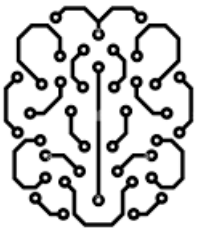
National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Speech and Language Processing Group

Auto-Compressing Networks

Diploma Thesis

of

EVANGELOS DOROVATAS



Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Athens, June 2025

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

Auto-Compressing Networks

Diploma Thesis

of

EVANGELOS DOROVATAS

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Approved by the examination committee on 01/07/2025.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Alexandros Potamianos	Athanasios Voulodimos	Athanasios Rodogiannis
Associate Professor, NTUA	Assistant Professor, NTUA	Associate Professor, NTUA

Athens, June 2025

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

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Evangelos Dorovatas

Electrical & Computer Engineer Graduate, NTUA

to my family and friends

Περίληψη

Τα βαθιά νευρωνικά δίκτυα με υπολειμματικές συνδέσεις (Υπολλειματικά Νευρωνικά Δίκτυα - ΥΝΔ) έχουν επιδείξει αξιοσημείωτη επιτυχία σε διάφορους τομείς, αλλά η αύξηση του βάθους τους συχνά εισάγει υπολογιστικό κόστος χωρίς αντίστοιχες βελτιώσεις στην ποιότητα των αναπαραστάσεων που μαθαίνουν. Σε αυτήν την εργασία, παρουσιάζουμε τα Αυτο-Συμπιεζόμενα Δίκτυα (ΑΣΔ), μια αρχιτεκτονική παραλλαγή των ΥΝΔ όπου οι σύντομες υπολειμματικές συνδέσεις αντικαθιστόνται από προσθετικές μακρινές εμπρόσθιες συνδέσεις από κάθε στρώμα στην έξοδο. Αναλύοντας την δομή που επιφέρει αυτήν η τροποποίηση στο δίκτυο, αποκαλύπτουμε μια μοναδική ιδιότητα των ΑΣΔ που ονομάζουμε *αυτο-συμπίεση*—την ικανότητα ενός δικτύου να συμπιέζει την πληροφορία κατά τη διάρκεια της εκπαίδευσης σε ένα υποσύνολο των συνολικών στρωμάτων του, αυτόματα μέσω αρχιτεκτονικού σχεδιασμού. Εξετάζοντας την αρχιτεκτονική, δείχνουμε θεωρητικά ότι αυτή η ιδιότητα προκύπτει από μοτίβα εκπαίδευσης στρώμα με στρώμα (layer-wise) στα ΑΣΔ, όπου τα στρώματα χρησιμοποιούνται δυναμικά κατά τη διάρκεια της εκπαίδευσης βάσει των απαιτήσεων του προβλήματος στο οποίο εκπαιδεύονται. Επιπλέον εξηγούμε μαθηματικά και δείχνουμε εμπειρικά ότι η αυτο-συμπίεση δεν συμβαίνει στα ΥΝΔ ή στα απλα Εμπρόσθια Νευρωνικά Δίκτυα (ΕΝΔ). Στη συνέχεια, διαπιστώνουμε πειραματικά ότι τα ΑΣΔ παρουσιάζουν ενισχυμένη ανθεκτικότητα στο θόρυβο σε σύγκριση με τα υπολειμματικά δίκτυα, ανώτερη επίδοση σε περιβάλλοντα χαμηλών δεδομένων, βελτιωμένες ικανότητες μεταφοράς μάθησης, και ξεχνούν σημαντικά λιγότερο (catastrophic forgetting), συγκεκριμένα έως και 18% λιγότερο, σε συνθήκες διαρκούς μάθησης, προτείνοντας ότι μαθαίνουν αναπαραστάσεις που γενικεύουν καλύτερα παρά τη χρήση λιγότερων παραμέτρων. Τα πειραματικά αποτελέσματά της παρούσας εργασίας δείχνουν 30-80% αρχιτεκτονική συμπίεση με διατήρηση υψηλής επίδοσης σε προβλήματα όρασης και φυσικής γλώσσας όταν τα ΑΣΔ ενσωματώνονται σε διάφορες αρχιτεκτονικές όπως Vision transformers, MLP-mixers, και BERT. Επιπλέον, δείχνουμε ότι όταν συνδυάζουμε τα ΑΣΔ με παραδοσιακές τεχνικές κλαδέματος (pruning), το κέρδος συμπίεσης διατηρείται και η αυτο-συμπίεση λειτουργεί συμπληρωματικά. Συνολικά, τα ευρήματα της παρούσας εργασίας καθιστούν τα ΑΣΔ ως μια πρακτική προσέγγιση για την ανάπτυξη αποδοτι-κών νευρωνικών αρχιτεκτονικών που προσαρμόζουν αυτόματα το υπολογιστικό τους αποτύπωμα στην πολυπλοκότητα του εκάστοτε προβλήματος.

Λέξεις Κλειδιά

Βαθιά Μάθηση, Αρχιτεκτονικές Νευρωνικών Δικτύων, Μάθηση Αναπαραστάσεων, Υπολλειματική Μάθηση, Διαρκής Μάθηση, Συμπίεση

Abstract

Deep neural networks with short residual connections have demonstrated remarkable success across domains, but increasing depth often introduces computational redundancy without corresponding improvements in representation quality, while potentially harming generalization in certain cases. In this work, we introduce Auto-Compressing Networks (ACNs), an architectural variant where additive long feedforward connections from each layer to the output replace traditional short residual connections. By analyzing the distinct dynamics induced by this modification, we reveal a unique property we coin as *auto-compression*—the ability of a network to organically compress information during training with gradient descent, through architectural design alone. Through auto-compression, information is dynamically "pushed" into early layers during training, enhancing their representational quality and revealing potential redundancy in deeper ones, resulting in a sparse yet powerful network at inference. We theoretically show that this property emerges from layer-wise training patterns present in ACNs, where layers are dynamically utilized during training based on task requirements. We also find that ACNs exhibit enhanced noise robustness compared to residual networks, superior performance in low-data settings, improved transfer learning capabilities, and mitigate catastrophic forgetting suggesting that they learn representations that generalize better despite using fewer parameters. Our results demonstrate up to 18% reduction in catastrophic forgetting and 30-80% architectural compression while maintaining accuracy across vision transformers, MLP-mixers, and BERT architectures. Furthermore, we demonstrate that when coupling ACNs with traditional pruning techniques, the compression gain persists and enables significantly better sparsity-performance trade-offs compared to conventional architectures. These findings establish ACNs as a practical approach to developing efficient neural architectures that automatically adapt their computational footprint to task complexity, while learning robust representations suitable for noisy real-world tasks and continual learning scenarios.

Keywords

Deep Learning, Neural Network Architectures, Representation Learning, Residual Learning, Continual Learning, Compression

Ευχαριστίες

Παραδίδοντας την διπλωματική μου εργασία, παραδίδω ουσιαστικά και την φοιτητική μου ταυτότητα και ιδιότητα, που με συντρόφευε τα τελευταία 6 χρόνια. Μέσα στα χρόνια αυτά έζησα μοναδικές εμπειρίες, γνώρισα απίθανους ανθρώπους, εξελίχθηκα και άλλαξα, εξερεύνησα τον κόσμο και τον εαυτό μου. Λίγο πριν αφήσω πίσω την φοιτητική μου ιδιότητα, λοιπόν, θέλω να ευχαριστήσω και αναφέρω όλους και όλα που επηρέασαν την πορεία μου αυτά τα χρόνια.

Πρώτα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Αλέξανδρο Ποταμιάνο, για την εμπιστοσύνη και τις αμέτρητες συζητήσεις και συμβουλές του, που πραγματικά καθόρισαν τον τρόπο σκέψης μου και φούντωσαν τον ενθουσιασμό μου για την έρευνα. Ήταν πραγματικά χαρά και τιμή μου αυτή η συνεργασία και αποδείχθηκε μια απολαυστική εμπειρία. Ακόμη θα ήθελα να ευχαριστήσω τον δρ. Γεώργιο Παρασκευόπουλο, που με βοήθησε σημαντικά και με κατεύθυνε, ιδιαίτερα στα πρώτα μου βήματα στην έρευνα. Συνεχίζοντας, θα ήθελα να πω ένα πολύ μεγάλο ευχαριστώ σε όλους τους συνοδοιπόρους μου όλα αυτά τα χρόνια, γνωστούς και γνωστές, φίλους και φίλες, αδέρφια μου στα αλήθεια, που έπαιξαν καθοριστικό ρόλο στο ποιος είμαι και το πως έφτασα εδώ, μοιράστηκαν μοναδικές εμπειρίες μαζί μου και παρέα αναρριχηθήκαμε από την εφηβεία στην ενηλικίωση. Πριν κλείσω, πρέπει να αναφέρω και την πόλη μου, την Αθήνα, μέσα στην οποία γαλουχήθηκα και υπήρξε η πιστή και σταθερή σύντροφος μου μέσα στα φοιτητικά μου χρόνια. Τέλος, δεν μπορώ να παραλείψω την οικογένεια μου, στην οποία οφείλω ένα μεγάλο ευχαριστώ για την διαρκή στήριξη και εμπιστοσύνη προς το πρόσωπο μου, και χωρίς την οποία δεν θα ήμουν ο ίδιος.

Κλείνοντας, το μέλλον μοιάζει αβέβαιο και συναρπαστικό, παραθέτω λοιπόν ένα στιχάκι που μοιάζει σχετικό:

"Τώρα στο τζάμι ένα καράβι εσκάρωσα
κι ένα του Μαγκρ στιχάκι έχω σκαλίσει:
«Τι θλίψη στα ταξίδια κρύβεται άπειρη!»
και εγώ για ένα ταξίδι έχω κινήσει".

Athens, June 2025

Evangelos Dorovatas

Table of Contents

List of Figures

List of Tables

Εκτεταμένη Περίληψη στα Ελληνικά

0.1 Εισαγωγή

Τα βαθιά νευρωνικά δίκτυα έχουν επιτύχει εντυπωσιακά αποτελέσματα σε ένα ευρύ φάσμα εφαρμογών, από την αναγνώριση εικόνων έως την επεξεργασία φυσικής γλώσσας (10; 11; 12). Παρά την επιτυχία τους, αυτά τα μοντέλα είναι υπολογιστικά πολύ ακριβά με συχνές αποτυχίες στη γενίκευση σε νέα πεδία διαφορετικά από τα συνολά εκπαίδευσης τους, σε αντίθεση με τα βιολογικά νευρωνικά δίκτυα που είναι σημαντικά πιο αποδοτικά και ευέλικτα. Η εισαγωγή των Υπολειμματικών Δικτύων (ResNets) αποτέλεσε σημαντικό βήμα προς την κατεύθυνση της αύξησης της επίδοσης και λειτουργικότηταςτων νευρωνικών δικτύων και αποτέλεσε την βάση της Βαθιάς Μάθησης, επιτρέποντας την εκπαίδευση πολύ βαθέων αρχιτεκτονικών με εκπληκτικά αποτελέσματα. Από την άλλη, αρκετές φορές η αύξηση του βάθους δεν οδηγεί σε αντίστοιχα ωφέλη ως προς την ποιότητα των αναπαραστάσεων (13), ενώ αυξάνει σημαντικά το υπολογιστικό κόστος, ενώ μπορεί να επιφέρει και αρνητική επίδραση στην ικανότητα γενίκευσης του δικτύου (14; 15). Στην παρούσα εργασία, παρουσιάζουμε τα Αυτο-Συμπιεζόμενα Δίκτυα (ΑΣΔ), μια νέα αρχιτεκτονική παραλλαγή των υπολλειματικών δικτύων, που αξιοποιεί την ιδιότητα της αυτο-συμπίεσης για να επιτύχει καλύτερη αποδοτικότητα και γενίκευση, χρησιμοποιώντας το βάθος δυναμικά ανάλογα με την φύση και την δυσκολία του προβλήματος. Στην συνέχεια, αρχικά συζητάμε το θεωρητικό υπόβαθρο απαραίτητο για την κατανόηση της εργασίας και έπειτα παρουσιάζουμε και αναλύουμε λεπτομερώς τα ΑΣΔ και τις ιδιότητές τους, τόσο θεωρητικά όσο και πρακτικά, συγκρίνοντας τα με τα Υπολλειματικά Νευρωνικά Δίκτυα (ΥΝΔ) και τα Εμπρόσθια Νευρωνικά Δίκτυα (ΕΝΔ).

0.2 Θεωρητικό Υπόβαθρο

Σε αυτήν την ενότητα, παρουσιάζουμε συνοπτικά το θεωρητικό υπόβαθρο που είναι σχετικό και χρήσιμο για την παρούσα εργασία. Πρώτα κάνουμε μια σύντομη ανασκόπηση στο πεδίο της Μηχανικής Μάθησης, στην συνέχεια συζητάμε το πεδίο της Μάθησης Αναπαραστάσεων και τέλος παρουσιάζουμε τις βασικές ιδέες και δουλειές των αρχιτεκτονικών με πολλαπλά μονοπάτια.

0.2.1 Μηχανική Μάθηση

Τύποι Μάθησης

- Η *επιβλεπόμενη μάθηση* (16) βασίζεται σε ένα σύνολο δεδομένων που περιέχει ζεύγη εισόδου-εξόδου, $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Σκοπός είναι η εκμάθηση της συσχέτισης

μεταξύ της εισόδου x και της επιθυμητής εξόδου y (ονομάζεται και ετικέτα), ώστε το μοντέλο να προβλέπει σωστά τις εξόδους για νέες, άγνωστες εισόδους. Εάν οι έξοδοι είναι κατηγορικές (διακριτές τιμές), πρόκειται για πρόβλημα ταξινόμησης, ενώ αν είναι συνεχείς, πρόκειται για παλινδρόμηση.

- Η *μη επιβλεπόμενη μάθηση* (16) χρησιμοποιεί μόνο δεδομένα εισόδου, χωρίς αντίστοιχες ετικέτες, δηλαδή $D = \{x_1, \ldots, x_n\}$. Ο στόχος είναι η ανακάλυψη εσωτερικών δομών, σχέσεων ή συσχετίσεων στα δεδομένα, μέσω τεχνικών όπως η ομαδοποίηση (clustering) ή η μείωση διαστάσεων.

- Η *ενισχυτική μάθηση* (17) στηρίζεται σε έμμεση ανατροφοδότηση, χωρίς την ύπαρξη ακριβών ετικετών αλλά συνήθως θετικού ή αρνητικού σήματος (επιβράβευση ή τιμωρία). Ο αλγόριθμος μαθαίνει μέσω δοκιμών και σφαλμάτων, λαμβάνοντας ενίσχυση (π.χ. θετική ή αρνητική ανταμοιβή) ανάλογα με την απόδοση των ενεργειών του σε ένα περιβάλλον.

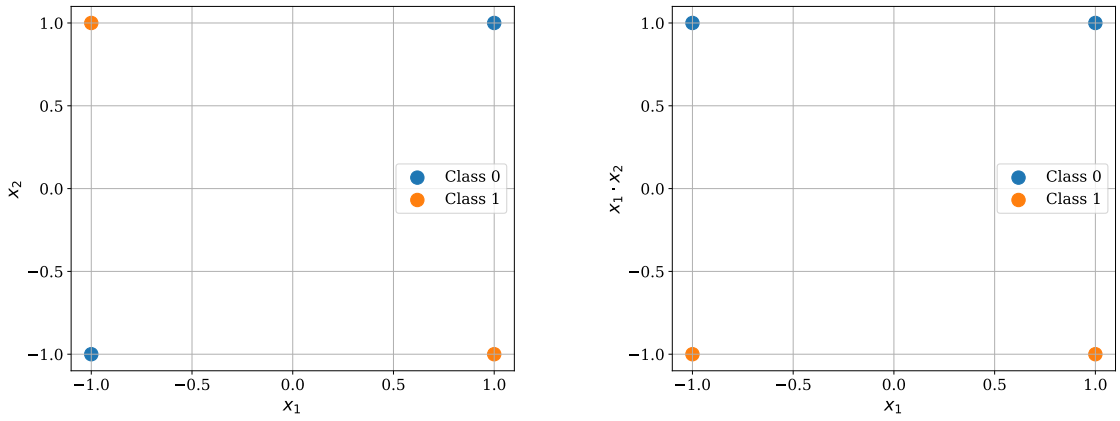
Διακρίσεις Μοντέλων Μια συνήθης διάκριση των μοντέλων μηχανικής μάθησης είναι σε *παραμετρικά* και *μη-παραμετρικά* μοντέλα (16). Στα παραμετρικά μοντέλα, ο αριθμός των παραμέτρων είναι σταθερός και ανεξάρτητος από το πλήθος των δεδομένων εκπαίδευσης. Είναι γενικά αποδοτικά σε υπολογιστικό επίπεδο και εύκολα στην εκπαίδευση, αλλά μπορεί να υστερούν σε εκφραστικότητα. Αντίθετα, τα μη-παραμετρικά μοντέλα προσαρμόζουν τη δομή τους ανάλογα με τον όγκο των δεδομένων, αποκτώντας μεγαλύτερη ευελιξία αλλά και αυξημένες υπολογιστικές απαιτήσεις.

Μια δεύτερη σημαντική διάκριση είναι μεταξύ *γραμμικών* και *μη-γραμμικών* μοντέλων. Τα γραμμικά μοντέλα ορίζουν την έξοδο ως γραμμική συνάρτηση των εισόδων και είναι κατάλληλα για προβλήματα με απλές, γραμμικά διαχωρίσιμες σχέσεις. Παρέχουν καλή κατανόηση και είναι υπολογιστικά αποδοτικά, αλλά αδυνατούν να αποδώσουν πολύπλοκα μοτίβα. Τα μη-γραμμικά μοντέλα, όπως τα νευρωνικά δίκτυα, είναι ικανά να μοντελοποιήσουν πολύπλοκες εξαρτήσεις, με τίμημα όμως την ανάγκη για πιο σύνθετους αλγορίθμους εκπαίδευσης και αυξημένους πόρους.

Μη Γραμμικοί Μετασχηματισμοί Εισόδου Ένα κλασικό παράδειγμα που αναδεικνύει την ανάγκη για μη-γραμμικούς μετασχηματισμούς (δηλαδή μη-γραμμικά μοντέλα) είναι το πρόβλημα ταξινόμησης XOR. Έστω είσοδοι $\mathbf{x} = (x_1, x_2) \in \{-1, 1\}^2$ και ετικέτα $y = x_1 \oplus x_2$, όπου $\oplus$ είναι η λογική πράξη XOR. Τα ζεύγη εισόδου-εξόδου είναι:

$$(-1,-1) \mapsto 0, \quad (-1,1) \mapsto 1, \quad (1,-1) \mapsto 1, \quad (1,1) \mapsto 0$$

Το πρόβλημα δεν είναι γραμμικά διαχωρίσιμο στο αρχικό χώρο, δηλαδή δεν υπάρχει γραμμικό μοντέλο (ευθεία) που να ταξινομεί σωστά όλα τα σημεία, όπως φαίνεται στο Σχήμα 1. Ωστόσο, με έναν μη-γραμμικό μετασχηματισμό, όπως $\phi(\mathbf{x}) = (x_1, x_1 \cdot x_2)$, τα σημεία απεικονίζονται σε νέο χώρο χαρακτηριστικών όπου γίνονται γραμμικά διαχωρίσιμα και μπορούν να διαχωρίστούν επιτυχώς. ΤΟ XOR αποτελεί ένα από παράδειγμα που αναδεικνύει τη σημασία των μη-γραμμικών μετασχηματισμών στη μηχανική μάθηση.



Σχήμα 1. *Το Πρόβλημα XOR.*

Εκπαίδευση Μοντέλων Η εκπαίδευση περιλαμβάνει τη χρήση μιας συνάρτησης κόστους (loss function) που μετρά την απόδοση του μοντέλου και ενός αλγορίθμου εκπαίδευσης που προσαρμόζει τις παραμέτρους. Για παράδειγμα, σε ένα πρόβλημα επιβλεπόμενης μάθησης, η συνάρτηση κόστους μετράει πόσο κοντά η έξοδος του μοντέλου είναι στην σωστή ετικέτα. Ο αλγόριθμος εκπαίδευσης ουσιαστικά εκφράζει μαθηματικά το πως πρέπει να μεταβληθούν οι παράμετροι του μοντέλου (μιλώντας για παραμετρικά μοντέλα) έτσι ώστε να βελτωθεί η συνάρτηση κόστους.

Τεχνητά Νευρωνικά Δίκτυα Τα τεχνητά νευρωνικά δίκτυα (ΤΝΔ) αποτελούν βασική κατηγορία παραμετρικών μοντέλων στη μηχανική μάθηση, εμπνευσμένα από τη δομή των βιολογικών νευρωνικών δικτύων στον εγκέφαλο (18; 19). Ένα ΤΝΔ αποτελείται από κόμβους (νευρώνες) και σταθμισμένες συνδέσεις, οργανωμένες σε στρώματα. Στην απλούστερη μορφή τους, τα εμπρόσθια νευρωνικά δίκτυα (ΕΝΔ), η είσοδος $x \in \mathbb{R}^d$ περνά διαδοχικά από στρώματα όπου εφαρμόζονται δύο βασικές λειτουργίες: **άθροισμα σταθμισμένο από τα βάρη των ακμών** και **μη-γραμμικές συναρτήσεις ενεργοποίησης**.

Για παράδειγμα, ένα μονοστρωματικό νευρωνικό δίκτυο υπολογίζει:

$$y = f(x) = \phi\left(\sum_{i=1}^{d} \theta_i x_i + b\right) = \phi(\theta^\top x + b),$$

όπου θ_i είναι τα βάρη, b η παράμετρος μετατόπισης (bias) και ϕ η μη-γραμμική συνάρτηση ενεργοποίησης, π.χ. `ReLU`, `sigmoid` ή `tanh`.

Οι παράμετροι θ "μαθαίνονται" από τα δεδομένα κατά την εκπαίδευση, ώστε το δίκτυο να παράγει σωστές προβλέψεις (δηλαδή σωστές ετικέτες σε σενάριο επιβλεπόμενης μάθησης) $\hat{y}_i = f(x_i; \theta)$ για κάθε δείγμα (x_i, y_i) του συνόλου εκπαίδευσης.

Εκπαίδευση Νευρωνικών Δικτύων Η εκπαίδευση νευρωνικών δικτύων γίνεται κυρίως με τη μέθοδο της **οπισθοδιάδοσης** (backpropagation) (20), η οποία χρησιμοποιεί τον κανόνα της αλυσίδας για να υπολογίσει τις παραγώγους της συνάρτησης κόστους ως προς τις παραμέτρους του δικτύου. Περιλαμβάνει ένα προωθητικό πέρασμα (forward pass), κατά το οποίο υπολογίζεται η έξοδος του μοντέλου και η συνάρτηση κόστους, και ένα οπισθοδρομικό

(backward pass), κατά το οποίο διαδίδονται οι παράγωγοι και πραγματοποιείται ανανέωση των βαρών.

Βαθιά Μάθηση Η βαθιά μάθηση εστιάζει στη χρήση *βαθιών* νευρωνικών δικτύων με πολλά στρώματα για την εκμάθηση πολύπλοκων μη γραμμικών συναρτήσεων (21). Το πλεονέκτημα τους είναι ότι μαθαίνουν αυτόματα από τα δεδομένα ιεραρχίες αναπαραστάσεων, οδηγώντας σε καλή απόδοση σε περίπλοκα προβλήματα του πραγματικού κόσμου όπως η αναγνώριση εικόνας ή η κατανόηση γλώσσας. Οι κυριότερες αρχιτεκτονικές νευρωνικών δικτύων που χρησιμοποιούνται στην Βαθιά Μάθηση είναι:

- **Εμπρόσθια Νευρωνικά Δίκτυα (ΕΝΔ)** (18; 19): Αποτελούνται από διαδοχικά πλήρως συνδεδεμένα στρώματα (συνδέσεις από κάθε νευρώνα σε κάθε νευρώνα του επόμενου στρώματος), όπου κάθε επίπεδο εφαρμόζει έναν γραμμικό μετασχηματισμό και μια μη-γραμμική συνάρτηση ενεργοποίησης (όπως προηγουμένως). Επιτρέπουν την εκμάθηση σύνθετων μη-γραμμικών συναρτήσεων και αποτελούν τη βάση των πιο σύνθετων αρχιτεκτονικών.

- **Συνελικτικά Νευρωνικά Δίκτυα (CNNs)** (22; 23; 24): Εξειδικευμένα για είσοδο με δομή πλέγματος (όπως εικόνες), εφαρμόζουν τοπικές συνελίξεις αντί για πλήρεις συνδέσεις μεταξύ στρωμάτων. Ενσωματώνουν δύο βασικές αρχιτεκτονικές ιδέες:

  - *Τοπικότητα*: Οι νευρώνες συνδέονται μόνο με μικρές περιοχές της εισόδου. Παίρνουν έμπνευση από την φύση των εικόνων, όπου συχνά τα αντικείμενα είναι εντοπισμένα σε μια περιοχή της εικόνας (εισόδου).
  - *Κοινή χρήση βαρών (weight sharing)*: Το ίδιο φίλτρο εφαρμόζεται σε πολλαπλές θέσεις (μέρη) της εισόδου (εικόνας), ψάχνοντας για συγκεκριμένα χαρακτηριστικά (π.χ. ακμές) κατά μήκος της εικόνας.

  Τα δίκτυα αυτά, μέσα από διαδοχικά στρώματα, μαθαίνουν ιεραρχικές αναπαραστάσεις από χαμηλού επιπέδου γεωμετρικά χαρακτηριστικά (π.χ. ακμές) έως και πιο υψηλού επιπέδου αντικείμενα (π.χ. πρόσωπα). Αξίζει να σημειωθεί, ότι η επιτυχία του συνελικτικού δικτύου *AlexNet* (24) στο ImageNet (25) αποτέλεσε κομβική στιγμή στην πρόοδο της βαθιάς μάθησης.

- **Αναδρομικά Νευρωνικά Δίκτυα (RNNs)** (26; 27): Σχεδιασμένα για σειριακά δεδομένα (π.χ. γλώσσα, ήχος), διαθέτουν μηχανισμό επανάληψης (recurrent connections) που επιτρέπει τη διατήρηση μνήμης (context vector) για μελλοντικές εισόδους. Αυτό το πλαίσιο επιτρέπει στο δίκτυο να επεξεργάζεται ακολουθίες αυθαίρετου μήκους. Η ιδέα είναι ότι σε ακολουθιακά δεδομένα υπάρχει έντονη χρονική εξάρτηση, οπότε η εισαγωγή μνήμης επιτρέπει μελλοντικές εισόδους να αλληλεπιδρούν με προηγούμενες. Παρά τις επιτυχίες τους, οι βασικές εκδόσεις RNNs έχουν περιορισμούς στην αποθήκευση μακροχρόνιων εξαρτήσεων, οι οποίοι αντιμετωπίστηκαν εν μέρει από έξυπνες επεκτάσεις όπως τα *LSTM* (28).

- **Μετασχηματιστές (Transformers)** (29): Εισήγαγαν τον μηχανισμό *προσοχής (attention)* (30), ο οποίος επιτρέπει τη δυναμική στάθμιση όλων των εισόδων κατά την

επεξεργασία μιας ακολουθίας, χωρίς την ανάγκη επαναληπτικής κατάστασης. Η ιδέα είναι ότι κάθε είσοδος έχει στην διάθεση της όλες τις προηγούμενες για άντληση πληροφορίας, αντίθετα με τα RNNs όπου όλες οι αλληλεπιδράσεις γίνονται μέσω μιας σταθερής σε διάσταση μνήμης. Η αρχιτεκτονική αυτή έφερε επανάσταση στην μοντελοποίηση ακολουθιακών δεδομένων (ιδιαίτερα μεγάλου μήκους) και αποτελεί τη βάση των σύγχρονων *Μεγάλων Γλωσσικών Μοντέλων*.

0.2.2 Μάθηση Αναπαραστάσεων

Από τις προηγούμενες ενότητες, είναι φανερό ότι οι αλγόριθμοι μηχανικής μάθησης εξαρτώνται σε μεγάλο βαθμό από τον τρόπο αναπαράστασης των δεδομένων. Για παράδειγμα, το πρόβλημα XOR δείχνει ότι κατάλληλοι μη γραμμικοί μετασχηματισμοί μπορούν να καταστήσουν ένα πρόβλημα γραμμικά διαχωρίσιμο, υποδεικνύοντας ότι βασικά μια «καλή» αναπαράσταση είναι αυτή που καθιστά το πρόβλημα (εύκολα) επιλύσιμο. Η διαδικασία που ακολουθήθηκε στο XOR, ωστόσο, δεν γενικεύεται εύκολα, και χρειάζεται αρκετό χρόνο από μηχανικούς για την εύρεση τέτοιων αναπαραστάσεων και χαρακτηριστικών ιδίως για δεδομένα υψηλής διάστασης (feature engineering). Αυτή η διαδικασία απαιτεί εξειδικευμένη γνώση και δυσκολεύει τη γενίκευση σε διαφορετικούς τομείς.

