

National Technical University of Athens
School of Electrical and Computer Engineering
Division of signals, control and robotics

Generalized Y-drop: A Loss Conductance-Based Dropout Framework for Neural Networks

Diploma Thesis

of

ANASTASIOS TSETSILAS

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Athens, January 2026



National Technical University of Athens
School of Electrical and Computer Engineering
Division of signals, control and robotics

Generalized Y-drop: A Loss Conductance-Based Dropout Framework for Neural Networks

Diploma Thesis

of

ANASTASIOS TSETSILAS

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Approved by the examination committee on 28th January 2026.

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

Athens, January 2026

Anastasios Tsetsilas
Graduate of School of Electrical and Computer
Engineering, National Technical University of Athens

Copyright © Anastasios Tsetsilas, 2026.
All rights reserved.

You may not copy, reproduce, distribute, publish, display, modify, create derivative works, transmit, or in any way exploit this thesis or part of it for commercial purposes. You may reproduce, store or distribute this thesis for non-profit educational or research purposes, provided that the source is cited, and the present copyright notice is retained. Inquiries for commercial use should be addressed to the original author.
The ideas and conclusions presented in this paper are the author's and do not necessarily reflect the official views of the National Technical University of Athens.

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.

Περίληψη

Το dropout είναι μια ευρέως χρησιμοποιούμενη τεχνική κανονικοποίησης σε βαθιά νευρωνικά δίκτυα, ωστόσο εφαρμόζει ομοιόμορφα πιθανότητες, αγνοώντας τη σημαντικότητα του κάθε νευρώνα. Το Y-drop αντιμετωπίζει αυτή την αδυναμία, χρησιμοποιώντας τη μετρική της αγωγιμότητας για να προσδιορίσει και να θέσει υψηλότερα ποσοστά dropout σε σημαντικούς νευρώνες, πετυχαίνοντας έτσι καλύτερη κανονικοποίηση, ωστόσο προσθέτει αυξημένη πολυπλοκότητα υπερ-παραμέτρων, αυξημένο υπολογιστικό κόστος και δεν είναι ευθυγραμμισμένη με τον στόχο βελτιστοποίησης. Για να αντιμετωπίσουμε αυτά τα ζητήματα, προτείνουμε το Generalized Y-drop, μια απλοποιημένη και πιο δυναμική μέθοδος, ανάθεσης πιθανοτήτων dropout που απαιτεί λιγότερη ρύθμιση και είναι ανθεκτική στον θόρυβο που είναι εγγενής στις μεθόδους απόδοσης βάσισμένες στην κλίσης. Κεντρική συνεισφορά είναι η αγωγιμότητα κόστους , μια νέα αναδιατύπωση της αγωγιμότητας που μετρά τη συνεισφορά κάθε νευρώνα στην τελική συνάρτηση κόστους αντί για την τελική πρόβλεψη. Αυτη η μετρική οδηγεί σε βελτιωμένη κανονικοποίηση τόσο σε συνελικτικά δικτυα όσο και σε μετασχηματιστές όρασης με διάφορα μεγέθη και σύνολα δεδομένων, ευθυγραμμίζοντας καλύτερα την κανονικοποίηση με τον στόχο βελτιστοποίησης και προωθώντας μια πιο ισορροπημένη κατανομή των συνεισφορών στο τελικό κόστος μεταξύ των νευρώνων κάθε επιπέδου σε περισσότερα δείγματα εκπαίδευσης , Συγκεκριμένα, υπό κανονικές ρυθμίσεις, το Generalized Y-drop με αγωγιμότητα κόστους πετυχαίνει βελτίωση της ακρίβειας έως και 0,84% στο ViT-Tiny που έχει εκπαιδευτεί στο CIFAR-100 σε σύγκριση με το απλό dropout και μπορεί να φτάσει έως και 1,18% σε πιο ακριβές παραλλαγές. Οι βελτίωσεις ακρίβειας εμφανίζονται με μεγαλύτερες αρχιτεκτονικές και σύνολα δεδομένων, πετυχαίνοντας έως και 0,48% στο ViT-Base που εκπαιδεύτηκε στο ImageNet-1K με κανονικές ρυθμίσεις. Μέσω πειραμάτων και αναλύσεων, εξηγούμε τα βασικά αιτία της βελτιωμένης κανονικοποίησης, κατανοούμε βασικές πτυχές και μοτίβα της αγωγιμότητας απώλειας, προτείνουμε ειδικές παραλλαγές για μετασχηματιστές όρασης, καθώς και τους περιορισμούς της μεθόδου. Επιπλέον, εισάγουμε μια παραλλαγή με αραιότερη ενημέρωση των πιθανοτήτων σε μεταγενέστερα στάδια που καταφέρνει να διατηρήσει το 92% των κερδών απόδοσης, μειώνοντας ταυτόχρονα το πρόσθετο υπολογιστικό κόστος κατά σχεδόν 40%. Συνολικά, η εργασία συμβάλλει στην ανάπτυξη αποτελεσματικότερων σχημάτων κανονικοποίησης που βασίζονται σε σήματα απόδοσης, βελτιώνοντας την σταθερότητα, την ευθυγράμμιση με τη βελτιστοποίηση και την πρακτική εφαρμοσιμότητα σε μεγαλύτερες αρχιτεκτονικές βαθιάς μάθησης.

Λέξεις Κλειδιά

Βαθιά μάθηση, Επεξηγηματικό AI, Dropout, Αγωγιμότητα, Μετασχηματιστές Όρασης

Abstract

Dropout is a widely used regularization technique in deep neural networks, yet the standard formulation applies uniform dropout rates, disregarding the relative importance of individual neurons. Y-drop is a method that addresses this limitation, using the Conductance explainability metric to identify and assign higher dropout rates to important neurons within a layer, achieving improved regularization, but it suffers from increased hyperparameter complexity, lack of task alignment and high computational cost. To address those issues we propose Generalized Y-drop, a simplified more dynamic dropout rate assignment framework that requires less tuning and is robust to the noise inherent in gradient-based attribution methods. Most notably, we introduce Loss Conductance, a novel loss-based reformulation of Conductance that measures each neuron's contribution to the training loss instead of the final prediction. This metric consistently improves generalization across convolutional and Vision Transformer architectures at various scales and dataset complexities by better aligning regularization with the optimization objective and promoting a more balanced distribution of loss contributions across neurons and training samples, mitigating the learning of superficial, dataset-specific patterns. More specifically, under standard settings Generalized Y-drop with Loss Conductance achieves up to 0.84% improvement in accuracy on ViT-Tiny trained on CIFAR-100 compared to standard dropout that can scale up to 1.18% in more expensive variants. Improvements appear with larger architectures and datasets achieving up to 0.48% on ViT-Base trained on ImageNet-1K in standard settings. Through heavy experimentation and analysis we explain the root cause of the improved regularization and understand key aspects and patterns of Loss Conductance, opening the way for further improvements, transformer specific optimizations, as well as presenting the limitations of the framework. Moreover we introduce a scheduled dropout-rate update frequency variant that manages to retain 92% of the performance gains while reducing the additional computational cost by almost 40%. Overall, this work advances conductance-based regularization by improving robustness, optimization alignment and practical applicability for scalable use in modern deep learning architectures.

Keywords

Deep Learning, Explainable AI, Regularization, Dropout, Conductance, Vision Transformers

In memory of my uncle, Athanasios

Ευχαριστίες

Με την παράδοση αυτής της διπλωματικής, κλείνει για μένα ένα σημαντικό κεφάλαιο, αυτό των σπουδών μου στη Σχολη Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Οι εμπειρίες, οι προκλήσεις, οι άνθρωποι και οι στιγμές που έζησα όλα αυτά τα χρόνια υπήρξαν καθοριστικές για τη διαμόρφωση του ατόμου που είμαι σήμερα και θα με συνοδεύουν για ολόκληρη τη ζωή μου.

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Αλέξανδρο Ποταμιάνο. Οι συμβουλές και η καθοδήγησή του, τόσο σε ακαδημαϊκό όσο και σε ανθρώπινο επίπεδο έπαιξαν κομβικό ρόλο στον τρόπο σκέψης μου και αποτέλεσαν πολύτιμες πηγές γνώσης και έμπνευσης καθ' όλη τη διάρκεια εκπόνησης της εργασίας. Επιπλέον, θα ήθελα να ευχαριστήσω ιδιαίτερα τον δρ. Ευθύμιο Γεωργίου για τον χρόνο που αφιέρωσε και τις συμβουλές του, οι οποίες συνέβαλαν ουσιαστικά στην ολοκλήρωση αυτής της δουλειάς. Στη συνέχεια, θέλω να ευχαριστήσω τους φίλους μου και ιδιαίτερα τους συμφοιτητές μου, οι οποίοι έκαναν αυτά τα χρόνια αξέχαστα, γεμάτα όμορφες στιγμές, δημιουργώντας συνεχώς ένα θετικό κλίμα και προσφέροντάς μου τη βοήθειά τους κάθε στιγμή που τη χρειαζόμουν. Τέλος, το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένειά μου, στους γονείς μου, Δημήτρη και Παρασκευή, και στην αδερφή μου, Χριστίνα. Η αγάπη, στήριξη και τυφλή εμπιστοσύνη που μου έδειξαν όλα αυτά τα χρόνια υπήρξαν υπήρξαν θεμέλιο για κάθε μου βήμα και μου έδωσαν την δυνατότητα να ζήσω αυτή την αξέχαστη εμπειρια.

Athens, January 2026

Anastasios Tsetsilas

Περιεχόμενα

Abstract **5**

Abstract **7**

Ευχαριστίες **11**

0 Εκτεταμένη Ελληνική Περίληψη **23**
0.1 Εισαγωγή . . . 23
0.2 Θεωρητικό Υπόβαθρο . . . 24
0.2.1 Μηχανική Μάθηση . . . 24
0.2.2 Επεξηγματικές Μετρικές . . . 27
0.2.3 Γενίκευση και κανονικοποίηση . . . 29
0.3 Generalized Y-drop . . . 32
0.3.1 Προβλήματα με το Y-drop . . . 32
0.3.2 Προτεινόμενες μέθοδοι . . . 33
0.4 Πειραματική Αξιολόγηση . . . 36
0.4.1 Ρυθμίσεις πειραμάτων . . . 36
0.4.2 Αποτελέσματα . . . 37
0.4.3 Ανάλυση Αγωγιμότητας Κόστους . . . 45
0.5 Συμπεράσματα . . . 49

1 Introduction **51**
1.1 Motivation . . . 51
1.2 Contributions . . . 52
1.3 Outline . . . 53

2 Machine Learning **55**
2.1 Introduction . . . 55
2.2 Machine Learning Variants . . . 56
2.3 Neural Networks . . . 57
2.3.1 Artificial Neural Networks . . . 57
2.3.2 Feedforward Neural Networks . . . 59
2.3.3 Convolutional Neural Networks . . . 59
2.3.4 Recurrent Neural Networks . . . 60
2.3.5 Transformers and Attention . . . 61
2.4 Training . . . 64

2.4.1 Loss Function . . . 64
2.4.2 Optimization . . . 64

3 Explainability Metrics **67**
3.1 Introduction . . . 67
3.2 Occlusion-Based Methods . . . 67
3.2.1 Occlusion Sensitivity . . . 68
3.2.2 RISE . . . 68
3.3 Gradient-Based Methods . . . 68
3.3.1 Vanilla Gradients (Saliency Maps). . . . 68
3.3.2 Input x Gradient. . . . 68
3.3.3 SmoothGrad. . . . 69
3.3.4 Integrated Gradients. . . . 69
3.3.5 Conductance . . . 70
3.4 Propagation/Decomposition-Based Methods . . . 71
3.4.1 Layer-wise Relevance Propagation (LRP). . . . 71
3.4.2 DeepLIFT. . . . 72

4 Generalization and Regularization **73**
4.1 Introduction . . . 73
4.2 Loss-Based Regularization . . . 73
4.2.1 L2 Regularization . . . 74
4.2.2 L1 Regularization . . . 74
4.2.3 Label Smoothing . . . 74
4.3 Normalization Techniques . . . 74
4.3.1 Batch Normalization . . . 74
4.3.2 Layer Normalization . . . 74
4.4 Data Augmentations . . . 75
4.4.1 AutoAugment . . . 75
4.4.2 RandAugment . . . 75
4.4.3 Repeated Augmentation. . . . 76
4.4.4 Cutout . . . 76
4.4.5 Random Erasing . . . 76
4.4.6 Mixup . . . 76
4.4.7 CutMix . . . 77
4.5 Drop-Based Regularization . . . 77
4.5.1 Dropout . . . 77
4.5.2 Standout. . . . 79
4.5.3 R-drop. . . . 79
4.5.4 Drop-Path (Stochastic Depth) . . . 79
4.5.5 Y-Drop . . . 79

5 Generalized Y-drop **83**
5.1 Introduction . . . 83
5.2 General Motivation . . . 83
5.3 Issues with Y-drop . . . 84
5.4 Proposed Methods . . . 85
5.4.1 Alternative Dropout Rate Assignments . . . 85
5.4.2 Loss Conductance . . . 87
5.4.3 Computationally Cheaper Variants . . . 88
5.4.4 Application on Transformer Architectures . . . 89
5.4.5 Scaling Generalized Y-Drop/Dropout . . . 91

6 Experiments **93**
6.1 Introduction . . . 93
6.2 Experimental Setup . . . 93
6.2.1 Introduction . . . 93
6.2.2 Architectures . . . 94
6.2.3 Datasets . . . 94
6.2.4 Training Details . . . 95
6.3 Dropout Rate Assignment . . . 96
6.3.1 Introduction . . . 96
6.3.2 Method Comparison . . . 97
6.3.3 Winsorization. . . . 98
6.3.4 Distribution Sharpness. . . . 99
6.3.5 Conclusion. . . . 100
6.4 Loss Conductance . . . 100
6.4.1 Introduction . . . 100
6.4.2 Results . . . 101
6.4.3 Train-Validation Loss Comparison . . . 101
6.5 Token Selection and Layer Normalization . . . 102
6.5.1 Introduction . . . 102
6.5.2 Token Selection . . . 102
6.5.3 Layer Normalization . . . 103
6.5.4 Conclusion . . . 104
6.6 Hyperparameter Impact . . . 104
6.6.1 Introduction . . . 104
6.6.2 Elasticity . . . 105
6.6.3 Annealing Factor . . . 105
6.6.4 Linear Interpolation Steps . . . 106
6.6.5 Conductance Calculation Samples. . . . 107
6.6.6 Conclusion . . . 107
6.7 Computational Cost Reduction Attempts . . . 108
6.7.1 Introduction . . . 108
6.7.2 Early Termination . . . 108

6.7.3 Infrequent updates . . . 109
6.7.4 Conclusion . . . 110
6.8 Imagenet, Larger Transformers and Data Augmentation . . . 110
6.8.1 Introduction . . . 110
6.8.2 Larger Vision Transformers . . . 111
6.8.3 DeiT Tests . . . 111
6.9 Loss Conductance a Deeper Analysis . . . 113
6.9.1 Introduction . . . 113
6.9.2 Loss Conductance during training . . . 113
6.9.3 Loss Conductance Test data . . . 115
6.9.4 Pattern validation Experiments . . . 117
6.9.5 Conclusion and Implications . . . 118

7 Conclusions and Future Work **121**
7.1 Conclusions . . . 121
7.2 Future Work . . . 122

Bibliography **133**

List of Abbreviations **135**

Κατάλογος Σχημάτων

1 Καμπύλες της συνάρτησης κόστους στο σύνολο εκπαίδευσης και αξιολόγησης στο ViT-Tiny στο CIFAR-100 με χρήση Dropout συγκριτικά με Generalized Y-drop (αριστερά: πραγματικές τιμές και για τις δύο εκτελέσεις, δεξιά: διαφορά μεταξύ εκπαίδευσης και αξιολόγησης και για τις δύο εκτελέσεις). 40

2 Interquartile range, διασπορα, Gini coefficient και διασπορά εποχής της αγωγιμότητας κόστους σε διαάφορα επίπεδα κατά την διάρκεια της εκπαίδευσης.Σύγκριση Dropout εναντίον Generalized Y-drop. 46

3 Άθροισμα της μέσης αγωιγμότητας κόστους ανα νευρώνα σε κάθε επίπεδο για ολόκληρο το σύνολο αξιολόγησης σε κάθε επίπεδο. Σύγκριση Dropout εναντιον Generalized Y-drop μονο στα τελευταία 6 blocks. 48

