National Technical University of Athens

School of Electrical and Computer Engineering

Division of Signals, Control and Robotics

A Clustering Method for Zero-Shot Learning

DIPLOMA THESIS of

Petros Georgoulas Wraight

Supervisor:
Petros Maragos, Professor
NTUA

Co-Supervisor:
Ioannis Kordonis, Assistant Professor
NTUA

Co-Supervisor:
Giorgos Retsinas, Postdoctoral Researcher
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

A Clustering Method for Zero-Shot Learning

DIPLOMA THESIS of

Petros Georgoulas Wraight

Supervisor:
Petros Maragos, Professor
NTUA

Co-Supervisor:
Ioannis Kordonis, Assistant Professor
NTUA

Co-Supervisor:
Giorgos Retsinas, Postdoctoral Researcher
NTUA

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

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

Athens, July 2024

.....

Petros Georgoulas Wraight
Graduate of Electrical and Computer Engineering NTUA

Περίληψη

Σε ορισμένα σενάρια αναγνώρισης αντικειμένων, τα δεδομένα με ετικέτες μπορεί να μην καλύπτουν όλες τις κατηγορίες. Η Μάθηση Χωρίς Παραδείγματα (ΜΧΠ) αντιμετωπίζει αυτό το ζήτημα αξιοποιώντας βοηθητικές πληροφορίες που περιγράφουν κάθε κατηγορία, επιδιώκοντας την ανάπτυξη ενός ταξινομητή ικανού να αναγνωρίζει δείγματα από κατηγορίες που δεν έχουν επισημανθεί με ετικέτες. Η ημιεπαγωγική ΜΧΠ (Transductive ZSL) επεκτείνει αυτή την έννοια αναγνωρίζοντας δείγματα από άγνωστες κατηγορίες, χρησιμοποιώντας πληροφορίες τόσο από ορατές όσο και από αόρατες κατηγορίες κατά τη διάρκεια της εκπαίδευσης. Σε αυτή τη διπλωματική εργασία, προτείνουμε μια νέα προσέγγιση για την ημιεπαγωγική ΜΧΠ ενσωματώνοντας έναν από τους καλύτερους αλγόριθμους μη επιβλεπόμενης συσταδοποίησης και τροποποιώντας τον σύμφωνα με τις ανάγκες μας. Από όσο γνωρίζουμε, αυτή είναι η πρώτη προσπάθεια γεφύρωσης του χάσματος μεταξύ των προόδων στη βιβλιογραφία μη επιβλεπόμενης συσταδοποίησης και της Μάθησης Χωρίς Παραδείγματα.

Αρχικά, χρησιμοποιούμε τον αλγόριθμο συσταδοποίησης για να χωρίσουμε όλες τις εικόνες στον οπτικό χώρο σε ξεχωριστά σύνολα. Στη συνέχεια, χρησιμοποιούμε την αντιστοιχία μεταξύ ορισμένων ομάδων και γνωστών κατηγοριών, για να βρούμε μια αμφιμονοσήμαντη απεικόνιση από τον σημασιολογικό χώρο, που περιέχει πρωτότυπα για κάθε κατηγορία, στις ομάδες του οπτικού χώρου. Χρησιμοποιώντας αυτήν την απεικόνιση, προβάλλουμε τα πρωτότυπα από τον σημασιολογικό χώρο στον οπτικό χώρο και ταξινομούμε δείγματα των άγνωστων κατηγοριών με βάση την απόστασή τους από αυτά τα προβεβλημένα πρωτότυπα.

Μέσω πειραμάτων σε δύο σύνολα δεδομένων, δείχνουμε την αποτελεσματικότητα της προσέγγισής μας σε ημιεπαγωγικές ΜΧΠ εργασίες. Η μέθοδος μας επιτυγχάνει απόδοση συγκρίσιμη με άλλους κορυφαίους αλγορίθμους ΜΧΠ στο σύνολο δεδομένων AwA2, χωρίς να απαιτείται εκπαίδευση από άκρο σε άκρο ή λεπτομερής προσαρμογή της αρχιτεκτονικής ResNet101 .

Λέξεις κλειδιά: Μάθηση Χωρίς Παραδείγματα, Συσταδοποίηση, Μηχανική Μάθηση, Νευρωνικά Δίκτυα, Όραση Υπολογιστών, Ευθυγράμμιση.

Abstract

In certain object recognition scenarios, labeled data might not cover all categories. Zero-shot learning (ZSL) addresses this issue by leveraging auxiliary information that describes each category, aiming to develop a classifier capable of recognizing samples from categories lacking labeled instances. Transductive ZSL extends this concept by recognizing instances from unseen classes using information from both seen and unseen classes during training. In this thesis, we propose a novel approach for transductive ZSL by integrating a SOTA unsupervised clustering algorithm and modifying it to our needs. To the best of our knowledge, this is the first attempt to bridge the gap between the advances in Unsupervised Clustering literature and Zero-Shot Learning.

Initially, we employ the clustering algorithm to partition all images in the visual space into distinct sets. Subsequently, we establish a correspondence between some clusters and known classes to find a bijective mapping from the semantic space, which contains prototypes for each class, to the visual space clusters. Using this learned mapping, we project prototypes from the semantic space to the visual space and classify instances of the unseen classes based on their distance to these projected prototypes.

Through experiments on two benchmark datasets, we demonstrate the effectiveness of our approach in transductive ZSL tasks. Our method achieves performance on a par with other state-of-the-art ZSL algorithms on the AwA2 dataset, without requiring end-to-end training or fine-tuning of the ResNet101 backbone.

Keywords: Zero-Shot Learning, Clustering, Machine Learning, Neural Networks, Computer Vision, Alignment.

Ευχαριστίες

Καθώς ολοκληρώνεται η φοίτησή μου στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, θα ήθελα αρχικά να ευχαριστήσω θερμά τον Καθηγητή κ. Πέτρο Μαραγκό για την εμπιστοσύνη που μου έδειξε, επιτρέποντάς μου να εκπονήσω την εργασία αυτή υπό την επίβλεψή του και για το ενδιαφέρον που μου καλλιέργησε για την επιστήμη της όρασης υπολογιστών και της μηχανικής μάθησης. Θα ήθελα επίσης εκφράσω την ειλικρινή μου ευγνωμοσύνη προς τους συν-επιβλέποντες μου, τον Επίκουρο Καθηγητή κ. Ιωάννη Κορδώνη και τον Δόκτορα κ. Γιώργο Ρετσινά. Η καθοδήγησή τους υπήρξε πολύτιμη, τόσο κατά την εκπόνηση της παρούσας πτυχιακής εργασίας όσο και μέσα από τις συμβουλές τους για το μέλλον μου.

Σημαντικότεροι συνοδοιπόροι μου σε αυτό το ακαδημαϊκό ταξίδι, όπως και σε όλα μου τα βήματα, ήταν η οικογένειά μου. Ευχαριστώ τους γονείς μου και τον αδερφό μου για την αδιάκοπη στήριξή τους. Τέλος, θα ήθελα να ευχαριστήσω τη Σοφία, τον Νικόλα, το Λεωνίδα και όλους τους φίλους μου για τις όμορφες αναμνήσεις και τη συντροφιά τους.

Πέτρος Γεωργούλας Ρέιτ
Ιούλιος 2024

Contents

List of Figures

List of Tables

Chapter 1

Εκτενής Περίληψη

1.1 Εισαγωγή

Η Τεχνητή Νοημοσύνη είναι κλάδος της πληροφορικής. Ασχολείται με την ανάπτυξη συστημάτων τα οποία μπορούν να εκτελούν εργασίες που, συνήθως, εκτελούν οι άνθρωποι. Καλύπτει ένα ευρύ φάσμα εφαρμογών όπως η εκμάθηση από δεδομένα, η αναγνώριση προτύπων, η κατανόηση και επεξεργασία φυσικής γλώσσας, και η αποκωδικοποίηση οπτικών εισροών. Στον πυρήνα της, η Τεχνητή Νοημοσύνη είναι η προσπάθεια δημιουργίας αλγορίθμων και μοντέλων τα οποία χρησιμοποιούν οι μηχανές για να εκτελέσουν διεργασίες όπως η λογική, ο προγραμματισμός ή η προσαρμογή σε νέες καταστάσεις, με τρόπο παρόμοιο με αυτόν που οι άνθρωποι εκτελούν μια αντίστοιχη γνωστική εργασία. Η απαρχή της Τεχνητής Νοημοσύνης εντοπίζεται στα πρώτα χρόνια της πληροφορικής. Ως εκ τούτου έχει υποστεί πολλές αλλαγές με την πάροδο του χρόνου εξαιτίας της ανάπτυξης στην υπολογιστική ισχύ, στον όγκο και στη διαθεσιμότητα-πρόσβασιμότητα των δεδομένων, και λόγω της ανάπτυξης της θεωρητικής κατανόησης.

Η Τεχνητή Νοημοσύνη μπορεί να χωριστεί γενικά σε δύο βασικούς τύπους: στενή/αδύναμη ΤΝ και γενική/ισχυρή ΤΝ. Η αδύναμη ΤΝ σχεδιάζεται για να αντιμετωπίζει συγκεκριμένες εργασίες όπως η αναγνώριση ομιλίας και προσώπου, η μετάφραση γλώσσας ή η ερμηνεία. Αυτό περιλαμβάνει εικονικούς βοηθούς, υπηρεσίες ροής, αλγόριθμους συστάσεων και λογισμικό αναγνώρισης εικόνας για ιατρική διάγνωση. Καθώς είναι αποτελεσματική, μόνο σε συγκεκριμένα σενάρια, δεν φέρει γενική νοημοσύνη και δεν μπορεί να ενεργήσει πέρα από τις οδηγίες που δίνονται για μια συγκεκριμένη δραστηριότητα.

Από την άλλη πλευρά, η γενική ΤΝ ή ισχυρή ΤΝ επιδιώκει τον στόχο της ανάπτυξης ανθρώπινων γνωστικών ικανοτήτων. Αυτό σημαίνει ότι ένα σύστημα με γενική νοημοσύνη θα καταλάβαινε, θα μάθαινε και θα εφάρμοζε γνώση σε πολλούς τομείς, δείχνοντας δημιουργικότητα όπως οι άνθρωποι. Παρόλο που η γενική ΤΝ είναι ακόμα μια θεωρητική έννοια και ένα σημαντικό θέμα έρευνας, η επιδίωξη αυτού του οράματος επιφέρει πρόοδο στον τομέα. Επί του παρόντος, μεγάλο μέρος της έρευνας στον τομέα της ΤΝ εστιάζει στη μείωση του χάσματος μεταξύ στενής και γενικής ΤΝ, εξερευνώντας νέα μοντέλα, αρχιτεκτονικές και εναλλακτικούς τρόπους εκπαίδευσης, που δείχνουν υποσχόμενοι στην προσέγγιση πιο γενικών μορφών νοημοσύνης.

Το πρώτο αυτό κεφάλαιο εξετάζει τι είναι η Τεχνητή Νοημοσύνη (ΤΝ) και γιατί είναι σημαντική σήμερα. Θα εξηγήσουμε με απλούς όρους τι είναι η μηχανική και βαθιά μάθηση, δύο από τους κύριους κλάδους της ΤΝ που σχετίζονται άμεσα με αυτή την εργασία. Στη συνέχεια, θα εξηγήσουμε τη Μάθηση Χωρίς Παραδείγματα, που είναι το κύριο θέμα αυτής της διπλωματικής. Αυτό το κεφάλαιο θα θέσει το πλαίσιο για μια πολύ πιο λεπτομερή και εστιασμένη συζήτηση στο επόμενο κεφάλαιο.

1.1.1 Μηχανική Μάθηση

Η μηχανική μάθηση (Machine Learning - ML) είναι ένα σημαντικό υποσύνολο της τεχνητής νοημοσύνης (Artificial Intelligence - AI) που εστιάζει στη διδασκαλία των μηχανών ώστε να επεξεργάζονται και να ερμηνεύουν δεδομένα πιο αποτελεσματικά [19]. Σε αντίθεση με τον παραδοσιακό προγραμματισμό, όπου οι υπολογιστές ακολουθούν ρητές οδηγίες, η μηχανική μάθηση τους επιτρέπει να μαθαίνουν από δεδομένα και να λαμβάνουν αποφάσεις με βάση τα μοτίβα που ανακαλύπτουν. Στη μηχανική μάθηση, ένα πρόγραμμα βελτιώνει την απόδοσή του σε εργασίες με την πάροδο του χρόνου, αποκτώντας εμπειρία από τα παραδείγματα στα οποία έχει εκτεθεί. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη όταν αντιμετωπίζονται μεγάλα και πολύπλοκα σύνολα δεδομένων, όπου η χειροκίνητη ανάλυση θα ήταν δύσκολη. Καθώς η διαθεσιμότητα εκτεταμένων συνόλων δεδομένων συνεχίζει να αυξάνεται, η ζήτηση για μηχανική μάθηση αυξάνεται και πολλές επιχειρήσεις αξιοποιούν την επιστήμη αυτή, για να εξάγουν πολύτιμες πληροφορίες από τα δεδομένα τους. Σήμερα, η μηχανική μάθηση είναι κάτι με το οποίο οι άνθρωποι αλληλεπιδρούμε καθημερινά, -συχνά χωρίς να το συνειδητοποιούμε- και επηρεαζόμαστε απ' αυτή στις επιλογές μας (από το ποια προϊόντα καταναλώνουμε έως και ποιες ταινίες παρακολουθούμε).

1.1.2 Βαθιά Μάθηση

Ιστορικά, οι συμβατικές μέθοδοι μηχανικής μάθησης ήταν περιορισμένες στην ικανότητά τους να χειρίζονται ακατέργαστα δεδομένα με άμεσο τρόπο. Η κατασκευή ενός συστήματος αναγνώρισης προτύπων ή μηχανικής μάθησης, απαιτούσε εξειδίκευση στον τομέα για τον σχεδιασμό προγραμμάτων ικανών να εξάγουν χαρακτηριστικά και να μετατρέπουν τα ακατέργαστα δεδομένα (όπως οι τιμές των pixel μιας εικόνας) σε ενα διάνυσμα χαρακτηριστικών κατάλληλο για έναν αλγόριθμο εκμάθησης (για παράδειγμα έναν ταξινομητή).

Η βαθιά μάθηση είναι μια εξειδικευμένη περιοχή εντός της μηχανικής μάθησης που δίνει έμφαση στη χρήση νευρωνικών δικτύων με πολλαπλά επίπεδα, γνωστά ως βαθιά νευρωνικά δίκτυα. Ένα σημαντικό πλεονέκτημα της βαθιάς μάθησης είναι η ικανότητά της για αυτόματη εκμάθηση αναπαραστάσεων μέσω των κρυφών της επιπέδων [9]. Τα βαθιά νευρωνικά δίκτυα κατασκευάζουν ιεραρχικές αναπαραστάσεις των δεδομένων μέσω πολλαπλών επιπέδων, με κάθε επίπεδο να συλλαμβάνει προοδευτικά πιο αφηρημένα χαρακτηριστικά [9]. Αυτή η διαδικασία αυτόματης εξαγωγής χαρακτηριστικών μειώνει την ανάγκη για "χειροποίητα" χαρακτηριστικά και εξειδικευμένη γνώση του τομέα, κάτι το οποίο καθιστά την εκπαίδευση αυτών των μοντέλων πιο άμεση. Σε ένα βαθύ νευρωνικό δίκτυο, τα κρυφά επίπεδα λειτουργούν ως μια σειρά φίλτρων, βελτιώνοντας τις αναπαραστάσεις των δεδομένων σταδιακά. Για παράδειγμα, τα αρχικά επίπεδα μπορεί να αναγνωρίζουν βασικά μοτίβα όπως οι ακμές σε μια εικόνα, ενώ τα βαθύτερα επίπεδα μπορούν να ανιχνεύουν πιο σύνθετες δομές όπως τα μέρη αντικειμένων.

Αυτή η μεθοδολογία είναι ιδιαίτερα αποτελεσματική για τη διαχείριση μεγάλων, σύνθετων συνόλων δεδομένων, γνωστά ως "big data", τα οποία γίνονται όλο και πιο διαθέσιμα τα τελευταία χρόνια. Η εμφάνιση των big data έχει ενισχύσει σημαντικά τις προοπτικές της βαθιάς μάθησης, καθώς οι αλγόριθμοι αυτοί αριστεύουν στην αποκάλυψη κρυφών προτύπων σε εκτενή σύνολα δεδομένων. Ο συνδυασμός άφθονων δεδομένων και αύξησης της υπολογιστικής ισχύος έχει επιτρέψει στα μοντέλα βαθιάς μάθησης να επιτυγχάνουν ανώτερες επιδόσεις σε σύγκριση με τους συμβατικούς αλγόριθμους μηχανικής μάθησης, σε εργασίες όπως η αναγνώριση εικόνας [20], [21], η κατανόηση φυσικής γλώσσας [21], η αναγνώριση ομιλίας [22] και πολλές άλλες [23].

1.1.3 Μάθηση Χωρίς Παραδείγματα

Οι Επιβλεπόμενες Μέθοδοι Ταξινόμησης (ΕΜΤ) στοχεύουν στην κατηγοριοποίηση των δεδομένων σε διακριτά σύνολα βάσει προκαθορισμένων κατηγοριών (ή κλάσεων), και προϋποθέτουν επαρκή δεδομένα εκπαίδευσης για κάθε κατηγορία. Ο ταξινομητής που προκύπτει περιορίζεται στην ταξινόμηση παραδειγμάτων εντός των κατηγοριών που καλύπτονται από το σύνολο εκπαίδευσης. Ωστόσο, στην πραγματικότητα μπορεί να εμφανιστούν περιπτώσεις όπου ορισμένες κλάσεις δεν εκπροσωπούνται καθόλου στο σύνολο εκπαίδευσης.

Σε τέτοιες περιπτώσεις, η εφαρμογή επιβλεπόμενων μεθόδων εκμάθησης είναι αδύνατο να εφαρμοστεί. Για να λυθεί αυτό το πρόβλημα, προτείνεται η Μάθηση Χωρίς Παραδείγματα (ΜΧΠ).

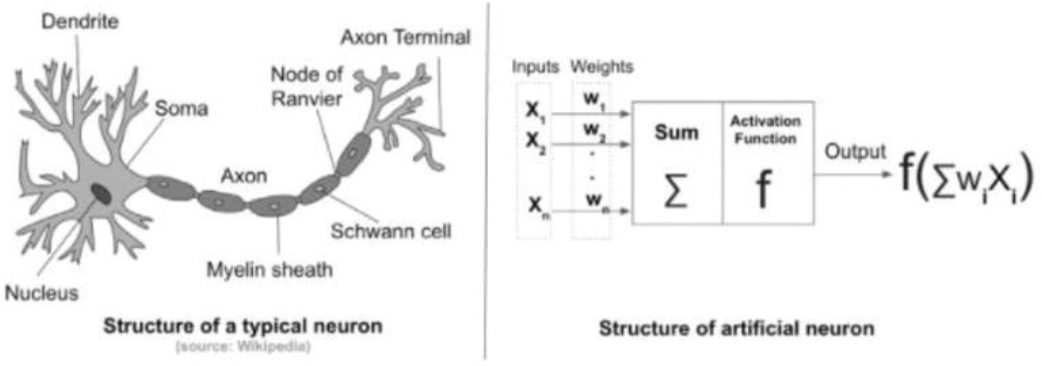
Για να κατανοήσουμε τη μάθηση χωρίς παραδείγματα, ας εξετάσουμε το παράδειγμα ενός ατόμου που δεν έχει δει ποτέ μια ζέβρα. Αν κάποιος του πει ότι η ζέβρα μοιάζει με άλογο αλλά με μαύρες και άσπρες ρίγες, μπορεί να χρησιμοποιήσει αυτήν την περιγραφή για να την αναγνωρίσει, βασισμένος στην προγενέστερη γνώση του για το τι είναι άλογο και τι είναι ρίγες. Οι άνθρωποι μπορούν να εξάγουν και να χρησιμοποιούν πληροφορίες υψηλού επιπέδου για να αναγνωρίζουν νέα αντικείμενα, συνδυάζοντας γνωστά χαρακτηριστικά. Ομοίως, ο στόχος της μάθησης χωρίς παραδείγματα είναι να επιτρέψει σε ένα μοντέλο μηχανικής μάθησης να αναγνωρίζει νέες κατηγορίες, χρησιμοποιώντας περιγραφές ή χαρακτηριστικά που οι νέες κατηγορίες μοιράζονται με άλλες ήδη γνωστές κατηγορίες. Στο προηγούμενο παράδειγμα, αν ένα μοντέλο γνωρίζει για τα άλογα και τις ρίγες, τότε θέλουμε να συνδυάσει αυτή τη γνώση για να αναγνωρίσει μια ζέβρα.

Επομένως, ο στόχος της μάθησης χωρίς παραδείγματα είναι να εξάγει γνώση από ήδη γνωστές κλάσεις και να την εφαρμόσει σε άγνωστες. Χρησιμοποιώντας περιγραφικές πληροφορίες και χαρακτηριστικά, τα μοντέλα ΜΧΠ μπορούν να κάνουν υποθέσεις για νέες κατηγορίες βασισμένες στις ομοιότητές τους με γνωστές. Αυτή η προσέγγιση επιτρέπει στα συστήματα μηχανικής μάθησης να είναι πιο ευέλικτα, και να μπορούν να εφαρμοστούν σε ένα ευρύτερο φάσμα σεναρίων χωρίς να απαιτούν εξαντλητικά σύνολα δεδομένων με ετικέτες για κάθε πιθανή κατηγορία.

1.2 Υπόβαθρο

1.2.1 Νευρωνικά Δίκτυα

Ο Νευρώνας



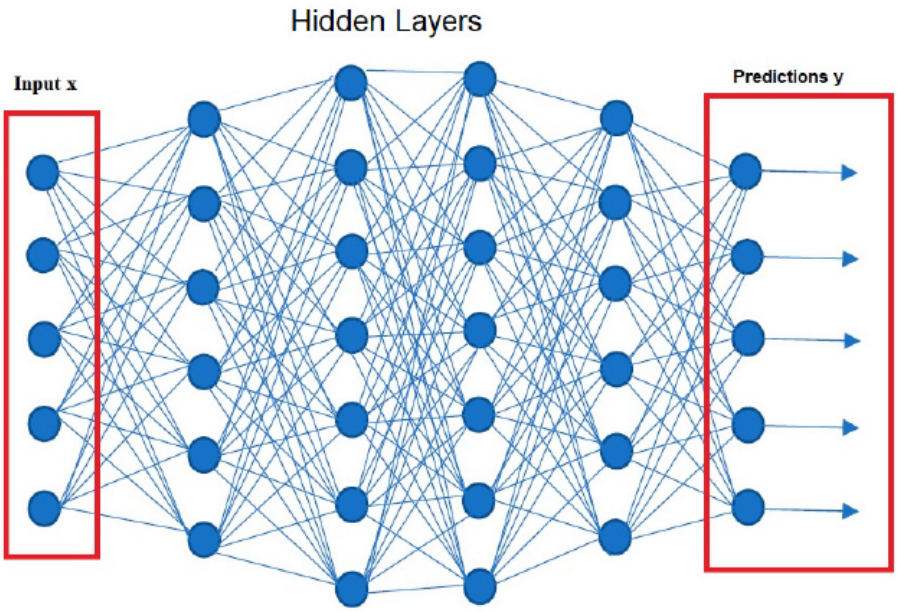
Σχήμα 1.1: Βιολογική Νευρώνες και Τεχνητή νουμοσύνη (Πηγή: [1])

Τα νευρωνικά δίκτυα, εμπνευσμένα από την αρχιτεκτονική και τη λειτουργία του ανθρώπινου εγκεφάλου, μιμούνται τη συμπεριφορά των νευρώνων. Σε αυτά τα δίκτυα, τα στρώματα τεχνητών νευρώνων συνεργάζονται για να επεξεργαστούν εισερχόμενες πληροφορίες (είσοδοι) και να παράγουν εξερχόμενες πληροφορίες (έξοδοι). Κάθε νευρώνας λαμβάνει ένα σύνολο εισόδων, οι οποίες πολλαπλασιάζονται με ρυθμιζόμενα βάρη που υπαγορεύουν την επίδραση κάθε εισόδου στο τελικό αποτέλεσμα. Επιπλέον, προστίθεται μια ρυθμιζόμενη πόλωση (bias) σε αυτό το σταθμισμένο άθροισμα. Το τελικό αποτέλεσμα περνά στη συνέχεια μέσω μιας συνάρτησης ενεργοποίησης, η οποία το μετατρέπει σε μια έξοδο, όπως μια πιθανότητα ή μια δυαδική απόφαση. Αυτός ο σύνθετος μηχανισμός επιτρέπει στα νευρωνικά δίκτυα να διακρίνουν περίπλοκα πρότυπα και να κάνουν προβλέψεις με βάση τα δεδομένα που αναλύουν. Συγκεκριμένα, αν $x \in \mathbb{R}^d$ είναι η είσοδος στον νευρώνα, τότε η έξοδος είναι

$$y = f\left(\sum_i w_i x_i + b\right) = f(w^T x + b) \tag{1.1}$$

όπου f ονομάζεται *συνάρτηση ενεργοποίησης*, $w = (w_1, \ldots, w_d) \in \mathbb{R}^d$ είναι τα βάρη που προς μάθηση και $b \in \mathbb{R}$ είναι η πόλωση προς μάθηση.

Πολυστρωματικοί Perceptrons



Σχήμα 1.2: Αναπαράσταση ενός Πολυστρωματικού Perceptron (Πηγή: [2])

Ένα Πολυστρωματικό Δίκτυο Perceptron (Multilayer Perceptron - MLP) είναι ένα τεχνητό νευρωνικό δίκτυο στο οποίο πολλαπλά επίπεδα νευρώνων στοιβάζονται το ένα μετά το άλλο, όπως φαίνεται στο Σχήμα 1.2. Οι νευρώνες σε ένα MLP οργανώνονται σε επίπεδα: ένα επίπεδο εισόδου, ένα ή περισσότερα επίπεδα που ονομάζονται κρυφά επίπεδα, και ένα επίπεδο από το οποίο λαμβάνουμε την έξοδο (το στρώμα εξόδου). Κάθε νευρώνας σε ένα επίπεδο συνδέεται με κάθε νευρώνα στο επόμενο επίπεδο, σχηματίζοντας ένα πλήρως συνδεδεμένο δίκτυο. Οι πληροφορίες που παρέχονται στο MLP τροφοδοτούνται στο επίπεδο εισόδου, το οποίο επεξεργάζεται περαιτέρω την πληροφορία

περνώντας τη μέσα από τα κρυφά επίπεδα. Κάθε ένα από τα κρυφά επίπεδα αλλάζει την αναπαράσταση της πληροφορίας χρησιμοποιώντας το μαθηματικό φορμαλισμό των νευρώνων και τελικά παράγει την έξοδο χρησιμοποιώντας το επίπεδο εξόδου.

Για ένα δεδομένο επίπεδο, η έξοδος y υπολογίζεται χρησιμοποιώντας την εξίσωση:

$$y = f(Wx + b) \tag{1.2}$$

όπου W είναι ο πίνακας των βαρών που συνδέουν τους νευρώνες του τρέχοντος επιπέδου, με τους νευρώνες του προηγούμενου επιπέδου, x είναι το διάνυσμα εισόδου, b είναι ένα διάνυσμα πολώσεων (bias), και f είναι η συνάρτηση ενεργοποίησης που εφαρμόζεται σε κάθε στοιχείο του εκάστοτε διανύσματος. Αυτή η διαδικασία επαναλαμβάνεται για κάθε επίπεδο, με την έξοδο ενός επιπέδου να λειτουργεί ως είσοδος για το επόμενο, επιτρέποντας έτσι στο δίκτυο να μαθαίνει πολύπλοκες, μη γραμμικές συναρτήσεις από την είσοδο στην έξοδο.

Βελτιστοποίηση Για Νευρωνικά Δίκτυα

Στην επιβλεπόμενη μηχανική μάθηση, ο στόχος είναι να βρούμε τις παραμέτρους θ που ελαχιστοποιούν το εμπειρικό σφάλμα. Το εμπειρικό σφάλμα ορίζεται ως ο μέσος όρος της συνάρτησης απώλειας L για όλα τα παραδείγματα εκπαίδευσης. Μαθηματικά, αυτό μπορεί να εκφραστεί ως:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i, \theta), y_i) \tag{1.3}$$

Εδώ, f είναι μια συνάρτηση που κάνει προβλέψεις, με παραμέτρους θ, η οποία λαμβάνει ως είσοδο το x_i και προβλέπει μια έξοδο $\hat{y}_i$. Η συνάρτηση απώλειας L μετρά τη διαφορά μεταξύ της προβλεπόμενης εξόδου $\hat{y}_i$ και της πραγματικής ετικέτας y_i.

Στο πλαίσιο των νευρωνικών δικτύων, η συνάρτηση f αντιπροσωπεύει ολόκληρο το δίκτυο, περιέχοντας πολλαπλά επίπεδα νευρώνων με διάφορες συναρτήσεις ενεργοποίησης, ενώ παράμετροι θ περιλαμβάνουν τα βάρη και τις πολώσεις όλων των νευρώνων στο δίκτυο. Ο στόχος της εκπαίδευσης ενός νευρωνικού δικτύου είναι η προσαρμογή αυτών των παραμέτρων ώστε το εμπειρικό σφάλμα $J(\theta)$ να ελαχιστοποιηθεί, κάτι το οποίο ελπίζουμε να οδηγήσει σε ακριβείς προβλέψεις στα δεδομένα δοκιμών (test data).

Συναρτήσεις Απώλειας Οι συναρτήσεις απώλειας παίζουν κρίσιμο ρόλο στην εκπαίδευση μοντέλων, καθώς μετρούν τη διαφορά μεταξύ της προβλεπόμενης εξόδου και του πραγματικού στόχου. Διάφοροι τύποι συναρτήσεων απώλειας χρησιμοποιούνται, ανάλογα με το πρόβλημα που καλούμαστε να λύσουμε.

Για προβλήματα παλινδρόμησης, όπου ο στόχος είναι η πρόβλεψη συνεχών τιμών, μια συνηθισμένη συνάρτηση απώλειας είναι η Μέση Τετραγωνική Απόκλιση (Mean Squared Error - MSE) και ορίζεται ως:

$$L_{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1.4}$$

Εδώ, η μεταβλητή y_i αντιπροσωπεύει την πραγματική τιμή, $\hat{y}_i$ είναι η προβλεπόμενη τιμή και n είναι ο αριθμός των δειγμάτων.

Για προβλήματα δυαδικής ταξινόμησης, όπου ο στόχος είναι η πρόβλεψη μιας από τις δύο πιθανές κατηγορίες, η πιο τυπική συνάρτηση απώλειας είναι η Δυαδική Διασταυρούμενη Εντροπία (Binary

Cross-Entropy - BCE) και ορίζεται ως:

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \tag{1.5}$$

Στην παραπάνω εξίσωση, η μεταβλητή y_i αντιπροσωπεύει την πραγματική ετικέτα (0 ή 1) και $\hat{y}_i$ είναι η προβλεπόμενη πιθανότητα, ότι το δείγμα ανήκει στην κατηγορία 1.

Τέλος, για προβλήματα ταξινόμησης πολλών κατηγοριών, όπου υπάρχουν περισσότερες από δύο κατηγορίες, η Διασταυρούμενη Εντροπία (Cross-Entropy) γενικεύεται για να χειριστεί πολλές κλάσεις. Η Διασταυρούμενη Εντροπία για ταξινόμηση πολλαπλών κατηγοριών δίνεται από τον τύπο:

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \tag{1.6}$$

Στην παραπάνω εξίσωση, C είναι ο αριθμός των κατηγοριών, $y_{i,c}$ είναι ο δυαδικός δείκτης (0 ή 1) εάν η ετικέτα της κατηγορίας c είναι η σωστή ταξινόμηση για την παρατήρηση i, και $\hat{y}_{i,c}$ είναι η προβλεπόμενη πιθανότητα ότι η παρατήρηση i ανήκει στην κατηγορία c.

Κανονικοποίηση

Η κανονικοποίηση είναι μια τεχνική που χρησιμοποιείται για να αποτραπεί η υπερπροσαρμογή στα μοντέλα μηχανικής μάθησης. Η υπερπροσαρμογή συμβαίνει όταν ένα μοντέλο μαθαίνει όχι μόνο τα μοτίβα στα δεδομένα εκπαίδευσης, αλλά και τον θόρυβο. Αυτό έχει ως αποτέλεσμα εξαιρετική απόδοση στο σύνολο των δεδομένων εκπαίδευσης αλλά κακή γενίκευση σε άγνωστα-νέα δεδομένα. Η κανονικοποίηση αντιμετωπίζει αυτό το πρόβλημα προσθέτοντας μια ποινή όσον αφορά στην πολυπλοκότητα του μοντέλου, αποθαρρύνοντας το από το να προσαρμοστεί στον θόρυβο. Με τον περιορισμό των παραμέτρων του μοντέλου ή την προσθήκη ενός όρου ποινής στη συνάρτηση απώλειας, η κανονικοποίηση βοηθά στην εκμάθηση απλούστερων μοντέλων, τα οποία γενικεύονται καλύτερα σε νέα δεδομένα [24].

- **L2-κανονικοποίηση** Η L2-κανονικοποίηση λειτουργεί προσθέτοντας έναν όρο ποινής στη συνάρτηση απώλειας, ο οποίος είναι ανάλογος με το άθροισμα των τετραγώνων των παραμέτρων του μοντέλου. Αυτό σημαίνει ότι για κάθε βάρος w στο δίκτυο, ο όρος λw^2 προστίθεται στη συνάρτηση απώλειας, όπου λ είναι η ισχύς της κανονικοποίησης.

- **L1-κανονικοποίηση** Η L1-κανονικοποίηση είναι μια άλλη τεχνική που χρησιμοποιείται για την αποτροπή της υπερπροσαρμογής στη μηχανική μάθηση. Προσθέτει έναν όρο ποινής που είναι ανάλογος με το άθροισμα των απόλυτων τιμών των παραμέτρων του μοντέλου. Έτσι για κάθε βάρος w, ο όρος $\lambda|w|$ προστίθεται στη συνάρτηση απώλειας. Αυτή η τεχνική ενθαρρύνει τη σποραδικότητα (sparsity) στις παραμέτρους του μοντέλου, καθώς τείνει να οδηγεί κάποια από τα βάρη στο μηδέν, πραγματοποιώντας ουσιαστικά επιλογή χαρακτηριστικών (feature selection).

- **Πρόωρη Διακοπή (Early-Stopping)** Η πρόωρη διακοπή είναι μια τεχνική κανονικοποίησης που χρησιμοποιείται για να αποτραπεί η υπερπροσαρμογή στα νευρωνικά δίκτυα παρακολουθώντας την απόδοση του μοντέλου σε ένα σύνολο επικύρωσης και διακόπτοντας την εκπαίδευση μόλις η απόδοση αρχίσει να επιδεινώνεται. Αυτή η μέθοδος αξιοποιεί την παρατήρηση, ότι ενώ το σφάλμα εκπαίδευσης τυπικά μειώνεται με την πάροδο του χρόνου, το σφάλμα επικύρωσης συχνά μειώνεται αρχικά, αλλά τελικά αρχίζει να αυξάνεται καθώς το μοντέλο αρχίζει να προσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης.

Ο Κανόνας της Αλυσίδας

Ας θεωρήσουμε ότι x είναι ένας πραγματικός αριθμός, και ότι οι f και g είναι συναρτήσεις από τον $\mathbb{R}$ στον $\mathbb{R}$. Ας υποθέσουμε ότι $y = g(x)$ και $z = f(g(x)) = f(y)$. Τότε ο κανόνας της αλυσίδας δηλώνει ότι

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} \tag{1.7}$$

Μπορούμε να γενικεύσουμε αυτό το αποτέλεσμα στην περίπτωση όπου κάποια συνάρτηση παίρνει διανύσματα ως είσοδο. Ας υποθέσουμε ότι $g : \mathbb{R}^m \to \mathbb{R}^n$ και $f : \mathbb{R}^n \to \mathbb{R}$. Τότε, αν $z = g(y)$ και $y = f(x)$, προκύπτει ότι

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i} \tag{1.8}$$

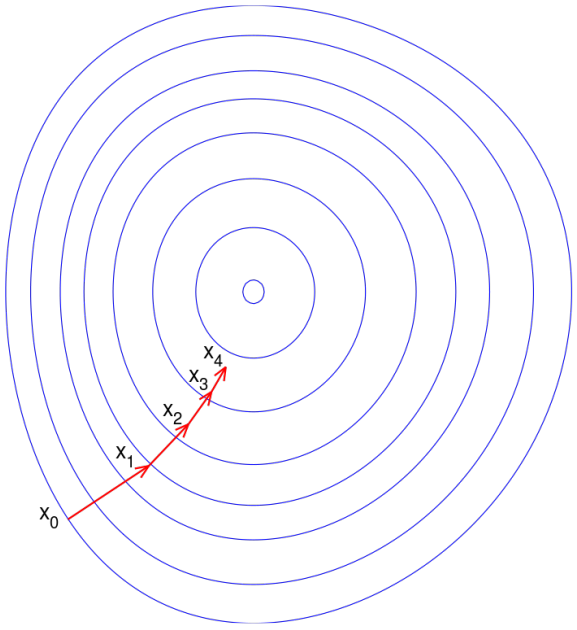
Σε διανυσματική μορφή αυτό μπορεί επίσης να γραφτεί ως:

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^T \cdot \nabla_y z \tag{1.9}$$

όπου $\frac{\partial y}{\partial x}$ είναι ο Ιακωβιανός πίνακας της συνάρτησης g.

Κατάβαση Κλίσης (Gradient Descent) Ας υποθέσουμε ότι έχουμε μια συνάρτηση $f : \mathbb{R}^n \to \mathbb{R}$. Για να βρούμε την κατευθυντική παράγωγο της f στην κατεύθυνση u στο σημείο x, εξετάζουμε τη συνάρτηση

$$g(t) = f(x + tu)$$



Σχήμα 1.3: Οπτικοποίηση της μεθόδου κατάβασης κλίσης. (Πηγή: [3])

Η παράγωγος της $g(t)$ δίνεται από τη σχέση $g'(t) = u^T \nabla_x f(x + tu)$. Υπολογίζοντας την στο $t = 0$, έχουμε $g'(0) = u^T \nabla_x f(x)$. Για να βρούμε την κατεύθυνση u στην οποία η f μειώνεται ταχύτερα, χρησιμοποιούμε το γεγονός ότι $u^T \nabla_x f(x) = ||u||_2 ||\nabla_x f(x)||_2 \cos\theta$ όπου θ είναι η γωνία μεταξύ των δύο διανυσμάτων. Είναι προφανές ότι η παραπάνω έκφραση ελαχιστοποιείται όταν το u δείχνει στην αντίθετη κατεύθυνση από αυτή της κλίσης. Επομένως, θα πρέπει να πάρουμε $u = -\nabla_x f(x)$.