Μάθηση Βαθιών Παραστάσεων Η εμφάνιση και επιτυχής εκπαίδευση των βαθιών νευρωνικών δικτύων μετέβαλε τη διαδικασία μάθησης αναπαραστάσεων. Συγκεκριμένα, τα δίκτυα αυτά μπορούν να μάθουν τις κατάλληλες αναπαραστάσεις απευθείας από τα δεδομένα, μειώνοντας σημαντικά τον χρόνο και την προσπάθεια μηχανικών να βρουν τα κατάλληλα χαρακτηριστικά. Έτσι, τα νευρωνικά δίκτυα επιτυγχάνουν τον μετασχηματισμό του σήματος εισόδου σε μια αναπαράσταση που διευκολύνει την επίλυση του εκάστοτε προβλήματος. Αυτή η ικανότητα καθιστά τις βαθιές αναπαραστάσεις ιδιαίτερα ισχυρές και αποτελεσματικές στην πράξη, ειδικά όταν υπάρχει η δυνατότητα προεκπαίδευσης με μεγάλη ποσότητα δεδομένων. Η βασική ιδέα είναι ότι, το μοντέλο μαθαίνει να οργανώνει και κατανοεί την εγγενή δομή των δεδομένων, μαθαίνοντας ουσιαστικά καλές αναπαραστάσεις αυτών και στην συνέχεια αυτές οι αναπαραστάσεις χρησιμοποιούνται με μόνο ένα μικρό αριθμό παραδειγμάτων με ετικέτες για την εκμάθηση κάποιου πιο ειδικού προβλήματος (31; 32; 21; 33).

Αυτο-επιβλεπόμενη Μάθηση Η αυτο-επιβλεπόμενη μάθηση έχει αναδειχθεί ως μια εξαιρετικά αποτελεσματική μορφή μη επιβλεπόμενης μάθησης, αντλώντας ουσιαστικά ιδέες και έμπνευση από το πεδίο της μάθησης αναπαραστάσεων. Συγκεκριμένα, τέτοιοι αλγόριθμοι εκμεταλλεύονται δεδομένα χωρίς ετικέτες δημιουργώντας ψευδο-στόχους, γεγονός που επιτρέπει την εκπαίδευση ισχυρών μοντέλων και την μάθηση γενικών αναπαραστάσεων χωρίς την ανάγκη ανθρώπινης επισημείωσης (επιβλεπόμενη μάθηση). Μοντέλα όπως το BERT (34) και η σειρά GPT (35) επέδειξαν την ικανότητα να μαθαίνουν γενικές γλωσσικές αναπαραστάσεις με χρήση τεράστιων συνόλων δεδομένων χωρίς ετικέτες. Αυτές οι αναπαραστάσεις μπορούν στην συνέχεια να προσαρμοστούν εύκολα (με λίγα ή και καθόλου δεδομένα) σε πιο στοχευμένα προβλήματα φυσικής γλώσσας, όπως για παράδειγμα ταξινόμηση συναισθήματος ή περίληψη. Τέλος, παρόμοιες τεχνικές έχουν εφαρμοστεί και σε άλλα πεδία όπως η

όραση (36), αποδεικνύοντας ότι η εκμάθηση αναπαραστάσεων από τεράστια δεδομένα χωρίς ετικέτες μπορεί να προσφέρει ισχυρή βάση για ένα ευρύ φάσμα εφαρμογών και προβλημάτων διαφορετικής φύσεως (π.χ. φυσική γλώσσα, όραση).

0.2.3 Αρχιτεκτονικές Νευρωνικών Δικτύων με πολλαπλά μονοπάτια

Η ανάπτυξη διαφορετικών τρόπων συνδεσιμότητας στα νευρωνικά δίκτυα έχει παίξει σημαντικό ρόλο στην πρόοδο και αναγνωρισιμότητα του πεδίου της Βαθιάς Μάθησης και αποτελεί άλλη μια σημαντική παράμετρο που καθορίζει την μάθηση αναπαραστάσεων. Συγκεκριμένα, νωρίς η ερευνητική κοινότητα ανακάλυψε ένα πρόβλημα των Εμπρόσθιων Νευρωνικών Δικτύων (ΕΣΔ), όπου τα gradients (κλίσεις) του δικτύου εξαφανίζονται ή μεγαλώνουν σημαντικά (εκρήγνυνται) καθώς περνούν μέσα από το δίκτυο όταν αυξάνεται το βάθος, εξαιτίας των διαδοχικών πολλαπλασιασμών και μη-γραμμικοτήτων (37; 38). Έτσι άρχισε να δίνεται έμφαση σε δίκτυα με πολλαπλά μονοπάτια και συνδέσεις συντομεύσεων, με την ιδέα ότι μέσω αυτών των συνδέσεων η πληροφορία μπορεί να ρέει καλύτερα στο δίκτυο και οι παράγωγοι να μην χάνονται καθώς αυξάνεται το βάθος, δίνοντας την δυνατότητα διάδοσης τους ανεμπόδιστα μέσω παρακάμψεων.

Συνδέσεις Συντομεύσεων (παρακάμψεων) Τα δίκτυα Υπερ-Διαδρομών (39) εισήγαγαν πύλες παράκαμψης που επιτρέπουν την ανεμπόδιστη ροή πληροφορίας σε πολύ βαθιά δίκτυα, αντιμετωπίζοντας τα προβλήματα με τις παραγώγους που αναφέραμε και εκπαιδεύοντας αποτελεσματικά δίκτυα με εκατοντάδες επίπεδα. Πειραματικά, αυτό οδήγησε σε καλύτερη γενίκευση και πιο εύρωστη εκπαίδευση καθώς αυξάνεται το βάθος. Η σχέση εισόδου-εξόδου για ένα στρώμα σε αυτά τα δίκτυα, από $y = F(x)$ που είναι σε ένα ΕΝΔ, μετατρέπεται σε:

$$y = T(x) \cdot F(x) + (1 - T(x)) \cdot x.$$

Στην συνέχεια, παρουσιάζονται τα Υπολλειματικά Νευρωνικά Δίκτυα (ΥΝΔ) (40) που αποτελούν μια απλοποιημένη περίπτωση των δικτύων Υπερ-Διαδρομών, αφαιρώντας τις πύλες και υιοθετώντας απευθείας ταυτοτικές συνδέσεις παράκαμψης, καταλήγοντας στον απλούστερο τύπο:

$$y = x + F(x).$$

Πολλές μελέτες έχουν αναδείξει τα πλεονεκτήματα των ΥΝΔ έναντι των ΕΝΔ, κυρίως ως προς τη δυναμική της εκπαίδευσης και τη διάδοση σήματος. Συγκεκριμένα, έχει δειχθεί ότι οι πολυδιάστατες μερικές παράγωγοι συμπεριφέρονται καλύτερα εξαιτίας αυτών των συνδέσεων συντομεύσεων, για παράδειγμα εξομαλύνοντας την επιφάνεια βελτιστοποίησης (41) (ουσιαστικά κάνοντας το πρόβλημα βελτιστοποίησης πιο εύκολο στο να βρει ένα καλό ελάχιστο). Ακόμη, οι παράγωγοι στα ΕΝΔ καθώς το βάθος αυξάνεται γίνονται πολύ θορυβώδεις (42) δυσκολεύοντας την εκπαίδευση, κάτι που μετριάζεται μέσω των παρακάμψεων των ΥΝΔ. Τέλος, έχει θεωρητικά αποδειχτεί (43) ότι αυτές οι συνδέσεις προκαλούν διατήρηση του μέτρου των σημάτων που διαδίδονται εμπρόσθια ή κατά την οπισθοδιάδοση στο δίκτυο, ουσιαστικά σταθεροποιώντας την εκπαίδευση.

Μια ανάλυση που εξηγεί τον όρο "αρχιτεκτονικές με πολλαπλά μονοπάτια" γίνεται στο

(3). Συγκεκριμένα, τα ΥΝΔ παρουσιάζονται ως σύνολα εξαρτώμενων επιμέρους δικτύων (ensemble). Έτσι, ένα ΥΝΔ n στρωμάτων περιέχει 2^n μονοπάτια διαφορετικού μήκους, καθώς το σήμα, καθώς περνάει το δίκτυο, μπορεί σε κάθε στρώμα να περάσει από μέσα ή από πάνω (μέσω της συντόμευσης).

Παραλλαγές αρχιτεκτονικών με πολλαπλά μονοπάτια Μετά την επιτυχία των Υπολειμματικών Νευρωνικών Δικτύων (ΥΝΔ), προτάθηκαν διάφορες παραλλαγές που τροποποιούν τη διασύνδεση των στρωμάτων πέρα από την απλή προσθετική σύνδεση "+1", με στόχο την αύξηση της εκφραστικότητας και της αποδοτικότητας. Ενδεικτικά, δοκιμάστηκαν ιδέες για αντικατάσταση της πρόσθεσης με concatenation ή αναδρομικές ιδέες (44; 45) και επέκταση των ΥΝΔ με εκπαιδεύσιμο γραμμικό συνδυασμό των προηγούμενων στρωμάτων (46), πιο πυκνή (σε βάθος και πλάτος) συνδεσιμότητα (47) αλλά και χρήση μηχανισμών προσοχής μεταξύ στρωμάτων (48). Συμπληρωματικά, υπήρξαν και διάφορες προσεγγίσεις που εξέταζαν παραλλαγές των ΥΝΔ κυρίως με τεχνικές όπως gating (δηλαδή $y = x + \mathbf{a} \cdot F(x)$) με σκοπό βελτιώσεις στη σταθερότητα και ταχύτητα εκπαίδευσης και στην ικανότητα γενίκευσης (49; 50; 15; 51).

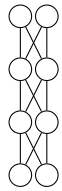
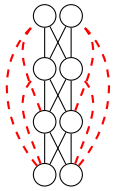
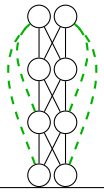
Πιθανός πλεονασμός παραμέτρων στα ΥΝΔ Παρά τα πλεονεκτήματα των ΥΝΔ, μελέτες δείχνουν ότι η εισαγωγή υπολλειματικών συνδέσεων μπορεί να οδηγεί σε υποεκπαίδευση στρωμάτων (52) καθώς και πλεονασμό στις παραμέτρους, ιδιαίτερα στα ανώτερα στρώματα (13). Αυτοί οι παράγοντες ενδέχεται να περιορίζουν τη γενίκευση (14) ή την ποιότητα των αναπαραστάσεων (15). Στο επόμενο κεφάλαιο, παρουσιάζουμε μια εναλλακτική αρχιτεκτονική που διατηρεί τα βασικά οφέλη των ΥΝΔ, διαθέτει δηλαδή πολλαπλά μονοπάτια ροής πληροφορίας, αλλά μετριάζει αυτές τις αδυναμίες, επιτυγχάνοντας εύρωστη μάθηση χωρίς πλεονασμό και αποδοτικότητα.

0.3 *Αυτο-Συμπιεζόμενα Δίκτυα*

0.3.1 Εισαγωγή

Παρά τις ποικίλες προσπάθειες της ερευνητικής κοινότητας σχετικά με την εισαγωγή διαφορετικών αρχιτεκτονικών νευρωνικών δικτύων με πολλαπλά μονοπάτια, όπως συζητήθηκε και στην προηγούμενη ενότητα, καμία από αυτές δεν έχει τύχει ευρείας αναγνώρισης. Ακόμη, καμία από αυτή δεν έχει ερευνήσει το πρόβλημα του πιθανού πλεονασμού στις παραμέτρους σε υπολλειματικές αρχιτεκτονικές. Στην παρούσα εργασία, προτείνουμε και μελετάμε μια αρχιτεκτονική διαφοροποίηση των αρχιτεκτονικών με πολλαπλά μονοπάτια, την οποία ονομάζουμε *Αυτο-Συμπιεζόμενα Δίκτυα* (ΑΣΔ). Συγκεκριμένα, όπως φαίνεται στον Πίνακα 1, τα ΑΣΔ σχηματίζονται ξεκινώντας από ένα απλό νευρωνικό δίκτυο με εμπρόσθιες συνδέσεις (ΕΝΔ) και προσθέτοντας επιπλέον μακρινές εμπρόσθιες συνδέσεις από κάθε στρώμα στο τελευταίο. Εξαιτίας αυτής της τροποποίησης, το δίκτυο έχει διαφορετική δομή σε σχέση με τα άπλα ΕΝΔ ή τα υπολλειματικά νευρωνικά δίκτυα (ΥΝΔ), και έτσι παρουσιάζει διαφορετική συμπεριφορά κατά την εκπαίδευση. Συγκεκριμένα, τα ΑΣΔ διαθέτουν μια ιδιότητα, την οποία ονομάζουμε *Αυτο-Συμπίεση*, όπου το δίκτυο κατά την διάρκεια

της εκπαίδευσης συμπιέζει την πληροφορία σε ένα υποσύνολο των πρώτων στρωμάτων του, τα οποία μπορούν να λύσουν αποτελεσματικά το παρόν πρόβλημα, αφήνοντας έτσι τα τελευταία του στρώματα χωρίς εκπαίδευση και αποκαλύπτοντας πιθανό πλεονασμό στις παραμέτρους του δικτύου. Στις επόμενες ενότητες παρουσιάζουμε αναλυτικά την προτεινόμενη αρχιτεκτονική, αναλύουμε θεωρητικά και πειραματικά την *Αυτο-Συμπίεση*, καθώς και τα αποτελέσματα που επιφέρει αυτή σε πρακτικό επίπεδο, συγκρίνοντας τα ΑΣΔ με τις προαναφερθείσες αρχιτεκτονικές.

Arch	Connectivity	Forward Propagation	Backward (Gradient) Propagation
FFN		$y_F = \prod_{i=1}^{L} w_i x_0$	$\frac{\partial y_F}{\partial w_i} = \underbrace{\left(\prod_{k=i+1}^{L} w_k\right)}_{\textit{backward term}} \underbrace{\left(\prod_{m=1}^{i-1} w_m\right)}_{\textit{forward term}} x_0$
ResNet		$y_R = \prod_{i=1}^{L}(1 + w_i)x_0$	$\frac{\partial y_R}{\partial w_i} = \underbrace{\left(\prod_{k=i+1}^{L}(1 + w_k)\right)}_{\textit{backward term}} \underbrace{\left(\prod_{m=1}^{i-1}(1 + w_m)\right)}_{\textit{forward term}} x_0$
ACN		$y_A = \left(1 + \sum_{i=1}^{L}\prod_{j=1}^{i} w_j\right) x_0$	$\frac{\partial y_A}{\partial w_i} = \underbrace{\left(1 + \sum_{j=i+1}^{L}\prod_{k=i+1}^{j} w_k\right)}_{\textit{backward term}} \underbrace{\left(\prod_{m=1}^{i-1} w_m\right)}_{\textit{forward term}} x_0$

Πίνακας 1. *Συνδεσιμότητα (2Δ περίπτωση) και διάδοση σήματος forward/backward (1Δ γραμμική περίπτωση) για ΕΝΔ, ΥΝΔ και ΑΣΔ.*

0.3.2 Η προτεινόμενη αρχιτεκτονική

Όπως αναφέρθηκε, ένα ΑΣΔ με L στρώματα λαμβάνει την παρακάτω μορφή:

$$x_i = f_i(x_{i-1}), \qquad y = \sum_{i=0}^{L} x_i \tag{1}$$

Με βάση την παραπάνω εξίσωση, παρατηρούμε ότι ένα ΑΣΔ είναι δίκτυο με πολλαπλά μονοπάτια λόγω των μακρινών εμπρόσθιων συνδέσεων που προστίθενται. Συγκεκριμένα ο αριθμός των μονοπατιών είναι ίσος με τον αριθμό των στρωμάτων L, τοποθετώντας τα ΑΣΔ μεταξύ των απλών ΕΝΔ που δεν διαθέτουν παρακάμψεις μεταξύ στρωμάτων και τα ΥΝΔ τα οποία διαθέτουν 2^L, για L στρώματα.

Εξισώσεις Παραγώγων και Ροή Πληροφορίας στις διάφορες αρχιτεκτονικές Συγκρίνουμε τις εξισώσεις παραγώγων και την ροή πληροφορίας στα ΑΣΔ συγκριτικά με τα ΕΝΔ και τα ΥΣΔ, στην περίπτωση ενός γραμμικού δικτύου L στρωμάτων και με σήμα εισόδου μίας διάστασης. Συμβολίζουμε με x_i την έξοδο του στρώματος i, με w_i το βάρος της στρώματος i,

με x_0 το σήμα εισόδου, και y_F, y_R, y_A είναι τα σήματα εξόδου για κάθε αρχιτεκτονική.

Παρατηρούμε στον Πίνακα 1 ότι σε κάθε περίπτωση η τελική εξίσωση παραγώγων για κάθε w_i μπορεί να αποσυντεθεί σε δύο όρους:

- Ο *εμπρόσθιος όρος* περιέχει το εμπρόσθιο σήμα από την είσοδο ως το στρώμα i και καθορίζει τη σταθερότητα του μέτρου της παραγώγου (δηλαδή αν το σήμα εξασθενεί ή αυξάνεται απότομα). Στα ΑΣΔ και ΕΝΔ αυτός ο όρος είναι ίδιος και δεν περιέχει παρακάμψεις για τα ενδιάμεσα στρώματα[1] και το μέτρο του τυπικά μειώνεται καθώς αυξάνεται το βάθος του δικτύου, λόγω πολλαπλασιασμών και μη-γραμμικοτήτων (υπό την υπόθεση αρχικοποίησης παραμέτρων κοντά στο μηδέν). Αντίθετα στα ΥΝΔ, ο αριθμός των μονοπατιών που περιλαμβάνονται σε αυτή την συνιστώσα αυξάνεται καθώς διασχίζονται περισσότερα στρώματα.

- Ο *οπίσθιος* όρος μεταφέρει την πληροφορία που σχετίζεται με την μάθηση. Στα μονοδιάστατα (1Δ) ΕΝΔ, αυτός ο όρος περιλαμβάνει ένα μονοπάτι. Ωστόσο, για τα 1Δ ΑΣΔ και 1Δ ΥΝΔ, έκαστες αρχιτεκτονικές πολλαπλών μονοπατιών, ο όρος αποτελείται από $L - i + 1$ και 2^{L-i} μονοπάτια, αντίστοιχα.[2] Συμβολίζουμε με **FG** τον *οπίσθιο* όρο.

Ο *οπίσθιος* όρος μπορεί να διαχωριστεί περαιτέρω σε δύο όρους:

- **Μονοπάτια που διασχίζουν το δίκτυο:** Σε όλες τις αρχιτεκτονικές, υπάρχει μια κυρίαρχη συνιστώσα στις εξισώσεις όπου η πληροφορία φτάνει σε ένα στρώμα μέσα από τα επόμενα από αυτό στρώματα, διασχίζοντας τα και ουσιαστικά καθορίζοντας πώς πρέπει να προσαρμοστούν τα βάρη αυτού του στρώματος με βάση τα επόμενα. Στις εξισώσεις αντιστοιχεί στον συνολικό *οπίσθιο* όρο των ΕΝΔ, ενώ για τα ΑΣΔ και ΥΝΔ περιλαμβάνει όλα τα μονοπάτια εκτός του μονοπατιού που φτάνει απευθείας από την έξοδο (ο όρος "1+" στις αντίστοιχες εξισώσεις). Σημειώνουμε ότι το μέτρο αυτής της συνιστώσας μειώνεται καθώς αυξάνεται το βάθος λόγω της μείωσης των μονοπατιών και την συμβολίζουμε ως **NG**.

- **Μονοπάτι απευθείας από την έξοδο του δικτύου:** Στα ΑΣΔ και ΥΝΔ υπάρχει επίσης μια διαδρομή που προέρχεται **απευθείας από την έξοδο**, λόγω των παρακάμψεων που δημιουργούν οι υπολλειματικες συνδέσεις και αντιστοιχεί στο σκέλος "1+" των παραπάνω εξισώσεων[3]. Συμβολίζουμε τον όρο ως **DG**. Ο όρος αυτός μεταφέρει την πληροφορία του πως πρέπει να μεταβληθεί το βάρος ενός στρώματος ώστε το συνολικό δίκτυο να βελτιωθεί στο πρόβλημα που εκπαιδεύεται. Η συμβολή του **DG** είναι πιο σημαντική στα ΑΣΔ, αφού ο αριθμός των μονοπατιών είναι γραμμικός ως προς το βάθος, σε αντίθεση με τα ΥΝΔ όπου είναι εκθετικός. Για παράδειγμα, στο στρώμα 2 ενός δικτύου $L = 12$, η **DG** αποτελεί 1 από τα 11 μονοπάτια στα ΑΣΔ, ενώ στα ΥΝΔ ανταγωνίζεται 127 άλλα μονοπάτια.

[1] Υπό την έννοια ότι το στρώμα a για να φτάσει στο στρώμα b πρέπει να διασχίσει όλα τα ενδιάμεσα.

[2] Αξίζει να αναφερθεί ότι το μονοπάτι του ΕΝΔ είναι υποσύνολο των μονοπατιών του ΑΣΔ, τα οποία με τη σειρά τους περιλαμβάνονται στα μονοπάτια του ΥΝΔ.

[3] Στη βιβλιογραφία ο όρος αυτός είναι γνωστός ως Direct Feedback Alignment (DFA) (53; 54)

Συνολικά, μπορούμε να γράψουμε:

$$\mathbf{FG} = \mathbf{DG} + \mathbf{NG}.$$

Ένμεση εκπαίδευση στρώμα με στρώμα στα ΑΣΔ Το γεγονός ότι τα ΑΣΔ διαθέτουν *εμπρόσθιο* όρο όμοιο με τα ΕΝΔ και *οπίσθιο* όρο με πολλαπλά μονοπάτια οδηγεί έμμεσα σε εκπαίδευση στρώμα με στρώμα (layer-wise), όπου τα πρώτα στρώματα εκπαιδεύονται πιο γρήγορα από τα πιο βαθειά αφού έχουν μεγαλύτερο σε μέτρο *εμπρόσθιο* όρο καθώς και μεγαλύτερο σε μέτρο *οπίσθιο* όρο λόγω περισσότερων μονοπατιών. Ακόμη διαθέτουν ένα ισχυρό (σε σχέση με τα ΥΝΔ) **DG** όρο που ωθεί τα στρώματα να μεταβάλλουν τα βάρη τους έτσι ώστε να μειώνεται άμεσα (λόγω της συνεισφοράς τους στο τελικό άθροισμα) το λάθος κατά την εκπαίδευση. Έτσι, η ιδέα είναι ότι αν τα πρώτα στρώματα, που εκπαιδεύονται ταχύτερα, μπορούν επιτυχώς να λύσουν το πρόβλημα, τα τελευταία στρώματα δεν χρησιμοποιούνται από το δίκτυο και καταλήγουν αυτόματα να είναι πλεονάζοντα. Αυτή ουσιαστικά η συμπεριφορά είναι η *Αυτο-Συμπίεση*, όπου το δίκτυο αυτόματα κατά την εκπαίδευση χρησιμοποιεί μόνο όσα στρώματα πραγματικά χρειάζεται για την επίλυση του προβλήματος (ελαχιστοποίηση του λάθους).

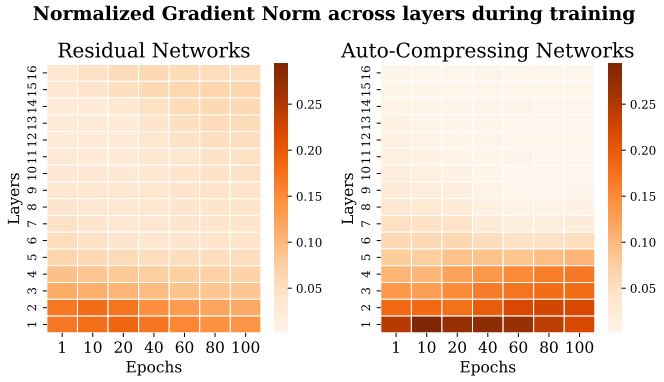
Σημειώνουμε πως αυτή η συμπεριφορά κατά την εκπαίδευση είναι διαφορετική από τα ΥΝΔ (όπως θα δείχνει και εμπειρικά στην επόμενη ενότητα) και αντιμετωπίζει ευθέως τον πιθανό πλεονασμό παραμέτρων στα ΥΝΔ, που συζητήθηκε προηγουμένως, μέσω της ιδιότητας της *Αυτο-Συμπίεσης*.

0.3.3 Πειράματα

Στις επόμενες ενότητες παρουσιάζουμε πειραματικά αποτελέσματα της προτεινόμενης αρχιτεκτονικής.

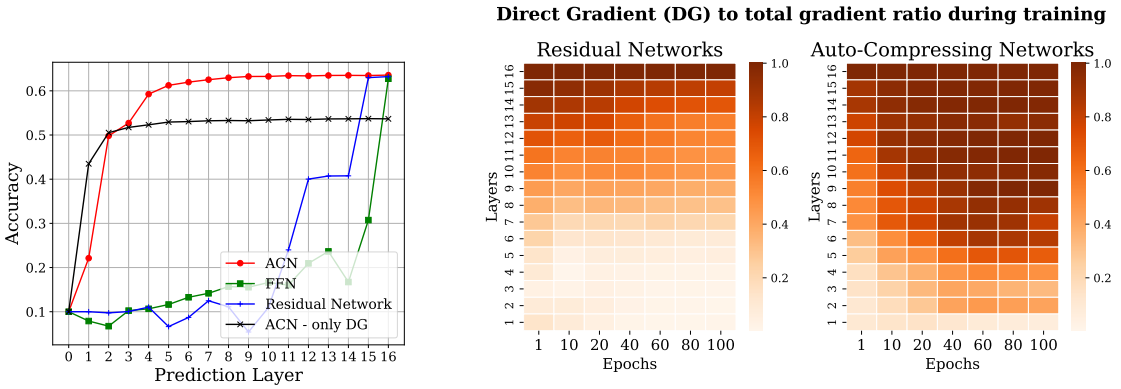
Μοντέλα: Υλοποιούμε ΑΣΔ σε αρχιτεκτονικές Transformer (29) και MLP-Mixer (4).

Σύνολα Δεδομένων: Χρησιμοποιούμε για ταξινόμηση εικόνων τα CIFAR-10/100 (5) και ImageNet-1K και για κατανόηση φυσικής γλώσσας τα BooksCorpus (55), English Wikipedia, SST-2 (56), QQP (57) και QNLI (58).



Σχήμα 2. *Τα μέτρα των gradients στα ΥΝΔ και ΑΣΔ κατά την διάρκεια της εκπαίδευσης.*

Αυτο-Συμπίεση στην πράξη Εκπαιδεύουμε ΑΣΔ, ΕΝΔ και ΥΝΔ ενσωματωμένα στην αρχιτεκτονική MLP-Mixer στο σύνολο δεδομένων CIFAR-10 για 100 εποχές. Επίσης εκπαιδεύουμε μία παραλλαγή ΑΣΔ που λαμβάνει σήμα κατά το backpropagation μόνο από τις μακρινές συνδέσεις (δηλαδή μόνο από το **DG** μονοπάτι). Τα αποτελέσματα φαίνονται στο Σχήμα 3 (αριστερά). Παρατηρούμε ότι μόνο τα ΑΣΔ παρουσιάζουν Αυτο-Συμπίεση. Ακόμη στο Σχήμα 3 (δεξιά) δείχνουμε ότι το **DG** είναι πιο ισχυρό στα ΑΣΔ από τα ΥΝΔ και τέλος στο Σχήμα 2 παρατηρούμε ότι τα μέτρα των παράγωγων των ΑΣΔ μικραίνουν με το βάθος ενώ στα ΥΝΔ είναι πιο ομοιόμορφα, επιβεβαιώνοντας την στρώμα με στρώμα εκπαίδευση που συζητήσαμε.

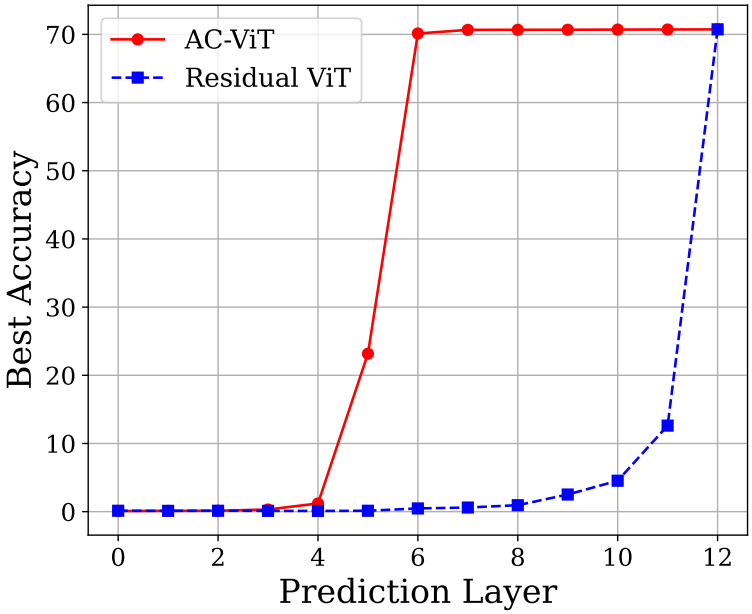


Σχήμα 3. ***(αριστερά)*** *Τα ΑΣΔ είναι η μόνη αρχιτεκτονική που πετυχαίνει Αυτο-Συμπίεση.* ***(δεξιά)*** *Ο λόγος του* ***DG*** *προς το* ***FG*** *στα ΥΝΔ (Residual Networks) και ΑΣΔ (Auto-Compressing Networks). Όπως φαίνεται, ο γραμμικός (έναντι εκθετικού) αριθμού μονοπατιών στα ΑΣΔ προκαλεί σημαντική συνεισφορά του* ***DG*** *στα πρώτα στρώματα του ΑΣΔ.*

Αυτο-Συμπιεζόμενοι Vision Transformers Στη συνέχεια, αξιολογούμε τα ΑΣΔ στο πλαίσιο των αρχιτεκτονικών τύπου Transformer, υλοποιώντας μία αυτο-συμπιεζόμενη εκδοχή του Vision Transformer (ViT) (6). Εκπαιδεύουμε έναν Vision Transformer (ViT) με μακρινές συνδέσεις (AC-ViT) και ένα με υπολλειματικές κοντινές συνδέσεις (Residual ViT) στο σύνολο δεδομένων ILSVRC-2012 ImageNet-1K[4]. Όπως παρατηρούμε στο Σχήμα 4 το ΑΣΔ πετυχαίνει σημαντική αυτο-συμπίεση χρησιμοποιώντας το μισό βάθος για να πετύχει την ίδια επίδοση με το ΥΝΔ που χρησιμοποιεί όλο το δίκτυο.