4 Intrinsic Dimension: Σύγκριση Dropout εναντιον Generalized Y-drop. . . . 48

2.1 Schematic differences between classification and regression. Picture taken from [1]. 56

2.2 A schematic of a feedforward neural network layer. Picture taken from [2]. 59

2.3 A schematic illustration of a Convolutional Neural Network architecture. Picture taken from [3]. 60

2.4 A standard RNN and its unfolded (time-unrolled) representation. Picture taken from [4]. 61

2.5 Illustration of the Transformer encoder–decoder architecture. Picture taken from [5]. 63

2.6 Schematic of the Vision Transformer. Picture taken from[6]. 64

4.1 Illustrative examples of randomly selected transforms nn RandAugment. Picture taken from [6]. 76

4.2 Visual comparison of Cutout, Mixup, and CutMix augmentations. Picture taken from [7]. 77

4.3 Illustration of an MLP before and after applying dropout. Picture taken from *Dive into Deep Learning* (d2l.ai). 78

4.4 Overview of the Y-drop procedure with conductance-based ranking and bucketed dropout. Taken from [8]. 80

5.1 Showcasing per token distribution,with cls token highlighted with blue . . 90

6.1 A histogram of the mean dropout rate assigned to neurons of a layer across an epoch using Ranking Rates. 99

6.2 Training and validation loss curves for ViT-Tiny on CIFAR-100 Dropout vs. Generalized Y-drop(left: actual values for both runs, right: difference between training and validation loss in both runs). 102
6.3 Validation loss for ViT-Tiny on CIFAR-100 vs. wall-clock time for Dropout vs. Generalized Y-drop. 108
6.4 Training and validation loss curves for DeiT-Tiny on CIFAR-100 Dropout vs. Generalized Y-drop(left: actual values for both runs, right: difference between training and validation loss in both runs). 113
6.5 Within-epoch variance across a layer during training. Comparison of Dropout vs. Generalized Y-drop. 114
6.6 Interquartile range, variance, and Gini coefficient of mean Loss Conductance across layers during training.Comparison of Dropout vs. Generalized Y-drop. 115
6.7 Dispersion metrics of mean Loss Conductance values for the whole test set in all layers. Comparison of Dropout vs. Generalized Y-drop: Actual values (left column) vs. differences between runs (right column). 116
6.8 Sum of all neurons mean Loss Conductance values for the whole test set in all layers. Comparison of Dropout vs. Generalized Y-drop(left: actual values for both runs, right: difference between runs). 117
6.10 Sum of mean Loss Conductance per nueuron values per layer for the whole test set in all layers.Comparison of Dropout vs. Generalized Y-drop in only the last 6 blocks (left: actual values for both runs, right: difference between runs). 118
6.11 Intrinsic Dimension: Comparison Dropout vs Generalized Y-drop (left: actual values for both runs, right: difference between runs). 119
6.9 Dispersion metrics and Sum of mean Loss Conductance values for the whole test set in all layers. Comparison of Generalized Y-drop vs. Generalized Y-drop with 4x samples in Conductance calculation: Actual values (left column) vs. differences between runs (right column). 120

Κατάλογος Πινάκων

1 Μεγέθη των μοντέλων ΐT που χρησιμοποιήθηκαν στα πειράματα. 36

2 Ακρίβεια σύνολου αξιολόγησης (%) στα μοντέλα CNN στο σύνολο CIFAR-100 δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. 38

3 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CiFAR-100 με διαφορετικές στρατηγικές Generalized Y-drop για διάφορες τιμές $p_{\max}$, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. Το ΔY-drop είναι η διαφορά με το Dropout. 38

4 Ακρίβεια σύνολου αξιολόγησης (%) στα μοντέλα CNN στο σύνολο CIFAR-100 χρησιμοποιώντας Generalized Y-drop με Αγωγιμότητα Κόστους έναντι της απλής Αγωγιμότητας, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. 39

5 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στα σύνολα CIFAR-100 και CIFAR-10 χρησιμοποιώντας Generalized Y-drop με Αγωγιμότητα Κόστους έναντι της απλής Αγωγιμότητας, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 39

6 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στα σύνολα CIFAR-100 και CIFAR-10 χρησιμοποιώντας Generalized Y-drop με διαφορετικές στρατηγικές επιλογής tokens, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 40

7 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στα σύνολα CIFAR-100 και CIFAR-10 χρησιμοποιώντας Generalized Y-drop με αγωγιμότητα κόστους υπολογισμένη πριν ή μετά το Layer Normalization, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 41

8 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές Elasticity. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 42

9 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές annealing factor. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 42

10 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές n_{steps}. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 43
11 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές n_{samples}. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 43
12 Ακρίβεια σύνολου αξιολόγησης (%) (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για το Generalized Y-drop με διαφορετικές στρατηγικές πρόωρου τερματισμού. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 44
13 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για το Generalized Y-drop με στρατηγικές αραιότερων ενημερώσεων. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 44
14 Ακρίβεια σύνολου αξιολόγησης (%) σε μεγαλύτερα μοντέλα ViT στο σύνολο ImageNet. (μέσος όρος ± τυπική απόκλιση για 3 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. 45
15 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο DeiT στο σύνολο ImageNet με χρήση αυξανόμενου Generalized Y-drop. (μέσος όρος ± τυπική απόκλιση για 3 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Drop-Path. 45
16 Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 οπου το Generalized Y-drop χρησιμοποιείται στα τελυταία μόνο μπλοκς ενώ τα πρώτα χρησιμοποιούν dropout.(μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout. . . 47

6.1 ViT model sizes used in experiments. 94
6.2 Test accuracy (%) of CNN models on the CIFAR-100 dataset reported as mean ± standard deviation across 5 runs. 97
6.3 Test accuracy (%) of ViT-Tiny model on the CIFAR-100 dataset using the different generalized Y-drop methods across different p_{max} values reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . 98
6.4 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-Relative with different $p_{\max}$ and winsorization settings (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. 99
6.5 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-Ranking with different power settings. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. 100
6.6 Test accuracy (%) of CNN models on the CIFAR-100 dataset using Y-drop with Loss Conductance vs standard Conductance, reported as mean ± standard deviation across 5 runs. 101
6.7 Test accuracy (%) of ViT-Tiny model on the CIFAR-100 and CIFAR-10 datasets using Y-drop Loss Conductance vs standard Conductance reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. 101

6.8 Test accuracy (%) of ViT-Tiny model on the CIFAR-100 and CIFAR-10 datasets using different token selection methods for Generalized Y-drop reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 103
6.9 Test accuracy (%) of ViT-Tiny model on the CIFAR-100 and CIFAR-10 datasets using Y-drop with metrics calculated before Layer Normalization vs after reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 103
6.10 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different elasticity values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 105
6.11 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different annealing factor values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 105
6.12 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different interpolation step amount values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 106
6.13 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different sample amount values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 107
6.14 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different early termination strategies. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 109
6.15 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different sample amount values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 109
6.16 Test accuracy (%) of Larger Vit Architectures on the ImageNet dataset reported as mean ± standard deviation across 3 runs. . . . 111
6.17 Test accuracy (%) of DeiT-Tiny on the ImageNet dataset using Generalized Y-drop reported as mean ± standard deviation across 3 runs. . . . 112
6.18 Test accuracy (%) of DeiT Architectures on the ImageNet dataset using scaling Y-drop reported as mean ± standard deviation across 3 runs. . . . 112
6.19 Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop being applied only in deeper layers, while the rest use dropout. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout. . . . 118

Κεφάλαιο 0

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Τα τελευταία χρόνια, η μηχανική μάθηση και ιδίως ο υποτομέας της, η βαθιά μάθηση, έχει επιτύχει εξαιρετικές επιδόσεις σε διάφορους τομείς, κάτι που ωστόσο χρειάστηκε μεγάλη αύξηση του μεγέθους, του βάθους και της πολυπλοκότητας των αρχιτεκτονικών. [9, 10]

Η εξέλιξη αυτή, ωστόσο, έχει καταστήσει ολοένα δυσκολότερη την κατανόηση της εσωτερικής λειτουργίας των μοντέλων και έχει οδηγήσει αυτά να αντιμετωπίζονται ουσιαστικα ως μαύρα κουτιά [11]. Για την αντιμετώπιση του προβλήματος, αναπτύχθηκε ένα ευρύ σύνολο μεθόδων στο πλαίσιο της Εξηγήσιμης Τεχνητής Νοημοσύνης (Explainable Artificial Intelligence) που παρέχουν πληροφορίες για την διαδικασία λήψης αποφάσεων των μοντέλων. Μια σημαντική υποκατηγορία είναι οι μέθοδοι απόδοσης (attribution methods) οι οποίες βαθμολογούν κομμάτια του μοντέλου όπως δεδομένα εισόδου, αναπαραστάσεις δεδομένων, νευρώνες ή επίπεδα (layers) ως προς την συμβολή του στην τελική πρόβλεψη.

Πέρα από την ενίσχυση της ερμηνευσιμότητας, οι επεξηγηματικές μετρικές έχουν αξιοποιηθεί και ως εργαλεία βελτίωσης της συμπεριφοράς των μοντέλων, για παράδειγμα στον εντοπισμό και τη διόρθωση σφαλμάτων στα δεδομένα εκπαίδευσης ή στη συμπίεση και το κλάδεμα μοντέλων (model compression and pruning). Μια ακόμη πολλά υποσχόμενη, αλλά συγκριτικά λιγότερο διερευνημένη, κατεύθυνση είναι η ενσωμάτωση τέτοιων σημάτων κατά τη διάρκεια της εκπαίδευσης, ώστε να καθοδηγείται η βελτιστοποίηση προς επιθυμητές λύσεις. Ένα παράδειγμα μιας τέτοιας μεθόδου είναι το Y-drop [8] που χρησιμοποιεί την αγωγιμότητα (Conductance) [12] ως ενδειξη σημασίας και με βάση αυτή, αναθέτει δυναμικά πιθανότητες dropout στους νευρώνες του δικτύου.

Η προσέγγιση αυτή οδηγεί σε ισχυρότερη κανονικοποίηση και σε καλύτερη τελική απόδοση σε σχέση με το κλασικό dropout, ωστόσο παρουσιάζει περιορισμούς που δυσχεραίνουν την πρακτική εφαρμογή και την κλιμακωση του. Καθώς απαιτεί ρύθμιση μεγάλου αριθμού υπερ-παραμέτρων, το σήμα σημαντικότητας δεν είναι άμεσα ευθυγραμμισμένο με τον εκπαιδευτικό στόχο και η μέθοδος συνεπάγεται σημαντικό πρόσθετο υπολογιστικό κόστος.

Στην παρούσα εργασία προτείνουμε το Generalized Y-drop, μια απλοποιημένη και πιο ευέλικτη μέθοδο δυναμικής ανάθεσης πιθανοτήτων dropout, το οποίο αντιμετωπίζει τις αδυναμίες του Y-drop μέσω νέων σχεδιαστικών επιλογών. Κεντρικό στοιχείο της προσέγγισής μας αποτελεί η εισαγωγή της αγωγιμότητας κόστους (Loss Conductance), μιας πρωτότυπης παραλλαγής της αγωγιμότητας που αποδίδει σημαντικότητα στους νευρώνες ως προς τη συ-

νάρτηση κόστους, και όχι ως προς την τελική πρόβλεψη. Μέσω πειραμάτων και αναλύσεων προσπαθουμε να ανακαλύψουμε τις πηγές της βελτιωμένης επίδοσης, να μελετήσουμε τους περιορισμούς της μεθόδου, να ερμηνεύσουμε τα προκύπτοντα μοτίβα στις μετρικές απόδοσης, καθώς και να προτείνουμε προσαρμογές για μεγιστοποίηση της αποτελεσματικότητας σε αρχιτεκτονικές μετασχηματιστών (Transformers).

0.2 Θεωρητικό Υπόβαθρο

0.2.1 Μηχανική Μάθηση

Ενώ η τεχνητή νοημοσύνη είναι η ευρύτερη έννοια της προσομοίωσης της ανθρώπινης νοημοσύνης, η μηχανική μάθηση ασχολείται με την ανάπτυξη τεχνικών που αυτόματα ανακαλύπτουν μοτίβα στα δεδομένα και τα χρησιμοποιούν για να προβλέψουν μελλοντικά δεδομένα δίχως την ανάγκη ρητού προγραμματισμού και ορισμού προκαθορισμένων κανόνων [13]. Διάφορες τεχνικές έχουν χρησιμοποιηθεί, απο πρώιμα στατιστικά μοντέλα έως βαθιά νευρωνικα δικτυα (deep neural networks) που μοντελοποιούν υψηλής πολυπλοκότητας μοτίβα και συσχετίσεις δεδομένων μέσω ιεραρχικών δομών τεχνητών νευρωνικών δικτύων (Artificial neural networks) δίχως την ανάγκη για σχεδίαση χαρακτηριστικών (feature engineering) και έχουν επιφέρει σημαντικές προόδους στην υπολογιστική όραση, την επεξεργασία φυσικής γλώσσας και την αναγνώριση ομιλίας. [9]

Είδη Μηχανικής Μάθησης

- **Επιβλεπόμενη μάθηση** Η επιβλεπόμενη μάθηση είναι ένας τύπος μηχανικής μάθησης όπου ένα μοντέλο εκπαιδεύεται σε ένα σε ένα σύνολο δεδομένων με ετικέτες για να μάθει την σχέση μεταξύ των δεδομένων εισόδου x_i και των αντίστοιχων ετικετών τους y_i με στόχο την πρόβλεψη των ετικετών σε νεα μη προηγουμένως εμφανισμένα δεδομένα. [14] Οι μέθοδοι χωρίζονται με βάση την φύση της ετικέτας σε μοντέλα ταξινόμησης που έχουν στόχο μια ή περισσότερες διακριτές κατηγορίες και παλινδρόμησης που έχουν στόχο μια συνεχή αριθμητική τιμή.

- **Μη επιβλεπόμενη μάθηση** Η μη επιβλεπόμενη μάθηση αναθέτει σε μοντέλα την ανακάλυψη της εγγενούς δομής, μοτίβων και σχέσεων σε δεδομένα δίχως ετικέτες. Ο στόχος είναι να βρεθούν κοινά χαρακτηριστικά που περιγράφουν την κατανομή των δεδομένων και με βάση αυτά είτε να ομαδοποιηθούν παρόμοια δεδομένα (αλγόριθμοι ομαδοποίησης) είτε να ανακαλυφθούν σχέσεις μεταξύ χαρακτηριστικών των δεδομένων(αλγόριθμοι συσχέτισης). [15]

- **Ενισχυτική Μάθηση** Η ενισχυτική μάθηση είναι ένας τύπος μηχανικής μάθησης όπου ένας πράκτορας (agent) μαθαίνει να παίρνει κατάλληλες αλληλουχίες αποφάσεων μέσω αλληλεπιδράσεων με ένα περιβάλλον για τις οποίες λαμβάνει ανταμοιβές ή ποινές ανάλογα με το πόσο επιθυμητές είναι. Μέσω πολλών επαναλήψεων αλληλεπίδρασης και ανατροφοδότησης με στόχο να μεγιστοποίησει την συνολική ανταμοιβή, μαθαίνει μια βέλτιστη διαδικασία λήψης αποφάσεων [16, 17].

Νευρωνικά Δίκτυα

- **Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks, ANNs)**
  Τα τεχνητά νευρωνικά δίκτυα ANN αποτελούν το θεμέλιο της σύγχρονης βαθιάς μάθησης. Αποτελούνται από νευρώνες που έχουν στόχο την προσομοίωση των βιολογικών νευρώνων του ανθρώπινου εγκεφάλου. Κάθε νευρώνας υπολογίζει ένα σταθμισμένο άθροισμα της εισόδου του και του εφαρμόζει μια μη γραμμική συνάρτηση ενεργοποίησης. Πολλαπλοι νευρώνες οργανώνονται σε ομάδες που ονομάζονται επίπεδα και κατάλληλη σύνθεση πολλών επιπέδων επιτρέπει μέσω εκπαίδευσης την προσέγγιση της πολύπλοκη σχέσης των δεδομένων και την ελαχιστοποίηση του σφάλματος για τον τελικό στόχο. [18, 19]

- **Προωθητικά Νευρωνικά Δίκτυα (FeedForward Neural Networks, FFNs)**
  Τα προωθητικά νευρωνικά δίκτυα αντιπροσωπεύουν την απλούστερη μορφή νευρωνικού δικτύου όσον αφορά τη συνδεσιμότητα με τους νευρώνες να είναι διατεταγμένοι σε μια ακολουθία επιπέδων και οι συνδέσεις ρέουν αυστηρά προς μία κατεύθυνση από το επίπεδο εισόδου προς το επίπεδου εξόδου δίχως συνδέσεις ανατροφοδότηση σε προηγούμενα επίπεδα. Κάθε επίπεδο παράγει μια νέα αναπαράσταση των δεδομένων με όλο πιο αφηρημένα και χρήσιμα για τον τελικό στόχο χαρακτηριστικά. [9]

- **Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks, CNNs)**
  Τα συνελικτικά νευρωνικά δίκτυα είναι μια εξειδικευμένη κατηγορία νευρωνικών δικτύων με στόχο την επεξεργασια δεδομένων με μορφή πλέγματος και παρουσιάζουν χωρικές ή τοπικές συσχετίσεις, όπως εικόνες, καρέ βίντεο ή φασματογράμματα. Χρησιμοποιούν την λειτουργία της συνέλιξης όπου συνελικτικά φίλτρα «σαρώσουν» όλη την είσοδο και υπολογίζουν τοπικά μοτίβα σε όλη την εικόνα. Με αυτόν τον τρόπο χαρακτηριστικά που μπορεί να εμφανιστούν σε οποιαδήποτε χωρική τοποθεσία, χωρίς την ανάγκη ξεχωριστής αναζήτησης για κάθε τοποθεσία και με χρήση λιγότερων υπολογιστικών παραμέτρων [20, 21]. Χρησιμοποιούνται ακολουθίες συνελλικτικών επιπέδων με τα πρώτα επίπεδα να ανιχνεύουν απλά γεωμετρικά χαρακτηριστικά όπως ακμές, γωνίες και υφές ενω βαθύτερα επίπεδα τα ενσωματώνουν ευρύτερες χωρικές πληροφορίες για να αναγνωρίσουν πιο σύνθετες δομές. [22]

