So, two questions arise.

- **Which tasks should be learned together?**
  The tasks should be related to each other, so that the model can exploit the commonalities and differences across tasks.
- **How should the tasks be learned together?**
  The weight sharing method should be suitable for the tasks, so that the model can exploit the commonalities and differences across tasks.

But the answer to these questions is not binary. The answer depends on the tasks, the datasets, the weight sharing method used in the MTL model, as well as the overall architecture and methods used to optimize the parameters of the model. So, whether two tasks should be learned together or not, is a question that needs to be answered on a case by case basis. Even if destructive interference occurs, it does not mean that the MTL model is not useful. One approach to alleviate negative transfer is to weight each of the losses to prevent tasks from overpowering others due to loss scale differences [55]. Also, several research on task relatedness has been conducted, such as Standley et al. in [56], where they proposed a framework to determine the most beneficial task grouping for a given set of tasks.

5.2 Weight Sharing Methods

When designing a multi-task learning architecture, there are many different factors to consider, such as the portion of the model's parameters that will be shared between tasks, and how to parameterize and combine task-specific and shared modules. Many of the proposed architectures for MTL try to balance the degree of information shared between tasks, as too much sharing will lead to negative transfer and can cause worse performance of joint multi-task models than individual models for each task, while too little sharing does not allow the model to effectively leverage information between tasks. Though, there are many different ways to share information between tasks, the most common ones are hard parameter sharing, soft parameter sharing and task routing.

5.2.1 Hard Parameter Sharing

Hard parameter sharing, introduced by Caruana in [53], is the most commonly used approach to MTL in neural networks, due its simplicity and effectiveness. It is generally applied by sharing the hidden layers between all tasks, while keeping several task-specific output layers, as can be seen in Fig. 5.2.1. The shared layers are trained jointly on all tasks, while the task-specific layers are trained independently on each task. The shared layers are usually the first layers of the network, while the task-specific layers are the last layers of the network. The intuition behind this approach is, that the shared layers will learn features, that are useful for all tasks, as early layers in a neural network learn more general features, while the task-specific layers will learn features that are only useful for their specific task.

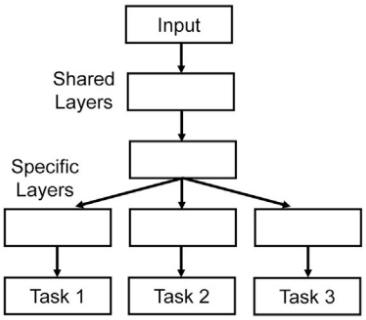


Figure 5.2.1: Hard Parameter Sharing scheme (from [57])

Hard parameter sharing greatly reduces the risk of overfitting, as proven by Baxter in [85]. In particular, the risk of overfitting the shared parameters is an order N - where N is the number of tasks - smaller than overfitting the task-specific parameters, i.e. the output layers. This observation can be explained as that the more tasks are learned simultaneously, the more the shared layers are forced to learn a representation that captures all of the tasks, and the less is the chance of overfitting on the original task. Obviously, learning more and more tasks underlies the risk of negative transfer, as the shared layers may learn a representation that is not useful for any of the tasks. This is why it is important to choose the right amount of tasks to be learned simultaneously, as well as the right amount of shared layers.

5.2.2 Soft Parameter Sharing

Soft parameter sharing, in contrast to hard parameter sharing, is a more flexible approach to MTL, that allows for more information sharing between tasks, as well as more task-specific layers. To be more specific, in soft parameter sharing, each task has its own model with its own parameters. The distance between the parameters of the model is then regularized to be similar to each other, as shown in Fig. 5.2.2. This is achieved by adding a penalty term to the loss function, that penalizes the difference between the parameters of the shared layers. Various regularization terms have been tested. For example, Yang and Hospedales used the trace norm in [86], which is the sum of the singular values of a matrix, while usually the regularization term is the L_1 norm or the L_2 norm of the difference between the parameters of the shared layers. The regularization term is oftern weighted by a hyperparameter λ, that controls the amount of information sharing between

tasks. The higher the value of λ, the more the shared layers are forced to learn a representation that captures all of the tasks, and the less is the chance of overfitting on the original task.

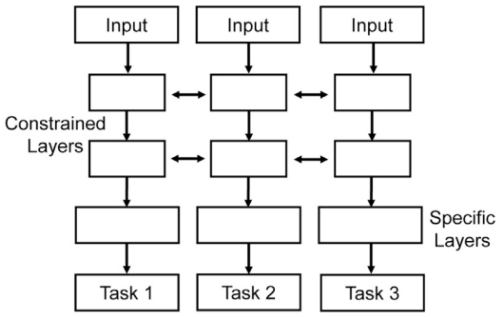


Figure 5.2.2: Soft Parameter Sharing scheme (from [57])

One of the most popular soft parameter sharing methods are the Cross-Stitch Networks, introduced by Misra et al. in [58]. The authors proposed the Cross-Stitch Networks as solution to experimenting with different share representations according to the hard parameter sharing method. As mentioned in the previous, different amount of sharing tends to work best for different tasks, so to evaluate multi-task learning between to tasks, every possible combination of shared layers should be tested. This search of optimal weight sharing of a model's layers demonstrated using the hard parameter sharing method in Fig. 5.2.3, still lacks control of the amount of information shared in each layer.

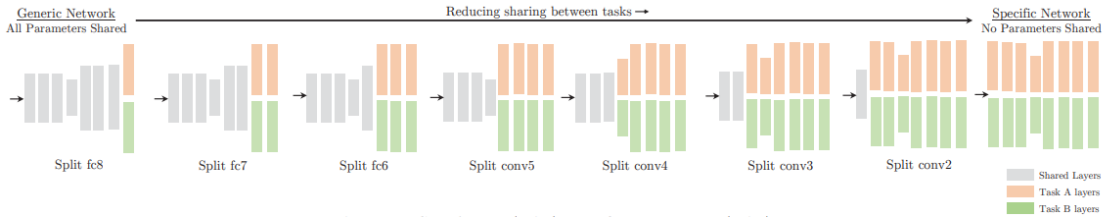


Figure 5.2.3: Different shared layers according to the hard parameter sharing method (from [58])

In particular, Cross Stitch Networks consist of a stack of N networks, one for each task, that are connected by cross-stitch units, as shown in Fig. 5.2.4. The cross-stitch units are trained jointly with the rest of the network, and the optimal combination of the outputs of the two networks is learned automatically. The layers that are connected by cross-stitch units are not necessarily the same for all tasks. For example, in the original paper, the authors experimented on AlexNet [87] applying cross-stitch units after every convolution activation map and after every pooling activation map, and found the latter performed better.

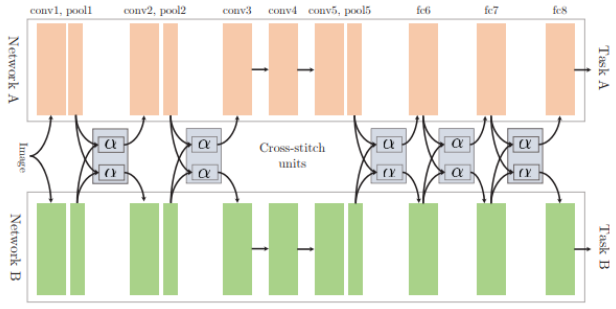


Figure 5.2.4: Cross stitch units applied on AlexNet (a popular CNN model) (from [58])

Cross-Stitch units try to find the best shared representations for multi-task learning. Each cross-stitch unit is a differentiable module, aiming to learn the optimal linear combination for a given set of tasks, as depticted

in Fig. 5.2.5.

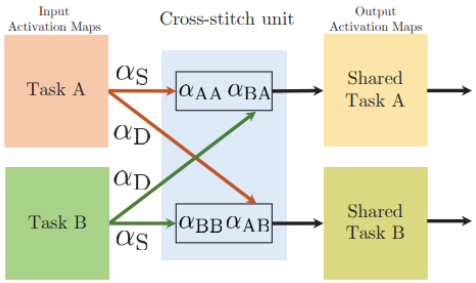


Figure 5.2.5: Cross Stitch Unit (from [58])

For instance, for two tasks, cross-stitch units are applied between layers of the task-specific models. At each layer of the network, these units learn a linear combination of the activation maps of the two tasks. For two activation maps x_A, x_B from layer l the cross-stich unit learns linear combinations $\tilde{x}_A$, $\tilde{x}_B$, parameterized by α. In particular, at location (i, j) in the activation map the following equations hold:

$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix} \tag{5.2.1}$$

where α_{AA} and α_{BB} can be renamed as α_S, the same-task values, since they weigh the activations of the same task, while α_{AB} and α_{BA} can be renamed as α_D, the different-task values, since they weigh the activations of another task. By varying α_S and α_D values, the unit can decide between shared and task-specific representations, or choose a middle ground if needed.

5.2.3 Task Routing

A more flexible approach to MTL proposed by Strezoski et al. in [59] is Task Routing Networks, which allow for fine-grained parameter sharing between tasks, that occurs at the feature level instead of the layer level. The novel component of this architecture is the Task Routing Layer, which applies a task-specific binary mask to the output of a convolutional layer, to which it is applied, zeroing out a subset of the computed features and effectively assigning a subnetwork to each task which overlaps with that of other tasks, as illustrated in Fig. 5.2.6.

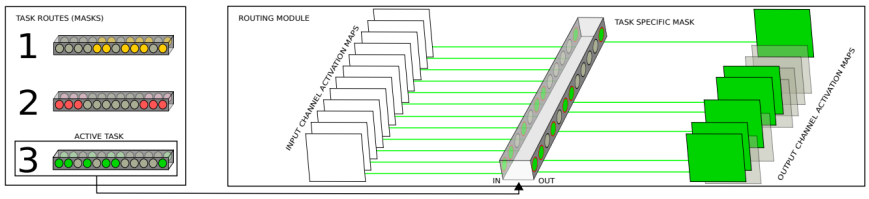


Figure 5.2.6: Activation of task specific convolutional layers according to task routing layer (from [59])

The binary masks are not learned, instead they are randomly initialized at the beginning of training and fixed from that point on. Although this random initialization does not allow for the possibility of a principled parameter sharing scheme between tasks, the user still has control over the degree of sharing between tasks through the use of a hyperparameter σ, known as the sharing ratio. The parameter σ takes values between 0 and 1, specifying the proportion of units in each layer, which are task-specific, and the random initialization of the binary masks in each layer are executed in a way to fit this constraint.

A Task Routing Layer (TRL) contains a task specific binary mask applied over the input of a convolutional layer, performing a conditional feature-wise transformation, that generates a masked output, which is then propagated to the next convolutional block. The feature-wise transform applied to the convolutional output

can affect the local running mean and variance, so the TRL is placed right after the batch normalization layer if present, as shown in Fig. 5.2.7.

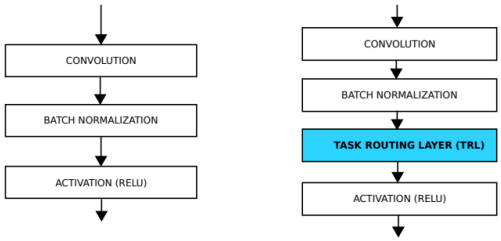


Figure 5.2.7: Task Routing Layer placement within a convolutional block (from [59])

5.2.4 Advanced Weight Sharing Methods

Besides the aforementioned weight sharing methods, there are more advanced ones, that try to combine the advantages of hard and soft parameter sharing. One of them is the Learned Weight Sharing (LWS) method, introduced by Prellberg et al. in [60], aiming to define an algorithm to learn the assignment between a shared set of weights and task-specific layers.

In Fig. 5.2.8 some different weight sharing techniques to solve a three-task MTL problem are shown. Each box depicts a network layer and the letters denote different weights. Colored partitions in each box illustrate which tasks share weights for that layer. So in a brief overview, this approach aims to liberate the weight sharing from restrictive approaches like hard parameter sharing, where all tasks share the same weights for a layer, or soft parameter sharing, where all tasks share the same weights for a layer, but with a penalty term to encourage similarity. Instead, LWS learns a task-specific assignment of weights to tasks, which is a more flexible approach, converting MTL into a task-assignment problem from previous data-dependent approaches.