Αυτο-Συμπίεση και Δυσκολία του προβλήματος Στη συνέχεια εξετάζουμε πως επηρεάζεται η ικανότητα αυτο-συμπίεσης ως συνάρτηση της δυσκολίας του προβλήματος που καλείται να λύσει το δίκτυο. Χρησιμοποιούμε τον αριθμό των κλάσεων ως έναν δείκτη για τη δυσκολία του προβλήματος ταξινόμησης εικόνων στο σύνολο δεδομένων CIFAR-10, δημιουργώντας υποσύνολα με 2, 5 και 10 κλάσεις. Για το πείραμα αυτό χρησιμοποιούμε την αρχιτεκτονική MLP-Mixer και εκπαιδεύουμε δύο παραλλαγές: την αρχική MLP-Mixer με υπολειμματικές συνδέσεις (Residual Mixer - ΥΝΔ) και την τροποποιημένη έκδοση (ΑΣΔ) με μακρινές συνδέσεις (AC-Mixer). Όπως παρατηρούμε στο Σχήμα 5, τα ΑΣΔ φαίνονται να προσαρμόζονται στην δυσκολία του προβλήματος, χρησιμοποιώντας διαδοχικά περισσότερα στρώματα

[4]το ΥΝΔ συγκλίνει σε 300 εποχές ενώ το ΑΣΔ χρειάζεται 700 εποχές.



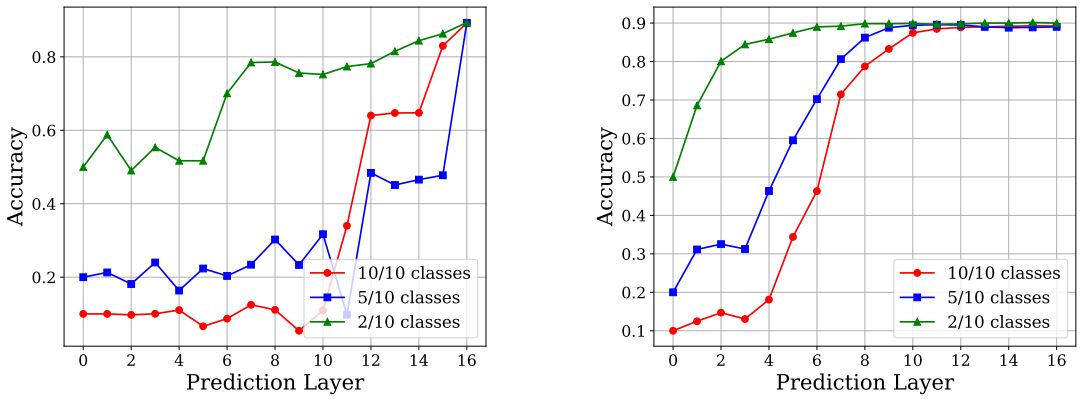
Σχήμα 4. *Επίδοση των ενδιάμεσων στρωμάτων των ΑΣΔ (AC-ViT) και ΥΝΔ (Residual ViT) στο ImageNet-1k, ενσωματωμένα σε αρχιτεκτονική Vision Transformer (ViT).*

καθώς κινούμαστε από 2- σε 10-κλάσεων πρόβλημα ταξινόμησης. Η συμπεριφορά αυτή δεν παρατηρείται στα ΥΝΔ, τα οποία χρησιμοποιούν όλα τα στρώματα ανεξάρτητα από την δυσκολία του προβλήματος. Συνολικά, φαίνεται ότι τα ΑΣΔ κάνουν δυναμική Αυτο-Συμπίεση με βάση το πρόβλημα χωρίς να χάνουν σε επίδοση.

Ικανότητες Γενίκευσης των ΑΣΔ Σε αυτήν την ενότητα, δείχνουμε πως οι αναπαραστάσεις που μαθαίνονται από το δίκτυο μέσω της αυτο-συμπίεσης γενικεύουν καλύτερα απέναντι σε θόρυβο στην είσοδο και σε περιπτώσεις λιγοστών δεδομένων εκπαίδευσης.

- **ΑΣΔ ενάντια σε θόρυβο εισόδου**: Χρησιμοποιούμε τα εκπαιδευμένα μοντέλα ViT προηγούμενης ενότητας και προσθέτουμε στην είσοδο αυξανόμενα επίπεδα προσθετικού θορύβου Gauss με τυπική απόκλιση $\sigma = 0.1, 0.2, 0.4$, καθώς και θόρυβο τύπου "salt-and-pepper" με ποσοστό αλλοιωμένων pixels $p = 1\%, 2\%, 10\%$. Τα αποτελέσματα (μέση ακρίβεια) που παρουσιάζονται στον Πίνακα 2, δείχνουν ότι τα ΑΣΔ είναι σημαντικά πιο εύρωστα απέναντι στο θόρυβο σε σχέση με τα ΥΝΔ, στα οποία ο θόρυβος μεταφέρεται σε όλο το δίκτυο εξαιτίας των υπολλειματικών συνδέσεων.

- **ΑΣΔ με λιγοστά δεδομένα εκπαίδευσης**: Επαναλαμβάνουμε το πείραμα του MLP-Mixer σε CIFAR-10 χρησιμοποιώντας το 1/10 των δεδομένων εκπαίδευσης για κάθε κλάση. Εκπαιδεύουμε για 150 εποχές και δείχνουμε στο Σχήμα 6 το σφάλμα εκπαίδευσης και test για τις δύο αρχιτεκτονικές. Όπως φαίνεται, τα ΑΣΔ πετυχαίνουν γρήγορα μικρό σφάλμα εκπαίδευσης και σημαντικά μικρότερο σφάλμα γενίκευσης (test) σε σχέση με τα ΥΝΔ, υποδεικνύοντας πως η προτεινόμενη αρχιτεκτονική είναι καταλληλότερη σε περιπτώσεις με λιγοστά δεδομένα.



(α΄) *Residual Mixer* **(β΄)** *AC-Mixer*

Σχήμα 5. *Συμπεριφορά (επίδοση) των ενδιάμεσων στρωμάτων σε ΑΣΔ και ΥΝΔ όταν μεταβάλλουμε τον αριθμό κλάσεων στο πρόβλημα ταξινόμησης εικόνων CIFAR-10 (μοντέλο MLP-Mixer).*

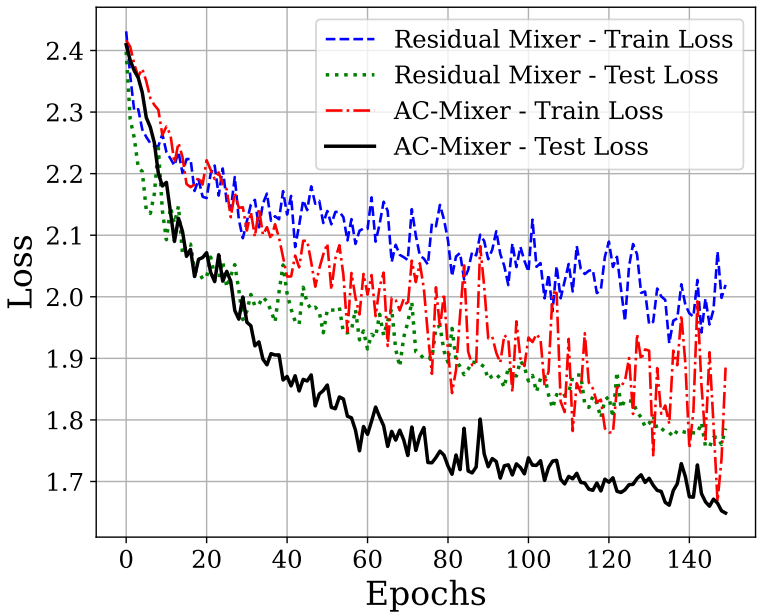
Model	Baseline	Gaussian Noise			Salt and Pepper Noise		
	w/o noise	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.4$	$p = 0.01$	$p = 0.05$	$p = 0.1$
Residual ViT	70.74	67.68	62.80	45.46	56.80	27.48	10.34
AC-ViT	70.76	69.50	64.54	51.89	59.80	36.35	19.98

Πίνακας 2. *Συμπεριφορά των ΑΣΔ και ΥΝΔ υπό την παρουσία θορύβου στην είσοδο (πείραμα ViT -ImageNet). Παρουσιάζεται η μέση ακρίβεια (%) απέναντι σε προσθετικό θόρυβο Gauss με και θόρυβο τύπου "salt-and-pepper".*

Αυτο-Συμπιεζόμενο BERT Στην συνέχεια, δοκιμάζουμε τα ΑΣΔ σε σύγχρονα μοντέλα κατανόησης φυσικής γλώσσας και συγκεκριμένα στο BERT. Η ιδέα είναι ότι κατά την διάρκεια της προ-εκπαίδευσης το δίκτυο μπορεί να χρησιμοποιεί όλα τα στρώματα για την κατανόηση της φυσικής γλώσσας που είναι ένα γενικό πρόβλημα και στην συνέχεια να χρησιμοποιεί μόνο ένα υποσύνολο αυτών όταν το προσαρμόζουμε σε πιο ειδικά προβλήματα (π.χ. ανάλυση συναισθήματος). Προ-Εκπαιδεύουμε ένα υπολλειματικό BERT (Residual BERT - ΥΝΔ) για μία εποχή και ένα ΑΣΔ μοντέλο BERT (AC-BERT) για δύο εποχές, στα BookCorpus και English Wikipedia και στην συνέχεια εκπαιδεύουμε σε πιο ειδικά και μικρά σύνολα δεδομένων γα να αξιολογήσουμε τα δυο μοντέλα (ανάλυση συναισθήματος: SST-2, παράφραση: QQP, ερωτο-απαντήσεις: QNLI). Τα αποτελέσματα του Σχήματος 7 (αριστερά) δείχνουν ότι το ΑΣΔ πετυχαίνει παρόμοια επίδοση με το ΥΝΔ και στα 3 σύνολα δεδομένων επιδεικνύοντας όμως ισχυρή Αυτο-Συμπίεση, αφού χρησιμοποιεί περίπου το 1/4 των στρωμάτων του BERT.

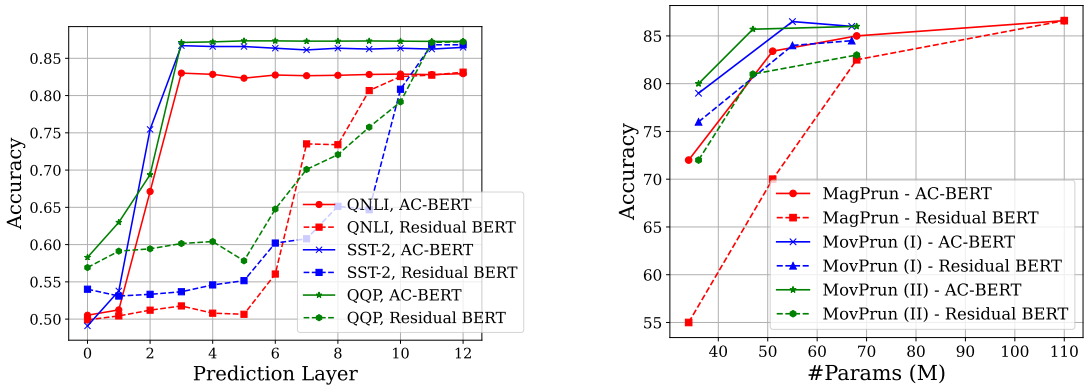
Ακόμη εξετάζουμε αν τα ΑΣΔ μπορούν να χρησιμοποιηθούν συμπληρωματικά με άλλες τεχνικές αφαίρεσης παραμέτρων, συγκεκριμένα τεχνικές pruning. Χρησιμοποιούμε δύο τέτοιες τεχνικές στο SST-2: (1) αφαίρεση παραμέτρων με βάση το μέτρο τους μετά το τέλος της εκπαίδευσης στο τελικό πρόβλημα και (2) σταδιακή αφαίρεση παραμέτρων [5] κατά την εκπαίδευση με βάση το μέτρο τους και τις παραγώγους τους ως προς το σφάλμα (Movement

[5]Στην περίπτωση (I) αφαιρούμε 20% των παραμέτρων μετά από κάθε εποχή, ενώ στην (II), 40% κάθε εποχή



Σχήμα 6. *Σφάλμα εκπαίδευσης και γενίκευσης (τεστ) των ΑΣΔ και ΥΝΔ σε MLP-Mixer στο CIFAR-10, σε σενάριο λιγοστών δεδομένων (100 δεδομένα ανά κλάση).*

Pruning(59)). Όπως παρατηρούμε στο Σχήμα 7 (δεξιά), η Αυτο-Συμπίεση των ΑΣΔ δρα συμπληρωματικά με αυτές τις τεχνικές και έτσι τα ΑΣΔ πετυχαίνουν σημαντικά καλύτερη ισορροπία μεταξύ αριθμού παραμέτρων (sparsity level) και επίδοσης.



Σχήμα 7. ***(αριστερά)*** *Απόδοση του προ-εκπαιδευμένου AC-BERT (ΑΣΔ) έναντι του προ-εκπαιδευμένου υπολειμματικού BERT (ΥΝΔ) μετά από προσαρμογή (fine-tuning) σε σύνολα δεδομένων με: ανάλυση συναισθήματος (SST-2), παράφραση (QQP), και ερωτο-απαντήσεις (QNLI).* ***(δεξιά)*** *Ακρίβεια έναντι μεγέθους μοντέλου του των δύο μοντέλων στο SST-2 όταν εφαρμόζεται Magnitude και Movement Pruning.*

Διαρκής (συνεχής) μάθηση και ΑΣΔ Η διαρκής (ή συνεχής) μάθηση (60; 61; 62; 63; 64; 65; 66) είναι ένα υποσύνολο των ειδών μάθησης όπου διαφορετικά υπό-προβλήματα παρουσιάζονται ακολουθιακά, χωρίς επανάληψη, και ο στόχος είναι, στο τέλος της συνολικής

εκπαίδευσης, δηλαδή αφού το μοντέλο εκπαιδευτεί σταδιακά σε όλα τα υπό-προβλήματα(με την σειρά, ένα-ένα), να παρουσιάζει συνολικά καλή επίδοση. Η κύρια δυσκολία αυτής της κατηγορίας προβλημάτων είναι ότι τα νευρωνικά δίκτυα όταν εκπαιδεύονται σε ένα πρόβλημα Α και μετά σε ένα πρόβλημα Β, τείνουν να "ξεχνούν" το Α. Η Αυτο-Συμπίεση είναι μια ιδιότητα που φαίνεται χρήσιμη σε μια τέτοια περίπτωση αφού το δίκτυο μπορεί αυτόματα να ανακαλύπτει ποιες παραμέτρους χρειάζεται για ένα πρόβλημα και να αφήνει τις υπόλοιπες για τα επόμενα. Για να επαληθεύσουμε την διαίσθηση μας, αξιολογούμε ΑΣΔ και ΥΝΔ σε MLP-Mixer, στο σύνολο δεδομένων split CIFAR-100, το οποίο περιλαμβάνει 20 διαδοχικά και μη επικαλυπτόμενα προβλήματα ταξινόμησης με 5 κλάσεις το καθένα (task-incremental learning (67)). Εκπαιδεύουμε για 10 εποχές σε κάθε υπό-πρόβλημα. Δοκιμάζουμε δύο αλγορίθμους συνεχούς μάθησης: απλή εκπαίδευση (Naive FT) και αλγόριθμο Synaptic Intelligence (SI) (65)[6]. Στα πειράματα αναφέρουμε το Average Forgetting, ορισμένο ως ο μέσος όρος της διαφοράς μεταξύ της καλύτερης απόδοσης ενός υπό-προβλήματος (άμεσα μετά την εκμάθησή του) και της τελικής απόδοσής του μετά την εκμάθηση όλων, καθώς και τη Μέση Ακρίβεια (Average Accuracy), που ορίζεται ως η μέση ακρίβεια σε όλα τα προβλήματα στο τέλος της εκπαίδευσης. Τα αποτελέσματα του Πίνακα 3 επιβεβαιώνουν τη διαίσθησή μας: τα ΑΣΔ ξεχνούν σημαντικά λιγότερο (έως και 18% βελτίωση) σε σύγκριση με τα ΥΝΔ. Ιδιαίτερα, με τη χρήση του αλγορίθμου Synaptic Intelligence (SI), η αύξηση του βάθους των ACNs οδηγεί σε μείωση του forgetting — μια ιδανική συμπεριφορά για τέτοιου είδους συστήματα, μιας και τα περισσότερα στρώματα χρησιμοποιούνται για την εκμάθηση περισσότερων υπό-προβλημάτων. Αντίθετα, τα ΥΝΔ ξεχνούν περισσότερο όσο αυξάνεται το βάθος, υποδεικνύοντας πιθανή υπερπροσαρμογή (overfitting). Επιπλέον, τα ΑΣΔ επιτυγχάνουν υψηλότερη μέση ακρίβεια και φαίνονται έτσι να είναι καταλληλότερα σε περιπτώσεις συνεχούς μάθησης.

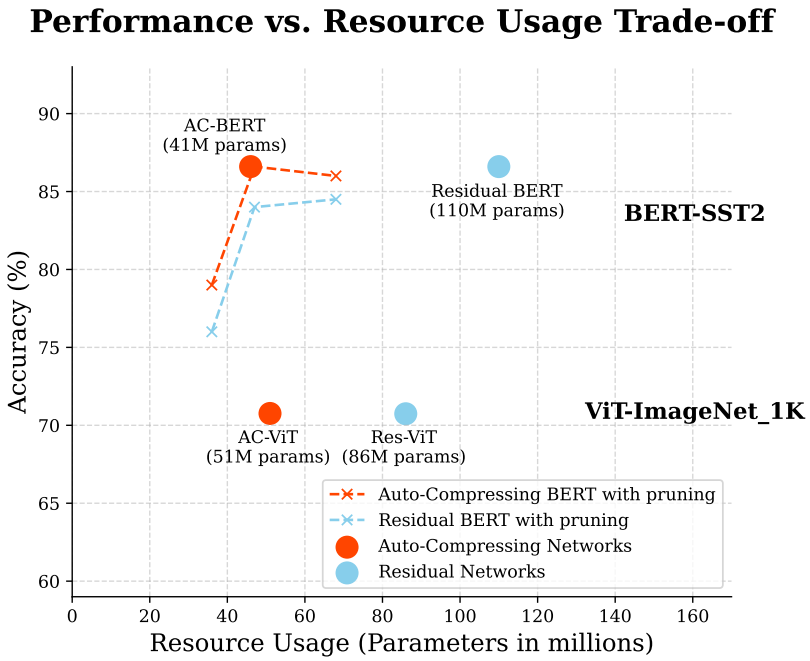
		Avg. Accuracy (%) ↑			**Avg. Forgetting (%) ↓**		
M.	**Arch**	$L = 5$	$L = 10$	$L = 15$	$L = 5$	$L = 10$	$L = 15$
*n*FT	ACN	**32.97** ± 2.4	32.94 ± 5.3	31.61 ± 2.2	46.55 ± 2.2	**45.46** ± 5.8	46.91 ± 2.4
	Res	31.77 ± 1.8	28.16 ± 1	26.14 ± 2.3	52.76 ± 2.3	54.89 ± 1.6	54.49 ± 2.2
SI	ACN	44.5 ± 2.2	46.1 ± 1.3	**46.2 ±** 0.8	35.7 ± 2.1	33.8 ± 0.4	32 **±** 1.8
	Res	43.47 ± 3.1	36.1 ± 5	**32.1** ± 0.8	42.4 ± 4.1	44.6 ± 3.7	50 ± 2.1

Πίνακας 3. *Μέση ακρίβεια και forgetting για τις διάφορες μεθόδους, δίκτυα και αρχιτεκτονικές στο Split CIFAR-100. Το M. σημαίνει Μέθοδος, το nFT συμβολίζει την απλή εκπαίδευση και το SI τον αλγόριθμο Synaptic Intelligence. Τα μοντέλα εκπαιδεύονται για 10 εποχές ανά πρόβλημα, όπου κάθε πρόβλημα αποτελείται από την ταξινόμηση 5 από τις 100 κλάσεις που παρουσιάζονται διαδοχικά. Το L δηλώνει τον αριθμό των στρωμάτων στην αρχιτεκτονική.*

[6]Ο αλγόριθμος αυτός προσθέτει έναν regularizer βασισμένο στις παραγώγους ως προς το σφάλμα για κάθε παράμετρο, συγκεκριμένα προκύπτει μια τιμή για κάθε παράμετρο ανάλογα με το πώς οι αλλαγές σε αυτήν επηρεάζουν το συνολικό σφάλμα στο τρέχον πρόβλημα κατά τη διάρκεια της εκπαίδευσης.

0.3.4 Σύνοψη των Αποτελεσμάτων

Στο Σχήμα 7 και στον Πίνακα 4 παρουσιάζονται συνοπτικά κάποια από τα αποτελέσματα των προηγούμενων πειραμάτων. Σημειώνουμε ότι σε όλα τα πειράματα παρατηρήσαμε ισχυρή Αυτο-Συμπίεση στα ΑΣΔ, η οποία μπορεί να επιταχύνει σημαντικά την χρήση του δικτύου μετά την εκπαίδευση αφού μπορούμε ανώδυνα να αφαιρέσουμε ένα μεγάλο ποσοστό των στρωμάτων του. Ακόμη και με την Αυτο-Συμπίεση, τα ΑΣΔ πετυχαίνουν επιδόσεις όμοιες με τα ΥΔΝ ενώ σε αρκετές περιπτώσεις γενικεύουν καλύτερα (θόρυβος στην είσοδο, λιγοστά δεδομένα εκπαίδευσης, διαρκής μάθηση).



Σχήμα 8. *Απόδοση ΑΣΔ έναντι ΥΝΔ συναρτήσει του αριθμού παραμέτρων, με και χωρίς pruning.*

Models	Accuracy ↑	#Params ↓	#Inference Layers ↓	Storage Size (MB) ↓
Res-Mixer on C10	90.12 ± 0.06	2.5M	16	17.6
AC-Mixer on C10	**90.24** ± 0.05	**1.8M**	**12**	**13.2**
Res-ViT on ImageNet	70.74 ± 0.09	86M	12	330
AC-ViT on ImageNet	**70.76** ± 0.12	**51M**	**7**	**195**
Res-BERT on SST-2	86.63 ± 0.09	110M	12	418
AC-BERT on SST-2	**86.68** ± 0.06	**46M**	**3**	**174**
Res-BERT on QNLI	**83.14** ± 0.07	110M	12	418
AC-BERT on QNLI	83.07 ± 0.1	**46M**	**3**	**174**
Res-BERT on QQP	87.2 ± 0.09	110M	12	418
AC-BERT on QQP	**87.3** ± 0.07	**46M**	**3**	**174**

Πίνακας 4. *Ανασκόπηση των πειραματικών αποτελεσμάτων, σε αρχιτεκτονικές ΑΣΔ και ΥΝΔ.*

0.3.5 Συμπεράσματα

Στην παρούσα εργασία, παρουσιάστηκε και μελετήθηκε μια νέα αρχιτεκτονική νευρωνικών δικτύων, τα Αυτο-Συμπιεζόμενα Δίκτυα (ΑΣΔ). Μέσω της διαφορετικής συνδεσιμότητας τους, σε σχέση με τα Εμπρόσθια και Υπολλειματικά Νευρωνικά Δίκτυα, τα ΑΣΔ παρουσιάζουν διαφορετικά χαρακτηριστικά κατά την εκπαίδευσή με αποτέλεσμα να καταλήγουν σε διαφορετικές αναπαραστάσεις σε σχέση με τα ΕΝΔ και ΥΝΔ. Συγκεκριμένα, τα δίκτυα αυτά εκπαιδεύονται έμμεσα στρώμα-με-στρώμα λόγω της αρχιτεκτονικής τους με αποτέλεσμα να παρουσιάζουν μια νέα ιδιότητα, την οποία ονομάσαμε *Αυτο-Συμπίεση*, όπου η πληροφορία κατά την εκπαίδευση συγκεντρώνεται σε ένα υποσύνολο των συνολικών στρωμάτων του δικτύου. Μέσω αυτής της ιδιότητας, τα ΑΣΔ εκπαιδεύουν εύρωστα τα πρώτα στρωματά τους και καταλήγουν σε ενδιαφέροντα πρακτικά πλεονεκτήματα, συγκριτικά με τα Υπολλειματικά Νευρωνικά Δίκτυα.

Σε όλα τα πειράματα, είδαμε ότι τα ΑΣΔ προσφέρουν συγκρίσιμη ή ανώτερη απόδοση από τα υπολλειματικά δίκτυα, με ταχύτερη εκτέλεση και μειωμένη χρήση μνήμης. Δείξαμε ακόμη ότι μεγάλο ποσοστό των άνω στρωμάτων καθίστανται πλεονάζοντα, συγκεντρώνοντας την πληροφορία στα κατώτερα επίπεδα σε διάφορα μοντέλα και σύνολα δεδομένων, όπως ταξινόμηση εικόνων με Transformer και MLP-Mixer ή κατανόηση γλώσσας με γλωσσικά μοντέλα τύπου BERT. Επιπλέον, τα ΑΣΔ παρουσιάστηκαν ανώτερα σε περιπτώσεις θορύβου στην είσοδο, σενάρια λιγοστών δεδομένων αλλά και περιπτώσεις διαρκούς μάθησης, αποκαλύπτοντας ότι έχουν δυνατότερες ικανότητες γενίκευσης σε τέτοια σενάρια, που μάλιστα είναι προβλήματα που συναντάμε πολύ συχνά στον πραγματικό κόσμο.

Κλείνοντας, μελλοντικές δουλείες θα μπορούσαν να επεκτείνουν τα ΑΣΔ σε μεγάλα γλωσσικά και πολυτροπικά μοντέλα και να μελετηθούν εκεί τα πλεονεκτήματα της αυτο-συμπίεσης. Ακόμη, ένα ενδιαφέρον πεδίο είναι η ανάπτυξη αλγορίθμων που προσαρμόζουν δυναμικά, κατά την εκτέλεση, τον αριθμό των στρωμάτων ανά δείγμα για βέλτιστη απόδοση και αποδοτικότητα, πάλι μέσω της αυτο-συμπίεσης. Σημαντική κρίνεται και η διερεύνηση του αυξημένου χρόνου εκπαίδευσης των ΑΣΔ σε σχέση με τα ΥΝΔ και η απάντηση στο ερώτημα αν αυτή η συμπεριφορά είναι αναγκαία για να υπάρξει Αυτο-Συμπίεση ή μπορεί να μετριαστεί. Τέλος, τα ΑΣΔ αποτελούν έναν από τους πολλούς τρόπους συνδεσιμότητας στρωμάτων σε νευρωνικά δίκτυα, περαιτέρω έρευνα και εξερεύνηση σε διαφορετικές ιδέες συνδεσιμότητας και συνδυασμό κοντινών και μακρινών συνδέσεων είναι φυσικό επακόλουθο.

Chapter 1

Introduction

1.1 Motivation

In recent years, deep learning has demonstrated remarkable success across a wide range of tasks and modalities, often matching or even exceeding human-level performance (10; 11; 12). The applications of these models are vast and diverse, with many focused on enhancing human welfare. To achieve such strong performance, these models rely on deep and expressive neural network architectures containing billions of parameters, which significantly drive up computational and storage demands. As these models continue to grow in size, adhering to scaling laws that link performance to the number of parameters and the volume of training data, they are effectively becoming huge powerhouses, increasing the energy consumption for both training and inference. This leads to significant consequences such as environmental impact and substantial economic costs.

Despite these impressive advances, artificial neural networks remain remarkably inefficient compared to their biological counterparts. While deep learning models require massive datasets and enormous computational resources, biological neural networks achieve superior capabilities with remarkable efficiency. Specifically, biological neural networks learn robust representations from sparse data points rather than millions of examples, are capable of continuously adapting to new information without catastrophic forgetting, generalize knowledge across domains with minimal crossover examples, and operate with exceptional energy efficiency compared to computational demands of artificial systems. This efficiency-performance balance achieved through evolution remains an aspiration for artificial systems. Researchers are exploring several promising directions to address these inefficiencies, effectively enhancing the representations of artificial deep neural networks and making them less computationally expensive. A central aspect of this is the architecture of the neural network, an ingredient that effectively acts as an inductive bias, shapes the training dynamics of the model and affects the representational quality and capabilities of it. The architecture determines how information flows through the network and influences what patterns the network can efficiently capture.