- **Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural networks, RNNs)**
  Τα αναδρομικά νευρωνικά δίκτυα είναι μια κατηγορία νευρωνικών δικτύων που έχουν σχεδιαστεί για να χειρίζονται διαδοχικά δεδομένα, όπου η σειρά εμφάνισης των δεδομένων είναι σημαντική και οι πληροφορίες πρέπει να διατηρούνται σε όλα τα χρονικά βήματα. Χρησιμοποιούν βρόχους ανατροφοδότησης που επιστρέφουν την έξοδο της μονάδας πίσω στην ίδια και επιτρέπουν στις πληροφορίες να παραμένουν στο χρόνο, καθιστώντας τα ιδανικά για εργασίες όπως η μοντελοποίηση γλώσσας, η αναγνώριση ομιλίας και η πρόβλεψη χρονοσειρών. [23]

- **Μηχανισμός προσοχής (Attention Mechanism) και Μετασχηματιστές (Transformers)**
  Ο μηχανισμός προσοχής εισήχθη ως λύση για την αντιμετώπιση των περιορισμών που αντιμετωπίζουν τα RNN κατά την επεξεργασία μακρών ακολουθιών εισόδου. Αντί να

συμπιέζει όλες τις προηγούμενες πληροφορίες σε μια κρυφή αναπαράσταση, ο μηχανισμός προσοχής υπολογίζει άμεσα τις εξαρτήσεις μεταξύ των εισόδων σε διαφορετικές θέσεις στην ακολουθία. Χωρίζει την είσοδο (tokens) και για την καθεμία δημιουργεί τρία διανύσματα, ενα διάνυσμα ερωτήματος(query) χ, το οποίο αντιπροσωπεύει τις πληροφορίες που αναζητά το token σε άλλα, ένα διάνυσμα κλειδιού(key) κ το οποίο αντιπροσωπεύει τις πληροφορίες του token που πιθανώς αναζητούν οι άλλες και ένα αξίας(value) που περιέχει την πραγματική πληροφορία. Βρισκοντας την ομοιότητα μεταξύ των κ και χ και με βάση αυτή δημιουργεί ένα σταθμισμένο άθροισμα απο διανύσματα πετυχαίνοντας παράλληλη επιλεκτική εστίαση για κάθε μέρος της εισόδου σε κάθε άλλο, αυξάνοντας την ταχύτητα και αποδοτικότητα σε σχέση με τα αναδρομικά νευρωνικά δίκτυα. [24]

Οι μετασχηματιστές είναι αρχιτεκτονικές που αποτελούνται απο ακολουθία απο μπλοκς τα οποία αποτελούνται από ένα μηχανισμό προσοχής και ένα προωθητικό νευρωνικό δίκτυο, το καθένα ακολουθούμενο απο μια αναδρομική σύνδεση με την είσοδο του(skip-connection) και μια συνάρτηση κανονικοποίησης επιπέδου (Layer Normalization). Η αρχιτεκτονική αυτη προσφέρει σταθερή, παράλληλη και εύκολα επεκτασιμη βαση για τις πρόσφατες επιτυχίες στην επεξεργασία φυσικής γλώσσας και στην όραση υπολογιστών μέσω διαφόρων παραλλαγών του όπως τα BERT, ViT (Μετασχηματιστές Ορασης) και GPT. [5]

- **Μετασχηματιστές όρασης (Vision Transformers, ViT)**
  Ο μετασχηματιστής όρασης είναι μια επέκταση της αρχιτεκτονικής του μετασχηματιστή για αναγνώριση εικόνων, η οποία ερμηνεύει τις εικόνες ως ακολουθίες tokens αντί να βασίζεται σε συνελιξεις. Η εικόνα εισόδου χωρίζεται σε ένα σε κομμάτια (patches) ίδιου μεγέθους, συνήθως 16x16 τα οποία προβάλλονται σε μορφή που τα επιτρέπει να αντιμετωπίζονται αναλογα με τα τις λέξεις στην επεξεργασία φυσικής γλώσσας. Έτσι μπορούν και κοντινές και μακρινές σχέσεις σε διάφορα κομμάτια της εικόνας να ανακαλυφθούν παράλληλα και μέσω της προσθήκης ενός ειδικού token, του [CLS] token, η οποία συγκεντρώνει τις συνολικές πληροφορίες της εικόνας, επιλέγει την τελική πρόβλεψη. Ο μετασχηματιστής όρασης απέδειξε ότι, όταν εκπαιδεύεται σε επαρκώς μεγάλα σύνολα δεδομένων, όπως το ImageNet-21k ή το JFT-300M, μπορεί να επιτύχει απόδοση συγκρίσιμη ή και ανώτερη από αυτή των υπερσύγχρονων συνελικτικών νευρωνικών δικτύων. [25]

Εκπαίδευση

- **Συναρτήσεις κόστους**
  Μια συνάρτηση κόστους αποτελεί ένα μέτρο που ποσοτικοποιεί το σφάλμα ενός μοντέλου και λειτουργεί ως οδηγός για την εύρεση των βέλτιστων παραμέτρων του μοντέλου, ώστε να ελαχιστοποιηθεί το εν λόγω σφάλμα. Στην εποπτευόμενη μάθηση, μετρά τη διαφορά μεταξύ της πρόβλεψης του μοντέλου $\hat{y}$ και της πραγματικής ετικέτας y, $L(y, \hat{y})$ και υπολογίζοντας την κλισης (gradient) της δημιουργείται ένα σήμα που ενημερώνει τις παραμέτρους του μοντέλου σε μια κατεύθυνση που μειώνει την εν

λόγω τιμή, καθοδηγώντας ιδανικά το μοντέλο προς μια βέλτιστη κατάσταση μηδενικού κόστους. Διαφορετικές συναρτήσεις κόστους χρησιμοποιούνται για διαφορετικές εργασίες, με τα προβλήματα παλινδρόμησης να χρησιμοποιούν συνήθως το μέσο τετραγωνικό σφάλμα ή το μέσο απόλυτο σφάλμα , τα οποία τιμωρούν τις μεγάλες αποκλίσεις από την τιμή-στόχο, ενώ οι εργασίες ταξινόμησης χρησιμοποιούν συνήθως την απώλεια διασταυρούμενης εντροπίας (Cross-Entropy loss) , η οποία ενθαρρύνει την υψηλή εμπιστοσύνη στην σωστή κατηγορία. [26]

- **Βελτιστοποίηση**
  Η βελτιστοποίηση είναι η επαναληπτική διαδικασία αλλαγής των παραμέτρων του δικτύου με σκοπό τον εντοπισμό του βέλτιστου συνόλου τιμών που ελαχιστοποιούν την επιλεγμένη συνάρτηση απώλειας. Ο βασικός αλγόριθμος για αυτό τον σκοπό είναι ο gradient descent που χρησιμοποιεί την κλίση της συνάρτησης κόστους του μοντέλου ως προς τα δεδομένα εκπαίδευσης για να βρει την κατεύθυνση προς την οποία πρέπει να αλλάξουν οι παράμετροι. Ωστόσο για καλύτερη απόδοση σύγχρονα νευρωνικά μοντέλα χρησιμοποιούν τον αλγόριθμο Stochastic Gradient Descent ή κάποιες από τις πιο σύγχρονες βελτιώσεις του, όπως momentum , Adam ή AdamW που έχουν την ίδια βασική ιδέα αλλα χρησιμοποιούν μικρά υποσύνολα των δεδομένων για να προσομοιώσουν την κλίση, αντί για όλα τα δεδομένα. [27]

0.2.2 Επεξηματικές Μετρικές

Λόγω της ραγδαίας αύξησης του μεγέθους, του βάθους και της πολυπλοκότητας των μοντέλων μηχανικής μάθησης, η κατανόηση του τρόπου με τον οποίο ένα μοντέλο λαμβάνει την τελική του απόφαση έχει γίνει πολύ δύσκολη. Ιδιαίτερα στη βαθιά μάθηση τα μοντέλα, όπου τα μοντέλα συνήθως λειτουργούν ως συστήματα «μαύρου κουτιού» των οποίων η διαδικασία λήψης αποφάσεων δεν είναι ερμηνεύσιμη λόγω της βαθιάς, πολυδιάστατης και μη γραμμικής φύσης τους. [11] Καθώς η κατανόηση αυτών των μοντέλων μπορεί να είναι κρίσιμη, ειδικά όταν εφαρμόζονται σε κρίσιμους τομείς όπως η υγειονομική περίθαλψη, τα αυτόνομα οχήματα ή η ποινική δικαιοσύνη, όπου η εμπιστοσύνη, η διαφάνεια, η δικαιοσύνη, η ανθεκτικότητα και η κανονιστική συμμόρφωση είναι ουσιαστικής σημασίας, έχουν αναπτυχθεί πολλές μέθοδοι για την επίλυση αυτού του ζητήματος. Αυτές οι μέθοδοι έχουν ομαδοποιηθεί υπό τον όρο Explainability Metrics στο γενικό πεδίο της Explainable Artificial Intelligence (XAI). [28, 29]

Ένα μεγάλο υποσύνολο αυτών των μεθόδων μπορεί να περιγραφεί τυπικά ως μέθοδοι απόδοσης (Attribution methods), οι οποίες στοχεύουν στην απονομή βαθμολογιών σημασίας σε στοιχεία ενός μοντέλου όπως δεδομένα εισόδου, αναπαραστάσεις δεδομένων, νευρώνες ή επίπεδα (layers) ως προς την συμβολή του στην τελική πρόβλεψη με τρόπο που οι άνθρωποι μπορούν να ερμηνεύσουν. Ο στόχος αυτών των μετρικών δεν είναι αποκλειστικά η ασφάλεια και ευθύνη αλλά και η επιστημονική γνώση, ο εντοπισμός σφαλμάτων και η βελτίωση των μοντέλων. [30] Στις ακόλουθες παραγράφους θα παρουσιαστούν οι κύριες κατηγορίες μεθόδων απόδοσης.

Μέθοδοι απόκρυψης (Occlusion based Methods) Οι μέθοδοι απόκρυψης στοχεύουν στον εντοπισμό σημαντικών τμημάτων/περιοχών της εισόδου, τροποποιώντας τα ή αφαιρώντας τα και μετρώντας την αλλαγή στην πρόβλεψη του μοντέλου. Η βασική ιδέα είναι ότι αν η αφαίρεση ενός τμήματος προκαλέσει μεγάλη αλλαγή, τότε το τμήμα είναι σημαντικό και προσφέρουν απόδοση σημασίας ανεξάρτητη από την αρχιτεκτονική του μοντέλου. Μερικές από τις πιο σημαντικές μεθόδους είναι η ευαισθησία απόκλεισης (Occlusion sensitivity) που «σέρνει» μια τετράγωνη μάσκα πάνω στην αρχική είσοδο και υπολογίζει πόσο κάθε τμήμα άλλαξε την εμπιστοσύνη στην κατηγορία-στόχο ή το RISE που χρησιμοποιεί πολλαπλές τυχαίες μάσκες και δημιουργεί έναν σταθμισμένο μέσο όρο τους για να δει πόσο κάθε εικονοστοιχείο επηρέασε την τελική πρόβλεψη. [22, 31]

Μέθοδοι βασισμένοι στην κλίση (Gradient based Methods) Οι επεξηγηματικές μετρικές με βάση την κλίση έχουν ως βασική ιδέα πως υψηλές τιμές κλίσης για ένα συγκεκριμένο χαρακτηριστικό του μοντέλου υποδηλώνουν ότι ελάχιστες αλλαγές σε αυτό το χαρακτηριστικό οδηγούν σε μεγάλη αλλαγή του τελικού αποτελέσματος, κάτι που καθιστά το χαρακτηριστικό αυτό σημαντικό. Είναι πιο αποτελεσματικά αποδοτικές μέθοδοι, καθώς οι αρχιτεκτονικές βαθιάς μάθησης έχουν βελτιστοποιήσει την αποδοτικότητα υπολογισμού της κλίσης· ωστόσο απλές μετρικές της κατηγορίας όπως η χρήση μόνο της κλίσης (vanilla gradients) ή το γινόμενο κλίσης και εισόδου (input x gradient) αντιμετωπίζουν προβλήματα λόγω μεγάλου θορύβου, κορεσμού και έλλειψης τοπικής ευαισθησίας. Ως αποτέλεσμα πολλές παραλλαγές και βελτιώσεις έχουν προταθεί για να αντιμετωπίσουν αυτά τα προβλήματα και στη συνέχεια θα παρουσιάσουμε δύο από αυτές που είναι κομβικές σε αυτή τη δουλειά. [32, 33, 34]

- **Ολοκληρωμένες κλίσεις (Integrated Gradients, IGs)** Προτάθηκαν ως λύση σε ορισμένους βασικούς περιορισμούς των απλών κλίσεων, όπως η ευαισθησία τους στον κορεσμό της κλίσης σε ορισμένα σημεία του δικτύου. Αντί για μία μόνο κλίση, οι IGs υπολογίζουν την επιρροή του κάθε κομματιού της εισόδου συσσωρεύοντας πληροφορίες κλίσης κατά μήκος μιας γραμμικής συνεχούς διαδρομής από ένα σημείο αναφοράς (baseline) x', συνήθως μια μαύρη εικόνα, έως την πραγματική είσοδο x. Αυτή η αλλαγή μετριάζει τα προβλήματα κορεσμού, αποφεύγοντας «επίπεδα» σημεία που προκαλούνται από ενεργοποιήσεις ReLU και θα αποτελούσαν πρόβλημα όταν χρησιμοποιείται μία μόνο κλίση. Αυτό έκανε τις IGs μια ισχυρή και αποτελεσματική και αποτέλεσαν βάση για πολλαπλές μελλοντικές παραλλαγές που πρότειναν πιο αποτελεσματικές βάσεις, διαδρομές ή ειδικές παραλλαγές για συγκεκριμένες αρχιτεκτονικές. [35]

  Η μετρική IG για το χαρακτηριστικό i είναι:

$$\mathrm{IG}_i(F, x; x') \;=\; (x_i - x_i') \int_0^1 \frac{\partial F(x' + a\,(x - x'))}{\partial x_i}\, da.$$

- **Αγωγιμότητα (Conductance)** Η αγωγιμότητα αποτελεί επέκταση των ιδεών των IGs, για να αποδοθεί η πρόβλεψη του μοντέλου στους νευρώνες του αντί για την είσοδο. Μπορεί να ερμηνευθεί ως μέτρηση της «ροής» της απόδοσης που διέρχεται από έναν

νευρώνα κατά τον υπολογισμό των IGs και υπολογίζεται μέσω αποσύνθεσης του υπολογισμού των IGs με τον κανόνα της αλυσίδας για να αποδώσει σε όλα τα επίπεδα του δικτύου μια τιμή σε κάθε νευρώνα που αντιπροσωπεύει τη συμβολή του. [12]

Για νευρώνα k στο επίπεδο l, η αγωγιμότητα του *για χαρακτηριστικό* i είναι:

$$Y^{(l)}_{k,i}(F, x; x') \;=\; (x_i - x'_i) \int_0^1 \frac{\partial F(x' + a\,(x - x'))}{\partial y^{(l)}_k} \frac{\partial y^{(l)}_k}{\partial x_i}\, da, \tag{1}$$

όπου $y^{(l)}_k$ είναι η έξοδος του νευρώνα k στο επίπεδο l για είσοδο $x + a\,(x - x')$ και $Y^{(l)}_{k,i}(F, x; x')$ είναι η αγωγιμότητά του σε σχέση με το χαρακτηριστικό i.

Οι τιμές αγωγιμότητας μπορούν να είναι θετικές ή αρνητικές, με τις θετικές τιμές να υποδηλώνουν ότι ένας νευρώνας συμβάλλει θετικά στην πρόβλεψη της εξόδου, τις αρνητικές τιμές να υποδηλώνουν αρνητική συμβολή και τις τιμές κοντά στο μηδέν να υποδηλώνουν ελάχιστη επίδραση.

Τόσο οι IGs όσο και η αγωγιμότητα, για λόγους πρακτικότητας, υπολογίζονται με χρήση μιας προσεγγιστικής μεθόδου, που αντικατέστησε το ολοκλήρωμα με άθροισμα, με συγκεκριμένο αριθμό διακριτών βημάτων n_c ανάμεσα στη βάση x' και την είσοδο x.

Μέθοδοι απόδοσης που βασίζονται στη διάδοση (Propagation-Based Methods) Οι μέθοδοι απόδοσης που βασίζονται στη διάδοση ή στην αποσύνθεση λειτουργούν αναδιανέμοντας τη σημασία μέσω του δικτύου από την έξοδο έως την αρχή, αποδίδοντας σε κάθε νευρώνα μια βαθμολογία ως προς τη συνεισφορά του στο αποτέλεσμα. Για κάθε αρχιτεκτονική χρησιμοποιούν προσαρμοσμένους κανόνες για τη διάδοση από κάθε επίπεδο στο προηγούμενο για να προσομοιώσουν τον τρόπο με τον οποίο οι πληροφορίες ρέουν μέσω του δικτύου αντί να βασίζονται στους τυπικούς κανόνες υπολογισμού της κλίσης. Η δημοτικότητα τέτοιων μεθόδων οφείλεται στους προσαρμοσμένους κανόνες διάδοσης που μπορούν να αποφύγουν προβλήματα που μαστίζουν τις μεθόδους που βασίζονται στην κλίση, όπως ο κορεσμός. Μερικές από τις πιο δημοφιλείς τεχνικές είναι η Layer-wise Relevance Propagation (LRP) που θέτει διάφορους κανόνες για τη διάδοση της πρόβλεψης του μοντέλου από επίπεδο σε επίπεδο, όπου όλοι συμμορφώνονται με ορισμένους θεωρητικούς κανόνες, και το DeepLiFT που χρησιμοποιεί κανόνες ανάλογους της διάδοσης της κλίσης στο δίκτυο αλλά σε κάθε επίπεδο συγκρίνει τη διαφορά της εξόδου για την είσοδο x με την έξοδο για μία βάση x'. [36, 37, 34]

0.2.3 Γενίκευση και κανονικοποίηση

Κάτα την εκπαίδευση ενός μοντέλου μηχανικής μάθησης ο βασικός στόχος είναι το μοντέλο να έχει καλή επίδοση σε νέα δεδομένα όχι μόνο σε αυτά στα οποία εκπαιδεύτηκε. Αυτη η ικανότητα ονομάζεται γενίκευση και δείχνει ότι το μοντέλο έμαθε πραγματικά μοτίβα για τον τύπο δεδομένων και τον στόχο εκπαίδευσης και δεν απομνημόνευσε απλά τα δείγματα εκπαίδευσης. Είναι συνηθισμένο πρόβλημα ιδιαίτερα σε μεγάλα και πολύπλοκα μοντέλα που λόγω του μεγάλου αριθμού παραμέτρων μπορούν με λεπτομέρεια να προσεγγίσουν θορυβώδη μοτίβα που υπάρχουν στα σύνολα εκπαίδευσης. Για να αντιμετωπιστεί

αυτό πολλές τεχνικές έχουν προταθεί για να περιορίσουν και να κατευθύνουν την εκπαίδευση του μοντέλου σε μια τελική κατάσταση που δίνει καλύτερα αποτελέσματα με νέα δεδομένα. Αυτές οι τεχνικές ονομάζοντα τεχνικές κανονικοποίησης και στις επόμενες παραγράφους θα αναλύσουμε μερικές απο τις πιο συνήθεις κατηγορίες τεχνικών γενίκευσης και κανονικοποίησης.