Για να καταλάβουμε πώς αυτό εφαρμόζεται στην πράξη, ας υποθέσουμε ότι έχουμε μια διαφορίσιμη συνάρτηση f. Τότε, χρησιμοποιώντας την σειρά Taylor, έχουμε ότι:

$$f(x(t) + \Delta x) \approx f(x(t)) + \nabla f(x)|_{x=x(t)} \Delta x \tag{1.10}$$

Εξετάζοντας την κατεύθυνση όπου η f μειώνεται ταχύτερα, όπως συζητήθηκε νωρίτερα, θέτουμε $\Delta x = -a \nabla f(x) | x = x(t)$, $a > 0$. Αυτό μας οδηγεί στον κανόνα ενημέρωσης:

$$x(t+1) = x(t) - a \nabla f(x)|_{x=x(t)} \tag{1.11}$$

και

$$f(x(t+1)) = f(x(t) + \Delta x) \approx f(x(t)) - a||\nabla f(x)|_{x=x(t)}||_2^2 \quad \leq f(x(t)) \tag{1.12}$$

Στην πράξη, η επιλογή του κατάλληλου ρυθμού μάθησης a είναι κρίσιμη για την απόδοση του αλγορίθμου κατάβασης κλίσης. Αν ο ρυθμός μάθησης είναι πολύ μικρός, ο αλγόριθμος μπορεί να συγκλίνει με πολύ αργό τρόπο, ενώ αν είναι πολύ μεγάλος, ο αλγόριθμος μπορεί συνεχώς να υπερβαίνει το ελάχιστο και να αποτύχει να συγκλίνει.

Βελτιστοποίηση Βαρών Νευρωνικού Δικτύου

Χρησιμοποιώντας τις προαναφερθείσες βασικές έννοιες, θα εξηγήσουμε πώς λειτουργεί η βελτιστοποίηση στα νευρωνικά δίκτυα, παραθέτοντας όλα τα βήματα με λεπτομέρεια. Αρχικά θυμίζουμε ότι ο στόχος είναι συνήθως να ελαχιστοποιήσουμε το εμπειρικό σφάλμα και στη συνέχεια προστίθεται ένας όρος κανονικοποίησης για να αποφευχθεί η υπερπροσαρμογή:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i; \theta), y_i) + \lambda R(\theta) \tag{1.13}$$

Για να ελαχιστοποιήσουμε το $J(\theta)$ αντί να χρησιμοποιούμε τον αλγόριθμο κατάβασης κλίσης, συνήθως χρησιμοποιούμε μια παραλλαγή του, που ονομάζεται στοχαστική κατάβαση κλίσης - ΣΚΚ (Stochastic Gradient Descent - SGD). Αν εφαρμόζαμε τον αλγόριθμο κατάβασης κλίσης στην παραπάνω εξίσωση, τότε θα έπρεπε να υπολογίσουμε την παράγωγο με βάση όλα τα δείγματα $\{x_i\}_{i=1}^n$, το οποίο σε μεγάλα σύνολα δεδομένων γίνεται υπολογιστικά ανέφικτο. Ο ΣΚΚ εξετάζει μόνο μια παρτίδα δειγμάτων σε κάθε του βήμα, το οποίο προσφέρει ευελιξία σε τέτοιες καταστάσεις. Ένα συχνό πρόβλημα που σχετίζεται με τον ΣΚΚ είναι ότι οι ενημερώσεις του μπορεί να επιφέρουν θόρυβο, οπότε ο αλγόριθμος ταλαντώνεται γύρω από το ελάχιστο. Για να λυθεί αυτό το πρόβλημα, οι μέθοδοι βελτιστοποίησης όπως ο Adam [25] χρησιμοποιούν ορμή (momentum) και προσαρμοστικό ρυθμό μάθησης.

Η οπισθοδιάδοση (backpropagation) είναι ο αλγόριθμος που χρησιμοποιείται για να υπολογίσει την παράγωγο για κάθε παράμετρο στα νευρωνικά δίκτυα, επιτρέποντας την εφαρμογή μεθόδων βελτιστοποίησης, όπως ο ΣΚΚ. Η οπισθοδιάδοσης στηρίζεται πλήρως στον κανόνα της αλυσίδας, ο οποίος επιτρέπει στις παραγώγους να μεταδίδονται προς τα πίσω, μέσω του δικτύου.

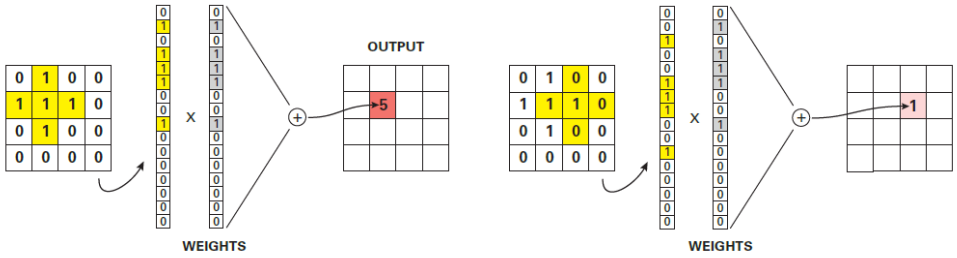
1. Προώθηση (Forward Pass): Για να εφαρμόσουμε την Εξίσωση 1.11 πρέπει να αξιολογήσουμε το $J(\theta)$ στο θ_t. Επομένως, υπολογίζουμε την έξοδο του δικτύου περνώντας τα δεδομένα εισόδου μέσα από κάθε επίπεδο.

2. Υπολογισμός Απώλειας: Υπολογίζουμε την απώλεια μεταξύ της εξόδου του δικτύου και των πραγματικών ετικετών χρησιμοποιώντας κάποια συνάρτηση κόστους (π.χ 1.4, 1.5).

3. Οπισθοδιάδοση (Backward Pass): Πρώτα υπολογίζουμε την παράγωγο της απώλειας σε σχέση με την έξοδο του δικτύου. Στη συνέχεια, χρησιμοποιούμε τον κανόνα της αλυσίδας για να μεταδώσουμε αυτούς τις παραγώγους προς τα πίσω μέσα από κάθε επίπεδο, υπολογίζοντας την παράγωγο του σφάλματος, σε σχέση με κάθε παράμετρο.

Συνελικτικά Νευρωνικά Δίκτυα

Η χρήση Πολυστρωματικών Perceptron's (MLP) για επεξεργασία εικόνας παρουσιάζει σημαντικές προκλήσεις [4]. Αρχικά, ας υπολογίσουμε τη διασταση της εισόδου: μια εικόνα με ύψος H, πλάτος W και C χρωματικά κανάλια θα απαιτούσε το πρώτο κρυφό επίπεδο να έχει διάσταση $H \times W \times C \times D$, όπου D είναι ο αριθμός των νευρώνων στο πρώτο επίπεδο. Επιπλέον, τα MLP's δεν είναι αναλλοίωτα σε μεταφορές (translation invariance) [4], [9]. Αυτό το χαρακτηριστικό μπορεί να οπτικοποιηθεί στο Σχήμα 1.4. Για να αντιμετωπιστούν αυτά τα προβλήματα, προτείνονται τα Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ). Η βασική τους ιδέα είναι ότι διαιρούν μια εικόνα σε επικαλυπτόμενες περιοχές και στη συνέχεια τις συγκρίνουν με μικρούς πίνακες βαρών, οι οποίοι συνήθως ονομάζονται *φίλτρα*. Αυτά τα φίλτρα μπορούν να θεωρηθούν ως ανιχνευτές, που αναγνωρίζουν σε ποιο βαθμό ένα συγκεκριμένο χαρακτηριστικό βρίσκεται σε μια εικόνα.



Σχήμα 1.4: Τα MLP's δεν είναι μεταφορικά αμετάβλητα (Πηγή: [4])

Συνελίξη Η διακριτή δισδιάστατη συνελίξη περιλαμβάνει δύο πίνακες και ορίζεται από τον τύπο:

$$S(i,j) = (I \circledast K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{1.14}$$

Αυτή η εξίσωση σημαίνει ότι παίρνουμε τον πίνακα της εικόνας I και το ανεστραμμένο φίλτρο K, και για κάθε θέση (i,j) στον προκύπτοντα πίνακα S, υπολογίζουμε το άθροισμα των γινομένων των επικαλυπτόμενων στοιχείων από την I και το K. Διαισθητικά, είναι σαν να μετακινούμε τον ανεστραμμένο φίλτρο κατά μήκος της εικόνας, υπολογίζοντας το σταθμισμένο άθροισμα των στοιχείων της εικόνας κάτω από το φίλτρο σε κάθε θέση.

Συνελικτικό Νευρωνικό Δίκτυο ως Εξαγωγέας Αναπαραστάσεων

Τα Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ) έχουν φέρει επανάσταση στον τομέα της υπολογιστικής όρασης, επιδεικνύοντας εξαιρετική απόδοση σε διάφορες σενάρια αναγνώρισης εικόνων [26], [27], [28], [29]. Όταν εκπαιδεύονται σε μεγάλα σύνολα δεδομένων, όπως το ImageNet [30], το οποίο περιέχει εκατομμύρια εικόνες με ετικέτες, τα ΣΝΔ έχουν δείξει την ικανότητά να μαθαίνουν πολύπλοκα οπτικά πρότυπα. Ένα χαρακτηριστικό παράδειγμα αυτού είναι η αρχιτεκτονική ResNet (Residual Network) [31], η οποία έχει επιτύχει κορυφαία αποτελέσματα σε διαγωνισμούς ταξινόμησης εικόνων.

Όπως συζητήθηκε προηγουμένως, ένα από τα κύρια πλεονεκτήματα των δικτύων αυτών, έγκειται στην ιεραρχική εκμάθηση χαρακτηριστικών. Τα αρχικά επίπεδα ενός ΣΝΔ συνήθως μαθαίνουν να ανιχνεύουν απλά χαρακτηριστικά όπως ακμές, υφές και βασικά σχήματα. Καθώς προχωρούμε βαθύτερα στο δίκτυο, τα επίπεδα αρχίζουν να συλλαμβάνουν πιο πολύπλοκα χαρακτηριστικά, όπως μέρη αντικειμένων [4]. Αυτός ο ιεραρχικός τρόπος μάθησης, επιτρέπει στα ΣΝΔ να δημιουργούν πλούσιες και με υψηλό βαθμό αφαίρεσης αναπαραστάσεις των εικόνων εισόδου.

Στην πράξη, τα ΣΝΔ που εκπαιδεύονται σε μεγάλης κλίμακας σύνολα δεδομένων συχνά χρησιμοποιούνται ως εξαγωγείς χαρακτηριστικών. Συγκεκριμένα, τα συνελικτικά επίπεδα (συνήθως τα τελευταία) αξιοποιούνται για την εξαγωγή χαρακτηριστικών υψηλού επιπέδου από εικόνες. Αυτά τα χαρακτηριστικά μπορούν στη συνέχεια να τροφοδοτηθούν σε άλλα μοντέλα μηχανικής μάθησης ή να προσαρμοστούν για συγκεκριμένες εργασίες. Αυτή η διαδικασία είναι γνωστή ως μεταφορά μάθησης (transfer learning) [32], όπου ένα προεκπαιδευμένο ΣΝΔ χρησιμοποιείται για την εξαγωγή σημαντικών χαρακτηριστικών από νέα δεδομένα, μειώνοντας σημαντικά τον χρόνο εκπαίδευσης και βελτιώνοντας την απόδοση σε εργασίες με περιορισμένα δεδομένα.

1.2.2 Μάθηση Χωρίς Παραδείγματα

Στη μάθηση χωρίς παραδείγματα - ΜΧΠ (zero-shot learning), στο χώρο των χαρακτηριστικών υπάρχουν δεδομένα με ετικέτα (επισημασμένα δείγματα) που παραπέμπουν σε γνωστές στην ΜΧΠ κατηγορίες οι οποίες ονομάζονται *Ορατές Κατηγορίες*. Επιπλέον, υπάρχουν μη επισημασμένα δείγματα στον ίδιο χώρο χαρακτηριστικών, τα οποία σχετίζονται με ένα ξεχωριστό σύνολο κατηγοριών γνωστές ως *αόρατες* κατηγορίες. Ο χώρος χαρακτηριστικών περιλαμβάνει διανύσματα που αντιπροσωπεύουν κάθε δείγμα, τα οποία υποθέτουμε ότι ανήκουν σε μια μόνο κατηγορία.

Έστω $\mathcal{S} = \{c_i^s | i = 1, \ldots, N_s\}$ το σύνολο των ορατών κατηγοριών, όπου κάθε c_i^s είναι μια ορατή κλάση. Έστω επίσης $\mathcal{U} = \{c_i^u | i = 1, \ldots, N_u\}$ το σύνολο των αόρατων κατηγοριών, όπου κάθε c_i^u είναι μια αόρατη κατηγορία. Σημειώνουμε ότι $\mathcal{S} \cap \mathcal{U} = \emptyset$ [12]. Έστω $\mathcal{X}$ ο χώρος χαρακτηριστικών, ο οποίος είναι D-διαστάσεων. Έστω $D^{tr} = \{(\mathbf{x}_i^{tr}, y_i^{tr}) \in \mathcal{X} \times \mathcal{S} | i = 1, \ldots, N_{tr}\}$ το σύνολο των επισημασμένων δεδομένων εκπαίδευσης που ανήκουν στις ορατές κατηγορίες. Για κάθε επισημειωμένο παράδειγμα $(\mathbf{x}_i^{tr}, y_i^{tr})$, $\mathbf{x}_i^{tr}$ είναι το παράδειγμα στον χώρο χαρακτηριστικών, ενώ y_i^{tr} είναι η αντίστοιχη ετικέτα κλάσης. Έστω $X^{te} = \{\mathbf{x}_i^{te} \in \mathcal{X} | i = 1, \ldots, N_{te}\}$ το σύνολο των παραδειγμάτων δοκιμών, όπου κάθε $\mathbf{x}_i^{te}$ είναι ένα παράδειγμα δοκιμών στον χώρο χαρακτηριστικών. Έστω $Y^{te} = \{y_i^{te} \in \mathcal{U} | i = 1, \ldots, N_{te}\}$ οι αντίστοιχες ετικέτες κατηγορίας για τα X^{te}.

Ορισμός 1.2.1 (Μάθηση Χωρίς Παραδείγματα [12])**.** Ο στόχος της μάθησης χωρίς παραδείγματα είναι να διδάξει έναν ταξινομητή $f_{\text{ZSL}}(\cdot) : \mathcal{X} \to \mathcal{U}$ ώστε να μπορεί να ταξινομήσει δείγματα δοκιμής X^{te} που ανήκουν στις αόρατες κατηγορίες $\mathcal{U}$.

Πρόσθετη Πληροφορία

Για να αντιμετωπιστεί η απουσία ετικετών στα δείγματα των αόρατων κατηγοριών στη μάθηση χωρίς παραδείγματα, οι πρόσθετες πληροφορίες είναι απαραίτητες. Οι υπάρχουσες προσεγγίσεις αντλούν έμπνευση από τον ανθρώπινο τρόπο σκέψης, όπου οι σημασιολογικές γνώσεις βοηθούν στην αναγνώριση άγνωστων οντοτήτων [33]. Για παράδειγμα, γνωρίζοντας ότι "μια τίγρη μοιάζει με μια μεγάλη γάτα με ρίγες" επιτρέπει την αναγνώριση μιας τίγρης ακόμη και χωρίς προηγούμενη έκθεση σε σχετικές εικόνες, βασιζόμενοι στη γνώση των γατών, των μεγεθών και των μοτίβων ρίγας [5]. Συνεπώς, οι βοηθητικές πληροφορίες στις μεθόδους μάθησης χωρίς παραδείγματα συνήθως περιλαμβάνουν σημασιολογικές λεπτομέρειες, οι οποίες σχηματίζουν έναν *σημασιολογικό χώρο* που περιλαμβάνει τόσο τις ορατές όσο και τις αόρατες κατηγορίες.

Στον σημασιολογικό χώρο, κάθε κλάση χαρακτηρίζεται από ένα διανύσμα, το οποίο αναφέρεται ως το *πρωτότυπο κατηγορίας*. Στη συνέχεια, ακολουθούμε το [34] για να ορίσουμε τον σημασιολογικό χώρο και το πρωτότυπο μιας κατηγορίας.

Ας ονομάσουμε $\mathcal{T}$ τον σημασιολογικό χώρο. Υποθέτουμε ότι ο $\mathcal{T}$ είναι M-διάστατος· συνήθως είναι ο $\mathbb{R}^M$ [12]. Ας ονομάσουμε $\mathbf{t}_i^s \in \mathcal{T}$ το πρωτότυπο κατηγορίας για την ορατή κατηγορία c_i^s και $\mathbf{t}_i^u \in \mathcal{T}$ το πρωτότυπο για την αόρατη κατηγορία c_i^u. Ας ονομάσουμε $T^s = \{\mathbf{t}_i^s | i = 1, \ldots, S\}$ το σύνολο που περιέχει τα πρωτότυπα των ορατών κατηγοριών και $T^u = \{\mathbf{t}_i^u | i = S+1, \ldots, S+U\}$ το σύνολο των πρωτοτύπων των αόρατων κατηγοριών. Ας ορίσουμε τη $\pi(\cdot) : \mathcal{S} \cup \mathcal{U} \to \mathcal{T}$ ως μια *συνάρτηση πρωτοτύπων κλάσης* που λαμβάνει ως είσοδο μια ετικέτα κατηγορίας και εξάγει το αντίστοιχο πρωτότυπο κλάσης (π.χ. $\pi($ζέβρα$) \in \mathbb{R}^M$).

Σενάρια Εκπαίδευσης

Όπως δηλώθηκε στον ορισμό 1.2.1, ο στόχος της παραδοσιακής μάθησης χωρίς παραδείγματα (ΜΧΠ) είναι να διδάξουμε έναν ταξινομητή $f^u(\cdot)$. Στην πράξη μπορεί να προκύψουν καταστάσεις όπου υπάρχουν πληροφορίες για ορισμένες αόρατες κατηγορίες εκ των προτέρων. Τότε το μοντέλο είναι *ημιεπαγωγικό* (transductive) σε σχέση με αυτές τις συγκεκριμένες κλάσεις. Σε αυτή την περίπτωση, έχουμε πληροφορίες και για τις αόρατες κλάσεις κατά τη διάρκεια της εκπαίδευσης. Συγκεκριμένα, οι Wang et al. [34] διακρίνουν τις ακόλουθες περιπτώσεις:

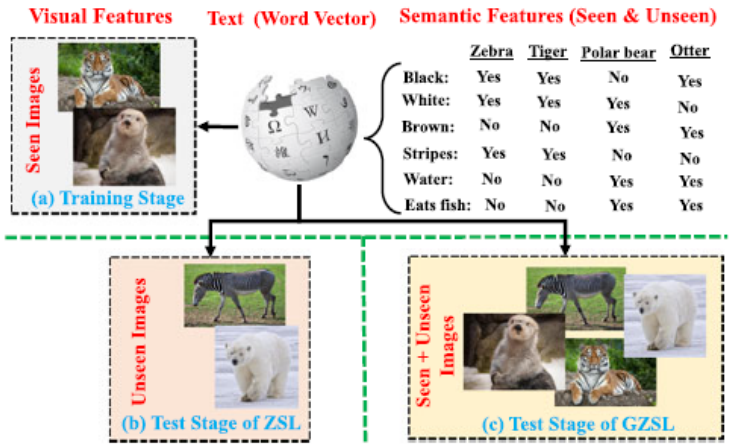
Ορισμός 1.2.2 (Σενάριο Επαγωγικών κλάσεων-Επαγωγικών Δειγμάτων)**.** Μόνο τα επισημειωμένα δείγματα εκπαίδευσης D^{tr} και τα πρωτότυπα των ορατών κατηγοριών T^s χρησιμοποιούνται στην εκπαίδευση του μοντέλου.

Ορισμός 1.2.3 (Σενάριο Ημιεπαγωγικών Κλάσεων-Ημιεπαγωγικών Δειγμάτων)**.** Τα επισημειωμένα δείγματα D^{tr}, τα πρωτότυπα των ορατών κατηγοριών T^s, τα μη επισημειωμένα δείγματα X^{te} και τα πρωτότυπα των αόρατων κατηγοριών T^u χρησιμοποιούνται στην εκπαίδευση του μοντέλου.

Σενάρια Αξιολόγησης

Στην παραδοσιακή μάθηση χωρίς παραδείγματα, το σύνολο δοκιμών περιλαμβάνει αποκλειστικά δείγματα από αόρατες κατηγορίες, ένα σενάριο που δεν είναι ρεαλιστικό στις πραγματικές εφαρμογές. Στην πράξη, τα δείγματα από ορατές κατηγορίες είναι πιο συχνά από αυτά των αόρατων κατηγοριών. Επομένως, είναι κρίσιμο να αναγνωρίζουμε ταυτόχρονα δείγματα και από τους δύο τύπους κλάσεων και να μην εστιάζουμε μόνο στα δείγματα των αόρατων κατηγοριών. Αυτό το σενάριο ονομάζεται *γενικευμένη μάθηση χωρίς παραδείγματα* (generalized zero-shot learning - GZSL) [35].

Ορισμός 1.2.4 (Γενικευμένη Μάθηση Χωρίς Παραδείγματα (ΓΜΧΠ) [5]). Ο στόχος της γενικευμένης μάθησης χωρίς παραδείγματα είναι η κατασκευή ενός ταξινομητή $f_{GZSL} : \mathcal{X} \rightarrow \mathcal{S} \cup \mathcal{U}$ που μπορεί να ταξινομήσει δείγματα του X^{te}, που ανήκουν τόσο στις ορατές κατηγορίες $\mathcal{S}$ όσο και στις αόρατες κατηγορίες $\mathcal{U}$.



Σχήμα 1.5: (β) Κατά τη φάση της αξιολόγησης, η ΜΧΠ μπορεί να αναγνωρίσει μόνο δείγματα από τις αόρατες κατηγορίες, ενώ η (γ) ΓΜΧΠ μπορεί να αναγνωρίσει δείγματα τόσο από τις ορατές όσο και από τις αόρατες κατηγορίες (Πηγή: [5])

Η γενικευμένη μάθηση χωρίς παραδείγματα (ΓΜΧΠ) δεν αναφέρει ρητά το σύνολο που χρησιμοποιείται κατά τη φάση της εκπαίδευσης, όπως και η μάθησης χωρίς παραδείγματα (ΜΧΠ). Ο παράγοντας διαφοροποίησης έγκειται στο ερώτημα που τίθεται στο μοντέλο κατά την διάρκεια της δοκιμής [5]. Στην κλασσική ΜΧΠ, οι ετικέτες των δειγμάτων του συνόλου δοκιμών X^{te} υποτίθεται ότι ανήκουν μόνο στις αόρατες κατηγορίες $\mathcal{U}$. Στην ΓΜΧΠ δεν γίνεται αυτή η υπόθεση, οπότε οι ετικέτες του X^{te} μπορεί να ανήκουν τόσο στο $\mathcal{S}$ όσο και στο $\mathcal{U}$. Αυτή η θεμελιώδης διαφορά συνοψίζεται στο κάτω μέρος του Σχήματος 1.5.

Ο Αλγόριθμος K-means

Ο αλγόριθμος συσταδοποίησης K-means είναι μια μέθοδος που χρησιμοποιείται στη μη επιβλεπόμενη μάθηση για να ομαδοποιήσει δεδομένα σε συστάδες με παρόμοια χαρακτηριστικά. Ο στόχος είναι να κατατμηθεί ένα σύνολο δεδομένων σε K διακριτές ομάδες (συστάδες) όπου κάθε σημείο δεδομένων ανήκει στη συστάδα με τη πλησιέστερη μέση τιμή. Αυτή η ομαδοποίηση βοηθά στην κατανόηση της δομής των δεδομένων, στον εντοπισμό προτύπων και στην απλοποίηση πολύπλοκων συνόλων δεδομένων.

Υπάρχουν αρκετοί λόγοι για τους οποίους αυτός ο αλγόριθμος είναι χρήσιμος:

- **Απλότητα και Ταχύτητα**: Ο K-means είναι εύκολο να κατανοηθεί και να υλοποιηθεί. Λειτουργεί επίσης αποδοτικά σε μεγάλα σύνολα δεδομένων, καθιστώντας το μια πρακτική επιλογή για πολλές πραγματικές εφαρμογές.

- **Αναγνώριση Προτύπων**: Μέσω της ομαδοποίησης παρόμοιων δεδομένων, ο K-means βοηθά στην αναγνώριση προτύπων και δομών μέσα στα δεδομένα που μπορεί να μην είναι άμεσα εμφανή.

- **Μείωση Διαστασικότητας**: Αν και ο κύριος στόχος είναι η ομαδοποίηση, τα παραγόμενα κέντρα μπορούν να χρησιμοποιηθούν για να αναπαραστήσουν το σύνολο δεδομένων σε πιο συμπαγή μορφή, βοηθώντας στη μείωση της διάστασής τους.

Διατύπωση του Προβλήματος Ας υποθέσουμε ότι έχουμε ένα σύνολο δεδομένων $x_1, \ldots, x_N$ που αποτελείται από N πραγματοποιήσεις μιας τυχαίας D-διάστατης μεταβλητής x. Ο στόχος μας είναι να διαχωρίσουμε το σύνολο δεδομένων σε K συστάδες. Τυπικά, αναζητούμε K σημεία $\mu_1, \ldots, \mu_K$ στα οποία αν αναθέσουμε τα σημεία $\{x_1, \ldots, x_N\}$, θα ελαχιστοποιηθεί ένα μέτρο σφάλματος.

Ακολουθώντας το [15], για κάθε σημείο δεδομένων x_n, ας εισαγάγουμε τον δυαδικό δείκτη $r_{nk} \in \{0, 1\}$. Αν το διάνυσμα x_i ανατεθεί στη συστάδα j, τότε $r_{ij} = 1$. Διαφορετικά, $r_{ij} = 0$. Όπως συνήθως στη μηχανική μάθηση, προσπαθούμε να ελαχιστοποιήσουμε κάποια συνάρτηση κόστους. Σε αυτή την περίπτωση, η πιο συνηθισμένη συνάρτηση κόστους είναι:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{1.15}$$

Η παραπάνω εξίσωση αντιπροσωπεύει το άθροισμα των τετραγωνικών αποστάσεων των σημείων του συνόλου δεδομένων από το κέντρο στο οποίο έχουν ανατεθεί. Προφανώς, οι βέλτιστες τιμές για τα r_{nk} και μ_k εξαρτώνται η μία από την άλλη. Για αυτό το λόγο, χρησιμοποιείται ένας αλγόριθμος δύο σταδίων.

Αρχικά, ας βρούμε τις βέλτιστες τιμές για τα r_{nk} δεδομένων των κέντρων μ_k. Οι όροι που αφορούν διαφορετικά δείγματα n είναι ανεξάρτητοι, επομένως μπορούμε να βελτιστοποιήσουμε για κάθε n ξεχωριστά επιλέγοντας το r_{nk} να είναι 1 για το k που δίνει την ελάχιστη τιμή του $||x_n - \mu_k||^2$. Με άλλα λόγια:

$$r_{nk} = \begin{cases} 1 & \text{έαν } k = \arg\min_j ||x_n - \mu_j||^2 \\ 0 & \text{αλλιώς} \end{cases} \tag{1.16}$$

Στη συνέχεια, ας βρούμε τις βέλτιστες τιμές για τα μ_k δεδομένων των αναθέσεων r_{nk}. Για να βρούμε τη βέλτιστη τιμή για το μ_k, απλώς θέτουμε την παράγωγο του J ως προς το μ_k ίση με το μηδέν, το οποίο δίνει:

$$2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \tag{1.17}$$

Λύνοντας την παραπάνω εξίσωση προς μ_k βρίσκουμε

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}} \tag{1.18}$$

1.3 Σχετιζόμενη βιβλιογραφεία

Όπως συζητήθηκε στο προηγούμενο κεφάλαιο, η μάθηση χωρίς παραδείγματα (ΜΧΠ) είναι ένας αναδυόμενος τομέας στη μηχανική μάθηση που αντιμετωπίζει την πρόκληση της ταξινόμησης δειγμάτων σε κατηγορίες για τις οποίες η μηχανή δεν είχε δείγματα με ετικέτα (labeled samples) κατά τη φάση της εκπαίδευσης. Οι κύριες προσεγγίσεις στη ΜΧΠ μπορούν να κατηγοριοποιηθούν σε δύο κύριους τύπους: μεθόδους βασισμένες σε αναπαραστάσεις και παραγωγικές μεθόδους.

- Μέθοδοι βασισμένες σε αναπαραστάσεις: Αυτές οι μέθοδοι στοχεύουν να βρουν έναν κοινό κρυφό χώρο όπου τόσο τα οπτικά χαρακτηριστικά (π.χ. εικόνες) όσο και τα πρωτότυπα (π.χ. χαρακτηριστικά κατηγοριών ή διανύσματα λέξεων) μπορούν να προβληθούν. Ο στόχος είναι να επιτραπεί η σύγκριση αυτών των προβολών χρησιμοποιώντας ένα μέτρο ομοιότητας. Για παράδειγμα, ένα μοντέλο βασισμένο σε αναπαραστάσεις προβάλει μια εικόνα ενός ζώου αόρατης κλάσης και το πρωτότυπο της σε έναν σημασιολογικό χώρο. Το μοντέλο στη συνέχεια ταξινομεί την εικόνα με βάση την εγγύτητά της στα πρωτότυπα διάφορων κλάσεων εντός αυτού του χώρου. Τέτοιες μέθοδοι βασίζονται σε μεγάλο βαθμό στην ποιότητα του χώρου αναπαραστάσεων και στο μέτρο ομοιότητας που χρησιμοποιείται: συνήθως την ομοιότητα συνημιτόνου ή την Ευκλείδεια απόσταση. Επιτυχημένα παραδείγματα μεθόδων βασισμένων σε αναπαραστάσεις περιλαμβάνουν μοντέλα που χρησιμοποιούν βαθιά νευρωνικά δίκτυα για να μάθουν αυτές τις προβολές, συλλαμβάνοντας περίπλοκες σχέσεις μεταξύ του οπτικού και του σημασιολογικού χώρου [16], [36], [37].

- Παραγωγικές Μέθοδοι: Σε αντίθεση με τις μεθόδους βασισμένες σε αναπαραστάσεις, οι παραγωγικές προσεγγίσεις στοχεύουν στη σύνθεση οπτικών χαρακτηριστικών ή ακόμα και ολόκληρων εικόνων για τις αόρατες κατηγορίες, αξιοποιώντας τη γνώση από τις ορατές κατηγορίες και τις σημασιολογικές τους περιγραφές (τα πρωτότυπά τους). Δημιουργώντας δείγματα για τις αόρατες κατηγορίες, το πρόβλημα ΜΧΠ μετατρέπεται σε ένα πρόβλημα επιβλεπόμενης μάθησης, όπου το σύνολο εκπαίδευσης πλέον περιλαμβάνει συνθετικά παραδείγματα και των αόρατων κατηγοριών. Τα Γενετικά Ανταγωνιστικά Δίκτυα (Generative Adversarial Networks) [38] και οι Μεταβολικοί Αυτόκωδικοποιητές (Variational Autoencoders) [39] χρησιμοποιούνται συνήθως σε αυτές τις μεθόδους για τη δημιουργία υψηλής ποιότητας και διαφοροποιημένων δειγμάτων που μιμούνται στενά πραγματικά παραδείγματα. Για παράδειγμα, δεδομένης μιας γλωσσικής περιγραφής ενός αόρατου ζώου, ένα GAN μπορεί να δημιουργήσει εικόνες που αντιστοιχούν σε αυτήν την περιγραφή, παρέχοντας έτσι τα απαραίτητα δεδομένα για την εκπαίδευση ενός ταξινομητή με επιβλεπόμενο τρόπο.

Δεδομένου ότι η μέθοδος μας βασίζεται σε αναπαραστάσεις, θα εξετάσουμε αυτές τις μεθόδους σε μεγαλύτερο βάθος και θα παρέχουμε περισσότερες λεπτομέρειες σχετικά με τις συναφείς εργασίες. Ωστόσο, για να παρέχουμε μια ολοκληρωμένη επισκόπηση του πεδίου, θα παρουσιάσουμε σύντομα μερικές από τις πιο σχετικές παραγωγικές μεθόδους. Ο ενδιαφερόμενος αναγνώστης μπορεί να διαβάσει το κεφάλαιο 4 για περισσότερες λεπτομέρειες.

Ο Αλγόριθμος SCAN

Η μέθοδος "SCAN: Learning to Classify Images without Labels" [40] είναι μια από τις καλύτερες μεθόδους (SOTA) για μη επιβλεπόμενη συσταδοποίηση εικόνων. Αυτή η μέθοδος περιλαμβάνει τρία κύρια βήματα: αυτοεπιβλεπόμενη μάθηση, ομαδοποίηση με τους πλησιέστερους γείτονες και βελτιστοποίηση μέσω αυτοεπισημείωσης (self-labeling).

Αναλυτική Εξήγηση της Διαδικασίας

- **Αυτοεπιβλεπόμενη Μάθηση**: Ένα συνελικτικό νευρωνικό δίκτυο (ΣΝΝ) εκπαιδεύεται χρησιμοποιώντας αυτοεπιβλεπόμενες εργασίες μάθησης. Τέτοιου είδους εργασίες είναι για παράδειγμα η πρόβλεψη της περιστροφής μιας εικόνας ή η διάκριση μεταξύ παραλλαγμένων εκδόσεων της ίδιας εικόνας. Αυτές οι εργασίες αναγκάζουν το δίκτυο να μάθει αναπαραστάσεις υψηλής ποιότητας που συλλαμβάνουν σημαντικά οπτικά χαρακτηριστικά.

- **Ομαδοποίηση με τους Πλησιέστερους Γείτονες**: Μετά την αυτοεπιβλεπόμενη μάθηση, εξάγονται χαρακτηριστικά από τις εικόνες. Για κάθε εικόνα, οι k-πλησιέστεροι γείτονες της εντοπίζονται με βάση την ομοιότητα των χαρακτηριστικών τους. Αυτοί οι πλησιέστεροι γείτονες χρησιμοποιούνται στη συνέχεια για την εκπαίδευση του *δικτύου ομαδοποίησης*. Η μέθοδος χρησιμοποιεί μια συνάρτηση απώλειας που μεγιστοποιεί την ομοιότητα (εσωτερικό γινόμενο) μεταξύ κάθε εικόνας και των χαρακτηριστικών των γειτόνων της, ενθαρρύνοντας το δίκτυο να παράγει διακριτές αναθέσεις συστάδων.

- **Βελτιστοποίηση μέσω Αυτοεπισημείωσης**: Οι αρχικές συστάδες που σχηματίζονται από τις αναθέσεις των πλησιέστερων γειτόνων χρησιμοποιούνται για τη δημιουργία ψευδο-ετικετών για τις εικόνες. Αυτές οι ψευδο-ετικέτες λειτουργούν ως ασθενείς ετικέτες για επιβλεπόμενη βελτιστοποίηση. Το δίκτυο στη συνέχεια βελτιστοποιείται μέσω εκπαίδευσης στα ψευδο-επισημειωμένα δεδομένα, χρησιμοποιώντας μια τυπική συνάρτηση απώλειας ταξινόμησης (π.χ. συνάρτηση απώλειας διασταυρούμενης εντροπίας). Αυτή η επαναληπτική διαδικασία βελτιώνει τις συστάδες, καθώς το δίκτυο μαθαίνει να παράγει πιο ακριβείς ψευδο-ετικέτες σε κάθε επανάληψη.

Κίνητρο

Στον ταχέως εξελισσόμενο τομέα της μάθησης χωρίς παραδείγματα (ΜΧΠ), οι μέθοδοι που βασίζονται σε παραγωγικά μοντέλα κυριαρχούν στη βιβλιογραφία. Αυτές οι μέθοδοι και ειδικά εκείνες που εμπλέκουν Παραγωγικά Ανταγωνιστικά Δίκτυα (GAN's), έχουν δείξει ενθαρρυντικά αποτελέσματα, συνθέτοντας οπτικά χαρακτηριστικά για αόρατες κατηγορίες και μετατρέποντας το πρόβλημα ΜΧΠ σε ένα συμβατικό πρόβλημα επιβλεπόμενης μάθησης. Ωστόσο, συχνά αντιμετωπίζουν προβλήματα αστάθειας κατά την εκπαίδευση, που μπορούν να οδηγήσουν σε ασυνεπή απόδοση. Για παράδειγμα, σε μια μελέτη από τους Chochlakis et al. [41], μια προσπάθεια να αναπαραχθούν τα αποτελέσματα της μεθόδου αιχμής (SOTA) f-VAEGAN [42] αποκάλυψε ότι τα αποτελέσματα αναπαραγωγής τους ήταν 4-8% χαμηλότερα από την αναφερόμενη απόδοση του αρχικού άρθρου, ανάλογα με το σύνολο δεδομένων. Αυτή η διαφορά δείχνει τις προκλήσεις και την ενδεχόμενη αναξιοπιστία που σχετίζονται με τις παραγωγικές προσεγγίσεις.

Από την άλλη πλευρά, οι μέθοδοι που βασίζονται σε αναπαραστάσεις προσφέρουν ένα απλούστερο και πιο εποπτικό πλαίσιο για τη Μάθηση Χωρίς Παραδείγματα. Αυτές οι μέθοδοι λειτουργούν με μια απλή αρχή: αναζητώντας έναν χώρο στον οποίο διαφορετικές κατηγορίες παρουσιάζουν ενδο-κατηγορική συμπάγεια (intra-class compactness) και δια-κατηγορική διαχωρισιμότητα (inter-class separability). Αυτή η θεμελιώδης ιδέα, η οποία εκφράστηκε για πρώτη φορά από τον αλγόριθμο TEDE [16], αποτελεί τη βάση πολλών προσεγγίσεων βασισμένων σε αναπαραστάσεις. Η απλότητα αυτής της ιδέας καθιστά τις μεθόδους βασισμένες σε αναπαραστάσεις πιο κατανοητές σε σύγκριση με τις γενετικές μεθόδους.