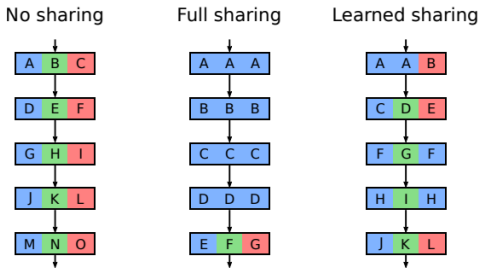


Figure 5.2.8: Different weight sharing methods between task specific neural networks (from [60])

In particular, to create a LWS multi-task model any neural network architecture is chosen as the base architecture and is duplicated once for each task to create task-specific networks. The last layer of each task-specific network is modified to have the appropriate number of outputs for the task, similar to the hard parameter sharing method. For the shared layers, a set of K weights is created and all of the N task-specific network layers are assigned a weight from its corresponding set. By assigning the same weight to multiple task-specific networks, weight sharing is achieved. The problem is now to find good assignments between the weights and task-specific network layers and at the same time train the weights themselves. To achieve this, the authors propose to use two optimization algorithms, one for the assignment problem and one for the weight optimization. Specifically, the assignment problem is solved with the Natural Evolution Strategy (NES) algorithm and the weight optimization is solved with Stochastic Gradient Descent (SGD) algorithm.

Natural Evolution Strategy refers to a class of black-box optimization algorithms, that update a search distribution in the direction of higher expected fitness using the natural gradient of the expected fitness. The natural gradient is an alternative to the standard gradient, that takes into account the curvature of the

parameter space, i.e., the second derivatives of the loss function with respect to the parameters, to adjust the learning rate for each parameter.

The search for good assignments and layer weights is cast as an optimization problem, expressed as follows:

$$\min_{\theta,\alpha} f(\theta, \alpha), \tag{5.2.2}$$

where $f : \Theta \times A \rightarrow R$ is the loss over all tasks, $\theta \in \Theta$ is a vector of all layer weights, and $\alpha \in A$ is an assignment of weights to task-specific network layers. The loss function f is differentiable wrt. θ, but black-box wrt. α. So, the authors propose a stochastic version of the problem, by introducing a probability distribution π over the set of all possible assignments of weights to task-specific network layers A with a probability density function $p(\alpha|\pi)$ - a statistical expression that defines a probability distribution for a continuous random variable, meaning that the probability of the random variable falling within any particular range of values can be computed by integrating the probability density function over that range. The optimization problem is then described as follows:

$$\min_{\theta,\pi} J(\theta, \pi) = E_{\pi}[f(\theta, \alpha)]. \tag{5.2.3}$$

This stochastic formulation makes the assignments able for optimization through π using the NES algorithm, but requires to sample assignments for the calculation of the gradient wrt. θ. The idea here is to turn the discrete, non-differentiable optimization problem over the assignments α into a continuous, differentiable optimization problem over the parameter π. This is done by introducing a probability distribution over the assignments and optimizing the parameters of this distribution. In practice, this means that instead of directly choosing the assignments, the algorithm chooses the parameters π, and the assignments are then sampled from the probability distribution defined by these parameters. The optimization problem is then solved by alternating between the following optimization steps.

Assignment Optimization

In order to optimize π while keeping θ fixed, the assignments of weights $a_1, ..., a_{\lambda_\pi}$ distributed according to $p(\alpha|\pi)$ are sampled and their loss values $l_i = f(\theta, \alpha_i)$ are calculated on the same batch of training data for all assignments. The authors then propose to use a Monte-Carlo estimate, with population size λ, of the gradient of the loss function wrt. π as follows:

$$\nabla_{\pi} J(\theta, \pi) \approx \frac{1}{\lambda_\pi} \sum_{i=1}^{\lambda_\pi} u_i \nabla_{\pi} \log p(\alpha_i|\pi), \tag{5.2.4}$$

where u_i denote the utility values, which are created by fitness shaping to make the algorithm invariant to the scale of the loss function. Loss values l_i are transformed into utility values using the following expression:

$$u_i = 2 \cdot \frac{l_i - 1}{\lambda_\pi - 1} - 1. \tag{5.2.5}$$

The gradient is then used to update the parameters of the probability distribution π with learning rate η_π, according to the following step in the direction of $\nabla_{\pi} J(\theta, \pi)$:

$$\pi + \eta_\pi \nabla_{\pi} J(\theta, \pi). \tag{5.2.6}$$

Weight Optimization

In order to optimize θ while keeping π fixed, the authors used a Monte-Carlo approximation. In particular, the assignments of weights to task-specific network layers $a_1, ..., a_{\lambda_\theta}$ distributed according to the pdf $p(\alpha|\pi)$ are sampled and backpropagation is performed for each sample. The same batch of training data is used for

the backpropagation step throughout this process for every assignment. The resulting gradients $\nabla_\theta f(\theta, \alpha_i)$ are averaged over all assignments, so that the final gradient is described by the following equation:

$$\nabla_\theta J(\theta, \pi) \approx \frac{1}{\lambda_\theta} \sum_{i=1}^{\lambda_\theta} \nabla_\theta f(\theta, \alpha_i). \tag{5.2.7}$$

Using this gradient, θ is updated by SGD with learning rate η_θ, according to the following step:

$$\theta - \eta_\theta \nabla_\pi J(\theta, \pi). \tag{5.2.8}$$

So, the LWS algorithm by alternating between the assignment optimization step and the weight optimization step finds the optimal assignment of shared weights to task-specific network layers and the optimal weights for each layer. For instance, in Fig. 5.2.9, given a total of $N = 12$ layers and $K = 3$ weights per weight set, the most probable assignment of weights is depticted, which is used for inference.

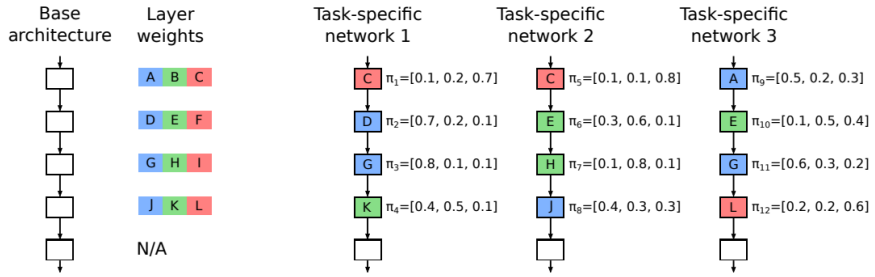


Figure 5.2.9: Example of how learned weight sharing shares information between task specific neural networks (from [60])

5.3 Multitask Loss Calculation Methods

Another important factor to consider when designing a multi-task learning architecture is how the multitask loss is calculated. The term multi-task loss reffers to the total loss combining the losses across all tasks, which is used to update the parameters of the model for all the tasks, as shown in Fig. 5.3.1.

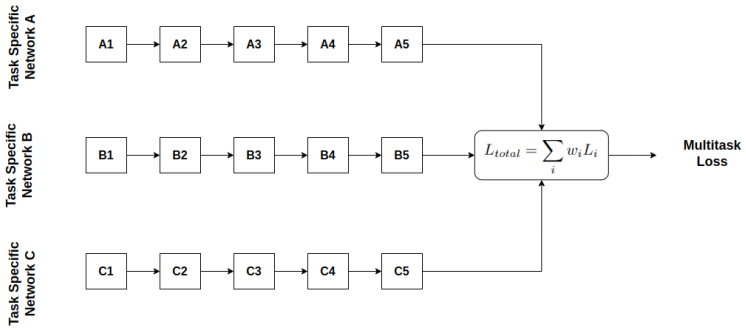


Figure 5.3.1: Multi-task Loss Calculation scheme

The most common methods for calculating the multi-task loss are the average loss calculation, the uncertainty in weighing losses and the dynamic weight average.

5.3.1 Average Loss Calculation

The multi-task loss can be calculated as the average of the losses of all tasks. This method is the simplest and most straightforward way to calculate the multi-task loss, as it treats all tasks equally. However, it does not take into account the difficulty of each task, which can lead to suboptimal results. The mathematical formulation of the average loss calculation is:

$$\mathcal{L}_{MTL} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i, \tag{5.3.1}$$

where N is the number of tasks and $\mathcal{L}_i$ is the loss of the i-th task.

Another way to calculate the multi-task loss is to use a weighted average, where each task is assigned a weight based on its importance. Although this method takes into account the difficulty of each task, the weights are decided in advance of the training of the model and do not change during training, making this approach quite arbitrary.

5.3.2 Uncertainty in Weighing Losses

In order to take into account the difficulty of each task, weights need to be assigned to each task. Following this method, the performance of the model depends heavily on the selection of weights, while searching for these optimal weightings is computationally expensive and time-consuming, especially for large models with numerous tasks.

In [61] A. Kendall et. al. propose a method to calculate the multi-task loss based on the uncertainty of weighing each task, meaning that the knowledge of which task is more important, in order to reach the optimal solution, is not known in advance and needs to be learned during training. To be more specific, the uncertainty in weighing losses method is defined as assigning a weight to each task, which is learned during training, based on the minimization of an objective function, that describes the model.

This method for calculating the multi-task loss is based on the idea that the uncertainty of each task can be used to determine the weight of the task. This approach falls under the category of task-dependent or homoscedastic uncertainty, which captures the uncertainty with respect to information which our data cannot explain but is independent on the input data. It is a quantity, which stays constant for all input data and varies between different tasks, that is why it can be described as task-dependent uncertainty.

The mathematical formulation of the uncertainty loss calculation for two tasks is:

$$\mathcal{L}_{MTL}(w, \alpha_1, \alpha_2) = \frac{1}{\alpha_1^2}\mathcal{L}_1(w) + \frac{1}{\alpha_2^2}\mathcal{L}_2(w) + \log(\alpha_1) + \log(\alpha_2) \ , \tag{5.3.2}$$

where α_1 and α_2 are the uncertainty weights for the two tasks, positive scalar, interpreted as the temperature in a Boltzmann distribution, where magnitude of these parameters determines how uniform (flat) the discrete distribution is and w is the model's parameters.

In practice, the network predicts the log variance, $\sigma_i := \log(\alpha_i^2)$, in order to be more numerically stable than regressing the variance, as the loss avoids any division by zero. Although, providing a way to choose better weights for the tasks, while training, the mathematical implementation is not optimal, as it leads to negative values for the variance, which is not acceptable in the context of uncertainty estimation. Solution to this problem is suggested in [62] from Liebel et. al., where the authors alter the uncertainty loss calculation to avoid negative values for the variance, by converting the regularization term $\log(\alpha_i^2)$ to $\log(1 + \alpha_i^2)$.

So, the new mathematical formulation of the uncertainty loss calculation for two tasks, referenced to as automatic weight loss, is:

$$\mathcal{L}_{MTL}(w, \alpha_1, \alpha_2) = \frac{1}{\alpha_1^2}\mathcal{L}_1(w) + \frac{1}{\alpha_2^2}\mathcal{L}_2(w) + \log(1 + \alpha_1^2) + \log(1 + \alpha_2^2) \ . \tag{5.3.3}$$

5.3.3 Dynamic Weight Average

Another method for calculating the multi-task loss is the dynamic weight average calculation [63], which assigns a weight to each task based on the rate of change of loss for each task during during previous iterations. It is easily implemented as it requires only the numerical values of the losses of each task at the current and previous iterations.

The mathematical formulation of the dynamic weight average loss calculation is:

$$\begin{aligned} \mathcal{L}_{MTL} &= \sum_{i=1}^{k} \lambda_k(t)\mathcal{L}_k, \\ \lambda_k(t) &= \frac{K \exp(w_k(t-1)/T)}{\sum_i \exp(w_i(t-1)/T)}, \ \text{with } w_k(t-1) = \frac{\mathcal{L}_k(t-1)}{\mathcal{L}_k(t-2)} \ , \end{aligned} \tag{5.3.4}$$

where L_k is the loss of the k-th task, T is the temperature term which controls the softness of task weighting, with large values resulting in a more even distribution between different tasks, $w_k(t-1)$ is the weight of the k-th task at time $t-1$ and $\lambda_k(t)$ is the weight of the k-th task at time t.

In the implementation, the loss value $L_k(t)$ is calculated as the average loss over several iterations, so it reduces the uncertainty from stochastic gradient descent and random training data selection. For $t = 1, 2$ the weights $w_k(t)$ are initialized to 1, but any non-balanced initialisation based on prior knowledge could also be introduced.

In practice, the dynamic weight average method will assign higher weights to tasks with lower loss rates, which means that the model will focus more on tasks that are improving more slowly, in order to achieve a better overall performance.

5.4 Other Approaches

Multi-task learning methods have been used in the past for action recognition and gesture recognition tasks, to improve the performance of the algorithms. In this section, some of the most important multi-task learning approaches, that have been used in action and gesture recognition, are presented.

5.4.1 Previous Work on Multi-task Learning in Action Recognition

On way to leverage from multi-task learning is to train a model on multiple datasets, that have the same modality. Karen Simonyan and Andrew Zisserman in their work [38] proposed the popular Two-Stream Convolutional Neural Network (TSN) architecture, which achieved remarkable results in video classification. This model needed a great amount of training data to achieve good performance, but in action recognition, where it was tested, the benchmark datasets of the time were not large enough. To address this issue, they proposed a multi-task learning approach so that the model could be trained on data from different datasets.