Κανονικοποίηση μέσω συνάρτησης κόστους (Loss-based regularization) Τεχνικές κοινωνικοποίησης μέσω συνάρτησης κόστους αλλάζουν την συνάρτηση κόστους ή προσθέτουν επιπλέον όρους με σκοπό να τροποποιήσουν τον υπολογισμό της και να αποθαρρύνουν από ορισμένες συμπεριφορές και να την οδηγήσουν σε μια πιο σταθερή και γενικευμένη τελική κατασταση. Παραδείγματα τέτοιων τεχνικών είναι η L1 και L2 κανονικοποίσηση που προσθέτουν το άθροισμα της 1ης ή της 2ης νόρμας όλων των παραμέτρων αντίστοιχα για να ενθαρρύνουν μικρές τιμές ή Label smoothing που αλλάζει τον υπολογισμό της Cross entropy συνάρτησης, δίνοντας μικρές πιθανότητες στις άλλες ετικέτες για να μην έχει το μοντέλο υπερβολική αυτοπεποίθηση στις προβλέψεις του [38, 39, 40].

Τεχνικές ομαλοποίησης (Normalization techniques) Οι τεχνικές ομαλοποίησης χρησιμοποιούνται για να βελτιώσουν την διαδικασία της εκπαίδευσης, μειώνοντας την αλλαγή στην κατανομή των ενεργοποιήσεων. Τεχνικές όπως batch normalization και layer normalization που κανονικοποιούν τα δεδομένα με βάση την μέση τιμή και διασπορά μεταξύ των δεδομένων στο batch και μεταξύ των χαρακτηριστικών του στοιχείου αντίστοιχα, επιτρέπουν ταχύτερες και πιο σταθερές διαδικασίες βελτιστοποίησης [41, 42].

Επαύξηση Δεδομένων (Data Augmentation) Οι τεχνικές επαύξησης δεδομένων, δημιουργούν νέα τεχνητά και ποικιλόμορφα δεδομένα χβρίς την ανάγκη συλλογής νέων πραγματικών δειγμάτων. Εφαρμόζεται ευρέως στην όραση υπολογιστών όπου τα δεδομένα είναι ευαίσθητα σε μικρές αλλαγές στην κλίμακα, τη γεωμετρία, το χρώμα, τη δομή κ.λ.π. που δίχως κατάλληλη ποικιλία σε παραλλαγές τους κατα την εκπαίδευση, το μοντέλο θα απομνημονευσει τα περιορισμένα χαρακτηριστικά του συνόλου εκπαίδευση και θα παρουσιάσει κακή γενίκευση. Τεχνικές όπως το Autoaugment και RandAugment προτείνουν πολιτικές για δημιουργία δεδομένων μέσω συνδυασμών πολλαπλών μετασχηματισμών όπως η περιστροφή, η αναστροφή, η κλιμάκωση,η ρύθμιση της φωτεινότητας κλπ σε εικόνες του συνόλου δεδομένων κρατώντας σταθερή την τελική ετικέτα για να δώσουν εναλλακτικές εκδοχές των δεδομένων στο μοντελο. Άλλες προσεγγίσεις όπως το Mixup και Cutmix δημιουργούν αναμείξεις εικόνων με διαφορετικές ετικέτες για να λυάνουν τα σύνορα μεταξύ των διαφορετικών ετικετών [43, 44, 45, 46, 47].

Τεχνικές αφαίρεσης (drop-based techniques) Οι τεχνικές αφαιρεσης προσθέτουν τυχαιότητα στην διαδικασία εκπαίδευσης, απενεργοποιώντας τυχαία νευρώνες, συνδέσεις ή ολόκληρα επίπεδα ενός νευρωνικού δικτύου. Αυτο μειώνει την αμοιβαία προσαρμογή (co-adaptation) χαρακτηριστικών μέσα στο δίκτυο, οδηγώντας σε πιο αποδοτική αξιοποίηση των παραμέτρων του δικτύου και λιγότερη εκμάθηση θορυβωδών μοτίβων του συνόλου εκπαίδευσης άρα και καλύτερη γενίκευση. Αν και υπάρχουν πολλές διαφορετικές τεχνικές και

παραλλαγές μερικές απο τις πιο σημαντικές για αυτή την εργασία παρουσιάζονται στην συνέχεια:

- **Dropout**
  Το Dropout είναι μια τεχνική που μηδενίζει τυχαία τις τιμές ενός υποσυνόλου των τιμών της εξόδου ενος επιπέδου κατά τη διάρκεια της εκπαίδευσης, με κάθε τιμή να απορρίπτεται ανεξάρτητα. Αυτή η διαδικασία έχει ως αποτέλεσμα τη μείωση της αμοιβαίας προσαρμογής των χαρακτηριστικών μεταξύ των νευρώνων και αναγκάζει το δίκτυο να κατανέμει τις πληροφορίες σε πολλά μονοπάτια αντι να βασίζεται αποκλειστικά σε ένα μικρό υποσύνολο. Ως αποτέλεσμα το μοντέλο αξιοποιεί ουσιαστικα περισσότερες από τις παραμέτρους του και παρουσιάζει ενισχυμένη γενίκευση [48].

- **Drop-Path (Stochastic Depth)**
  Το Drop-Path, επίσης γνωστό ως Stochastic Depth, επεκτείνει τις ιδέες του dropout απο μεμονωμένες τιμές σε αφαίρεση ολόκληρων επιπέδων. Στόχος της τακτικής είναι κατα την διάρκεια της εκπαίδευσης να εκπαιδεύονται τυχαία υπο-δίκτυα μικρότερου βάρους, επιτρέποντας πιο σταθερή εκπαίδευση σε βαθύτερες αρχιτεκτονικές κρατώντας και την ομαλότερη εκπαίδευση πιο ρηχών αρχιτεκτονικών αλλα και την αποδοτικότητα του βαθύτερου μοντέλου στο τέλος της εκπαίδευσης [49].

- **Y-drop**
  Το Y-drop είναι μια παραλλαγή του dropout που χρησιμοποιεί την μετρική της αγωγιμότητας για να προσαρμόζει δυναμικά την πιθανότητα dropout ανά νευρώνα, ανάλογα με τη σημαντικότητά του.

  Σε κάθε βήμα εκπαίδευσης, πριν από την κλασική ενημέρωση των παραμέτρων, χρησιμοποιείται ένα υποσύνολο B_c δείγματα χρησιμοποιείται για τον υπολογισμο της της μέσης αγωγιμότητας των νευρώνων σε ένα επίπεδο. Με βάση αυτή την τιμή χωρίζονται σε 2 ομάδες,τους $w_w\%$ με τις χαμηλότερες τιμές που χαρακτηρίζονται ως αδύναμοι, ενώ οι υπόλοιποι $w_s\%$ στην άλλη ομάδα χαρακτηρίζονται ως ισχυροί. Για κάθε νευρώνα, η πιθανότητα dropout δειγματοληπτείται από το δεξί μισο μιας κανονική κατανομή με παραμέτρους (μ_L, σ_L) για τους αδύναμους και (μ_H, σ_H) για τους ισχυρούς, με $\mu_H > \mu_L$. Επιπλέον, υπολογιζεται κινητός μέσος όρος ανά νευρώνα για την για την αναπροσαρμογή των τιμών του επιπέδου μετα το dropout, χρησιμοποιώντας την υπερ παράμετρο elasticity a και η μέθοδος ενεργοποιείται μετά από το αρχικό ποσοστό K των εποχών, annealing factor, καθώς στις πρώτες εποχές οι εκτιμήσεις της αγωγιμότητας είναι λιγότερο ουσιώδεις και χρησιμοποιείται απλό dropout.

  Η μέθοδος εφαρμόστηκε στα επίπεδα του προωθητικού κομματιου (feed-forward) σε διάφορες αρχιτεκτονικές συνελικτικών δικτύων οδήγησε σε μεγαλύτερη συμμετοχή περισσότερων νευρώνων στην τελική πρόβλεψη, ισχυρότερη κανονικοποίηση και βελτιωμένη επίδοση σε σχέση με το απλό dropout [8].

0.3 Generalized Y-drop

Σε αυτή την ενότητα παρουσιάζουμε τη συνεισφορά μας στη βελτίωση της τακτικής Y-drop. Αρχικά, συνοψίζουμε τα βασικά μειονεκτήματα που περιορίζουν την πρακτική εφαρμογή της και επιλογές που θεωρούμε είναι μη βέλτιστες. Στη συνέχεια, περιγράφουμε τις προτεινόμενες τροποποιήσεις μας για την αντιμετώπιση αυτών των αδυναμιών και εισάγουμε το Generalized Y-drop, ένα πιο πρακτικό και ευέλικτο δυναμικό σχήμα dropout.

0.3.1 Προβλήματα με το Y-drop

Ρύθμιση παραμέτρων: Ένα από τα σημαντικότερα μειονεκτήματα είναι ο μεγάλος αριθμός επιπλέον υπερ-παραμέτρων σε σχέση με το κλασικό dropout. Αντί για μία μόνο πιθανότητα, απαιτείται ρύθμιση παραμέτρων που σχετίζονται με τον διαχωρισμό σε δύο ομάδες (μέγεθος ομάδων, μέσες τιμές, τυπικές αποκλίσεις και μέγιστες τιμές ανά ομάδα). Η πρόσθετη αυτή πολυπλοκότητα κανει τη διαδικασία εύρεσης κατάλληλων ρυθμίσεων δύσκολη και δαπανηρή, ιδίως κατά τη μεταφορά της μεθόδου σε νέες αρχιτεκτονικές όπου ενδεχομένως πρέπει να βρεθούν νέες ισορροπίες.

Χρηση πληροφορίας αγωγιμότητας: Ο απλος διαχωρισμός των νευρώνων σε δύο ομάδες δεν αποτυπώνει τις σχετικές διαφορές μεταξύ των τιμών αγωγιμότητας. Έτσι, νευρώνες με οριακά μεγαλύτερη αγωγιμότητα από γειτονικούς μπορεί να κατατάσσονται σε διαφορετική ομάδα και να λαμβάνουν δυσανάλογα διαφορετικές πιθανότητες dropout. Μια πιο ομαλή ανάθεση, πιθανώς βασισμένη στη σχετική σημαντικότητα, θα μπορούσε να οδηγήσει σε καλύτερα αποτελέσματα.

Σύγκριση με άλλες μεθόδους: Λόγω πρόσθετου θορύβου και μηχανισμών περικοπής (clipping), η μέση πιθανότητα dropout στο Y-drop δεν είναι σταθερή ούτε εύκολα ελεγχόμενη. Αυτό δυσχεραίνει τόσο τη ρύθμιση της μεθόδου, ειδικά σε αρχιτεκτονικές όπως οι μετασχηματιστές που συχνά απαιτούν μικρότερες τιμές dropout, όσο και τη δίκαιη σύγκριση με άλλες τεχνικές, καθώς τα κέρδη ενδέχεται να οφείλονται απλώς σε αύξηση της μέσης πιθανότητας dropout.

Ευθυγράμμιση με τον στόχο βελτιστοποίησης: Η μετρικη της αγωγιμότητας περιγράφει την συμβολή κάθε νευρώνα στην τελική πρόβλεψη, ωστόσο δεν δείχνει κατά πόσο αυτή η συμβολή βοήθησε στην επίτευξη του πραγματικού στόχου. Ένας νευρώνας μπορεί να επηρεάζει έντονα την έξοδο, αλλά προς λανθασμένη κατεύθυνση· σε αυτή την περίπτωση, η «αφαίρεσή» του μέσω αυξημένου dropout μπορεί να μην είναι επιθυμητή. Συνεπώς μια εναλλακτική μετρική που αντικατοπτρίζει αυτή την διαφορά θα μπορούσε να αποτελεί πιο κατάλληλη ένδειξη σημαντικότητας.

Υπολογιστικό κόστος: Το πιο ουσιαστικό πρακτικό μειονέκτημα είναι το αυξημένο υπολογιστικό κόστος, καθώς απαιτείται ο υπολογισμός της αγωγιμότητας σε κάθε βήμα εκπαίδευσης. Παρότι ένα πρόσθετο κόστος είναι αναπόφευκτο δεδομένου ότι ο υπολογισμός της μετρικής είναι απαραίτητος, είναι σημαντικό να αναπτυχθούν παραλλαγές που κάνουν την μέθοδο φθηνότερη.

0.3.2 Προτεινόμενες μέθοδοι

Εναλλακτικές τακτικές ανάθεσης πιθανότητας dropout Για την αντιμετώπιση των περιορισμών της αρχικής τεχνικής, συγκεκριμένα του μεγάλου αριθμού υπερ-παραμέτρων, της ελλιπούς αξιοποίησης της πληροφορίας της αγωγιμότητας και της δυσκολίας σύγκρισης με άλλες τεχνικές προτείνουμε εναλλακτικές στρατηγικές ανάθεσης πιθανοτήτων dropout. Η συνολική διαδικασία της μεθόδου παραμένει ουσιαστικά ίδια, ωστόσο αντικαθιστούμε τον διακριτό χωρισμό σε ομάδες και τις αντίστοιχες υπερ παραμέτρους με δύο μόνο υπερ-παραμέτρους, μία μέση πιθανότητα *dropout* p_{mean}, γύρω από την οποία κανονικοποιούνται οι τιμές, και μία μέγιστη τιμή p_{max}, ώστε να αποφεύγονται υπερβολικά μεγάλες πιθανότητες.

Στόχος είναι η μέση πιθανότητα *δροπουτ* των νευρώνων ενός επιπέδου να ισούται με p_{mean}, χωρίς καμία τιμή να υπερβαίνει το p_{max}, ενώ οι νευρώνες με υψηλότερη αγωγιμότητα λαμβάνουν μεγαλύτερη πιθανότητα *δροπουτ* σε σχέση με νευρώνες χαμηλότερης αγωγιμότητας. Προτείνουμε δύο στρατηγικές που ικανοποιούν τα παραπάνω, ώστε να αξιολογηθεί ο βαθμός στον οποίο η ανάθεση πρέπει να εξαρτάται από την κατανομή των τιμών. Τα Ranking Rates, που βασίζονται αποκλειστικά στην κατάταξη (ιεραρχία) των τιμών εντός κάθε επιπέδου, και Relative Rates, που λαμβάνουν υπόψη τη σχετική διαφορά μεταξύ των τιμών.

Επιπλέον, για κάθε στρατηγική εξετάζουμε ένα προαιρετικό βήμα: στα Ranking Rates την ύψωση των πιθανοτήτων σε δύναμη, ώστε να διερευνηθεί αν η εντονότερη διαφοροποίηση στα άκρα βελτιώνει την επίδοση, και στα Relative Rates την περικοπή ακραίων τιμών, ώστε η ανάθεση να είναι πιο ανθεκτική σε πιθανά προβλήματα με έκτοπες τιμες (outliers).

Με περισσότερη λεπτομέρεια οι μέθοδοι είναι οι εξής:

Ranking Rates

- **Κατάταξη των νευρώνων:**
  Για καθε επίπεδο l, οι νευρώνες κατατάσσονται με βάση τις μέσες τιμές αγωγιμότητας ανά μονάδα $Y^{(l)}$
  $$r^{(l)} = \text{rank}(Y^{(l)})$$
  οπου $r \in [0, N^{(l)}]$ με $N^{(l)}$ να είναι ο αριθμός των νευρώνων στο επίπεδο $N^{(l)}$.

- **Μετατροπη σε πιθανότητα με ύψωση σε δύναμη:** Οι κατατάξεις μετατρέπονται σε προκαταρκτικές πιθανότητες *dropout* χρησιμοποιώντας μια μετατροπή βασισμένη σε δύναμη:
  $$p'^{(l)} = p_{\text{max}} \cdot \left(\frac{r^{(l)} - 0.5}{N^{(l)}} \right)^{\text{power}}$$
  Στις περισσότερες πειραματικές ρυθμίσεις η παράμετρος ποωερ είναι ίση με 1 εκτός αν αναφέρεται ρητά διαφορετικά. Ως παράμετρος ελέγχει την οξύτητα της κατανομής δροπουτ, με μεγαλύτερες τιμές να τονίζουν τη διαφορά μεταξύ των κορυφαίων και των κατώτερων νευρώνων, ενώ μικρότερες τιμές παράγουν μια πιο ομαλή κατανομή.