Βασιζόμενοι σε αυτή την αρχή, η έρευνά μας στοχεύει να αξιοποιήσει τις προόδους στη βιβλιογραφία της μη επιβλεπόμενης συσταδοποίησης, η οποία επικεντρώνεται εγγενώς στην επίτευξη ενδο-κατηγορικής συμπάγειας και δια-κατηγορικής διαχωρισιμότητας. Με τον συνδυασμό της τεχνικής συσταδοποίησης SCAN με τις μεθόδους ΜΧΠ που βασίζονται σε αναπαραστάσεις, φιλοδοξούμε να αναπτύξουμε ένα πιο αξιόπιστο πλαίσιο για τη μάθηση χωρίς παραδείγματα. Συνοψίζοντας, ενώ οι μέθοδοι που βασίζονται σε παραγωγικά μοντέλα έχουν τα πλεονεκτήματά τους, οι προκλήσεις αστάθειας που εμφανίζουν, υποδεικνύουν τη σκοπιμότητα μιας περεταίρω επανεξέτασης των προσεγγίσεων βασισμένων σε αναπαραστάσεις. Επιπλέον, πιστεύουμε ότι αυτή η προσέγγιση απλοποιεί σημαντικά την μεθοδολογία των γενετικών μεθόδων.

1.4 Προτεινόμενη Μέθοδος

Στην παρούσα διπλωματική, αντιμετωπίζουμε το πρόβλημα της ημιεπαγωγικής μάθησης χωρίς παραδείγματα - ΜΧΠ (transductive zero-shot learning - ZSL). Στη ΜΧΠ, το μοντέλο εκπαιδεύεται με ένα σύνολο κατηγοριών, αλλά κατά τη διάρκεια της δοκιμής αναμένεται να αναγνωρίσει και να ταξινομήσει παραδείγματα από κατηγορίες που δεν έχει δει κατά την εκπαίδευση. Αυτό συνεπάγεται ότι το μοντέλο πρέπει να γενικεύσει τη γνώση του σε αόρατες κατηγορίες βασιζόμενο σε κάποια μορφή σημασιολογικής κατανόησης. Στην ημιεπαγωγική μάθηση, το μοντέλο στοχεύει στην πρόβλεψη ετικετών για ένα σύνολο μη επισημειωμένων δεδομένων, εκμεταλλευόμενο συνήθως τις σχέσεις μεταξύ επισημειωμένων και μη επισημειωμένων δεδομένων [43]. Αυτό διαφέρει από το σύνηθες σενάριο της επαγωγικής μάθησης, όπου το μοντέλο μαθαίνει μια γενική αντιστοιχία από την είσοδο στην έξοδο, βασισμένη αποκλειστικά σε επισημειωμένα δεδομένα. Στην ημιεπαγωγική μάθηση, οι προβλέψεις του μοντέλου μπορεί να εξαρτώνται μόνο από την κατανομή των μη επισημειωμένων δεδομένων [43].

Συνδυάζοντας αυτές τις δύο έννοιες, η ημιεπαγωγική μάθηση χωρίς παραδείγματα αντιμετωπίζει το σενάριο όπου το μοντέλο πρέπει να ταξινομήσει παραδείγματα από τις αόρατες κατηγορίες, ενώ έχει πρόσβαση σε ένα σύνολο επισημειωμένων δειγμάτων που ανήκουν στις ορατές κατηγορίες και ένα σύνολο μη επισημειωμένων δειγμάτων που μπορεί να ανήκουν τόσο στις ορατές όσο και στις αόρατες κατηγορίες. Ο στόχος είναι να αξιοποιήσουμε τις σημασιολογικές σχέσεις μεταξύ των ορατών και των αόρατων κατηγοριών, προκειμένου να βελτιώσουμε την απόδοση ταξινόμησης του μοντέλου, παρά την έλλειψη άμεσης πρόσβασης σε επισημειωμένα παραδείγματα για τις αόρατες κατηγορίες.

Η προσέγγισή μας περιλαμβάνει την συσταδοποίηση όλων των περιπτώσεων από τις ορατές και αόρατες κατηγορίες, χρησιμοποιώντας ένα *δίκτυο συσταδοποίησης*, το οποίο περιλαμβάνει μια προεκπαιδευμένη ραχοκοκαλιά (pretrained backbone) και ένα μικρό νευρωνικό δίκτυο στην κορυφή της. Πιο συγκεκριμένα, μια προεκπαιδευμένη ραχοκοκαλιά αναφέρεται σε μια αρχιτεκτονική νευρωνικού δικτύου που έχει εκπαιδευτεί σε ένα μεγάλο σύνολο δεδομένων για μια συγκεκριμένη εργασία και τα μαθημένα βάρη του χρησιμοποιούνται στη συνέχεια ως αφετηρία για την εξαγωγή χαρακτηριστικών, συχνά για μια διαφορετική εργασία. Η χρήση μιας προεκπαιδευμένης ραχοκοκαλιάς προσφέρει πολλά οφέλη. Επιτρέπει την αξιοποίηση της γνώσης που αποκτήθηκε από μεγάλα σύνολα δεδομένων, η οποία μπορεί να βοηθήσει στη βελτίωση της απόδοσης σε εργασίες με περιορισμένα δεδομένα εκπαίδευσης. Επιπλέον, μπορεί να επιταχύνει τον χρόνο εκπαίδευσης, δεδομένου ότι η ραχοκοκαλιά έχει ήδη μάθει να εξάγει χρήσιμα χαρακτηριστικά. Στην παρούσα εργασία επιλέξαμε το μοντέλο ResNet101 ως τη ραχοκοκαλιά μας [44]. Ακολουθώντας το [40], στην κορυφή της ραχοκοκαλιάς προστίθεται ένα μικρό νευρωνικό δίκτυο το οποίο έχει έναν πολύ συγκεκριμένο στόχο. Ο στόχος είναι να κατασκευαστεί ένας χώρος όπου οι εικόνες που ανήκουν στην ίδια κατηγορία είναι κοντά μεταξύ τους, ενώ οι εικόνες που ανήκουν σε διαφορετικές κατηγορίες είναι μακριά η μία από την άλλη. Αυτή η προσέγγιση αξιοποιεί τόσο τη γενική γνώση που είναι κωδικοποιημένη στην προεκπαιδευμένη ραχοκοκαλιά, όσο και τις συγκεκριμένες απαιτήσεις της εργασίας που αντιμετωπίζονται από το μικρό νευρωνικό δίκτυο, το οποίο ειδικεύεται για αυτή την εργασία, οδηγώντας σε μια αποτελεσματική λύση για το πρόβλημα της συσταδοποίησης.

Μετά από αυτό το βήμα, προβάλλουμε όλες τις εικόνες στον οπτικό χώρο που δημιουργείται από το εκπαιδευμένο δίκτυο συσταδοποίησης. Στη συνέχεια, υπολογίζουμε το μέσο κάθε ορατής κατηγορίας και χρησιμοποιούμε τον αλγόριθμο K-means για να εκτιμήσουμε τους μέσους των αόρατων κατηγοριών. Θα αναφερόμαστε στις μέσες τιμές που αναφέραμε πιο πάνω, ως οπτικούς αντιπροσώπους.

Έχοντας βρει τους οπτικούς αντιπροσώπους, υποθέτουμε ότι τα δείγματα της εκάστοτε κατη-

γορίας είναι καλά ομαδοποιημένα μεταξύ τους, ενώ τα δείγματα από διαφορετικές κατηγορίες είναι μακριά το ένα από το άλλο. Εάν αυτή η υπόθεση ισχύει, τότε οι οπτικοί αντιπρόσωποι είναι αντιπροσωπευτικά σημεία που χαρακτηρίζουν κάθε κλάση. Αυτά τα σημεία συλλαμβάνουν την ουσία κάθε κατηγορίας, επιτρέποντάς μας να μετρήσουμε την ομοιότητα ή την απόσταση μεταξύ μιας εικόνας εισόδου και του κέντρου κάθε κλάσης. Η ταξινόμηση τότε γίνεται θέμα της ανάθεσης της εικόνας εισόδου στην κατηγορία της οποίας το κέντρο είναι πλησιέστερο, δεδομένου κάποιου μέτρου απόστασης, όπως η Ευκλείδεια απόσταση ή η ομοιότητα συνημιτόνου.

Έτσι, το πρόβλημα καταλήγει στην ανάθεση ενός ονόματος κατηγορίας (π.χ. σκύλος) σε κάθε οπτικό αντιπρόσωπο. Η αντιστοιχία μεταξύ των γνωστών κατηγοριών, των οποίων οι ετικέτες χρησιμοποιούνται κατά την εκπαίδευση, και ενός υποσυνόλου των οπτικών αντιπροσώπων είναι ήδη γνωστή, όπως και η αντιστοιχία τους με τα γνωστά σημασιολογικά διανύσματα κατηγοριών. Αυτό το σύνολο αντιστοιχιών μπορεί να θεωρηθεί ως συνάρτηση, την οποία θα θέλαμε να επεκτείνουμε με τον 'καλύτερο' δυνατό τρόπο, προκειμένου να αντιστοιχίσουμε το σύνολο των σημασιολογικών διανυσμάτων των άγνωστων κατηγοριών, στο σύνολο των άγνωστων οπτικών αντιπροσώπων, με έναν αμφιμονοσήμαντο τρόπο. Η υποκείμενη υπόθεση είναι ότι οι δύο χώροι μοιράζονται κάποια μορφή δομικής ομοιότητας, κάτι που δεν είναι δεδομένο.

Το περιγραφόμενο πρόβλημα παρουσιάζει ομοιότητες με μια καλά μελετημένο πρόβλημα στον τομέα της επεξεργασίας φυσικής γλώσσας, γνωστό ως bilingual lexicon induction (BLI) [45], [46]. Το BLI στοχεύει να βρει αντίστοιχες λέξεις ή φράσεις σε δύο διαφορετικές γλώσσες, συνήθως από συγκρίσιμα σώματα κειμένων, χωρίς επίβλεψη για όλα τα ζεύγη [46]. Όπως και στο σενάριό μας, όπου στοχεύουμε να καθιερώσουμε αντιστοιχίες μεταξύ των κέντρων των συστάδων και των σημασιολογικών διανυσμάτων, το BLI αξιοποιεί δομικές ομοιότητες μεταξύ των δύο γλωσσικών χώρων για να συμπεράνει αντιστοιχίες.

Υιοθετώντας τεχνικές από πρόσφατες προόδους στον τομέα [46], όπως η επιβλεπόμενη ευθυγράμμιση προβολών (supervised embedding alignment) και η μη επιβλεπόμενη αντιστοίχιση κατανομών (unsupervised distribution alignment), το πλαίσιο που χρησιμοποιούμε βελτιστοποιεί από κοινού τόσο τις γνωστές αντιστοιχίες, όσο και τις δομικές ομοιότητες μεταξύ των σημασιολογικών και οπτικών χώρων, προκειμένου να βρεθεί μια αρχική αντιστοίχιση από το σημασιολογικό χώρο στον οπτικό χώρο. Στη συνέχεια, χρησιμοποιείται ο αλγόριθμος Iterative Procrustes Refinement (IPR) [46], προκειμένου να ευθυγραμμιστούν προοδευτικά περισσότερα ζεύγη σημείων και να βελτιωθεί η αντιστοίχιση με ένα εναλλακτικό σχήμα δύο σταδίων.

Τέλος, όταν όλα τα σημεία έχουν αντιστοιχιστεί με τη διαδικασία που περιγράφηκε παραπάνω, προκύπτει μια τελική αντιστοίχιση, την οποία χρησιμοποιούμε για να προβάλλουμε τα σημασιολογικά διανύσματα στον οπτικό χώρο. Στη συνέχεια, κάθε εικόνα ταξινομείται στο πλησιέστερο προβεβλημένο σημασιολογικό διάνυσμα.

Από όσο γνωρίζουμε, υπάρχουν μόνο λίγες μέθοδοι στην επαγωγική μάθηση χωρίς παραδείγματα που χρησιμοποιούν την ομαδοποίηση ως βασικό στοιχείο της προσέγγισής τους [47], [48]. Για αυτό το λόγο, η μέθοδός μας η οποία αξιοποιεί την συσταδοποίηση τόσο για τις ορατές όσο και για τις αόρατες κατηγορίες, αντιπροσωπεύει μια συμβολή σε αυτόν τον τομέα. Σημειώνουμε ότι η προσέγγισή μας μοιράζεται ομοιότητες με τη [49] που πρώτη εισήγαγε την ιδέα της αντιστοίχισης κάθε πρωτοτύπου κατηγορίας σε ένα συγκεκριμένο κέντρο με έναν αμφιμονοσήμαντο τρόπο, χρησιμοποιώντας δομικούς περιορισμούς για τον οπτικό και σημασιολογικό χώρο. Ωστόσο, δεν καταφέραμε να επαληθεύσουμε τα αποτελέσματά της προαναφερόμενης εργασίας στα πειράματά μας.

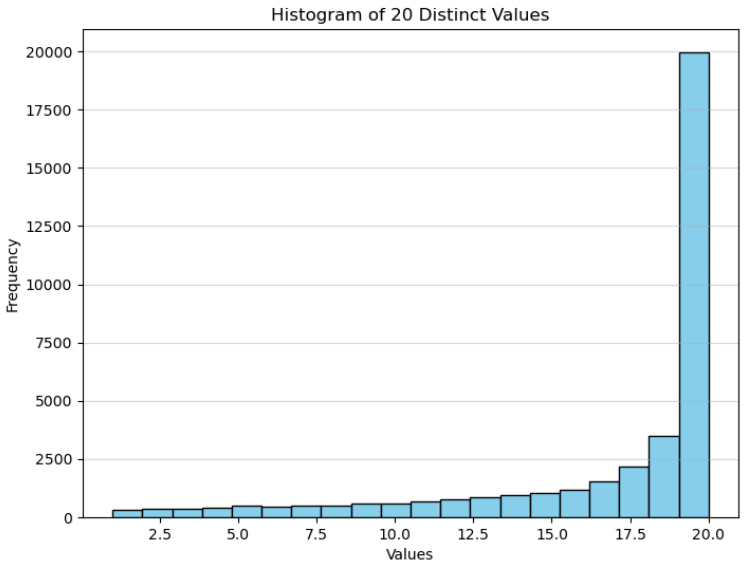
Ορισμός Προβλήματος Στην περίπτωση μας, έχουμε N_s επισημειωμένα δείγματα $D_s = \{(x_i^s, y_i^s)|i = 1, \ldots, N_s\}$, όπου το x_i^s είναι μια εικόνα και το $y_i^s \in \mathcal{S} = \{1, 2, \ldots, S\}$ είναι η αντίστοιχη ετικέτα η οποία ανήκει σε μια από τις S συνολικά ορατές κατηγορίες. Επίσης, μας

δίνονται N_u μη επισημειωμένα δείγματα $D_u = \{x_i^u | i = 1, \ldots, N_u\}$ που προέρχονται από τις αόρατες κατηγορίες $\mathcal{U} = \{S+1, \ldots, S+U\}$. Σημειώνεται ότι $\mathcal{S} \cap \mathcal{U} = \emptyset$, αλλά οι κατηγορίες συσχετίζονται σε έναν σημασιολογικό χώρο $\mathcal{T}$. Στόχος μας είναι να προβλέψουμε τις ετικέτες $y_i^u \in \mathcal{U}$, δεδομένων των εικόνων $x_i^u \in D_u$ και των συνόλων πρωτοτύπων T^s και T^u.

1.4.1 Βήμα Συσταδοποίησης

Σε αυτό το βήμα της μεθόδου μας, χρησιμοποιούμε τον αλγόριθμο SCAN για να ομαδοποιήσουμε όλες τις εικόνες [40] (δείγματα που ανήκουν σε ορατές και αόρατες κατηγορίες). Συγκεκριμένα, οι Gasnbeke et al. [40] ξεκίνησαν προεκπαιδεύοντας ένα νευρωνικό δίκτυο χρησιμοποιώντας τον αλγόριθμο SimCLR [50] ή τον αλγόριθμο MOCO [51]. Μετά τη φάση της προεκπαίδευσης, παρατηρήθηκε ότι οι εικόνες που ανήκουν στην ίδια κατηγορία τείνουν να ομαδοποιούνται μαζί [40].

Στην παρούσα μελέτη, αποφασίσαμε να μην χρησιμοποιήσουμε τέτοιους αλγόριθμους, καθώς απαιτούν τη χρήση επαρκώς μεγάλου μεγέθους παρτίδας (batch size) και ενός μεγάλου νευρωνικού δικτύου. Αντ΄ αυτού, αξιοποιήσαμε το προεκπαιδευμένο δίκτυο ResNet101, το οποίο είχε εκπαιδευτεί στο εκτεταμένο σύνολο δεδομένων ImageNet. Αυτή η απόφαση ελήφθη για να χρησιμοποιήσουμε τα χαρακτηριστικά τα οποία ήδη έχει μάθει το δίκτυο αυτό. Το ιστόγραμμα στο Σχήμα 1.6 απεικονίζει την κατανομή των σωστών γειτονικών αντιστοιχιών μεταξύ των 20 πλησιέστερων γειτόνων κάθε δείγματος στο σύνολο δεδομένων AwA2 [6]. Κάθε μπάρα στο ιστόγραμμα αντιπροσωπεύει τη συχνότητα των δειγμάτων για τα οποία εντοπίστηκε ένας συγκεκριμένος αριθμός σωστών γειτόνων. Για παράδειγμα, μια μπάρα σε μια συγκεκριμένη τιμή x υποδηλώνει τον αριθμό των δειγμάτων με αυτόν τον ακριβή αριθμό σωστών γειτόνων.



Σχήμα 1.6: Τα γειτονικά δείγματα τείνουν να ανήκουν στην ίδια κλάσση

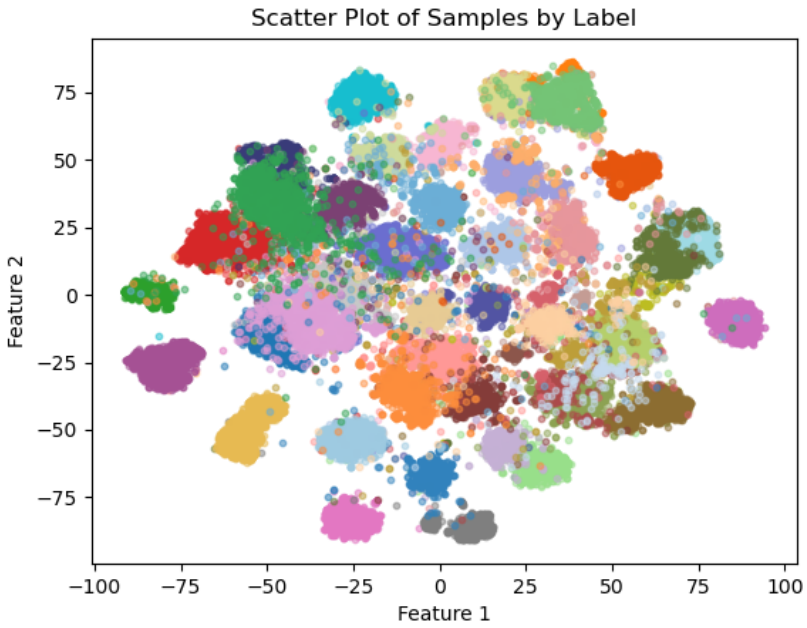
Εξαγωγή Πλησιέστερων Γειτόνων

Στην προηγούμενη ενότητα, αναφέραμε τη χρήση μιας μεγάλης προεκπαιδευμένης ραχοκοκαλιάς για την απόκτηση οπτικών χαρακτηριστικών. Ωστόσο, η απλή εφαρμογή του K-means στα ληφθέντα χαρακτηριστικά μπορεί να οδηγήσει σε εκφυλισμό συστάδων [40]. Επιπλέον, όπως φαίνεται στο

σχήμα 1.7, τα προκύπτοντα χαρακτηριστικά δεν είναι επαρκώς γραμμικά διαχωρίσιμα. Αυτή η έλλειψη σαφούς διαχωρισμού υποδεικνύει ότι η απλή εφαρμογή της ομαδοποίησης K-means σε αυτό το πλαίσιο θα ήταν ακατάλληλη και θα μπορούσε να οδηγήσει σε ανακριβή και μη βέλτιστα αποτελέσματα. Για αυτό το λόγο, υιοθετούμε τον αλγόριθμο SCAN [40], αναζητώντας μια καλύτερη μεθοδολογία.

Συγκεκριμένα, χρησιμοποιώντας το μοντέλο ResNet101 , έχουμε αποκτήσει προβολές για ολόκληρο το σύνολο δεδομένων $D = D_s \cup D_u$. Στη συνέχεια, για κάθε μη επισημειωμένο δείγμα $x_i \in D_u$, εξάγουμε τους πλησιέστερους γείτονές του στον οπτικό χώρο που παράγει το ResNet. Πιο συγκεκριμένα, δεδομένου ότι υποθέτουμε ότι όλα τα μη επισημειωμένα δείγματα ανήκουν στις αόρατες κατηγορίες, επιλέγουμε μόνο γείτονες από το σύνολο δεδομένων D_u και όχι από ολόκληρο το σύνολο δεδομένων $D_s \cup D_u$. Τώρα, για κάθε δείγμα x_i στο σύνολο δεδομένων $D_s \cup D_u$, ορίζουμε το σύνολο N_{x_i} ως εξής: Αν η ετικέτα του x_i είναι γνωστή, δηλαδή $x_i \in D_s$, το σύνολο N_{x_i} περιέχει όλα τα δείγματα που έχουν την ίδια ετικέτα με το x_i. Διαφορετικά, αν το δείγμα είναι μη επισημειωμένο, δηλαδή $x_i \in D_u$, το σύνολο N_{x_i} περιέχει τους προαναφερθέντες K πλησιέστερους γείτονες. Με άλλα λόγια:

$$N_{x_i} = \begin{cases} \{x : x \text{ και } x_i \text{ έχουν την ίδια ετικέτα}\} & \text{εάν } x_i \in D_s \\ \{x : x \text{ είναι ένας από τους } K \text{ πλησιέστερους γείτονες του } x_i\} & \text{εάν } x_i \in D_u \end{cases} \tag{1.19}$$



Σχήμα 1.7: Οπτικοποίηση TSNE του συνόλου δεδομένων AwA2 [6]. Διαφορετικές κατηγορίες απεικονίζονται με διαφορετικό χρώμα.

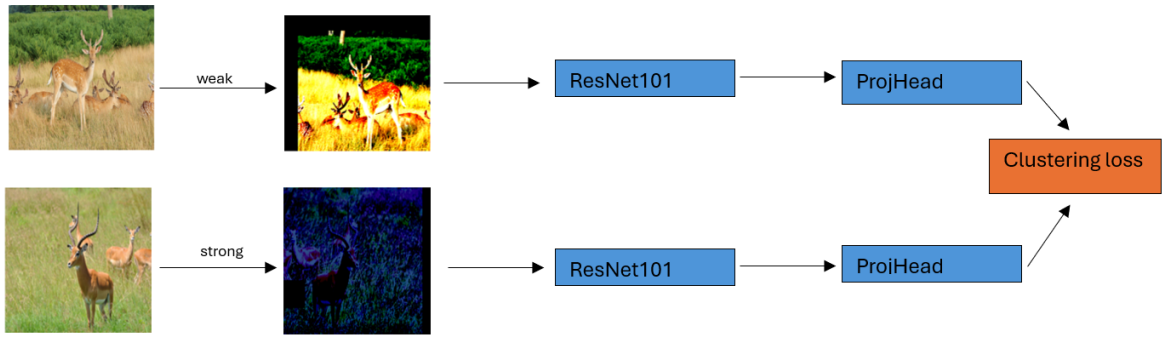
Συνάρτηση Απώλειας Ο στόχος είναι να διδάξουμε μια συνάρτηση ομαδοποίησης g_θ, η οποία περιγράφεται από ένα νευρωνικό δίκτυο με βάρη θ, ώστε να ταξινομεί ένα δείγμα x_i και τους εξορυγμένους γείτονές του N_{x_i} μαζί [40]. Το g_θ καταλήγει σε μια συνάρτηση softmax για να εκτελέσει μια ανάθεση στις συστάδες και επομένως $g_\theta(x) \in [0,1]^{|\mathcal{S}\cup\mathcal{U}|}$. Η πιθανότητα το δείγμα x_i να ανατεθεί στη συστάδα k συμβολίζεται ως $g_\theta^k(x_i)$. Για να μάθουμε τα βάρη του g_θ, ελαχιστοποιείται

το ακόλουθο κόστος [40]:

$$\mathcal{L}_u = -\frac{1}{|D|}\sum_{x\in D}\sum_{y\in N_x}\log\left(g_\theta(x)^T\cdot g_\theta(y)\right) + \lambda \sum_{k\in\{1,\dots,|\mathcal{S}\cup\mathcal{U}|\}} g_\theta^{'k}\log g_\theta^{'k}$$
$$\text{όπου } g_\theta^{'k} = \frac{1}{|D|}\sum_{x\in D} g_\theta^k(x) \tag{1.20}$$

Είναι εύκολο να δει κανείς ότι ο πρώτος όρος στην εξίσωση 5.2 ελαχιστοποιείται, αν όλα τα δείγματα $x \in D$ ανατεθούν στην ίδια συστάδα με τους γείτονές τους N_x, με την υψηλότερη δυνατή πιθανότητα. Επομένως, ο πρώτος όρος στην εξίσωση 5.2 μπορεί να θεωρηθεί ως ένας *όρος συνέπειας*. Ο δεύτερος όρος μπορεί να θεωρηθεί ως η αρνητική εντροπία της τυχαίας μεταβλητής $g_\theta^{'k}$. Είναι γνωστό ότι η εντροπία μιας τυχαίας μεταβλητής μεγιστοποιείται από την ομοιόμορφη κατανομή. Επομένως, ο δεύτερος όρος στην εξίσωση 5.2 αποτρέπει το μοντέλο από το να καταρρεύσει, αναθέτοντας όλα τα δείγματα $x \in D$ σε λίγες συστάδες.

Μια σχηματική αναπαράσταση του προαναφερθέντος αλγορίθμου φαίνεται στο Σχήμα 1.8.



Σχήμα 1.8: Απεικόνιση του βήματος ομαδοποίησης του αλγορίθμου μας. Αφού επιλεγεί ένα σύνολο γειτόνων για κάθε εικόνα στο σύνολο δεδομένων, επιλέγεται μια τυχαία εικόνα και ένας από τους γείτονές της. Η αρχική εικόνα υφίσταται ασθενή παραλλαγή (augmentation), ενώ ο γείτονάς της υφίσταται ισχυρή παραλλαγή (augmentation). Και οι δύο εικόνες στη συνέχεια περνούν από την προεκπαιδευμένη ραχοκοκαλιά ResNet101, η οποία παραμένει παγωμένη (frozen) κατά τη διάρκεια της εκπαίδευσης. Οι προκύπτουσες προβολές περνούν στη συνέχεια στο δίκτυο συσταδοποίησης για να υπολογιστεί η απώλεια ομαδοποίησης.

1.4.2 Επιλογή Αντιπροσώπων

Μετά την εκπαίδευση του δικτύου g_θ, ο στόχος μας είναι να επιλέξουμε ένα σημείο στον χώρο προβολής που προκύπτει από το g_θ, για κάθε κατηγορία. Για τον σκοπό αυτό, απορρίπτουμε το τελευταίο πλήρως συνδεδεμένο στρώμα του δικτύου g_θ. Ας ονομάσουμε το υπόλοιπο μέρος ως g_ϕ. Το πρώτο μας βήμα είναι να υπολογίσουμε τις προβολές των δειγμάτων των ορατών κατηγοριών $E^s = \{g_\phi(x_1^s),\dots,g_\phi(x_{N_s}^s)\}$ και των δειγμάτων των αόρατων κατηγοριών $E^u = \{g_\phi(x_1^u),\dots,g_\phi(x_{N_u}^u)\}$. Για κάθε ορατή κατηγορία, επιλέγουμε τη μέση τιμή της ως τον αντίστοιχο αντιπρόσωπο, δηλαδή για κάθε $k \in \mathcal{S}$,

$$\mu_k = \frac{1}{|X_k|}\sum_{j:y_j^s=k} g_\phi(x_j^s), \qquad \text{όπου } X_k = \{x_i^s : y_i^s = k\} \tag{1.21}$$

Χρησιμοποιώντας την παραπάνω εξίσωση, μπορούμε να ορίσουμε το σύνολο $M^s = \{\mu_1, \ldots, \mu_S\}$, το οποίο θα ονομάσουμε *αντιπροσώπους των ορατών κατηγοριών*. Παρατηρούμε ότι $|M^s| = |\mathcal{S}|$. Δεδομένου ότι οι ετικέτες του συνόλου D_u δεν είναι γνωστές, δεν μπορούμε να ακολουθήσουμε παρόμοιο τρόπο συλλογισμού. Επομένως, χρησιμοποιούμε τον αλγόριθμο K-means, αρχικοποιώντας τον με τον πραγματικό αριθμό των αόρατων κατηγοριών. Τα προκύπτοντα κέντρα τα ονομάζουμε $M^u = \{\mu_{S+1}, \ldots, \mu_{S+U}\}$. Θα αναφερόμαστε στο σύνολο M^u ως *αντιπροσώπους των αόρατων κατηγοριών*. Πάλι, παρατηρούμε ότι $|M^u| = |\mathcal{U}|$.

1.4.3 Εύρεση Μιας Αμφιμονοσήμαντης Αντιστοιχίσης

Ο τελικός μας στόχος είναι να προσδιορίσουμε την αντιστοιχία μεταξύ του συνόλου M^u και του συνόλου T^u (και συνεπώς του συνόλου $\mathcal{U}$), δεδομένης της αντιστοιχίας μεταξύ των συνόλων M^s και του συνόλου πρωτοτύπων των ορατών κατηγοριών T^s. Αυτό το πρόβλημα μπορεί να θεωρηθεί ως ένα ημι-επιβλεπόμενο πρόβλημα ευθυγράμμισης επιφανειών [52] και είναι καλά μελετημένο στον τομέα της Επεξεργασίας Φυσικής Γλώσσας, με τη μορφή της δίγλωσσης μετάφρασης λέξεων (bilingual word translation)[46], [45], [53]. Στην περίπτωσή μας, τα σύνολα M^s και T^s μπορούν να θεωρηθούν ως ευθυγραμμισμένες προβολές διανυσμάτων. Πιο συγκεκριμένα, αναζητούμε μια συνάρτηση f_θ, η οποία αντιστοιχεί κάθε στοιχείο του συνόλου πρωτοτύπων των αόρατων κατηγοριών T^u σε ένα μοναδικό στοιχείο των αντιπροσώπων των αόρατων κατηγοριών M^u, δεδομένου ότι κάθε στοιχείο του συνόλου πρωτοτύπων των ορατών κατηγοριών T^s πρέπει να απεικονίζεται στον αντίστοιχο αντιπρόσωπο της ορατής κατηγορίας του συνόλου M^s.

Για να μάθουμε τις παραμέτρους της συνάρτησης f_θ ακολουθούμε την προσέγγιση των [46] χρησιμοποιώντας μη επιβλεπόμενη αντιστοίχιση κατανομών και ευθυγράμμιση γνωστών ζευγών λέξεων.

Μη Επιβλεπόμενη Αντιστοίχιση Κατανομών Δεδομένων των συνόλων $M^s \cup M^u$ και $T^s \cup T^u$, η απώλεια $\mathcal{L}_{\theta|D}$ στοχεύει να αντιστοιχίσει την κατανομή και των δύο χώρων. Συγκεκριμένα, οι παράμετροι θ του f_θ εκπαιδεύονται ώστε να εξαπατούν έναν διακριτή D, ο οποίος με τη σειρά του εκπαιδεύεται να διαφοροποιεί μεταξύ των αντιπροσώπων $M^s \cup M^u$ και των απεικονίσεων πρωτοτύπων $f_\theta(T^s \cup T^u) = \{f_\theta(t_1^s), \ldots, f_\theta(t_S^s), f_\theta(t_{S+1}^u), \ldots, f_\theta(t_{S+U}^u)\}$. f_θ. Οι f_θ και D βελτιστοποιούνται εναλλάξ χρησιμοποιώντας τα παρακάτω κριτήρια:

$$\begin{aligned} \mathcal{L}_{D|\theta} &= -\frac{1}{S+U} \sum_{x \in T^s \cup T^u} \log(1 - D(f_\theta(x))) - \frac{1}{S+U} \sum_{x \in M^s \cup M^u} \log D(x) \\ \mathcal{L}_{\theta|D} &= -\frac{1}{S+U} \sum_{x \in T^s \cup T^u} \log D(f_\theta(x)) \end{aligned} \tag{1.22}$$

Ευθυγράμμιση γνωστών αντιπροσώπων με γνωστά πρωτότυπα Δεδομένων των συνόλων M^s και T^s, η συνάρτηση f_θ θα πρέπει να προβάλλει τα στοιχεία του T^s όσο το δυνατόν πιο «κοντά» στους αντίστοιχους ορατούς αντιπροσώπους. Αυτό μπορεί να διατυπωθεί ως εξής:

$$\mathcal{L}_{\theta|alig} = -\frac{1}{S} \sum_{i=1}^{S} \frac{f_\theta(t_i)^T \cdot \mu_i}{||f_\theta(t_i)||_2 \cdot ||\mu_i||_2} \tag{1.23}$$

Η τελική συνάρτηση απώλειας για τη συνάρτηση αντιστοίχισης f_θ διατυπώνεται ως εξής:

$$\mathcal{L} = \mathcal{L}_{\theta|D} + \mathcal{L}_{\theta|alig} + \mathcal{L}_{D|\theta} \tag{1.24}$$

1.4.4 Επαναληπτική Βελτίωση της Αντιστοίχισης

Μια κοινή μέθοδος βελτίωσης της συνάρτησης f_θ είναι η επαναληπτική επέκταση των αντιστοιχιών μεταξύ των συνόλων M^s και T^s και στη συνέχεια η βελτίωση της συνάρτησης προβολλής f_θ ως βήμα μετα-επεξεργασίας [46], [45]. Αυτή η διαδικασία βελτίωσης αρχικά βρίσκει το ζεύγος σημείων στα σύνολα M^u και T^u που τοποθετούνται πιο κοντά από τη συνάρτηση f_θ, και στη συνέχεια ενημερώνει τη συνάρτηση f_θ λαμβάνοντας υπόψη τόσο τις αντιστοιχίες του T^s με το M^s όσο και το νέο ζεύγος σημείων.

Για τον σκοπό αυτό, το διάνυσμα παραμέτρων που αποτυπώθηκε στην παραπάνω ενότητα ορίζεται ως θ_0. Ομοίως, τα σύνολα M^u και T^u μπορούν να θεωρηθούν ως αρχικά σύνολα χωρίς αντιστοιχίες και συνεπώς θα τα ορίσουμε ως T_0^u και M_0^u. Για να ποσοτικοποιήσουμε την ομοιότητα μεταξύ των σημείων στα M_0^u και T_0^u, θεωρούμε το σύνολο $C_0 = M_0^u \times f_{\theta_0}(T_0^u) \subseteq \mathbb{R}^D \times \mathbb{R}^D$, όπου D είναι η διάσταση εξόδου του νευρωνικού δικτύου g_ϕ (βλ. ενότητα 5.3). Ας θεωρήσουμε ότι η συνάρτηση $sim(\cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ είναι μια συνάρτηση ομοιότητας με:

$$sim(x, y) = \frac{x^T y}{||x||_2 ||y||_2}$$

Δεδομένου του παραπάνω μέτρου ομοιότητας, ας επιλέξουμε

$$(x_1, y_1) = \underset{(x,y)\in M_0^u \times T_0^u}{\arg\max} \; sim(C_0) \tag{1.25}$$

Το επόμενο βήμα μας είναι να ορίσουμε τα «νέα» σύνολα $M_1^s = M_0^s \cup \{x_1\}$, $T_1^s = T_0^s \cup \{y_1\}$ και $M_1^u = M_0^u \setminus \{x_1\}$, $T_1^u = T_0^u \setminus \{y_1\}$. Στη συνέχεια βελτιώνουμε τη συνάρτηση f_θ, δεδομένων των νέων συνόλων M_1^s και T_1^s. Πιο συγκεκριμένα, αποθηκεύουμε τα διανύσματα που βρέθηκαν στα σύνολα M_1^s και $f_\theta(T_1^s)$, στους πίνακες X_1 και Y_1 ως στήλες αντίστοιχα. Οι νέες παράμετροι θ_1 βρίσκονται λύνοντας το πρόβλημα βελτιστοποίησης

$$\theta_1 = \arg\max_\theta \mathrm{Tr}(X_1^T Y_1) \tag{1.26}$$

όπου

$$\begin{aligned} X_1 &= \begin{pmatrix} \frac{\mu_1}{||\mu_1||_2} & \cdots & \frac{\mu_S}{||\mu_S||_2} & \frac{x_1}{||x_1||_2} \end{pmatrix} \in \mathbb{R}^{D\times(S+1)} \\ Y_1 &= \begin{pmatrix} \frac{f_\theta(t_1^s)}{||f_\theta(t_1^s)||} & \cdots & \frac{f_\theta(t_S^s)}{||f_\theta(t_S^s)||_2} & \frac{f_\theta(y_1)}{||f_\theta(y_1)||_2} \end{pmatrix} \in \mathbb{R}^{D\times(S+1)} \end{aligned} \tag{1.27}$$

Κατά συνέπεια, τα σύνολα M_1^u και T_1^u χρησιμοποιούνται για να ταιριάξουν ένα ζεύγος σημείων από τους αντιπροσώπους των αόρατων κατηγοριών και τα πρωτότυπα των αόρατων κατηγοριών, χρησιμοποιώντας το f_{θ_1}. Αυτή η διαδικασία συνεχίζεται επαγωγικά μέχρι να υπάρξει ένα $n \in \mathbb{N}$, τέτοιο που $M_n^u = T_n^u = \emptyset$. Δεδομένου ότι $|M_0^u| = |T_0^u| = U$, είναι εύκολο να δει κανείς ότι $n = U$. Επομένως, το τελικό διάνυσμα παραμέτρων που επιλέγεται για τη χαρτογράφηση f_θ είναι θ_U.

1.4.5 Ταξινόμηση

Μετά το βήμα επαναληπτικής βελτίωσης της χαρτογράφησης, έχουμε μια μαθημένη παράμετρο θ_U. Για να ταξινομήσουμε μια εικόνα $x \in D_u$, πρώτα την προβάλλουμε στον οπτικό χώρο που παράγεται από το εκπαιδευμένο νευρωνικό δίκτυο g_ϕ και στη συνέχεια βρίσκουμε το πλησιέστερο προβεβλημένο πρωτότυπο κατηγορίας. Με άλλα λόγια, η προβλεπόμενη κατηγορία $c^\star$ δίνεται από

$$c^\star = \underset{c\in\mathcal{S}\cup\mathcal{U}}{\arg\max} \left\{ \frac{f_{\theta_U}(\pi(c))^T \cdot g_\phi(x)}{||f_{\theta_U}(\pi(c))||_2 ||g_\phi(x)||_2} \right\} \tag{1.28}$$

όπου $\pi(\cdot) : \mathcal{S} \cup \mathcal{U} \to \mathcal{T}$ είναι μια συνάρτηση πρωτοτύπων κλάσης.