A crucial factor in the efficiency of biological neural networks is their connectivity patterns. Unlike the predominantly layered and densely connected artificial networks, biological neural structures combine both short and long-range connections in sophisticated ways (68). The brain exhibits a multiscale organization where local processing occurs

alongside global integration through strategic long-range projections. This arrangement creates information highways that allow signals to bypass intermediate processing when appropriate, optimizing both processing speed and computational efficiency. Additionally, biological networks employ adaptive pruning through experience-dependent exploration. During development and learning, the brain initially forms numerous synaptic connections, many of which are later eliminated through competitive processes that retain only functionally important connections. This "synaptic pruning" (69) is guided by neural activity patterns that emerge during interaction with the environment, effectively sculpting the network based on experiential demands rather than predetermined architectural constraints (70; 71).

Drawing inspiration from these biological principles, the research community has explored biologically-inspired approaches including sparsely connected models where only a fraction of possible connections are utilized, and growing neural networks that add connections or neurons as needed during training. Neural architecture search techniques automate the discovery of optimal connectivity patterns, effectively exploring the vast functional space of possible architectures to find those that balance performance and efficiency (72).

In another direction, again inspired from these biological principles, researchers have experimented with various connectivity patterns in artificial networks. Skip connections create direct pathways for information flow across multiple layers "information highways"), mitigating vanishing gradient problems while creating shortcuts analogous to long-range connections in biological systems. More recently, attention mechanisms have been introduced dynamically establishing connections between all positions in a sequence, mimicking the brain's ability to selectively focus on relevant information while ignoring irrelevant inputs. More specifically, since the introduction of Highway Networks (39), which first proposed additive skip connections with learned gating, a wide array of architectural innovations have emerged. Residual Networks (ResNets) (40) simplified this approach by replacing gates with identity mappings, becoming the de facto standard. Subsequent models explored alternative connectivity strategies like concatenation (44), recursive, multi-depth compositions (45), learnable (experience-based) weighted combinations (46) and cross-layer attention (48).

While the majority of these architectural innovations—ranging from identity mappings and dense concatenation to attention-based fusion—have primarily aimed to improve task performance, training efficiency, or optimization dynamics, a complementary line of research has raised concerns about the effective utilization of network depth. For instance, (14) demonstrated that randomly dropping layers during training can actually improve generalization and reduce overfitting, suggesting that not all layers are equally essential. Similarly, (52) found that skip connections may lead to certain layers being under-trained or effectively bypassed altogether. More recently, studies have uncovered significant parameter redundancy in large-scale foundation models, particularly in their deeper layers, indicating that much of the model capacity may remain unused (13).

These findings suggest that despite their empirical success, residual-style architectures may suffer from inefficient depth utilization and parameter usage that can poten-

tially harm the generalization capabilities of the model. In this work, we take a step toward addressing this gap by proposing ***Auto-Compressing Networks (ACNs)***, which leverage architectural bias to alter training dynamics and promote efficient and adaptive depth usage, based on the task at hand, throughout learning. Starting from a standard feedforward neural network (FFN), we introduce *long connections* from each intermediate layer (including the input embedding) to the output, which are summed to form a multi-path architectural variant of Residual Networks. By analyzing the resulting gradient dynamics, we uncover implicit *layer-wise training* characteristics, where layers are dynamically utilized throughout learning, and information naturally concentrates in a subset of the network's layers. We refer to this behavior as *Auto-Compression*. This mechanism enables ACNs to learn compressed representations that offer practical advantages, including increased robustness to noise, improved data efficiency, and reduced catastrophic forgetting in continual learning scenarios. Across diverse tasks, ACNs demonstrate substantial architectural compression—akin to synaptic pruning in biological networks, though realized in a layer-wise manner—while maintaining strong generalization. Notably, they match or exceed the performance of Residual Networks, despite using significantly fewer parameters.

1.2 Contributions

The main contributions of this thesis are the following:

- We propose and investigate, both theoretically and empirically, a neural network architectural modification in connectivity patterns, termed Auto-Compressing Networks (ACNs)—a variant of standard feedforward and residual networks. By analyzing the distinct training dynamics introduced by this modification, we uncover a unique property we coin as *auto-compression*: the ability of a network to organically compress information into a subset of its layers during training with gradient descent, through architectural design alone.
- We provide a detailed analysis of the gradient dynamics of ACNs, along with residual and feedforward networks, shedding light on their distinct behaviors and arguing that different connectivity patterns result in unique training regimes that drive distinct learned representations.
- We provide an extensive literature review of prior work on connectivity patterns in neural networks, including variations of short- and long-range connections, and broader efforts that introduce multiple information pathways through skip connections—challenging the traditional paradigm of densely layered, short-range connectivity in standard feedforward architectures.
- We implement ACNs in fully connected and transformer-based architectures, finding that they match or outperform residual baselines, while 30–80% of top layers effectively become redundant as information concentrates in the lower layers. Notably, ACNs are hardware-friendly and require no specialized software.

- We show that ACNs learn representations that are more robust against noise and generalize better in low-data regimes compared to residual architectures.
- We argue that auto-compression offers a natural pathway to continual learning by preserving unused parameters for new tasks and utilizing different parameters for different tasks. We empirically validate this by showing that both naively FT or coupling ACNs with a well-know regularization based CL technique significantly outperforms Residual Networks in forgetting.
- We demonstrate that regularization-based approaches (relying on intermediate losses) that try to compress information into a subset of the full network's layers are highly sensitive to hyperparameter tuning and weaker at transfer learning, compared to ACNs.
- We pair ACNs with widely-used baseline pruning techniques, demonstrating that their organically compressed representations significantly amplify the effectiveness of traditional compression methods, achieving superior levels of sparsity compared to residual architectures.

1.3 Outline

The rest of this thesis is organized as follows:

- **Chapter 2** introduces fundamental concepts and definitions from Machine and Deep Learning that support the remainder of the thesis.
- **Chapter 3** presents the historical development and core ideas of Representation Learning, emphasizing the importance of learning robust representations for machine learning algorithms.
- **Chapter 4** explores Multi-Path architectures, such as Residual Networks. We discuss their motivation, historical evolution, and various design directions. The chapter concludes by highlighting some limitations of Residual Networks, which motivate our proposed approach.
- **Chapter 5** introduces the main contribution of this thesis: a novel architectural variant called *Auto-Compressing Networks*. We describe the architecture, provide theoretical analysis comparing its dynamics to Feedforward and Residual Networks, and present empirical results demonstrating its effectiveness and unique advantages.
- **Chapter 6** concludes the thesis by discussing biological inspirations and key differences from Residual Networks. We summarize our findings, discuss limitations, and propose future research directions.

Chapter 2

Machine Learning

2.1 Introduction

Machine Learning is a sub-field of Artificial Intelligence. According to Murphy (16) "we define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planning how to collect more data!)". In essence, machine learning is the science of learning from data.

In its early stages, the field faced several limitations that hindered its widespread adoption. These included limited computational power, scarcity of large datasets, and a lack of scalable algorithms capable of learning from complex data. As a result, much of early artificial intelligence research focused on rule-based systems, where expert knowledge was encoded manually through "if-then" rules. One prominent example of this approach is ELIZA (73), an early natural language processing program that mimicked conversation through scripted responses without any real understanding or learning capability.

As compute and data became more available, traditional machine learning systems started appearing, typically relying on handcrafted features, domain-specific heuristics, and statistical models. These approaches required significant human expertise to engineer representations and often involved in-depth inspection of the problem domain. Researchers, in effect, were designing the solution architecture manually—limiting the system's ability to generalize and scale, especially in high-dimensional or ambiguous data environments. However, this paradigm began to shift with the advent of neural networks, a class of models inspired by the structure and function of biological neural systems. Instead of manually specifying the solution logic, handcrafting and engineering features, neural networks learn to map directly inputs to outputs through exposure to data—implicitly uncovering patterns from examples, utilizing them for future predictions.

2.2 Types of Learning

Machine Learning can be categorized into different paradigms based on the nature of the available data and the presence or absence of feedback signals during training, as well as the form that such feedback may take.

2.2.1 Supervised Learning

In the **supervised learning** (16) setting, a dataset D is provided, consisting of input-output pairs:

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}.$$

Each x_i typically represents a D-dimensional input feature vector (though in general, it can be a more complex structure), and each y_i is the corresponding *ground truth label* or *target output*.

If y_i belongs to a *finite set of categories*, the task is known as **classification**, where the goal is to assign each input x_i to a class label y_i. On the other hand, if y_i is a *real-valued* quantity (often a K-dimensional vector), the task is referred to as **regression**.

In supervised learning, the objective is to train a model using a training algorithm to learn the underlying mapping from inputs to outputs—that is, to accurately predict y_i given the corresponding x_i. We assume that each pair (x_i, y_i) is drawn from an unknown target function f^*, such that:

$$y_i = f^*(x_i), \quad \forall i \in \{1, \ldots, n\}$$

This function f^* represents the true mapping that the model aims to approximate. For example, in a classification task, f^* is the function that assigns each input to its correct category.

2.2.2 Unsupervised Learning

In the **unsupervised learning** (16) setting, a dataset D is provided, consisting of inputs only:

$$D = \{x_1, \ldots, x_n\}.$$

Each x_i represents an input feature vector, but in this case, there are no associated labels. Unlike supervised learning, we are not given input-output pairs that can guide the model to learn a mapping. Instead, the task is to discover, without supervision, underlying patterns, structures, or regularities within the data. This setting is inherently more abstract than supervised learning, as there are no ground truth labels to evaluate or guide the model's output. As a result, unsupervised learning requires alternative strategies to uncover hidden structures—such as clustering or dimensionality reduction—with the aim of understanding, explaining, or organizing the data in a meaningful way.

2.2.3 Reinforcement Learning

In **reinforcement learning** (17), feedback is provided alongside input features, but unlike supervised learning, it does not take the form of explicit correct outputs (such as the correct category). Instead, the feedback is weaker and indirect—typically indicating only whether the model's output was correct or not. The challenge there is to interpret and use this limited signal effectively, so that the model gradually learns, through trial

(producing outputs) and error (receiving the feedback signal), to produce outputs that lead to positive feedback more consistently and efficiently.

2.3 Parametric vs Non-Parametric Models

A broad distinction among machine learning models can be made based on whether they feature a fixed number of parameters-—referred to as *parametric models*—or a number of parameters that grows with the amount of training data—known as *nonparametric models* (16). Parametric models are typically faster and more efficient to use (constant complexity as a function of the training set size), but they come with the drawback of making stronger assumptions about the underlying data (e.g., the target function f^*). In contrast, nonparametric models can model a wider range of functions, however, they often become computationally expensive for large datasets, as the number of parameters can grow significantly with the size of the data.

Parametric models are designed by embedding certain assumptions about the underlying data into the model's parameters—assumptions commonly referred to as *inductive biases*. These biases define different families of models, each with distinct capabilities and characteristics, providing practitioners with a range of options tailored to the specific nature of the problem at hand. A fundamental distinction among these model families lies in whether the model's output is a *linear* or *non-linear* function of the input. This distinction significantly impacts both the expressiveness of the model and the training algorithm used. Linear models are generally simpler and easier to train, but they are limited in the kinds of patterns they can capture. Non-linear models, on the other hand, are more powerful and flexible, capable of capturing complex relationships, but they typically require more sophisticated training procedures.

Why nonlinear transformation of the input? A classic and insightful example that motivates the use of non-linear transformations is the **XOR problem**. Consider a binary classification task where the input vector $x = (x_1, x_2) \in \{-1, 1\}^2$ and the label is defined as:

$$y = x_1 \oplus x_2,$$

where $\oplus$ denotes the logical XOR (exclusive OR) operation. The four input-output pairs are:

$$(-1,-1) \to 0, \quad (-1,1) \to 1, \quad (1,-1) \to 1, \quad (1,1) \to 0.$$

This problem is *not linearly separable* in the input space, as no straight line can separate the points with label 1 from those with label 0 (see Figure 2.1 (left)), meaning that we cannot find values for ∂ that solve the problem. However, if we introduce a non-linear transformation, for example:

$$\phi(x) = (x_1, x_1 \cdot x_2),$$

we effectively map the data into a different feature space where a linear classifier can now separate the two classes, as shown in Figure 2.1 (right).

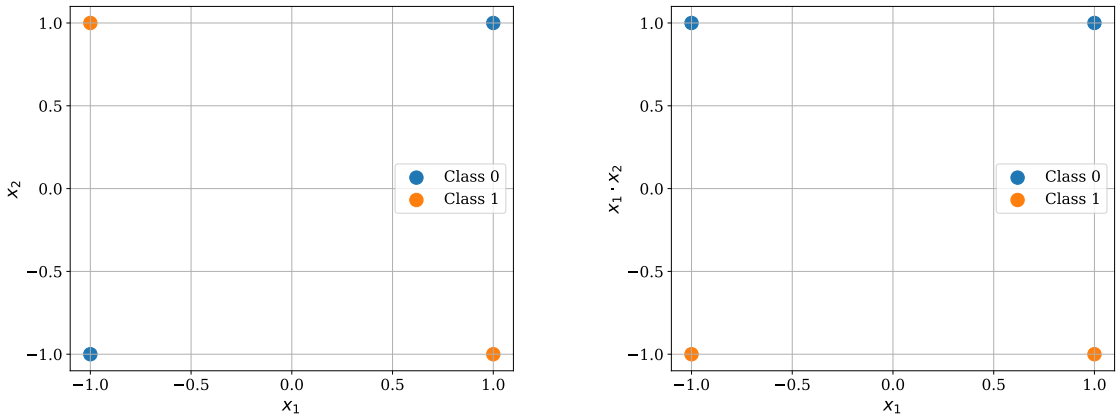


Figure 2.1. *The XOR problem in the original input space (left) and in a transformed feature space (right), where it is linearly separable.*

2.4 Training

Training is the process of learning the parameters of a parametric machine learning model, based on a given dataset D. Two core components are central to this process: the *loss function* and the *training algorithm*.

Loss Function A loss function defines the task that the model is intended to solve and quantitatively measures how well the machine learning model performs on it. For example, in supervised learning, it measures the discrepancy between the model's predicted output and the ground truth label. It is referred to as a function because it takes as input both the model's prediction and the target output, and returns a value indicating the quality of the prediction. The term *loss* reflects the idea that the function quantifies how much the model "loses" or underperforms relative to the ideal case, where all predictions perfectly match the ground truth labels and the loss is zero. The loss function plays a central role during training, as it defines the objective that the model seeks to minimize.

Training Algorithm While the loss function quantifies the model's performance and serves as a kind of "progress bar" during training, it does not by itself specify how to adjust the model's parameters to improve that performance—that is, to further minimize the loss. This is the role of the *training algorithm*, which provides a set of rules or equations that determine how each parameter should be updated in order to minimize the loss function effectively.

These two components, along with the model family (e.g., linear or non-linear models), are deeply interconnected. The choice of model family influences the properties of the loss function (such as convexity), which in turn determines the nature of the training algorithm used to minimize it. Two broad categories of training algorithms arise based on these properties: *closed-form* (or global) solutions and *iterative* optimization methods.

In closed-form solutions, one can derive an explicit analytical expression that directly specifies the optimal value of each parameter to minimize the loss function. This is often

possible when the model is linear and the loss function is convex. In contrast, when using non-linear models, the resulting loss function is typically non-convex and does not permit a closed-form solution. In such cases, training relies on iterative optimization methods—most commonly gradient-based approaches like *gradient descent*—where the update rules specify how each parameter should be adjusted incrementally based on the gradient of the loss with respect to that parameter. These updates are repeated until the loss is sufficiently minimized.

While convex optimization converges, in theory, starting from any initial parameters gradient descent applied to non convex loss functions has no such convergence guarantee and is sensitive to the initial values of the parameters.

2.5 Generalization, Overfitting and Regularization

The central goal of any machine learning algorithm is to perform well on newly introduced, unseen inputs. In the context of supervised learning, this means that after training on a given dataset D, the model should be able to make correct predictions for an unseen input x', where $x' \notin D$. This indicates that the model has successfully learned the underlying mapping from inputs to outputs, i.e., it has approximated the function f^* such that $y = f^*(x)$.

To evaluate this capability, practitioners typically reserve a separate dataset D_{test}, containing input-output pairs drawn from the same true distribution as the training data. This test set is not used during training and serves as a proxy for assessing the *generalization* ability of the model—its capacity to perform well on unseen data.

Why not evaluate performance solely on the training set? The key issue is that the model has already seen the training data during the learning process and may have *memorized* it, especially if the model is highly expressive (consisting of a lot of parameters) and/or the data are sparse (so a trivial solution can be easily found). In such cases, the model may achieve very low training error without actually learning meaningful abstractions or rules that generalize beyond the training set. Consequently, although the model performs well on D, it may perform poorly on D_{test}, revealing that it has learned a function f that diverges significantly from f^* (outside of D). This phenomenon is known as *overfitting*—when a model fits the training data too closely and fails to generalize to new inputs.

A common technique to prevent overfitting and improve generalization is *regularization*, which involves introducing constraints on the model's parameters. The core idea is to discourage the model from learning trivial or overly complex solutions that simply memorize the training data, and instead guide it to utilize its parameters in a way that captures meaningful abstractions from the data. These constraints typically act as priors that promote certain desirable properties—such as sparsity or low parameter norms—that have been empirically shown to enhance generalization. Regularization thus serves as a form of inductive bias, steering the learning process toward solutions that are more likely to generalize well to unseen data.

2.6 Neural Networks

Neural networks are among the most prominent machine learning models today, playing a key role in the rise of the field and serving as the foundation of modern *Deep Learning*. Inspired by the structure of the brain—specifically biological neural networks, which are responsible for learning and organizing information, among others, in the brain—researchers introduced *Artificial Neural Networks* (ANNs) as computational analogs (18; 19). ANNs are composed of nodes (also called neurons) and weighted connections between them, organized into layers. In their simplest form, known as *feedforward networks*, neural networks are structured as directed acyclic graphs through which input features are passed and progressively transformed by two core mechanisms: **weighted sums** and **non-linear activation functions**. These transformations are applied layer by layer, ultimately producing the network's output.

At their core, Artificial Neural Networks (ANNs) are parametric machine learning models, where the parameters—typically denoted as θ—correspond to the weights of the connections between neurons. These parameters are learned from data and adjusted through training, based on a training algorithm, to *approximate an underlying function of the data*, i.e. $f = f(x; \theta)$. In a supervised learning setting, for example, neural networks are trained to approximate the target function f^*, based on the given dataset D. The goal is to learn the parameters θ such that, for a given input x_i, the network's output $\hat{y}_i = f(x_i; \theta)$ closely matches the true output $y_i = f^*(x_i)$. In other words, the training process adjusts the parameters θ so that the model can accurately produce the correct output (i.e., match the ground truth label y_i) for each corresponding input x_i.

As a simple example, consider a single-layer neural network that receives a d-dimensional input vector $x = [x_1, x_2, \ldots, x_d]$. The network first computes a weighted sum of the inputs, adds a bias term, and then applies a non-linear activation function $\phi(\cdot)$ to produce the final output:

$$y = f(x) = \phi\left(\sum_{i=1}^{d} \theta_i x_i + b\right),$$

where θ_i are the learnable weights, b is the bias term, and ϕ is a non-linear activation function such as `ReLU`, `sigmoid`, or `tanh`. Using vector notation, this can be written as:

$$y = f(x) = \phi(\theta^\top x + b).$$

2.6.1 Training

ANNs are predominantly trained with gradient-based learning, which iteratively modifies the parameters θ of the network in order to improve the predictions of it. This comes from the fact that ANNs' power and capabilities come from the introduction of non-linear functions, thus making most interesting and commonly used loss functions, nonconvex. The algorithm used to train these models is called **Backpropagation** (20). In essence, this algorithm computes the gradients of the loss function with respect to each parameter by

leveraging the compositional structure of neural networks, thereby deriving the gradient descent update rules used to iteratively minimize the loss during training. It consists of two main phases:

1. Forward pass: The input is propagated through the network to produce a prediction. This predicted output, along with the target label, is passed to the loss function, which computes a loss value indicating the model's performance.

2. Backward pass: The loss is then propagated backward through the network, starting from the output layer. Gradients of the loss with respect to each parameter are computed, layer by layer, and used to update the parameters accordingly.

A key aspect of backpropagation in deep neural networks is that the gradient update for the parameters of layer l depends on the gradients from layer $l + 1$. As a result, the backward pass must proceed sequentially from the final layer (close to the loss function) toward the input layer, making the entire process inherently sequential.

2.7 Deep Learning

Deep Learning is a subfield of machine learning that focuses on models with deep compositional structures—most notably, deep neural networks—to solve complex tasks. Its popularity surged following a major breakthrough in image classification (i.e. classifying images into different categories like car or bird), where a deep neural network significantly outperformed traditional approaches in a benchmark competition (10). This marked the first compelling evidence that deep neural networks could surpass classical machine learning methods, which relied on handcrafted features and domain-specific engineering, and had previously dominated the field.

2.7.1 Deep Feedforward Networks

Deep Feedforward Networks are the simplest form of neural networks with multiple layers. Each layer consists of neurons that are *fully connected* to the next one, meaning every neuron is connected to all neurons in the subsequent layer. These networks are called *feedforward* because they form a directed acyclic graph through which the input flows in one direction—forward—being progressively transformed at each layer via weighted sums and non-linear activation functions, until the final output is produced, as mentioned previously.

Because of this layered structure, feedforward networks are typically expressed as compositions of functions. Specifically, each layer i applies a function $f^{(i)}$, and for a network of depth L, the full model is written as:

$$f(x) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(x).$$

This can be viewed as the input being gradually transformed into increasingly abstract *representations* as it traverses the layers.

A helpful way to understand the power of these transformations is through the XOR example discussed earlier. In the raw input space, no choice of parameters θ could solve the classification task. However, with a suitable non-linear transformation of the input, we were able to move to a feature space where the problem became linearly separable. While such a transformation can be engineered manually for simple, well-understood problems like XOR, this is rarely feasible in complex, real-world settings. This is precisely where deep neural networks prove valuable: they automatically learn powerful, non-linear transformations through their layered architecture. These transformations produce feature spaces—such as $f^{(1)}(x), f^{(2)}(f^{(1)}(x))$, and so on—where the problem may become linearly separable, even when that was not possible in the original input space.

2.7.2 The value of depth

But why depth? Theoretically, it has been well established that an ANN with a single hidden layer containing a sufficient number of sigmoid units can approximate any decision boundary (74). This foundational result implies that, in theory, shallow networks have the same representational power as deep networks—that is, they can approximate the same class of functions.

However, deep neural networks are not just about expressivity; they are motivated by the principle of hierarchical feature learning. These models, as described before, build multiple levels of abstraction and compositional representations, where the input x is successively transformed into increasingly abstract latent features: first into $f^{(1)}(x)$, then $f^{(2)}(f^{(1)}(x))$, and so on, resulting in feature spaces that are better-suited (e.g. more discriminative in the case of classification) for the task at hand.

Several explanations have been proposed to account for the practical advantages of depth. It has been both shown that deep networks can represent certain families of functions much more efficiently—that is, using significantly fewer parameters—than their shallow counterpart (75). This has been further investigated and proved for the case of compositional functions (76), and it has been further argued that many real-world problems can be efficiently solved by compositional algorithms, like deep neural networks (76).

Empirically, shallow networks have been shown to match the generalization performance of deep networks only when trained to mimic their outputs in a teacher-student setup (77). Moreover, it has been observed that, under a fixed parameter budget, deeper networks tend to generalize better than shallow ones (78).

Another influential line of work (79; 80) has demonstrated that depth, when combined with more traditional learning techniques such as layer-wise unsupervised pre-training, can act as a powerful inductive bias for parameter initialization, improving generalization. Moreover, deep models trained in this manner have been observed to "disentangle" the underlying structure of the data more effectively—that is, to represent inputs in terms of underlying explanatory factors that help the model generate accurate predictions.[1]

[1]Here we introduce, somewhat briefly, the term *explanatory factors*, which commonly refers to the latent causes or generative factors of the observed data. Discovering these factors can significantly aid in solving the learning problem. For a more thorough discussion, see (21).

2.7.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (22; 23) are a specialized class of neural networks designed for processing data with a grid-like topology, such as images. Unlike fully connected networks that use dense matrix multiplications, CNNs apply local convolutional filters, meaning that each node is connected only to a local neighborhood of the input rather than to every other node.

They typically consist of multiple layers, where early layers emphasize local interactions between parts of the image, enabling the network to detect simple spatial, geometric patterns such as edges or textures. As we move to deeper layers, CNNs progressively integrate information from larger portions of the input, allowing them to detect increasingly abstract and semantic features, such as object parts or entire objects. CNNs incorporate two key inductive biases, inspired by the human visual system and properties of natural images:

1. **Sparsity/Locality of Connections:** Neurons are only connected to small, localized regions of the input, based on the idea that meaningful visual features often arise from local patterns.

2. **Weight Sharing:** The same set of weights (i.e., a convolutional filter) is applied across different spatial locations, enabling the network to detect the same feature (e.g., a vertical edge) in multiple parts of the image.

By combining these properties with a hierarchical structure, CNNs effectively extract local spatial features in early layers and progressively build higher-level semantic representations in deeper layers. This layered abstraction enables CNNs to learn powerful and efficient representations for image recognition and related tasks.

Convolutional Neural Networks (CNNs) played a crucial role in the rise of Deep Learning, as they were among the first neural network architectures to demonstrate strong performance in pattern recognition tasks—most notably in handwritten digit recognition (22). Their impact became especially prominent in 2012, when AlexNet (24), a CNN-based model, won the ImageNet classification competition (25). This was the first time a neural network achieved such a result in this large-scale visual recognition challenge, marking a turning point and sparking a renewed wave of research and interest in deep learning. To illustrate the hierarchical feature learning of CNNs, the authors include in their paper figures showing some of the filters learned by the first layer of AlexNet. These filters highlight simple geometric patterns—such as edges and color gradients—that the network learns to detect in its early layers. Each filter effectively scans different regions of the image, identifying whether a specific feature (e.g., a vertical edge) is present in that region.

2.7.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (26; 27) are another class of neural networks, also inspired by biological observations suggesting recurrent (feedback) connections in the

brain, designed specifically for processing sequential data, such as language, speech, or time-series signals. Unlike feedforward networks, RNNs contain cycles in their connections, meaning that the output of a unit can be fed back into itself, directly or indirectly. This feedback loop allows the network to retain information from previous steps, effectively creating a form of memory. Typically, they receive input sequences as vectors at different time steps—for example, feature vectors representing words in a sentence when processing language.

The key idea is that when processing input at the current time step, the network also uses a hidden state vector, a *context vector*, that summarizes information from all previous time steps. This context vector acts as context, influencing predictions at later points in the sequence. This temporal feedback and memory mechanism distinguishes RNNs from feedforward and convolutional networks, which lack such a notion of sequential context. Importantly, the length of this context is not fixed; the hidden state can, in principle, carry information from the very start of the sequence up to the current step.

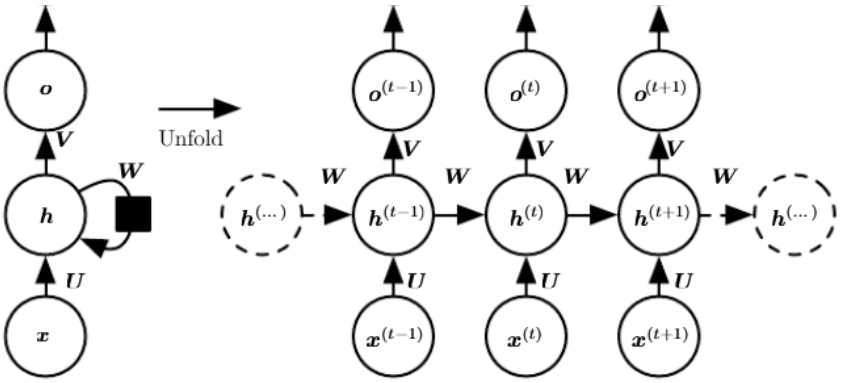


Figure 2.2. *A graphical illustration of an RNN (figure taken from (1)).*

RNNs perform two key operations at each time step by processing the current input together with the previous context: (1) they produce an output, whose nature varies depending on the task—for example, predicting the next word or classifying the part of speech, and (2) they update the context vector, incorporating new information from the current input to be used in future steps. A graphical illustration of this architecture is shown in Figure 2.2. As described, each input is processed along with the hidden vector that summarizes past information through recurrent connections. The network then produces an output and updates the context vector for the next time step.

Another observation and a shared characteristic with CNNs, is that RNNs also use weight sharing: the same set of weights (denoted as *U, W, V* in the figure) is applied at every time step in the sequence. This allows the network to efficiently process variable-length inputs and learn meaningful abstractions or patterns that generalize across different parts of the sequence. Finally, it is common for RNNs to consist of multiple layers, especially when solving complex tasks, to increase their expressivity and representational power.