- **Κεντράρισμα γύρω απο την μέση τιμή:**
  Οι τελικές πιθανότητες κεντράρονται γύρω από την μέση πιθανότητα δροπουτ $p_{\text{μεαν}}$, με έναν παράγοντα κλιμάκωσης τ, ο οποίος υπολογίζεται με δυαδική αναζήτηση, εξασφα-

λίζοντας ότι όλες οι τιμές παραμένουν εντός του έγκυρου εύρους:

$$p^{(l)} = p_{\text{mean}} + \tau \cdot (p'^{(l)} - \text{mean}(p'^{(l)})), \qquad \tau = \max \text{ s.t. } p^{(l)} \in [0, p_{\max}]$$

Relative Rates

- **Ελεγχός έκτοπων τιμών (outliers) (Προεραιτικό):** Καθώς αυτή η μέθοδος θα μπορούσε να είναι πιο ευάλωτη σε έκτοπες τιμές που οδηγούν σε ορισμένους νευρώνες να λαμβάνουν εξαιρετικά χαμηλές ή υψηλές τιμές δροπουτ, πραγματοποιήθηκαν ορισμένες δοκιμές όπου οι κορυφαίες και οι κατώτερες x% των τιμών Αγωγιμότητας περικόβονται. Αυτό το βήμα δεν εκτελείται στην βασική ρύθμιση και θα αναφέρεται ρητά όταν χρησιμοποιείται στα επόμενα αποτελέσματα.

- **Κανονικοποίηση και κλιμάκωση:** Οι τιμές αγωγιμότητας $Y^{(l)}$ κανονικοποιούνται χρησιμοποιώντας χρησιμοποίωντας min–max μετατροπή και επανακλιμακώνονται στο διάστημα $[0, p_{\max}]$

$$p'^{(l)} = p_{\max} \cdot \frac{Y^{(l)} - \min(Y^{(l)})}{\max(Y^{(l)}) - \min(Y^{(l)})}$$

- **Κεντράρισμα γύρω απο την μέση τιμή:** Παρόμοια με τις τιμές κατάταξης, οι πιθανότητες κεντράρονται γύρω από την μέση πιθανότητα dropout p_{mean} , με έναν παράγοντα κλιμάκωσης τ που τις διατηρεί εντός του έγκυρου εύρους:

$$p^{(l)} = p_{\text{mean}} + \tau \cdot (p'^{(l)} - \text{mean}(p'^{(l)})), \qquad \tau = \max \text{ s.t. } p^{(l)} \in [0, p_{\max}]$$

Αγωγιμότητα κόστους (Loss Conductance) Προτείνουμε μια εναλλακτική διατύπωση της αγωγιμότητας, η οποία, αντί να αποτιμά τη συνεισφορά ενός νευρώνα στην τελική πρόβλεψη, αποτιμά τη συνεισφορά του στην τελική τιμή της συνάρτησης κόστους. Τη μετρική αυτή την ονομάζουμε Αγωγιμότητα Κόστους (Loss Conductance) και στόχος της είναι να παρέχει ένα σήμα σημαντικότητας άμεσα ευθυγραμμισμένο με τον στόχο βελτιστοποίησης.

Ο τύπος της νέας μετρικής δίνεται ως εξής για όλα τα χαρακτηριστικά εισόδου i οπου $\mathcal{L}$ είναι η συνάρτηση κόστους:

$$\Upsilon_k^{(l)}(F, x; x') \approx \sum_{j=1}^{n_c} \frac{\partial \mathcal{L}(F(x^{(j)}), t)}{\partial k} \left(F_k^{(l)}(x^{(j)}) - F_k^{(l)}(x^{(j-1)}) \right). \tag{2}$$

όπου $x^{(j)} = x' + \frac{j}{n_c}(x - x')$.

Με τη συγκεκριμένη αναδιατύπωση η σημασία των τιμών αλλάζει σε θετική αγωγιμότητα κόστους να υποδηλώνει ότι ο νευρώνας συνέβαλε σε αύξηση του κόστους, αρνητική αγωγιμότητα κόστους ότι συνέβαλε σε μείωση του κόστους, ενώ τιμές κοντά στο μηδέν υποδηλώνουν αμελητέα επίδραση στη συνάρτηση κόστους.

Συνεπώς, ως πιο «σημαντικοί» θεωρούνται οι νευρώνες με με χαμηλότερες (πιο αρνητικές) τιμές αγωγιμότητά κόστους, δηλαδή εκείνοι που μειώνουν περισσότερο το κόστος. Για τον λόγο αυτό, σε αυτούς αναθέτουμε υψηλότερες πιθανότητες dropout, επιδιώκοντας ισχυρότερη κανονικοποίηση.

Υπολογιστικά φθηνότερες παραλλαγές Για τη μείωση του υπολογιστικού κόστους της μεθόδου, διατηρώντας παράλληλα συγκρίσιμες βελτιώσεις στην απόδοση, εξετάσαμε παραλλαγές που απαιτούν λιγότερους υπολογισμούς της αγωγιμότητας κόστους. Η βασική υπόθεση είναι ότι το σήμα της μετρικής είναι πιο σημαντικό στα πρώτα στάδια της σε σχέση με τα τελευταία ή τουλάχιστον ο ρυθμός μεταβολής των τιμών της για κάθε νευρώνα μειώνεται, άρα και η ιεραρχία/σχετικές τιμές παραμένουν σχεδόν σταθερές και δεν χρειάζεται συχνή αλλαγή των πιθανοτήτων dropout.

Οι στρατηγικές που δοκιμάσαμε ήταν οι εξής:

- **Πρόωρος τερματισμός:** Μετά από ένα προκαθορισμένο ποσοστό εποχών, είτε επιστρέφουμε στο κλασικό dropout είτε διατηρούμε τις τελευταίες υπολογισμένες πιθανότητες dropout χωρίς περαιτέρω ανανεώσεις.

- **Αραιότερες ενημερώσεις:** Αντί να υπολογίζεται η αγωγιμότητά κόστους σε κάθε βήμα εκπαίδευσης, ο υπολογισμός πραγματοποιείται ανά διάστημα (*ιντεραλ*) x βημάτων. Εξετάζουμε τόσο την περίπτωση σταθερού διαστήματος καθ' όλη τη διάρκεια της εκπαίδευσης όσο και την περίπτωση προγραμματισμένης αύξησης του διαστήματος, όπου ξεκινά από 1 και αυξάνεται συνολικά n_{up} φορές κατά 1, σε ισομερή τμήματα της εκπαίδευσης.

Προσαρμογή σε Μετασχηματιστές Σε αυτη την εργασία δοκιμάσαμε την μέθοδο κυρίως σε μετασχηματιστές εικόνας. Στη συγκεκριμένη αρχιτεκτονική, το dropout εφαρμόζεται συνήθως στα attention weights, στο projection layer του μηχανισμού προσοχής και στα δύο γραμμικά επίπεδα του προωθητικού τμήματος (feed-forward). Ωστόσο, ο υπολογισμός μέσων τιμών αγωγιμότητας από πολλά δείγματα για στα attention weights δεν είναι ουσιώδης, καθώς αυτά εκφράζουν σχέσεις μεταξύ tokens και μεταβάλλονται έντονα ανά είσοδο, δίχως κάποια ουσιαστική ένδειξη σημασίας. Για τον λόγο αυτό, διατηρούμε απλό dropout στα attention weights και εφαρμόζουμε Generalized Y-drop μόνο στα υπόλοιπα τρία γραμμικά επίπεδα.

Επιπλέον η αρχιτεκτονικη αυτή έχει διαφορές που πρέπει να εξεταστούν:

- **Επιλογή token:** Για κάθε δειγμα έχουμε πολλαπλές τιμές για κάθε token. Ενω το αρχικό πλάνο ήταν να χρησιμοποιηθεί ό μέσος όρος του παρατηρήθηκε πως οι τιμές αγωγιμότητας του [CLS] token παρουσιάζει μια πολύ πλουσιότερη και πιο διαφοροποιημένη κατανομή αγωγιμότητας σε σχέση με τα άλλα tokens. Συνεπώς, χρησιμοποιούμε αποκλειστικά την τιμή του [CLS] για την ανάθεση πιθανοτήτων dropout, επιλογή που οδήγησε σε καλύτερα αποτελέσματα.

- **Layer Normalization:** Μια άλλη βασική διαφορά μεταξύ των αρχιτεκτονικών μετασχηματιστή είναι πως πολλά επίπεδα ακολουθούνται από αναδρομικες συνδέσεις (skip-connection) και λειτουργία Layer Normalization. Αρχικά, η αγωγιμότητα κόστους υπολογιζόταν μετά το Layer Normalization, ωστόσο πειραματικά διαπιστώθηκε ότι ο υπολογισμός πριν το Layer Normalization και πριν το skip connection αποδίδει σταθερότερα και καλύτερα αποτελέσματα.

Αυξανόμενο Generalized Y-drop Σε πειράματα με χρήση πολλών τεχνικών επαύξησης δεδομένων παρατηρήθηκε πως και το dropout και το Generalized Y-drop έκαναν την εκπαίδευση πολυ ασταθής και σε μεγαλύτερα μοντέλα οδηγούσε σε αποτυχία σύγκλισης. Ο βασικός λόγος ηταν οτι ο συνδιασμος dropout και επαυξησης δεδομένων δημιουργουσε πολύ θόρυβο στα αρχικά επίπεδα, ως αποτέλεσμα χρησιμοποιούμε Αυξανόμενο Generalized Y-drop, όπου η μέση πιθανότητα dropout είναι μηδενική στο πρώτο block του Transformer και αυξάνεται γραμμικά στα βαθύτερα blocks έως μια τελική μέγιστη τιμή. Αυτό μειώνει τον θόρυβο στα πρώιμα επίπεδα και έκανε την μέθοδ πιο ανταγωνιστικη σε τέτοιες συνθήκες εκπαίδευσης.

0.4 Πειραματική Αξιολόγηση

0.4.1 Ρυθμίσεις πειραμάτων

Σε αυτή την ενότητα περιγράφονται συνοπτικά οι ρυθμίσεις των πειραμάτων για το Generalized Y-drop, συμπεριλαμβανομένων των αρχιτεκτονικών, των συνόλων δεδομένων και των βασικών υπερ-παραμέτρων/τεχνικών εκπαίδευσης.

Αρχιτεκτονικές Τα πειράματα πραγματοποιήθηκαν σε συνελικτικά νευρωνικά δίκτυα (CNNs) και σε Μετασχηματιστές Όρασης (Vision Transformers).

Χρησιμοποιήθηκαν δύο CNN αρχιτεκτονικές,οι M1 και M2. Και οι 2 έχουν το ίδιο συνελικτικο κομμάτι με 3 συνελικτικα επίπεδα με 96,128 και 256 φίλτρα αντίστοιχα αλλά διαφέρουν το προωθητικό τμήμα με την M1 διαθέτει 2 πλήρως συνδεδεμένα επίπεδα με 2048 νευρώνες, ενώ η M2 διαθέτει 3 επίπεδα με 4096 νευρώνες. Dropout χρησιμοποιείται στα πλήρως συνδεδεμένα επίπεδα και το αντικαθιστουμε με Generalized Y-drop.

Τα πειράματα με μετασχηματιστές όρασης εγινα με 3 μοντέλα διαφορετικού μεγέθους τα ViT-Tiny, Vit-Small και ViT-Base με τις διαφορές τους να φαίνονται στον πίνακα 1. Επιπλέον, πραγματοποιήθηκαν πειράματα με DeiT μοντέλα,που έχουν την ίδια αρχιτεκτονική και μεγέθη με τα ViT, αλλά ενσωματώνουν label smoothing και πολλαπλες τεχνικές επαύξησης δεδομένων, όπως Repeated Augmentation, RandAugment, Mixup, CutMix και Random Erasing.

Πίνακας 1. *Μεγέθη των μοντέλων ïT που χρησιμοποιήθηκαν στα πειράματα.*

Μοντέλο	**ViT-Tiny**	**ViT-Small**	**ViT-Base**
Διάσταση Διανυσμάτων (Embedding)	192	384	768
Αριθμός κεφαλών (heads)	3	6	12
Αριθμός μπλοκ	12	12	12
Παράμετροι	5 Εκ.	22 Εκ.	86 Εκ.
Ανάλυση	224x224	224x224	224x224

Σύνολα Δεδομένων Χρησιμοποιούμε 3 σύνολα δεδομένων. Τα CIFAR-10 και CIFAR-100 που αποτελούνται από τις ίδιες 50000 εικόνες εκπαίδευσης και 10000 εικόνες αξιολόγησης,

μεγέθους 32×32 (στους μετασχηματιστες μετατρέπονται σε 224×224), οι οποίες χωρίζονται σε 10 και 100 ξεχωριστές ετικέτες αντίστοιχα. Επιπλέον χρησιμοποιούμε το Imagenet-1k που περιέχει 1.28 εκατομμυρια εικόνες εκπαίδευσης και 50000 αξιολόγησης χωρισμένες σε 1000 διαφορετικές ετικέτες.

Λεπτομέρειες Εκπαίδευσης

- **Αριθμός εκτελέσεων**
  Τα αποτελέσματα αναφέρονται ως μέσος όρος και τυπική απόκλιση της μέγιστης ακρίβειας σε πολλαπλές τυχαίες εκτελέσεις. Για CIFAR-10/100 πραγματοποιήθηκαν 5 εκτελέσεις, ενώ για ImageNet-1k 3.

- **Παράμετροι Generalized Y-drop / Dropout / Drop-Path**
  Σε όλα τα πειράματα εκτός αν ρητά αναφέρεται χρησιμοποιήθηκαν elasticity = 0.01, annealing factor K = 10%, 5 διακριτά βήματα και το $\frac{1}{5}$ τυχαίων δειγμάτων από κάθε batch εκπαίδευσης για τον υπολογισμό της αγωγιμότητας. (51 δείγματα οταν batch 256 και 204 δειγματα οταν batch 1024)

  Στα CNNs θέσαμε pmean = 0.35 και pmax = 0.7 ενω στα πειράματα με ViT/DeiT είχαμε pmean = 0.1 και pmax = 0.5.

  Στη σύγκριση με απλό dropout χρησιμοποιείται p=pmean ενω όταν χρησιμοποιούμε Drop-Path ώστε να προκύπτει συγκρίσιμη μέση ένταση regularization σε όλο το δίκτυο, τέθηκε μέγιστη πιθανότητα 0.2 (αφού στο πρώτο μπλοκ έχει μέση τιμή 0 και αυξάνεται γραμμικά μέχρι μία μέγιστη τιμή στο τελευταίο. Εξαίρεση αποτελούν τα τελικά πειράματα στο DeiT όπου χρησιμοποιείται αυξανόμενο Generalized Y-drop, που έχει ίδια μορφή με το Drop-Path, που και στα δύο θέτουμε μέγιστη τιμή 0.1, σύμφωνα με την βιβλιογραφία του DeiT.

- **Εκπαίδευση CNNs.**
  Τα μοντέλα εκπαιδεύτηκαν για 150 εποχές, batch size 256, σταθερό learning rate 0.001.

- **Εκπαίδευση Transformers**
  Τα πειράματα έγιναν για 300 εποχές (εκτός αυτών με CIFAR-10 που έγιναν για 400), με learning rate 0.001 με cosine annealing, weight decay 0.05 και 10 εποχές warmup. Batch size 256 για CIFAR-10/100 και 1024 για ImageNet-1k.

0.4.2 Αποτελέσματα

Ανάθεση πιθανοτήτων Dropout Σε αυτή την ενότητα θα συγκρίνουμε τις δυο εναλλακτικές τακτικές στρατηγικές ανάθεσης πιθανοτήτων dropout τόσο στα δύο CNNs όσο και στο ViT-Tiny με το σύνολο CIFAR-100. Στους μετασχηματιστές συγκρίνουμε και πολλαπλές τιμές $p_{\max}$ στε να αξιολογηθεί η επίδρασή του ανά μέθοδο και να εντοπιστεί η βέλτιστη τιμή. Τέλος, εξετάζουμε το προαιρετικό βήμα κάθε στρατηγικής, με στόχο να αποτιμηθεί η συμβολή τους και να εξαχθούν συμπεράσματα για τη συμπεριφορά της μεθόδου.

Πίνακας 2. *Ακρίβεια σύνολου αξιολόγησης (%) στα μοντέλα CNN στο σύνολο CIFAR-100 δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις.*

Μέθοδος	**CIFAR-100**			
	M_1		M_2	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	60.35 ± 0.44	-	59.02 ± 0.38	-
Y-Relative	60.39 ± 0.46	+ 0.04	59.31 ± 0.58	+ 0.29
Y-Ranking	**60.67** ± **0.45**	**+ 0.32**	**59.73** ± **0.22**	**+ 0.71**

Πίνακας 3. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CiFAR-100 με διαφορετικές στρατηγικές Generalized Y-drop για διάφορες τιμές $p_{\max}$, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. Το ΔY-drop είναι η διαφορά με το Dropout.*

Μέθοδος	**ViT-Tiny (CIFAR-100)**			
	Y-Ranking		Y-Relative	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	-	68.08 ± 0.34	-
Drop-Path	65.99 ± 0.05	- 2.09	65.99 ± 0.05	- 2.09
G. Y-drop $p_{\max}$ 0.3	68.24 ± 0.53	+ 0.16	68.23 ± 0.42	+ 0.15
G. Y-drop $p_{\max}$ 0.5	**68.46** ± **0.30**	**+ 0.38**	68.23 ± 0.23	+ 0.15
G. Y-drop $p_{\max}$ 0.7	68.29 ± 0.33	+ 0.21	68.30 ± 0.45	+ 0.22

Τα αποτελέσματα δείχνουν ότι, τόσο στα CNNs όσο και στο ViT-Tiny, οι νέες στρατηγικές υπερέχουν του απλού dropout και ανάμεσα στις δύο τα Ranking Rates πετυχαίνουν τις καλύτερες επιδόσεις, γεγονός που υποδηλώνει ότι προσεγγίσεις λιγότερο εξαρτημένες από την ακριβή μορφή της κατανομής αγωγιμότητας είναι πιο σταθερές και λιγότερο ευάλωτες στον θόρυβο που συχνά συναντάται σε μετρήσεις απόδοσης βασισμένες σε κλίση, όπως η αγωγιμότητα.