1.5 Πειραματικά Αποτελέσματα

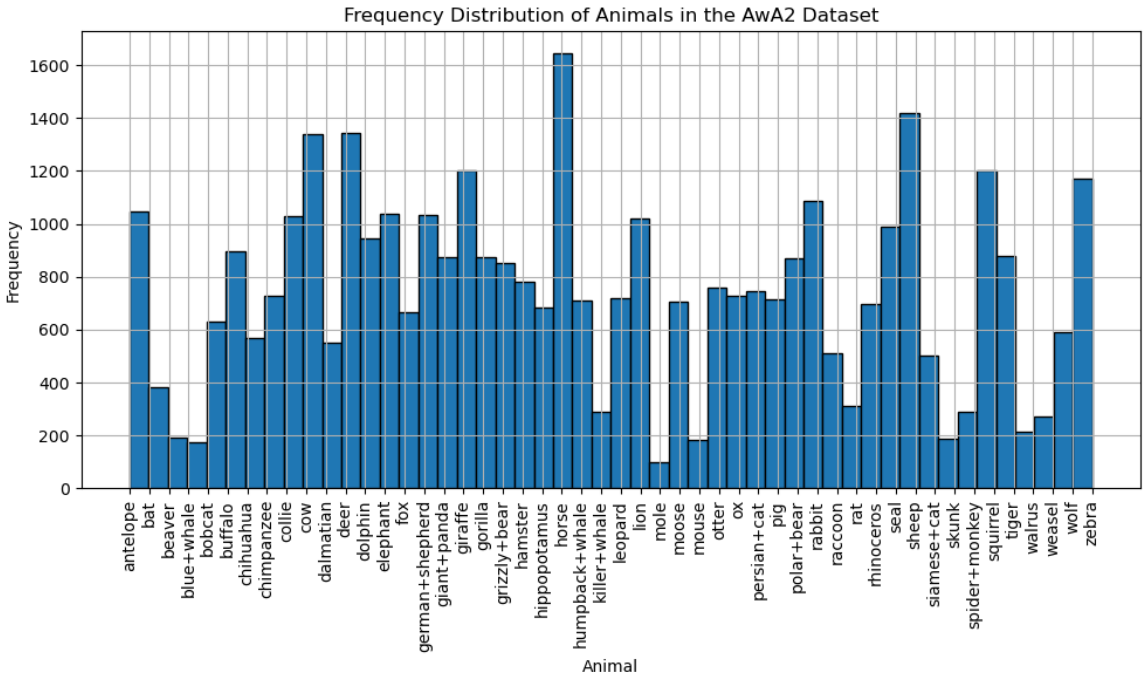
1.5.1 Σύνολα Δεδομένων

Για να εξετάσουμε τη μέθοδό μας, διεξάγουμε πειράματα σε δύο ευρέως χρησιμοποιούμενα σύνολα δεδομένων στη ΜΧΠ: Πιο συγκεκριμένα, το AwA2 και [6] και το CUB [7]. Ακολουθώντας την ίδια δομή με άλλες μεθόδους, υιοθετούμαι δύο διαφορετικές στρατηγικές διαχωρισμού δεδομένων:

- **Κανονικός Διαχωρισμός (Standard Split - SS)**: Ο κανονικός διαχωρισμός ορατών/αόρατων κατηγοριών προτάθηκε πρώτα στο [17] και στη συνέχεια χρησιμοποιήθηκε ευρέως στις περισσότερες εργασίες ΜΧΠ [54],[55],[56],[14].
- **Διαχωρισμός που προτάθηκε στο [18] (Proposed Split - PS)**: Αυτός ο διαχωρισμός προτάθηκε για να αφαιρέσει τις επικαλυπτόμενες κατηγορίες του συνόλου δεδομένων ImageNet-1K, δεδομένου ότι χρησιμοποιείται για την προεκπαίδευση του μοντέλου ΣΝΝ, ResNet101.

Σύνολο Δεδομένων "Animals with Attributes 2"

Το σύνολο δεδομένων "Animals with Attributes 2" (AWA2) [6] αποτελείται από 37,322 εικόνες από 50 κατηγορίες ζώων, όπου οι 40 κατηγορίες είναι ορατές κατά την εκπαίδευση, ενώ οι υπόλοιπες 10 κατηγορίες είναι αόρατες κατά την εκπαίδευση. Κάθε κατηγορία σχετίζεται με ένα συνεχές διάνυσμα χαρακτηριστικών 85 διαστάσεων.



Σχήμα 1.9: Κατανομή συχνότητας των κατηγοριών στο σύνολο δεδομένων AwA2 [6]

Είναι εμφανές από το σχήμα 1.9 ότι η κατανομή των κατηγοριών εντός του συνόλου δεδομένων είναι μη-ισορροπημένη, γεγονός που αποτελεί σημαντική πρόκληση για την ημιεπαγωγική μάθηση χωρίς παραδείγματα (ΜΧΠ). Αναλυτικότερα, ένα μη-ισορροπημένο σύνολο δεδομένων μπορεί

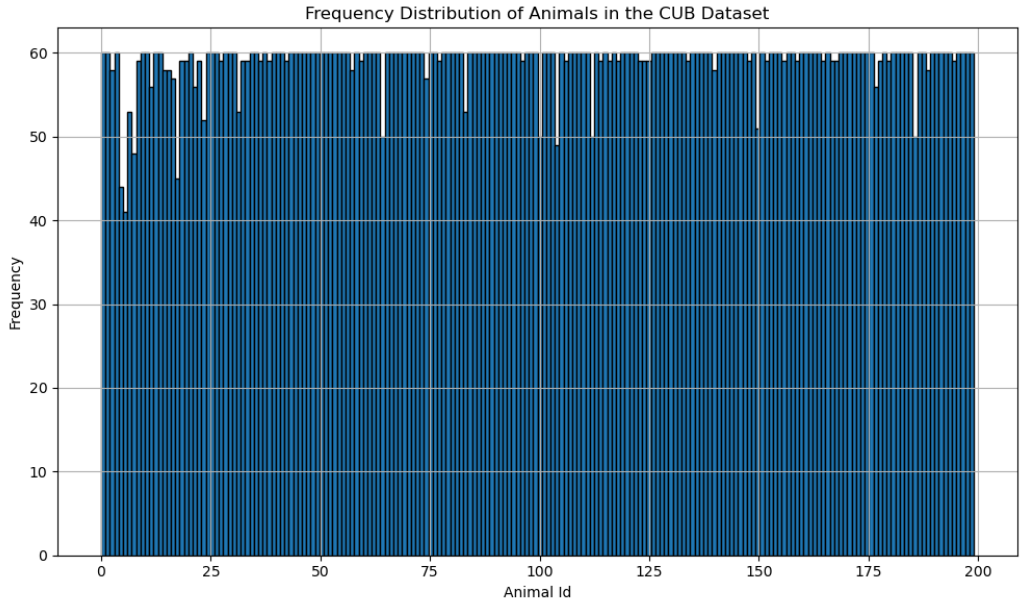
να οδηγήσει σε μεροληπτικές προβλέψεις του μοντέλου. Οι κατηγορίες με λιγότερα παραδείγματα ενδέχεται να μην εκπροσωπούνται επαρκώς κατά την εκπαίδευση, με αποτέλεσμα την χειρότερη απόδοση ως προς αυτές τις κλάσεις κατά τη δοκιμή. Γενικά, η μη-ισορροπημένη κατανομή κατηγοριών εισάγει πολυπλοκότητες τόσο στη διαδικασία μάθησης όσο και στις δυνατότητες γενίκευσης των μοντέλων ΜΧΠ.

Σύνολο Δεδομένων CUB

Το σύνολο δεδομένων Caltech-SCSD Birds-200-2011 (CUB) περιέχει 11,788 εικόνες από 200 είδη πτηνών. Ένας κλασσικός διαχωρισμός διαιρεί αυτά τα είδη πτηνών σε 150 ορατές κατηγορίες και 50 αόρατες κατηγορίες. Για κάθε κατηγορία, παρέχεται ένα διάνυσμα 312 διαστάσεων.

Αν και το ιστόγραμμα του συνόλου δεδομένων (σχήμα 1.10) υποδεικνύει ισορροπημένη κατανομή των κατηγοριών, είναι σημαντικό να σημειωθεί ότι κάθε κατηγορία περιλαμβάνει το πολύ 60 δείγματα. Με τόσο περιορισμένα δεδομένα ανά κατηγορία, η ικανότητα του μοντέλου να γενικευτεί αποτελεσματικά σε αόρατες κατηγορίες γίνεται πιο δύσκολη, καθώς μπορεί να δυσκολευτεί να συλλάβει την πλήρη μεταβλητότητα και πολυπλοκότητα κάθε κατηγορίας.

Επιπλέον, η λεπτομερής φύση του συνόλου δεδομένων εντείνει περαιτέρω τη δυσκολία της εργασίας. Όπως φαίνεται στο σχήμα 6.3, οι κατηγορίες είναι οπτικά παρόμοιες, απαιτώντας από το μοντέλο να διακρίνει λεπτές διαφορές μεταξύ τους. Αυτό αποτελεί ένα σημαντικό σημείο που θα συζητηθεί περαιτέρω στα επόμενα κεφάλαια.



Σχήμα 1.10: Κατανομή συχνότητας των κατηγοριών στο σύνολο δεδομένων CUB [7]

Λεπτομέρειες Υλοποίησης

Για το πρώτο βήμα της μεθόδου μας, επιλέξαμε να χρησιμοποιήσουμε 20 γείτονες, όπως στις περισσότερες περιπτώσεις που μελετήθηκαν στο [40]. Η επίδραση της μεταβολής του αριθμού των γειτόνων θα εξεταστεί λεπτομερώς στην Ενότητα 6.4. Οι υλοποιήσεις των νευρωνικών δικτύων που συζητήθηκαν στο προηγούμενο κεφάλαιο συνοψίζονται στον Πίνακα 1.1. Όλα τα νευρωνικά δίκτυα εκπαιδεύτηκαν χρησιμοποιώντας τον βελτιστοποιητή Adam [57] με ρυθμό μάθησης 10^{-4}. Ο χώρος προβολής του g_ϕ, όπου πραγματοποιείται η ταξινόμηση, είναι 2048 διαστάσεων, όσο είναι και η

διάσταση εξόδου του προεκπαιδευμένου μοντέλου ResNet101. Το βήμα ομαδοποίησης της μεθόδου μας αξιοποιεί δύο είδη ενισχύσεων (augmentations): 'ασθενής' και 'ισχυρή'. Ακολουθώντας το [58], η ασθενής ενίσχυση είναι μια τυπική ενίσχυση flip-and-shift. Πιο συγκεκριμένα, αναστρέφουμε τυχαία τις εικόνες οριζόντια με πιθανότητα 50% και τις μετατοπίζουμε τυχαία έως 12.5% κάθετα και οριζόντια. Για τις 'ισχυρές' ενισχύσεις επιλέγεται ο RandAugment [59]. Αυτός ο αλγόριθμος επιλέγει τυχαία μετασχηματισμούς για κάθε εικόνα, χρησιμοποιώντας μια μεταβλητή που ελέγχει το μέγεθος όλων των παραμορφώσεων και βρίσκεται σε ένα προκαθορισμένο εύρος. Η παράμετρος λ που ζυγίζει τον όρο εντροπίας στην εξίσωση 5.2 τίθεται 2,5. Τόσο τα διανύσματα που βρίσκονται στον σημασιολογικό χώρο όσο και στον οπτικό χώρο (ο χώρος προβολής του g_ϕ) κανονικοποιούνται ώστε να έχουν νόρμα 1, πριν το βήμα ευθυγράμμισης.

Function	Implementation
g_θ	FC + ReLu + FC + ReLu + FC + ReLu + FC + Softmax
f_θ	FC + ReLu + FC + ReLu
D	FC + ReLu + FC + Softmax

Πίνακας 1.1: Λεπτομέρειες υλοποίησης των τριών νευρωνικών δικτύων που συζητήθηκαν στο κεφάλαιο 1.4. FC σημαίνει ότι έχουμε ένα πλήρως συδεδεμένο επίπεδο ενώ ReLu είναι η γνωστή συνάρτηση ενεργοποίησης.

1.5.2 Αποτελέσματα ταξινόμησης

Στις επόμενες δύο ενότητες, παρουσιάζουμε τα τελικά αποτελέσματά μας τόσο στο τις συμβατικό, όσο και στο γενικευμένο σενάριο Μάθησης Χωρίς Παραδείγματα. Στο συμβατικό σενάριο ΜΧΠ, υποθέτουμε ότι όλα τα τεστ δείγματα ανήκουν στις αόρατες κατηγορίες, ενώ στο σενάριο Γενικευμένης ΜΧΠ, υποθέτουμε ότι τα δείγματα τεστ ανήκουν τόσο στις ορατές όσο και στις αόρατες κατηγορίες. Τα αποτελέσματα για το συμβατικό σενάριο αναφέρονται υπό τον κανονικό διαχωρισμό (SS) [17] και τα αποτελέσματα για τη γενικευμένη υπό τον προτεινόμενο διαχωρισμό (PS) [18].

Η Μέση Ακρίβεια Κατηγορίας (MCA) είναι η δημοφιλής μετρική αξιολόγησης στη ΜΧΠ [37]. Στο συμβατικό σενάριο, η MCA υπολογίζεται μόνο στα τεστ δεδομένων των αόρατων κατηγοριών ($Y = \mathcal{U}$).

$$MCA = \frac{1}{|\mathcal{U}|} \sum_{y \in \mathcal{U}} acc_y \tag{1.29}$$

όπου acc_y είναι η top-1 ακρίβεια της κατηγορίας y από τα τεστ δεδομένα των αόρατων κατηγοριών. Στο γενικευμένο σενάριο, ο χώρος των ετικετών προέρχεται από την ένωση των ορατών και αόρατων κατηγοριών ($Y = \mathcal{S} \cup \mathcal{U}$) [37].

$$\mathcal{H} = \frac{2 * MCA_{ts} * MCA_{tu}}{MCA_{ts} + MCA_{tu}} \tag{1.30}$$

όπου MCA_{ts} είναι η μέση ακρίβεια κατηγορίας των τεστ δεδομένων των ορατών κατηγοριών, MCA_{tu} είναι η μέση ακρίβεια κατηγορίας των τεστ δεδομένων των αόρατων κατηγοριών και ($\mathcal{H}$) είναι ο αρμονικός τους μέσος όρος.

Αποτελέσματα υπό το κλασσικό σενάριο

Method		AwA2	CUB
I	CONSE [60]	45.6	34.3
	DAP [61]	44.1	40.0
	SSE [62]	60.1	43.9
T	MFMR [63]	-	47.5
	SE-ZSL [64]	68.3	54.1
	ALE_trans [65]	70.7	54.5
	GFZSL (G) [66]	78.6	50.0
	TEDE [16]	77.5	67.8
	Bi-VAEGAN (G) [67]	95.8	76.8
	ICPC (F) [37]	96.1	84.1
T	Ours	95.7	48.0

Πίνακας 1.2: Σύγκριση απόδοσης ΜΧΠ διαφορετικών μεθόδων στα σύνολα δεδομένων AwA2 και CUB , στο συμβατικό σενάριο ΜΧΠ, χρησιμοποιώντας τον κανονικό διαχωρισμό [17]. (I) υποδεικνύει ότι μια μέθοδος είναι επαγωγική, ενώ (T) υποδεικνύει ότι μια μέθοδος είναι ημιεπαγωγική. (F) υποδεικνύει ότι μια μέθοδος εκτελεί βελτίωση (fine-tuning) της ραχοκοκαλιάς ResNet101 και συνεπώς η σύγκριση δεν είναι δίκαιη. (G) υποδεικνύει ότι μια μέθοδος είναι γενετική. (-) σημαίνει ότι τα αντίστοιχα αποτελέσματα δεν αναφέρθηκαν.

Ο Πίνακας 1.2 παρέχει μια συγκριτική ανάλυση διαφόρων μεθόδων ΜΧΠ στα σύνολα δεδομένων AwA2 και CUB στο συμβατικό σενάριο ΜΧΠ. Οι μέθοδοι που παρατίθενται περιλαμβάνουν επαγωγικές (I) και ημιεπαγωγικές (T) προσεγγίσεις, με τη μέθοδο μας να ανήκει στις ημιεπαγωγικές. Σημειώνουμε είναι ότι ο πίνακας περιλαμβάνει αποκλειστικά μεθόδους βασισμένες σε προβολές (embedding-based methods), εκτός από τις μεθόδους Bi-VAEGAN [68] και GFZSL [66], που είναι κορυφαίες γενετικές προσεγγίσεις. Η μέθοδος μας επιδεικνύει ανταγωνιστική απόδοση, επιτυγχάνοντας 95.7% στο σύνολο δεδομένων AwA2 και 48.0% στο σύνολο δεδομένων CUB, όπως αναφέρθηκε σε πάνω από 50 δοκιμές για να εξασφαλιστεί η συνέπεια των αποτελεσμάτων.

Σε σύγκριση με τις άλλες μεθόδους, η προσέγγισή μας υπερέχει έναντι αρκετών επαγωγικών μεθόδων και είναι συγκρίσιμη με τις μεθόδους ICPC [37] και Bi-VAEGAN [68], οι οποίες είναι αλγόριθμοι αιχμής (SOTA) στην Μάθηση Χωρίς Παραδείγματα.

Αποτελέσματα υπό το Γενικευμένο Σενάριο

Ο Πίνακας 1.3 παρουσιάζει την απόδοση διαφόρων μεθόδων ΜΧΠ στα σύνολα δεδομένων AwA2 και CUB στο γενικευμένο σενάριο ΜΧΠ. Οι μέθοδοι είναι είτε επαγωγικές (I) είτε ημιεπαγωγικές (T) και κατηγοριοποιούνται ως βασισμένες σε προβολές (E) , όπως η δική μας, ή γενετικές (G). Τα αποτελέσματα της μεθόδου μας είναι ο μέσος όρος 50 εκτελέσεων. Στο σύνολο δεδομένων AwA2 , η μέθοδός μας επιτυγχάνει 89.8% για τις ορατές κατηγορίες (MCA_{ts}), 73.2% για τις αόρατες κατηγορίες (MCA_{tu}) και έναν αρμονικό μέσο όρο (H) 80.7%. Στο σύνολο δεδομένων CUB , η μέθοδός μας επιτυγχάνει 51.6% για MCA_{ts}, 46.7% για MCA_{tu} και έναν αρμονικό μέσο όρο 49.0%. Είναι αξιοσημείωτο ότι η μέθοδός μας ξεπερνά όλες τις επαγωγικές μεθόδους, κάτι που είναι αναμενόμενο, αφού έχουμε πρόσβαση σε περισσότερες πληροφορίες (μη επισημασμένα δείγματα από τις αόρατες κατηγορίες).

Method		AwA2			CUB		
		MCA_{ts}	MCA_{tu}	$\mathcal{H}$	MCA_{ts}	MCA_{tu}	$\mathcal{H}$
I	F-CLSWGAN (G) [69]	81.8	14.0	23.9	33.1	21.8	26.3
	SP-AEN (G) [70]	90.9	23.3	37.1	38.6	24.9	30.3
	DEM (E) [71]	86.4	30.5	45.1	25.6	34.3	20.5
	ALE (E) [72]	68.9	52.1	59.4	36.6	42.6	39.4
	LisGAN (G) [73]	-	-	-	37.8	42.9	40.2
T	DSRL (E) [74]	-	-	-	25.0	17.7	20.7
	GFZSL (G) [66]	-	-	-	45.8	24.9	32.2
	ALE_trans (E) [65]	73.0	12.6	21.5	45.1	23.5	30.9
	PREN (E) [75]	88.6	32.4	47.4	55.8	35.2	43.1
	f-VAEGAN (G) [42]	88.6	84.8	86.7	65.1	61.4	63.2
	Bi-VAEGAN (G) [68]	91.0	90.0	90.4	71.7	71.2	71.5
T	Ours (E)	89.8	73.2	80.7	51.6	46.7	49.0

Πίνακας 1.3: Σύγκριση απόδοσης ΜΧΠ διαφορετικών μεθόδων στα σύνολα δεδομένων AwA2 και CUB, στο γενικευμένο σενάριο ΜΧΠ, χρησιμοποιώντας τον προτεινόμενο διαχωρισμό [18]. (I) υποδεικνύει ότι μια μέθοδος είναι επαγωγική, ενώ (T) υποδεικνύει ότι μια μέθοδος είναι ημιεπαγωγική. (F) υποδεικνύει ότι μια μέθοδος εκτελεί βελτίωση (fine-tuning) της ραχοκοκαλιάς ResNet101. (G) υποδεικνύει ότι μια μέθοδος είναι γενετική. (-) σημαίνει ότι τα αντίστοιχα αποτελέσματα δεν αναφέρθηκαν.

Τέλος παρατηρούμε ότι έχουμε μείωση της μετρικής MCA όσο αφορά στις αόρατες κλάσεις στο γενικευμένο σενάριο. Αυτό είναι πλήρως αναμενόμενο, λόγω του προβλήματος bias (δείτε το κεφάλαιο 3.2.6).

Περιορισμοί

Η μέθοδος μας βασίζεται σε μεγάλο βαθμό στον αρχικό αριθμό σωστών γειτόνων που επιλέγονται στον αρχικό χώρο του ResNet101. Αυτό συμβαίνει επειδή το δίκτυο ομαδοποίησης προσπαθεί να αναθέσει ένα δείγμα και τους επιλεγμένους γείτονές του στην ίδια συστάδα. Όπως παρατηρείται στο Σχήμα 6.9, στο σύνολο δεδομένων CUB, περισσότεροι από τους μισούς γείτονες που επιλέχθηκαν ανήκουν σε διαφορετικές κατηγορίες, γεγονός που προφανώς θα οδηγήσει σε μη βέλτιστα αποτελέσματα. Για την επίλυση αυτού του προβλήματος, πειραματιστήκαμε με αλγορίθμους προεκπαίδευσης, όπως παραλλαγές των αλγορίθμων SimCLR [76] και MOCO [77]. Ωστόσο, αυτές οι μέθοδοι απαιτούν μεγάλες παρτίδες (batch size), που υπερέβησαν τους υπολογιστικούς μας πόρους, με αποτέλεσμα την αδυναμία βελτίωσης του αριθμού των σωστών γειτόνων. Επιπλέον, εξερευνήσαμε άλλες προσεγγίσεις ομαδοποίησης από τη βιβλιογραφία της ημι-επιβλεπόμενης συσταδοποίησης (semi-supervised clustering)[78], [79], [80]. Τα αποτελέσματά τους ήταν πολύ κατώτερα από την τεχνική ομαδοποίησης που εφαρμόσαμε, επιδεικνύοντας σημαντικά χαμηλότερους δείκτες ACC και NMI.

Συζήτηση

Σε αυτή τη μελέτη, επιτύχαμε αξιοσημείωτα αποτελέσματα στο σύνολο δεδομένων AwA2, ανταγωνιζόμενοι αποτελεσματικά τη μέθοδο Bi-VAEGAN [68], μια μέθοδο αιχμής (SOTA) γενετικής

μάθησης, τόσο στο συμβατικό όσο και στο γενικευμένο σενάριο. Συγκεκριμένα, στη συμβατικό σενάριο, η μέθοδός μας υπολείπονταν της Bi-VAEGAN [68] κατά μόλις 0.1%. Ωστόσο, τα αποτελέσματά μας στο σύνολο δεδομένων CUB ήταν σχετικά χαμηλότερα σε σύγκριση με άλλες SOTA μεθόδους. Αυτό αποδίδεται στον αρχικό χώρο του ResNet101 , όπου επιλέξαμε τους πλησιέστερους γείτονες για κάθε μη επισηεπισημειωμένο δείγμα. Το ιστόγραμμα 6.8 αποκάλυψε ότι πολλά δείγματα έχουν μεγάλο αριθμό γειτόνων από διαφορετικές κατηγορίες, προκαλώντας προβλήματα ταξινόμησης. Αυτός ο περιορισμός είναι εγγενής καθώς χρησιμοποιούμε το ResNet101 ως τη ραχοκοκαλιά του μοντέλου μας.

Προσπάθειες για την προεκπαίδευση της ραχοκοκαλιάς χρησιμοποιώντας αλγορίθμους όπως οι SimCLR [76] και MOCO [77] ήταν ανεπιτυχείς λόγω περιορισμένων υπολογιστικών πόρων. Αυτοί οι αλγόριθμοι θα μπορούσαν να έχουν βελτιώσει την ομαδοποίηση των δειγμάτων από διαφορετικές κατηγορίες, οδηγώντας σε βελτιωμένη αρχικοποίηση του αλγορίθμου μας. Για να ενισχύσουμε περαιτέρω το επιχείρημά μας, μια ανάλυση ανώτατου ορίου μετά την ομαδοποίηση έδειξε ότι για το σύνολο δεδομένων CUB, το ανώτατο όριο ήταν 53.55%, ενώ το τελικό αποτέλεσμα ταξινόμησης μας ήταν 48.0%. Αυτό υποδηλώνει ότι καλύτερη ομαδοποίηση θα μπορούσε να οδηγήσει σε βελτιωμένα αποτελέσματα ταξινόμησης.

Chapter 2

Introduction

Contents

2.1 Artificial Intelligence

Artificial Intelligence is a branch of computer science that deals explicitly with the development of systems that can do tasks that, usually, humans have to execute[81]. It spans a broad spectrum of applications like learning from data, recognizing patterns, solving challenging problems, understanding and processing natural language, and deciphering visual inputs. At its heart, AI is the attempt to create algorithms and models that machines could use to perform behaviors such as reasoning, planning, or adaptation to novel situations in a fashion quite similar to how humans would go about performing a given cognitive task. The history of AI dates back to early computer science [82]. It has therefore undergone many changes over time due to advances in computational power, data volumes, availability, and theoretical understanding.

AI can be broadly divided into two basic types: narrow/weak AI and general/robust AI [83]. Weak AI is designed to address specific tasks as in speech and facial recognition [84], language translation [85], or interpreting. This includes virtual assistants, streaming services, recommendation algorithms, and image recognition software for medical diagnostics. As it is narrowly efficient, it has no general intelligence and cannot act further than the instructions given for a particular activity

On the other hand, general AI or strong AI pursues the goal of developing human-like cognitive abilities [83]. This means that an AI with general intelligence would understand, learn, and apply knowledge across a variety of tasks, showing creativity like humans. Although general AI is still a theoretical concept and a major research topic, pursuing this vision drives many advancements in the field [86]. Currently, much of the research in the AI domain focuses on narrowing the gap between narrow and general AI, exploring new models, architectures and training alternatives, which show promise in approximating more general forms of intelligence [87].

This first chapter goes further into what Artificial Intelligence (AI) is and why it is essential today. We will explain in straightforward terms what machine and deep learning are, the two main branches of AI which are directly related with this thesis. Then, we will introduce zero-shot learning, which is the main focus of our work. This chapter will prepare the reader for a much more detailed and focused discussion in the next chapter.

2.1.1 Brief History

The term "Artificial Intelligence" itself was first coined by John McCarthy in 1956 at the Dartmouth Conference, starting AI as a formal academic field. Early research mainly focused on problem-solving and symbolic methods, with the then-pioneering development of the Logic Theorist by Allen Newell and Herbert A. Simon, capable of proving mathematical theorems. AI developed considerably in the 1960s and 1970s with early machine learning algorithms and natural language processing systems. For example, there existed programs like ELIZA, which simulated human conversation, and SHRDLU, which manipulated objects in a virtual environment using natural language commands. However, by the mid-1970s, progress slowed, leading to the "AI Winter" due to unmet expectations and technical limitations.

The 1980s saw a resurgence, where the technology was made to use AI to address particular problems, an example being the expert systems built around MYCIN for medical diagnoses. Interest dimmed again by the late 1980s because these systems required vast amounts of manual effort. New momentum came in the 1990s and the early 2000s with better computational power, more data, and advanced algorithms underpinning AI. Among them was IBM's Deep Blue with

its defeat of chess champion Garry Kasparov in 1997.

In the 21st century, AI has become attached to everyday life, with applications like virtual assistants Siri and Alexa, autonomous vehicles, content suggestions and many more. Breakthroughs in deep learning, such as Google's AlphaGo defeating a world champion Go player in 2016 and the release of ChatGPT in 2022, highlight AI's potential.

Figure 2.1: World Chess Champion Garry Kasparov faces IBM's Deep Blue in their historic 1997 rematch, a pivotal moment in the advancement of artificial intelligence (Source: [8]).

2.2 Machine Learning

Machine learning (ML) is an essential subset of artificial intelligence (AI) that focuses on teaching machines to process and interpret data more effectively [19]. Unlike traditional programming, where machines follow explicit instructions, ML enables them to learn from data and make decisions based on the patterns they discover. In ML, a computer program improves its performance on tasks over time by gaining experience from the instances it has been exposed to. This approach is especially useful when dealing with large and complex datasets, where manual analysis would be challenging. As the availability of extensive datasets continues to expand, the demand for machine learning is increasing, and many industries are leveraging ML to extract valuable information from their data. Today ML is something humans interact with on a daily basis, often without realizing it, influencing choices from the products we browse to the movies we watch.

The different types and algorithms of machine learning are shown in Figure 2.3 and Table 2.1 and will be explained in more detail below. Classical machine learning is categorized based on how an algorithm improves its predictions, with four main methodologies: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning [2].

Method	Applications
Linear Regression	Fits a linear equation to observed data to predict continuous outcomes.
Decision Tree	Uses a tree-like model of decisions and their possible consequences to classify data into categories.
Support Vector Machine	Finds a hyperplane that best separates different classes in the feature space.
Naive Bayes	Applies Bayes' theorem with strong independence assumptions between features to classify data.
K-Nearest Neighbors	Classifies data based on the closest training examples in the feature space.
Gaussian Mixture Model	Represents data as a mixture of multiple Gaussian distributions to capture complex patterns.
Artificial Neural Network	Mimics the human brain by using layers of interconnected nodes to learn from data and make predictions.

Table 2.1: Summary of well known ML algorithms.

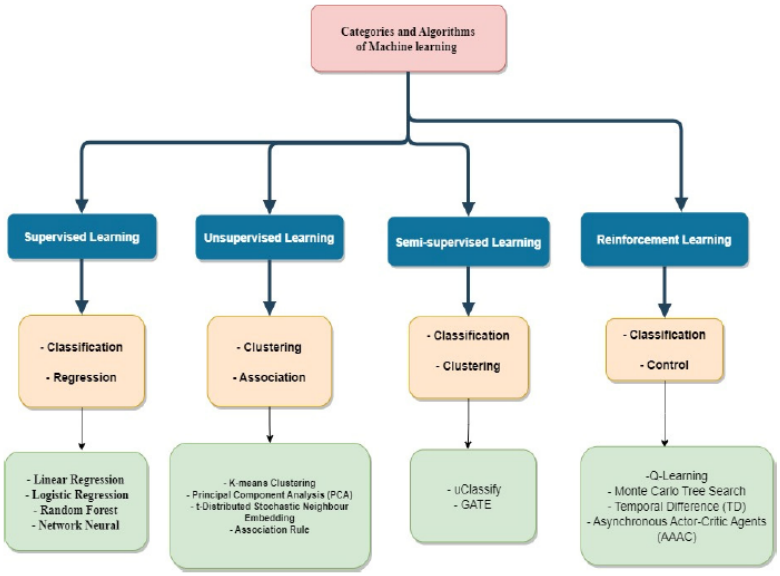


Figure 2.2: Different machine learning categories and algorithms (Source: [2]).

2.2.1 Supervised Learning

Supervised learning is a type of machine learning where the model is trained on a labeled dataset. In this approach, each training example consists of an input paired with the correct output, which serves as a guide for the learning process. The goal of supervised learning is to learn a mapping from inputs to outputs that can be used to make predictions on new, unseen instances. Common tasks in supervised learning include classification, where the model predicts discrete categories, and regression, where it predicts continuous values. Typically, these algorithms are trained in order to minimize some kind of error between the models outputs and the ground truth labels which are provided.

2.2.2 Unsupervised Learning

Unsupervised learning is a type of machine learning where the model is trained on data without labels. Unlike supervised learning, where each training example comes with an associated correct output, unsupervised learning algorithms must identify patterns and relationships in the data on their own. The primary goal is to explore the underlying structure of the data, for example through clustering or dimensionality reduction. In clustering, the algorithm groups similar data points together, while in dimensionality reduction, it simplifies the data by reducing the number of variables. The key difference between unsupervised and supervised learning is that unsupervised learning does not rely on labeled data, making it particularly useful for tasks where labeling is impractical or impossible.

2.2.3 Semi-Supervised Learning

Semi-supervised learning is a machine learning approach that combines both labeled and unlabeled data during training. This method leverages the small amount of labeled data to guide the learning process, while also utilizing the vast amounts of unlabeled data to improve the models performance. By incorporating unlabeled data, semi-supervised learning can often achieve better accuracy than purely supervised learning, especially when labeled data is scarce or expensive to obtain [58]. This approach is particularly useful in real-world scenarios where obtaining a fully labeled dataset is impractical, but there is an abundance of unlabeled data available.

2.2.4 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment to maximize cumulative rewards. Unlike supervised learning, where the algorithm is trained on labeled data, or unsupervised learning, where the algorithm discovers patterns in unlabeled data, RL learns through trial and error feedback. The agent takes actions in the environment, receives feedback in the form of rewards or penalties, and adjusts its behavior accordingly to maximize the long/short-term reward.

2.3 Deep Learning

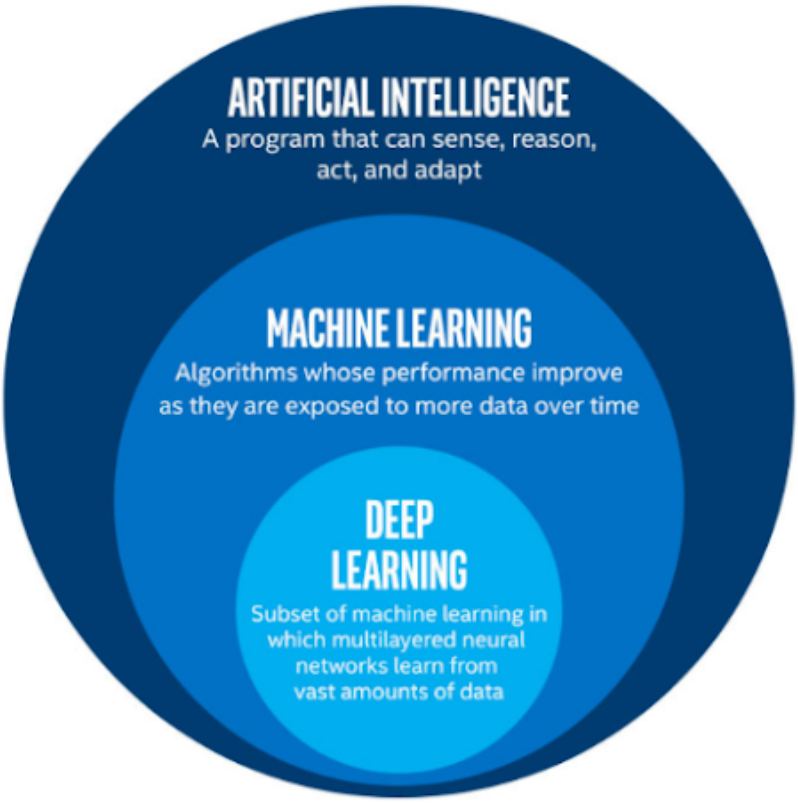


Figure 2.3: Deep Learning versus Machine Learning versus Artificial Intelligence

Historically, conventional machine learning methods were limited in their ability to handle raw data directly. Building a pattern recognition or machine learning system required careful engineering and significant domain expertise to design feature extractors that transformed raw data (like pixel values of an image) into an internal representation or feature vector suitable for the learning subsystem (for example a classifier).

Deep learning is a specialized area within machine learning that emphasizes the use of neural networks with numerous layers, commonly known as deep neural networks. A significant benefit of deep learning is its capacity for automatic representation learning through its hidden layers [9]. Deep neural networks construct hierarchical representations of data through multiple layers, with each layer capturing progressively abstract features [9]. This automatic feature extraction process minimizes the need for hand-crafted features and domain-specific knowledge, which makes the training of such models more straightforward. In a deep neural network, hidden layers act as a series of filters, refining data representations incrementally. For example, initial layers might identify basic patterns like edges in an image, while deeper layers can detect more complex structures such as object parts. This concept is illustrated in Figure 2.5.

This methodology is especially effective for managing large, complex datasets, commonly referred to as "big data", which are becoming more and more available in recent years (see Figure 2.4). The advent of big data has significantly enhanced the prospects for deep learning, as these algorithms excel at uncovering hidden patterns in extensive datasets . The combination of abundant data and advancements in computational power has enabled deep learning models

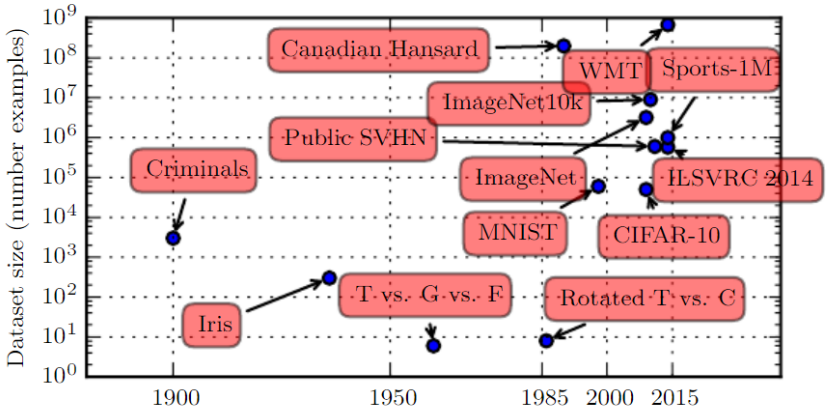


Figure 2.4: Increase of dataset size over the years (Source: [9]).