In particular, the network was trained on two datasets, the UCF-101 and the HMDB-51 datasets, which consist of videos of mostly different human actions, but they have the same modality, which is RGB data. So, a simple combination of the two datasets was not feasible, due to the fact that the datasets had action categories that overlapped. Instead, they proposed a multi-task learning approach, where the model shares the parameters of the first layers of the network, but has separate branches for the last layers, corresponding to the different datasets. Each of the layers is equipped with its own loss function, which operates only on the videos, coming from the respective dataset. The overall training loss is computed as the sum of of the losses of the individual tasks and the network weight derivatives can be found by backpropagation.

The results showed that the network was able to generalize better on both datasets, compared to training on each dataset separately, thus showing the benefits of multi-task learning in cross-dataset action recognition with the same modality.

Another way to benefit from multi-task learning is to train a model to perform different tasks or using different modalities. In the work of [64], the authors proposed a multi-task framework for jointly estimating 2D or 3D human poses from monocular color images and classifying human actions from video sequences. The goal of this multi-task method is to jointly handle pose estimation and action recognition, prioritizing the use of predicted poses on action recognition, while benefiting from shared computations between the two tasks. An overview of the multi-task learning framework is shown in Fig. 5.4.1.

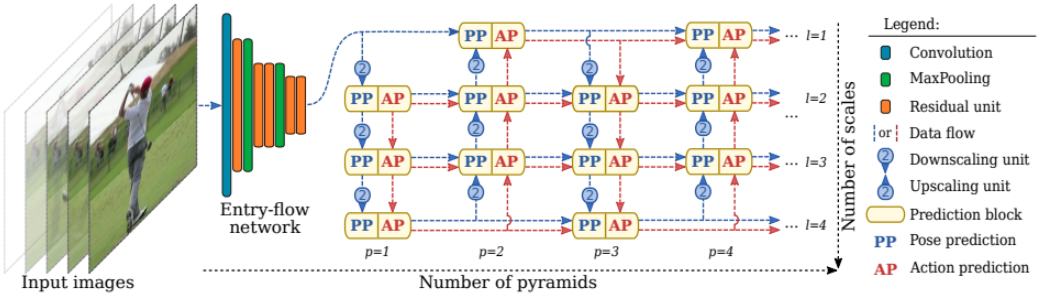


Figure 5.4.1: Overview of the multi-task network (from [64])

The multi-task network can operate into two distinct modes, so it can handle both poses and actions. In the first mode, the single frame processing mode, the network processes a single frame to estimate poses so only the layers related to pose estimation, the layers corresponding to the blue arrows in the figure above, are active. In the second mode, the video clip processing mode, the network processes a video clip to estimate poses and classify actions, so all the layers are active. In this mode, the information flow from single frame processing layers, corresponding to pose estimation, and from video clip processing layers, corresponding to action recognition, are independently propagated from one prediction block to another.

The prediction block perfmors computations to handle multimodal data (single frames and video clips) in a unified way, in order to predict poses and actions simultaneously and re-injected these predictions into the

network for further refinement. To be more precise, the prediction block consists of three tensors : a tensor of single frame features, which is propagated from one prediction block to another, a tensor of video clip features, exclusively used for action predictions, also propagated from one prediction block to another and a tensor of multi-task (single frame) features used for both pose and action. The tensor of multi-task features performs upscaling, downsampling computations as well as convolutions, to handle the multimodal data in a way that the network can integrate high level pose information with low level visual features.

5.4.2 Previous Work on Multi-task Learning in Gesture Recognition

In gesture recognition, multi-task learning has also been used to combine different tasks. To be more precise, in the work of [65], a multi-task learning framework to handle gesture segmentation and gesture recognition was proposed. A convolutional neural network is used learn segmentation information with depth modality supervision during the training process, while requiring only RGB modality during inference. The depth modality contains prior information for the location of the gesture, thus it can be used as the supervision for gesture segmentation. The proposed architecture achieves better results than the state-of-the-art methods of the time, proving the benefits of multi-task learning in gesture recognition.

5.4.3 Previous Work on Multi-task Learning with Transfer Learning

As mentioned in the previous sections, in multi-task learning tasks are learned in parallel during training, requiring the parameters of the model to be trainable during the whole training process, while in transfer learning a model is trained on a source task and then fine-tuned on a target task, requiring some of the parameters of the model to be "frozen", not trainable, during the training of the target task. Although, these two methods are different, they can be combined to improve the performance of the model.

In [66] the authors proposed a multi-task learning framework, that combines multi-task learning with transfer learning, to improve the performance of the model. To be more specific, the model is trained in two phases. In the first training phase the model is trained on multiple tasks simultaneously, according to the multi-task learning theory, and then this model is used as a pre-trained model for single-task learning on the tasks, that were used in the multi-task learning part, thus following a transfer learning scheme. The results showed that the proposed method outperformed the single-task learning, as well as the multi-task learning, performances for the tasks.

5.5 Proposed Method

This diploma thesis experiments on the effectiveness of multitask learning in combining the tasks of action and gesture recognition. Since, the answer to the question of whether multitask learning can be used to jointly train two tasks is not binary, different multitask learning architectures are implemented and tested, to evaluate their performance. Specifically, the three different MTL models, that were implemented, follow different weight sharing methods, hard parameter sharing, soft parameter sharing and a more complex method aiming to leverage the advantages of both hard and soft parameter sharing, respectively. The proposed multitask learning approach falls into the category of cross-dataset, since the two tasks are trained on different datasets. Also, it can be considered as a homogeneous multitask learning approach, since the two tasks are of the same type, classification and specifically they are recognition tasks. Lastly, the proposed approach can be further categorized as a homogeneous-feature multitask learning approach, since the two tasks use RGB data as input.

Moreover, the proposed idea is extented to evaluate the three different MTL architectures on combining the tasks of gesture recognition using different modalities, RGB and Depth data. This alternative approach falls into the category of cross-modality multitask learning, since the two tasks are trained on different modalities, RGB and depth. Also, it can be considered as a same-dataset multitask learning approach, since the two different modalities are of the same dataset. Lastly, the proposed approach can be further categorized as a homogeneous multitask learning approach, since both modalities are used for the same task, gesture recognition.

5.5.1 Video Preprocessing

In computer vision, deep learning models have a fixed input size, but on the other hand, videos could have variable length. So, the videos need to be preprocessed to be of a fixed length. Some of the most common methods to preprocess videos are to split each video in clips of fixed frame length or to sample a fixed number of frames from each video.

Video Split in Clips

In this approach, each video is split into clips of a fixed frame length. A standard frame length F is chosen and then the number of clips per video is calculated by dividing the total number of frames of the video by the length of the clip and rounding up to the nearest integer. After, the clips are created by taking the first F frames of the video, then the next F frames and so on, until the end of the video. The step between the start of a clip and the start of the next clip is possible to not be equal to the length of the clip, so overlapping clips are created. Following this splitting technique, each video corresponds to a number of clips. These clips are then used as samples for the model, leading to two methods to evaluate the model's performance, per video and per clip.

- **Per Video:** The model is trained and tested on the clips of each video separately. The model's prediction for each video is extracted by the average of the predictions of its clips.
- **Per Clip:** The model is trained and tested on each clip separately. The model's prediction for each clip defines and its performance.

This approach has the advantage of using the whole video as input to the model, but it has the disadvantage of creating a large number of samples, which can lead to computational inefficiency.

Video Sampling

In this approach, a fixed number of frames is sampled from each video. A standard number of frames N is chosen and then the frames are downsampled or upsampled to match the number of frames to N. A common method to downsample frames is to sample the frames uniformly, while a common method to upsample frames is to let the video complete and the missing frames are filled with the last frame of the video, creating a still video after the video.

This method has the advantage of creating a fixed number of samples for each video, which leads to computational efficiency, but it has the disadvantage of lowering the video's temporal information clarity.

In this diploma thesis, the video sampling method is used to preprocess the videos, due to its computational efficiency and previous work successfully used it without hurting the performance of the model. The number of frames N is chosen to be 32, which is a common number of frames used in video classification tasks.

5.5.2 Multi-task Dataset

The two proposals for the multi-task learning approach on action and gesture recognition and on RGB and Depth require samples of different datasets and different modalities respectively. So, the standard dataset structure, where batches of samples are created from the same dataset of the same modality, is not applicable. Instead, the dataset is implemented according to the multi-task learning dataset structure implemented in [60]. Specifically, J. Prellberg and O. Kramer first split the dataset into subsets to create task specific datasets. Then, they combined these task specific datasets into a dataset, where each sample is a dictionary, with keys representing the tasks and values representing the samples of the tasks, as can be seen for a two task example in Fig. 5.5.1. This dataset structure is a multi-task dataset and is followed in this diploma thesis.

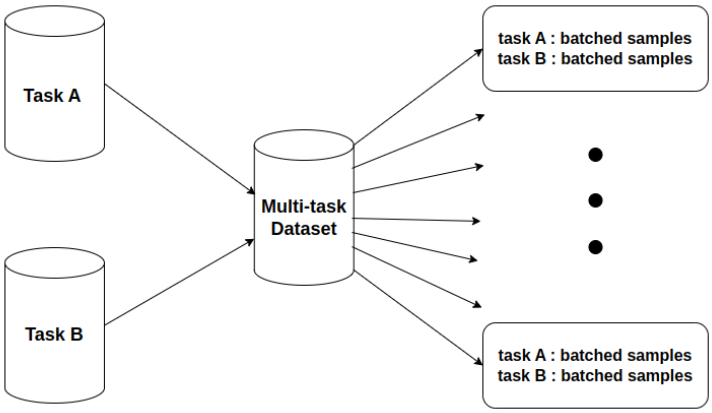


Figure 5.5.1: Multi-task Dataset Structure

For the multi-task dataset consisting of actions and gestures the action and the gesture datasets were combined as explained before, so that each sample of the multi-task dataset is a dictionary with keys representing the tasks of action and gesture recognition and values representing the samples of the tasks, correspondingly. For the multi-task dataset consisting of RGB and Depth modalities, using a similar approach led to each sample of the dataset to be a dictionary with keys representing the modalities of RGB and Depth and values representing the samples of the modalities, respectively.

5.5.3 Multi-task Learning Architectures

The weight sharing methods used to achieve the joint parameter optimization of the two tasks as multi-task learning literature suggests are hard parameter sharing, soft parameter sharing and learned weight sharing. These methods are so robust that any deep learning architecture can be used as a backbone model for the multi-task learning framework. In this diploma thesis, the model used as a backbone model is the ResNet-18 3D. The ResNet-18 3D architecture is chosen because it is a widely used architecture for video classification tasks and many variants of it exist that can be used for better fitting the data. Also, the ResNet-18 3D has publicly available pre-trained weights on the Kinetics-400 dataset, which can be used for better performance on the tasks individually.

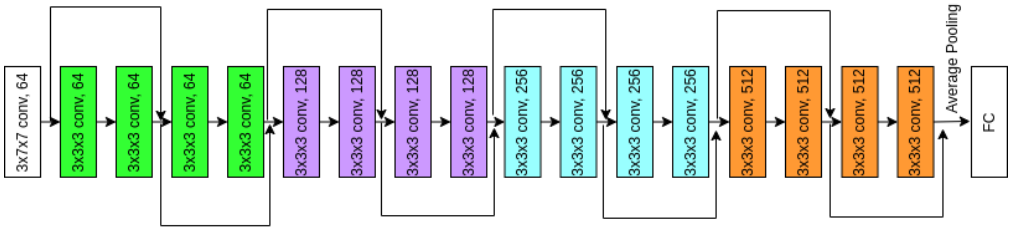


Figure 5.5.2: ResNet-18 3D architecture

To be more specific, the ResNet-18 3D takes as input a 5-D tensor representing a 4-D volume or video clip. The dimensions typically include the batch size, the number of channels, the depth, the height and the width of the input data. As illustrated in Fig. 5.5.2, the ResNet-18 3D model is a 3D convolutional neural network architecture, which is a variant of the ResNet architecture, that is designed to work with 3D data, such as video data. Specifically, the ResNet-18 3D architecture consists of 18 layers, including 17 convolutional layers and 1 fully connected layer. Each group of coloured convolutional layers in the figure represents a ResNet-Layer, which is a set of Residual Blocks. Consequently, each Residual Block is a group of convolutional layers, that are connected by a skip connection. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function, which are not depticted in the figure for simplicity. The skip connections, on the other hand, are used to add the input of a layer to the output of a layer further down the network, so downsampling is performed on the input of the residual block to match the dimensions of the outputs using a max pooling layer. The skip connection allows the gradient to flow through the network more easily, which helps to prevent the vanishing gradient problem.