Ως προς την επιλογή του $p_{\max}$, μικρές τιμές όπως 0.3 έχουν μικρό κέρδος και μεγαλύτερη αστάθεια, υποδεικνύοντας ότι απαιτείται ευρύ φάσμα πιθανοτήτων ώστε να διαφοροποιείται ουσιαστικά η «τιμωρία» των σημαντικότερων νευρώνων από τους υπόλοιπους, ωστόσο μεγάλες τιμές όπως το 0.7 την αυξάνουν υπερβολικά και τελικά υποβαθμίζουν την επίδοση.

Τέλος, τα προαιρετικά βήματα δεν αποδείχθηκαν ωφέλιμα. Στα Relative Rates, η περικοπή (clipping) του 1% των μέγιστων και ελάχιστων τιμών σχεδόν εκμηδένισε τα κέρδη, κάτι που δείχνει ότι η βασική λειτουργία της μεθόδου είναι να ανακαλύπτει και να θέτει κατάλληλες τιμές dropout σε αυτές τις ακραίες τιμές. Απο την άλλη στα Ranking Rates η ύψωση σε δύναμη 2 μείωσε το κέρδος ακρίβειας κατά 0.1, περαιτέρω δείχνοντας πως δυσανάλογα μεγάλες τιμές dropout στα άκρα, είναι επιζήμιες και υπάρχει ανάγκη για κατάλληλη ισορροπία.

Αγωγιμότητα κόστους Επαναλαμβάνουμε τα πειράματα που παρουσίασαν τις βέλτιστες επιδόσεις στα CNNs και στο ViT-Tiny, αντικαθιστώντας τη μετρική απο αγωγιμότητα σε αγωγιμότητα κόστους. Επιπλέον, για την αρχιτεκτονική ViT-Tiny επεκτείνουμε την αξιολόγηση και στο σύνολο CIFAR-10.

Πίνακας 4. *Ακρίβεια σύνολου αξιολόγησης (%) στα μουτέλα CNN στο σύνολο CIFAR-100 χρησιμοποιώντας Generalized Y-drop με Αγωγιμότητα Κόστους έναντι της απλής Αγωγιμότητας, δείχνουτας του μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις.*

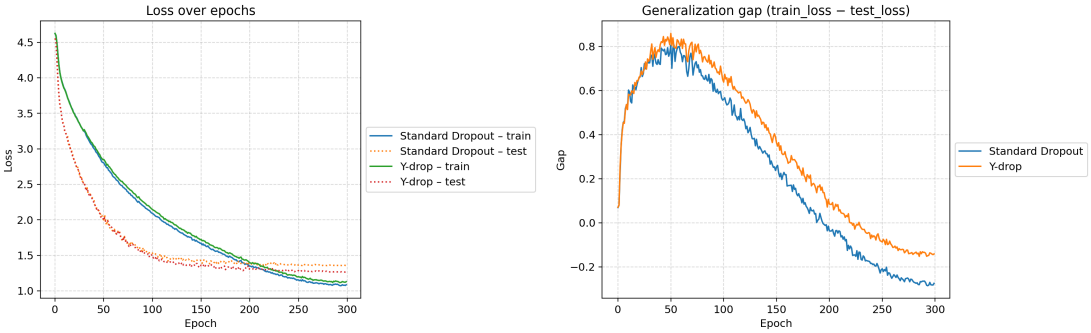
Μέθοδος	**CIFAR-100**			
	M_1		M_2	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	60.35 ± 0.44	-	59.02 ± 0.38	-
G. Y-Drop	60.67 ± 0.45	+ 0.32	59.73 ± 0.22	+ 0.71
G. Y-Drop Κόστους	**60.80** ± **0.39**	**+ 0.45**	**59.84** ± **0.41**	**+ 0.82**

Πίνακας 5. *Ακρίβεια σύνολου αξιολόγησης (%) στο μουτέλο ViT-Tiny στα σύνολα CIFAR-100 και CIFAR-10 χρησιμοποιώντας Generalized Y-drop με Αγωγιμότητα Κόστους έναντι της απλής Αγωγιμότητας, δείχνουτας του μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

Μέθοδος	**ΐT-Τινψ**			
	CIFAR-100		CIFAR-10	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	-	91.91 ± 0.32	-
Drop-Path	65.99 ± 0.05	- 2.09	90.8 ± 0.25	- 1.11
G. Y-drop	68.46 ± 0.30	+ 0.38	92.25 ± 0.17	+ 0.34
G. Y-drop Κόστους	**68.55** ± **0.28**	**+ 0.47**	**92.34** ± **0.22**	**+ 0.43**

Παρατηρούμε πως σε όλα τα πειράματα η αγωγιμότητα κόστους οδήγησε σε καλύτερες επιδόσεις απο την απλή αγωγιμότητα, κάτι που δείχνει πως η αλλαγή οδήγησε ένα σήμα σημαντικότητας πιο ευθυγραμμισμένο με τον στόχο εκπαίδευσης που παρέχει αποτελεσματικότερη και πιο στοχευμένη ανάθεση πιθανοτήτων dropout.

Επιπλέον, συγκρίνοντας τις καμπύλες της συνάρτησης κόστους στο σύνολο εκπαίδευσης και αξιολόγησης με χρήση Generalized Y-drop και dropout κατα την πάροδο της εκπαίδευσης, βλέπουμε πως το πρώτο πετυχαίνει μικρότερες τιμές στο σύνολο αξιολόγησης ενώ ταυτόχρονα έχει μεγαλύτερες στο σύνολο εκπαίδευσης, ενα εμφανές σημάδι βελτιωμένης κανονικοποίησης.



Εικόνα 1. *Καμπύλες της συνάρτησης κόστους στο σύνολο εκπαίδευσης και αξιολόγησης στο ViT-Tiny στο CIFAR-100 με χρήση Dropout συγκριτικά με Generalized Y-drop (αριστερά: πραγματικές τιμές και για τις δύο εκτελέσεις, δεξιά: διαφορά μεταξύ εκπαίδευσης και αξιολόγησης και για τις δύο εκτελέσεις).*

Επιλογή token και Layer Normalization Σε αυτή την ενότητα αξιολογούμε δύο σχεδιαστικές επιλογές για την εφαρμογή της μεθόδου σε Vits, (ι) τη χρήση αποκλειστικά των τιμών αγωγιμότητας κόστους του [CLS] token έναντι του μέσου όρου όλων των tokens και (ιι) υπολογισμός της αγωγιμότητας κόστους πριν ή μετα το Layer Normalization, με στόχο τον εντοπισμό των καταλληλότερων ρυθμίσεων.

Πίνακας 6. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στα σύνολα CIFAR-100 και CIFAR-10 χρησιμοποιώντας Generalized Y-drop με διαφορετικές στρατηγικές επιλογής tokens, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

Μέθοδος	**ViT-Tiny**			
	CIFAR-100		CIFAR-10	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	-	91.91 ± 0.32	-
G. Y-drop Κόστους Μέσος όρος Μαρκών	68.46 ± 0.30	+ 0.38	92.25 ± 0.17	+ 0.34
G. Y-drop Κόστους [CLS] Μάρκα	**68.55 ± 0.28**	**+ 0.47**	**92.34 ± 0.22**	**+ 0.43**

Τα αποτελέσματα δείχνουν ότι ο μέσος όρος όλων των tokens οδηγεί σε χαμηλότερες επιδόσεις σε σχέση με τη χρήση μόνο της [CLS]. Το εύρημα αυτό στηρίζει την υπόθεση ότι η πλουσιότερη και πιο διαφοροποιημένη κατανομή τιμών τοψ [CLS] token, είναι πιο αντιπροσωπευτικό σήμα σημαντικότητας, ενώ τα υπόλοιπα tokens ενσωματώνουν κυρίως τοπική πληροφορία που δεν είναι πάντα χρήσιμη για τον συνολικό στόχο.

Πίνακας 7. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στα σύνολα CIFAR-100 και CIFAR-10 χρησιμοποιώντας Generalized Y-drop με αγωγιμότητα κόστους υπολογισμένη πριν ή μετά το Layer Normalization, δείχνοντας τον μέσο όρο και την τυπική απόκλιση για 5 εκτελέσεις. Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

Μέθοδος	**ViT-Tiny**			
	CIFAR-100		CIFAR-10	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	-	91.91 ± 0.32	-
G. Y-drop μετά Norm	68.46 ± 0.30	+ 0.38	92.25 ± 0.17	+ 0.34
G. Y-drop πριν Norm	68.36 ± 0.36	+ 0.28	92.27 ± 0.33	+ 0.36
G. Y-drop κόστους μετά Norm	68.55 ± 0.28	+ 0.47	92.34 ± 0.22	+ 0.43
G. Y-drop κόστους πριν Norm	**68.85 ± 0.06**	**+ 0.77**	**92.38 ± 0.19**	**+ 0.47**

Απο την άλλη με αγωγιμότητα κόστους οι επιδόσεις βελτιώνονται όταν την υπολογίζουμε πριν το Layer Normalization. Συνεπώς αφαιρώντας την πληροφορία απο το skip connection παίρνουμε καθαρότερη εκτίμηση της σημασίας για κάθε νευρώνα του επιπέδου.

Με βάση αυτά τα αποτελέσματα, όλα τα υπόλοιπα πειράματα με μετασχηματιστες θα γίνουν με χρήση αγωγιμότητα κόστους του [CLS] token υπολογισμένη πριν το Layer Normalization.

Επιρροή υπερ παραμέτρων Κάνουμε πειράματα που εξετάζουμε την επίδραση βασικών υπερ-παραμέτρων της μεθόδου. Συγκεκριμένα, μελετάμε: (i) την elasticity, ώστε να αξιολογηθεί η ευαισθησία της μεθόδου στον ρυθμό αλλαγής του κινητού μέσου όρου, (ii) τον annealing factor, για να προσδιοριστεί πότε οι τιμές αγωγιμότητας κόστους γίνονται ουσιώδεις και πότε είναι σκόπιμη η ενεργοποίηση της μεθόδου, (iii) τον αριθμό διακριτών βημάτων n_{steps} και (iv) αριθμό δειγμάτων n_{samples} για υπολογισμό της μετρικής ανά ενημέρωση στε να διερευνηθεί αν ένας πιο λεπτομερής ή πιο αντιπροσωπευτικός στο σύνολο δεδομένων αντίστοιχα υπολογισμός της μετρικής μεταφράζεται σε ουσιαστικό κέρδος απόδοσης.

- **Elasticity:** Είτε διπλασιάζοντας ή υπο διπλασιάζοντας την τιμή αρχική τιμή του 0.01 δεν επηρεάζει ουσιαστικά τα αποτελέσματα, γεγονός που υποδηλώνει ότι η μέθοδος είναι ανθεκτική ως προς αυτή την υπερ-παράμετρο και η επιλογή μιας οποιαδήποτε μικρής τιμής είναι θεμιτή.

Πίνακας 8. *Ακρίβεια σύνολου αξιολόγησης (%) στο μουτέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές Elasticity. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
Elasticity a=0.005	68.77 ± 0.39	+0.69
Elasticity a=0.01	**68.85 ± 0.06**	**+0.77**
Elasticity a=0.02	68.83 ± 0.46	+0.75

- **Annealing factor:** Τα αποτελέσματα έδειξαν πως αργή ενεργοποίηση(μετά το 20% της εκπαίδευσης) μειώνει αισθητά τα κέρδη, υποδεικνύοντας ότι η συμβολή της μεθόδου στα πρώτα στάδια είναι κρίσιμη. Παράλληλα, η ενεργοποίηση από την αρχή της εκπαίδευσης εξακολουθεί να αποδίδει, ωστόσο τα καλύτερα αποτελέσματα παρατηρούνται με ενεργοποίηση στο 5%, όπου δίνεται σύντομος χρόνος το μοντέλο να σταθεροποιήσει αρχικές αναπαραστάσεις και να αποκτήσει ουσιώδης τιμές αγωγιμότητας κόστους.

Πίνακας 9. *Ακρίβεια σύνολου αξιολόγησης (%) στο μουτέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές annealing factor. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
Ann. Factor K=0(%)	68.89 ± 0.30	+0.81
Ann. Factor K=5(%)	**68.92 ± 0.25**	**+0.84**
Ann. Factor K=10(%)	68.85 ± 0.06	+0.77
Ann. Factor K=20(%)	68.45 ± 0.28	+0.37
Ann. Factor K=50(%)	68.37 ± 0.22	+0.29

- $n_{\textbf{steps}}$**:** Παρατηρούμε πως η μείωση των διακριτών σημάτων σε σχέση με την αρχική τιμή (5) οδηγεί σε σημαντική μείωση των κερδών. Από την άλλη η αύξηση οδηγεί σε οριακή αύξηση της επίδοσης, κάτι που ωστόσο συνδυάζεται με σημαντική αύξηση του υπολογιστικού κόστους. Συνεπώς, τα 5 βήματα αποτελούν ικανοποιητικό συμβιβασμό μεταξύ ακρίβειας υπολογισμού και κόστους.

Πίνακας 10. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές n_{steps}. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
n_{steps}=2	68.35 ± 0.28	+0.27
n_{steps}=3	68.53 ± 0.14	+0.45
n_{steps}=5	68.85 ± 0.06	+0.77
n_{steps}=10	68.90 ± 0.21	+0.82
n_{steps}=20	**68.92 ± 0.27**	**+0.84**

- $n_{\textbf{samples}}$**:** Παρατηρούμε πως υπάρχει σαφής συσχέτιση μεταξύ περισσότερων δειγμάτων και καλύτερης απόδοσης, γεγονός που δείχνει ότι ο μέσος όρος αγωγιμότητας κόστους χρειάζεται κατάλληλα μεγάλο αριθμό δεδομένων για να είναι αντιπροσωπευτικός του συνόλου δεδομένων και να δείχνει ποιοι νευρώνες είναι πραγματικά οι σημαντικότεροι. Ωστόσο όπως και με τα n_{steps} , το κόστος αυξάνεται σημαντικά με αύξηση των n_{samples}, περιορίζοντας την πρακτική αξιοποίηση των βελτιωμένων επιδόσεων.

Πίνακας 11. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για Y-drop με διαφορετικές τιμές $n_{samples}$. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
n_{samples} = 26	68.64 ± 0.17	+0.56
n_{samples} = 51	68.85 ± 0.06	+0.77
n_{samples} = 102	68.98 ± 0.23	+0.90
n_{samples} = 204	**69.26 ± 0.34**	**+1.18**

Προσπάθειες μείωσης κόστους Σε αυτή την ενότητα αξιολογούμε στρατηγικές μείωσης του υπολογιστικού κόστους μέσω λιγότερων υπολογισμών της αγωγιμότητας κόστους.

- **Πρόωρος τερματισμός:** παρατηρούμε πως όσο νωρίτερα γίνεται τερματισμός της μεθόδου τόσο μειώνεται και το κέρδος στην ακρίβεια, με μεγαλύτερη πτώση όταν απλώς διατηρούνται σταθερές οι πιθανότητες dropout της τελευταίας ενημέρωσης, σε σύγκριση με την επιστροφή σε κλασικό dropout. Το αποτέλεσμα υποδηλώνει ότι η μέθοδος παραμένει χρήσιμη σε όλη τη διάρκεια της εκπαίδευσης και ότι η σχετική σημαντικότητα των νευρώνων συνεχίζει να μεταβάλλεται, καθιστώντας αναγκαίες ενημερώσεις και σε μεταγενέστερα στάδια.

Πίνακας 12. *Ακρίβεια σύνολου αξιολόγησης (%) (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για το Generalized Y-drop με διαφορετικές στρατηγικές πρόωρου τερματισμού. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
G. Y-drop 50(%) εκπαιδευσης, τελος ενημερώσεων	68.26 ± 0.28	+0.16
G. Y-drop 50(%) εκπαίδευσης μετα dropout	68.35 ± 0.19	+0.27
G. Y-drop 75(%) εκπαίδευσης, τελος ενημερώσεων	68.47 ± 0.28	+0.39
G. Y-drop 75(%) εκπαίδευσης μετα dropout	68.62 ± 0.19	+0.54
G. Y-drop 100(%) εκπαίδευσης	**68.85 ± 0.06**	**+0.77**

- **Αραιότερες ενημερώσεις :** Μεγαλώνοντας το διάστημα ενημέρωσης καθόλη την διάρκεια της εκπαίδευσης μειώνει σημαντικά τις επιδόσεις δείχνοντας ότι είναι σημαντική η τακτική ανανέωση των τιμών. Αντίθετα, η παραλλαγή με προγραμματισμένη αύξηση του διαστήματος παραμένει ιδιαίτερα ανταγωνιστική. Ειδικά όταν το διάστημα αυξάνεται σε δύο σημεία της εκπαίδευσης, επιτυγχάνει παρόμοιες επιδόσεις με την πλήρη μέθοδο, απαιτώντας περίπου 40% λιγότερους υπολογισμούς αγωγιμότητας κόστους. Το εύρημα αυτό ενισχύει την υπόθεση ότι ο ρυθμός μεταβολής των σχετικών τιμών της μετρικής μειώνεται σε μεταγενέστερα στάδια και ανοίγει τον δρόμο για μεθόδους που μπορούν να πετύχουν εξίσου καλά αποτελέσματα για πολύ μικρότερο κόστος.