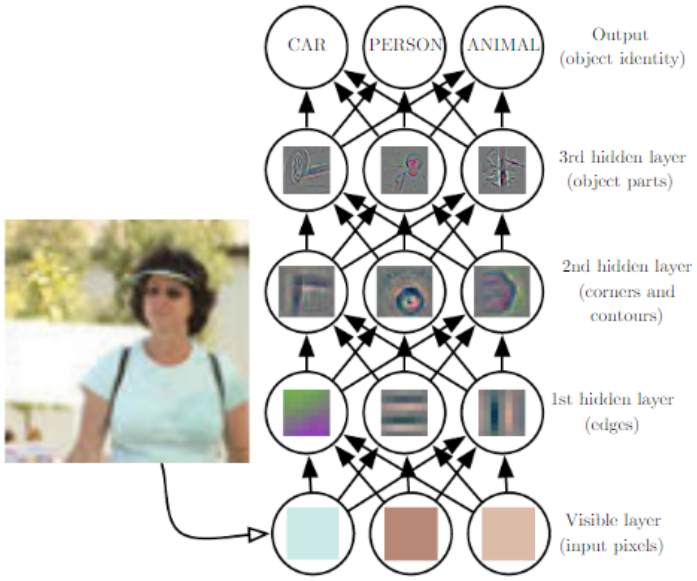


Figure 2.5: Illustration of a deep learning model (Source: [9]).

to achieve superior performance than conventional machine learning algorithms (see Table 2.1) in tasks such as image recognition [20], [21], natural language understanding [21], speech recognition [22], predicting the activity of potential drug molecules [88] and many more [23].

2.4 Zero-Shot Learning

Supervised classification methods aim to categorize instances into distinct sets based on predefined classes, assuming adequate training data for each category. The resulting classifier is limited to classifying instances within the classes covered by the training set. However, practical scenarios may arise where certain classes lack sufficient representation in the training set. The following are some frequent application scenarios, where such a situation might arise:

- *Large number of classes.* In many applications, the number of possible classes can be

extremely large, making it impractical to collect labeled examples for every class. For instance, in fine-grained image recognition tasks like identifying bird species [7] or plant varieties, the sheer number of categories can be overwhelming.

- *Rare Classes.* Some classes may be rare or infrequently encountered, leading to a scarcity of labeled instances. For example, in medical diagnosis, rare diseases may have very few documented cases, making it difficult to train a model using traditional supervised learning methods.

- *Dynamic Environments.* In dynamic environments where new classes continually emerge, traditional supervised learning models would require constant retraining with new labeled data.

- *High cost of labeling.* A prime example is the image semantic segmentation problem [89]. This task entails pixel-level classification, making precise labeling challenging.

In such applications, there are classes with no labeled instances, which renders supervised learning methods impossible to apply. To solve this problem zero-shot learning (ZSL) is proposed.

To understand zero-shot learning, let us consider the example of a person who has never seen a zebra before. If someone tells them that a zebra looks like a horse but with black and white stripes, they can use this description to recognize a zebra based on their prior knowledge of what a horse looks like. Humans can naturally extract and use high-level information to identify new objects by combining familiar characteristics. Similarly the goal of zero-shot learning is to enable a machine learning model to classify new categories by using descriptions or attributes shared with other known categories. In our previous example, if a model knows about horses and stripes, then we would like it to combine this knowledge to recognize a zebra.

Therefore the goal of zero-shot learning is to extract knowledge from already known classes and apply it to unknown ones. By using descriptive information and attributes, ZSL models can make guesses about new categories based on their similarities to familiar ones. This approach allows machine learning systems to be more flexible, by being applied to to a broader range of scenarios without requiring exhaustive labeled datasets for every possible category.

Chapter 3

Background

Contents

3.1 Neural Networks

3.1.1 The Neuron

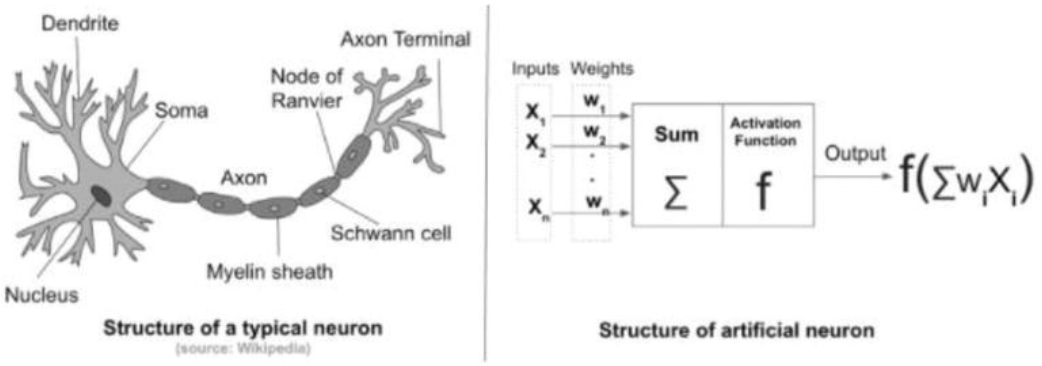


Figure 3.1: Biological neurons and artificial intelligence (Source: [1])

Neural networks, inspired by the architecture and functionality of the human brain, emulate the behavior of neurons. In these networks, layers of artificial neurons collaborate to process inputs and generate outputs. Each neuron receives a set of inputs, which are multiplied by adjustable weights that dictate the impact of each input on the overall output. Additionally, a modifiable bias is added to this weighted sum. The resultant combined input is then passed through an activation function, which converts it into a meaningful output, such as a probability or binary decision. This sophisticated mechanism enables neural networks to discern intricate patterns and make informed predictions based on the data they analyze. Specifically if $x \in \mathbb{R}^d$ is the input to the neuron, then the output is

$$y = f\left(\sum_i w_i x_i + b\right) = f(w^T x + b) \tag{3.1}$$

where f is called an *activation function*, $w = (w_1, \ldots, w_d) \in \mathbb{R}^d$ are the learnable weights and $b \in \mathbb{R}$ is the learnable bias.

Activation Functions

Activation functions are a critical component of neural networks, serving to introduce non-linearity into the model. This non-linearity enables the network to learn and represent complex patterns in the data. Without activation functions, a neural network would simply be a linear model, regardless of the number of layers it has, and would therefore lack the capacity to model intricate relationships [9]. Some of the most common activation function are the following:

- **Linear**: The linear activation function is basically the identity function. It has the form:

$$f(x) = x \tag{3.2}$$

- **ReLu**: The ReLu activation function uses a threshold at zero with the form:

$$f(x) = \max\{0, x\} \tag{3.3}$$

- **Sigmoid**: The Sigmoid activation function maps all of $\mathbb{R}$ in the range $(0, 1)$. It has the form:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

  In many occasions the output of this function is considered a probability.

- **Tanh**: The Tanh activation function outputs values between -1 and 1, mapping positive values to $(0, 1)$ and negative values to $(-1, 0)$. It has the form:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.5}$$

- **Softmax**: The softmax activation function $f : \mathbb{R}^n \to \mathbb{R}^n$ takes as input a vector $x \in \mathbb{R}^n$ and outputs a vector with the same dimension, whose components lie in $(0, 1)$. It is often used in multiclass classification scenarios:

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3.6}$$

The function discussed above can be visualized in Figure 3.2

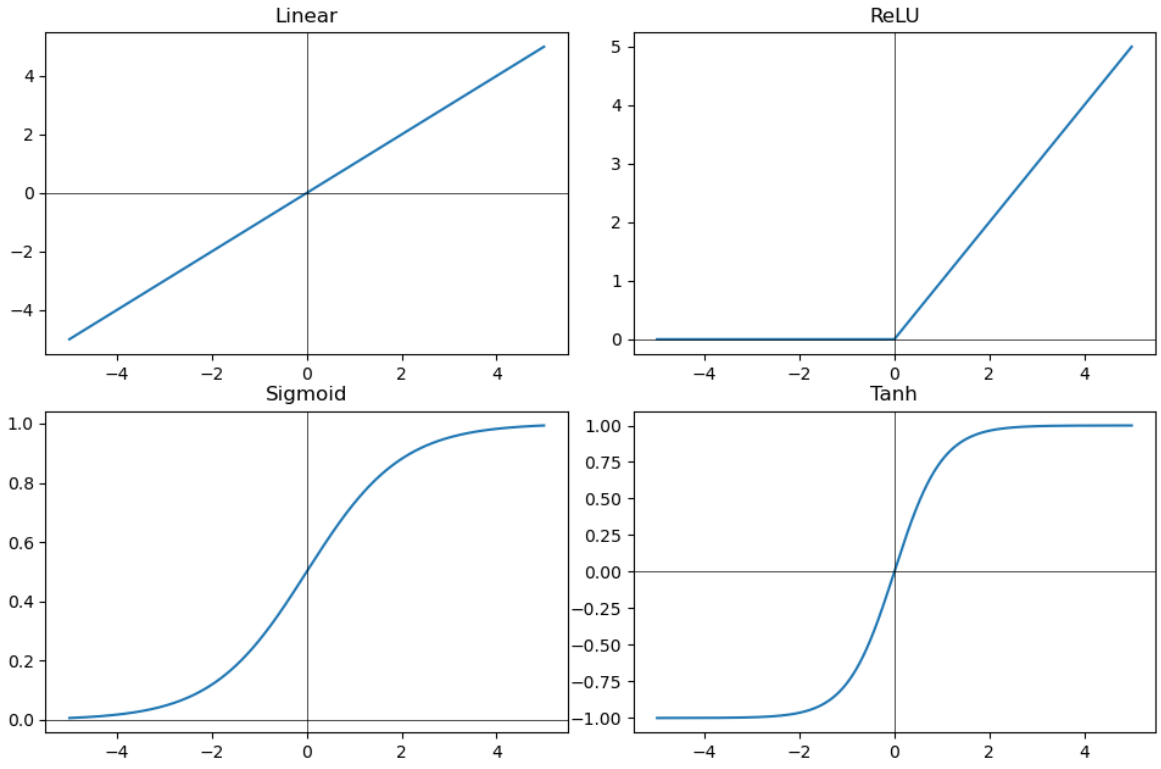


Figure 3.2: Different activation functions

3.1.2 Multilayer Perceptron

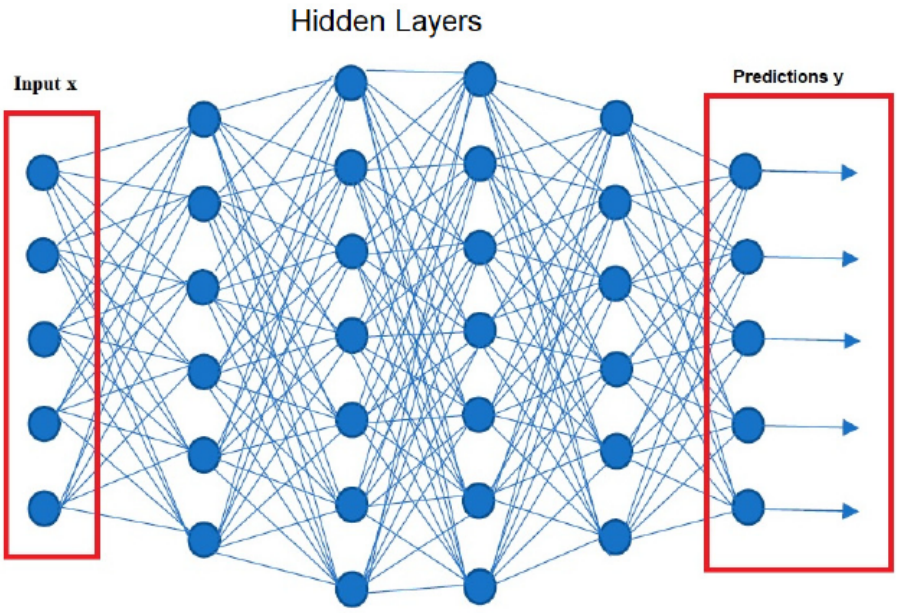


Figure 3.3: Representation of a Multilayer Perceptron (Source: [2])

A Multilayer Perceptron (MLP) is an artificial neural network in which multiple layers of neurons are stacked one after the other, as shown in Figure 3.3. Neurons in an MLP are organized into layers: an input layer, one or more layers called hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the subsequent layer, forming a fully connected network. The input provided to the MLP is fed into the input layer, which further processes the information and passes it through the hidden layers. Each of the hidden layers changes the presentation of the information using the mathematical formulation of the neurons and finally produces the output using the output layer.

For a given layer, the output y is calculated using the equation:

$$y = f(Wx + b) \tag{3.7}$$

where W is the matrix of learnable weights connecting the neurons of the current layer to the neurons of the previous layer, x is the input vector, b is a vector of learnable biases, and f is the activation function applied element-wise (see section 3.1.1). This process is repeated for each layer, with the output of one layer serving as the input to the next, thereby enabling the network to learn complex, non-linear mappings from inputs to outputs.

3.1.3 Optimization For Neural Networks

In supervised machine learning, the goal is to find the parameters θ that minimizes the empirical risk. The empirical risk is defined as the average of the loss function L over all training examples. Mathematically, this can be expressed as:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i, \theta), y_i) \tag{3.8}$$

Here f is the predictive function parameterized by θ, which takes as input x_i and predicts an output $\hat{y}_i$. The loss function L measures the discrepancy between the predicted output $\hat{y}_i$ and the true label y_i.

In the context of neural networks, the function f represents the entire network, containing multiple layers of neurons with various activation functions. The parameters θ include the weights and biases of all neurons in the network. The objective of training a neural network is to adjust these parameters such that the empirical risk $J(\theta)$ is minimized, which we hope will lead to accurate predictions on the test data.

Loss functions Loss functions play a crucial role in model training as they measure the discrepancy between the predicted output and the true target. Different types of loss functions are used depending on the specific task at hand.

For regression tasks, where the goal is to predict continuous values, the Mean Squared Error (MSE) loss is commonly used and it is defined as:

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.9}$$

Here y_i represents the true value, $\hat{y}_i$ is the predicted value and n is the number of samples.

For binary classification tasks, where the goal is to predict one of two possible classes, the Binary Cross-Entropy (BCE) is a common loss, which is defined as

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.10}$$

In Equation 3.10, y_i represents the true binary label (0 or 1), and $\hat{y}_i$ is the predicted probability of the sample belonging to class 1.

Finally, for multi-class classification tasks, where there are more than two classes, the Cross-Entropy loss is generalized to handle multiple classes. The Cross-Entropy loss for multi-class classification is given by:

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{3.11}$$

In Equation 3.11, C is the number of classes, $y_{i,c}$ is the binary indicator (0 or 1) if class label c is the correct classification for observation i, and $\hat{y}_{i,c}$ is the predicted probability that observation i belongs to class c.

Regularization

Regularization is a technique used to prevent overfitting in machine learning models. Overfitting occurs when a model learns not only the underlying patterns in the training data but also the noise. This results in excellent performance on the training set but poor generalization to unseen data. Regularization addresses this problem by adding a penalty to the model's complexity, discouraging it from fitting the noise. By constraining the model parameters or adding a penalty term to the loss function, regularization promotes simpler models that generalize better to new data.

- **L2-regularization** L2-regularization works by adding a penalty term to the loss function, which is proportional to the sum of the squares of the model parameters. That is, for every weight w in the network, the term λw^2 is added to the loss function where λ is the regularization strength.
- **L1-regularization** L1 regularization is another technique used to prevent overfitting in machine learning. It adds a penalty proportional to the sum of the absolute values of the model parameters, so for every weight w, the term $\lambda|w|$ is added to the loss function. This technique encourages sparsity in the model parameters, since it tends to drive some of the weights to zero, effectively performing feature selection by excluding irrelevant features.
- **Early-Stopping** Early stopping is a regularization technique used to prevent overfitting in neural networks by monitoring the model's performance on a validation set and halting training once performance starts to degrade. This method leverages the observation that while training error typically decreases over time, validation error often decreases initially but eventually starts to increase as the model begins to overfit the training data.

The Chain Rule

Let x be a real number, and let f and g both be functions mapping from a real number to a real number. Suppose that $y = g(x)$ and $z = f(g(x)) = f(y)$. Then the chain rule states that

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} \tag{3.12}$$

We can generalized this result in the case where some functions take vectors as input. Suppose that $g : \mathbb{R}^m \to \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. Then if $z = g(y)$ and $y = f(x)$ it follows that

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i} \tag{3.13}$$

In vector form this can be written also as:

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^T \cdot \nabla_y z \tag{3.14}$$

where $\partial y / \partial x$ is the Jacobian matrix of the function g.

Gradient Descent Suppose we have a function $f : \mathbb{R}^n \to \mathbb{R}$. To find the directional derivative of f in the direction u at point x, we consider the function

$$g(t) = f(x + tu)$$

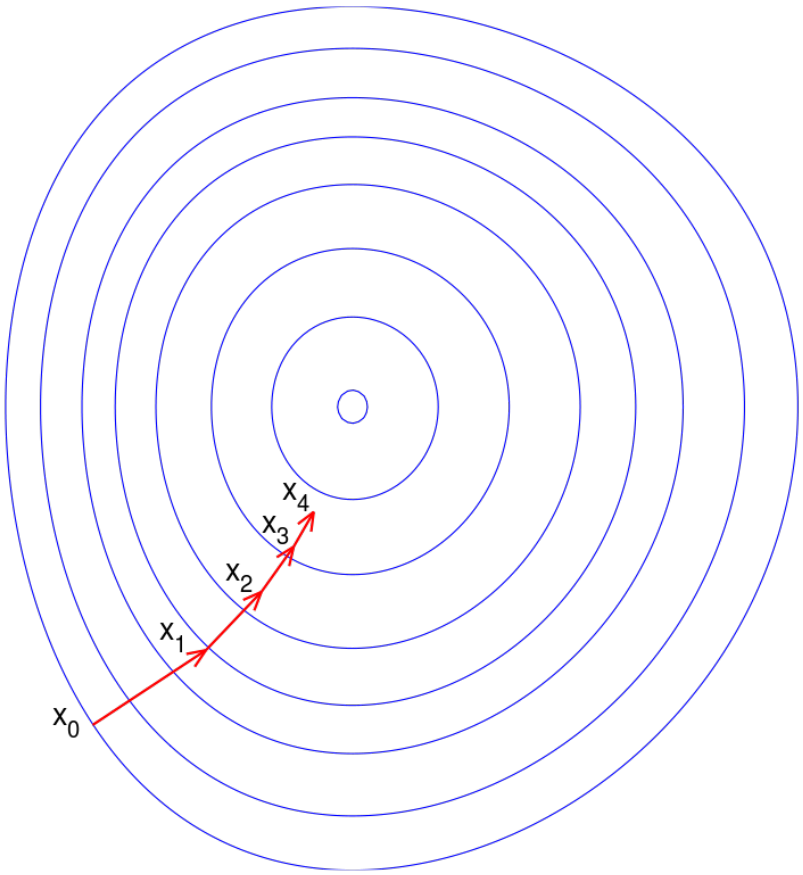


Figure 3.4: Visualization of the gradient descent method. (Source: [3])

The derivative of $g(t)$ is given by $g'(t) = u^T \nabla_x f(x + tu)$. Evaluating this at $t = 0$, we get $g'(0) = u^T \nabla_x f(x + tu)$. To find the direction u in which f decreases the fastest we use that fact that $u^T \nabla_x f(x) = ||u||_2 ||\nabla_x f(x)||_2 \cos\theta$ where θ is the angle between the two vectors. It is evident that the above expression is minimized when u points in the opposite direction of the gradient. Thus we should take $u = -\nabla_x f(x)$.

To understand how this is applied in practice, suppose we have a differentiable function f. Then using Taylor's series we have that:

$$f(x(t) + \Delta x) \approx f(x(t)) + \nabla f(x)|_{x=x(t)} \Delta x \tag{3.15}$$

Considering the direction where f decreases the fastest, discussed earlier, we set $\Delta x = -a \nabla f(x)|_{x=x(t)}$, $a > 0$. This leads us to the update rule

$$x(t+1) = x(t) - a \nabla f(x)|_{x=x(t)} \tag{3.16}$$

and

$$\begin{aligned} f(x(t+1)) = f(x(t) + \Delta x) &\approx f(x(t)) - a||\nabla f(x)|_{x=x(t)}||_2^2 \\ &\leq f(x(t)) \end{aligned} \tag{3.17}$$

In practice, choosing the appropriate learning rate a is crucial for the performance of the gradient descent algorithm. If the learning rate is too small, the algorithm may converge slowly. If it is too large, the algorithm may overshoot the minimum and fail to converge.

Optimizing Neural Networks Weights

Using the aforementioned basic concepts, we shall explain how optimization works in neural networks in a end-to-end fashion. Recall that the goal is usually to minimize the empirical loss. Then a regularization term is added to avoid overfitting:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i;\theta), y_i) + \lambda R(\theta) \tag{3.18}$$

In order to minimize $J(\theta)$ instead of using the gradient descent algorithm, a variant called stochastic gradient descent (SGD) is usually used. If we applied equation 3.16 to equation 3.18 then the gradient with respect to all samples $\{x_i\}_{i=1}^{n}$ would have to be computed, which in large datasets becomes computationally intractable. SGD only considers a batch of samples for each gradient step, which offers scalability in such situations. An often related problem to SGD is that the individual updates can be noisy, causing the algorithm to oscillate around the minimum. To solve this problem optimization methods such as Adam [25] use momentum, and adaptive learning rate.

Backpropagation is the algorithm used to compute gradients for each parameter in neural networks, enabling the application of gradient-based optimization methods like SGD. The essence of backpropagation is the chain rule of calculus, which allows the gradient to be efficiently propagated backward through the network.

1. Forward Pass: In order to apply Equation 3.16 we must evaluate $J(\theta)$ at θ_t. Therefore we compute the output of the network by passing the input data through each layer.

2. Loss Computation: Calculate the loss between the network's output and the true labels (see for example Eq. 3.10, 3.9).

3. Backward Pass: Firstly compute the gradient of the loss with respect to the output of the network. Then use the chain rule described in Equation 3.13 to propagate these gradients backward through each layer, computing the gradient with respect to each parameter.

3.1.4 Universal Approximation Properties and Depth

Linear models use matrix multiplication to map features to outputs, inherently limiting them to linear functions. This simplicity makes them easy to train since their loss functions typically result in convex optimization problems. However, in many real world scenarios, the relation between the input and the output might be highly non-linear. Feedforward neural networks with hidden layers provide a solution, as showed by the universal approximation theorem ([9], [4]). This theorem states that a feedforward network with a linear output layer and at least one hidden layer employing a "squashing" activation function, like the logistic sigmoid, can approximate any Borel measurable function with any desired accuracy, given a sufficient number of hidden units [9], [4]. These networks can also approximate the function's derivatives with high accuracy.

According to Goodfellow et al [9], choosing a deep model conveys certain beliefs about the nature of the function being learned. He points out that when we select a particular machine learning algorithm, we implicitly assert assumptions about the function's structure. He also underlines that deep models encapsulate the belief that the function involves the composition

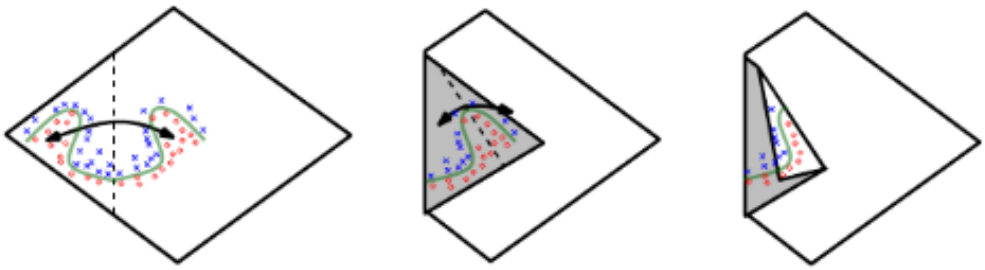


Figure 3.5: Deep neural networks involve the composition of several simpler functions. (Source: [9])

of several simpler functions and that from a representation learning viewpoint, this suggests that the learning task involves identifying a set of underlying factors of variation that can be described in terms of other simpler factors.

3.1.5 Convolutional Neural Networks

Using Multilayer Perceptrons (MLP's) for image processing presents significant challenges [4]. Firstly, consider the dimensionality of the input: an image of height H, width W and C color channels would require the first hidden layer to have a dimension of $H \times W \times C \times D$, where D is the number of neurons in the first hidden layer. Secondly, MLP's lack translation invariance [4], [9]. This characteristic can be visualized in Figure 3.6. To face these problems convolutional neural networks (CNN's) are proposed. Their basic idea is to divide an image into overlapping regions and then compare them with small weight matrices, often called *filters*. These filters can be thought of as detectors that recognise to which degree a particular feature lies in an image.

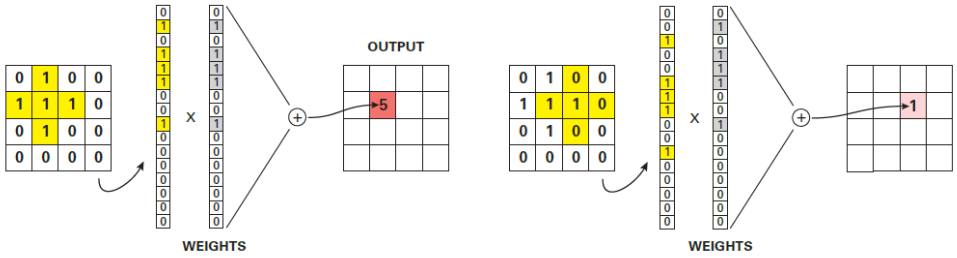


Figure 3.6: MLP's are not translation invariant (Source: [4])

Convolution Discrete two-dimensional convolution involves two matrices and is defined by the formula:

$$S(i,j) = (I \circledast K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.19}$$

This equation means that we take the image matrix I and the flipped kernel matrix K, and for each position (i,j) in the resulting matrix S, we compute the sum of element-wise multiplications of overlapping elements from I and K. The operation slides the flipped kernel across the image, calculating the weighted sum of the pixels under the kernel at each position.

Convolution is communative [9], meaning we can also write the result as:

$$S(i,j) = (K \circledast I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.20}$$

This alternate form shows that the order of the matrices in the convolution operation does not affect the result. It's like keeping the kernel centered at $(0,0)$ and sliding the image across it.

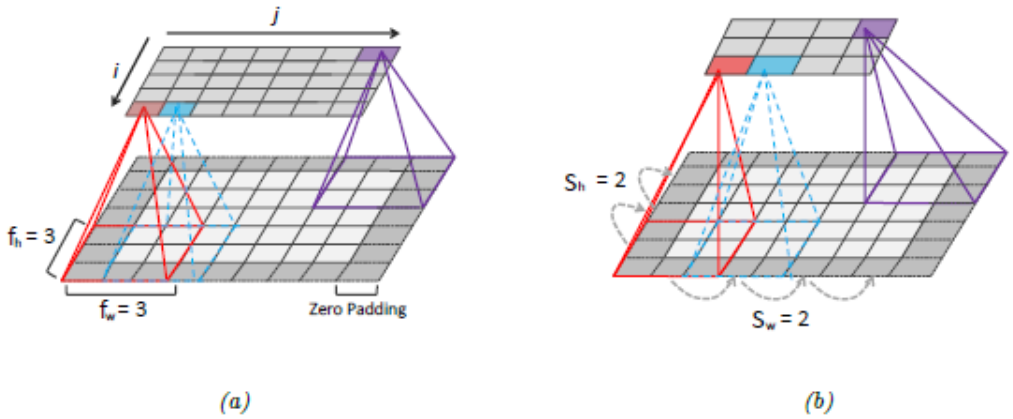


Figure 3.7: (a) convolution of an 2D image with a 2D kernel. Padding and stride are set to one. (b) convolution of an 2D image with a 2D kernel. Padding is set to one while stride is two. (Source: [4])

Stride and Zero Padding Stride refers to the number of pixels by which we slide the kernel across the input image. For example a stride of 1 means that we move the filter one pixel at a time, resulting in a highly detailed feature map, as shown in part (a) of Figure 3.7. A stride of 2 (Figure 3.7 (b)) would lead to smaller output dimension, which would help in capturing the essence of features while reducing the computational complexity. Zero padding involves adding a border of zeros around the input image. It is a hyperparameter which further controls the dimension of the output map. Both (a) and (b) in Figure 3.7 use zero padding of one.

Pooling layer A pooling layer is an essential component of CNN's which reduces the spatial dimensions of the input feature maps, thereby lowering the computational load and helping to make the network more efficient to spatial variations. The pooling layer summarizes the features within a particular region of the input, providing a form of down-sampling. Some of the most common pooling layers are max pooling, an operation that selects the maximum value within a defined window and average pooling, which computes the average of the values within the window. Such layers not only significantly reduce the computational burden of the algorithm, but also make the representation approximately *invariant* to small translations of the input [9].

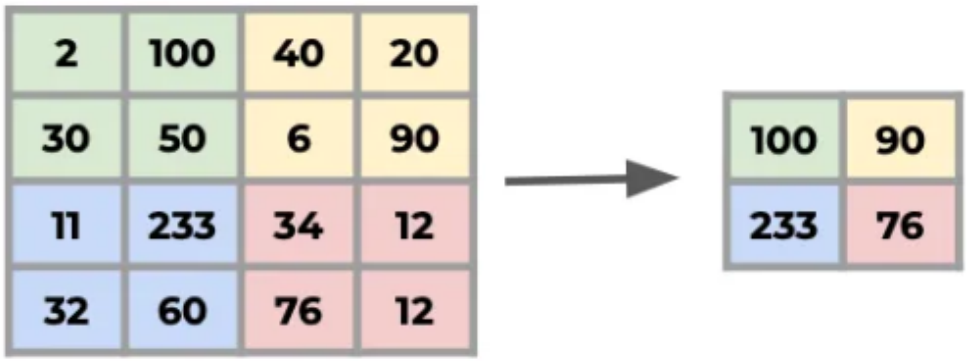


Figure 3.8: Example of the max pooling operation, using a kernel of size 2×2 and stride 2. (Source: [10])

In practice, images are not strictly 2D but 3D, as they contain color channels. Thus, the filters in a CNN must also operate in the depth dimension, and therefore the convolution is three dimensional. The output of a 3D convolution is a 3D volume where each slice is a response to the filter applied across the input.

Parameter Sharing in CNN's One of the key features of CNNs is parameter sharing. In the context of 3D convolution, parameter sharing implies that the same filter (set of weights) is used across different spatial and depth locations of the input. This greatly reduces the number of parameters and computations required, compared to fully connected layers, where each element in the input volume would have a unique weight associated with it. This is another reason which explains why these kind of networks can detect patters regardless of their position in the input volume.

Convolutional Neural Networks as Feature Extractors

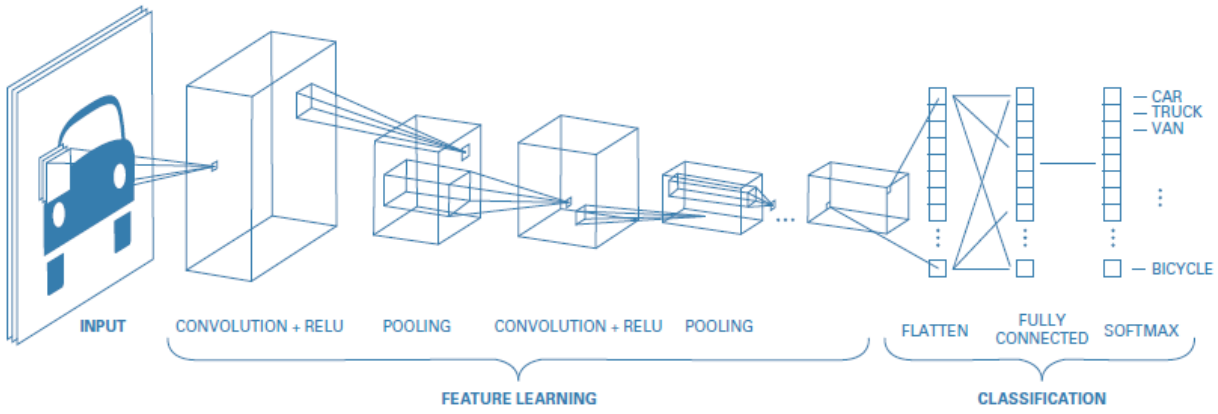


Figure 3.9: Convolutional Neural Network Architecture for Image Classification (Source: [11])

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, demonstrating remarkable performance on various image recognition tasks [26], [27], [28], [29]. Trained on large datasets, such as ImageNet [30], which contains millions of labeled images, CNNs have

shown their capability to learn complex visual patterns. A prime example of this is the ResNet (Residual Network) architecture [31], which has achieved state-of-the-art results in image classification competitions.

As discussed before, one of the key strengths of CNNs lies in their hierarchical feature learning. Early layers of a CNN typically learn to detect simple features such as edges, textures, and basic shapes. As we move deeper into the network, the layers start to capture more complex features, such as object parts [4]. This hierarchical learning allows CNNs to build a rich and abstract representation of the input images.

In practice, CNNs trained on large-scale datasets are often used as feature extractors. Specifically, the convolutional layers (usually the later ones) are leveraged to extract high-level features from images. These features can then be fed into other machine learning models or fine-tuned for specific tasks. This process is known as transfer learning [32], where a pre-trained CNN is used to extract meaningful features from new data, significantly reducing the training time and improving performance on tasks with limited data.

Figure 3.9 illustrates the process of using CNNs for feature extraction. There we can see an input image being processed through several convolutional and pooling layers, with ReLU activations. The resulting feature maps from the convolutional layers are then flattened and passed through fully connected layers for final classification. In this setup, the convolutional layers act as the feature extractors, capturing intricate patterns and structures within the images, ultimately enabling robust and accurate image classification.

3.2 Zero-Shot Learning

We emphasize that the analysis following in this particular section, heavily relies on the excellent reviews of [5] and [34].

In zero-shot learning, labeled training instances exist in the feature space, covering classes known as *seen* classes. Additionally, there are unlabeled testing instances in the same feature space, associated with a separate set of classes known as *unseen* classes. The feature space comprises vectors representing each instance, each assumed to belong to a single class.

Let $\mathcal{S} = \{c_i^s | i = 1, \ldots, N_s\}$ be the set of seen classes, where each c_i^s is a seen class. Also denote $\mathcal{U} = \{c_i^u | i = 1, \ldots, N_u\}$ as the set of unseen classes, where each c_i^u is an unseen class. Note that $\mathcal{S} \cap \mathcal{U} = \emptyset$ [12]. Denote $\mathcal{X}$ as the feature space, which is D-dimensional; usually it is a real number space $\mathbb{R}^D$ [12]. Denote $D^{tr} = \{(\mathbf{x}_i^{tr}, y_i^{tr}) \in \mathcal{X} \times \mathcal{S} | i = 1, \ldots, N_{tr}\}$ as the set of labeled training data belonging to seen classes; for each labeled instance $(\mathbf{x}_i^{tr}, y_i^{tr})$, $\mathbf{x}_i^{tr}$ is the instance in the feature space, while y_i^{tr} is the corresponding class label. Denote $X^{te} = \{\mathbf{x}_i^{te} \in \mathcal{X} | i = 1, \ldots, N_{te}\}$ as the set of testing instances, where each $\mathbf{x}_i^{te}$ is a testing instance in the feature space. Denote $Y^{te} = \{y_i^{te} \in \mathcal{U} | i = 1, \ldots, N_{te}\}$ as the corresponding class labels for X^{te}.

Definition 3.2.1 (zero shot learning [12])**.** The goal of zero-shot learning is to learn a classifier $f_{\text{ZSL}}(\cdot) : \mathcal{X} \to \mathcal{U}$ that can classify testing instances X^{te} belonging to the unseen classes $\mathcal{U}$.

From the definition, it's evident that zero-shot learning primarily aims to transfer the knowledge embedded in the training instances D^{tr}, to the task of classifying testing instances. It is interesting to point out that the label spaces covered by the training and testing instances are disjoint, making ZSL a subset of transfer learning (TL) [32].

3.2.1 Auxiliary Information

To address the absence of labeled instances of the unseen classes in zero-shot learning, auxiliary information is essential. Existing approaches draw inspiration from human cognition, where semantic background knowledge aids in recognizing unfamiliar entities [33]. For instance, knowing that "a tiger resembles a large cat with stripes' enables recognition of a tiger even without prior exposure to relevant images, based on knowledge of cats and stripe patterns [5]. Consequently, auxiliary information in zero-shot learning methods typically comprises semantic details, which form a *semantic space* containing both seen and unseen classes.

In the semantic space, each class is characterized by a vector representation, which is referred to as the *class prototype* of the class. In what is next, we follow [34] in order to define the semantic space and the prototype of a class.

Denote $\mathcal{T}$ as the semantic space. Suppose $\mathcal{T}$ is M-dimensional; it is usually $\mathbb{R}^M$ [12]. Denote $\mathbf{t}_i^s \in \mathcal{T}$ as the class prototype for seen class c_i^s, and $\mathbf{t}_i^u \in \mathcal{T}$ as the class prototype for unseen class c_i^u. Let $T^s = \{\mathbf{t}_i^s | i = 1, \ldots, S\}$ be the set containing the prototypes of seen classes, and $T^u = \{\mathbf{t}_i^u | i = S+1, \ldots, S+U\}$ the set of prototypes of unseen classes. Let $\pi(\cdot) : \mathcal{S} \cup \mathcal{U} \to \mathcal{T}$ be a *class prototyping function* that takes a class label as input and outputs the corresponding class prototype (e.g $\pi(\text{zebra}) \in \mathbb{R}^M$). In section 3.2.4 we will analyze different kinds of semantic spaces.

3.2.2 Learning Settings

As stated in definition 3.2.1, the goal of traditional ZSL is to learn a classifier $f^u(\cdot)$. In practice situations can occur, where information about some testing classes is known a priory. Then the model is *transductive* with respect to these specific testing instances. In this setting, information about the testing classes is also known during training. Concretely Wang et al. [34] discriminates between the following settings:

Definition 3.2.2 (Class-Inductive Instance-Inductive (CIII) Setting)**.** Only labeled training instances D^{tr} and seen class prototypes T^s are used in model training.

Definition 3.2.3 (Class-Transductive Instance-Inductive (CTII) Setting)**.** Labeled training instances D^{tr}, seen class prototypes T^s, and unseen class prototypes T^u are used in model training.

Definition 3.2.4 (Class-Transductive Instance-Transductive (CTIT) Setting)**.** Labeled instances D^{tr}, seen class prototypes T^s, unlabeled testing instances X^{te}, and unseen class prototypes T^u are used in model learning.

3.2.3 Testing Settings

In traditional zero-shot learning (ZSL), the test set exclusively comprises samples from unseen classes, a scenario which is not practical real-world applications. In practice, data samples from seen classes are more frequent than those from unseen ones. Therefore, it's crucial to simultaneously recognize samples from both categories rather than focusing only on the unseen class samples. This setup is termed *generalized zero-shot learning* (GZSL) [35].