So, this architecture is altered to fit the different weight sharing methods in each multi-task learning approach tested. In all three multi-task learning approaches, the first layers use pretrained weights from the Kinetics-400 dataset, while the last layers are trained from scratch, as represented with a red colour outline and green colour outline correspondingly, as can be seen in Fig. 5.5.3, Fig. 5.5.4 and Fig. 5.5.5. This is done to leverage the advantages of transfer learning, while also allowing the model to benefit from multi-task learning.

Hard Parameter Sharing

As mention before, the hard parameter sharing method is achieved by sharing some of the hidden layers between all tasks, in order to let all tasks optimize these layers' parameters jointly, while some layers are implemented to be task-specific and are trained independently on each task. According to this hypothesis, the ResNet-18 3D architecture keeps all but the last layer the same and the last layer is replaced by two task-specific output layers, as can be seen in Fig. 5.5.3. So, weights are shared between the two tasks up to the last layers, which are task-specific.

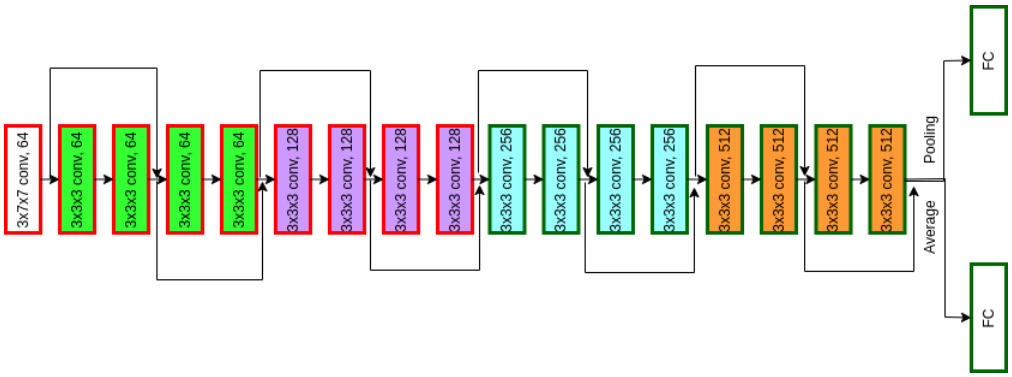


Figure 5.5.3: Hard Parameter Sharing ResNet-18 3D Architecture

Soft Parameter Sharing

In the soft parameter sharing method, task specific neural networks are created and connections between certain layers are used, thus allowing the tasks to share information. The ResNet-18 3D architecture is altered to fit the soft parameter sharing method, as can be seen in Fig. 5.5.4. Two task-specific ResNet-18 3D architectures are created and cross-stitch units are used to connect the two architectures, so that the tasks can share information between them. So, the weights between the two networks are jointly optimized, with the help of the cross-stitch units, while the weights inside each network are optimized independently.

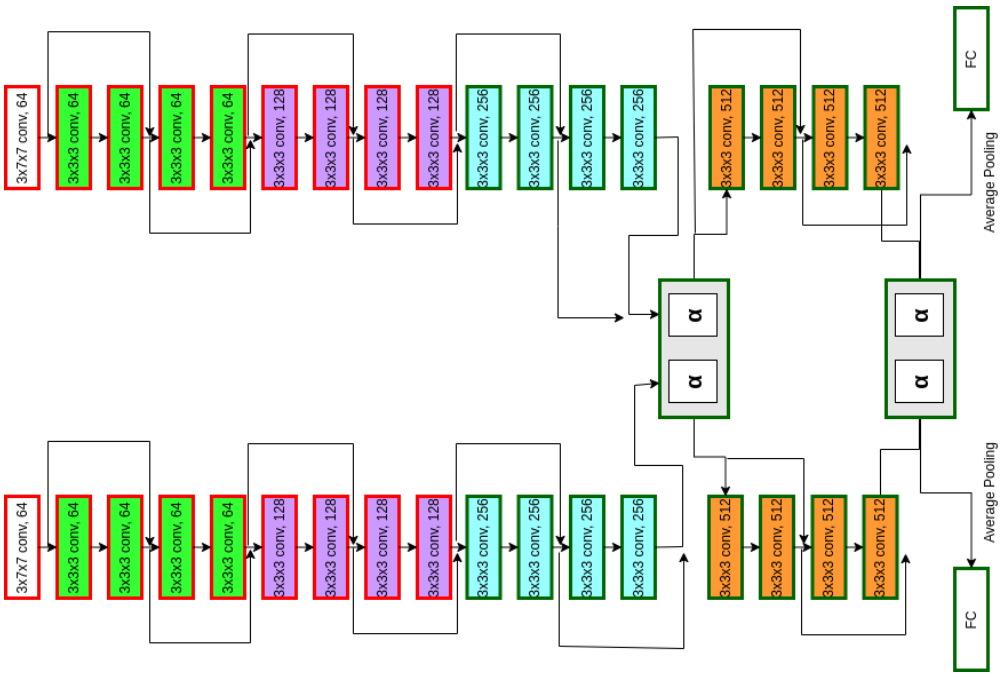


Figure 5.5.4: Soft Parameter Sharing ResNet-18 3D Architecture

Learned Weight Sharing

In the learned weight sharing method, task specific neural networks are created and some of the layers are shared between the networks. The ResNet-18 3D architecture is adjusted to fit the learned weight sharing method, as can be seen in Fig. 5.5.5. As explained before, a set of weights is assigned to the shared layers to achieve weight sharing. This set of weights as well as the weights of the model are optimized during training, by alternative optimization steps from the Natural Evolution Strategy (NES) algorithm and the Stochastic Gradient Descent (SGD) algorithm.

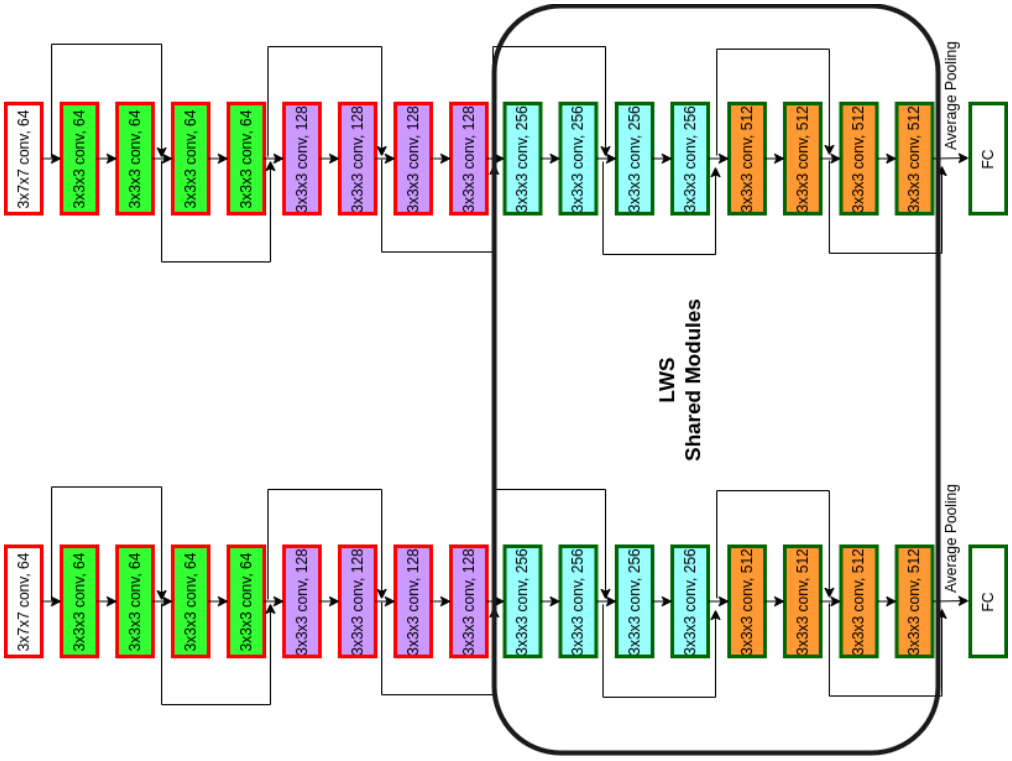


Figure 5.5.5: Learned Weight Sharing ResNet-18 3D Architecture

5.5.4 Training Scheme

A standard dataset structure for a single task learning problem consists of batches of samples, that are drawn from the dataset and fed through the model. A standard training scheme for a dataset like this, would be to take all the batches of the dataset pass them through the model one by one and update the model's weights for each batch, according to the optimization problem of the model, which is to minimize the loss between the model's predictions and the ground truth labels. This process where all the batches of the dataset are passed through the model once is called an epoch. This training process is repeated for a number of epochs, until the model converges to an optimal solution.

The Multi-task Dataset structure used in this diploma thesis does not follow the standard dataset structure, since the batches of the dataset consist of samples of different tasks, consisting of datasets with unequal number of samples. This uneven distribution of samples can lead to false results in evaluation, since the multi-task learning methods will be compared to the single task learning methods. If a training scheme that has epochs were to be used, the performance of the model cannot be compared to the single task learning methods, since the model in one epoch will iter through all the sample of the bigger dataset once, but for the smaller dataset some samples will be iterated more than once in the same epoch. So, if we were to compare the performance of the multi-task learning methods to the single task learning methods for the same number of epochs, this approach might lead to false results, giving the impression that the multi-task learning methods are better than the single task learning methods, but the truth will be that these models will have seen more samples in the same number of epochs.

To avoid this problem, an alternative training scheme is implemented, for both multi-task and single task learning methods, where the model is trained for a number of iterations, instead of epochs. In each iteration, a batch of samples is drawn from the multi-task dataset and fed through the model. The model's weights are updated for each batch, according to the optimization problem of the model. This process is repeated for a number of iterations, until the model converges to an optimal solution. This training scheme allows the model to see the same number of samples in each iteration, so the performance of the multi-task learning methods can be compared to the single task learning methods, since the model will have seen the same number of samples in the same number of iterations.

Chapter 6

Experimental Results

6.1 Baseline Architectures Results

In order to prove the effectiveness of the proposed ideas, the performance of each of the multi-task learning (MTL) models will be compared against the baseline models. The baseline models are single-task learning models (STL) trained separately on each task. The task of action recognition will be represented by two datasets, the UCF-101 and the NTU-RGB+D dataset, while the task of gesture recognition will also be represented by two datasets, the IsoGD and the NVGesture dataset. All models are trained using only the RGB modality of the data. The models that are trained on gesture datasets are trained also using the depth modality of the data, to have a baseline for the multi-modal models. The performance of the models will be evaluated using the accuracy metric, where the accuracy is the percentage of the correctly classified samples. Both single-task learning and multi-task learning models will be trained for 20,000 iterations, to have a fair comparison between them.

The baseline neural network chosen to be used in the STL and MTL experiments is the ResNet-18 3D architecture. The ResNet-18 3D architecture is chosen because it is a widely used architecture for video classification tasks and many variants of it exist that can be used for better fitting the data. Also, the ResNet-18 3D has publicly available pre-trained weights on the Kinetics-400 dataset, which can be used for better performance on the tasks individually.

To be accurate we evaluate the performance of the ResNet-18 3D architecture on the UCF-101, NTU-RGB+D, IsoGD, and NVGesture for different weight initialization for the use of pre-trained weights on the Kinetics-400 dataset, were different amount of parameters are left trainable during the training process. Each table describes the number of fine-tuned layers of each model, the trainable parameters and the accuracy of the model on the respective dataset. In fine-tuned layers column, we refer to the number of Residual Layers that are left trainable, so layer 1 refers to the first Residual Layer, which includes 4 convolutional layers as explained before, and so the pattern goes on.