Πίνακας 13. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 για το Generalized Y-drop με στρατηγικές αραιότερων ενημερώσεων. (μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
G. Y-drop διάστημα 1	**68.85 ± 0.06**	**+0.77**
G. Y-drop διάστημα 2	68.48 ± 0.32	+0.40
G. Y-drop διάστημα 5	68.30 ± 0.37	+0.22
G. Y-drop με πρόγραμμα n_{up}=1	68.65 ± 0.22	+0.57
G. Y-drop με πρόγραμμα n_{up}=2	**68.79 ± 0.17**	**+0.71**

Imagenet,Μεγαλύτεροι Μετασχηματιστές και επαυξηση δεδομένων Πραγματοποιήσαμε πειράματα στο πιο απαιτητικό σύνολο δεδομένων ImageNet-1k, χρησιμοποιώντας μεγαλύτερα ViT μοντέλα (ViT-Small και ViT-Base), με στόχο να αξιολογηθεί η βιωσιμότητα της μεθόδου σε δυσκολότερες συνθήκες εκπαίδευσης. Και στις δύο περιπτώσεις, το Generalized Y-drop οδήγησε σε βελτιωμένη απόδοση σε σχέση με το απλό dropout, δείχνοντας ότι τα οφέλη του διατηρούνται και σε μεγαλύτερα μεγέθη.

Πίνακας 14. *Ακρίβεια σύνολου αξιολόγησης (%) σε μεγαλύτερα μουτέλα ViT στο σύνολο ImageNet. (μέσος όρος ± τυπική απόκλιση για 3 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

Μέθοδος	**ImageNet**			
	ViT – Small		*ViT – Base*	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Dropout	76.3 ± 0.26	-	76.14 ± 0.32	-
Drop-Path	74.51 ± 0.16	-1.79	73.93 ± 0.18	-2.11
G. Y-drop	**76.79** ± **0.21**	**+ 0.49**	**76.70** ± **0.20**	**+ 0.56**

Επιπλέον, εξετάσαμε τη συμβατότητα της μεθόδου σε συνθήκες έντονης επαύξησης δεδομένων, μέσω πειραμάτων σε αρχιτεκτονικές DeiT. Σε αυτην την αρχιτεκτονική, το κλασικό dropout συνήθως αντικαθίσταται από Drop-Path, καθώς αυτό προσφέρει καλύτερα αποτελέσματα, ωστόσο είναι σημαντικό να ελέγξουμε αν η βελτιωμένη μας εκδοχή μπορεί να το ξεπεράσει Τα αρχικά πειράματα δεν ήταν ενθαρρυντικά με την εκπαίδευση με Generalized Y-drop ή με dropout να υπολείπεται σημαντικά του Drop-Path στο DeiT-Tiny, ενώ στο DeiT-Base παρατηρήθηκε ακόμη και αποτυχία σύγκλισης. Αυτό το αποτέλεσμα οδήγησε σε αντικατάσταση της μεθόδου με το αυξανομενο Generalized Y-drop.

Πίνακας 15. *Ακρίβεια σύνολου αξιολόγησης (%) στο μουτέλο DeiT στο σύνολο ImageNet με χρήση αυξανόμενου Generalized Y-drop. (μέσος όρος ± τυπική απόκλιση για 3 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Drop-Path.*

Μέθοδος	**ImageNet**			
	DeiT – Tiny		*DeiT – Base*	
	Ακρίβεια	ΔY-drop	Ακρίβεια	ΔY-drop
Drop-Path	72.00 ± 0.34	-	**81.79** ± **0.34**	-
Αυξανόμενο Dropout	72.02 ± 0.09	0.02	80.92 ± 0.07	-0.87
Αυξανόμενο G. Y-drop	**72.38** ± **0.26**	**+ 0.38**	80.95 ± 0.03	-0.84

Παρόλο που η αλλαγή αυτή ήταν πιο ανταγωνιστική, με ελαφρώς καλύτερες επιδόσεις απο το Drop-Path στο DeiT-Tiny, στο μεγαλύτερο DeiT-Base ήταν αισθητά κατώτερο. Αυτά τα αποτελέσματα επιβεβαιώνουν τα ευρήματα πολλών δημοσιεύσεων [50] πως υπό ισχυρή επαύξηση δεδομένων, το περιθώριο βελτίωσης από επιπλέον regularization μειώνεται ή ακόμα γίνεται αρνητικό. Ως αποτέλεσμα η ηπιότερη μορφή κανονικοποίησης του Drop-Path είναι πιο αποτελεσματική απο το Αυξανόμενο Generalized Y-drop.

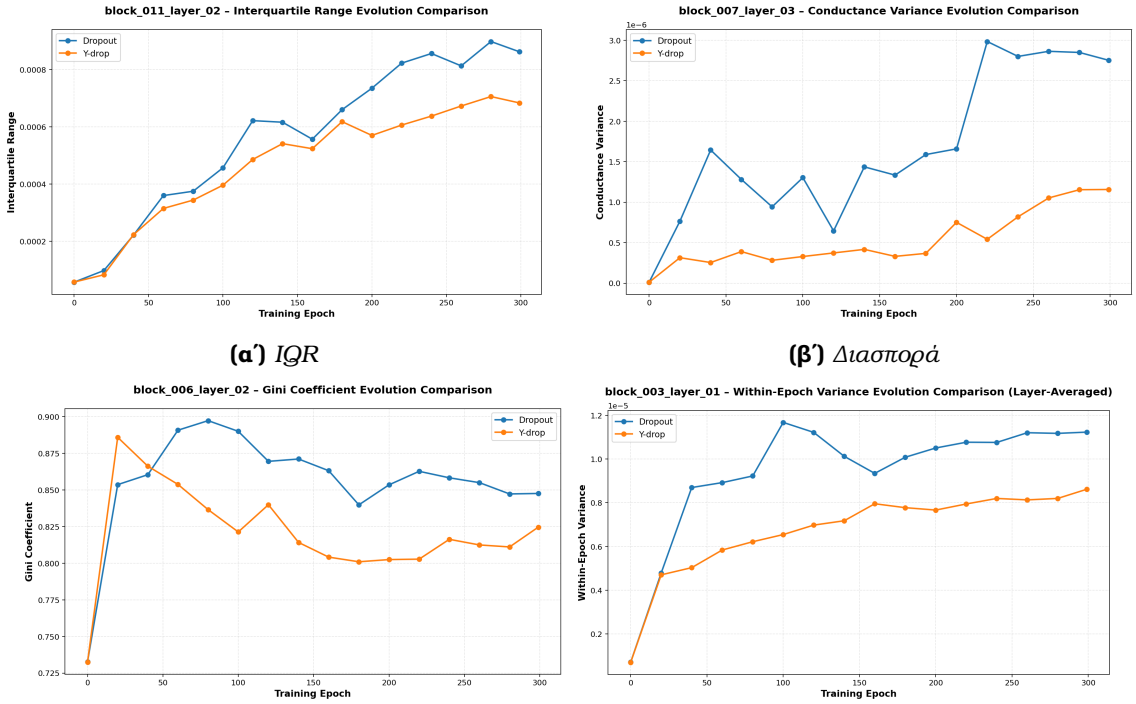
0.4.3 Ανάλυση Αγωγιμότητας Κόστους

Σε αυτή την ενότητα εξετάζουμε πιο αναλυτικά την αγωγιμότητα κόστους. Εξετάζουμε την εξέλιξή της κατά την εκπαίδευση (στο σύνολο εκπαίδευσης) και τη μορφή της στο τέλος της (στο σύνολο αξιολόγησης), ώστε να συγκρίνουμε τη συμπεριφορά της όταν χρησιμοποιείται

Generalized Y-drop έναντι του κλασικού dropout. Επιπλέον, συγκρίνουμε τα μοτίβα που εμφανίζονται σε μοντέλα με διαφορετικές επιδόσεις και ρυθμίσεις εκπαίδευσης και επιδόσεις (όλα ViT-Tiny στο CIFAR-100), και σε διαφορετικά βάθη της αρχιτεκτονικής, με στόχο να εξαχθούν συμπεράσματα για το τι αποτυπώνει η μετρική, τι υποδηλώνει για τη λειτουργία του δικτύου και πιθανούς περιορισμούς της τρέχουσας μορφής του Generalized Y-drop.

Αγωγιμότητα κόστους κατά την εκπαίδευση Σε επιλεγμένες εποχές της εκπαίδευσης υπολογίζουμε, για πολλαπλά batches του συνόλου εκπαίδευσης, τη μέση αγωγιμότητα κόστους ανά νευρώνα (χρησιμοποιώντας μόνο το token [CLS]). Από τις τιμές αυτές εξάγουμε, για κάθε νευρώνα, τη μέση τιμή και τη διασπορά ανά τα batches, που ονομάζουμε διασπορά-εποχής (within-epoch variance).

Σε κάθε επίπεδο υπολογίζουμε, με χρήση των μέσων τιμών των νευρώνων του, δείκτες κατανομής των τιμών μεταξύ των νευρώνων (διασπορά, IQR και δείκτη Gini), ώστε να εκτιμήσουμε πόσο άνισα κατανέμεται η συνεισφορά στο κόστος ανά επίπεδο. Παράλληλα, υπολογίζουμε τη μέση διασπορά-εποχής ανά επίπεδο, ώστε να αποτυπώσουμε πόσο μεταβάλλεται, κατά μέσο όρο, η συνεισφορά κάθε νευρώνα μεταξύ διαφορετικών δειγμάτων.



(α΄) *IQR* **(β΄)** *Διασπορά*

(γ΄) *Gini* **(δ΄)** *Διασπορά-Εποχής*

Εικόνα 2. *Interquartile range, διασπορα, Gini coefficient και διασπορά εποχής της αγωγιμότητας κόστους σε διαάφορα επίπεδα κατά την διάρκεια της εκπαίδευσης.Σύγκριση Dropout εναντίον Generalized Y-drop.*

Σε όλες τις παραπάνω μετρικές, το μοντέλο που εκπαιδεύεται με Generalized Y-drop εμφανίζει χαμηλότερες τιμές στο μεγαλύτερο μέρος της εκπαίδευσης σε σχέση με το μοντέλο με dropout σε όλα σχεδόν τα επίπεδα. Το εύρημα αυτό υποδηλώνει ότι η μέθοδος οδηγεί σε πιο ίση και ομοιόμορφη κατανομή της συνεισφοράς μεταξύ νευρώνων και δειγμάτων, περιορίζοντας την κυριαρχία μικρών υποομάδων νευρώνων και αποτρέποντας την

υπερ-εξειδίκευση σε θορυβώδη χαρακτηριστικά συγκεκριμένων δειγμάτων, προς όφελος πιο γενικεύσιμων αναπαραστάσεων.

Αγωγιμότητα κόστους στο σύνολο αξιολόγησης Υπολογίζουμε, για πλήρως εκπαιδευμένα μοντέλα, τη μέση αγωγιμότητα κόστους ανά νευρώνα για όλο σύνολο αξιολόγησης (για κάθε δείγμα χρησιμοποιούμε το άθροισμα όλων των token). Παρουσιάζουμε το άθροισμα των τιμών ανά επίπεδο και συγκρίνουμε την κατανομή του μεταξύ μοντέλων εκπαιδευμένων με dropout και με Generalized Y-drop.

Παρατηρείται ένα χαρακτηριστικό μοτίβο: το μοντέλο με Generalized Y-drop (που επιτυγχάνει και υψηλότερη ακρίβεια) εμφανίζει υψηλότερες θετικές τιμές στα πρώτα επίπεδα, ενώ στα βαθύτερα επίπεδα εμφανίζει πιο αρνητικές (χαμηλότερες) τιμές σε σχέση με το μοντέλο που εκπαιδεύτηκε με απλό dropout. Αρχικά θεωρήσαμε πως το μοτίβο να αποτελεί παρενέργεια, είτε λόγω θορύβου σε μετρικές βασισμένες σε κλίση ειδικά σε βαθύτερες αρχιτεκτονικές, είτε λόγω επιλογών όπως η χρήση του token [CLS] στα πρώιμα επίπεδα.

Για να επιβεβαιώσουμε ή να καταρρίψουμε τις υποθεσεις μας και να εξηγήσουμε το μοτίβο κάνουμε τα εξής πειράματα.

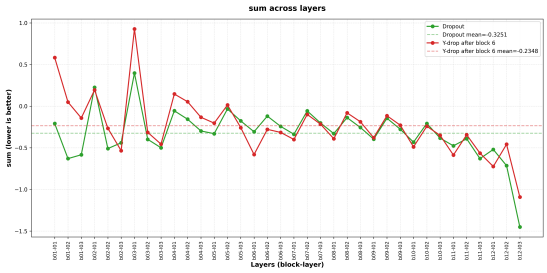
- Πρώτον, συγκρίναμε δύο μοντέλα Generalized Y-drop με διαφορετικό $n_{\text{σαμπλες}}$. Το μοντέλο με περισσότερα δείγματα ανά ενημέρωση (και καλύτερη επίδοση) εμφάνισε το ίδιο μοτίβο εντονότερα, γεγονός που αποδυναμώνει την υπόθεση ότι πρόκειται για τυχαίο θόρυβο.

- Δεύτερον, αντικαταστήσαμε το Generalized Y-drop στα πρώτα επίπεδα με απλό dropout. Περιορίζοντας τη μέθοδο στα τελευταία 6 blocks επιτυγχάνουμε παρόμοια τελική ακρίβεια με την εφαρμογή της σε όλο το δίκτυο, κάτι που δείχνει ότι η κύρια επίδραση της μεθόδου εντοπίζεται στα βαθύτερα blocks.

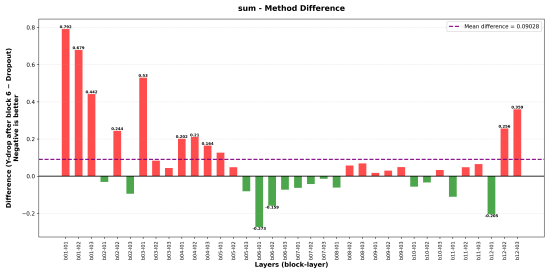
Πίνακας 16. *Ακρίβεια σύνολου αξιολόγησης (%) στο μοντέλο ViT-Tiny στο σύνολο CIFAR-100 οπου το Generalized Y-drop χρησιμοποιείται στα τελυταία μόνο μπλοκς ενώ τα πρώτα χρησιμοποιούν dropout.(μέσος όρος ± τυπική απόκλιση για 5 εκτελέσεις). Το ΔY-drop είναι η απόλυτη αλλαγή σε σχέση με το Dropout.*

ViT-Tiny (CIFAR-100)		
Μέθοδος	Ακρίβεια	ΔY-drop
Dropout	68.08 ± 0.34	–
G. Y-drop τελυταία 3 μπλοκς	68.61 ± 0.34	+0.53
G. Y-drop τελυταία 6 μπλοκς	**68.78 ± 0.24**	**+0.70**
G. Y-drop τελυταία 10 μπλοκς	**68.73 ± 0.32**	**+0.65**
G. Y-drop 12(όλα) μπλοκς	**68.85 ± 0.06**	**+0.77**
G. Y-drop 12(όλα) μπλοκς,στα πρώτα 2 μέσος όρος tokens	68.60 ± 0.47	+0.52

Ωστόσο, το μοτίβο στα αθροίσματα αγωγιμότητας κόστους παραμένει και στα πρώτα επίπεδα, υποδεικνύοντας ότι σχετίζεται με τη συνολική επίδοση του μοντέλου και όχι με ανεπιθύμητη παρενέργεια του Generalized Y-drop.