Definition 3.2.5 (Generalized Zero Shot Learning (GZSL) [5])**.** The goal of generalized zero-shot learning, is to construct a classifier $f_{\text{GZSL}} : \mathcal{X} \to \mathcal{S} \cup \mathcal{U}$ that can classify testing instances X^{te} that belong to both the seen classes $\mathcal{S}$ and the unseen classes $\mathcal{U}$.

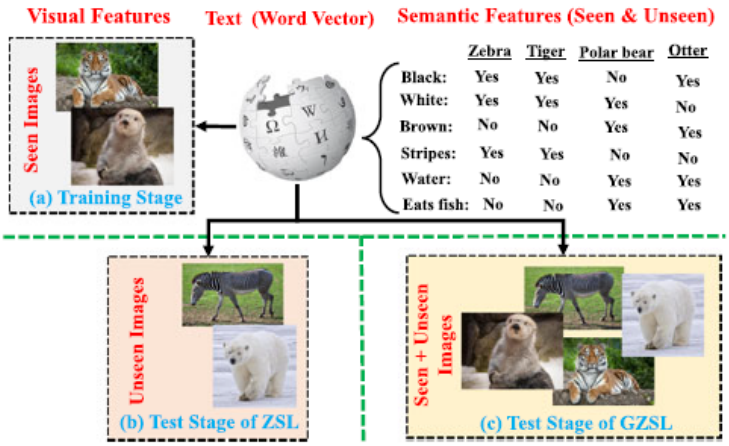


Figure 3.10: (b) During the test phase, ZSL can only recognize samples from the unseen classes, while (c) GZSL is able to recognize samples from both seen and unseen classes (adapted from [5])

Generalized zero-shot learning (GZSL) as defined in Definition 3.2.5 does not explicitly mention the set used during the training phase, just like the definition of zero-shot learning (ZSL) (see definition 3.2.1). The differentiating factor lies in the query posed to the model during testing [5]. In conventional ZSL, the labels of test set samples X^{te} are presumed to belong to unseen classes $\mathcal{U}$. In GZSL such an assumption is not made, so the labels of X^{te} can be both in $\mathcal{S}$ and $\mathcal{U}$. This fundamental difference is summarized in the lower part of Figure 3.10.

3.2.4 Semantic Spaces

In section 3.2.1 we briefly mentioned the semantic space $\mathcal{T}$, where each class is represented as an M-dimensional vector. In this section we will analyze how $\mathcal{T}$ is constructed

Attributes

In the context of zero-shot learning, the attribute space is a semantic space where each class is represented by a set of human-interpretable attributes or factors that characterize it. This method of representation is particularly advantageous because it aligns closely with how humans perceive and categorize objects, making it more intuitive. For instance, in an animal classification task, attributes might include features such as "has stripes," "has wings," "is aquatic," or "is nocturnal." These attributes provide a clear and descriptive way to define each class, facilitating easier understanding and manipulation by humans.

Consider the example of distinguishing between different types of animals. A zebra can be characterized by attributes such as "has black and white stripes," "is a mammal," and "lives in grasslands." In contrast, a lion might be described with attributes like "has a mane," "is a carnivore," and "lives in savannas." When an unseen class, such as an okapi, is introduced, it can be described using a combination of known attributes, such as "has stripes on legs" and "is a mammal."

These attributes can be represented as a binary vector, where each element indicates the presence (1) or absence (0) of a specific attribute. For example, if we define the attributes as follows: [has stripes, has mane, is a mammal, is a carnivore, lives in grasslands, lives in savannas],

a zebra could be represented by the binary vector [1, 0, 1, 0, 1, 0], while a lion would be [0, 1, 1, 1, 0, 1]. An okapi might then be represented as [1, 0, 1, 0, 1, 0], showing its similarity to the zebra in terms of attributes. This idea can be visualized in Figure 3.11.

This attribute-based approach allows for the recognition and classification of new, unseen classes based on their semantic similarities to known classes.

Figure 3.11: (b) During the test phase, ZSL can only recognize samples from the unseen classes, while (c) GZSL is able to recognize samples from both seen and unseen classes (adapted from [12])

Word Vectors

Another approach to constructing semantic spaces in zero-shot learning involves using embeddings of class names. This method leverages natural language processing techniques to generate vector representations of class names, enabling the model to capture semantic relationships between different classes. Techniques like Word2Vec [90] and GloVe [91] are commonly used for this purpose in ZSL [54], [55]. These methods create word embeddings by analyzing large text corpora, capturing the contextual meaning of words.

Word2Vec, for instance, learns word associations by predicting a word based on its surrounding context within a sentence. This results in vectors where semantically similar words are positioned close to each other in the embedding space. Similarly, GloVe (Global Vectors for Word Representation) combines the benefits of both local context-based learning and global statistical information, producing embeddings that effectively capture word relationships. When applied to class names, these embeddings provide an alternative way to encode semantic information in an unsupervised manner. For example, the class names "zebra" and "horse" might be represented by embeddings that reflect their semantic similarity based on contextual appearance in text.

This unsupervised technique simplifies the construction of prototypes, because it does not require manual labeling of attributes. The embeddings provided, can then be used to infer relationships between seen and unseen classes, based on their relative position in the semantic

space.

3.2.5 Embedding Spaces

The embedding space in which classification will be conducted plays a crucial role in Zero-Shot learning. Specifically most of the ZSL methods find a common embedding space in which both the image feature vectors and class prototypes are projected, and then perform nearest neighbor search [54],[55],[92],[93]. According to [5] there are three primary embedding spaces: *the visual space, the semantic space, or an intermediate latent space*. Each of these has its weaknesses that impact the effectiveness of classification in different ways.

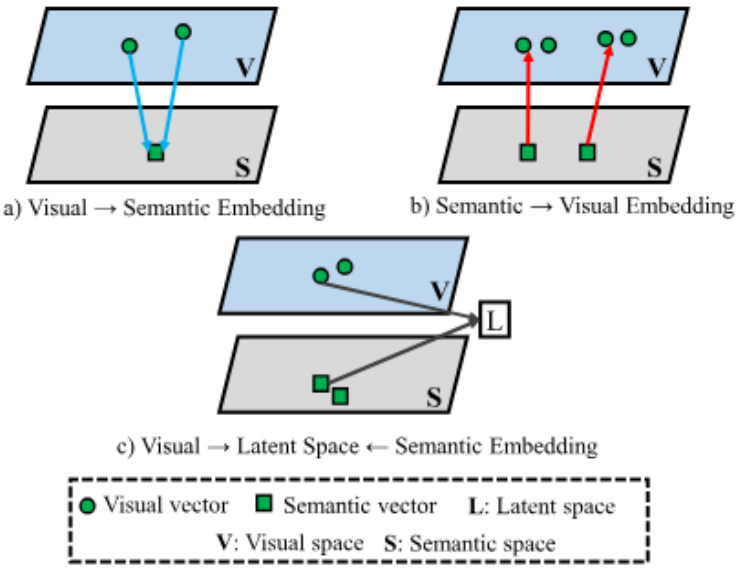


Figure 3.12: Different kinds of embedding spaces (Source: [5])

- **Visual Space**: In the visual space, features are extracted directly from images using convolutional neural networks (CNN's). This space is beneficial because it leverages rich visual information that is often sufficiently discriminative for object classification [31]. However, visual embeddings can suffer from high variance and the presence of outliers as we will see in chapter 6.

- **Semantic Space**: The semantic space utilizes high-level descriptions or attributes of classes, attribute lists, or word embeddings (see section 3.2.4). In some cases this space aligns with human understanding and leverages descriptive information to relate seen and unseen classes. However, projecting data into the semantic space introduces the so-called *hubness problem*, which we shall further analyze in section 3.2.6. In summary, this problem arises when certain points in the semantic space become "hubs" that are nearest neighbors to many other points. This causes the model to make biased predictions, reducing the overall performance of the classifier.

- **Latent Space**: Latent embedding spaces represent an intermediate approach, where both visual and semantic features are projected into a common embedding space, overcoming the hubness problem. One issue with this approach is the potential bias towards the seen

classes, especially when the model primarily focuses on labeled instances during training. This can lead the algorithm to overfit the seen classes and therefore poorly generalize to the unseen classes, undermining the end-goal of ZSL

3.2.6 Challenges In Zero-Shot Learning

Bias Towards The Seen Classes

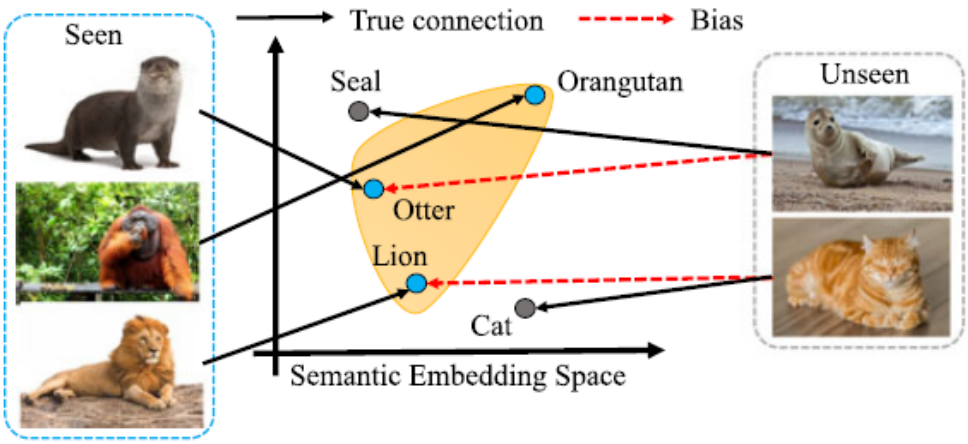


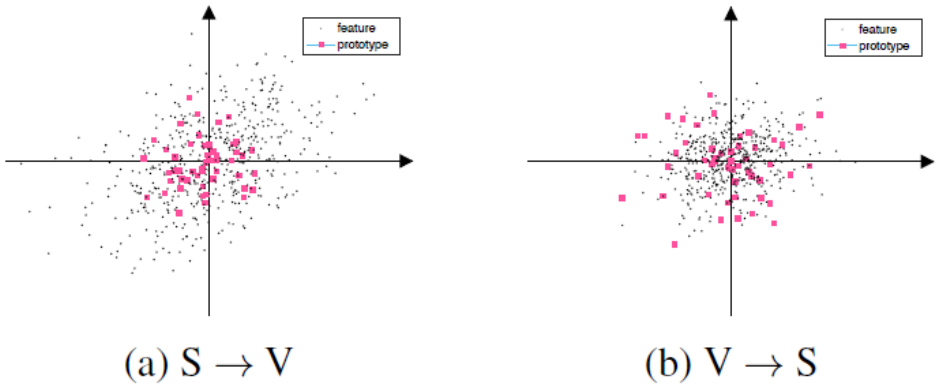
Figure 3.13: A schematic view of the bias in ZSL (Source: [13])

One of the significant challenges in conventional ZSL (see def. 3.2.2,3.2.3) is the bias towards seen classes. This bias occurs because the algorithm only has access to labeled samples from the seen classes during training. The naive approach of simply training the model to project the labeled visual samples as close as possible to their corresponding prototype, may cause the model to inaccurately project samples from the unseen classes in the embedding space, away from their prototype. Figure 3.13 illustrates this issue, where samples from unseen classes, such as the seal and the cat, are incorrectly projected closer to seen class prototypes like otter and lion respectively.

The Hubness Problem

As discussed in section 3.2.5, this problem arises when learning projection function from the visual space to the semantic space. This problem is a particular aspect of the *curse of dimensionality* [14], which renders some class prototypes as the nearest neighbors of the vast majority of projected instances. In order to be precise, Zhang et al. [14] studied the simple case, where one wants to connect the visual and the semantic, using ridge regression [94].

Specifically, he showed under the aforementioned formulation, that if samples from the visual space are projected to the semantic space, then the mapped source data are likely to be closer to the space origin than the target data (the prototypes). This can be visualized in Figure 3.14



(a) S → V (b) V → S

Figure 3.14: A schematic view of the difference between projecting to the visual-semantic space (adapted from [14])

To overcome this issue, many authors opt to utilize the visual embedding space [14], or a latent intermediate space [92].

3.3 The K-means algorithm

K-means clustering is a method used in unsupervised learning to group data into clusters that have similar characteristics. The goal is to partition a dataset into K distinct groups (clusters) where each data point belongs to the cluster with the nearest mean. This grouping helps in understanding the underlying structure of the data, identifying patterns, and simplifying complex datasets.

There are several reasons why this algorithm is useful:

- **Simplicity and Speed**: K-means is straightforward to understand and implement. It also works efficiently on large datasets, making it a practical choice for many real-world applications.
- **Patterns Recognition**: By grouping similar data points, K-means helps in identifying patterns and structures within the data that might not be immediately apparent.
- **Dimensionality Reduction**: Although the primary goal is clustering, the centroids themselves can be used to represent the dataset in a reduced form, aiding in dimensionality reduction.

Problem Statement Suppose we have a data set $\{x_1, \ldots, x_N\}$ consisting of N observations of a random D-dimensional Euclidean variable x. Our goal is to partition the data set into K clusters. Formally, we seek K points $\mu_1, \ldots, \mu_K$ to which if we were to assign the points $\{x_1, \ldots, x_N\}$, a measure of disparity would be minimized.

Following [15] for each data point x_n let us introduce the binary indicator $r_{nk} \in \{0, 1\}$. If the vector x_i is assigned to cluster j then $r_{ij} = 1$. Otherwise $r_{ij} = 0$. As usual in machine learning, We try to minimize some objective function. In this case, the most common objective function

is

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{3.21}$$

Equation 3.21 represents the sum of squared distances of the dataset points, to their assigned center. Clearly, the optimal values for $\{r_{nk}\}$ and $\{\mu_k\}$ depend on each other. Therefore, a two-stage algorithm is used.

Let us first find the optimal values for r_{nk} given the centers μ_k. The terms involving different samples n are independent and so we can optimize for each n separately by choosing r_{nk} to be 1 for the k which given the minimum value of $||x_n - \mu_k||^2$. In other words

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||x_n - \mu_j||^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

Next, let us find the optimal values for $\{\mu_k\}$ given the assignments $\{r_{nk}\}$. To find the optimal value for μ_k we simply set the derivative of J with respect to μ_k to zero which gives:

$$2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \tag{3.23}$$

Solving this equation for μ_k gives:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}} \tag{3.24}$$

Becuse each stage of the algorithm reduces the objective function J and $J \geq 0$, it follows that the algorithm will converge.

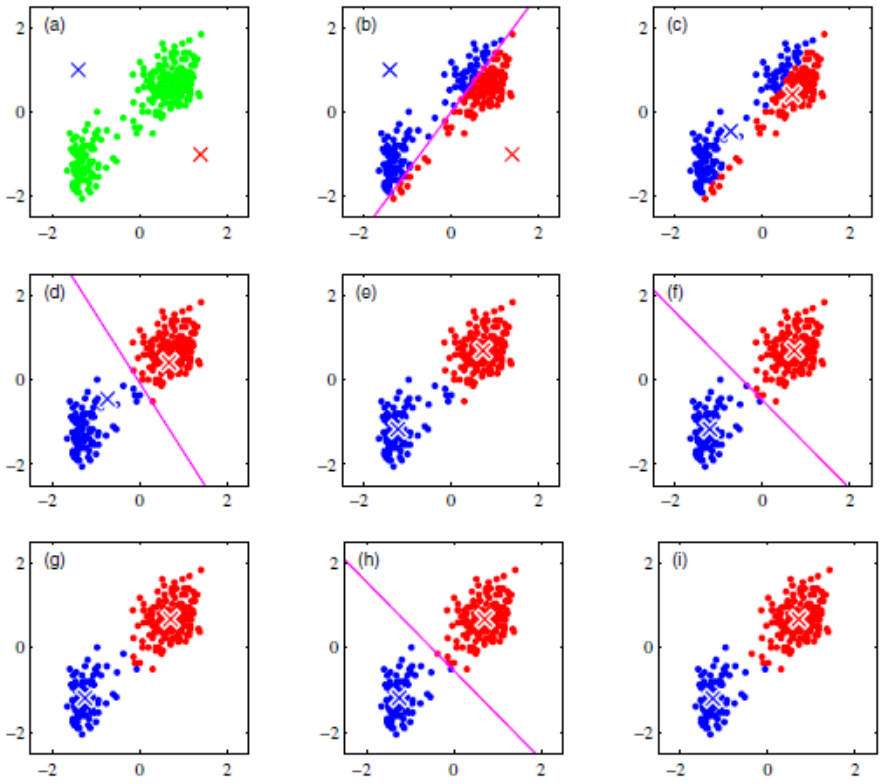


Figure 3.15: Illustration of the K-means algorithm (adapted from [15])

Chapter 4

Related work

Contents

4.1 Zero-Shot Learning

As discussed in chapter 3 Zero-shot learning (ZSL) is an emerging field in machine learning that addresses the challenge of classifying instances from classes that were not seen during the training phase. The primary approaches in ZSL can be broadly categorized into two main types: embedding-based methods and generative methods.

- **Embedding-based Methods**: these methods aim to find a common latent space where both visual features (e.g., images) and semantic representations (e.g., class attributes or word vectors) can be projected. The goal is to enable the comparison of these projections using a similarity measure. For instance, an embedding-based model might map an image of an unseen animal into a semantic space where the attributes of that animal, described in words, are also mapped. The model then classifies the image based on its proximity to the semantic representations of various classes within this space. Methods like these rely heavily on the quality of the embedding space and the similarity measure used, typically cosine similarity or Euclidean distance. Successful examples of embedding-based methods include models that utilize deep neural networks to learn these projections effectively, capturing intricate relationships between visual and semantic domains [16], [36], [37].
- **Generative Methods**: In contrast with embedding-based methods, generative approaches aim to synthesize visual features or even entire images for the unseen classes, leveraging the knowledge from seen classes and their semantic descriptions. By generating samples for the unseen classes, the ZSL problem is transformed into a conventional supervised learning problem where the training dataset now includes synthetic examples of the unseen classes. Generative Adversarial Networks (GANs) [38] and Variational Autoencoders (VAEs) [39] are commonly used in these methods to generate high-quality and diverse samples that closely mimic real-world instances. For example, given a textual description of an unseen animal, a GAN can generate images that correspond to that description, thereby providing the necessary data to train a classifier in a supervised manner.

As our method is embedding-based, we review these in more depth and provide more details on the associated advances. However, to provide a comprehensive overview of the field and acknowledge the diversity of approaches within zero-shot learning, in what follows, we briefly present some of the most relevant generative methods. These generative models, though not the principal focus of our work, play a pivotal role in advancing ZSL from another direction.

4.1.1 Embedding Based Methods

One of the first approaches to solve the Zero-Shot Learning problem was the DAP [61] algorithm. This algorithm operates by learning a set of probabilistic classifiers for each attribute based on the training data, which consists of images from known classes and their corresponding attribute vectors. For each attribute, a classifier is trained. These classifiers are then combined to form a complete image-attribute model. To perform classification, the Bayes rule is used in order to find the posterior probability of each unseen class given an image. The core idea of Akata et al. [72] is to directly learn a linear function to measure the compatibility between images and the label embeddings (prototypes). Then the learned compatibility function is used to assign each image in the test set to the prototype which maximizes this function. The method SJE [55] tries to optimize a billinear compatibility function optimizing a structural SVM loss. ESZSL [95] also

learns a linear compatibility function. Yet another linear model was proposed by Li et al. [93] who modeled inter and intra-class relations within a triplet loss framework. The classification is conducted in a common latent embedding space.

In [96] these ideas were extended by introducing an auto-encoder framework. Even though the model is still linear with respect to it's parameters the authors proposed a novel reconstruction loss which helps alleviating the bias problem (see section 3.2.6).

The Latent embedding Model (LATEM) [97] is a non-linear model, which extends the bilinear compatibility models by incorporating latent variables. Instead of learning a single bilinear map, this method learns multiple linear maps and uses latent variables to select the appropriate map for each image-class pair, making the overall model piecewise linear. CVCZSL [98] generates a prototype sample for each class, and trains a feedforward neural network that maps the class semantic vectors from the semantic space to the prototypes in the visual space. Furthermore, an episode-based training scheme is proposed to enhance the model's generalizability to new ZSL tasks.

ConSE [60] tried to leverage an existing n-way image classifier. The method maps images into a semantic embedding space by using a convex combination of the class label embedding vectors. Specifically, the classifier's probabilistic predictions for different labels are used to compute a weighted combination of these label embeddings, resulting in a continuous embedding vector for each image. Another approach was proposed by Bucher et al. [99] who tried utilizing metric learning in order to acquire a linear map, projecting the visual features in the semantic space, and also a Mahalanobis distance matrix.

An example of transductive, embedding based methods is the QFSL algorithm [100], which projects the visual features of the seen classes into a number of fixed points in a semantic space using a MLP, while forcing unlabeled target data to map to other points, thereby reducing the bias towards source classes (see section 3.2.6). The QFSL model is implemented as a deep neural network that consists of a visual embedding sub-net, a visual-semantic bridging sub-net, a scoring sub-net, and a classifier. The model is trained end-to-end, allowing for the optimization of the entire network, including the visual embedding and the bridging sub-nets. The key idea of the TEDE algorithm [16] is to construct an embedding space that ensures both intra-class compactness and inter-class separability to improve zero-shot learning (ZSL) performance. The method proposed involves a two-branch deep embedding model, which simultaneously maps semantic descriptions and visual samples into a joint embedding space. This method extends to the transductive scenario by using high-confidence predictions to iteratively refine the embedding space. Their idea can be visualized in Figure 4.1.

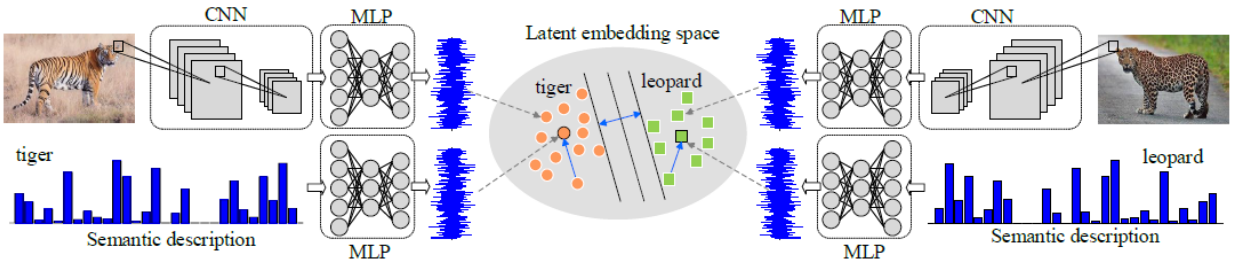


Figure 4.1: The architecture of the two branch network used by the TEDE algorithm [16] (adapted from [16]).

To the best of our knowledge there are only a few methods for transductive ZSL which

use clustering at their core [48], [49]. "Semi-Supervised Zero-Shot Learning by a Clustering-based Approach" [48] seeks a linear transformation that maps class signatures onto deep visual features, ensuring that mapped signatures of seen classes are close to their corresponding visual features and that unlabeled data are close to the mapped signatures of one of the unseen classes. Furthermore, a clustering algorithm assigns labels to instances of unseen classes based on the proximity of these instances to the cluster centers, which correspond to the mapped signatures. The method iteratively refines the mapping and label assignments. BMVSc [49] on the other hand addresses the domain shift problem using a visual structure constraint. This method uses the clustering property of visual features to ensure that the projected semantic vectors align well with the visual feature centers, leading to better generalization to unseen classes. One of the methods proposed is the bipartite matching based constraint, which ensures one-to-one mapping between projected semantic centers and visual cluster centers using bipartite matching distance.

4.1.2 Generative Methods

The "Creativity Inspired Zero-Shot Learning" (CIZSL) [101] method uses a GAN framework where a generator creates visual features from text descriptions, and a discriminator differentiates between real and fake features while classifying seen classes. The generator is trained with a creativity-inspired loss to produce realistic yet distinct features for unseen classes by maximizing entropy over seen classes. Hallucinated text descriptions, created by interpolating between seen class descriptions, guide the generator. This approach allows the generated features to be used for training classifiers, effectively transforming zero-shot learning into a supervised learning problem.

LisGAN [73] synthesizes fake visual features based on semantic attributes, and a discriminator distinguishes between real and fake features while ensuring inter-class discrimination. The method introduces "soul samples" as invariant representations that regularize the generator to produce realistic and semantically meaningful features. Soul samples act as meta-representations for each class, ensuring generated features closely align with these prototypes.

f-VAEGAN [42] introduces a method that combines Variational Auto-encoders and Generative Adversarial Networks to combine the best aspects of both methodologies and enhance the feature generating process.

4.2 The SCAN Algorithm

The method "SCAN: Learning to Classify Images without Labels" [40] is a SOTA method for unsupervised image clustering. This method involves three main steps: self-supervised learning, clustering with nearest neighbors, and fine-tuning through self-labeling.

Detailed Process Explanation

- **Self-Supervised Learning**: A convolutional neural network (CNN) is trained using self-supervised learning tasks. Common tasks include predicting the rotation of an image or distinguishing between augmented versions of the same image. These tasks force the network to learn high-quality feature representations that capture important visual characteristics.

- **Clustering with Nearest Neighbors**: After self-supervised learning, features are extracted from the images. For each image, its k-nearest neighbors are identified based on their features similarity. These nearest neighbors are then used in order to train the *clustering network*. The method uses a loss function that maximizes the similarity (dot product) between each image and its neighbors' feature representations, encouraging the network to produce consistent and discriminative cluster assignments

- **Fine-Tuning through Self-Labeling**: Initial clusters formed by nearest neighbor assignments are used to generate pseudo-labels for the images. These pseudo-labels serve as weak labels for supervised fine-tuning. The network is then fine-tuned by training it on the pseudo-labeled data, using a standard classification loss (e.g. cross-entropy loss). This iterative process refines the clusters, as the network learns to produce more consistent and accurate pseudo-labels with each iteration.

4.3 Motivation

In the rapidly evolving field of zero-shot learning (ZSL), generative-based methods have garnered significant attention and are currently dominating the literature. These methods, especially those involving Generative Adversarial Networks (GANs), have shown promising results by synthesizing visual features for unseen classes and effectively transforming the ZSL problem into a conventional supervised learning scenario. However, they often face instability issues during training, which can lead to inconsistent performance. For instance, in a study by Chochlakis et al. [41], an attempt to replicate the results of the state-of-the-art f-VAEGAN method [42] revealed that their replication results were 4-8% lower than the original paper's reported performance, depending on the dataset. This discrepancy highlights the challenges and potential unreliability associated with generative approaches.

On the other hand, embedding-based methods offer a simpler and more intuitive framework for zero-shot learning. These methods operate on a straightforward principle: seeking a space in which different classes exhibit intra-class compactness and inter-class separability. This fundamental concept, first articulated by the TEDE algorithm [16], is the cornerstone of many embedding-based approaches. The simplicity of this idea makes embedding-based methods more accessible and easier to understand compared to their generative counterparts.

Building on this principle, our research aims to leverage advancements in the unsupervised clustering bibliography, which inherently focuses on achieving intra-class compactness and inter-class separability. By integrating the SCAN clustering technique with embedding-based ZSL methods, we aspire to develop a more reliable framework for zero-shot learning. In summary, while generative-based methods have their merits, their instability challenges necessitate a reconsideration of embedding-based approaches. Furthermore, we believe that this approach considerably simplifies the underlying methodology of generative based methods.

Chapter 5

Methodology

Contents

5.1 Algorithm Overview

In this thesis we address the transductive zero-shot learning (ZSL) problem. In ZSL, the model is trained with a set of classes, but during testing, it is expected to recognize and classify instances from classes that were not seen during training. This implies that the model needs to generalize its knowledge to unseen classes based on some form of semantic understanding. In transductive learning, the model aims to predict labels for a set of unlabeled data points, typically exploiting the relationships between labeled and unlabeled instances [43]. This differs from the typical inductive learning setting, where the model learns a general mapping from input to output, based solely on labeled data. In transductive learning, the model's predictions may depend explicitly on the distribution of the unlabeled data [43].

Combining these two concepts, transductive zero-shot learning deals with the scenario where the model has to classify instances from the unseen classes, while having access to a set of labeled instances which belong to the seen classes, and a set of unlabeled instances which may belong to both seen and unseen classes. The goal is to leverage the semantic relationships between seen and unseen classes, in order to improve the model's classification performance, despite not having direct access to labeled examples for the unseen classes.

Our approach involves clustering all instances from both seen and unseen classes, using a *clustering network*, which comprises of a pretrained backbone and a small neural network on top of it. In more detail, a pretrained backbone refers to a neural network architecture that has been trained on a large dataset, for a particular task and its learned weights are then used as a starting point, in order to extract features, often for a different task. Using a pretrained backbone offers several benefits. It allows leveraging knowledge learning from large datasets, which can help improve performance on tasks with limited training data. Additionally, it can speed up training time, since the backbone has already learned useful features. In this work we have chosen the ResNet101 model as our backbone [44]. Following [40], on top of the backbone, a task-specific neural network is added, to take advantage of the extracted features and fine-tune the entire procedure for a specific task, which in this case is image clustering. The goal is to construct a space, where images belonging to the same class are close to each other, while images belonging to different classes are far away from one another. This approach leverages both the general knowledge encoded in the pretrained backbone and the specific task requirements addressed by the task-specific head, resulting in an efficient solution for the clustering task.

After this step, we project all images in the visual space generated by the trained clustering network. After that, we compute the mean of each seen class and utilize the K-means algorithm to estimate the means of the unseen classes. The means mentioned above, will be referred to as visual representatives.

Having found the visual representatives we hypothesize that instances from the same class are well clustered together, while samples from different classes are far away from one another. If this hypothesis holds, then the visual representatives are compact and representative points that characterize each class. These points effectively capture the essence of each class, allowing us to measure the similarity or distance between an input image and each class center. Classification then becomes a matter of assigning the input image to the class whose center it is closest to, in terms of some distance metric, such as Euclidean distance or cosine similarity.

Thus the problem boils down to assigning a class name to each visual representative. The correspondence between the known classes, whose labels are used during training, and a subset of the visual representatives, is already known, and so is their correspondence to the known class semantic vectors. This set of correspondences can be thought of as a function, which we would

like to extend in the "best" possible way, in order to map the unknown class semantic vectors set, to the unknown visual representatives set, in a one-to-one and onto fashion. The underlying assumption is that the two spaces share some kind of structural similarity, which is not granted for certain.

The problem described shares parallels with a well-studied task in the field of natural language processing, known as bilingual lexicon induction (BLI) [45],[46]. BLI aims to find corresponding words or phrases in two different languages, typically from comparable corpora, without explicit supervision for all pairs [46]. Much like in our scenario, where we aim to establish correspondences between image cluster means and semantic vectors, BLI leverages structural similarities between the two language spaces to infer correspondences.

By adopting techniques from recent advancements in the field [46], such as supervised embedding alignment and unsupervised distribution matching, the framework we use jointly optimizes both labeled correspondences and underlying structural similarities between the semantic and visual spaces, in order to find an initial linear mapping from the semantic space to the visual space. Then, the Iterative Procrustes Refinement (IPR) algorithm is used [46], in order to progressively align more pairs of points and then refine the mapping in a two stage alternating scheme.

Finally, when all points have been paired by the aforementioned procedure, a final mapping emerges, which we then use to project the semantic vectors to the visual space. Then each image is classified to it's closest projected semantic vector.

To the best of our knowledge, there are only a few methods in transductive zero-shot learning (ZSL) that utilize clustering as a core component of their approach [47], [48]. For this reason, our proposed method which leverages clustering for both seen and unseen classes, represents a contribution to this area. We note that our approach shares similarities with [49] that first introduced the idea of matching each class prototype to a particular center in a one-to-one fashion, using structural constraints of the visual and semantic spaces. However, we were unable to verify the aforementioned work's results in our experiments.

Problem Definition In this setting, we have N_s source labeled samples $D_s = \{(x_i^s, y_i^s)|i = 1, \ldots, N_s\}$, where x_i^s is an image and $y_i^s \in \mathcal{S} = \{1, 2, \ldots, S\}$ is the corresponding label within total S source classes. We are also given N_u unlabeled target samples $D_u = \{x_i^u|i = 1, \ldots, N_u\}$ that are from target classes $\mathcal{U} = \{S + 1, \ldots, S + U\}$. From definition 3.2.1 it follows that $\mathcal{S} \cap \mathcal{U} = \emptyset$, but the classes are associated in a semantic space $\mathcal{T}$ (see section 3.2.1). Our goal is to predict the labels $y_i^u \in \mathcal{U}$, given the images $x_i^u \in D_u$ and the prototype sets T^s and T^u (see section 3.2.1).

5.2 Clustering Step

In this step of our method, we use the SCAN algorithm in order to cluster all images [40] (seen and unseen classes). Concretely, Gasnbeke et al. [40] began by pretraining a neural network using the SimCLR algorithm [50] or the MOCO algorithm [51]. After the pretraining phase, it is observed that images belonging to the same class tend to be clustered together [40].

In our study, we opted not to engage in such algorithms, since they involve employing a sufficiently large batch size and a sizable neural network. Instead, we leveraged the pretrained ResNet101 network, which had undergone training on the extensive ImageNet dataset. This decision was made to utilize the established features and representations already captured by ResNet101. The histogram in figure 5.1 illustrates the distribution of correct neighbor matches among the 20 nearest neighbors of each sample in the AwA2 dataset [6]. Each bar on the histogram represents the frequency of samples for which a specific number of correct neighbors was identified. For instance, a bar at a particular x-value signifies the number of samples with that exact count of correct neighbors.

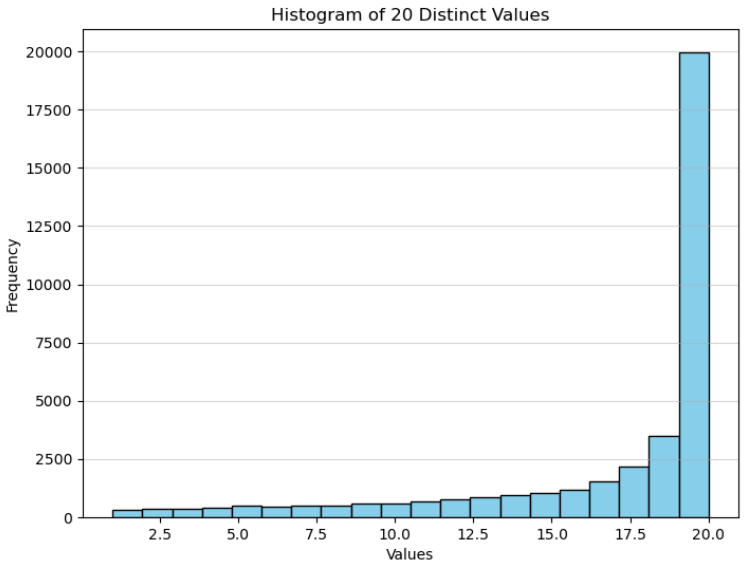


Figure 5.1: Neighboring samples tend to be instances of the same semantic class

5.2.1 Mining Nearest-Neighbors

In the previous section, we motivated the use of a large pretrained backbone in order to obtain some semantically meaningful visual features. However, naively applying K-means on the obtained features can lead to cluster degeneracy [40]. Furthermore, as can be seen in figure 5.2, the resulting features are not adequately linearly separable. This lack of clear separation indicates that simply applying K-means clustering in this context would be inappropriate and may lead to inaccurate and suboptimal results. Therefore we adopt the SCAN algorithm [40], opting for a better methodology.

Concretely, by using the ResNet101 model, we have obtained embeddings for the whole dataset $D = D_s \cup D_u$. Then, for each unlabeled sample $x_i \in D_u$, we mine it's nearest neighbors in the ResNet embedding space. In more detail, since we are assuming that all unlabeled samples

belong to the unseen classes, we only select neighbors from the dataset D_u and not the whole dataset $D_s \cup D_u$. Now for each sample sample x_i in the dataset $D_s \cup D_u$ we define the set N_{x_i} as follows: If the label of x_i is known, i.e $x_i \in D_s$ the set N_{x_i} contains all the samples which have the sample label as x_i. Otherwise if the sample is unlabeled, i.e $x_i \in D_u$, the set N_{x_i} contains the aforementioned K nearest neighbors. In other words:

$$N_{x_i} = \begin{cases} \{x : x \text{ and } x_i \text{ share the same label}\} & \text{if } x_i \in D_s \\ \{x : x \text{ is one of the } K \text{ nearest neighbors of } x_i\} & \text{if } x_i \in D_u \end{cases} \tag{5.1}$$

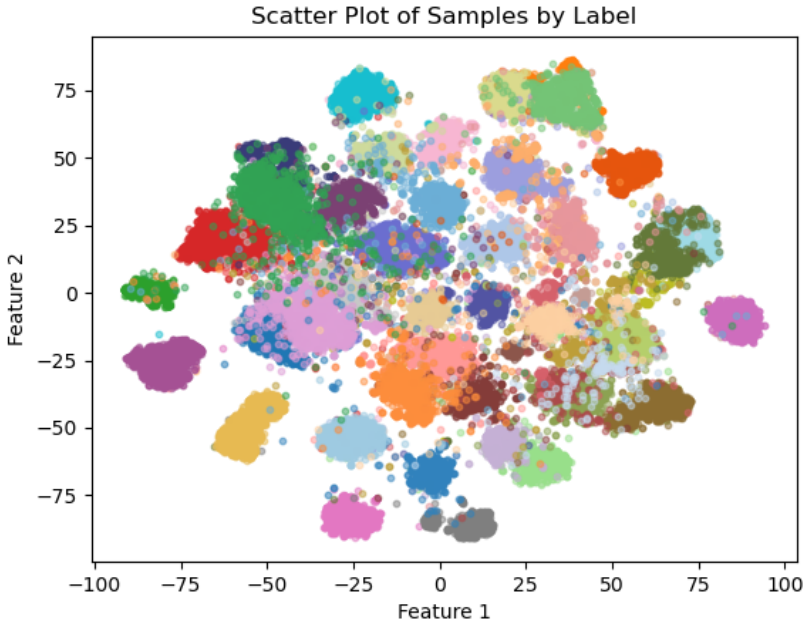


Figure 5.2: TSNE visualization of the AwA2 dataset [6]. Different classes are plotted with different color.