6.1.1 ResNet on UCF-101

In the following table, the performance of the ResNet-18 3D architecture on the split-1 UCF-101 dataset is shown. From the results, it can be seen that the best performance is achieved when the weights are initialized with the pre-trained weights on the Kinetics-400 dataset. In particular, the accuracy of the model increases as the more layers are initialized with the pre-trained weights and keeping these weights fixed. The best performance is achieved when all layers but the last fully connected layer are initialized with the pre-trained weights and kept "frozen", so that they are not updated during training. This approach achieves a high accuracy of 85.33% on the split-1 of UCF-101 dataset, due to the fact that the pre-trained weights on the Kinetics-400 dataset are close to the optimal weights for the UCF-101 dataset.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33,218,085	23.82
Kinetics-400	ALL	33,218,085	24.12
Kinetics-400	layers 1, 2, 3, 4 + fc	33,189,733	32.45
Kinetics-400	layers 2, 3, 4 + fc	32,746,853	41.27
Kinetics-400	**layers 3, 4 + fc**	**31,189,093**	**58.26**
Kinetics-400	layers 4 + fc	24,960,613	77.32
Kinetics-400	fc	51,813	85.33

Table 6.1: Accuarices ResNet-18 on UCF-101 Dataset with Fine Tuning Pretrained Model on Kinetics-400

6.1.2 ResNet on NTU-RGB+D

The ResNet-18 3D architecture is trained on the NTU-RGB+D dataset and the results are shown in the following table. it can be seen that using the pre-trained weights of the Kinetics-400 dataset for the initialization of the ResNet-18 3D architecture on the NTU-RGB+D dataset results in the better performance. The two datasets are in comparable size and the pre-trained weights on the Kinetics-400 dataset even though they are close to the optimal weights for the NTU-RGB+D dataset they are not optimal. So, this architecture needs

more trainable layers to be able to adapt to the NTU-RGB+D dataset. The best performance is achieved when only the last 2 Residual Layers and the fully connected layer are left trainable, while the rest of the layers are initialized with the pre-trained weights and kept "frozen". This approach achieves an accuracy of 55.29% on the subject split of the NTU-RGB+D dataset.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33,227,832	12.76
Kinetics-400	ALL	33,227,832	12.91
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	22.78
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	48.18
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**55.29**
Kinetics-400	layers 4 + fc	24,970,360	56.68
Kinetics-400	fc	61,560	23.45

Table 6.2: Accuarices ResNet-18 on NTU-RGB+D Dataset RGB Data with Fine Tuning Pretrained Model on Kinetics-400

6.1.3 ResNet on IsoGD

In the following tables, the performance of the ResNet-18 3D architecture on the IsoGD dataset is shown. On the first table, only the RGB data of the IsoGD dataset are used, while on the second one, only the depth data of the IsoGD dataset are used. The results are similar, since both models leverage from the information transferred from the pre-trained weights on the Kinetics-400 dataset. The best performance is achieved when the last Residual Layer and the fully connected layer are left trainable, while the rest of the layers are initialized with the pre-trained weights and kept "frozen". This approach achieves an accuracy of 26.92% on the IsoGD dataset using the RGB data and an accuracy of 29.8% on the IsoGD dataset using the depth data. This behavior is due to the fact that the IsoGD dataset is a large dataset of gesture classes and the Kinetics-400 dataset, on which the ResNet model was trained, consists of action classes. So the model needs more trainable layers to be able to adapt to the new dataset.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33.294.009	1.4
Kinetics-400	ALL	33.294.009	1.5
Kinetics-400	layers 1, 2, 3, 4 + fc	33.265.657	7.91
Kinetics-400	layers 2, 3, 4 + fc	32.822.777	18.42
Kinetics-400	**layers 3, 4 + fc**	**31.265.017**	**25.67**
Kinetics-400	layer 4 + fc	25.036.537	26.92
Kinetics-400	fc	127.737	11.75

Table 6.3: Accuarices ResNet-18 on IsoGD Dataset RGB Data with Finetuning Pretrained Model on Kinetics-400

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33.294.009	2.2
Kinetics-400	ALL	33.294.009	2.6
Kinetics-400	layers 1, 2, 3, 4 + fc	33.265.657	14.64
Kinetics-400	layers 2, 3, 4 + fc	32.822.777	19.63
Kinetics-400	**layers 3, 4 + fc**	**31.265.017**	**23.19**
Kinetics-400	layer 4 + fc	25.036.537	29.8
Kinetics-400	fc	127.737	10.6

Table 6.4: Accuarices ResNet-18 on IsoGD Dataset Depth Data with Finetuning Pretrained Model on Kinetics-400

6.1.4 ResNet on NVGesture

The ResNet-18 3D architecture is trained on the NVGesture dataset for the RGB and depth modalities separately and the results are shown in the following tables. These experiments are similar to the ones on the IsoGD dataset, since the best performance is achieved when the last Residual Layer and the fully connected layer are left trainable, while the rest of the layers are initialized with the pre-trained weights and kept "frozen". This approach achieves an accuracy of 25.52% on the NVGesture dataset using the RGB data and an accuracy of 32.29% on the NVGesture dataset using the depth data.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33.182.972	1.1
Kinetics-400	ALL	33.182.972	1.3
Kinetics-400	layers 1, 2, 3, 4 + fc	33.150.745	4.15
Kinetics-400	layers 2, 3, 4 + fc	32.707.865	18.26
Kinetics-400	**layers 3, 4 + fc**	**31.150.105**	**20.54**
Kinetics-400	layer 4 + fc	24.921.625	25.52
Kinetics-400	fc	12.825	8.3

Table 6.5: Accuarices ResNet-18 on NVGesture Dataset RGB Data with Finetuning Pretrained Model on Kinetics-400

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33.182.972	3.01
Kinetics-400	ALL	33.182.972	3.24
Kinetics-400	layers 1, 2, 3, 4 + fc	33.150.745	7.02
Kinetics-400	layers 2, 3, 4 + fc	32.707.865	23.12
Kinetics-400	**layers 3, 4 + fc**	**31.150.105**	**28.71**
Kinetics-400	layer 4 + fc	24.921.625	32.29
Kinetics-400	fc	12.825	14.52

Table 6.6: Accuarices ResNet-18 on NVGesture Dataset Depth Data with Finetuning Pretrained Model on Kinetics-400

6.1.5 ResNet on NTU-RGB+D - split in sets of actions and gestures

The NTU-RGB+D dataset is a large dataset containing a large number of classes including both action classes, such as sitting down and standing up, and gesture classes, such as hand waving and clapping. In order to evaluate the proposed MTL models, which evaluate whether the tasks of action and gesture recognition can be learned simultaneously, the NTU-RGB+D dataset is split into two parts, one containing only the action classes and the other containing only the gesture classes.

The split is done manually by selecting the classes that are considered as action classes and the classes that are considered as gesture classes. This arbitrary split can vary as some classes can be considered as both action and gesture classes. In this work, the gesture classes chosen from the NTU-RGB+D are the following: *clapping (A10), hand waving (A23), pointing to something with finger (A31), rub two hands together (A34), salute (A38), put the palms together (A39), cross hands in front (say stop) (A40), use a fan (with hand or paper)/feeling warm (A49), point finger at the other person (A54), handshaking (A58), hush (quite) (A67), thumb up (A69), thumb down (A70), make ok sign (A71), make victory sign (A72), snapping fingers (A77), apply cream on hand back (A86), shake fist (A93), hands up (both hands) (A95), cross arms (A96), arm circles (A97), arm swings (A98), high-five (A112), finger-guessing game (playing rock-paper-scissors) (A120).*

So, in order to evaluate the proposed MTL models, the ResNet model is trained on the NTU-RGB+D dataset for the split containing only the action classes (**NTU_ar**) and the split containing only the gesture classes from the combinations of the NTU-RGB+D dataset with the IsoGD (**NTU_gr_IsoGD**) and NVGesture (**NTU_gr_NVGesture**) datasets. When combining the sets of gesture classes from the NTU-RGB+D dataset with the IsoGD and NVGesture datasets, some of the classes are common between the datasets.

The common classes are *ok sign, victory sign* for the combination of NTU-RGB+D with IsoGD and *ok sign, thumb up* for the combination of NTU-RGB+D with NVGesture. In the new sets of NTU-RGB+D with IsoGD gesture classes and NTU-RGB+D with NVGesture gesture classes, these classes have samples from the NTU-RGB+D dataset and the IsoGD and NVGesture datasets, respectively.

The results presented in the following tables show similar behavior to the aforementioned experiments. For the model trained on the action classes split, the best performance is achieved when the last Residual Layer and the fully connected layer are left trainable, while the rest of the layers are initialized with the pre-trained weights and kept "frozen". This approach achieves an accuracy of 58.72% on the subject split of the NTU-RGB+D dataset. For the model trained on the gesture classes split with the IsoGD dataset, the best performance is achieved when the last 2 Residual Layers and the fully connected layer are left trainable, while the rest of the layers are initialized with the pre-trained weights and kept "frozen". This approach achieves an accuracy of 42.68% on the subject split of the NTU-RGB+D dataset. For the model trained on the gesture classes split with the NVGesture dataset, the best performance is achieved when the last 2 Residual Layers and the fully connected layer are left trainable, while the rest of the layers are initialized with the pre-trained weights and kept "frozen". This approach achieves an accuracy of 67.23% on the subject split of the NTU-RGB+D dataset.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33,227,832	16.88
Kinetics-400	ALL	33,227,832	18.12
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	29.93
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	51.33
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**58.72**
Kinetics-400	layers 4 + fc	24,970,360	60.94
Kinetics-400	fc	61,560	31.95

Table 6.7: Accuarices ResNet-18 on NTU_ar RGB Data with Fine Tuning Pretrained Model on Kinetics-400

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33,227,832	4.82
Kinetics-400	ALL	33,227,832	6.33
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	9.38
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	39.72
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**42.68**
Kinetics-400	layers 4 + fc	24,970,360	42.6
Kinetics-400	fc	61,560	29.49

Table 6.8: Accuarices ResNet-18 on NTU_gr_IsoGD RGB Data with Fine Tuning Pretrained Model on Kinetics-400

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33,227,832	24.23
Kinetics-400	ALL	33,227,832	29.67
Kinetics-400	layers 1, 2, 3, 4 + fc	33,199,480	41.07
Kinetics-400	layers 2, 3, 4 + fc	32,756,600	56.55
Kinetics-400	**layers 3, 4 + fc**	**31,198,840**	**67.23**
Kinetics-400	layers 4 + fc	24,970,360	65.96
Kinetics-400	fc	61,560	43.19

Table 6.9: Accuarices ResNet-18 on NTU_gr_NVGesture RGB Data with Fine Tuning Pretrained Model on Kinetics-400

6.1.6 Summary of the ResNet baseline models

In this section, the performance of the ResNet-18 3D architecture on the UCF-101, NTU-RGB+D, IsoGD, and NVGesture datasets is evaluated. The results show that the best performance is achieved when some of the layers are left trainable, while the rest of the layers are initialized with the pre-trained weights on the Kinetics-400 dataset and kept "frozen". To be able, to leverage both outcomes of transfer learning and multi-task learning the proposed MTL models will have the last 2 Residual Layers and the fully connected layer left trainable, while the rest of the layers will be initialized with the pre-trained weights and kept "frozen". This structure is followed so that the multi-task learning method can work in the last trainable layers, that learn the task-specific features, while the first the layers, that learn more common features of the tasks, will not be updated during the training process, but will be set according to the pre-trained weights on the Kinetics-400 dataset.

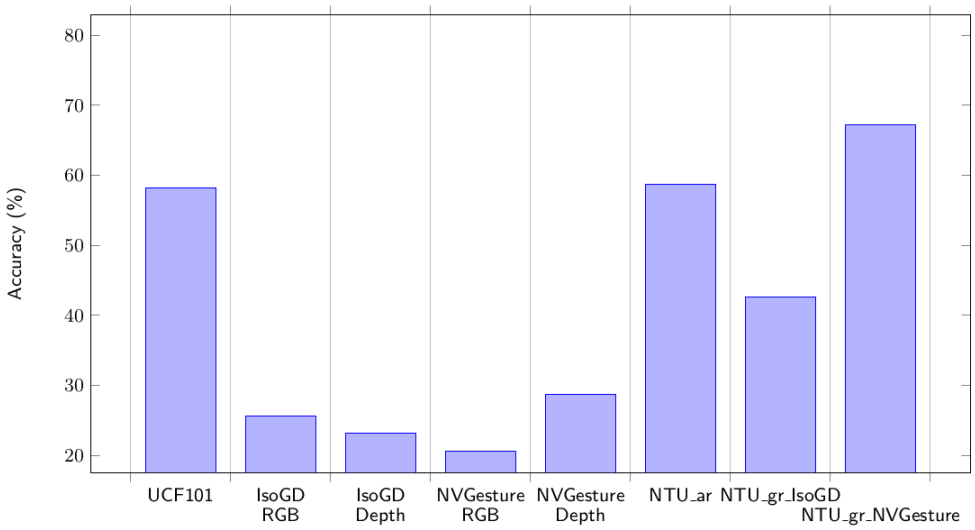


Figure 6.1.1: Single-task learning results for the ResNet-18 3D architecture

6.2 Action and Gesture Recognition Multi-task Learning

In order to prove the effectiveness of the proposed multi-task learning for action and gesture recognition, 3 different multi-task learning methods and 3 different multi-task loss calculation methods are implemented, thus resulting to 9 different experiments for each pair of action and gesture datasets. For the multi-task learning methods, the hard parameter sharing, the cross-stitch and the learned weight sharing methods are chosen, while for the multi-task loss calculation methods, the average, the uncertainty in weighing loss and the dynamic weight average are used.