While RNNs had been the primary choice for sequential tasks, achieving notable success in language understanding and generation, they exhibit a key limitation as the length of the input sequence increases. This limitation arises from the fixed dimensionality of the

context vector, which constrains the amount of information it can retain. Consequently, information from earlier positions in long sequences tends to be lost or diluted over time. The context vector plays a crucial role in sequential processing, as it summarizes relevant past information essential for understanding dependencies within the sequence. Therefore, the inability to effectively preserve and access information from distant time steps can significantly degrade the overall performance of RNNs on long sequences. Despite the introduction of more sophisticated RNN variants such as Long Short-Term Memory networks (LSTMs) (28), which incorporate gating mechanisms to selectively write, delete, and read information from the context vector, significant improvements in long-range context integration and sequential processing were only achieved with the advent of a new architecture, discussed in the following section.

2.7.5 Transformers and the Attention Mechanism

Transformers (29) handle long-term dependencies far more effectively than previous architectures. Since their introduction, they have propelled deep learning to unprecedented prominence and have become the foundational architecture behind today's widely known large language models, which power assistants and chatbots. At the heart of the Transformer architecture lies the *attention mechanism* (30)—a powerful method for dynamically routing information across different positions in a sequence. This mechanism enables the model to capture complex dependencies between inputs at different timestamps, regardless of their distance from each other.

For each input vector at time step i, the attention mechanism computes a new representation y_i by attending to all other input vectors x_j (for $j \leq i$) and aggregating them according to their relevance to x_i. This is typically done in three steps:

1. **Scoring:** Compute a relevance score between x_i and each x_j, using a similarity function such as the dot product (often scaled):

$$\text{score}(x_i, x_j) = x_i \cdot x_j$$

2. **Weighting:** Normalize the scores using the softmax function to obtain attention weights:

$$a_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{k \leq i} \exp(\text{score}(x_i, x_k))}$$

3. **Aggregation:** Compute the output vector y_i as a weighted sum of all input vectors, where the weights reflect the relevance of each input to x_i:

$$y_i = \sum_{j \leq i} a_{ij} x_j$$

This process allows each position in the sequence to dynamically integrate information from the entire history of the sequence up to that point directly, and not through a fixed-sized context vector.

A limitation of the basic attention formulation described earlier is that the same input vector x is used for three distinct purposes:

- To **query** other vectors (i.e., to look for relevant information),
- To act as a **key** (i.e., to represent information that others may attend to),
- To serve as a **value** (i.e., the actual content to be retrieved when attended to).

Using the same representation for all three roles can limit the model's expressiveness. To address this, Transformers introduce three distinct learned linear projections of the input vector, mapping it into separate subspaces dedicated to each role. These projections are represented by matrices W^Q, W^K, and W^V, which transform each input x into:

$$q = W^Q x, \quad k = W^K x, \quad v = W^V x$$

Here, q, k, and v are referred to as the *query*, *key*, and *value* vectors, respectively.

Given a sequence of input vectors (e.g., $X = [x_1, x_2, \ldots, x_n]^T$), we can compute the attention output for all positions in a matrix form as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.1}$$

Where:

- $Q = XW^Q$ is the matrix of query vectors,
- $K = XW^K$ is the matrix of key vectors,
- $V = XW^V$ is the matrix of value vectors,
- d_k is the dimensionality of the key vectors (used to scale the dot product),
- softmax$(\cdot)$ is applied row-wise to produce normalized attention weights.

This final formulation enables each position in the sequence to attend selectively to others, based on content similarity, with greater expressiveness due to the separation of roles via distinct projections. A graphic illustration of the process is shown in Figure 2.3.

Transformers, therefore, are composed of multiple identical layers—referred to as *Transformer blocks*—each of which integrates a self-attention mechanism and a position-wise feedforward network. These components are augmented with residual (skip) connections and layer normalization to facilitate stable and efficient training of deep architectures, as illustrated in Figure 2.3. This modular design enables the network to model complex dependencies and hierarchical representations across sequence elements.

Finally, each Transformer block typically employs *multi-head attention*, which consists of multiple parallel attention mechanisms, each with its own set of learned projection matrices (W^Q, W^K, W^V), as illustrated in Figure 2.4. The idea behind this design is to allow different heads to learn different rules. Each attention "head" learns to capture potentially distinct patterns or dependencies within the input sequence—for example,

some heads may focus on syntactic relationships while others learn to capture semantic associations. The outputs of all heads are then concatenated and linearly projected to form the final attention output.

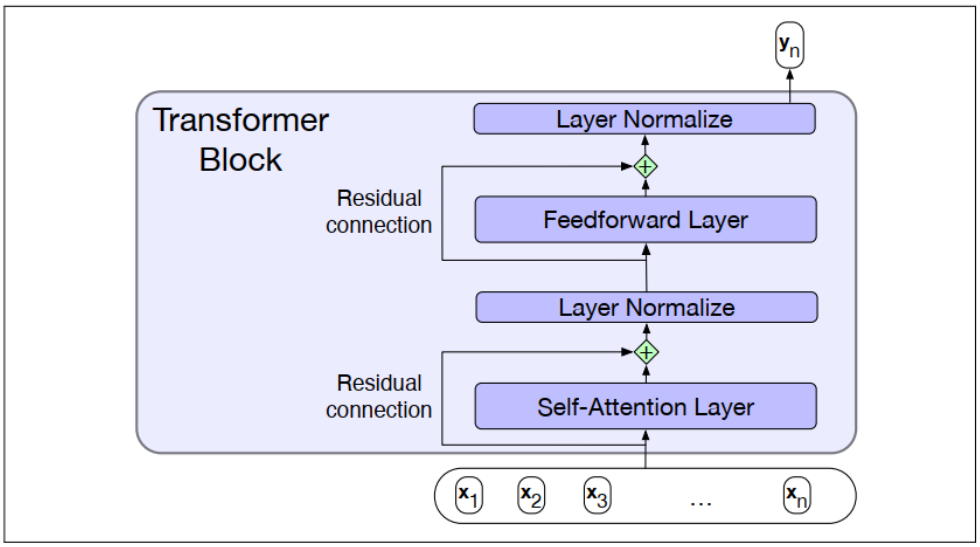


Figure 2.3. *A graphical illustration of a Transformer block (figure taken from (2)).*

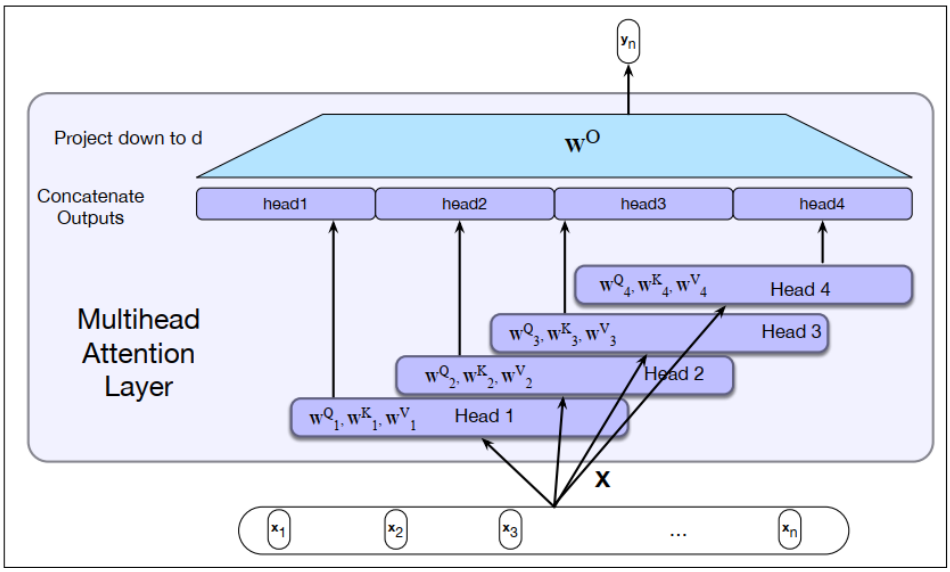


Figure 2.4. *A graphical illustration of Multi-Head Attention (figure taken from (2)).*

Chapter 3

Representation Learning

3.1 Introduction

In the previous sections, we introduced the central ideas of machine learning, including the different types of learning, the goal of learning, and how it is typically performed. As discussed, machine learning is fundamentally the science of learning from data. A crucial aspect of this process is how data are represented. For example, in the XOR problem, we observed that using the raw input features led to failure, whereas applying a well-chosen non-linear transformation—based on insights into the structure of the data—enabled a successful solution. This illustrates that the performance of machine learning models is highly dependent on the quality of the data representation, or features. This raises a fundamental question: *what constitutes a good representation of the data, and more importantly, how can we discover such representations in a general and automated manner?*

To formulate the problem more concretely, we adopt the perspective of supervised learning. We begin with linear models and gradually extend to neural networks and deep representations. As discussed earlier, the goal of supervised learning is to learn a function $y = f(x; w)$, where f belongs to a family of functions parameterized by learnable weights w. In the simplest case of linear models, we have $y = w^\top x$, representing the family of linear decision boundaries. However, this family has limited representational capacity and can only solve linearly separable classification problems. To increase expressivity, we can introduce a non-linear transformation $\phi(x)$ of the input, leading to the model $y = w^\top \phi(x)$. In this formulation, $\phi(x)$ defines a new set of features. As seen in the XOR example, identifying an appropriate non-linear transformation can map the problem into a space where it becomes linearly separable, allowing effective learning using simple linear models. In (21), the authors hypothesize that good representations can effectively *disentangle the underlying explanatory factors of variation* in the data. In simpler terms, we can postulate that good representations are the ones that makes learning the task at hand easier. That is, they can reveal meaningful patterns or structure inherent in the data that are useful for solving the target task. This perspective aligns with the XOR example, where an appropriately chosen transformation exploited the structure of the input and revealed a space in which the problem became trivial to solve. That also means that the choice of representation will usually depend on the target task.

Feature Engineering While in this simpler case we were able to identify an effective representation through direct observation and intuition, this approach does not scale to complex, high-dimensional real-world problems. As such, traditional machine learning pipelines trying to tackle these complex, real-world problems, were often requiring significant domain-specific knowledge and expertise. Practitioners would spend considerable effort analyzing data and designing handcrafted representations tailored to the specific task. This process, known as *feature engineering*, involved constructing pre-processing steps and data transformations after carefully studying the underlying nature of the problem, in an effort to discover representations suitable for the task.

Although feature engineering enabled many early successful applications across various domains, it has several limitations. It is labor-intensive, requires deep problem-specific insight, and is difficult to generalize across different domains. More importantly, it exposes a fundamental limitation of traditional learning algorithms: their inability to autonomously extract and organize meaningful abstractions from raw data. Instead, they rely heavily on human-crafted features to operate effectively, a reliance that hinders scalability, robustness, and cross-domain generalization—qualities.

At a more conceptual level, as argued in (21), truly intelligent systems must go beyond surface-level pattern recognition and strive to understand the world in a deeper, more structured manner. This form of understanding can only emerge if a system is capable of identifying and disentangling the underlying explanatory factors that give rise to the observed data. In other words, intelligence involves uncovering the latent causes and abstract concepts that are hidden within the raw, low-level sensory inputs—much like how humans perceive and interpret their environment. Learning such representations is not merely a technical convenience but a fundamental prerequisite for developing robust, generalizable, and human-like learning systems.

Learning Deep Representations The reliance on human-designed feature engineering changed significantly with the rise of deep neural networks. Instead of depending on hand-crafted features and manually defined data representations, deep networks are capable of learning these representations ϕ directly from the data. This process is guided only by a few general and simple inductive biases introduced by researchers. In this setting, the model takes the form

$$y = w^\top \phi(x) = w^\top \phi(x; \vartheta),$$

where the representation function ϕ is parameterized by ϑ and learned jointly with the weights w during training. This formulation is quite general, allowing us to transition from simple models—where the function family ϕ is limited in expressiveness—to highly complex models with rich representational capacity. Additionally, it is flexible in terms of incorporating human prior knowledge, as any desired inductive biases or domain-specific assumptions can be encoded directly into the design of ϕ.

As an example, we can view deep feedforward networks for supervised classification as performing a form of representation learning. Typically, the final layer of such networks

is a simple linear classifier, which we can denote as w. The preceding layers are responsible for learning a transformation of the input into a new representation that makes classification easier, i.e. go from x to $\phi(x;\theta)$. Because the entire network is trained using a supervised objective, the hidden layers—especially those closer to the output—tend to learn representations that progressively disentangle the underlying structure of the data. As a result, classes that are not linearly separable in the original input space may become linearly separable in the space defined by the top hidden layers. In essence, this mirrors the same approach we followed in the XOR problem: transforming the input into a space where the task becomes linearly separable. However, in the case of deep neural networks, this process is fully automated—the network learns the appropriate transformation of the input on its own, effectively discovering a useful representation for the task at hand.

Disentangling Representation and Task Learning While the previous example focused on the supervised learning setting, deep representations can also be learned in a purely unsupervised manner. In this case, the goal is to uncover patterns and meaningful abstractions inherent in the data—often by attempting to model the underlying data distribution itself. Once such representations are learned, they can be used to a variety of target tasks involving the same input domain. These tasks can then be learned with minimal supervision, typically requiring only a small number of labeled examples to adapt the learned representations to the specific objectives.

The core hypothesis is that learning data representations and task-specific mappings can be effectively decoupled. In the framework $y = w^T\phi(x;\theta)$, we can separate the learning process into two complementary phases: first learning the representation function $\phi(x;\theta)$ to capture the underlying structure and patterns in the data, then learning the task-specific mapping w. The key insight is that once the model has learned to organize and understand the intrinsic structure of the data through θ, only a small number of labeled examples are needed to learn the task-specific parameters w. This is because the representation $\phi(x;\theta)$ already encodes rich, generalizable features that can be readily adapted to target tasks through simple transformations. This decoupling is particularly powerful because learning good representations θ from large amounts of unlabeled data provides a strong foundation that significantly reduces the annotation requirements for downstream tasks.

In the upcoming sections, we discuss why deep neural networks are able to learn these useful representations, presenting hypotheses about the role of inductive biases embedded in their design, the training algorithms and objectives employed to optimize these representations, and finally how such general representations can be effectively utilized across a variety of downstream tasks.

3.2 Greedy Layer-wise Pre-training

Historically, one of the earliest successful and well-known cases of deep representation learning was unsupervised pre-training. In the early development of deep neural networks, training all layers jointly using a single objective often proved ineffec-

tive. A breakthrough approach that emerged was *greedy layer-wise unsupervised pre-training* (31; 32; 81), which became one of the first demonstrations of how decoupling representation learning from task-specific learning could be effective.

The core idea involves training the deep network one layer at a time in an unsupervised manner: at each step, a single layer l is trained with an unsupervised objective, to extract useful features from the output of the previously trained layers, while the parameters of all preceding layers are frozen (i.e., not updated). This unsupervised phase allows the model to learn hierarchical representations that capture the underlying structure of the data. After pre-training, a second phase known as *supervised fine-tuning* follows. In this step, a classification head is placed on top of the final hidden layer and trained using labeled data to solve a specific task. Fine-tuning can involve updating only the parameters of the classification head or jointly optimizing the entire network. This two-stage training process helped overcome optimization difficulties and demonstrated the value of learning general-purpose representations prior to task-specific learning.

But why does this two-stage learning procedure work? The authors of (80) directly address this question, offering arguments supported by empirical evidence. They propose that greedy layer-wise unsupervised pre-training acts as a form of regularization, effectively guiding the model parameters toward regions of the parameter space that are more constrained and structured. This resembles the role of traditional regularization techniques, which discourage overfitting by limiting the model's capacity to fit arbitrary patterns in the data. More importantly, they argue that unsupervised pre-training leads to parameter configurations that are often (the cases that it is successful) well-suited for supervised fine-tuning—provided that the features learned from the data, i.e., learning a function $f(x)$, are relevant to the downstream supervised task. In the paper, they formulate it from a probabilistic perspective, postulating that this unsupervised pre-training is useful for supervised fine-tuning when *"learning $P(x)$ is useful for learning $P(y|x)$"*.

3.3 Joint Supervised Pre-training and Transfer Learning

Following the success of unsupervised pre-training, a parallel yet conceptually distinct approach emerged: *supervised pre-training followed by transfer learning*. Rather than relying on unsupervised objectives to initialize deep networks, this method leverages large labeled datasets—such as ImageNet in vision (82) or SNLI in text (83)—as a source of supervision for learning general-purpose representations. In contrast to greedy layer-wise schemes, these models are trained end-to-end from the outset using standard supervised training objectives, learning features that not only support the source task but also transfer effectively to new downstream tasks.

The core intuition is similar: decoupling representation learning from task-specific learning. By first training on a broad and diverse labeled corpus, models acquire a strong prior over useful input structures, which can then be refined for target tasks with limited supervision. This transfer typically takes the form of fine-tuning, either updating all model parameters or just a subset, depending on data availability and task similarity. In vision, pre-training on ImageNet followed by fine-tuning on CIFAR-10 (5), Pascal VOC (84),

or medical imaging benchmarks (85) became a widely adopted pipeline. In NLP, similar gains were observed by fine-tuning models trained on entailment or sentence classification tasks (33; 86). Although the reliance on labeled data limited scalability compared to emerging self-supervised methods, supervised pre-training laid crucial groundwork for the transfer learning paradigm that underlies many modern systems.

3.4 Self-Supervised Learning

Finally, *Self-Supervised Learning* emerged as a powerful form of unsupervised learning that leverages unlabeled data to generate pseudo-labels for training. This approach effectively addresses one of the fundamental limitations of traditional supervised learning: the scarcity and high cost of labeled data.

The convergence of three technological advances in the late 2010s catalyzed an explosion in self-supervised learning capabilities. First, the availability of massive amounts of unlabeled data from the internet provided the raw material needed for these hungry algorithms. Second, the proliferation of powerful GPUs enabled the computational muscle required to process datasets at unprecedented scales. Third, and perhaps most crucially, the Transformer architecture proved exceptionally well-suited for parallel processing and capturing long-range dependencies in sequential data.

BERT (Bidirectional Encoder Representations from Transformers) (87) was the first major breakthrough in self-supervised learning for natural language processing. It demonstrated that pre-training on large-scale unlabeled text corpora using the *masked language modeling* objective—where the model learns to predict randomly masked tokens based on their surrounding context—could yield a single model that achieves state-of-the-art performance across a wide range of NLP tasks with minimal task-specific fine-tuning. This highlighted the power of representation learning on unlabeled data and its strong complementarity to downstream tasks, as only a small amount of labeled data is required for fine-tuning. A key factor in BERT's success was its bidirectional attention mechanism: each token attends to all other tokens in the input sequence, both preceding and succeeding. This allowed BERT to capture rich contextual dependencies from both directions simultaneously, resulting in deeper and more informative representations.

Building on this foundation, the GPT (Generative Pre-trained Transformer) series (35) showed the remarkable potential of *autoregressive* language modeling at scale. By simply training transformers to predict the next word in a sequence across internet-scale text corpora, these models developed increasingly sophisticated language understanding and generation capabilities from completely unlabeled data. Again, minimal supervised fine-tuning can lead to effective downstream task adaptation.

The core idea behind these models, pre-trained using self-supervised learning algorithms, is to act as generalists—learning to understand and represent the input modality (in this case, language) in a way that downstream tasks can leverage with minimal task-specific tuning. By capturing broad, reusable features during pre-training, these models provide a strong foundation for a wide range of target applications. Finally, it is worth noting that these self-supervised learning strategies have been successfully extended to

other modalities, such as vision (36), where the core idea remains the same: learning general-purpose visual representations from unlabeled data that can be effectively transferred to a variety of downstream tasks, such as image classification or object detection in an image.

Chapter 4

Multi-Path Neural Network Architectures

4.1 Introduction

We have previously discussed the importance of representations in machine and deep learning, as well as how specific algorithms and architectures can provide better conditions for learning. In particular, incorporating appropriate inductive biases can significantly enhance the efficiency and effectiveness of deep neural networks. In this chapter, we present a historical overview and discussion of the evolution of connectivity patterns in neural networks, focusing on the connectivity between layers and the emergent information pathways throughout the network[1]. The central theme is how information flows from input to output and how gradients are propagated backward during learning.

The evolution of connectivity patterns has played a pivotal role in the progress of deep learning. For several decades, feedforward neural networks dominated the field, establishing themselves as the prevailing paradigm through their conceptual simplicity and extensive study. In these conventional architectures, information flows uni-directionally from input to output layers through sequential transformations, with each layer's activations serving as inputs to the subsequent layer. Essentially, each layer is directly connected only to the next layer. A fundamental challenge in training deep neural networks using gradient descent is the vanishing or exploding gradient problem, where gradients decay or increase exponentially as they are backpropagated through many layers (37; 38). This can happen because essentially both forward and backward signals are propagated through sequences of matrix multiplications, and so their magnitudes can diminish to near zero or explode, depending on the values of these matrices, either case leading to unstable training. This limitation significantly constrained the practical depth and thus the representational capacity of feedforward networks for many years, until the introduction (in a practical and efficient implementation) of a new family of *multi-path* architectures, featuring skip connections between layers that create multiple paths for information to flow inside the network. While multi-path network architectures date back to the 1980s, with early work exploring cascade structures in fully connected networks trained layer by layer to improve training stability (88), it was not until the mid-2010s that these architectures gained prominence and began to dominate the field.

[1] This differs from connectivity between inputs in a sequence, as in RNNs, though related formulations have been explored in the literature.

4.2 Shortuct Connections

4.2.1 Highway Networks

Highway networks (39) were among the first practical multi-path architectures. They introduced gated bypass paths that allowed for effective training of networks with hundreds of layers. Specifically, these architectures modified the input-output relationship of a layer from the standard form $y = F(x)$ to a more general formulation:

$$y = T(x) \cdot F(x) + C(x) \cdot x,$$

effectively introducing additional signal pathways—termed *information highways*—to facilitate the flow of both forward and backward signals. The motivation stemmed from the challenges of training deep neural networks, both in terms of optimization (89) and vanishing gradients problem. By allowing information to flow through the network without modification, these pathways aimed to mitigate these issues. In practice, they set $C = 1 - T$, where:

$$T = \sigma(\mathbf{W}_T^\top x + \mathbf{b}_T).$$

Empirical results from the paper, demonstrated that while shallow feedforward networks trained effectively, deeper plain architectures struggled. In contrast, highway networks exhibited significantly improved training performance with increasing depth, achieving lower training losses. More importantly, this translated to improved generalization: deeper highway networks achieved lower test errors, underscoring the conclusion that depth is indeed beneficial for generalization—provided the architecture supports efficient training.

4.2.2 Residual Networks

In a similar spirit, the authors of (40) found that deeper convolutional neural networks (CNNs) not only suffer from a decrease in generalization performance, often due to overfitting, but also experience a decrease in training performance, underpinning once more the inherent optimization difficulty of training *deep* neural networks. To address this, they introduced residual connections (or identity mappings), proposing that learning residual functions relative to identity mappings simplifies optimization, forming Residual Networks (ResNets). Essentially, Residual Networks are a subset of Highway Networks, removing the learned gating functions and adopted direct identity skip connections, resulting in:

$$y = x + F(x).$$

This simpler form, introduced in this work without the addition of extra trainable parameters, enabled the training of very deep neural networks with strong generalization performance and has since come to dominate modern deep learning architectures. Several studies have investigated the advantages of residual connections over standard feedforward architectures, particularly in the context of training dynamics and signal

propagation.

Specifically, (42) demonstrated that in deep feedforward networks, the correlation of gradients decays exponentially with depth, resulting in gradient patterns that resemble white noise—a phenomenon they term the *shattered gradients* problem. They argue that this phenomenon makes training difficult and unstable since commonly used optimization techniques assume that gradients vary smoothly in the parameter space.In contrast, they show that introducing residual connections mitigates this issue, resulting in sublinear decay in gradient correlation , offering an insight into why residual networks can be trained more effectively as depth increases.

In a complementary line of work, (41) investigate the effect of residual connections on the loss landscape of neural networks. Their findings reveal that: (1) increasing network depth significantly impacts the geometry of the loss surface, introducing high levels of non-convexity, and (2) incorporating residual connections helps to smooth the landscape, reducing this non-convexity commonly observed in deep feedforward networks. An additional insight from their study is the empirical correlation between the geometry of the loss landscape and generalization performance: flatter, less severe non-convex landscapes tend to correspond to lower test error. This offers further support for the argument that residual networks not only improve trainability but also lead to better generalization.

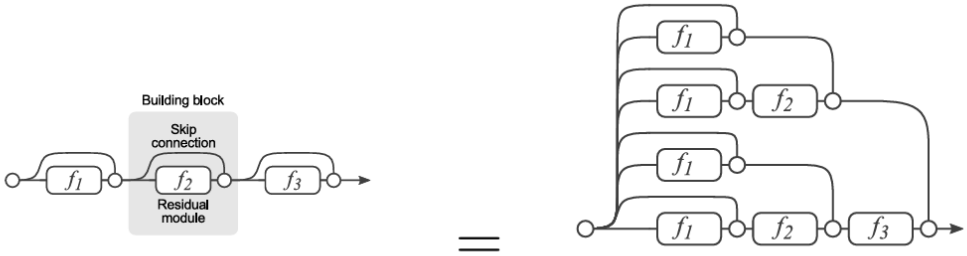
Finally, (43) provided both theoretical analysis and empirical evidence showing that the skip connections in Residual Networks promote improved norm preservation of forward and backward signals. This property leads to more robust and stable backpropagation, effectively solving the vanishing gradient problem. Interestingly, their results indicate that increasing network depth further enhances this norm preservation effect. This work offers a complementary explanation for the success of residual connections and highlights their crucial role in enabling the training of deep networks.

4.2.3 Residual Networks as Ensemble of (relatively) Shallow Networks

In (3), Residual Networks are analyzed through the lens of ensemble learning. An ensemble of neural networks refers to a collection of different models that collectively contribute to producing the final output. The authors argue that a residual network with n layers can be interpreted as a collection of 2^n distinct paths of varying lengths, as shown in Figure 4.1 for 3 layers. At each layer, the signal has the option to either skip the layer or pass through it, leading to an exponential number of possible paths—or *information highways* as discussed earlier.

Despite sharing parameters, these paths behave as an ensemble of dependent subnetworks, as supported by empirical evidence. This stands in stark contrast to traditional deep feedforward networks, which offer only a single computational path from input to output. Specifically, the authors showed that randomly removing or permuting small subsets of layers has minimal impact on overall performance—a common characteristic of ensembles, where the degradation of a subset does not catastrophically affect the whole system. Furthermore, their analysis reveals that most of these paths are relatively shallow, with backward gradients frequently vanishing after traversing only a small portion

of the total depth. This insight leads to the characterization of Residual Networks as behaving like ensembles of shallow networks.



(a) Conventional 3-block residual network (b) Unraveled view of (a)

Figure 4.1. *Paths in a Residual Network (figure taken from (3)).*

4.2.4 Residual Variants

Following the success of Residual Networks, various architectural modifications have been proposed to further improve efficiency and performance. The core idea is that, in essence, Residual Networks combine outputs from all preceding layers at layer i through a static element-wise addition. This fixed form of feature aggregation limits the flexibility of the network in learning more effective representations. Consequently, several works have explored more expressive and adaptive strategies for feature fusion, aiming to enhance representation learning and, ultimately, model performance. To this end, DenseNets (44) proposed using feature concatenation—rather than addition—to model the input-output relationships between layers, enabling more flexible and expressive feature reuse. Similarly, FractalNets (45) introduced a recursive, tree-like architecture that combines subnetworks of varying depths to enhance feature fusion.

More recent works have explored learned weighted averaging of layer outputs (46), attention-based fusion across block outputs (48), and denser inter-layer connectivity patterns (47). Other studies have focused on improving the residual pathways directly by introducing scalar gates or modulation mechanisms in either the residual or main stream, aiming to boost training stability and representation quality (49; 50; 15; 51). In the domain of neural machine translation, researchers have drawn inspiration from both vision and language architectures to combine information across layers, facilitating richer semantic and spatial propagation throughout the network (90; 91).

4.2.5 Potential Redundancy in Residual Architectures

While Residual Networks, as discussed, offer numerous advantages—such as more robust training and improved generalization—there remain several aspects that are worth some further examination and discussion. As highlighted in the section interpreting Residual Networks as ensembles, these architectures exhibit a notable resilience to layer dropping and permutation. In (14), it was further observed that dropping subsets of layers during training can reduce overfitting and improve generalization. In a related

study, (52) showed that introducing skip connections between layers can lead to parts of the network being effectively bypassed and under-trained. More recently, research has revealed substantial parameter redundancy in large-scale foundation models, particularly within their deeper layers (e.g., (13)).

All these observations can be unified under the perspective that, although residual architectures facilitate training via multiple signal pathways, these same pathways can sometimes act as shortcuts that cause certain components to be either underutilized or prone to overfitting—ultimately limiting effective generalization. Supporting this concern, (15) demonstrated that unscaled residual connections can degrade the quality of generative representation learning, offering a concrete case where standard (unregularized) residual connections negatively impact performance. Thus, an open question remains: can we design alternative architectures that retain the key benefits of Residual Networks—such as multiple signal pathways and efficient gradient flow—while mitigating drawbacks such as redundancy and shortcut overuse, effectively resulting in better representation learning?

In the next chapter, which serves also as the main contribution of this thesis, we take a step towards answering this question by proposing, analyzing, and empirically evaluating such an alternative architecture, showing promising improvements and distinct learning dynamics.