Loss function The goal is to learn a clustering function g_θ, which is described by a neural network with weights θ- that classifies a sample x_i and its mined neighbors N_{x_i} together [40]. g_θ terminates in a softmax function to perform a soft assignment over the clusters and therefore $g_\theta(x) \in [0,1]^{|\mathcal{S}\cup\mathcal{U}|}$. The probability of sample x_i being assigned to cluster k is denoted as $g_\theta^k(x_i)$. In order to learn the weights of g_θ the following optimization objective is minimized [40]

$$\begin{aligned} \mathcal{L}_u &= -\frac{1}{|D|}\sum_{x\in D}\sum_{y\in N_x}\log\left(g_\theta(x)^T \cdot g_\theta(y)\right) + \lambda \sum_{k\in\{1,\dots,|\mathcal{S}\cup\mathcal{U}|\}} g_\theta^{'k}\log g_\theta^{'k} \\ \text{where } g_\theta^{'k} &= \frac{1}{|D|}\sum_{x\in D} g_\theta^k(x) \end{aligned} \tag{5.2}$$

It is easy to see that the first term in equation 5.2 is minimized, if all samples $x \in D$ are assigned to the same cluster as their neighbors N_x, with the highest possible probability. Thus, the first term in Eq. 5.2 can be seen as a *consistency term*. The second term can be seen as the negative entropy of the random variable $g_\theta^{'k}$. It is well known that the entropy of a random variable is

maximized by the uniform distribution. Therefore the second term in eq. 5.2 prevents the model from collapsing, by assigning all the samples $x \in D$ to a few clusters.

A schematic representation of the aforementioned algorithm can be seen in Figure 5.3

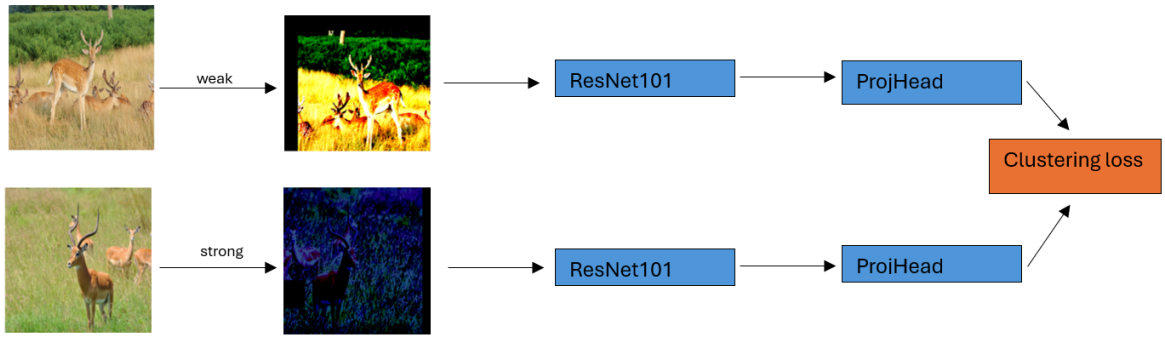


Figure 5.3: Semantic illustration of the clustering step of our algorithm. After a set of neighbors is selected for each image in the dataset, a random image and one of its neighbors are selected. The original image is weakly augmented, while its neighbor is strongly augmented. Both images are then passed to the ResNet101 backbone which is kept frozen during training. The resulting embeddings are then passed to the projection head, in order to calculate the clustering loss.

5.3 Representatives selection

After training the network g_θ, our goal is to select a point in the embedding space which emerges from g_θ, for each class. For this purpose, we discard the last fully connected layer of the network g_θ. Let us denote the remaining part as g_ϕ. Our first step is to calculate the embeddings of the seen class samples $E^s = \{g_\phi(x_1^s), \ldots, g_\phi(x_{N_s}^s)\}$ and the unseen class samples $E^u = \{g_\phi(x_1^u), \ldots, g_\phi(x_{N_u}^u)\}$. For each seen class, we select its mean, as the corresponding representative, i.e: for each $k \in \mathcal{S}$,

$$\mu_k = \frac{1}{|X_k|} \sum_{j:y_j^s=k} g_\phi(x_j^s), \qquad \text{where } X_k = \{x_i^s : y_i^s = k\} \tag{5.3}$$

Using equation 5.3, we can define the set $M^s = \{\mu_1, \ldots, \mu_S\}$, which we shall call *the seen class representatives*. Observe that $|M^s| = |\mathcal{S}|$. Since the labels of the set D_u are not known, we can not follow a similar way of reasoning. Therefore we use the K-means algorithm, initializing it with the real number of unseen classes. We denote the resulting centers as $M^u = \{\mu_{S+1}, \ldots, \mu_{S+U}\}$. We shall refer to the set M^u, as the *unseen class representatives*. Again, we observe that $|M^u| = |\mathcal{U}|$.

5.4 Finding A Bijective Mapping

Our final goal is to identify the correspondence between the set M^u and the set T^u (and therefore the set $\mathcal{U}$), given the correspondence between the sets M^s and the seen prototypes set T^s. This problem can be seen as a semi-supervised manifold alignment problem [52] and it is well studied in the field of Natural Language Processing, in the form of bilingual word translation [46], [45],

[53]. In our case, the sets M^s and T^s can be seen as aligned vector embeddings. More specifically, we are looking for a function f_θ, which maps each element of the unseen class prototypes set T^u to a unique element of the unseen class representatives M^u, given that each element of the seen class prototypes set T^s should be mapped to it's corresponding seen class representative of the set M^s.

In order to learn the parameters of the function f_θ we follow [46] utilizing unsupervised distribution matching and aligning known word pairs.

Unsupervised Distribution Matching Given the sets $M^s \cup M^u$ and $T^s \cup T^u$, the loss $\mathcal{L}_{\theta|D}$ aims to match the distribution of both embedding spaces. In particular, the parameters θ of f_θ are learned so as to fool a discriminator D, which in turn is trained to differentiate between the representatives $M^s \cup M^u$ and the mapped prototypes $f_\theta(T^s \cup T^u) = \{f_\theta(t_1^s), \ldots, f_\theta(t_S^s), f_\theta(t_{S+1}^u), \ldots, f_\theta(t_{S+U}^u)\}$. f_θ and D are alternatively optimized using the following objectives:

$$\begin{aligned} \mathcal{L}_{D|\theta} &= -\frac{1}{S+U} \sum_{x \in T^s \cup T^u} \log(1 - D(f_\theta(x))) - \frac{1}{S+U} \sum_{x \in M^s \cup M^u} \log D(x) \\ \mathcal{L}_{\theta|D} &= -\frac{1}{S+U} \sum_{x \in T^s \cup T^u} \log D(f_\theta(x)) \end{aligned} \tag{5.4}$$

Aligning known class representatives-known class prototypes Given the sets M^s and T^s, the function f_θ should map the elements of T^s, as "close" as possible to their corresponding seen representative. This can be formulated as:

$$\mathcal{L}_{\theta|alig} = -\frac{1}{S} \sum_{i=1}^{S} \frac{f_\theta(t_i)^T \cdot \mu_i}{||f_\theta(t_i)||_2 \cdot ||\mu_i||_2} \tag{5.5}$$

The final loss function for the mapping function f_θ is formulated as

$$\mathcal{L} = \mathcal{L}_{\theta|D} + \mathcal{L}_{\theta|alig} + \mathcal{L}_{D|\theta} \tag{5.6}$$

5.5 Iterative Mapping Refinement

A common method of improving the function f_θ is iteratively expanding the correspondences between the sets M^s and T^s and then refining the mapping function f_θ as a post-processing step [46], [45]. This refinement procedure first finds the pair of points in the sets M^u and T^u that are the closest matched by f_θ, and then updates f_θ by considering both the correspondences of T^s with M^s and the new pair of points.

To this end, the parameter vector found in section 5.4 is denoted as θ_0. Similarly, the sets M^u and T^u can be thought of as unmatched-initial sets and thus we shall denote them as T_0^u and M_0^u. In order to quantify the similarity between the points in M_0^u and T_0^u we consider the set $C_0 = M_0^u \times f_{\theta_0}(T_0^u) \subseteq \mathbb{R}^D \times \mathbb{R}^D$, where D is the output dimension of the neural network g_ϕ (see section 5.3). Let $sim(\cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a similarity function with:

$$sim(x, y) = \frac{x^T y}{||x||_2 ||y||_2}$$

Given the above measure of similarity, let us choose

$$(x_1, y_1) = \underset{(x,y)\in M_0^u \times T_0^u}{\arg\max} \; sim(C_0) \tag{5.7}$$

Our next step is to define the "new" sets $M_1^s = M_0^s \cup \{x_1\}$, $T_1^s = T_0^s \cup \{y_1\}$ and $M_1^u = M_0^u \backslash \{x_1\}$, $T_1^u = T_0^u \setminus \{y_1\}$. Afterwards we refine the mapping f_{θ_0}, given the new sets M_1^s and T_1^s. In more detail, we store the vectors found in the sets M_1^s and $f_{\theta_0}(T_1^s)$, in the matrices X_1 and Y_1 as columns respectively. The new parameters θ_1 are found by solving the optimization problem

$$\theta_1 = \arg\max_{\theta} \mathrm{Tr}(X_1^T Y_1) \tag{5.8}$$

where

$$\begin{aligned} X_1 &= \left(\frac{\mu_1}{||\mu_1||_2} \quad \cdots \quad \frac{\mu_S}{||\mu_S||_2} \quad \frac{x_1}{||x_1||_2} \right) \in \mathbb{R}^{D\times(S+1)} \\ Y_1 &= \left(\frac{f_\theta(t_1^s)}{||f_\theta(t_1^s)||} \quad \cdots \quad \frac{f_\theta(t_S^s)}{||f_\theta(t_S^s)||_2} \quad \frac{f_\theta(y_1)}{||f_\theta(y_1)||_2} \right) \in \mathbb{R}^{D\times(S+1)} \end{aligned} \tag{5.9}$$

Consequently the sets M_1^u and T_1^u are used in order to match a pair of points from the unseen class representatives and the unseen class prototypes, using f_{θ_1}. This process continues inductively until there is an $n \in \mathbb{N}$, such that $M_n^u = T_n^u = \emptyset$. Since $|M_0^u| = |T_0^u| = U$, it's easy to see that $n = U$. Therefore the final parameter vector chosen for the mapping f_θ is θ_U.

5.6 Classification

After the iterative mapping refinement step, we are given a learned parameter θ_U. In order to classify an image $x \in D_u$ we first project it in the visual space generated by the trained neural network g_ϕ and then find the nearest projected class prototype. In other words the predicted class $c^\star$ is given by

$$c^\star = \underset{c\in\mathcal{S}\cup\mathcal{U}}{\arg\max} \left\{ \frac{f_{\theta_U}(\pi(c))^T \cdot g_\phi(x)}{||f_{\theta_U}(\pi(c))||_2 ||g_\phi(x)||_2} \right\} \tag{5.10}$$

where $\pi(\cdot) : \mathcal{S} \cup \mathcal{U} \to \mathcal{T}$ is a class prototyping function (see section 3.2.1).

Chapter 6

Experimental Results

Contents

6.1 Datasets

To demonstrate the results of our method, experiments are conducted on two widely-used ZSL benchmark datasets; AwA2 [6], and CUB [7]. Following the same configuration as other methods, two different data split strategies are adopted:

- **Standard Split (SS)**: The standard seen/unseen class split was first proposed in [17] and then widely used in most ZSL works [54],[55],[56],[14].
- **Proposed Split (PS)**: This split was proposed by [18] in order to remove the overlapped ImageNet-1K dataset classes from target domain since it is used to pre-train the CNN, ResNet101 model.

6.1.1 Animals with Attributes 2 dataset

The Animals with Attributes 2 dataset (AWA2) [6] consists of 37,322 images of 50 classes of animals where 40 classes are seen for training while the remaining 10 classes are unseen during training. Each class is associated with an 85-dimension continuous attribute vector. In figure 6.1 four random images from four different classes are shown for visualization and understanding purposes.

(a) Image of an otter

(b) Image of a blue whale

(c) Image of an antelope

(d) Image of a skunk

Figure 6.1: Four randomly selected images from the AwA2 dataset [6]

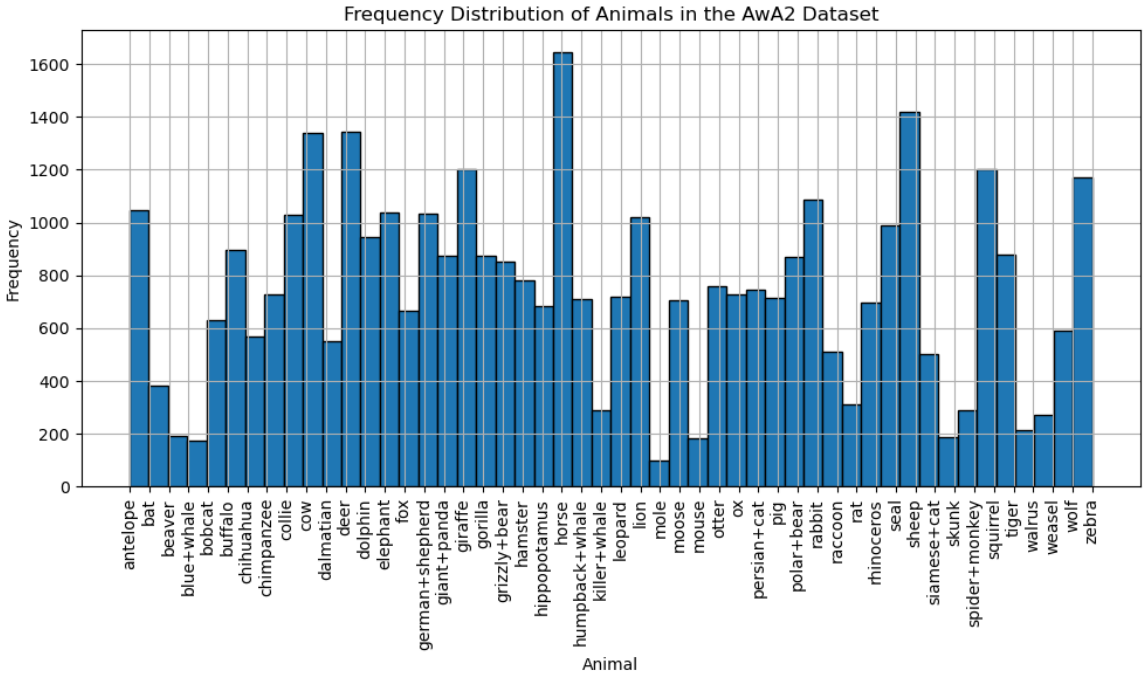


Figure 6.2: Frequency distribution of classes in the AwA2 dataset [6]

It is evident from figure 6.2 that the class distribution within the dataset is imbalanced, which poses a significant challenge for zero-shot learning (ZSL). In more detail, an imbalanced dataset can lead to biased model predictions. Classes with fewer instances may not be adequately represented during training, resulting in a poorer performance during testing. Overall, the class imbalance introduces complexities in both the learning process and the generalization capabilities of ZSL models.

6.1.2 CUB dataset

The Caltech-SCSD Birds-200-2011 (CUB) dataset contains 11,788 images of 200 fine-grained bird species. A standard split divides these bird species in to 150 seen classes and 50 unseen classes. For each class, a 312-dimension continuous attribute vector is provided.

Although the dataset's histogram (figure 6.4) indicates a seemingly balanced distribution of classes, it's essential to note that each category comprises no more than 60 samples. This relatively small sample size per class presents a considerable challenge for the task at hand. With such limited data per class, the model's ability to generalize effectively to unseen classes becomes more difficult, as it may struggle to capture the full variability and complexity of each category.

Moreover, the fine-grained nature of the dataset further compounds the difficulty of the task. As can be seen in figure 6.3 the classes are visually similar, requiring the model to discern subtle differences between them. This is a focal point which will be further discussed in the following chapters.

(a) Image of a Great Crested Flycatcher

(b) Image of a Gray Kingbird

(c) Image of a Chipping Sparrow

(d) Image of a Canada Warbler

Figure 6.3: Four randomly selected images from the CUB dataset [7]

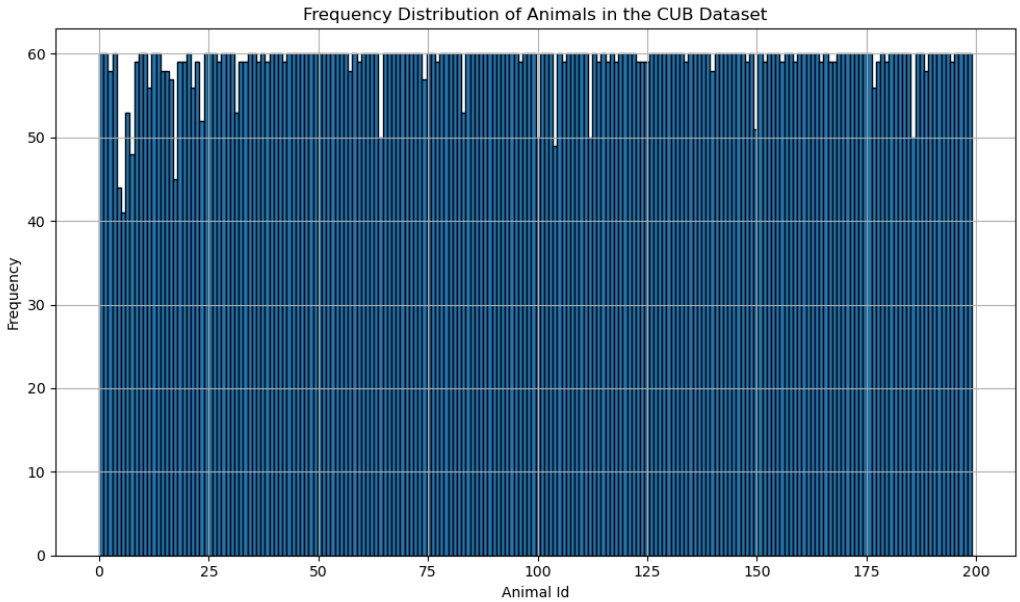


Figure 6.4: Frequency distribution of classes in the CUB dataset [7]

6.2 Evaluation Metrics

In our proposed method, we employ a multi-step approach to address the challenges of zero-shot learning. By isolating each stage of the pipeline, we can identify which specific components may be contributing to any discrepancies or sub-optimal performance in the final results. Assessing each step individually is essential for ensuring the effectiveness of the proposed approach for zero-shot learning. It allows us to identify and address potential issues, and iteratively refine the methodology for better results.

For the first step, which involves selecting nearest neighbors in the ResNet101 embedding space, we aim to evaluate the accuracy of this process in identifying similar instances within the dataset. Specifically, we seek to ensure that the selected neighbors share the same label as the instance under consideration. While there isn't a specific metric tailored to this step, we conduct visual evaluation of the resulting histograms to assess the efficacy of neighbor selection.

In the second step, we use a small projection head, in order to cluster images belonging to the same class together. In order to access the resulting clustering quality we use the accuracy (ACC) [102] and normalized mutual information (NMI) [103] metrics.

Finally, after the alignment step, the Mean Class Accuracy (MCA) [37] is used to evaluate the classification results.

Function	Implementation
g_θ	FC + ReLu + FC + ReLu + FC + ReLu + FC + Softmax
f_θ	FC + ReLu + FC + ReLu
D	FC + ReLu + FC + Softmax

Table 6.1: Implementation of the three non-linear mapping functions, discussed in chapter 5. FC stands for a fully connected layer while ReLu is the known activation function.

Implementation Details

For the first step of our method, we chose to use 20 neighbors as in most cases studied in [40]. The impact of varying the number of neighbors will be explored in detail in Section 6.4. The neural network implementations discussed in the previous chapter are summarized in Table 6.1. All neural networks were trained using the Adam optimizer [57] with a learning rate of 10^{-4}. The embedding space of g_ϕ, where classification occurs, is 2048-dimensional, consistent with the output dimension of the pre-trained ResNet101 model. The clustering step of our method leverages two kinds of augmentations: "weak" and "strong". Following [58], weak augmentation is a standard flip-and-shift augmentation. In more detail, we randomly flip images horizontally with a probability of 50% and we randomly translate images up to 12.5% vertically and horizontally. For "strong" augmentations RandAugment [59] is selected. This algorithm randomly selects transformations for each image, using a magnitude that controls the severity of all distortions which lies in a predefined range. The parameter λ weighting the entropy term in equation 5.2 is set to 2,5. Both the vectors found in the semantic space and visual space (the g_ϕ embedding space) are unit normalized before the alignment step (as discussed in chapter 5).

6.3 Nearest-Neighbors selection

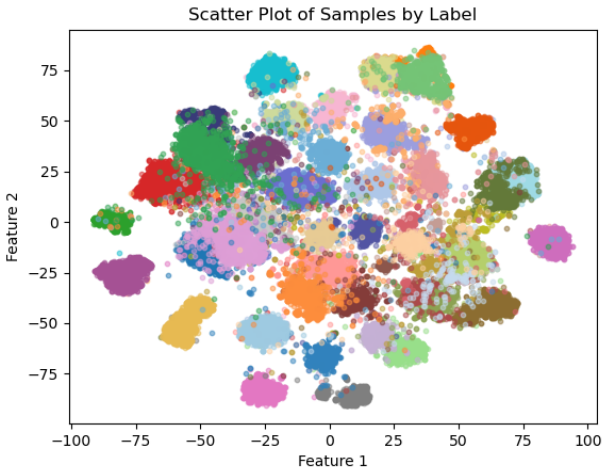


Figure 6.5: TSNE visualization of all samples from the AwA2 dataset [6].

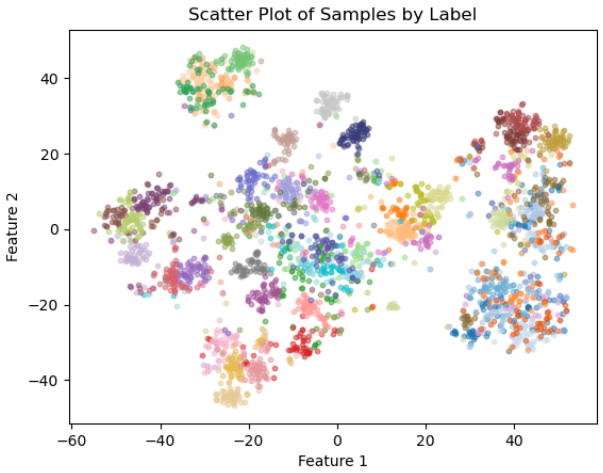


Figure 6.6: TSNE visualization of samples from 50 random classes from the CUB dataset.

To generate the nearest neighbor sets discussed in the previous section, we utilize the pretrained ResNet101 model [31] to project all images. This step is crucial in assessing the effectiveness of our approach, as it directly impacts the clustering results. We employ the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [104] to visualize the distribution of the projected samples. The degree to which samples of the same class tend to cluster together is of utmost importance, as it influences the final clustering outcomes. This is particularly relevant as we aim to cluster samples together with their neighbors, as described in Equation 5.2. Figure 6.5 presents the distribution of all samples from the AwA2 dataset, providing insight into the overall structure of the data. Additionally, Figure 6.6 showcases the t-SNE plot for a subset of 50 randomly selected classes from the CUB dataset, as visualizing all classes is impractical, due to their large number.

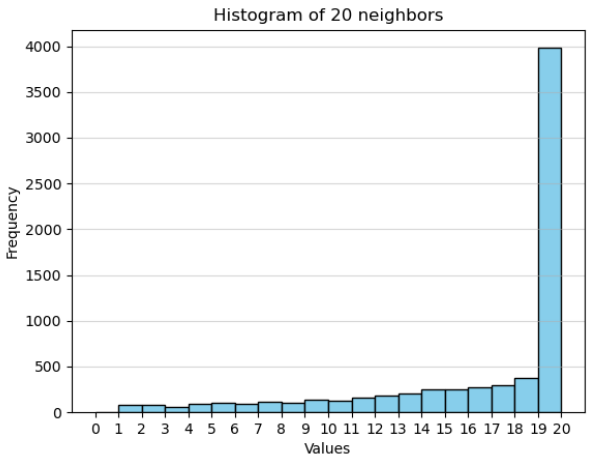


Figure 6.7: correct neighbors histogram of unlabeled instances from the AwA2 dataset

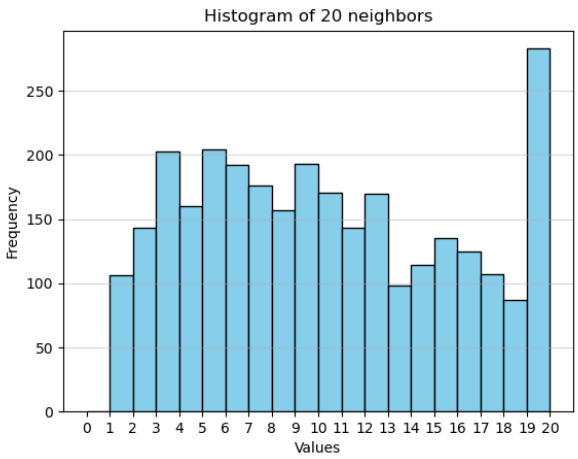


Figure 6.8: correct neighbors histogram of unlabeled instances from the CUB dataset.

Figures 6.7 and 6.8 provide insight into the degree to which unlabeled samples tend to share nearest neighbors with the same label, based on their 20 nearest neighbors. As we operate under the assumption that all unlabeled samples belong to unseen classes, we restrict our search space exclusively to the unlabeled set when mining nearest neighbors. This is because samples from the labeled set are expected to belong to different categories.

Figure 6.9 illustrates the mean and standard deviation of correct neighbors in both datasets. It is evident that in the CUB dataset, over half of the 20 neighbors selected for each image, do not belong in the same class with it. This finding will significantly impact the effectiveness of our proposed method, as will be discussed later on.

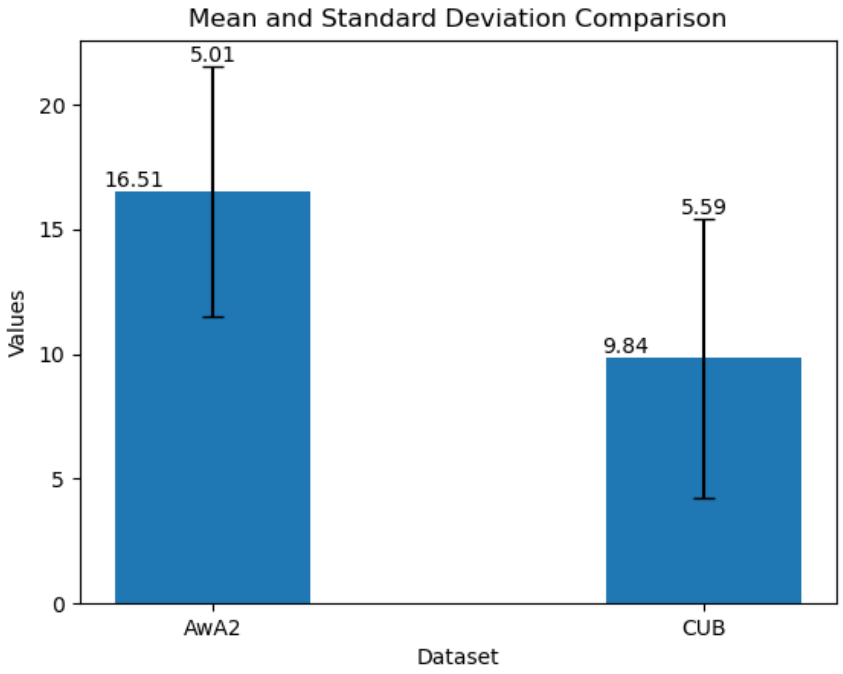


Figure 6.9: bar plot of the mean and standard deviation of correct neighbors for both the AwA2 and CUB datasets. The Mean is represented using a blue bar for each dataset and the Standard Deviation can be found in the black vertical line in each bar.

6.4 Clustering Step

This section investigates the effectiveness of the clustering step in both datasets, which is based on the results of the pretrained ResNet101 model discussed in the previous section. In each epoch, the clustering efficiency is improved by minimizing the loss function presented in Equation 5.2.

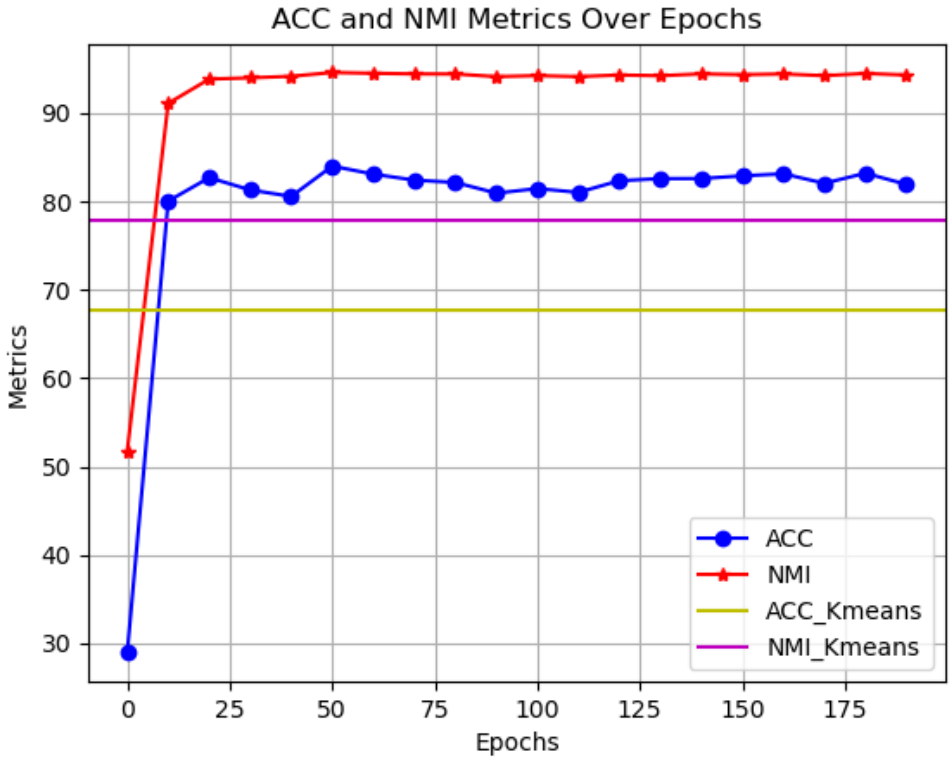


Figure 6.10: ACC and NMI Metrics Over Epochs on the AwA2 Dataset: The plot shows the clustering performance of a neural network (blue circles and red lines) and K-means clustering (yellow circles and purple lines) across 200 epochs.

Figures 6.10 and 6.11 present the clustering performance metrics of our neural network on the AwA2 and CUB datasets over 200 epochs. The metrics displayed are Accuracy (ACC) and Normalized Mutual Information (NMI).

The neural network's ACC metric, represented by blue circles, shows a significant improvement during the initial epochs. Specifically, there is a steep increase in accuracy from epoch 0 to epoch 20, after which the curve begins to stabilize. This trend indicates that the network rapidly learns the distinguishing features of the dataset in the early stages of training.

Similarly, the NMI metric, denoted by red crosses, follows a comparable trend with an initial sharp rise. By epoch 20, both ACC and NMI metrics reach a relatively high plateau, suggesting that the network has captured the essential relationships within the data. The subsequent epochs show minor fluctuations in both metrics, which may be attributed to the fine-tuning of the network's weights.

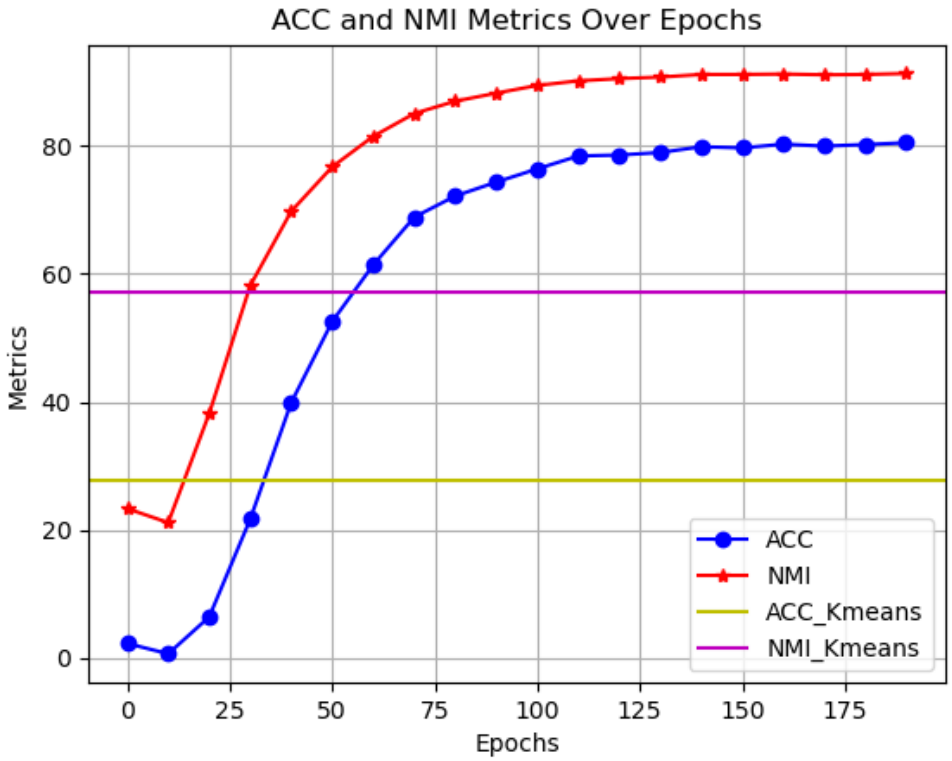


Figure 6.11: ACC and NMI Metrics Over Epochs on the CUB Dataset: The plot shows the clustering performance of a neural network (blue circles and red crosses) and K-means clustering (yellow circles and purple crosses) across 200 epochs.

In contrast with the dynamic learning process of the neural network, the performance metrics obtained through K-means clustering remain constant throughout the epochs. The ACC metric for K-means, shown by the yellow line, and the NMI metric, represented by the purple line, provide baseline performance measures for comparison.

The K-means clustering results exhibit significantly lower values compared to the neural network's metrics. The constant values highlight the nature of K-means as an unsupervised method that does not benefit from iterative learning across epochs. Unlike the neural network, K-means clustering applies a fixed algorithm that does not adapt or improve over time.

In Table 6.2, we present the pseudo upper bound accuracy metrics for the unseen classes, for both the AwA2 and CUB datasets, under the assumption of an optimal alignment between the semantic space prototypes and the visual space means. These pseudo upper bounds represent the best possible mean class accuracy (MCA) on the unseen classes if the alignment were perfect after the representatives selection step (see section 5.3). Specifically, the alignment is considered optimal under the assumption that the representatives chosen by the K-means algorithm remain fixed. In other words, if we were to re-implement the K-means algorithm, the arrangement of the representatives in the feature space would perhaps change, leading to a different set of upper bounds. Therefore, from a mathematical perspective, these are not true upper bounds, which is why we refer to them as pseudo upper bounds.

The table compares two approaches: Applying K-means clustering to the ResNet101 features

and applying K-Means to the trained neural network's embedding space (the g_ϕ embedding space discussed in section 5.2.1). The accuracy values shown for each method indicate the potential maximum performance of our model under ideal conditions, for the zero-shot classes. The results demonstrate a clear improvement in pseudo upper bound accuracy when using the neural network compared to the original ResNet101 space. For the AwA2 dataset, the pseudo upper bound accuracy with K-means is 87.68%, whereas it increases to 96.2% when using the neural network. Similarly, for the CUB dataset, the pseudo upper bound accuracy improves from 42.33% with K-means to 53.55% with the neural network.

These findings suggest that the neural network approach is more effective at capturing and aligning the semantic relationships in the feature space compared to the simpler K-means clustering method. The neural network likely benefits from its ability to fine-tune the extracted features from the ResNet101 backbone, creating a more discriminative feature space that better separates different classes, including those not seen during training.

The improvement in the pseudo upper bound accuracy metrics shows that using a neural network for clustering, instead of just applying K-means directly to ResNet101 features, gives a more accurate depiction of the underlying data structure.

Pseudo Upper Bounds	AwA2	CUB
K-means	87.68%	42.33%
Neural Network	96.2%	53.55%

Table 6.2: Pseudo upper bound comparison between simply applying K-means to the ResNet101 features and the features generated by the clustering network.

In summary, the results depicted in Table 6.2 highlight the advantage of incorporating a task-specific neural network on top of the pretrained ResNet101 backbone. This approach improves the feature space by increasing the potential classification accuracy for unseen classes in our transductive zero-shot learning scenario.

6.4.1 Effect of the number of neighbors selected

The relationship between the number of neighbors selected for the set N_x (see section 5.2 and 6.4 for the meaning of pseudo upper) and the ACC and NMI metrics for both datasets is illustrated in the Figures 6.12 and 6.13. Tables 6.3 and 6.4 show the exact numerical results achieved by the selection of different numbers of neighbors, when performing the clustering step.

	AwA2						
number of neighbors	2	5	10	20	50	100	200
ACC	80.36	85.53	84.11	84.54	82.82	83.10	83.47
NMI	93.65	94.92	94.80	95.18	94.52	94.24	94.71

Table 6.3: The exact value of ACC and NMI metrics achieved by selecting a different number of neighbors regarding the AwA2 dataset.

	CUB						
number of neighbors	2	5	10	15	30	50	80
ACC	74.09	75.59	74.47	75.73	76.26	73.89	73.16
NMI	87.12	88.60	88.34	88.68	88.87	88.45	87.93

Table 6.4: The exact value of ACC and NMI metrics achieved by selecting a different number of neighbors regarding the CUB dataset.

From the plots, we observe that the fluctuations in both ACC and NMI metrics are relatively minor across different number of neighbors. This indicates that the clustering method is robust to changes in this hyperparameter, maintaining stable performance across a broad range of neighbor values, as it was previously also noted in [40]. Such stability is positive, as it suggests that the method does not require precise tuning to achieve satisfactory results, making it easier to apply in various scenarios.