The accuracies of the single task learning ResNet models, with the last 2 Residual Layers and the fully connected layer left trainable, and the rest of the model initialized with the pre-trained weights and kept "frozen", are summarized as follows.

- UCF-101: 58.26%
- IsoGD: 25.67%
- NVGesture: 20.54%
- NTU_ar (NTU-RGB+D action classes): 58.72%
- NTU_gr_IsoGD (NTU-RGB+D gesture classes + IsoGD): 42.68%
- NTU_gr_NVGesture (NTU-RGB+D gesture classes + NVGesture): 67.23%

Also, it is important to mention that the trainable parameters of the single task learning models are appromaximately 35 million, for the chosen structure of this neural network. This aspect is important to be compared with the trainable parameters of the multi-task learning models, in order to avoid leading to the conclusion that the MTL models are better just because they have more trainable parameters, but focus only the effectiveness of sharing knowledge between the tasks.

6.2.1 Hard Parameter Sharing (HPS)

Hard Parameter Sharing model are simple alternatives to single task learning models, since they differ only in the last task-specific layers, thus the parameters of the following models are a little bit more than the single task learning models.

When the hard parameter sharing method is used on the UCF-101 and the IsoGD datasets the multi-task learning framework achieves a performance of 57.36% and 25.45% respectively, as the best result. This performance is appromaximately the same as the single task learning model, meaning that this method is not the best choice for this set of datasets.

Loss Method	Dataset	
	UCF-101	IsoGD
Single Task Learning		
-	58.26	25.67
Hard Parameter Sharing		
average	54.16	20.49
dwa	57.36	25.45
uncertainty	60.53	20.02

Table 6.10: Accuarices of Hard Parameter Sharing (HPS) ResNet-18 on UCF-101 and IsoGD Datasets

On the other hand, when the hard parameter sharing method is used on the UCF-101 and the NVGesture datasets the multi-task learning framework achieves a performance of 66.24% and 33.68% respectively, as the best result, for the uncertainty in weighing loss as the multi-task loss calculation method. This performance is better than the single task learning model for both datasets.

Loss Method	Dataset	
	UCF-101	NVGesture
Single Task Learning		
-	58.26	20.54
Hard Parameter Sharing		
average	64.34	27.86
dwa	58.6	25.57
uncertainty	66.24	33.68

Table 6.11: Accuarices of Hard Parameter Sharing (HPS) ResNet-18 on UCF-101 and NVGesture Datasets

For the multi-task learning framework on the NTU-RGB+D and the IsoGD datasets the best performance is achieved when the hard parameter sharing method is used with the average loss calculation method, with an accuracy of 70.65% and 60.25% respectively, outperforming the single task learning model performances by over 10% for both datasets.

Loss Method	Dataset	
	NTU_ar	NTU_gr_IsoGD
Single Task Learning		
-	58.72	42.68
Hard Parameter Sharing		
average	70.65	60.25
dwa	64.25	58.87
uncertainty	49.42	56.57

Table 6.12: Accuarices of Hard Parameter Sharing (HPS) ResNet-18 on NTU_ar and NTU_gr_IsoGD

For the multi-task learning framework on the NTU-RGB+D and the NVGesture datasets similar behavior is observed, as the best performance is achieved when the hard parameter sharing method is used with the average loss calculation method, achieving an accuracy of 72.53% and 74.33% respectively.

Loss Method	Dataset	
	NTU_ar	NTU_gr_NVGesture
Single Task Learning		
-	58.72	67.23
Hard Parameter Sharing		
average	71.91	74.33
dwa	72.53	73.85
uncertainty	72.51	72.80

Table 6.13: Accuarices of Hard Parameter Sharing (HPS) ResNet-18 on NTU_ar and NTU_gr_NVGesture

All in all, hard parameter sharing models, using different combinations of datasets and loss calculation methods, could lead to optimal results since all the combinations outperform the single task learning model, thus proving the effectiveness of weight sharing. These models only have a little bit more trainable parameters than the single task learning models, yet still achieved better results. Only the combination of the UCF-101 and the IsoGD datasets could not outperform its STL counterparts, due to the fact that the gesture dataset was quite bigger than the action dataset, thus the model was not able to learn the action dataset properly.

6.2.2 Soft Parameter Sharing (SPS)

For the soft parameter sharing method, the cross-stitch networks are chosen, which require to implement task-specific networks, and then combine them using a cross-stitch unit. This structure is more complex than the single task learning models, thus the number of trainable parameters is higher than the single task learning models, appromaximately twice the size (70 million), since the model will learn two tasks. Moreover,

in the cross-stitch unit, the parameter σ is used to control the amount of information that is shared between the tasks. As mentioned in the original paper the choice can not be arbitrary, but it should be chosen emperically for each dataset. So, we experiment with different values of σ to find the optimal one for each dataset, choosing between 0.2 and 0.8.

When cross-stitch networks are used for the set of UCF-101 and IsoGD datasets, the best performance is achieved when the parameter σ is set to 0.8, with the loss method to be the uncertainty in weighing loss, achieving an accuracy of 70.24% and 35.79% respectively. This performance is better than the single task learning models, a behavior that it was not observed for the hard parameter sharing method.

Loss Method	Dataset	
	UCF-101	IsoGD
Single Task Learning		
-	58.26	25.67
Cross-Stitch (s=0.2)		
average	67.64	26.6
dwa	62.2	30.96
uncertainty	67.62	26.25
Cross-Stitch (s=0.8)		
average	64.22	26.78
dwa	63.1	32.03
uncertainty	70.24	35.79

Table 6.14: Accuarices of Soft Parameter Sharing (SPS) ResNet-18 on UCF-101 and IsoGD Datasets

Moreover, for the UCF-101 and the NVGesture datasets, the best performance is achieved when the parameter σ is set to 0.2, with the loss method to be again the uncertainty in weighing loss, with an accuracy of 71.66% and 43.24%.

Loss Method	Dataset	
	UCF-101	NVGesture
Single Task Learning		
-	58.26	20.54
Cross-Stitch (s=0.2)		
average	69.94	36.17
dwa	71.66	37.42
uncertainty	71.66	43.24
Cross-Stitch (s=0.8)		
average	73.14	35.76
dwa	67.94	34.93
uncertainty	68.75	39.09

Table 6.15: Accuarices of Soft Parameter Sharing ResNet-18 on UCF-101 and NVGesture Datasets

For the datasets of NTU-RGB+D and IsoGD, the best performance is achieved for the soft parameter sharing method, when the parameter σ is set to 0.2, with the loss method to be the uncertainty in weighing loss, with an accuracy of 74.45% and 62.14%.

Loss Method	Dataset	
	NTU_ar	NTU_gr_IsoGD
Single Task Learning		
-	58.72	42.68
Cross-Stitch (s=0.2)		
average	72.27	60.13
dwa	71.84	60.91
uncertainty	74.45	62.14
Cross-Stitch (s=0.8)		
average	71.82	59.3
dwa	72.02	60.55
uncertainty	73.66	60.8

Table 6.16: Accuarices of Soft Parameter Sharing (SPS) ResNet-18 on NTU_ar and NTU_gr_IsoGD

For the datasets of NTU-RGB+D and NVGesture, the best performance is achieved for the soft parameter sharing method, when the parameter σ is set to 0.8, with the loss method to be the dynamic weight average, with an accuracy of 74.03% and 75.51%.

Loss Method	Dataset	
	NTU_ar	NTU_gr_NVGesture
Single Task Learning		
-	58.72	67.23
Cross-Stitch (s=0.2)		
average	74.2	74.49
dwa	73.51	74.59
uncertainty	74.17	73.9
Cross-Stitch (s=0.8)		
average	72.65	74.21
dwa	74.03	75.51
uncertainty	72.25	73.54

Table 6.17: Accuarices of Soft Parameter Sharing (SPS) ResNet-18 on NTU_ar and NTU_gr_NVGesture

To sum up, for the soft parameter sharing method, all the combinations of datasets and loss calculation methods outperform the single task learning model, which leads to the conclusion that the soft parameter sharing method is an effective approach for training the model on the action and gesture recognition. To be more precise, the uncertainty in weighing loss method is the best choice for most of the combinations, while the value for the parameter σ differs for each dataset.

6.2.3 Learned Weight Sharing (LWS)

For the implementation of a learned weight sharing model, task-specific networks are implemented, thus leading to a model with a number of trainable parameters appromaximately twice the size of the single task learning models (70 million), since the model will learn two tasks.

When the learned weight sharing method is used on the UCF-101 and the IsoGD datasets, the best performance is achieved when the loss method is the dynamic weight average, with an accuracy of 70.58% and 34.9%.

Loss Method	Dataset	
	UCF-101	IsoGD
Single Task Learning		
-	58.26	25.67
Learned Weight Sharing		
average	64.39	35.2
dwa	70.58	34.9
uncertainty	64.79	28.7

Table 6.18: Accuarices of Learned Weight Sharing (LWS) ResNet-18 on UCF-101 and IsoGD Datasets

For the UCF-101 and the NVGesture datasets, the best performance is achieved again when the loss method is the dynamic weight average, with an accuracy of 71.42% and 43.04%.

Loss Method	Dataset	
	UCF-101	NVGesture
Single Task Learning		
-	58.26	20.54
Learned Weight Sharing		
average	69.73	39.62
dwa	71.42	43.04
uncertainty	72.27	37.21

Table 6.19: Accuarices of Learned Weight Sharing (LWS) ResNet-18 on UCF-101 and NVGesture Datasets

For the NTU-RGB+D and the IsoGD datasets, the best performance is achieved when the loss method is the average loss calculation, with an accuracy of 73.03% and 56.27%.

Loss Method	Dataset	
	NTU_ar	NTU_gr_IsoGD
Single Task Learning		
-	58.72	42.68
Learned Weight Sharing		
average	73.03	56.27
dwa	72.6	53.64
uncertainty	72.48	53.07

Table 6.20: Accuarices of Learned Weight Sharing (LWS) ResNet-18 on NTU_ar and NTU_gr_IsoGD

When the learned weight sharing method is used on the NTU-RGB+D and the NVGesture datasets, the best performance is achieved when the loss method is the dynamic weight average, with an accuracy of 73.41% and 75.47%.

Loss Method	Dataset	
	NTU_ar	NTU_gr_NVGesture
Single Task Learning		
-	58.72	67.23
Learned Weight Sharing		
average	73.45	74.43
dwa	73.41	75.47
uncertainty	72.96	76.4

Table 6.21: Accuarices of Learned Weight Sharing (LWS) ResNet-18 on NTU_ar and NTU_gr_NVGesture

So, for the learned weight sharing method, the best performance is achieved when the loss method is the dynamic weight average, for most of the combinations of datasets. This leads to the conclusion that the learned weight sharing method is effective for training the model on the action and gesture datsets.

6.2.4 Summary of the MTL methods results

So, for the action and gesture recognition datasets, the best performances for each combination of sets with the best multi-task loss calculation method are summarized in the Fig. 6.2.1.