Chapter 5

Auto-Compressing Networks

5.1 Introduction

Despite the breadth of the research, presented in the previous chapter, exploring different connectivity patterns across domains—with goals ranging from improved expressivity and representation learning to increased training efficiency—none of these potential improvements have achieved broad adoption beyond standard residual connections. Furthermore, none of these works have explicitly investigated the parameter redundancy issue that potentially exists in these architectures based on evidence from discussed prior literature. In this work, we explore an architectural variant where additive long feedforward connections from each layer to the output replace traditional short residual connections as shown in Table 5.1 and Figure 5.1, introducing Auto-Compressing Networks (ACNs). ACNs showcase a unique property we coin as *auto-compression*—the ability of a network to organically compress information during training with gradient descent, through architectural design alone, dynamically pushing information to bottom layers, enhancing their representational quality, and naturally revealing redundant in deeper layers. We theoretically investigate the emergence of this property by analyzing the gradient dynamics of networks with different connectivity patterns. As illustrated in Figure 5.2 (top), ACNs (right) demonstrate layer-wise training patterns in which early layers receive significantly stronger gradients during the initial stages of training, in contrast to the more uniform gradient distribution observed in Residual Networks (left). In the following sections, we start by presenting a theoretical analysis of the gradient dynamics of ACNs, along with residual and feedforward networks, shedding light on their distinct behaviors and arguing that different connectivity patterns result in distinct learned representations. Next, we implement ACNs in modern architectures and we empirically demonstrate a broad range of advantages that ACN-learned representations offer compared to residual or feedforward architectures, including: enhanced information compression, superior generalization, reduced catastrophic forgetting, and efficient transferability.

5.2 The proposed Architecture

The core idea behind ACNs is to force each layer to produce features that are directly useful for prediction. In this manner, when the last layers are pruned, earlier layers can

be used for prediction directly without the need for further fine-tuning. As shown graphically in Figure 5.1, we form ACNs by starting from a generic feedforward architecture in (a) and adding long connections from each layer directly to the output and summing them up in (b). During training, information is naturally "pushed" towards the early layers ultimately allowing the removal of redundant top layers during inference without performance loss in (c). Concretely, we propose replacing the residual short connections with long connections, as described in Eq. 5.1 and shown in Table 5.1 for a network of depth L[1]:

$$x_i = f_i(x_{i-1}), \qquad y = \sum_{i=0}^{L} x_i \tag{5.1}$$

In ACNs the output of each layer [2] is directly connected to the output of the network, and thus is directly optimized by the objective function during gradient descent training. Furthermore, the number of possible shortcuts is equal to the number of layers L. We find this simplified structure maintains the improved signal flow that shortcut connections provide, while also introducing the ability to detect potential parameter redundancy in the architecture[3].

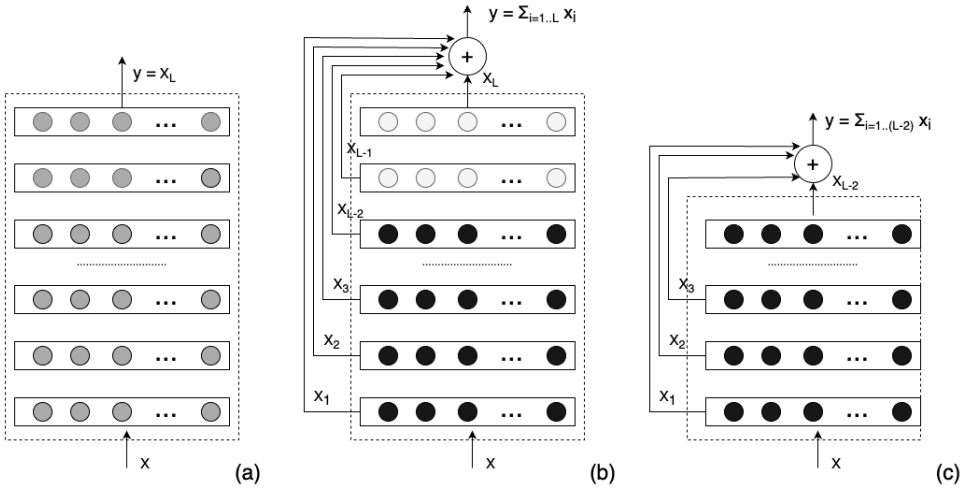


Figure 5.1. *Main concept: (a) Start from a neural network with L layers. (b) Add residual long connections from each layer to the output of the network and sum them (also remove any existing short residual connections - if any). During training of the resulting ACN network the majority of the information (shown here as darker vs lighter circles) will naturally concentrate at the lower layers. (c) You may now safely remove the top (two in our example) layers during inference without any performance loss.*

5.2.1 Gradient Propagation Across Network Architectures

To understand how different neural network architectures behave during training, we analyze, in this section, their gradient flow characteristics. We examine and compare

[1]We note that a classification head can be built on top of y.
[2]Also the embedded input, represented with x_0 in equation 5.1.
[3]We note that long connections are a strict subset of the 2^L shortcut connections in residual networks.

the forward and backward pass dynamics of three architectures: traditional feedforward networks (FFN), residual networks (ResNet), and the proposed auto-compressing networks (ACN). For clarity and mathematical tractability, we consider linear neural networks of depth L and derive the equations for the 1-dimensional case (they can be expanded to N-dimensional inputs).

Notation: x_i is the output of layer i, w_i is the weight of layer i (the weight used to construct x_i), x_0 is the input (after a potential initial embedding operation) and y_F, y_R, y_A is the output for each architecture.

Figure 5.2. ***(top)*** *ACNs vs Residual Networks gradient flow across layers during training for MLP-Mixer architecture (4) on CIFAR-10 (5), showcasing implicit layer-wise training and information concentration on the bottom layers for ACNs. On the other hand, Residual Networks show higher gradient norms (information concentration) in early and deep layers, while middle layers receive significantly lower gradients (suggesting potential parameter redundancy).* ***(bottom)*** *ACNs vs Residual Networks incremental performance contribution across layers during training for ViT architecture (6) on ImageNet-1K, revealing auto-compression by gradual layer-wise training in ACNs (task-learning starts from shallow layers and gets "pushed" to deeper layers to maximize performance). In Residual Networks, as shown in the Figure task learning happens in the 2-3 final layers.*

FFN forward pass

$$y_F(= x_L) = \prod_{i=1}^{L} w_i x_0 \tag{5.2}$$

FFN backward pass for weight i

$$\frac{\partial y_F}{\partial w_i} = \frac{\partial y_F}{\partial x_i}\frac{\partial x_i}{\partial w_i} \tag{5.3}$$

$$\frac{\partial y_F}{\partial w_i} = \underbrace{\left(\prod_{k=i+1}^{L} w_k\right)}_{\textit{backward} \text{ term}} \underbrace{\left(\prod_{m=1}^{i-1} w_m\right)}_{\textit{forward} \text{ term}} x_0 \tag{5.4}$$

ResNet forward pass

$$x_i = w_i x_{i-1} + x_{i-1} = (1 + w_i)x_{i-1} \tag{5.5}$$

$$y_R = \prod_{i=1}^{L}(1 + w_i)x_0 \tag{5.6}$$

ResNet backward pass for weight i

$$\frac{\partial y_R}{\partial w_i} = \frac{\partial y_R}{\partial x_i}\frac{\partial x_i}{\partial w_i} = \left(\prod_{k=i+1}^{L}(1 + w_k)\right)\left(\prod_{m=1}^{i-1}(1 + w_m)\right)x_0 \tag{5.7}$$

$$\frac{\partial y_R}{\partial w_i} = \underbrace{\left(1 + \sum_{k=i+1}^{L} w_k + \sum_{i+1\le k<j\le L} w_k w_j + \cdots + \prod_{k=i+1}^{L} w_k\right)}_{\textit{backward} \text{ term}} \underbrace{\left(\prod_{m=1}^{i-1}(1 + w_m)\right)}_{\textit{forward} \text{ term}} x_0 \tag{5.8}$$

or equivalently:

$$\frac{\partial y_R}{\partial w_i} = \underbrace{\left(1 + \sum_{k=1}^{L-i+1} \texttt{sum of } \binom{L-i+1}{k} \texttt{ w k-tuples}\right)}_{\textit{backward} \text{ term}} \underbrace{\left(\prod_{m=1}^{i-1}(1 + w_m)\right)}_{\textit{forward} \text{ term}} x_0 \tag{5.9}$$

ACN forward pass

$$y_A = x_0 + \sum_{i=1}^{L} x_i = x_0 + \sum_{i=1}^{L}\prod_{j=1}^{i} w_j x_0 \tag{5.10}$$

ACN backward pass for weight i

$$\frac{\partial y_A}{\partial w_i} = \frac{\partial y_A}{\partial x_i}\frac{\partial x_i}{\partial w_i} = \left(1 + \sum_{k=i+1}^{L} \frac{\partial x_k}{\partial x_i}\right) x_{i-1} \tag{5.11}$$

$$\frac{\partial y_A}{\partial w_i} = \underbrace{\left(1 + \sum_{j=i+1}^{L} \prod_{k=i+1}^{j} w_k\right)}_{\textit{backward}\text{ term}} \underbrace{\left(\prod_{m=1}^{i-1} w_m\right)}_{\textit{forward}\text{ term}} x_0 \tag{5.12}$$

5.2.2 Emergent Gradient Paths

The *forward* and *backward* components: As seen in the previous equations, the gradient of w_i can be generally decomposed in two terms, coined as the *forward* and the *backward* terms.

- The *forward* term consists of the forward propagated signal up to layer i, essentially the input embedding transformed as it traverses the layers up to i, and it primarily governs the stability of the gradient flow (whether the signal vanishes or explodes). In ACNs and FFNs this term is the same, consists of one path, and its norm typically decreases in practice as the network depth increases, due to multiplications and non-linearities (assuming close to zero initialization). In ResNets, the number of paths included in the component grows as more layers are traversed, since each layer adds its output to the the residual stream.

- The *backward* term carries the information relevant to learning (coming from the loss and traversing subsequent layers) and thus, its structure influences the behavior and characteristics of the representations learned by the network during training with backpropagation. In 1D FFNs, the backward term contains a single path from the last layer L to the layer i currently being trained. However, for 1D ACNs and 1D ResNets, both *multi-path* archiitectures, the backward term consists of $L - i + 1$ and 2^{L-i} paths, respectively. It is worth mentioning that the FFN path is a subset of the ACN paths, which in turn are contained in the ResNet paths, so, for the set of paths B of the backward term one may write:

  $$B_{FFN} \subset B_{ACN} \subset B_{ResNet}.$$

  Thus, in this 1D case, the transition from FFNs to ACNs, and subsequently to ResNets, reflects a progression in the number of effective information paths: from a single path in FFNs, to a number of paths linear in the number of layers in ACNs, and finally to an exponential number of paths in ResNets. We will use the notation **FG** when referring to the full gradient including all backward paths (the full backward signal of a layer).

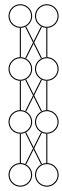
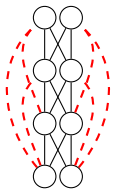
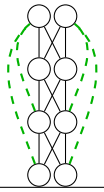
Arch	Connectivity	Forward Propagation	Backward (Gradient) Propagation
FFN		$y_F = \prod_{i=1}^{L} w_i x_0$	$\frac{\partial y_F}{\partial w_i} = \underbrace{\left(\prod_{k=i+1}^{L} w_k\right)}_{\textit{backward}\ \text{term}} \underbrace{\left(\prod_{m=1}^{i-1} w_m\right)}_{\textit{forward}\ \text{term}} x_0$
ResNet		$y_R = \prod_{i=1}^{L}(1 + w_i)x_0$	$\frac{\partial y_R}{\partial w_i} = \underbrace{\left(\prod_{k=i+1}^{L}(1 + w_k)\right)}_{\textit{backward}\ \text{term}} \underbrace{\left(\prod_{m=1}^{i-1}(1 + w_m)\right)}_{\textit{forward}\ \text{term}} x_0$
ACN		$y_A = \left(1 + \sum_{i=1}^{L}\prod_{j=1}^{i} w_j\right) x_0$	$\frac{\partial y_A}{\partial w_i} = \underbrace{\left(1 + \sum_{j=i+1}^{L}\prod_{k=i+1}^{j} w_k\right)}_{\textit{backward}\ \text{term}} \underbrace{\left(\prod_{m=1}^{i-1} w_m\right)}_{\textit{forward}\ \text{term}} x_0$

Table 5.1. *Connectivity (2D case), Forward and Backward Propagation (1D linear case) for FFN, ResNet, and ACN architectures.*

Decomposition of the *backward* term: The *backward* component can be further decomposed into two distinct terms: one term that captures the propagation of gradients through the network layers, and another term that corresponds to a direct gradient path from the output to each layer.

- **The gradient paths through the network:** In all architectural variants, there is a dominant component (the majority of paths) that backpropagates information using the network weights, traversing subsequent subnetworks (subsets of layers). This component carries the information on how the weights of a given layer should be adjusted in relation to the following layers, enabling the entire network to reduce the error. In the above equations, it corresponds to the single backward path in Eq. 5.4 for FFNs, while for ACNs and ResNets it contains all the paths except for the direct path "1+" in the backward term of Eq. 5.12 and Eq. 5.8, respectively. We will refer to this term as the network-mediated gradient or **NG**. We note that the norm of this component decreases in ResNets and ACNs as we move to deeper layers, since the number of paths diminishes.

- **The gradient path directly from the output:** Interestingly, in both ACNs and ResNets, there is also a path that comes **directly from the output**, the "1+" component in the backward terms. This path exists because, in these architectures, the output of each layer is directly connected to the final output of the network through the skip connections. This backward path has been previously explored as an alternative to traditional backpropagation, with a significant body of work dedicated to it (53; 54). This line of research is referred to as Direct Feedback Alignment (DFA), and thus we use the term direct gradient or **DG** to describe this mechanism

throughout this work. This path carries information on how a layer's weights must change so that the output of this layer directly lowers the error. Finally, the **DG** path's contribution is more significant in ACNs compared to ResNets due to ACNs' linear (rather than exponential) total path count, something that becomes more apparent in deep networks and especially for the training of early layers, i.e., for large $L - i$. For example, when training the second layer of a $L = 12$ layer network, **DG** is one of 11 ACN backward paths, while for ResNets the **DG** is competing with another 127 paths (of the **NG** term). This further accelerates training of the early layers.

In summary, we can write:

$$\mathbf{FG} = \mathbf{DG} + \mathbf{NG}.$$

5.2.3 Implicit Layer-Wise Training in ACNs

Unlike the symmetric forward and backward terms of ResNets and FFNs, ACNs feature a forward term identical to FFNs for intermediate layers (single path) and a backward term more similar to ResNets (multiple paths that grow linearly vs exponentially with depth). This design creates an implicit layer-wise training dynamic, where deeper layers are trained at a slower rate compared to earlier layers, since they have a **weaker forward component** (assuming close-to-zero initialization) and a **smaller number of backward paths**.

Main Claim: Enhancing the performance of intermediate layer predictions (through a strong DGcomponent) coupled with the implicit layer-wise training, as discussed above, can lead to robust progressive training of the network in a layer-by-layer manner, while also potentially identifying parameter—whole-layer—redundancy within the architecture, effectively compressing compressing information into a sufficient subset of layers during training. The idea is that when training a randomly initialized neural network of depth L, if the first k layers (that are trained faster) are able to solve the task (minimize the error), then the network will utilize only them, leaving the remaining deeper layers untrained and effectively leading to information compression. Thus, we postulate that: 1) a strong **DG** component coupled with a weaker feed-forward signal leads implicitly to efficient layer-wise training, and 2) inadvertently, architecturally-induced layer-wise training results in a form of **structural learning** where information is naturally pushed to early layers and later layers become redundant (effectively identity mappings). We refer to this new class of networks as **auto-compressors** since they naturally shed their redundant layers during backpropagation simply via architectural design.

5.2.4 A Note on Addressing some of Residual Network Limitations through ACNs

From the analysis conducted on our proposed architecture thus far, it appears that we are taking meaningful steps toward addressing several of the potential issues of Residual Networks, raised in the final section of the previous chapter. Specifically: (1) the induced training dynamics in ACNs promote a form of layer-wise training, where each layer is

robustly optimized only if necessary. This behavior serves as a mechanism for identifying and managing parameter redundancy in the initialized network, and may lead to improved representation learning by ensuring that layers are trained meaningfully when required; and (2) ACNs retain the multi-path nature characteristic of Residual Networks, thereby preserving the benefits of having multiple information pathways for both forward and backward signal propagation—an essential property for effective training in deep neural architectures.

We now turn our attention to empirically validating these theoretical properties. Specifically, we aim to investigate how ACNs behave in practice, evaluate their ability to mitigate parameter redundancy, and assess their capacity to facilitate improved representation learning.

5.2.5 A Toy Demonstration

Following the theoretical analysis of gradient dynamics, we first validate our key claims through a simple demonstration that illustrates the core compression mechanism. Specifically, to demonstrate how ACNs naturally compress information into early layers during training, we perform a simple toy experiment involving a 1D linear feedforward network with three layers and weights w_1, w_2 and w_3 for each layer, respectively. The dataset comprises pairs drawn from the function $y = 2x$. We consider two architectures: the first employs residual (short) connections, while the second utilizes long connections (ACN). For this toy problem, both the residual and ACN networks should ideally require only a single layer to successfully accomplish the task, i.e., $w_1 = 1$, $w_2 = 0$, $w_3 = 0$ is the most efficient solution:

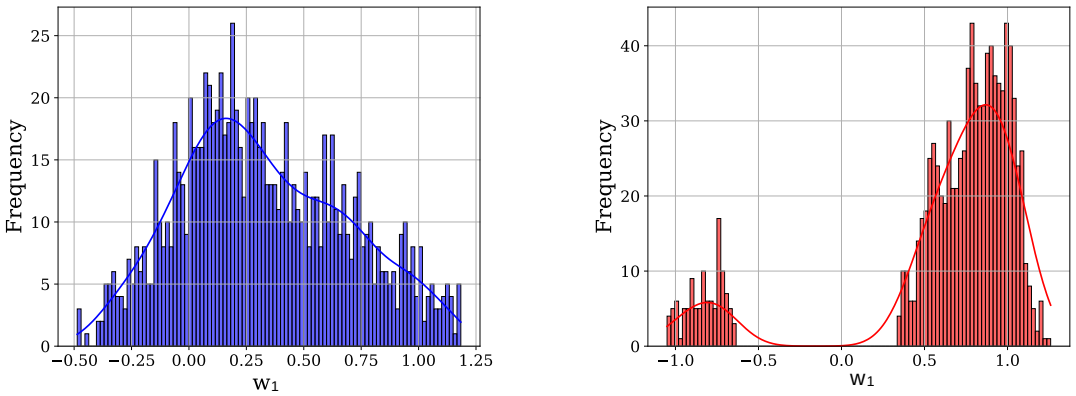
$$\text{ResNet:} \quad \hat{y_R} = (1 + w_1)(1 + w_2)(1 + w_3)x = 2x, \tag{5.13}$$

$$\text{ACN:} \quad \hat{y_A} = (1 + w_1 + w_1w_2 + w_1w_2w_3)x = 2x. \tag{5.14}$$

We perform this simple training experiment multiple times ($N = 1000$), each time generating 1000 examples of input x with values between -10 and 10. Both models are trained for 300 epochs and the weights are initalized with values around zero either uniformly in $[-1, 1]$ or following a normal distribution. Fig. 5.3 illustrates the distribution of learned w_1 values for both architectures.

For residual architectures in (a) we observe a pretty wide distribution of w_1 values centered around 0.26; indeed $w_1 = w_2 = w_3 \approx 0.26$ is a valid solution of Eq. 5.13 . Thus, in this example, *residual networks have the tendency to utilize all layers equally*, even if a sparse solution exists. ACNs, however, typically converge to solutions where w_1 is close to 1, allowing correct predictions from the first layer[4]. So in this simple example, *long connections in ACNs induce implicit depth regularization*, guiding the network toward sparse solutions.

[4]Note that initialization plays a crucial role, as w_1 sometimes converges near -1 for ACNs. Further, the solution that ACN converges at is $w_1 \approx 0.9$, $w_2 \approx 0.11$, $w_3 \approx 0$, so it just gets pretty close to the most efficient solution but it does not achieve perfect compression.



(a) *Residual Network* **(b)** *Auto-Compressing Network*

Figure 5.3. *Histogram (1000 runs) of the weight of the first layer w_1 of a 1D linear feedforward residual network with three layers solving $y = 2x$, utilizing either: (a) residual connections or (b) long connections (ACN).*

The careful reader will observe that this behavior arises from the weight asymmetry in Eq. 5.14, where there are three terms that include w_1, two terms with w_2 and a single term with w_3. In terms of derivative flow, the **DG** component for each layer is pretty strong compared to **NG**. Compare this with the weight symmetric Eq. 5.13, where the **NG** component dominates. The residual network naturally converges to the $w_1 = w_2 = w_3$ symmetric solution[5].

5.3 Experiments

In this section, we move from theoretical analysis and simple demonstrations to implementing auto-compressing networks in modern deep learning architectures. We apply our approach to state-of-the-art neural network models across diverse tasks and datasets, demonstrating that the information compression effect observed in our theoretical analysis manifests consistently in practice. Our experimental validation spans multiple domains and model architectures. We implement ACNs using variants of the Transformer (29) for language and vision tasks and MLP-Mixer (4) for vision tasks. This allows us to evaluate our approach on diverse benchmarks including image classification (CIFAR-10, ImageNet-1K), sentiment analysis, and language understanding.

Experimental Setup: For ACNs, for each input token, we compute a final output vector y^t (where t is the sequence index) by summing the output representations of all intermediate layers along with the input embedding, as shown in Eq. 5.1. To generate classification predictions, we either apply a pooling layer to these vectors for image classification or use the final representation of the `[class]` (`[CLS]`) token for text classification. For a network of

[5]Another way to interpret Eq. 5.14 is that we have superimposed four feedforward networks: the identity network, a single layer, two layer and three layer network. The important tweak here is that their weights are tied, e.g., w_1 is common for all network depths, which biases the network towards a shallow solution.

depth L, making predictions using k intermediate layers involves computing y_k^t for each token, which is the sum of intermediate representations **up to layer** k. This summed representation is then passed to a single global classification head, which is trained once and shared across all sub-networks (we do not retrain or create separate classification heads for each depth configuration). This approach yields $L + 1$ sub-networks, ranging from using only the input embedding (the first sub-network) to the full network (the last sub-network). For example, the network shown in Figure 5.1(c) would be the $L - 2$ sub-network (utilizing layers 1 to $L - 2$), whereas the network in in Figure 5.1(b) would be the full network (all layers included). When evaluating residual network baselines, we follow standard practice: to assess the network at depth k, we simply take the output y_k^t of the kth layer as our representation. This provides a natural comparison point to ACNs at equivalent depths. All other procedures remain the same. In all figures, prediction layer 0 refers to the input embedding passed through the classification head for prediction.

5.3.1 Auto-Compression via Direct Gradient Flow

Here, we validate the main claim established in the previous section; that the presence of a strong **DG** component coupled with implicit layer-wise training dynamics drives auto-compression .

We train feedfoward (FFN), residual and auto-compressing variants incorporated in the MLP-Mixer architecture (4) on CIFAR-10 dataset (5) for 100 epochs. To emphasize the role of the DG gradient in auto-compression, we also train an ACN variant receiving gradients only from the long connections (only **DG** component). This experiment allows us to isolate the effects of different gradient components and connectivity patterns on information compression and overall task performance. From Figure 5.4(left), we observe that among ACNs, FFNs, and Residual Networks, only ACNs exhibit auto-compression. Moreover, ACNs utilizing only the direct gradient (**DG**) still achieve significant auto-compression, highlighting the importance of a strong **DG** component to achieve this behavior[6] and explaining why FFNs do not exhibit auto-compression, as they lack a direct gradient term (Equation 5.4). In the case of Residual Networks, we previously argued that the exponential number of gradient paths substantially diminishes the influence of the direct gradient (**DG**) on the overall gradient, one of the key components crucial for auto-compression. To further illustrate this, Figure 5.4(right) presents the ratio of **DG** to the full gradient **FG** across layers during training for both AC and Residual variants. The results indicate a significantly higher **DG** to **FG** ratio in ACNs, confirming the increased contribution of direct gradients in the early layers of auto-compressing architectures compared to residual networks and explaining the auto-compression property. Furthermore, from Figure 5.2(top) we observe that ACNs (right) demonstrate a concentrated gradient pattern with stronger signals in early layers and stronger patterns of **layer-wise learning**. Residual Networks (left) exhibit a more "uniform layer learning" pattern, whereas deeper layers show increasing gradient contribution in later epochs, suggesting task-specific adaptation as training

[6]ACNs with only the **DG** component under-perform, underpinning the importance of the **NG** component for maximizing performance.

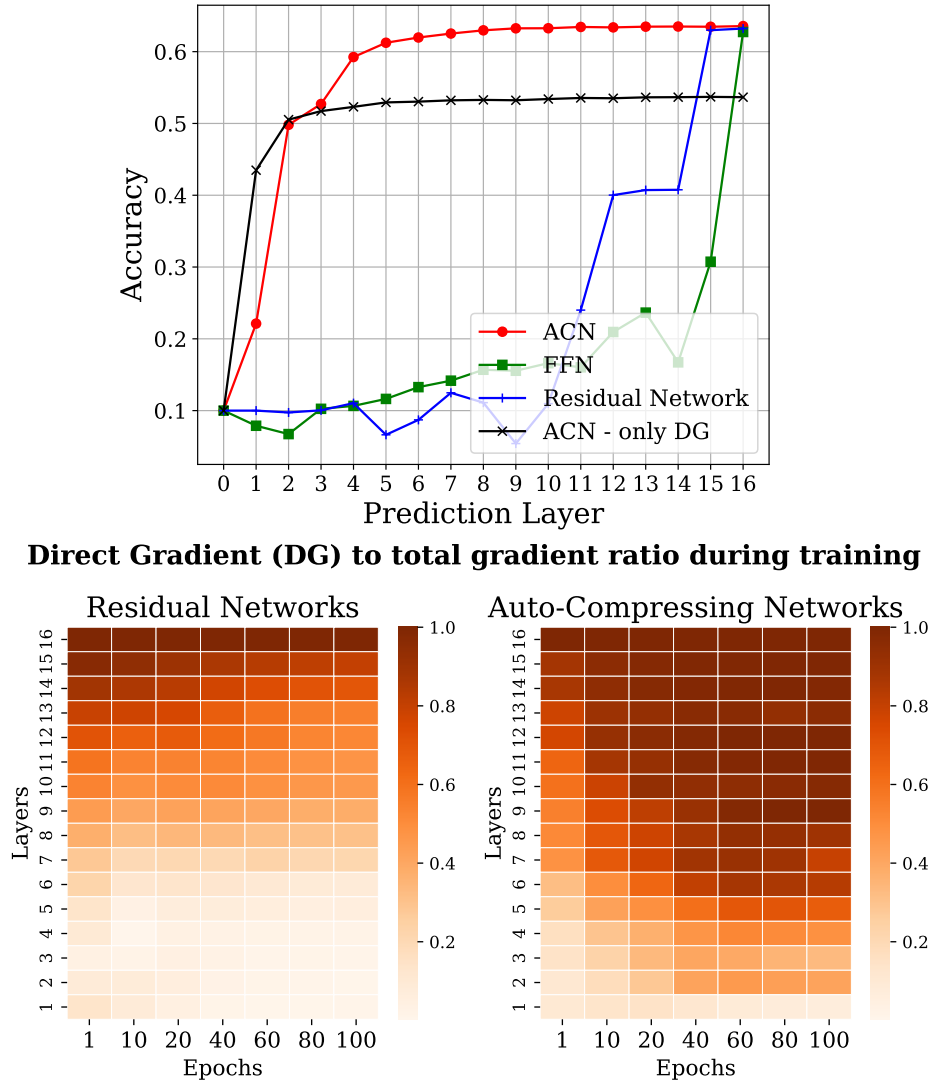


Figure 5.4. ***(top)*** *ACNs are the only architectural variant that achieves auto-compression.* ***(bottom)*** *The ratio of direct gradient* ***DG*** *to the total gradient* ***FG*** *in auto-compressing vs residual architectures. The exponential (vs linear) number of paths in Residual Networks decreases the influence of* ***DG*** *in the training dynamics of the network compared to Auto-Compressing Networks.*

progresses. Both architectures display distinct first-epoch behavior, with Residual Networks showing more uniformity immediately, while Auto-compressing Networks establish their hierarchical gradient pattern from the start. Interestingly, the pattern observed in Residual Networks indicates that high gradient norms are primarily concentrated in the early and deep layers, while middle layers receive significantly lower gradients, suggesting potential redundancy. Overall, this gradient pattern reveals that layer-wise training dynamics, the other key factor for auto-compression, is absent in ResNets.

In summary, the auto-compressing architecture featuring a strong **DG** component along with implicit layer-wise training dynamics pushes information to the bottom layers of the network, achieving layer-wise structural learning (auto-compression) while main-

taining strong performance.