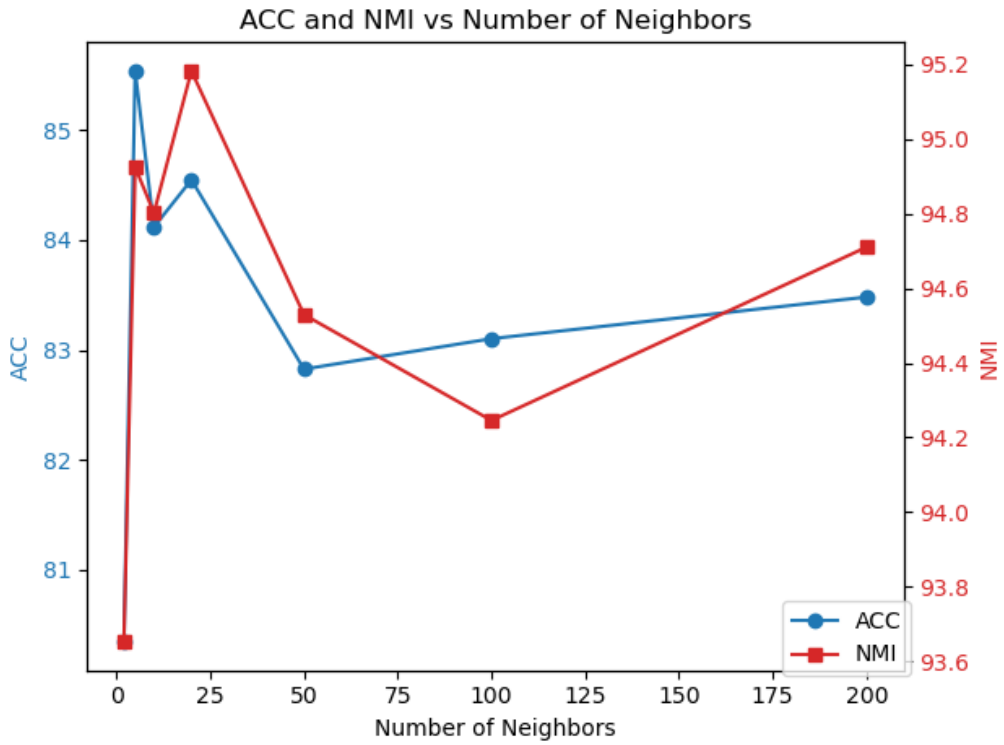


Figure 6.12: Performance metrics ACC and NMI as a function of the number of neighbors in the model for the AwA2 dataset. The blue line represents the ACC metric, and the red line represents the NMI metric.

Analyzing Figure 6.12 which corresponds to the AwA2 dataset, we observe that the ACC metric peaks around 20 to 30 neighbors, indicating an optimal range for this hyperparameter. While the NMI metric exhibits fluctuations, it is the ACC metric that aligns more closely with our final step. Furthermore, during training we also noted that the NMI metric fluctuates by 1-3% even with the same number of neighbors.

For the CUB dataset, both the ACC and NMI metrics peak at around 30 neighbors, as it can be seen in Figure 6.13. From Figure 6.4, it is evident that no class contains more than 60 samples. This limitation in class size explains why selecting a large number of neighbors is sub-optimal. This phenomenon is evident in the plot, where performance metrics decline when the number of neighbors surpasses 30. Therefore, keeping the number of neighbors within a moderate range ensures that the selected neighbors are more likely to belong to the same class.

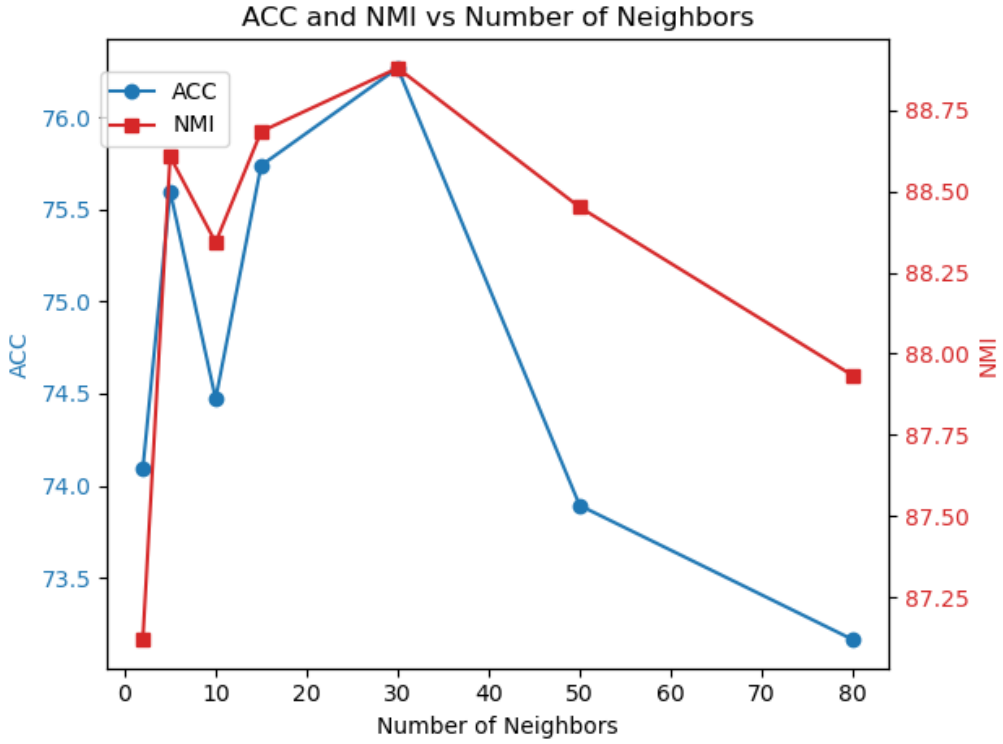


Figure 6.13: Performance metrics ACC and NMI as a function of the number of neighbors in the model for the CUB dataset. The blue line represents the ACC metric, and the red line represents the NMI metric.

Figure 6.5 illustrates a TSNE plot of all classes using the pretrained ResNet101 network. We observe that while there is some degree of clustering, the classes are not distinctly separated and there is considerable overlap among them. This overlap suggests that simply using this pretrained backbone might yield sub-optimal results. After applying our clustering step, however, a notable improvement in the class structure is observed.

The classes in Figure 6.14 become more compact and distinctly separated, demonstrating the effectiveness of the clustering method. This enhanced separation is crucial for downstream tasks, as it facilitates more accurate classification and better overall performance. The transition from overlapping to well-separated clusters underscores the capability of the algorithm to refine the embeddings, making the classes more distinguishable, which is a desirable state for the final step of our algorithm.

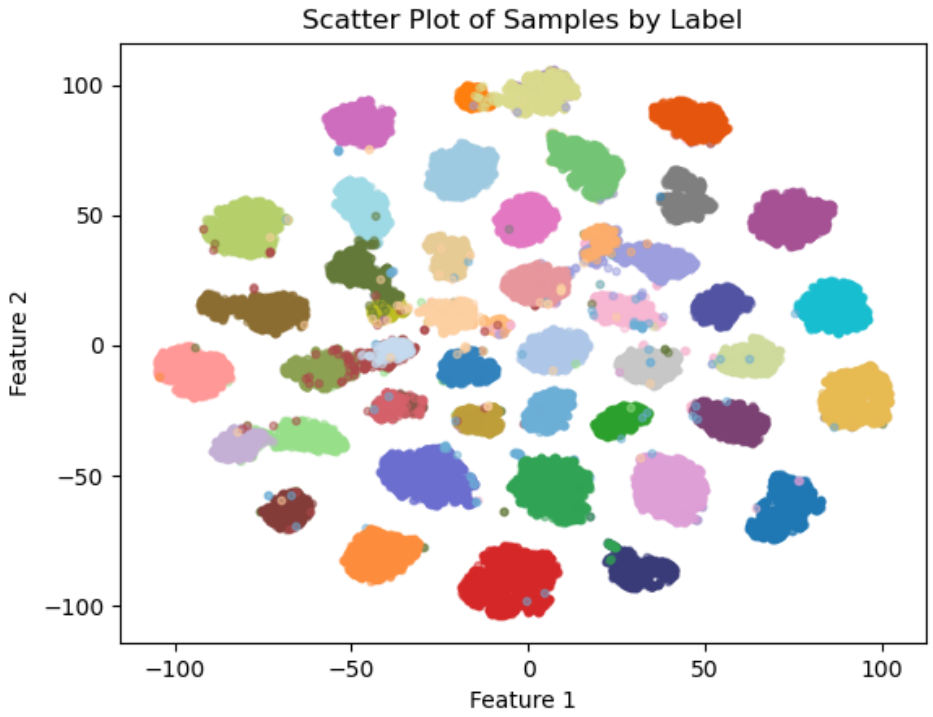


Figure 6.14: TSNE visualization of AwA2 dataset after training the clustering network. (This is the g_ϕ embedding space).

6.5 Classification Results

In the following two subsections, we report our final results under both the conventional and generalized Zero-Shot Learning settings. In conventional ZSL, all test instances are assumed to belong to the unseen classes, while in the Generalized ZSL scenario, test instances are assumed to belong to both the seen and unseen classes. Results for the conventional setting are reported under the standard split (SS) [17] and results for the generalized under the proposed split (PS) [18].

The Mean Class Accuracy (MCA) is the popular evaluation metric in Zero-Shot Learning [37]. In the conventional setting, MCA is only calculated on the test unseen domain ($Y = \mathcal{U}$).

$$MCA = \frac{1}{|\mathcal{U}|} \sum_{y \in \mathcal{U}} acc_y \tag{6.1}$$

where acc_y is the top-1 accuracy of class y from test unseen data. In the generalized settings, the label space comes from the union of the seen and unseen domains ($Y = \mathcal{S} \cup \mathcal{U}$) [37].

$$\mathcal{H} = \frac{2 * MCA_{ts} * MCA_{tu}}{MCA_{ts} + MCA_{tu}} \tag{6.2}$$

where MCA_{ts} is the mean class accuracy of the test seen data, MCA_{tu} is the mean class accuracy of the test unseen data and ($\mathcal{H}$) is their harmonic mean.

6.5.1 Classification Results for the AwA2 Dataset

Results under the Conventional Setting

Method		MCA
I	CONSE [60]	45.6
	DAP [61]	44.1
	SSE [62]	60.1
T	SE-ZSL [64]	68.3
	ALE_trans [65]	70.7
	GFZSL (G) [66]	78.6
	TEDE [16]	77.5
	Bi-VAEGAN (G) [67]	95.8
	ICPC (F) [37]	96.1
T	Ours	95.7

Table 6.5: ZSL performance comparison of different methods on the AwA2 dataset in the conventional ZSL setting, using the standard split [17]. (I) indicates that a method is inductive while (T) indicates that a method is transductive. (F) indicates that a method fine-tunes the ResNet101 backbone, and thus the comparison is not fair. (G) indicates that a method is generative. (-) implies that the corresponding results were not reported.

Table 6.5 provides a comparative analysis of various Zero-Shot learning methods on the AwA2 dataset in the conventional ZSL setting. The methods listed include both inductive (I) and transductive (T) approaches, with our method being transductive. It is noteworthy that the table consists exclusively of embedding based methods (see section 4.1.1), except for the Bi-VAEGAN [68] and GFZSL [66] methods, which are state-of-the-art generative approaches (as discussed in section 4.1.2). Our method demonstrates competitive performance, achieving an 95.7% on the AwA2 dataset, as reported over 50 runs to ensure robustness.

In comparison to the other methods, our approach outperforms several inductive methods and is on a par with the ICPC [37] and Bi-VAEGAN [68] methods, which are SOTA algorithms in Zero-Shot Learning.

Results under the Generalized Setting

Table 6.6 shows the performance of different zero-shot learning methods on the AwA2 dataset in the generalized ZSL setting. The methods are either inductive (I) or transductive (T), and are categorized as embedding-based (E) like ours, or generative (G). Our method's results are averaged over 50 runs. On the AwA2 dataset, our method achieves 89.8% for seen classes (MCA_{ts}), 73.2% for unseen classes (MCA_{tu}), and a harmonic mean (H) of 80.7%. It is noteworthy that our method outperforms all inductive based methods, which is after all anticipated, since we have access to more information (unlabeled samples from the zero-shot classes).

Finally, the reduction in mean class accuracy for zero-shot classes in the generalized setting compared to the conventional setting (Tables 6.5, 6.6), is justified by the bias problem, which was discussed in section 3.2.6.

Method		AwA2		
		MCA_{ts}	MCA_{tu}	$\mathcal{H}$
I	F-CLSWGAN (G) [69]	81.8	14.0	23.9
	SP-AEN (G) [70]	90.9	23.3	37.1
	DEM (E) [71]	86.4	30.5	45.1
	ALE (E) [72]	68.9	52.1	59.4
T	ALE_trans (E) [65]	73.0	12.6	21.5
	PREN (E) [75]	88.6	32.4	47.4
	f-VAEGAN (G) [42]	88.6	84.8	86.7
	Bi-VAEGAN (G) [68]	91.0	90.0	90.4
T	Ours (E)	89.8	73.2	80.7

Table 6.6: ZSL performance comparison of different methods on the AwA2 dataset in generalized ZSL setting, using the proposed split [18]. (I) indicates that a method is inductive while (T) indicates that a method is transductive. (G) indicates that a method is generative and (E) means that a method is embedding based (see section 4.1.1).

6.5.2 Limitation of the CUB Dataset

Impact of Correct Neighbors for the CUB dataset

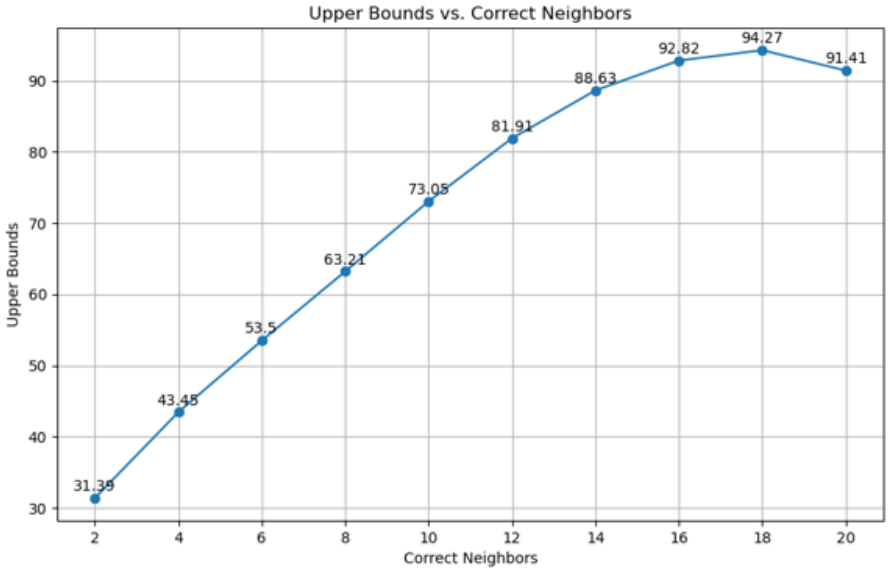


Figure 6.15: Mean Class Accuracy pseudo upper bounds vs. correct neighbors on the CUB dataset.

Figure 6.15 shows the pseudo upper bounds of our method on zero-shot classes within the CUB dataset, where our method's performance has been suboptimal, after the representatives selection step (see section 5.3 and 6.4 for the meaning of the term "pseudo upper bound"). These upper bounds represent the mean class accuracy of zero-shot classes, given scenarios where 2, 4, 6,

..., 20 neighbors are correctly identified for each sample out of the 20 selected. However, it is important to note that this graph simplifies the problem considerably due to several assumptions:

1. **Deterministic Number of Correct Neighbors**: In the provided analysis, the number of correct neighbors is fixed, resulting in zero variance. In practice, this is not the case, as the number of correct neighbors would vary as we can see in Figure 6.9.

2. **Random Selection of Correct Neighbors**: The analysis assumes that the correct neighbors for a sample are selected randomly each time. However, in reality, if for example y is a nearest neighbor of x, then the probability of x being a nearest neighbor of y is much higher.

Despite these simplifications, the analysis underscores the importance of increasing the number of correct neighbors in the initial step of our method. As illustrated in Figure 6.15, there is a clear positive correlation between the number of correct neighbors and the pseudo upper bounds of class accuracy. Improving the accuracy of neighbor selection can thus significantly enhance the performance of our zero-shot learning method on CUB.

As per our previous conversation, the method heavily relies on the initial number of correct neighbors selected in the original ResNet101 space. This is because the clustering network tries to assign an instance and its selected neighbors to the same cluster. As observed in Figure 6.9, in the CUB dataset, more than half of the neighbors selected belong to different classes, which obviously will lead to sub-optimal results. To resolve this problem we experimented with pretraining algorithms such as variations of the SimCLR [76] and MOCO [77] algorithms. However, these methods require large batch sizes, which exceeded our computational resources, resulting in the inability to improve the number of correct neighbors. Additionally, we explored other clustering approaches from the semi-supervised clustering bibliography [78], [79], [80]. Their results were far inferior to the clustering technique we applied, demonstrating considerably lower ACC and NMI metrics.

Results under the Conventional Setting

	Method	*MCA*
I	CONSE [60]	34.3
	DAP [61]	40.0
	SSE [62]	43.9
T	MFMR [63]	47.5
	SE-ZSL [64]	54.1
	ALE_trans [65]	54.5
	GFZSL (G) [66]	50.0
	TEDE [16]	67.8
	Bi-VAEGAN (G) [67]	76.8
	ICPC (F) [37]	84.1
T	Ours	48.0

Table 6.7: ZSL performance comparison of different methods on the CUB dataset, in conventional ZSL setting, using the standard split [17]. (I) indicates that a method is inductive while (T) indicates that a method is transductive. (F) indicates that a method fine-tunes the ResNet101 backbone, and thus the comparison is not fair. (G) indicates that a method is generative.

Results under the Generalized Setting

Method		CUB		
		MCA_{ts}	MCA_{tu}	$\mathcal{H}$
I	F-CLSWGAN (G) [69]	33.1	21.8	26.3
	SP-AEN (G) [70]	38.6	24.9	30.3
	DEM (E) [71]	25.6	34.3	20.5
	ALE (E) [72]	36.6	42.6	39.4
	LisGAN (G) [73]	37.8	42.9	40.2
T	DSRL (E) [74]	25.0	17.7	20.7
	GFZSL (G) [66]	45.8	24.9	32.2
	ALE_trans (E) [65]	45.1	23.5	30.9
	PREN (E) [75]	55.8	35.2	43.1
	f-VAEGAN (G) [42]	65.1	61.4	63.2
	Bi-VAEGAN (G) [68]	71.7	71.2	71.5
T	Ours (E)	51.6	46.7	49.0

Table 6.8: ZSL performance comparison of different methods on the CUB dataset in generalized ZSL setting, using the proposed split [18]. (I) indicates that a method is inductive while (T) indicates that a method is transductive. (G) indicates that a method is generative and (E) means that a method is embedding based (see section 4.1.1).

6.6 Discussion

In this study, we achieved notable results on the AwA2 dataset, competing effectively with Bi-VAEGAN [68], a state-of-the-art (SOTA) generative method in the conventional setting. Specifically, in this setting, our method was outperformed by the Bi-VAEGAN [68] method by only 0.1%. However, our results on the CUB dataset were relatively lower compared to other SOTA methods. We attribute this to the original ResNet101 space, where we selected the nearest neighbors for each unlabeled sample. Histogram 6.8 revealed that many samples have a large number of neighbors from different classes, causing classification issues. This limitation is inherent as we use ResNet101 as our backbone, which is pretrained on the ImageNet dataset. Since ImageNet contains only a few classes of birds, the model struggles to extract discriminative features for the large number of different bird species in the CUB dataset, leading to a sub-optimal embedding space.

Attempts to pretrain the backbone using algorithms such as SimCLR [76] and MOCO [77] were unsuccessful due to limited computational resources. These algorithms could have enhanced the clustering of samples from different classes, leading to an improved initialization of our algorithm. To further substantiate our point, a post-clustering pseudo upper bound analysis (see Figure 6.15) showed that for the CUB dataset, an increased number of correct neighbors would lead to a significant increase in the pseudo upper bound of the Mean Class Accuracy (MCA) metric on the unknown classes. The real pseudo upper bounds of our method are 96.2% for the AwA2 dataset and 53.55% for the CUB dataset, while our results are 95.7% and 48% respectively. This implies that the alignment step is effective, and the results of our method are restricted by the structure of the ResNet101 space, especially for the CUB dataset.

Chapter 7

Conclusion and Discussion

Contents

7.1 Conclusion

Zero-shot learning (ZSL) addresses a significant challenge in machine learning: the classification of instances from classes that were not encountered during the training phase. This capability is crucial for real-world applications where it is impractical to obtain labeled data for every possible class. Examples include identifying rare species in ecological studies, detecting emerging cyber threats, and recognizing novel objects in autonomous systems.

To tackle the ZSL problem, in chapter 4 we presented two primary approaches: embedding-based methods and generative-based methods. Embedding-based methods aim to project both visual features and semantic representations into a shared latent space, allowing for the comparison and classification of unseen classes based on their proximity within this space. Generative methods, on the other hand, synthesize visual features for unseen classes using models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), transforming the ZSL task into a conventional supervised learning problem with synthetic data.

In this thesis, we propose a novel approach for transductive ZSL by integrating a state-of-the-art unsupervised clustering algorithm, specifically the SCAN [40] algorithm, and adapting it to suit our needs. To the best of our knowledge, this is the first attempt to bridge the gap between advancements in the unsupervised clustering literature and zero-shot learning. Our method leverages the strengths of SCAN to enhance the clustering of visual features, ensuring better classification of unseen classes

Our proposed method demonstrates performance on par with other state-of-the-art ZSL algorithms on the AwA2 dataset, achieving these results without the need for end-to-end training or fine-tuning of the ResNet101 backbone.

In summary, while our method shows competitive performance on the AwA2 dataset, it is constrained by the quality of initial neighbor selection in the ResNet101 space, particularly evident in the CUB dataset results. This limitation underscores the need for further improvements through enhanced pretraining techniques or more effective clustering algorithms.

7.2 Future Work

The current study highlights several areas for future research to further enhance the performance and applicability of our proposed zero-shot learning (ZSL) method.

Firstly, we aim to implement pretraining algorithms such as SimCLR and MOCO, which have shown great promise in improving the clustering of samples from different classes. However, these algorithms require significant computational resources that exceed our current capabilities. Securing the necessary computational resources will enable us to leverage these advanced pretraining techniques, potentially leading to better initialization and improved clustering results, thereby enhancing the overall performance of our ZSL model.

Secondly, we plan to develop our method into an end-to-end framework rather than maintaining the current modular structure. Currently, our approach involves two distinct steps: the clustering step and the subsequent bijective mapping to connect the semantic space to the visual space. Integrating these steps into a end-to-end model would reduce potential sources of error, and likely improve the model's accuracy. An initial attempt at creating an end-to-end method, is briefly mentioned in the Appendix.

Finally, we intend to explore more state-of-the-art (SOTA) clustering algorithms. By experimenting with a broader range of advanced clustering techniques, we aim to identify methods

that can further enhance the clustering quality and, consequently, the performance of our ZSL model.

Chapter 8

Appendix

Analysis of an End-to-End Method

In our pursuit of improving the zero-shot learning (ZSL) framework, we experimented with an end-to-end approach designed to integrate clustering into a unified model. Despite the theoretical advantages of reducing potential error sources and enhancing model accuracy, our implementation of this end-to-end method did not yield the desired results.

The end-to-end method we explored comprises two distinct model: a clustering network and a discriminator. The network's role is to project samples from the visual space into the semantic space. The discriminator operates within the semantic space, distinguishing between the projected visual samples and the class prototypes that reside there. The training of the clustering network is guided by an objective function with three key aims: (1) assigning an image and its mined neighbors to the same cluster, (2) projecting images with known labels close to their corresponding class prototypes, and (3) fooling the discriminator. Then a pseudo-labeling strategy is adopted, to progressively incorporate the testing samples with pseudo labels into the training scheme [16].

Problem Definition In this setting, we have N_s source labeled samples $D_s = \{(x_i^s, y_i^s)|i = 1, \ldots, N_s\}$, where x_i^s is an image and $y_i^s \in \mathcal{S} = \{1, 2, \ldots, S\}$ is the corresponding label within total S source classes. We are also given N_u unlabeled target samples $D_u = \{x_i^u|i = 1, \ldots, N_u\}$ that are from target classes $\mathcal{U} = \{S + 1, \ldots, S + U\}$. From definition 3.2.1 it follows that $\mathcal{S} \cap \mathcal{U} = \emptyset$, but the classes are associated in a semantic space $\mathcal{T}$ (see section 3.2.1). Our goal is to predict the labels $y_i^u \in \mathcal{U}$, given the images $x_i^u \in D_u$ and the prototype sets T^s and T^u (see section 3.2.1).

By using the ResNet101 model, we have obtained embeddings for the whole dataset $D = D_s \cup D_u$. Then, for each unlabeled sample $x_i \in D_u$, we mine it's nearest neighbors in the ResNet embedding space. In more detail, since we are assuming that all unlabeled samples belong to the unseen classes, we only select neighbors from the dataset D_u and not the whole dataset $D_s \cup D_u$. Now for each sample sample x_i in the dataset $D_s \cup D_u$ we define the set N_{x_i} as follows: If the label of x_i is known, i.e $x_i \in D_s$ the set N_{x_i} contains all the samples which have the sample label as x_i. Otherwise if the sample is unlabeled, i.e $x_i \in D_u$, the set N_{x_i} contains the aforementioned K nearest neighbors. In other words:

$$N_{x_i} = \begin{cases} \{x : x \text{ and } x_i \text{ share the same label}\} & \text{if } x_i \in D_s \\ \{x : x \text{ is one of the } K \text{ nearest neighbors of } x_i\} & \text{if } x_i \in D_u \end{cases} \tag{8.1}$$

The goal is to learn a clustering function g_θ, which is described by a neural network with weights θ- that classifies a sample x_i and its mined neighbors N_{x_i} together. g_θ terminates in a softmax function to perform a soft assignment over the clusters and therefore $g_\theta(x) \in [0,1]^{|\mathcal{S}\cup\mathcal{U}|}$. The probability of sample x_i being assigned to cluster k is denoted as $g_\theta^k(x_i)$. In order to have access to the embeddings generated by the clustering network and not only the resulting probabilities, we denote the clustering network without the final linear and softmax layers as g_ϕ.

Finally there will be a discriminator $D : \mathcal{T} \to [0,1]$ which will be trained to distinguish between the projected visual features and the class prototypes.

Optimizing the discriminator D Given the clustering networks weights θ, ϕ

$$\mathcal{L}_{D|W,\theta,\phi} = -\frac{1}{N_s+N_u}\sum_{x\in D_s\cup D_U}\log(1-D(g_\phi(x))) - \frac{1}{S+U}\sum_{y\in\mathcal{S}\cup\mathcal{U}}\log(D(\pi(y)))$$

Optimizing the clustering network θ, ϕ Given the discriminator D

$$\mathcal{L}_{\theta,\phi|W,D} = \underbrace{-\frac{1}{N_s+N_u}\sum_{x\in D}\sum_{y\in N_x}\log(g_\theta(x)^T g_\theta(y))}_{\text{clustering loss}} + \underbrace{\lambda\sum_{k\in\{1,\dots,|\mathcal{S}\cup\mathcal{U}|\}} g_\theta^{'k}\log g_\theta^{'k}}_{\text{entropy over the clusters}}$$
$$\underbrace{-\frac{1}{N_s}\sum_{(x_i^s,y_i^s)\in D_s} f_s(g_\phi(x_i^s),\pi(y_i^s))}_{\substack{\text{project the labeled samples}\\ \text{close to their corresponding}\\ \text{class prototype}}} \underbrace{-\frac{1}{N_s+N_u}\sum_{x\in D_s\cup D_u}\log D(g_\phi(x))}_{\text{fool the discriminator}}$$

where $g_\theta^{'k} = \frac{1}{|D|}\sum_{x\in D} g_\theta^k(x)$

Pseudo-Labeling After training the aforementioned models, a pseudo-labeling strategy is adopted [16]. Specifically, all samples are projected into the semantic space. Unlabeled images closest to the unseen class prototypes are selected and assigned these labels as pseudo-labels. Then the two models are retrained using the augmented training dataset D_s^{aug}. Specifically we select the top-M high-confidence testing samples that are predicted to belong to an unseen class, using the similarity measure $f_s(g_\phi(x_i^u), \pi(y))$, where $y \in \mathcal{U}$. This process continues inductively, until there are no instances predicted to belong to the unseen classes.

Results In the inductive setting, the results of our end-to-end method were not as promising as anticipated. Specifically, the accuracy on the AwA2 dataset ranged from 60% to 65%, while for the CUB dataset, the accuracy was between 31% and 34%. These outcomes indicate that our approach, despite its theoretical advantages, struggled to achieve competitive performance in practice, particularly on the CUB dataset.

Bibliography

[1] A. El-Shahat, *Advanced applications for artificial neural networks*. BoD–Books on Demand, 2018.

[2] M. M. Taye, "Understanding of machine learning with deep learning: architectures, workflow, applications and future directions," *Computers*, vol. 12, no. 5, p. 91, 2023.

[3] W. contributors, "Gradient descent." https://en.wikipedia.org/wiki/Gradient_descent, 2024. Accessed: 2024-06-20.

[4] K. P. Murphy, *Probabilistic machine learning: an introduction*. MIT press, 2022.

[5] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C. P. Lim, X.-Z. Wang, and Q. J. Wu, "A review of generalized zero-shot learning methods," *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[6] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251–2265, 2019.

[7] C. Wah, S. Branson, P. Perona, and S. Belongie, "Multiclass recognition and part localization with humans in the loop," in *2011 International Conference on Computer Vision*, pp. 2524–2531, IEEE, 2011.

[8] A. Schulz, "25 years ago: Deep blue beats kasparov," 2021.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] E. Howell, "Pooling layers for convolutional neural networks (cnn)." https://towardsdatascience.com/pooling-layers-for-convolutional-neural-networks-cnn-6cf24806 2024. Accessed: 2024-06-20.

[11] M. Mishra, "Convolutional neural networks explained." https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, 2020. Accessed: 2024-06-20.

[12] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 951–958, IEEE, 2009.

[13] F. Zhang and G. Shi, "Co-representation network for generalized zero-shot learning," in *International Conference on Machine Learning*, pp. 7434–7443, PMLR, 2019.

[14] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2021–2030, 2017.

[15] C. M. Bishop, "Pattern recognition and machine learning," *Springer google schola*, vol. 2, pp. 1122–1128, 2006.

[16] L. Zhang, P. Wang, L. Liu, C. Shen, W. Wei, Y. Zhang, and A. Van Den Hengel, "Towards effective deep embedding for zero-shot learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2843–2852, 2020.

[17] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 951–958, IEEE, 2009.

[18] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[19] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[21] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2012.

[22] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pp. 196–201, IEEE, 2011.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[24] C. F. G. D. Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," *ACM Computing Surveys (CSUR)*, vol. 54, no. 10s, pp. 1–25, 2022.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] H. H. Aghdam, E. J. Heravi, and D. Puig, "Recognizing traffic signs using a practical deep neural network," in *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 1*, pp. 399–410, Springer, 2015.

[27] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1360–1371, 2005.

[28] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung, "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural computation*, vol. 22, no. 2, pp. 511–538, 2010.

[29] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[31] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016.

[32] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[33] Z. Fu, T. Xiang, E. Kodirov, and S. Gong, "Zero-shot object recognition by semantic manifold distance," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2635–2644, 2015.

[34] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019.

[35] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 52–68, Springer, 2016.

[36] Y. Annadani and S. Biswas, "Preserving semantic relations for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7603–7612, 2018.

[37] H. Yang, B. Sun, B. Li, C. Yang, Z. Wang, J. Chen, L. Wang, and H. Li, "Iterative class prototype calibration for transductive zero-shot learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 3, pp. 1236–1246, 2022.

[38] M. Durgadevi *et al.*, "Generative adversarial network (gan): a general review on different variants of gan and applications," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, pp. 1–8, IEEE, 2021.

[39] D. P. Kingma, M. Welling, *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[40] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, "Scan: Learning to classify images without labels," in *European conference on computer vision*, pp. 268–285, Springer, 2020.

[41] G. Chochlakis, "Using artificial neural networks for zero-shot learning," 2020.

[42] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, "f-vaegan-d2: A feature generating framework for any-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10275–10284, 2019.

[43] S. Rahman, S. Khan, and N. Barnes, "Transductive learning for zero-shot object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6082–6091, 2019.

[44] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern recognition*, vol. 90, pp. 119–133, 2019.

[45] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," *arXiv preprint arXiv:1710.04087*, 2017.

[46] B. Patra, J. R. A. Moniz, S. Garg, M. R. Gormley, and G. Neubig, "Bilingual lexicon induction with semi-supervision in non-isometric embedding spaces," *arXiv preprint arXiv:1908.06625*, 2019.

[47] Q. Yue, J. Cui, L. Bai, J. Liang, and J. Liang, "A zero-shot learning boosting framework via concept-constrained clustering," *Pattern Recognition*, vol. 145, p. 109937, 2024.

[48] S. M. Shojaee and M. S. Baghshah, "Semi-supervised zero-shot learning by a clustering-based approach," *arXiv preprint arXiv:1605.09016*, 2016.

[49] Z. Wan, D. Chen, Y. Li, X. Yan, J. Zhang, Y. Yu, and J. Liao, "Transductive zero-shot learning with visual structure constraint," *Advances in neural information processing systems*, vol. 32, 2019.

[50] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[51] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[52] J. Ham, D. Lee, and L. Saul, "Semisupervised alignment of manifolds," in *International Workshop on Artificial Intelligence and Statistics*, pp. 120–127, PMLR, 2005.

[53] M. Artetxe, G. Labaka, and E. Agirre, "Learning bilingual word embeddings with (almost) no bilingual data," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 451–462, 2017.

[54] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 819–826, 2013.

[55] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, "Evaluation of output embeddings for fine-grained image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2927–2936, 2015.

[56] J. Song, C. Shen, Y. Yang, Y. Liu, and M. Song, "Transductive unbiased embedding for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1024–1033, 2018.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in neural information processing systems*, vol. 33, pp. 596–608, 2020.

[59] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

[60] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, "Zero-shot learning by convex combination of semantic embeddings," *arXiv preprint arXiv:1312.5650*, 2013.

[61] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 951–958, IEEE, 2009.

[62] Z. Zhang and V. Saligrama, "Zero-shot learning via semantic similarity embedding," in *Proceedings of the IEEE international conference on computer vision*, pp. 4166–4174, 2015.

[63] X. Xu, F. Shen, Y. Yang, D. Zhang, H. Tao Shen, and J. Song, "Matrix tri-factorization with manifold regularizations for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3798–3807, 2017.

[64] V. K. Verma, G. Arora, A. Mishra, and P. Rai, "Generalized zero-shot learning via synthesized examples," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4281–4289, 2018.

[65] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 819–826, 2013.

[66] V. K. Verma and P. Rai, "A simple exponential family framework for zero-shot learning," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part II 10*, pp. 792–808, Springer, 2017.

[67] Z. Wang, Y. Hao, T. Mu, O. Li, S. Wang, and X. He, "Bi-directional distribution alignment for transductive zero-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19893–19902, 2023.

[68] Z. Wang, Y. Hao, T. Mu, O. Li, S. Wang, and X. He, "Bi-directional distribution alignment for transductive zero-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19893–19902, 2023.

[69] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata, "Feature generating networks for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5542–5551, 2018.

[70] L. Chen, H. Zhang, J. Xiao, W. Liu, and S.-F. Chang, "Zero-shot visual recognition using semantics-preserving adversarial embedding networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1043–1052, 2018.

[71] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2021–2030, 2017.

[72] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 819–826, 2013.

[73] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, and Z. Huang, "Leveraging the invariant side of generative zero-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7402–7411, 2019.

[74] M. Ye and Y. Guo, "Zero-shot classification with discriminative semantic representation learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7140–7148, 2017.

[75] M. Ye and Y. Guo, "Progressive ensemble networks for zero-shot recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11728–11736, 2019.

[76] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[77] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[78] B. Sun, P. Zhou, L. Du, and X. Li, "Active deep image clustering," *Knowledge-Based Systems*, vol. 252, p. 109346, 2022.

[79] M. Śmieja, Ł. Struski, and M. A. Figueiredo, "A classification-based approach to semi-supervised clustering with pairwise constraints," *Neural Networks*, vol. 127, pp. 193–203, 2020.

[80] Y. Ren, K. Hu, X. Dai, L. Pan, S. C. Hoi, and Z. Xu, "Semi-supervised deep embedded clustering," *Neurocomputing*, vol. 325, pp. 121–130, 2019.

[81] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[82] J. Weizenbaum, "Contextual understanding by computers," *Communications of the ACM*, vol. 10, no. 8, pp. 474–480, 1967.

[83] O. Kuusi and S. Heinonen, "Scenarios from artificial narrow intelligence to artificial general intelligence—reviewing the results of the international work/technology 2050 study," *World Futures Review*, vol. 14, no. 1, pp. 65–79, 2022.

[84] S. K. Gaikwad, B. W. Gawali, and P. Yannawar, "A review on speech recognition technique," *International Journal of Computer Applications*, vol. 10, no. 3, pp. 16–24, 2010.

[85] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain, "Machine translation using deep learning: An overview," in *2017 international conference on computer, communications and electronics (comptelix)*, pp. 162–167, IEEE, 2017.

[86] E. Y. Chang, "Examining gpt-4: Capabilities, implications and future directions," in *The 10th International Conference on Computational Science and Computational Intelligence*, 2023.

[87] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[88] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *Journal of chemical information and modeling*, vol. 55, no. 2, pp. 263–274, 2015.

[89] Z. Li, E. Gavves, T. Mensink, and C. G. Snoek, "Attributes make sense on segmented objects," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pp. 350–365, Springer, 2014.

[90] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[91] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[92] L. Zhang, P. Wang, L. Liu, C. Shen, W. Wei, Y. Zhang, and A. Van Den Hengel, "Towards effective deep embedding for zero-shot learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 2843–2852, 2020.

[93] Y. L. Cacheux, H. L. Borgne, and M. Crucianu, "Modeling inter and intra-class relations in the triplet loss for zero-shot learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10333–10342, 2019.

[94] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

[95] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *International conference on machine learning*, pp. 2152–2161, PMLR, 2015.

[96] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3174–3183, 2017.

[97] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 69–77, 2016.

[98] K. Li, M. R. Min, and Y. Fu, "Rethinking zero-shot learning: A conditional visual classification perspective," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3583–3592, 2019.

[99] M. Bucher, S. Herbin, and F. Jurie, "Improving semantic embedding consistency by metric learning for zero-shot classiffication," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pp. 730–746, Springer, 2016.

[100] J. Song, C. Shen, Y. Yang, Y. Liu, and M. Song, "Transductive unbiased embedding for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1024–1033, 2018.

[101] M. Elhoseiny and M. Elfeki, "Creativity inspired zero-shot learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5784–5793, 2019.

[102] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, pp. 478–487, PMLR, 2016.

[103] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002.

[104] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.