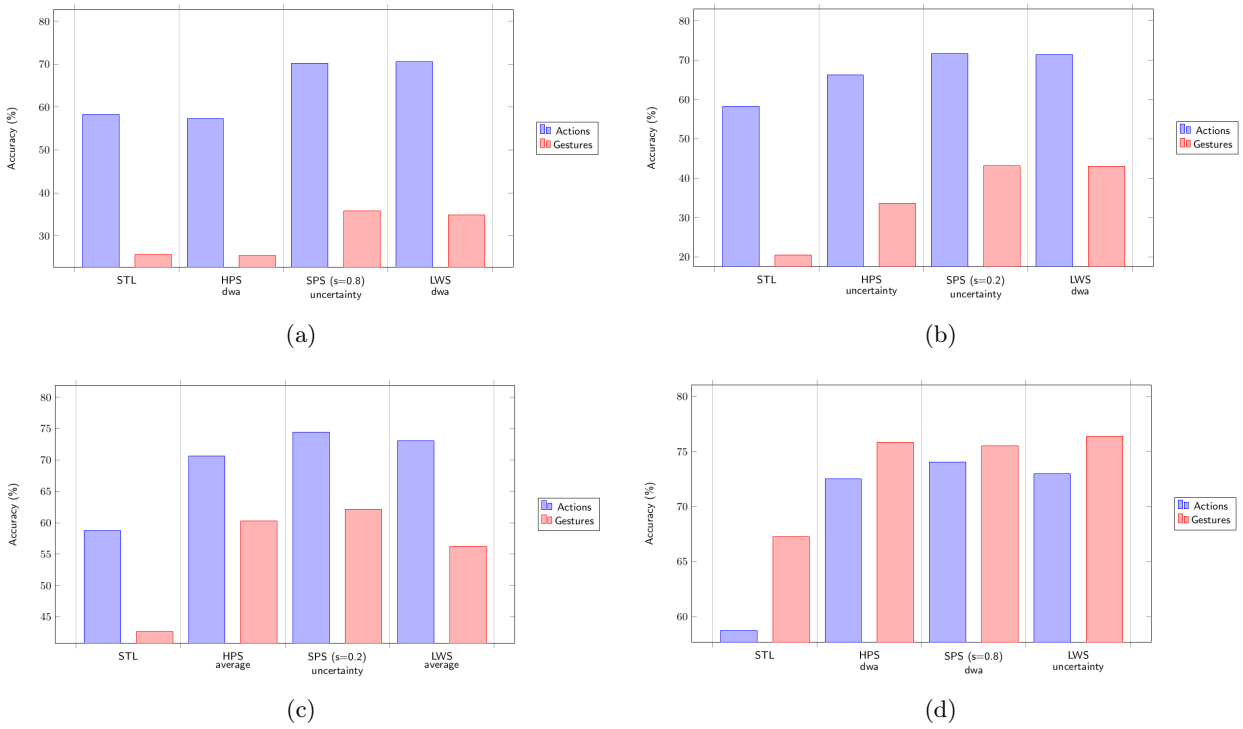


Figure 6.2.1: Results of the MTL methods for action and gesture datasets: (a) UCF101-IsoGD, (b) UCF101-NVGesture, (c) NTU_ar-NTU_gr_IsoGD, (d) NTU_ar-NTU_gr_NVGesture.

As can be seen from these representations, the choice of the multi-task loss calculation method is not standard, but it is shown that more complex methods, like the uncertainty in weighing loss and the dynamic weight average, are more effective than the average loss calculation method. Moreover, the choice of the multi-task learning method is also important, as the soft parameter sharing method outperforms the other methods. The Learned Weight Sharing method is also effective, yet still achieves lower results than the cross-stitch networks. The hard parameter sharing method is the least effective, but due to its simplicity and the fact that has appromaximately the same number of trainable parameters as the single task learning models, it is still a good choice for training the MTL model with action and gesture datasets. Overall, the multi-task

learning methods are effective for training the model on the action and gesture datasets, as they outperform the single task learning models, especially when the action dataset is greater in size than the gesture dataset, thus proving the effectiveness of sharing knowledge between the tasks.

6.3 RGB and Depth Modalities Multi-Modal Results

The ResNet-18 3D baseline architecture is altered to accept multimodal data as input, by creating two branches of different Residual Layers, one for the RGB data and one for the depth data. The output of these branches is then fused using an average fusion at the end of the model, thus creating the final prediction is made. This model is trained on the IsoGD and NVGesture datasets, where RGB and Depth data are available. The model is trained for 20,000 iterations, to set a common ground for comparison with the proposed multi-task multi-modal learning framework. As in the previous experiments, the performance of the model is evaluated using the top-1 accuracy metric, while pretrained weights from the Kinetics-400 dataset are used for initialization.

6.3.1 ResNet on IsoGD

For the multimodal data of the IsoGD dataset, the performance of the ResNet-18 3D follows the same pattern as the single modality case, as shown in the following table. Moreover, as described in literature, the multimodal model performs better than the single modality models, as the model can learn from both modalities and fuse the information to make a better prediction. The best performance is achieved when the last two Residual layers and the fully connected layer are left trainable, while the rest of the model is "frozen". A structure that achieves an accuracy of 30.61% on the IsoGD dataset.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33.294.009	8.76
Kinetics-400	ALL	33.294.009	9.87
Kinetics-400	layers 1, 2, 3, 4 + fc	33.265.657	14.26
Kinetics-400	layers 2, 3, 4 + fc	32.822.777	25.85
Kinetics-400	**layers 3, 4 + fc**	**31.265.017**	**30.61**
Kinetics-400	layer 4 + fc	25.036.537	28.91
Kinetics-400	fc	127.737	15.34

Table 6.22: Accuarices ResNet-18 on IsoGD Dataset RGB and Depth Data with Finetuning Pretrained Model on Kinetics-400

6.3.2 ResNet on NVGesture

The ResNet-18 3D model trained on the NVGesture dataset for the multimodal case presents the same behavior as the IsoGD dataset, as can be seen in the following table, with the best structure to be leaving the last Residual layer and the last fully connected layer trainable, achieving an accuracy of 46.49% on this dataset.

Pretrained	Fine Tuned Layers	Trainable Parameters	Accuracy
from scratch	-	33.182.972	5.19
Kinetics-400	ALL	33.182.972	5.22
Kinetics-400	layers 1, 2, 3, 4 + fc	33.150.745	9.95
Kinetics-400	layers 2, 3, 4 + fc	32.707.865	28.82
Kinetics-400	**layers 3, 4 + fc**	**31.150.105**	**32.57**
Kinetics-400	layer 4 + fc	24.921.625	46.49
Kinetics-400	fc	12.825	24.69

Table 6.23: Accuarices ResNet-18 on NVGesture Dataset RGB and Depth Data with Finetuning Pretrained Model on Kinetics-400

6.3.3 Summary of the ResNet multimodal models

In this section, the performance of the ResNet-18 3D multi-modal model on the IsoGD and NVGesture datasets was evaluated. The results show imporevement in comparison to the single modality models. For

the evaluation of the multi-task multi-modal learning framework, the structure of the model is chosen to be the same as previous models, where the last 2 Residual Layers and the fully connected layer are left trainable, while the rest of the model is kept "frozen".

6.4 RGB and Depth Modalities Multi-task Learning

In order to prove the effectiveness of the proposed multi-task learning for action and gesture recognition, 3 different multi-task learning methods and 3 different multi-task loss calculation methods are implemented, thus resulting to 9 different experiments. For the multi-task learning methods, the hard parameter sharing, the cross-stitch and the learned weight sharing methods are chosen, while for the multi-task loss calculation methods, the average, the uncertainty in weighing loss and the dynamic weight average are used.

The accuracies of the single task learning ResNet models, with the last 2 Residual Layers and the fully connected layer left trainable, and the rest of the model initialized with the pre-trained weights and kept "frozen", are summarized as follows.

- IsoGD (RGB): 25.67%
- IsoGD (Depth): 23.19%
- IsoGD (RGB + Depth): 30.61%
- NVGesture (RGB): 20.54%
- NVGesture (Depth): 28.71%
- NVGesture (RGB + Depth): 32.57%

6.4.1 Multi-Modal Multi-Task Learning on IsoGD

The multimodal multitask learning experiments on the IsoGD dataset are presented in this section for the RGB and Depth modalities. The results presented in Table 6.24 show that for all the experiments, the proposed multi-task multi-modal learning framework outperforms the single task learning model as well as the multi-modal model. The best performance is achieved when the parameter sharing method is chosen to be the soft parameter sharing method, with the cross stitch implementation for $\sigma = 0.8$ and the loss calculation method to be the dynamic weight average, with an accuracy of 48.91%, which is more than 15% better than the single task learning model and the multi-modal model.

Multi-task method	Loss Method	Accuracy
Hard Parameter Sharing	average	43.82
Hard Parameter Sharing	dwa	44.33
Hard Parameter Sharing	uncertainty	46.09
Cross-Stitch (s=0.2)	average	46.95
Cross-Stitch (s=0.2)	dwa	43.66
Cross-Stitch (s=0.2)	uncertainty	45.72
Cross-Stitch (s=0.8)	average	43.45
Cross-Stitch (s=0.8)	dwa	48.91
Cross-Stitch (s=0.8)	uncertainty	45.34
Learned Weight Sharing	average	46.42
Learned Weight Sharing	dwa	47.3
Learned Weight Sharing	uncertainty	46.04

Table 6.24: Accuarices of Multimodal Multitask ResNet-18 on IsoGD Dataset for the RGB and Depth Data

6.4.2 Multi-Modal Multi-Task Learning on NVGesture

The multimodal multitask learning experiments on the IsoGD dataset are presented in this section for the RGB and Depth modalities. The results are presented in Table 6.25 show that for all the experiments, the proposed multi-task multi-modal learning framework outperforms the single task learning model as well as the multi-modal model. The best performance is achieved when the parameter sharing method is chosen to be the soft parameter sharing method, with the cross stitch implementation for $\sigma = 0.8$ and the loss

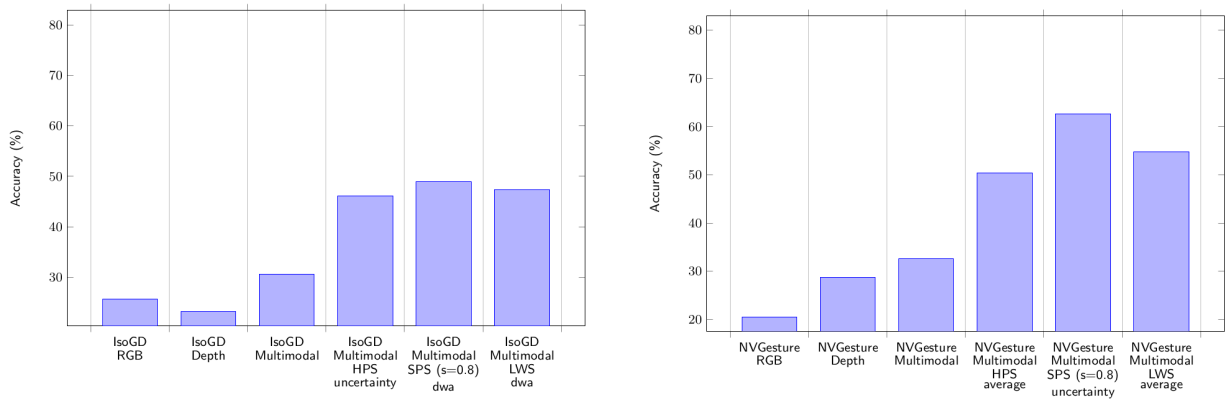
calculation method to be the uncertainty in weighing losses, with an accuracy of 48.91%, which is more than 10% better than the single task learning model and the multi-modal model.

Multi-task method	Loss Method	Accuracy
Hard Parameter Sharing	average	50.41
Hard Parameter Sharing	dwa	46.47
Hard Parameter Sharing	uncertainty	45.23
Cross-Stitch (s=0.2)	average	52,7
Cross-Stitch (s=0.2)	dwa	54.98
Cross-Stitch (s=0.2)	uncertainty	51.87
Cross-Stitch (s=0.8)	average	51.66
Cross-Stitch (s=0.8)	dwa	55.6
Cross-Stitch (s=0.8)	uncertainty	62.66
Learned Weight Sharing	average	54.77
Learned Weight Sharing	dwa	50.41
Learned Weight Sharing	uncertainty	51.24

Table 6.25: Accuarices of Multimodal Multitask ResNet-18 on NVGesture Dataset for the RGB and Depth Data

6.4.3 Summary of Multi-Modal Multi-Task Learning Framework

So, the proposed multi-task multi-modal learning framework is effective for the IsoGD and NVGesture datasets, as the model can learn from both modalities and fuse the information to make a better prediction. The best performance is achieved when the parameter sharing method is chosen to be the soft parameter sharing method, with the cross stitch implementation for $\sigma = 0.8$ and the loss calculation method to be the dynamic weight average for the IsoGD dataset and the uncertainty in weighing losses for the NVGesture dataset, but all the other methods also perform well, as can be seen from Fig. 6.4.1.



(a) (b)

Figure 6.4.1: Multimodal Multi-task learning for RGB and Depth data: (a) IsoGD (b) NVGesture

Chapter 7

Conclusion

7.1 Summary

In this thesis, the effectiveness of the proposed multi-task learning for action and gesture recognition is evaluated. The datasets used for these tasks are human-centric, focusing on the recognition of human actions and gestures in videos. While many real life applications require the recognition of both actions and gestures, the research community handles these tasks seperately. The proposed multi-task learning framework aims to address this issue by learning both tasks simultaneously.

The variety in different actions and gestures datasets used, as well as the different weight sharing methods implemented, with the combination of different loss calculation methods, provide a comprehensive evaluation of the proposed multi-task learning framework. The experiments conducted demonstrate that the proposed multi-task learning framework is effective for the tasks of action and gesture recognition. The results show that the proposed models achieve better performance compared to the single task models, showing the benefits of multi-task learning in action and gesture recognition.