5.3.2 Auto-Compressing Vision Transformers

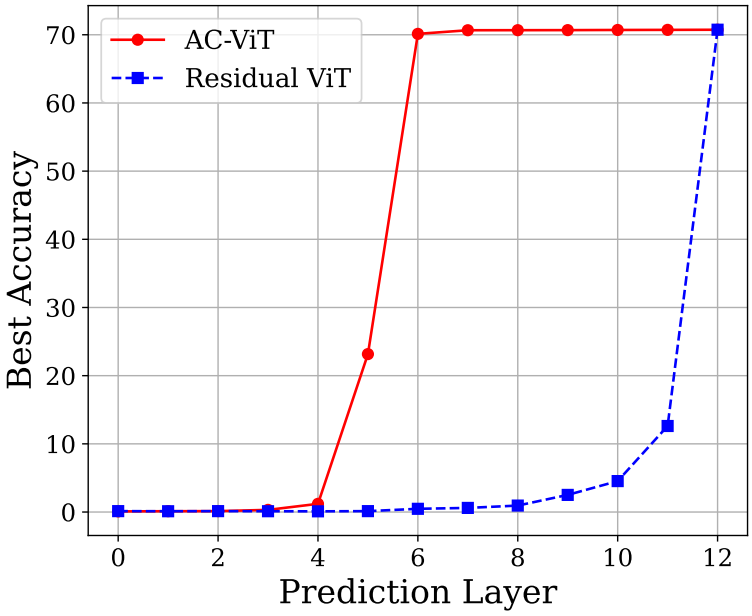


Figure 5.5. *Performance of intermediate layers of AC vs Residual Vision Transformers trained on ImageNet-1K.*

Next, we evaluate ACNs in the context of transformer architectures by implementing an auto-compressing variant of Vision Transformer (ViT) (6). We train a Vision Transformer (ViT) with long connections (AC-ViT) from scratch on the ILSVRC-2012 ImageNet-1K, following the training setup in the original paper. For both models we use 256 batch size due to memory constraints. AC-ViT converges at 700 epochs, while the Residual ViT converges at 300 epochs[7].

As shown in Fig. 5.5, AC-ViT reaches top performance at only 6 layers while the vanilla ViT needs all 12 layers to reach similar performance, effectively suggesting that *ACNs can improve inference time and memory consumption without sacrificing performance*. To gain more intuition about the training dynamics and task learning of the two variants, in Figure 5.2(right) we plot the incremental layer performance contribution (difference in accuracy of subnetwork $i+1$ to subnetwork i) to track the behavior of intermediate layers throughout training. The key observation is that ACNs (right) are trained in a layer-wise fashion where early layers are trained at a faster rate and task-relevant information is gradually pushed only to a subset of the deeper layers, achieving strong performance along with auto-compression. In the contrary, the Residual variant performs task-learning in the last 2-3 layers, effectively utilizing the full network to achieve top performance.

[7]In this more challenging setting, we observe a trade-off between training and inference time, which is partially aleviated using a parameterization similar to DiracNets (92) for the MLP layers, specifically $\hat{W} = (I + W)$.

We also conduct a transfer learning experiment on CIFAR-10 dataset, to test the behavior of the auto-compressing architecture compared to residual network, when transfering to a donstream task. As we observe from Figure 5.6, AC-ViT is on-par with the residual variant, showing no transfer learning degradation while maintaining the 50% layer compression rate.

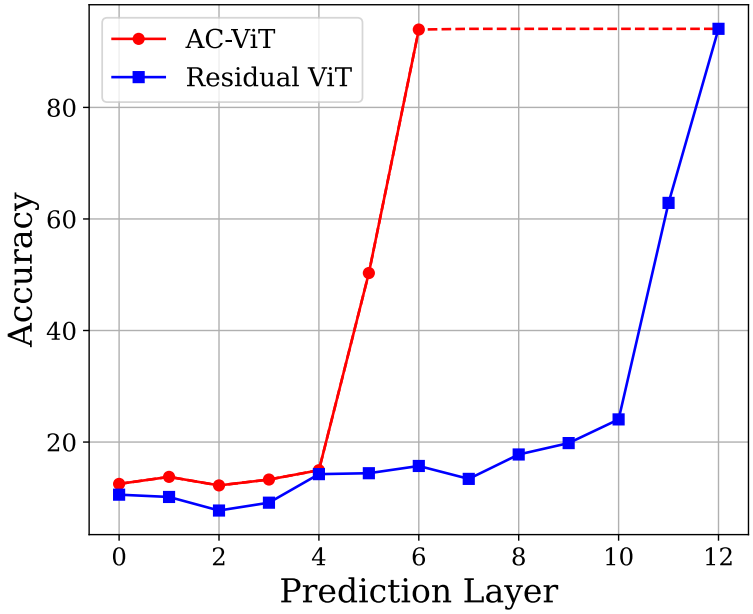
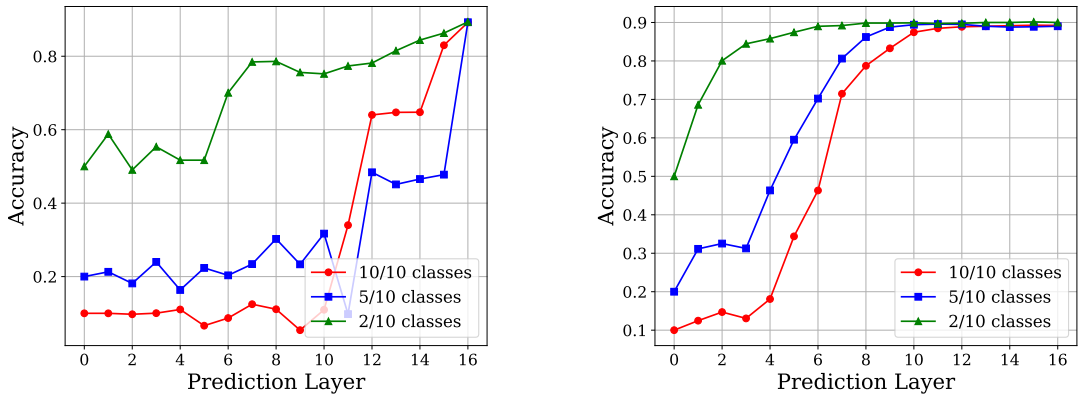


Figure 5.6. *Fine-Tuning the previously trained ViTs on CIFAR-10. The auto-compressing architecture transfers to downstream tasks as effectively as the residual counterpart, despite utilizing half the number of layers.*

5.3.3 The Effect of Task Difficulty

Intuitively, overparameterized networks trained on easier tasks should demonstrate higher levels of redundancy. Therefore, ACNs should converge to utilizing fewer layers as task difficulty decreases. To verify this, we use the number of classes as a proxy for task difficulty for image classification on the CIFAR-10 dataset (5). Specifically, we create subsets of 2, 5, and 10 classes, the assumption being that binary classification should be easier than 10-class classification. For this experiment we utilize MLP-Mixer (4) and train two variants, the original MLP-Mixer with residual connections and the modified MLP-Mixer with long connections (AC-Mixer). The MLP Mixers have 16 layers with a hidden size of 128. The patch size is 4 (the input is 32x32, 3 channels). The MLP dimension D_C is 512, while D_S is 64. We are using the AdamW optimizer (93) with a maximum learning rate of 0.001 and a Cosine Scheduler with Warmup. The batch size is 64. Results are presented in Fig 5.7. We observe that indeed *AC-Mixer converges to solutions with larger effective depth, as the task "difficulty" increases.* Specifically, in this experiment, ACN needs 8, 10 and 12 layers for the 2, 5 and 10-class classification problem, respectively. In contrast, the Residual Mixer converges to solutions where the full depth of the network

is utilized, irrespective of the task difficulty [8].



(a) *Residual Mixer* **(b)** *AC-Mixer*

Figure 5.7. *Performance of the intermediate layers as the number of classes (and examples) in the CIFAR-10 dataset increases from 2, to 5 to 10 classes: (a) Residual Mixer vs (b) AC-mixer.*

5.3.4 Generalization Capabilities of Auto-Compressors

While ACNs demonstrate effective parameter reduction through architectural compression, a key question remains: do these compressed representations offer additional benefits beyond parameter efficiency? In this section, we investigate whether the concentrated information in ACNs' early layers leads to improved generalization capabilities compared to traditional residual architectures. This point is crucial, as one could argue that compression alone can be achieved through external techniques such as pruning or knowledge distillation. However, this is not entirely accurate, as such external methods typically introduce additional computational costs and require careful hyperparameter tuning on top of the standard training process, compared to ACNs that exhibit natural auto-compression during training. Nevertheless, beyond this implicit compression, further demonstrating that the proposed architecture not only mitigates redundancy but also enhances representation learning and generalization significantly strengthens its overall contribution. To investigate this, we explore two critical aspects of generalization: robustness to input noise and performance in low-data regimes.

Robustness to Input Noise

Robustness to Data Sparsity Next, we experimentally compare the performance of residual and long connections architectures in low-data scenarios. For this purpose, we create a random subset of CIFAR-10 (5) by retaining only 100 samples per class, resulting in a total of 1000 examples. Using the same training settings and models as

[8]The Residual Mixer was trained for 300 epochs, while AC-Mixer for 420 epochs to reach the performance of its residual counterpart.

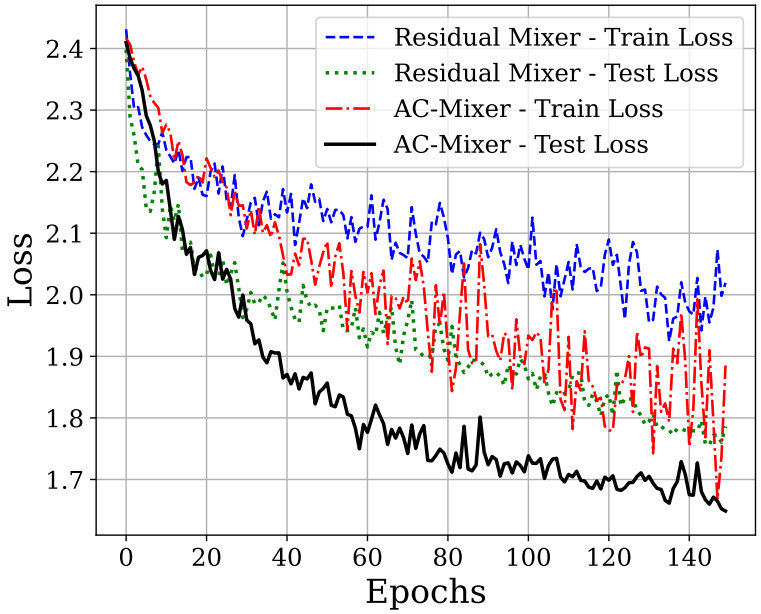


Figure 5.8. *Train and Test Loss of AC-Mixer and Residual Mixer on CIFAR-10* ***with limited data*** *(100 samples per class).*

Model	Baseline	Gaussian Noise			Salt and Pepper Noise		
	w/o noise	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.4$	$p = 0.01$	$p = 0.05$	$p = 0.1$
Residual ViT	70.74	67.68	62.80	45.46	56.80	27.48	10.34
AC-ViT	70.76	69.50	64.54	51.89	59.80	36.35	19.98

Table 5.2. *Robustness (average accuracy %) of ViT with long connections (AC-ViT) and with residual connections (Residual ViT) to additive Gaussian noise and salt-and-pepper noise on ImageNet-1K test set.*

described in Section 5.3.3, we train both architectures for 150 epochs to assess how fast the training and test loss decrease, as a proxy for the generalization capabilities of each architecture. Results shown in Fig. 5.8 reveal that ACNs achieve lower training and test loss in fewer epochs compared to residual networks. This faster convergence in loss metrics is a strong indication that auto-compressing networks can be effectively utilized in scenarios with limited data.

5.3.5 Auto-Compressing Encoder Architectures for Language Modeling

As previously discussed, recent studies have demonstrated that foundation models suffer from parameter redundancy, especially in their deeper layers (eg. (13)). This characteristic is crucial today, in the context of large language and multimodal models, which are typically pre-trained as general-purpose models before being adapted to specific downstream tasks. Since these specialized applications may not require the full parameter capacity of the base model, learned representations (through architectural choices) that facilitate subsequent compression and pruning become crucial. In this section, we conduct a preliminary study on the effectiveness of the ACN architecture in general pre-

training (masked language modeling with a BERT architecture) followed by fine-tuning and pruning. The results show that ACNs learn compact representations that: 1) achieve on-par performance with the residual architecture on transfer learning tasks, while utilizing significantly fewer parameters, and 2) complement post-training pruning techniques, enhancing their effectiveness.

Masked Language Modeling and Transfer Learning with ACNs Next, we compare the ACN and residual architectures in the standard BERT pre-training and fine-tuning paradigm. Using the original BERT pretraining corpus (BooksCorpus (55) and English Wikipedia), we train both architectures to equivalent loss values; the AC-BERT variant requires two epochs vs one epoch for the residual baseline. Following pre-training, we fine-tune both models on three GLUE benchmark datasets (56): SST-2 sentiment analysis (57), QQP paraphrasing, and QNLI question answering (58).

Figure 5.9 (top) demonstrates a key advantage of the ACN architecture: it naturally converges to using significantly fewer layers (approximately 75% less layers) while maintaining performance comparable to the full residual network. These results suggest promising applications for ACNs in large language models, where pre-training could be performed with long connections, allowing downstream tasks to adaptively utilize only the necessary subset of layers during fine-tuning.

Post-Training Pruning with AC-Encoders ACN's primary advantage lies in its inherent compression capabilities during training, suggesting that when combined with pruning techniques, it should significantly outperform traditional residual architectures. To provide validation for this hypothesis, we conducted experiments using magnitude and movement pruning (59)[9], two commonly employed baseline pruning techniques. For Magnitude pruning, we consider the setting where the pruning happens after fine-tuning on the downstream task. For Movement pruning, we follow a gradual fine-tune and prune curriculum, where in setting (I): 20% of the parameters are pruned after each epoch, whereas in setting (II): we prune 40% of the parameters after an epoch. Results are shown when fine-tuning of the SST-2 dataset sentiment analysis task. Figure 5.9 (bottom) confirms our hypothesis: ACNs consistently demonstrate superior compression-performance trade-offs compared to standard architectures, with their advantage becoming more pronounced at higher compression rates. This indicates that ACNs' architectural design naturally leads to more efficient parameter utilization, creating representations that are inherently more amenable to further pruning. While these preliminary results validate our approach to addressing parameter redundancy, they also point toward promising future directions. We anticipate that combining pre-trained ACN architectures with state-of-the-art pruning methods will result in extremely efficient, high-performing models, though rigorous validation of this hypothesis requires further investigation.

[9]Movement pruning differs from magnitude pruning in that it evaluates the importance of weights based on their cumulative parameter updates during training, rather than solely on their absolute values. This approach enables more informed removal of less critical weights, potentially preserving model performance better while achieving sparsity.

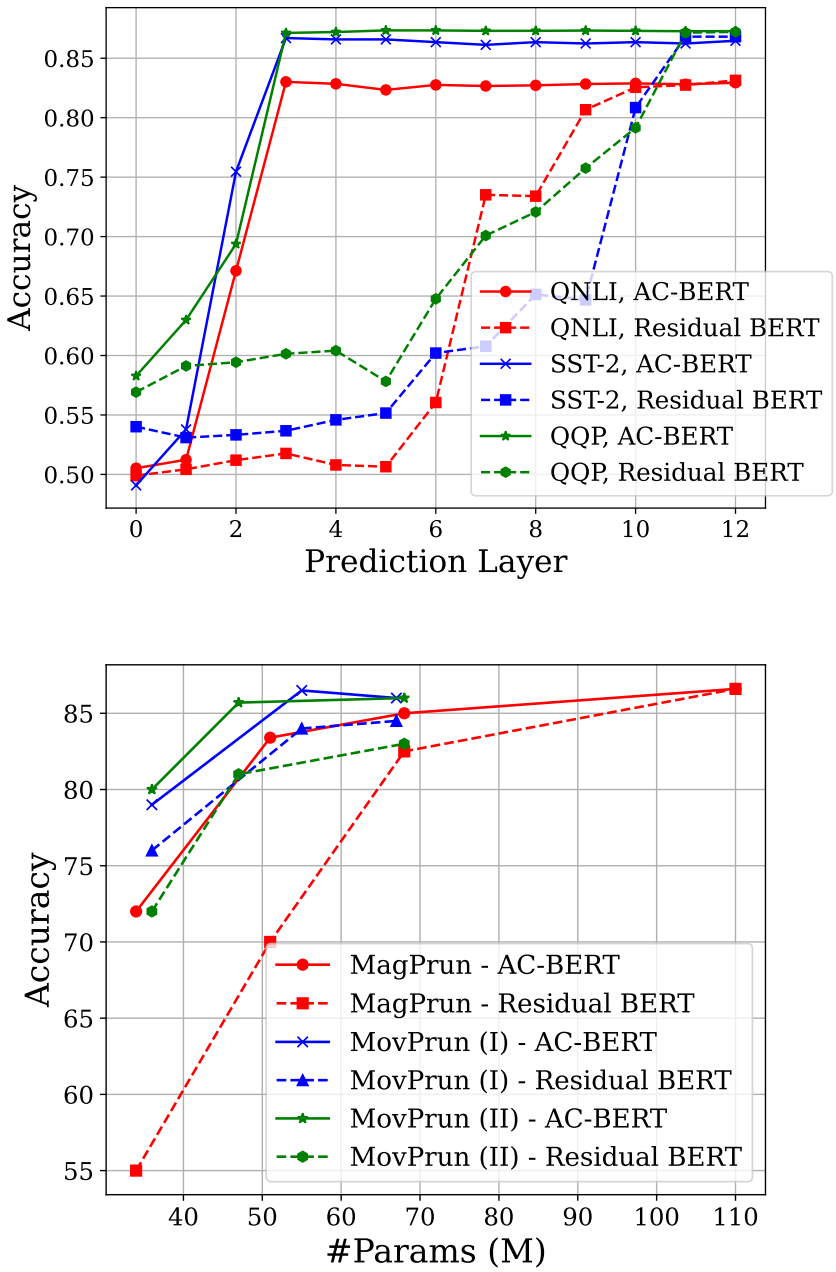


Figure 5.9. ***(top)*** *Downstream performance of AC-BERT vs residual BERT on three GLUE tasks: sentiment analysis (SST-2), paraphrasing (QQP), and question answering (QNLI).* ***(bottom)*** *Accuracy vs model size of AC-BERT and Residual BERT on SST-2 when pruned with Magnitude and Movement Pruning.*

5.3.6 Auto-Compressing Architectures vs. Layer-wise Loss Regularization

Parameter redundancy, and specifically potential layer redundancy, in residual architectures is a phenomenon that has been well documented(52; 3; 14), ar argued. Recent works (8; 7) have attempted to address this through regularization-based layer-wise structural learning approaches during training, specifically by adding losses to all intermediate layers of the network and using a weighted sum of them as the total loss, a technique

formally introduced in (9) for improved training. Such loss-based regularization methods rely heavily on precise tuning of intermediate loss weights, creating practical challenges. If early-layer loss weights are set too high, the network risks overfitting and poor generalization; if set too low, performance improves gradually across layers with no clear cutoff point, reaching optimal results only at the final layer. This sensitivity to hyperparameter selection makes it difficult to reliably identify an optimal depth for inference using loss-based regularization. ACNs address this challenge through architectural design rather than regularization, naturally compressing information without requiring complex hyperparameter tuning.

ACNs achieve better generalization To evaluate hyperparameter sensitivity in regularization-based approaches, we compare several methods on the CIFAR-10 dataset using MLP-Mixer architectures. Our comparison includes: 1) our proposed AC-Mixer, 2) an unregularized Residual Mixer as baseline, 3) a Residual Mixer with the setup of (7) (Aligned), 4) a Residual Mixer with the setup of (8) (LayerSkip), with the rotational early exit curriculum with $p_{max} = 0.1$, $e_{scale} = 0.2$ and $C_{rot,R} = 15$, and 5) a Residual Mixer with a baseline vanilla deep supervision (9) where all intermediate losses before the final layer are weighted with $\lambda = 0.1$ (DeepSup). This comparative analysis reveals how different approaches respond to their respective hyperparameter configurations. We follow the training pipeline as described in the previous section and track the performance of all intermediate layers. Our experiments (Figure 5.10) demonstrate that while regularization approaches are highly sensitive to intermediate loss weights, creating a trade-off between performance and compression, ACNs consistently achieve strong results through their inherent architectural properties. Specifically, ACNs match the performance of unregularized Residual Networks, while effectively determining a shallower cutoff layer. While careful tuning of regularization methods can potentially match ACN's performance, our approach provides a more elegant and robust solution that requires no parameter adjustment while maintaining high performance and achieving sparsity.

ACNs show stronger Transfer Learning capabilities To evaluate whether different layer compression approaches learn generalizable representations, we conduct a transfer learning experiment from CIFAR-100 to CIFAR-10 (5). This setup allows us to assess how well each model's learned representations transfer to a similar task. In regularization-based layer compression methods, explicitly training all layers to directly minimize a task loss through intermediate supervision can lead to overfitting, as shown in the previous section, which can further result in weaker transfer capabilities on downstream tasks. In contrast, ACNs' implicit compression mechanism naturally balances generalizability and task performance without imposing external constraints.

To ensure fair comparison, we train all models to achieve comparable performance on the CIFAR-100 pre-training task, enabling direct assessment of their transfer capabilities to CIFAR-10. The results in Figure 5.11 confirm our analysis: ACNs demonstrate superior performance on the downstream CIFAR-10 task compared to regularization-based methods, even when upstream CIFAR-100 task performance is similar. This provides

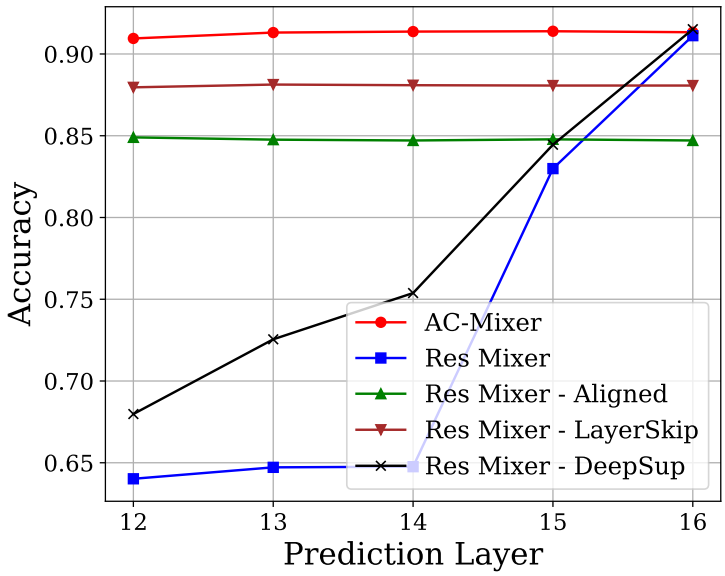
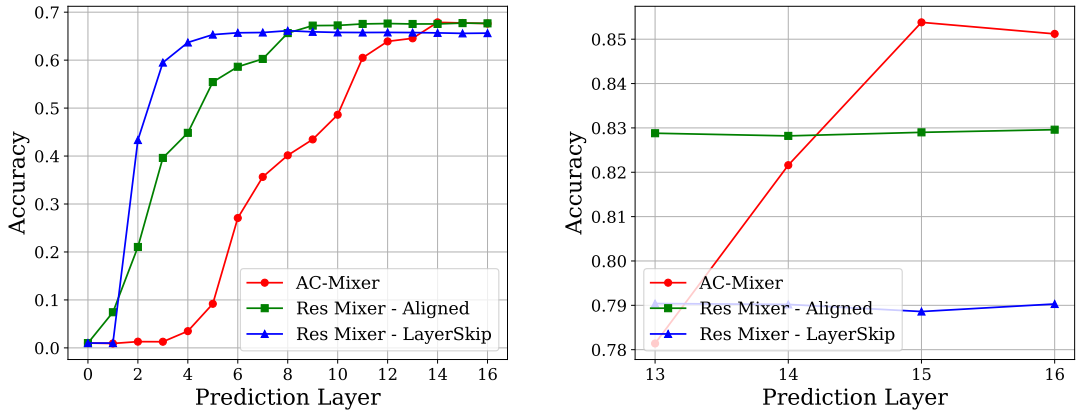


Figure 5.10. *Performance of intermediate layers on CIFAR-10 of AC-Mixer, unregularized Residual Mixer and Residual-Mixer with intermediate losses based on (7) (Aligned), (8) (LayerSkip) and (9) (DeepSup). For better resolution only the last 4 layers are shown in the plot.*

additional evidence that the representations learned by ACNs are more generalizable and thus exhibit greater transferability.

5.3.7 Mitigating Forgetting in Continual Learning with Auto-Compressors

Continual learning involves training models on a sequence of tasks without access to past data, aiming to retain performance on previous tasks while learning new ones (60; 61). A central challenge in CL is catastrophic forgetting—the tendency of neural networks to overwrite old knowledge when updated with new data. Common approaches include data replay methods (62; 63) and regularization techniques that penalize changes to important parameters (64; 65; 66). We've already demonstrated that ACNs, through implicit layer-wise training, dynamically allocate parameters based on task demands while preserving redundant parameters for future tasks. Conversely, Residual Networks optimized for efficient task learning risk overfitting and suboptimal parameter usage in these sequential learning settings. To test our claims, we evaluate both architectures on the split CIFAR-100 continual learning benchmark, comprising 20 sequential disjoint 5-class classification tasks, focusing on task-incremental learning (67) where task identity is known. We utilize MLP-Mixer architectures and we test two continual learning algorithms trained for 10 epochs for each task: naive fine-tuning (Naive FT) and Synaptic Intelligence (SI) (65), which adds a gradient-based regularizer to each parameter depending on how changes in it affect the total loss in a task over the training trajectory. We are using the same MLP-Mixer setup with the Cifar-10 experiment (see above). We train for 10 epochs in each task, using AdamW with learning rate of 0.001 and a batch size of 64. For



(a) *C-100 performance.* **(b)** *C-100 to C-10 Transfer.*

Figure 5.11. *Transfer learning performance (C-100 to C-10) of AC-Mixer and Residual-Mixer with intermediate losses based on (7) (Aligned) and (8) (LayerSkip).*

Synaptic Intelligence we use a coefficient $\lambda = 1$. Across experiments, we report Average Forgetting, defined as the mean difference between a task's best performance (right after it is learned) and its final performance after all tasks are learned, and Average Accuracy, defined as the mean accuracy over all tasks at the end of training. We expect gradient-based regularization methods to perform particularly well with ACNs since unused, redundant parameters receive small gradients, making their detection easier compared to Residual Networks where gradients are more uniformly distributed (see gradient heatmaps, Fig. 5.2(left)). Results in Table 5.3 confirm our intuition: ACNs consistently exhibit significantly less forgetting (up to 18% improvement) compared to Residual Networks. Notably, with SI, increasing ACN depth decreases forgetting—an ideal behavior for CL systems where increasing network capacity reduces forgetting—while Residual Networks show the opposite pattern, indicating potential overfitting. ACNs also achieve better average accuracy across all tasks, further establishing them as a more suitable architecture for continual learning.

		Avg. Accuracy (%) ↑			**Avg. Forgetting (%) ↓**		
M.	**Arch**	$L = 5$	$L = 10$	$L = 15$	$L = 5$	$L = 10$	$L = 15$
*n*FT	ACN	32.97 ± 2.4	32.94 ± 5.3	31.61 ± 2.2	46.55 ± 2.2	45.46 ± 5.8	46.91 ± 2.4
	Res	31.77 ± 1.8	28.16 ± 1	26.14 ± 2.3	52.76 ± 2.3	54.89 ± 1.6	54.49 ± 2.2
SI	ACN	44.5 ± 2.2	46.1 ± 1.3	46.2 **±** 0.8	35.7 ± 2.1	33.8 ± 0.4	32 **±** 1.8
	Res	43.47 ± 3.1	36.1 ± 5	32.1 ± 0.8	42.4 ± 4.1	44.6 ± 3.7	50 ± 2.1

Table 5.3. *Average accuracy and forgetting across layers, methods, and architectures on the Split CIFAR-100 continual learning benchmark. M. stands for Method, nFT stands for naive Fine-Tuning and SI for Synaptic Intelligence. Models are trained for 10 epochs per task, where each task consists of classifying 5 out of 100 classes presented sequentially. L denotes the number of layers in the architecture. ACNs consistently* ***forget less*** *and they also* ***do not waste capacity****.*

These results demonstrate that ACNs are superior to Residual Networks in terms of information organization throughout the network. In particular, the gradient patterns across layers are more distinct and structured, making them not only more interpretable but also more amenable to visualization and analysis. This clarity allows for straightforward identification of which layers are essential and how much each contributes to performance—evident from gradient visualizations, incremental accuracy plots, and accuracy-vs-layer analyses. Moreover, this structured behavior provides a strong foundation for downstream algorithms, such as continual learning algorithms in this case, which benefit significantly from the clearer and more predictable training dynamics exhibited by ACNs.