Moreover, a multi-modal multi-task learning model is developed, where RGB and Depth data, are learned jointly. A variety of problems in computer vision uses multi-modal data, as different modalities can provide complementary information. Though these modalities combine for the final prediction, most approaches learn from each modality separately. The proposed multi-modal multi-task learning framework aims to show that learning both RGB and Depth data jointly can improve the performance of the model. The proposed model is evaluated on the respective single task learning models as well as the multi-modal model using all the weight sharing methods and loss calculation methods, that were used for the multi-task learning experiments. The results show that the proposed multi-modal multi-task learning framework is effective for the RGB and Depth data, as the model can learn from both modalities and fuse the information to make a better prediction.

7.2 Future Work

Multi-task learning is a field of machine learning with numerous alternative approaches and methods. In this thesis, an exploration of the baselines of multi-task learning in the context of action and gesture recognition has been conducted. However, there are many other aspects of multi-task learning that could be explored in future work.

One potential avenue for future work is to explore the potential of multi-task learning in the context of other related tasks, such as action localization, action detection, and action segmentation. In addition, the potential of multi-task learning in the context of other architectures, such as 3D CNNs, could be explored.

Another potential avenue for future work is to explore the potential of leveraging the knowledge learned across different datasets and different modalities. This hypothesis leads to the idea of creating a multi-modal multi-task learning model that can learn from multiple datasets and multiple modalities simultaneously.

Moreover, these multi-task learning models use as input in the inference the task of the sample to be predicted. This could be the next step in the evolution of these models, as the model could learn the task of the sample as well. This could be achieved by adding an additional attention mechanism to the model, which could learn the task of the sample and use this information to make a better prediction.

Finally, more research could be conducted on the multi-task multi-modal learning framework, exploring the different fusion mechanisms.

Bibliography

[1] Sun, Z. et al. "Human action recognition from various data modalities: A review". In: *IEEE transactions on pattern analysis and machine intelligence* 45.3 (2022), pp. 3200–3225.

[2] Akula, A., Shah, A. K., and Ghosh, R. "Deep learning approach for human action recognition in infrared images". In: *Cognitive Systems Research* 50 (2018), pp. 146–154.

[3] Horn, B. K. and Schunck, B. G. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[4] Janiesch, C., Zschech, P., and Heinrich, K. "Machine learning and deep learning". In: *Electronic Markets* 31.3 (2021), pp. 685–695.

[5] Rosenblatt, F. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[6] Lopez-Bernal, D. et al. "Education 4.0: teaching the basics of KNN, LDA and simple perceptron algorithms for binary classification problems". In: *Future Internet* 13.8 (2021), p. 193.

[7] Sharma, S., Sharma, S., and Athaiya, A. "Activation functions in neural networks". In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[8] Kandel, I. and Castelli, M. "Transfer learning with convolutional neural networks for diabetic retinopathy image classification. A review". In: *Applied Sciences* 10.6 (2020), p. 2021.

[9] Sarker, I. H. "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions". In: *SN Computer Science* 2.6 (2021), p. 420.

[10] Sazli, M. H. "A brief review of feed-forward neural networks". In: *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering* 50.01 (2006).

[11] O'Shea, K. and Nash, R. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[12] Li, G. et al. "Efficient binary 3D convolutional neural network and hardware accelerator". In: *Journal of Real-Time Image Processing* 19.1 (2022), pp. 61–71.

[13] Gholamalinezhad, H. and Khosravi, H. "Pooling methods in deep neural networks, a review". In: *arXiv preprint arXiv:2009.07485* (2020).

[14] Ioffe, S. and Szegedy, C. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[15] Bjorck, N. et al. "Understanding batch normalization". In: *Advances in neural information processing systems* 31 (2018).

[16] Rguibi, Z. et al. "Cxai: Explaining convolutional neural networks for medical imaging diagnostic". In: *Electronics* 11.11 (2022), p. 1775.

[17] Zhou, Y. "Natural language processing with improved deep learning neural networks". In: *Scientific Programming* 2022 (2022), pp. 1–8.

[18] Rau, F., Soto, I., and Zabala-Blanco, D. "Forescating Mobile Network Traffic based on Deep Learning Networks". In: *2021 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE. 2021, pp. 1–6.

[19] Kingma, D. P. and Ba, J. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[20] Das Dawn, D. and Shaikh, S. H. "A comprehensive survey of human action recognition with spatio-temporal interest point (STIP) detector". In: *The Visual Computer* 32 (2016), pp. 289–306.

[21] Xie, H. et al. "GPU-based fast scale invariant interest point detector". In: *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2010, pp. 2494–2497.

[22] Harris, C., Stephens, M., et al. "A combined corner and edge detector". In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.
[23] Laptev, I. "On space-time interest points". In: *International journal of computer vision* 64 (2005), pp. 107–123.
[24] Lowe, D. G. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.
[25] Bay, H. et al. "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.
[26] Dalal, N. and Triggs, B. "Histograms of oriented gradients for human detection". In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893.
[27] Laptev, I. et al. "Learning realistic human actions from movies". In: *2008 IEEE conference on computer vision and pattern recognition*. IEEE. 2008, pp. 1–8.
[28] Sivic and Zisserman. "Video Google: A text retrieval approach to object matching in videos". In: *Proceedings ninth IEEE international conference on computer vision*. IEEE. 2003, pp. 1470–1477.
[29] Eddy, S. R. "Hidden markov models". In: *Current opinion in structural biology* 6.3 (1996), pp. 361–365.
[30] Suthaharan, S. and Suthaharan, S. "Support vector machine". In: *Machine learning models and algorithms for big data classification: thinking with examples for effective learning* (2016), pp. 207–235.
[31] Xanthopoulos, P. et al. "Linear discriminant analysis". In: *Robust data mining* (2013), pp. 27–33.
[32] Abdi, H. and Williams, L. J. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.
[33] Albawi, S., Mohammed, T. A., and Al-Zawi, S. "Understanding of a convolutional neural network". In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.
[34] Tran, D. et al. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.
[35] Karpathy, A. et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.
[36] Soomro, K., Zamir, A. R., and Shah, M. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).
[37] Zhang, Y. et al. "Egogesture: a new dataset and benchmark for egocentric hand gesture recognition". In: *IEEE Transactions on Multimedia* 20.5 (2018), pp. 1038–1050.
[38] Simonyan, K. and Zisserman, A. "Two-stream convolutional networks for action recognition in videos". In: *Advances in neural information processing systems* 27 (2014).
[39] Chen, Y. et al. "Two-stream network for sign language recognition and translation". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17043–17056.
[40] He, K. et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
[41] Ramzan, F. et al. "A deep learning approach for automated diagnosis and multi-class classification of Alzheimer's disease stages using resting-state fMRI and residual neural networks". In: *Journal of medical systems* 44 (2020), pp. 1–16.
[42] Hara, K., Kataoka, H., and Satoh, Y. "Learning spatio-temporal features with 3d residual networks for action recognition". In: *Proceedings of the IEEE international conference on computer vision workshops*. 2017, pp. 3154–3160.
[43] Wang, Y. et al. "Explainable deep learning for biomarker classification of oct images". In: *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*. IEEE. 2020.
[44] Hara, K., Kataoka, H., and Satoh, Y. "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6546–6555.
[45] Wu, X. et al. "Gesture recognition based on transfer learning". In: *2019 Chinese Automation Congress (CAC)*. IEEE. 2019, pp. 199–202.
[46] Sarhan, N. and Frintrop, S. "Transfer learning for videos: from action recognition to sign language recognition". In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 1811–1815.

[47] Carreira, J. and Zisserman, A. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[48] Liu, J. et al. "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding". In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2684–2701.

[49] Kuehne, H. et al. "HMDB: a large video database for human motion recognition". In: *2011 International conference on computer vision*. IEEE. 2011, pp. 2556–2563.

[50] Wan, J. et al. "Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2016, pp. 56–64.

[51] Molchanov, P. et al. "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4207–4215.

[52] Materzynska, J. et al. "The jester dataset: A large-scale video dataset of human gestures". In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019.

[53] Caruana, R. "Multitask learning". In: *Machine learning* 28 (1997), pp. 41–75.

[54] Zhang, Y. and Yang, Q. "A survey on multi-task learning". In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2021), pp. 5586–5609.

[55] Lakkapragada, A. et al. "Mitigating negative transfer in multi-task learning with exponential moving average loss weighting strategies (student abstract)". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 13. 2023, pp. 16246–16247.

[56] Standley, T. et al. "Which tasks should be learned together in multi-task learning?" In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9120–9132.

[57] Deng, C. et al. "Integrating machine learning with human knowledge". In: *Iscience* 23.11 (2020).

[58] Misra, I. et al. "Cross-stitch networks for multi-task learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3994–4003.

[59] Strezoski, G., Noord, N. v., and Worring, M. "Many task learning with task routing". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1375–1384.

[60] Prellberg, J. and Kramer, O. "Learned weight sharing for deep multi-task learning by natural evolution strategy and stochastic gradient descent". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[61] Kendall, A., Gal, Y., and Cipolla, R. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.

[62] Liebel, L. and Körner, M. "Auxiliary tasks in multi-task learning". In: *arXiv preprint arXiv:1805.06334* (2018).

[63] Liu, S., Johns, E., and Davison, A. J. "End-to-end multi-task learning with attention". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1871–1880.

[64] Luvizon, D. C., Picard, D., and Tabia, H. "Multi-task deep learning for real-time 3D human pose estimation and action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 43.8 (2020), pp. 2752–2764.

[65] Fan, D. et al. "Multi-task and multi-modal learning for rgb dynamic gesture recognition". In: *IEEE Sensors Journal* 21.23 (2021), pp. 27026–27036.

[66] Mehmood, T. et al. "Combining multi-task learning with transfer learning for biomedical named entity recognition". In: *Procedia Computer Science* 176 (2020), pp. 848–857.

[67] Fazekas, S. et al. "Artificial intelligence and neural networks in radiology–Basics that all radiology residents should know". In: *Imaging* 14.2 (2022), pp. 73–81.

[68] Das, S. "Spatio-temporal Attention Mechanisms for Activity Recognition". PhD thesis. Université Côte d'Azur, 2020.

[69] Liu, J., Akhtar, N., and Mian, A. "Skepxels: Spatio-temporal image representation of human skeleton joints for action recognition." In: *CVPR workshops*. 2019, pp. 10–19.

[70] Kuang, Y. et al. "Multi-modal gesture recognition with voting-based dynamic time warping". In: *International Journal of Advanced Robotic Systems* 16.6 (2019), p. 1729881419892398.

[71] Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[72] Burkart, N. and Huber, M. F. "A survey on the explainability of supervised machine learning". In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 245–317.
[73] Sun, S. et al. "A survey of optimization methods from a machine learning perspective". In: *IEEE transactions on cybernetics* 50.8 (2019), pp. 3668–3681.
[74] Alzubaidi, L. et al. "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions". In: *Journal of big Data* 8 (2021), pp. 1–74.
[75] Feng, J. et al. "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks". In: *Physical Review E* 100.3 (2019), p. 033308.
[76] Carpenter, K. A. et al. "Deep learning and virtual drug screening". In: *Future medicinal chemistry* 10.21 (2018), pp. 2557–2567.
[77] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).
[78] Shahroudy, A. et al. "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1010–1019.
[79] Pham, H.-H. et al. "Exploiting deep residual networks for human action recognition from skeletal data". In: *Computer Vision and Image Understanding* 170 (2018), pp. 51–66.
[80] Jain, M., Jégou, H., and Bouthemy, P. "Improved motion description for action classification". In: *Frontiers in ICT* 2 (2016), p. 28.
[81] Bellon, X. J. O. R. P. and Oishi, D. G. T. "Advances in Depth Image Analysis and Applications". In: (2013).
[82] Kuang, Y. et al. "Multi-modal gesture recognition with voting-based dynamic time warping". In: *International Journal of Advanced Robotic Systems* 16.6 (2019), p. 1729881419892398.
[83] Wang, W. et al. "Medical Gesture Recognition Method Based on Improved Lightweight Network". In: *Applied Sciences* 12.13 (2022), p. 6414.
[84] Sun, X. et al. "A deep multitask learning approach for air quality prediction". In: *Annals of Operations Research* 303 (2021), pp. 51–79.
[85] Baxter, J. "A Bayesian/information theoretic model of learning to learn via multiple task sampling". In: *Machine learning* 28 (1997), pp. 7–39.
[86] Yang, Y. and Hospedales, T. M. "Trace norm regularised deep multi-task learning". In: *arXiv preprint arXiv:1606.04038* (2016).
[87] Krizhevsky, A., Sutskever, I., and Hinton, G. E. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).