5.4 Summary of Results

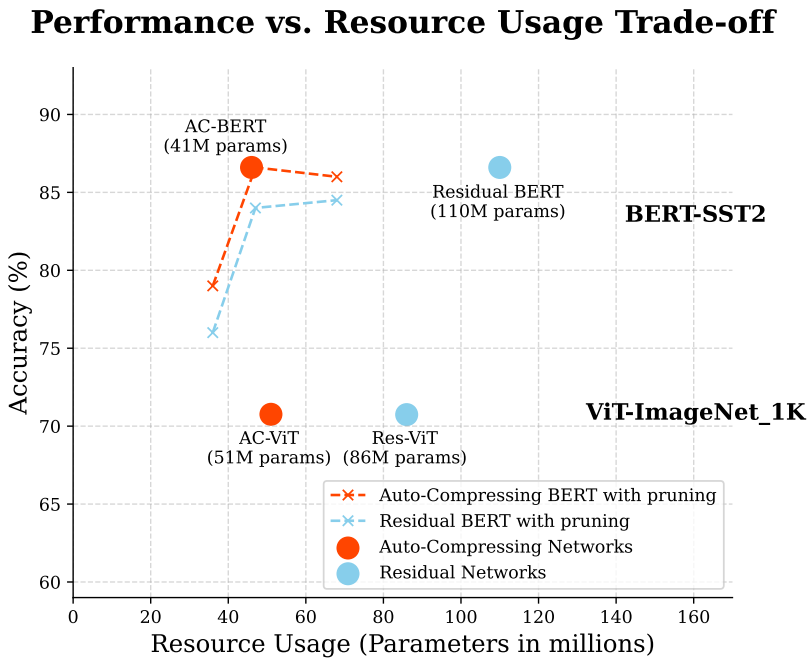


Figure 5.12. *ACNs vs Residual Networks performance vs number of parameters, with and without pruning.*

From our experiments, we conclude that ACNs are able to “push" information down to the early neural layers without performance degradation, resulting in a sparse high-performing network, effectively revealing potential redundant layers and driving the pruning process. This pruning significantly reduces memory requirements and accelerates inference. In all of the experiments we observed that the converged depth between train and validation/test sets matched. Thus, pruning layer depth is a meta-parameter determined directly on the validation set, eliminating the need for a separate pruning procedure after training. Additionally, utilizing ACNs makes it straightforward to produce and distribute differently sized variants of the same architecture with a single training run—for example, distributing tiny, small, medium, and large versions of the model. In Figure 5.12, we show the performance of the pruned ACN models compared to their respective resid-

ual baselines across various experiments. We also note that utilizing the pruned AC-ViT instead of the residual ViT, we can reduce inference time from 13.9 miliseconds to 8.3 miliseconds (CPU).

Models	Accuracy ↑	#Params ↓	#Inference Layers ↓	Storage Size (MB) ↓
Res-Mixer on C10	90.12 ± 0.06	2.5M	16	17.6
AC-Mixer on C10	**90.24** ± 0.05	**1.8M**	**12**	**13.2**
Res-ViT on ImageNet	70.74 ± 0.09	86M	12	330
AC-ViT on ImageNet	**70.76** ± 0.12	**51M**	**7**	**195**
Res-BERT on SST-2	86.63 ± 0.09	110M	12	418
AC-BERT on SST-2	**86.68** ± 0.06	**46M**	**3**	**174**
Res-BERT on QNLI	**83.14** ± 0.07	110M	12	418
AC-BERT on QNLI	83.07 ± 0.1	**46M**	**3**	**174**
Res-BERT on QQP	87.2 ± 0.09	110M	12	418
AC-BERT on QQP	**87.3** ± 0.07	**46M**	**3**	**174**

Table 5.4. *Summary of experimental results (detailed in the previous sections) comparing auto-compressing with residual architectures.*

Beyond parameter efficiency, our experiments demonstrated several additional advantages of ACNs. In noise robustness tests, AC-ViT maintained 51.89% accuracy under severe Gaussian noise $\sigma = 0.4$ compared to ResNet-ViT's 45.46%, and nearly doubled performance (19.98% vs 10.34%) under heavy salt-and-pepper noise at $p = 0.1$. When trained on limited data (100 samples per class), ACNs converged to lower training and test losses significantly faster than residual networks. In transfer learning from CIFAR-100 to CIFAR-10, ACNs achieved 85.7% accuracy compared to 83.2% for the best regularization-based approach while using fewer parameters. Finally, when combined with pruning techniques on language tasks, ACNs maintained 80% accuracy at sparsity levels where residual models dropped to 65%, demonstrating that architectural compression and pruning techniques are complementary rather than redundant.

Chapter 6

Discussion, Conclusions, Limitations and Future Work

6.1 Discussion

Biological motivation for long-connections: Brain Neural Networks (BNNs) combine short and long connections (68), where short connections form dense sub-network hubs, and long connections sparsely link these hubs. Typically, short connections are more numerous and have stronger synaptic weights (94). In (95), the authors show that short connections more efficiently route information across brain areas and sub-networks. Long connections are key for functional diversity, offering unique inputs and novel targets for outputs across sub-networks. The importance of long connections for BNNs is highlighted in imaging (96) and computational modeling studies (97) showing "evidence both of local over-connectivity and of long-distance under-connectivity" in BNNs of individuals on the autistic spectrum (98). This served as our main motivation for exploring long connections in search of architectures that can lead to better representations, improved generalization and enhanced performance in complex tasks.

Biological motivation for auto-compressing networks: The brain itself has mechanisms for creating efficient and robust biological networks. One key efficiency mechanism is synaptic pruning. During early development, an excess of synapses is formed and progressively eliminated through activity-dependent pruning (99). Early studies (69) measured synaptic density across different ages and found that it peaks around 1–2 years of age, followed by a decline to approximately 50% by adulthood. This approach of early overconnectivity followed by pruning has been shown to train neural networks exhibiting significant efficiency and robustness (100). ACNs can be viewed as an initial architectural approach to determining the essential number of layers while learning a task (experience-based), by starting from an overparameterized network at initialization and exploring the parameter space during training. Note, however, that as we have experimentally verified in Section 5.3.5, computational ANN pruning algorithms (motivated by BNN's "use it or lose it" synaptic pruning) can be effectively combined with the ACN architecture to achieve even greater compression gains.

Connection to layer-wise training: Greedy layer-wise training (101; 102) was a popular method for training deep neural networks in a sequential manner, inspired by

cognitive neural development in the prefrontal cortex (103). In our analysis of ACN's dynamics, we show that ACNs naturally exhibit similar layer-wise training behavior, where early layers train first followed by deeper layers. Unlike traditional layer-wise training that requires explicitly freezing layers and careful hyperparameter tuning, this sequential training emerges automatically in ACNs due to their long-connection architecture, effectively providing a "one-shot" version of layer-wise training.

ACNs vs ResNets connectivity patterns: Residual Networks were motivated by improving training robustness - their skip connections were designed to facilitate gradient flow and enable stable training of networks of great depth. However, this architectural innovation had an unexpected benefit: ResNets also demonstrated better generalization compared to standard feedforward networks. This improved generalization appears to stem from the ensemble-like behavior created by the multiple paths through which information can flow. ACNs take inspiration from biological networks' sparse but strategic connectivity patterns. By maintaining direct long-range connections to the output while reducing local skip connections, ACNs achieve both stable training and enhanced generalization through a different mechanism. Rather than relying on dense connectivity and ensemble-like behavior, ACNs encourage the development of more abstract and integrated representations in earlier layers. This architectural choice appears to better support generalization while maintaining robustness in the training. A promising future direction is to integrate small-world network properties into artificial network architectures.

A Remark: Altering connectivity patterns in artificial networks may provide valuable insights that could be mapped back to biological network structures. This approach may contribute to a deeper understanding of why evolutionary processes have led to the specific connectivity patterns observed in biological systems.

6.2 Conclusions

In this paper, we introduced Auto-Compressing Networks (ACNs), a novel family of architectures that naturally compress information into shallow layers of a neural network during training, through architectural design alone. By replacing residual (short) connections with long connections from all layers to the output, ACNs leverage gradient-based optimization to automatically organize information in early layers without requiring explicit compression techniques or additional regularization objectives. By analyzing the gradient dynamics of ACNs, feedforward and residual networks, we revealed how ACNs' unique connectivity patterns fundamentally alter training dynamics, resulting in learned representations distinct from feed-forward and residual architectures. Specifically, the existence of a strong direct feedback signal (from the output directly to each layer) coupled with a weaker (compared to residual networks) forward signal induces an implicit layer-wise training which, as we argued and empirically validated, drives the auto-compression.

Our comprehensive experiments demonstrated that ACNs achieve comparable or superior performance to residual networks while enabling significant practical benefits, for all modalities and baseline architecture tested. Through extensive experiments with both fully connected and transformer-based implementations, we showed that 30-80% of the

top layers in ACNs become effective identity mappings as information concentrates in the bottom layers. This natural compression leads to accelerated inference and reduced memory consumption—all without sacrificing accuracy. We also conducted preliminary experiments on masked language modeling with auto-compressing architectures. There, we showed that pretrained auto-compressing encoders can dynamically adapt to downstream tasks by naturally utilizing only a subset of their pretrained layers without performance degradation, a behavior absent in the residual counterpart. More interestingly, when combined with standard pruning techniques, ACNs' organically compressed representations significantly amplified the effectiveness of these methods, achieving better sparsity vs performance trade-offs compared to residual architectures. We consider combining strong pre-trained ACN architectures with state-of-the-art pruning methods as future work. Overall, as shown in Figure 5.12, we found that ACNs achieve better efficiency-performance balance across multiple tasks including ImageNet, CIFAR-10, and BERT-SST2, maintaining high accuracy while utilizing significantly fewer parameters compared to traditional residual networks. Furthermore, when combined with traditional pruning techniques, ACNs consistently maintain their performance advantage over residual networks across all sparsity levels.

In summary, our empirical results demonstrate that ACNs, when compared to Residual Networks, offer the following advantages:

- Achieve stronger information compression without compromising performance;
- Generalize more effectively in noisy and low-data regimes;
- Significantly mitigate catastrophic forgetting in continual learning tasks;
- Outperform recent information compression regularization-based in both generalization capabilities and transfer learning.

Concluding, Auto-Compressing Networks (ACNs), building on implicit regularization through architectural design insights, represent a promising step towards more self-adapting neural architectures that allocate resources based on the task at hand, while learning sparse yet robust representations. Future research could expand ACNs to self-supervised and multi-task settings, leveraging the pre-training and fine-tuning paradigm. ACNs also hold promise for generative tasks, reducing inference costs and energy consumption. Additionally, developing inference-time algorithms that dynamically adjust the number of layers per sample for optimal performance and efficiency is an intriguing direction for future work. Last but not least, ACNs are only one possible long-connection architecture out of the many that are worth investigating further.

6.3 Limitations

Due to resource constraints, our proposed architecture was evaluated solely on relatively small scale tasks; however, it demonstrated robust and promising performance across various modalities, datasets, and state-of-the-art architectures within this scope.

To fully assess its potential and limitations, further testing on a broader range of tasks is essential. Additionally, applying our method in self-supervised and multi-task learning settings, such as training large-scale language models or multimodal models, represents a significant and exciting avenue for future research.

Furthermore, in our experiments, we observed a trade-off between training and inference cost when choosing between short residual connections and long connections. It is possible that ACN's inherent sparsity and longer training time is what makes them more robust to noise and more efficient in low-data settings. Preliminary experiments further strenghten this belief: as the representations learned by earlier layers become more discriminative focusing on the task at hand, ACNs can still effectivelly transfer their knowledge to downstream tasks.

6.4 Future Work

As discussed in the Limitations section, a natural extension of this work involves applying the proposed architecture to self-supervised and multi-task learning scenarios. This includes training large-scale language models or multimodal models, which represents a promising and exciting direction for future research. In the case of multimodal inputs, it would be particularly interesting to investigate optimal connectivity patterns and fusion strategies across modalities.

Another compelling direction is leveraging ACNs for dynamic inference. In this setting, the model can adaptively decide at inference time which layer to exit from, based on the difficulty of the input. ACNs are especially well-suited for this, as their structure offers a natural way to assess difficulty through intermediate accuracies, offering a method to determine the appropriate early exit point for a given input.

Finally, taking inspiration from the discussion on biological motivation and the connectivity patterns observed in brain networks, a fascinating direction for future research is the design of hybrid architectures that combine both short and long-range connections—effectively interpolating between ACNs and Residual Networks. Such networks would exhibit hybrid connectivity patterns, potentially moving closer to the Small-World characteristics found in biological systems. This raises the question of whether neural networks with graph-theoretic properties similar to those of the brain—such as high clustering and short path lengths—can achieve improved efficiency and generalization.

In this context, general graph-theoretic metrics and measures could be employed to analyze and better understand the structural properties of these architectures. Moreover, we note that the interpolation between ACNs and ResNets is both natural and straightforward to implement, as illustrated in the algorithm below:

ALGORITHM 6.1: *Forward pass of ACNs* ***(a=0)*** *and ResNets* ***(a=1)***

$x \leftarrow \text{emb}(input)$
$current \leftarrow x$
for each `layer` in `layers` **do**
    $x_{\text{out}} \leftarrow \text{layer}(x)$
    $current \leftarrow current + x_{\text{out}}$
    $x \leftarrow x_{\text{out}} + \mathbf{a} \cdot x$
end for
$x_{\text{cls}} \leftarrow current$
$x_{\text{cls}} \leftarrow \text{cls}(x_{\text{cls}})$

Bibliography

[1] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] Daniel Jurafsky και James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rdη έκδοση, 2025. Online manuscript released January 12, 2025.

[3] Andreas Veit, Michael J Wilber και Serge Belongie. *Residual networks behave like ensembles of relatively shallow networks*. *Advances in neural information processing systems*, 29, 2016.

[4] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit και others. *Mlp-mixer: An all-mlp architecture for vision*. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[5] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. σελίδες 32–33, 2009.

[6] Alexey Dosovitskiy. *An image is worth 16x16 words: Transformers for image recognition at scale*. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Jiachen Jiang, Jinxin Zhou και Zhihui Zhu. *Tracing Representation Progression: Analyzing and Enhancing Layer-Wise Similarity*. *The Thirteenth International Conference on Learning Representations*, 2024.

[8] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman και others. *LayerSkip: Enabling early exit inference and self-speculative decoding*. *arXiv preprint arXiv:2404.16710*, 2024.

[9] Chen Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang και Zhuowen Tu. *Deeply-supervised nets*. *Artificial intelligence and statistics*, σελίδες 562–570. Pmlr, 2015.

[10] Alex Krizhevsky, Ilya Sutskever και Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. *Advances in neural information processing systems*, 25, 2012.

[11] Yann LeCun, Yoshua Bengio και Geoffrey Hinton. *Deep learning*. *nature*, 521(7553):436–444, 2015.

[12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever και Dario Amodei. *Language Models are Few-Shot Learners*, 2020.

[13] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso και Daniel A Roberts. *The unreasonable ineffectiveness of the deeper layers*. *arXiv preprint arXiv:2403.17887*, 2024.

[14] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra και Kilian Q Weinberger. *Deep networks with stochastic depth*. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, σελίδες 646–661. Springer, 2016.

[15] Xiao Zhang, Ruoxi Jiang, William Gao, Rebecca Willett και Michael Maire. *Residual Connections Harm Generative Representation Learning*. *arXiv preprint arXiv:2404.10947*, 2024.

[16] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[17] Richard O Duda, Peter E Hart και others. *Pattern classification*. John Wiley & Sons, 2006.

[18] Warren Mcculloch και Walter Pitts. *A Logical Calculus of Ideas Immanent in Nervous Activity*. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[19] Frank Rosenblatt. *The perceptron: A perceiving and recognizing automaton*. Report, Project PARA, Cornell Aeronautical Laboratory, 1957.

[20] David E Rumelhart, Geoffrey E Hinton και Ronald J Williams. *Learning representations by back-propagating errors*. *nature*, 323(6088):533–536, 1986.

[21] Yoshua Bengio, Aaron Courville και Pascal Vincent. *Representation learning: A review and new perspectives*. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[22] Y. Lecun, L. Bottou, Y. Bengio και P. Haffner. *Gradient-based learning applied to document recognition*. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Kunihiko Fukushima και Sei Miyake. *Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position*. *Pattern Recognition*, 15(6):455–469, 1982.

[24] Alex Krizhevsky, Ilya Sutskever και Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Advances in Neural Information Processing Systems*F. Pereira, C.J. Burges, L. Bottou και K.Q. Weinberger, επιμελητές, τόμος 25. Curran Associates, Inc., 2012.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein και others. *Imagenet large scale visual recognition challenge*. *International journal of computer vision*, 115:211–252, 2015.

[26] Michael I. Jordan. *Chapter 25 - Serial Order: A Parallel Distributed Processing Approach*. *Neural-Network Models of Cognition*John W. Donahoe και Vivian Packard Dorsel, επιμελητές, τόμος 121 στο *Advances in Psychology*, σελίδες 471–495. North-Holland, 1997.

[27] Jeffrey L. Elman. *Finding structure in time*. *Cognitive Science*, 14(2):179–211, 1990.

[28] Sepp Hochreiter και Jürgen Schmidhuber. *Long Short-Term Memory*. *Neural Comput.*, 9(8):1735–1780, 1997.

[29] A Vaswani. *Attention is all you need*. *Advances in Neural Information Processing Systems*, 2017.

[30] Dzmitry Bahdanau, Kyunghyun Cho και Yoshua Bengio. *Neural machine translation by jointly learning to align and translate*. *arXiv preprint arXiv:1409.0473*, 2014.

[31] Marc'Aurelio Ranzato, Fu Jie Huang, Y Lan Boureau και Yann LeCun. *Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition*. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 1–8, 2007.

[32] Yoshua Bengio, Pascal Lamblin, Dan Popovici και Hugo Larochelle. *Greedy Layer-Wise Training of Deep Networks*. *Advances in Neural Information Processing Systems*B. Schölkopf, J. Platt και T. Hoffman, επιμελητές, τόμος 19. MIT Press, 2006.

[33] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault και Antoine Bordes. *Supervised learning of universal sentence representations from natural language inference data*. *EMNLP*, 2017.

[34] Jacob Devlin. *Bert: Pre-training of deep bidirectional transformers for language understanding*. *arXiv preprint arXiv:1810.04805*, 2018.

[35] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell και others. *Language models are few-shot learners*. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[36] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski και Armand Joulin. *Emerging properties in self-supervised vision transformers*. *Proceedings of the IEEE/CVF international conference on computer vision*, σελίδες 9650–9660, 2021.

[37] Y. Bengio, P. Simard και P. Frasconi. *Learning long-term dependencies with gradient descent is difficult*. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[38] Sepp Hochreiter. *The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions*. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998.

[39] Rupesh Kumar Srivastava, Klaus Greff και Jürgen Schmidhuber. *Highway networks*. *arXiv preprint arXiv:1505.00387*, 2015.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Deep residual learning for image recognition*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 770–778, 2016.

[41] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer και Tom Goldstein. *Visualizing the loss landscape of neural nets*. *Advances in neural information processing systems*, 31, 2018.

[42] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan Duo Ma και Brian McWilliams. *The shattered gradients problem: If resnets are the answer, then what is the question?* *International conference on machine learning*, σελίδες 342–350. PMLR, 2017.

[43] Alireza Zaeemzadeh, Nazanin Rahnavard και Mubarak Shah. *Norm-preservation: Why residual networks can become extremely deep?* *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3980–3990, 2020.

[44] Gao Huang, Zhuang Liu, Laurens Van Der Maaten και Kilian Q Weinberger. *Densely connected convolutional networks*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 4700–4708, 2017.

[45] Gustav Larsson, Michael Maire και Gregory Shakhnarovich. *Fractalnet: Ultra-deep neural networks without residuals*. *arXiv preprint arXiv:1605.07648*, 2016.

[46] Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret και Martin Jaggi. *DenseFormer: Enhancing Information Flow in Transformers via Depth Weighted Averaging*. *arXiv preprint arXiv:2402.02622*, 2024.

[47] Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min και Xun Zhou. *Hyper-Connections*. *The Thirteenth International Conference on Learning Representations*, 2025.

[48] Muhammad ElNokrashy, Badr AlKhamissi και Mona Diab. *Depth-wise attention (dwatt): A layer fusion method for data-efficient classification*. *arXiv preprint arXiv:2209.15168*, 2022.

[49] Pedro HP Savarese, Leonardo O Mazza και Daniel R Figueiredo. *Learning identity mappings with residual gates*. *arXiv preprint arXiv:1611.01260*, 2016.

[50] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell και Julian McAuley. *Rezero is all you need: Fast convergence at large depth*. *Uncertainty in Artificial Intelligence*, σελίδες 1352–1361. PMLR, 2021.

[51] Kirsten Fischer, David Dahmen και Moritz Helias. *Field theory for optimal signal propagation in ResNets*. *arXiv preprint arXiv:2305.07715*, 2023.

[52] Guillaume Alain και Yoshua Bengio. *Understanding intermediate layers using linear classifier probes*. *arXiv preprint arXiv:1610.01644*, 2016.

[53] Arild Nøkland. *Direct feedback alignment provides learning in deep neural networks*. *Advances in neural information processing systems*, 29, 2016.

[54] Maria Refinetti, Stéphane d'Ascoli, Ruben Ohana και Sebastian Goldt. *Align, then memorise: the dynamics of learning with feedback alignment*. *International Conference on Machine Learning*, σελίδες 8925–8935. PMLR, 2021.

[55] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba και Sanja Fidler. *Aligning books and movies: Towards story-like visual explanations by watching movies and reading books*. *Proceedings of the IEEE international conference on computer vision*, σελίδες 19–27, 2015.

[56] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy και Samuel R Bowman. *GLUE: A multi-task benchmark and analysis platform for natural language understanding*. *arXiv preprint arXiv:1804.07461*, 2018.

[57] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng και Christopher Potts. *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu και Steven Bethard, επιμελητές, σελίδες 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics.

[58] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev και Percy Liang. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*Jian Su, Kevin Duh και Xavier Carreras, επιμελητές, σελίδες 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics.

[59] Victor Sanh, Thomas Wolf και Alexander Rush. *Movement pruning: Adaptive sparsity by fine-tuning*. *Advances in neural information processing systems*, 33:20378–20389, 2020.

[60] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh και Tinne Tuytelaars. *A continual learning survey: Defying forgetting in classification tasks*. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[61] Liyuan Wang, Xingxing Zhang, Hang Su και Jun Zhu. *A comprehensive survey of continual learning: Theory, method and application*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[62] Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl και Christoph H Lampert. *icarl: Incremental classifier and representation learning*. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, σελίδες 2001–2010, 2017.

[63] David Lopez-Paz και Marc'Aurelio Ranzato. *Gradient episodic memory for continual learning*. *Advances in neural information processing systems*, 30, 2017.

[64] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska και others. *Overcoming catastrophic forgetting in neural networks*. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[65] Friedemann Zenke, Ben Poole και Surya Ganguli. *Continual learning through synaptic intelligence*. *International conference on machine learning*, σελίδες 3987–3995. PMLR, 2017.

[66] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach και Tinne Tuytelaars. *Memory aware synapses: Learning what (not) to forget*. *Proceedings of the European conference on computer vision (ECCV)*, σελίδες 139–154, 2018.

[67] Gido MVan de Ven και Andreas S Tolias. *Three scenarios for continual learning*. *arXiv preprint arXiv:1904.07734*, 2019.

[68] Danielle Smith Bassett και ED Bullmore. *Small-world brain networks*. *The neuroscientist*, 12(6):512–523, 2006.

[69] Huttenlocher Peter R. *Synaptic density in human frontal cortex — Developmental changes and effects of aging*. *Brain Research*, 163(2):195–205, 1979.

[70] Jean Pierre Changeux και Antoine Danchin. *Selective stabilisation of developing synapses as a mechanism for the specification of neuronal networks*. *Nature*, 264:705–712, 1976.

[71] L. C. Katz και C. J. Shatz. *Synaptic Activity and the Construction of Cortical Circuits*. *Science*, 274(5290):1133–1138, 1996.

[72] Thomas Elsken, Jan Hendrik Metzen και Frank Hutter. *Neural architecture search: A survey*. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[73] Joseph Weizenbaum. *ELIZA—a computer program for the study of natural language communication between man and machine*. *Commun. ACM*, 9(1):36–45, 1966.

[74] G. Cybenko. *Approximation by superpositions of a sigmoidal function*. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[75] Olivier Delalleau και Yoshua Bengio. *Shallow vs. Deep Sum-Product Networks*. *Advances in Neural Information Processing Systems*J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira και K.Q. Weinberger, επιμελητές, τόμος 24. Curran Associates, Inc., 2011.

[76] Hrushikesh Mhaskar, Qianli Liao και Tomaso Poggio. *When and why are deep networks better than shallow ones?* *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, σελίδα 2343–2349. AAAI Press, 2017.

[77] Jimmy Ba και Rich Caruana. *Do deep nets really need to be deep?* *Advances in neural information processing systems*, 27, 2014.

[78] David Eigen, Jason Rolfe, Rob Fergus και Yann LeCun. *Understanding deep architectures using a recursive convolutional network*. *arXiv preprint arXiv:1312.1847*, 2013.

[79] Yoshua Bengio. *Deep learning of representations: Looking forward*. *International conference on statistical language and speech processing*, σελίδες 1–37. Springer, 2013.

[80] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre Antoine Manzagol, Pascal Vincent και Samy Bengio. *Why Does Unsupervised Pre-training Help Deep Learning?* *Journal of Machine Learning Research*, 11(19):625–660, 2010.

[81] Geoffrey E. Hinton, Simon Osindero και Yee Whye Teh. *A Fast Learning Algorithm for Deep Belief Nets*. *Neural Computation*, 18(7):1527–1554, 2006.

[82] Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li και Li Fei-Fei. *ImageNet: A large-scale hierarchical image database*. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[83] Samuel R Bowman, Gabor Angeli, Christopher Potts και Christopher D Manning. *A large annotated corpus for learning natural language inference*. *EMNLP*, 2015.

[84] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn και Andrew Zisserman. *The Pascal Visual Object Classes (VOC) Challenge*. *International Journal of Computer Vision*, 2010.

[85] Hoo Chang Shin, Holger R Roth, Le Gao, Liang Lu, Ziyue Xu, Isabel Nogues, Jing Yao, Daniel Mollura και Ronald M Summers. *Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning*. *IEEE Transactions on Medical Imaging*, 2016.

[86] Jeremy Howard και Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. *ACL*, 2018.

[87] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, σελίδες 4171–4186, 2019.

[88] Scott Fahlman και Christian Lebiere. *The cascade-correlation learning architecture*. *Advances in neural information processing systems*, 2, 1989.

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. *Proceedings of the IEEE international conference on computer vision*, σελίδες 1026–1034, 2015.

[90] Zi Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi και Tong Zhang. *Exploiting deep representations for neural machine translation*. *arXiv preprint arXiv:1810.10181*, 2018.

[91] Fisher Yu, Dequan Wang, Evan Shelhamer και Trevor Darrell. *Deep layer aggregation*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 2403–2412, 2018.

[92] Sergey Zagoruyko και Nikos Komodakis. *Diracnets: Training very deep neural networks without skip-connections*. *arXiv preprint arXiv:1706.00388*, 2017.

[93] I Loshchilov. *Decoupled weight decay regularization*. *arXiv preprint arXiv:1711.05101*, 2017.

[94] Sarah Feldt Muldoon, Eric W Bridgeford και Danielle S Bassett. *Small-world propensity and weighted brain networks*. *Scientific reports*, 6(1):22057, 2016.

[95] Richard F. Betzel και Danielle S. Bassett. *Specificity and robustness of long-distance connections in weighted, interareal connectomes*. *Proceedings of the National Academy of Sciences*, 115(21):E4880–E4889, 2018.

[96] Christine Ecker, Susan Y Bookheimer και Declan G M Murphy. *Neuroimaging in autism spectrum disorder: brain structure and function across the lifespan*. *The Lancet Neurology*, 14(11):1121–1134, 2015.

[97] James L McClelland. *The basis of hyperspecificity in autism: A preliminary suggestion based on properties of neural nets*. *Journal of Autism and Developmental Disorders*, 30:497–502, 2000.

[98] Sam Wass. *Distortions and disconnections: Disrupted brain connectivity in autism*. *Brain and Cognition*, 75(1):18–28, 2011.

[99] Jill Sakai. *How synaptic pruning shapes neural wiring during development and, possibly, in disease*. *Proceedings of the National Academy of Sciences*, 117(28):16096–16099, 2020.

[100] Saket Navlakha, Alison L Barth και Ziv Bar-Joseph. *Decreasing-rate pruning optimizes the construction of efficient and robust distributed networks*. *PLoS computational biology*, 11(7):e1004347, 2015.

[101] Geoffrey E. Hinton, Simon Osindero και Yee Whye Teh. *A Fast Learning Algorithm for Deep Belief Nets*. *Neural Computation*, 18(7):1527–1554, 2006.

[102] Yoshua Bengio, Pascal Lamblin, Dan Popovici και Hugo Larochelle. *Greedy layer-wise training of deep networks*. *Advances in neural information processing systems*, 19, 2006.

[103] Javier DeFelipe. *The Evolution of the Brain, the Human Nature of Cortical Circuits, and Intellectual Creativity*. *Frontiers in Neuroanatomy*, 5, 2011.

List of Abbreviations

ΑΣΔ	Αυτο-Συμπιεζόμενα Δίκτυα
ΕΝΔ	Εμπρόσθια Νευρωνικά Δίκτυα
ΥΝΔ	Υπολλειματικά Νευωρνικά Δίκτυα
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron
ACN	Auto-Compressing Network
FFN	Feed Forward Network
ResNet	Residual Network
DG	Direct Gradient
FG	Full Gradient
NG	Network-mediated Gradient
BERT	Bidirectional Encoder Representations from Transformers
CL	Continual Learning
FT	Fine-Tuning
SI	Synaptic Intelligence
C-10	CIFAR-10
C-100	CIFAR-100
DeepSup	Deep Supervision