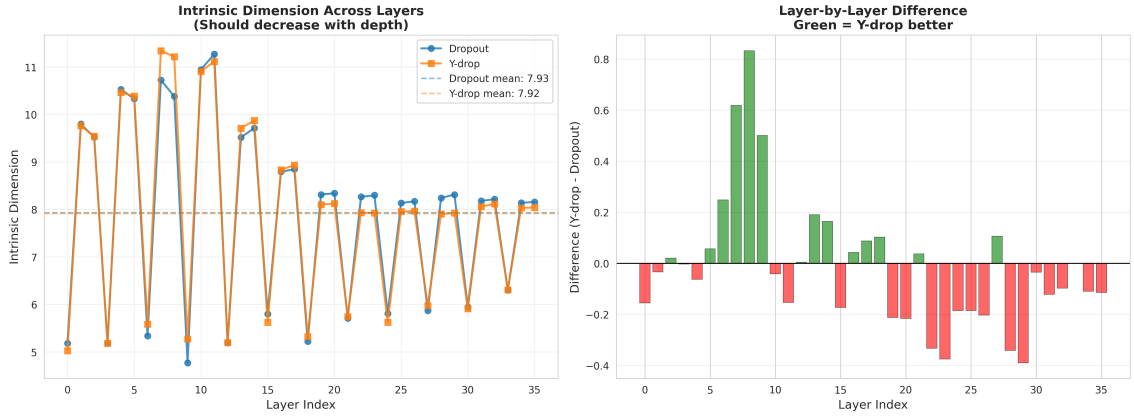
(α′) *Αθροισμα Αγωγιμότητας κόστους - Dropout εναντιον Generalized Y-drop μονο στα τελευταία 6 blocks*



(β′) *Αθροισμα Αγωγιμότητας κόστους - Διαφορά ανάμεσα στις εκτελέσεις*

Εικόνα 3. *Άθροισμα της μέσης αγωγιμότητας κόστους ανα νευρώνα σε κάθε επίπεδο για ολόκληρο το σύνολο αξιολόγησης σε κάθε επίπεδο. Σύγκριση Dropout εναντιον Generalized Y-drop μονο στα τελευταία 6 blocks.*

Το παραπάνω παρουσιάζουν αναλογίες με ευρήματα από τη μετρική Intrinsic Dimension, η οποία εκτιμά τον ελάχιστο αριθμό διαστάσεων που απαιτούνται για να περιγραφεί ένα σύνολο τιμών, στην περίπτωση μας η έξοδος ενός επιπέδου. Έχει παρατηρηθεί ότι η intrinsic dimension αυξάνεται στα πρώτα επίπεδα, όπου το δίκτυο «επεκτείνει» και αναζητά χαρακτηριστικά (encoder-like επίπεδα), ενώ μειώνεται στα βαθύτερα επίπεδα, όπου διατηρεί και συμπιέζει μόνο την απαραίτητη πληροφορία για την τελική απόφαση (decoder-like επίπεδα). Επιπλέον, σε μοντέλα υψηλότερης απόδοσης η συμπεριφορά αυτή τείνει να είναι εντονότερη



Εικόνα 4. *Intrinsic Dimension: Σύγκριση Dropout εναντιον Generalized Y-drop.*

Αντίστοιχα η διαφορα συμπεριφορά της αγωγιμότητας κόστους σε διαφορετικά βάθη μπορεί να εξηγηθεί με βάση αυτή την διαφορετική συμπεριφορά των μοντέλων. Στα πρώιμα, encoder-like επίπεδα, οι νευρώνες δεν οδηγούν άμεσα σε μείωση της συνάρτησης κόστους όταν εξετάζονται απομονωμένοι απο το υπολοιπο δίκτυο, με αποτέλεσμα να εμφανίζονται περισσότερο θετικές τιμές, οι οποίες σε πιο καλύτερα μοντέλα είναι πιο αυξημένες καθώς πραγματοποιείται βαθύτερη αναζήτηση χαρακτηριστικών. Αντίθετα, στα βαθύτερα, decoder-like επίπεδα, οι νευρώνες συνδέονται πιο άμεσα με τη μείωση του κόστους και εμφανίζουν αρνητικές τιμές, οι οποίες γίνονται πιο έντονες στα καλύτερα μοντέλα.

Η παραπάνω ερμηνεία εξηγεί και γιατί το Generalized Y-drop έχει ισχυρότερη επίδραση στα βαθύτερα επίπεδα: η ανάθεση αυξημένου dropout σε νευρώνες με πολύ αρνητική αγω-

γιμότητα κόστους συνδέεται άμεσα με τη λειτουργία των decoder-like blocks. Παράλληλα, υποδεικνύει ότι στα πρώτα επίπεδα ενδέχεται να απαιτούνται εναλλακτικες στρατηγικές ή ηπιότερη εφαρμογή της μεθόδου.

0.5 Συμπεράσματα

Σε αυτη την εργασία βελτιώσαμε και επεκτείναμε την τεχνική Y-drop, αναδεικνύοντας βασικές αδυναμίες της και προτείνοντας στοχευμένες λύσεις. Εισάγουμε το Generalized Y-drop, το οποίο χρησιμοποιεί απλοποιημένη ανάθεση πιθανοτήτων dropout μέσω των Ranking Rates, μειώνοντας σημαντικά την ανάγκη ρύθμισης υπερ-παραμέτρων και διευκολύνοντας την εφαρμογή του σε διαφορετικές αρχιτεκτονικές.

Κεντρική συνεισφορά αποτελεί η αγωγιμότητα κόστους, μια νέα παραλλαγή της αγωγιμότητας που υπολογίζει τη συνεισφορά των νευρώνων ως προς τη συνάρτηση κόστους αντί για την τελική πρόβλεψη. Αυτή, η πιο ευθυγραμμισμένη με το στόχο βελτιστοποίησης, μετρική οδήγησε σε ισχυρότερη κανονικοποίηση όπου το Generalized Y-drop αναθέτοντας αυξημένες πιθανότητες dropout σε νευρώνες που συνδέονται με έντονη μείωση του κόστους στο σύνολο εκπαίδευσης, προκαλεί αυξημένη τελική ακρίβεια, περιοριζει την εκμάθηση θορυβωδών μοτίβων και προάγει πιο ομοιόμορφη κατανομή συνεισφοράς μεταξύ των νευρώνων ενός επιπέδου σε περισσότερα δείγματα.

Διαπιστώσαμε ότι η μέθοδος παραμένει χρήσιμη σε όλα τα στάδια της εκπαίδευσης και ότι επωφελείται από μεγαλύτερο αριθμό δειγμάτων ανά ενημέρωση, χωρίς να απαιτεί κατ' ανάγκη πιο λεπτομερή (και ακριβότερο) υπολογισμό της μετρικής. Επιπλέον, τα κέρδη στην τελική απόδοση διατηρούνται και σε μεγαλύτερες αρχιτεκτονικές και σε πιο απαιτητικά σύνολα δεδομένων.

Προτείναμε επίσης προσαρμογές ειδικά για Μετασχηματιστες όρασεις για να μεγιστοποιήσουμε το τελικό κέρδος και αξιολογήσαμε τη μέθοδο υπό συνθήκες έντονης επαύξησης δεδομένων, όπου, όπως και το κλασσικό dropout, η επίδρασή της εμφανίζεται περιορισμένη.

Παράλληλα, εξετάσαμε στρατηγικές μείωσης κόστους και καταλήξαμε σε μια προγραμματισμένη μέθοδο, η οποία εκμεταλλεύεται τον μικρότερο ρυθμό αλλαγής της μετρικής σε μεταγενέστερα στάδια όπου και αραιώνει το διάστημα μεταξύ ενημερώσεων των πιθανοτήτων dropout, επιτυγχάνοντας συγκρίσιμες επιδόσεις με σημαντικά μικρότερο υπολογιστικό κόστος.

Τέλος, πραγματοποιήσαμε βαθύτερη ανάλυση της Αγωγιμότητας κόστους, αναδεικνύοντας και εξηγώντας τη διαφορετική συμπεριφορά της στα πρώιμα έναντι των βαθύτερων επιπέδων, καθώς και το γεγονός ότι το Generalized Y-drop είναι πιο κατάλληλο και αποτελεσματικό κυρίως στα βαθύτερα τμήματα του δικτύου.

Συνολικά, η εργασία αυτή συμβάλλει στην ανάπτυξη αποτελεσματικότερων σχημάτων κανονικοποίησης που βασίζονται σε σήματα απόδοσης, τόσο μέσω της βελτίωσης του Y-drop όσο και μέσω της εισαγωγής της αγωγιμότητας κόστους μιας νέας μετρικής που επιτρέπει ακριβέστερη αναγνώριση σημαντικών νευρώνων ως προς τον εκπαιδευτικό στόχο.

Chapter 1

Introduction

1.1 Motivation

In recent years, machine learning and in particular its subfield deep learning, has achieved major breakthroughs across a wide range of tasks and data modalities, often surpassing traditional statistical models while eliminating the need for manual feature engineering [9, 10]. To achieve these impressive performances required models to scale substantially in depth, width, and architectural complexity, making interpreting how these models operate internally and understanding how they arrive at their final predictions increasingly difficult. This has resulted in deep neural networks functioning as black boxes [11], creating ethical dilemmas especially in critical fields like healthcare and automated vehicles where trust, transparency, fairness, robustness and regulatory compliance are essential [28, 29].

To address this issue, many works have emerged under the umbrella of Explainability Metrics, within the broader field of Explainable Artificial Intelligence (XAI), that try to provide insight into the decision-making process of deep learning. Among them, a prominent subfield is attribution methods, which assign importance scores to elements such as input features, neurons, layers, or learned representations, indicating their contribution to the model's output in a way humans can interpret.

While attribution metrics and XAI in general were developed originally to improve the interpretability of models, they can also serve as levers to change and improve that behaviour. Many methods have been used for data debugging to improve the quality of the dataset using feedback from the model [51, 52, 53] or model compression and pruning to create smaller, more efficient networks with faster inference [54, 55, 56, 57].

Another promising but comparatively underexplored direction is injecting attribution feedback during training to steer optimization by penalizing undesirable behaviors or rewarding desirable ones. A successful example of such a method is Y-drop [8], which uses the Conductance attribution metric [12] to dynamically adjust neuron dropout rates based on their contribution to the model's prediction. Following the insights of other dynamic dropout rate assignment methods like Standout [58], it showcased that assigning higher dropout probabilities to neurons deemed most influential according to Conductance improves regularization and final performance compared to standard dropout across several Convolutional Neural Network (CNN) architectures and datasets.

Despite its reported benefits, the original Y-drop framework exhibits several limitations that restrict its practicality and scalability. First, the dropout rate assignment mechanism requires tuning a large amount of hyperparameters and introduces various forms of noise and dropout rate clipping. This makes hyperparameter optimization a significant challenge, especially when integrating it in new architectures that requires a different hyperparameter balance and makes comparison with other drop-based regularizations difficult. Furthermore, Y-drop does not fully exploit the granularity of the attribution signal, as neurons are grouped into only two discrete buckets rather than being treated according to their relative importance. Conductance itself also may not be an optimal metric as its importance does not directly align with the training objective and neurons negatively guiding predictions may also be considered important. Finally, the method substantially increases the training computational cost as an additional forward–backward pass is required to calculate Conductance at every training iteration.

In this thesis, we address these limitations by proposing Generalized Y-drop, a simplified and more flexible dynamic dropout framework that is easier to apply across architectures and requires significantly less hyperparameter tuning. We introduce and evaluate alternative dropout rate assignment strategies that rely on relative neuron importance, demonstrating consistent performance gains across both CNNs and Vision Transformer architecture. Central to this work is the introduction of Loss Conductance, a novel reformulation of the Conductance metric that attributes neuron importance with respect to the training loss rather than the final prediction. This more objective aligned metric achieves increased regularization capabilities compared to the standard Conductance. Additionally, we evaluate several proposed computationally cheaper Generalized Y-drop variants, showcasing that through an appropriate scheduled dropout-rate update frequency strategy the method can achieve comparable performance while significantly reducing the additional computational cost.

Throughout this work, we conduct extensive empirical analyses to identify the sources of performance gains, introduce Transformer specific modifications to maximize effectiveness and examine the limitations of the proposed framework. Moreover we analyze the behavior of Loss Conductance across layers, training stages, and inference, providing insights into how attribution patterns evolve in deep architectures, how they correlate with performance and how Generalized Y-drop influences these dynamics.

Overall, this thesis aims to deliver a more robust, computationally efficient and easily applicable dropout framework than the original Y-drop while also introducing Loss Conductance , a novel loss based attribution metric, opening the way for further development of loss based attribution signals and possible integration in deep neural network training.

1.2 Contributions

The main contributions of this thesis are the following:

- We introduce simplified, less hyperparameter-heavy dropout rate assignments methods that take into account relative neuron importance and can be used across layer

attribution methods as well as compare their impact and behaviour when using the Conductance metric, selecting the best for our Generalized Y-drop framework.

- We present Loss Conductance, a new formulation of the Conductance metric that attributes neuron importance with respect to the training loss instead of the final prediction, providing a more objective-aligned and informative attribution signal that improves regularization.
- We integrate Generalized Y-drop into the Vision Transformer (ViT) architecture, addressing Transformer-specific challenges such as token selection, residual connections, and layer normalization to maximize performance.
- We conduct extensive ablation studies across CNNs and Transformers to analyze the impact on performance caused by outliers, dropout rate distributions, model and dataset size, attribution granularity, amount of Loss Conductance samples, and the importance of the method across stages of training and architectural depth. These experiments aim to identify the key factors responsible for performance gains, help select optimal configurations and address limitations.
- We test several computationally cheaper Generalized Y-drop variants and propose a scheduled dropout-rate update frequency strategy that significantly reduces the number of auxiliary forward–backward passes while producing comparable performance to the default method.
- We provide a deeper analysis of Loss Conductance patterns across layers, training stages, and inference, revealing how neuron contributions evolve in deep architectures. We evaluate how patterns change in better performing models, how Generalized Y-drop impacts them and provide theoretical interpretations of what they imply for the function of deep networks.

1.3 Outline

The rest of this thesis is organized as follows:

- **Chapter 2** introduces fundamental concepts of machine learning and deep learning, providing the theoretical background necessary for the rest of this work.
- **Chapter 3** presents an in-depth overview of explainability metrics, with a particular focus on attribution methods. It showcases the key sub-groups, their motivation, theoretical basis and reviews some important techniques.
- **Chapter 4** reviews the most widely used methods for improving generalization and regularization in deep neural networks, with an emphasis on computer vision archi-

tectures. The chapter discusses different regularization paradigms and representative methods within each category.

- **Chapter 5** introduces the proposed methodology of this thesis. It outlines the motivation behind the work, analyzes the limitations of existing approaches and presents the proposed modifications, metrics, and design choices in detail.

- **Chapter 6** reports the experimental evaluation of the proposed methods. It describes the experimental setup, presents the results and provides detailed analysis and key insights to explain the observed behaviors and performance gains.

- **Chapter 7** concludes the thesis, summarizes the key findings of this work and proposes potential directions for future research.

Chapter 2

Machine Learning

Artificial Intelligence (AI) is a broad field within computer science, originally coined by John McCarthy in 1956, defined as the creation of technologies that enable machines to perform tasks commonly associated with human intelligence such as learning, comprehension, problem solving, decision making, creativity, and autonomy [59]. Machine Learning (ML) is one of the most significant subfields within AI and involves the development of systems that autonomously learn from data. This section will dive into the core ideas behind ML, with emphasis on neural networks and deep learning that drive modern technologies and are central to this work.

2.1 Introduction

Whereas AI is the broader concept of simulating human intelligence, ML is concerned with developing machines that are able to automatically detect patterns in data and use these patterns to predict future data, without the need for explicit programming and predefined rules [13]. A widely used definition by Mitchell describes it as: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [60].

Methods in the field tend to require plentiful data and computational resources to solve tasks effectively. As these became more widely available, traditional statistical learning increased in prominence, combining statistical models and handcrafted features. While effective, these methods required extensive feature engineering and struggled with high-dimensional unstructured data. This limitation motivated the rise of deep learning, which employs artificial neural networks (ANNs) drawing inspiration from biological neural systems and capable of capturing intricate patterns and associations. As defined by Goodfellow et al., deep learning is a class of ML methods that model high-level abstractions in data through hierarchical neural architectures [9]. These techniques have achieved major advances in computer vision, natural language processing, speech recognition, and other domains with rich unstructured inputs, reducing the need for manual feature engineering.

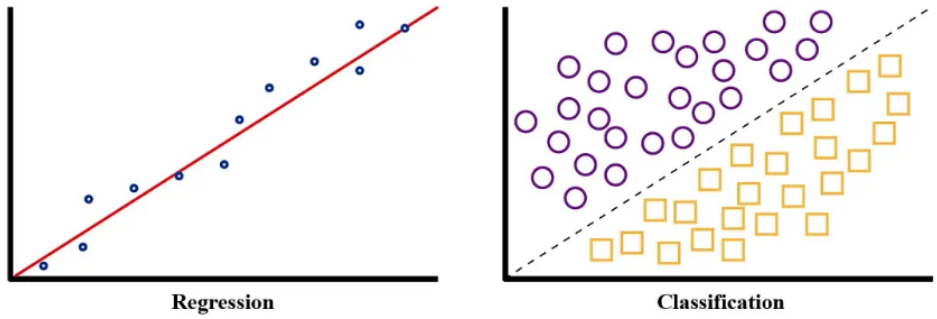


Figure 2.1. *Schematic differences between classification and regression. Picture taken from [1].*

2.2 Machine Learning Variants

Supervised Learning

Supervised learning is a type of machine learning where a model is trained on a labeled dataset to learn the relationship between input variables $\mathbf{x}_i$ and their corresponding targets y_i [14]. Formally, the goal is to approximate an unknown function $f : \mathcal{X} \to \mathcal{Y}$ by leveraging a collection of training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ and use the learned patterns to make predictions on previously unseen or unlabeled inputs [13].

Supervised methods are broadly categorized into *classification* and *regression*, depending on the nature of the target.

Classification. In classification tasks, the objective is to assign inputs to one or more predefined discrete categories. A classifier learns decision boundaries in feature space that separate classes, enabling it to determine the most probable label for a new input based on its characteristics [61].

Classification problems can be further differentiated by the structure of their label space. Depending on the amount of different labels that can be assigned to an input, problems can be split into *Binary classification* problems where only 2 possible labels exist (e.g., disease diagnosis (sick / not sick)) and *Multiclass classification* where more that 2 labels are available (e.g., identifying the animal in an image (cat, dog, horse, bird, etc.)). Additionally multiclass classifications problems can be split based on the amount of labels that can be assigned to an input, with *Single-label classifications* where only a single label is assigned to each input (e.g. disease Diagnosis (covid-19, pneumonia, none , etc)) and *Multi-label classification* where on or more labels are assigned to an input (e.g sentiment classification (“happy” and “excited” at the same time)).

Regression. Regression addresses predictive tasks in which the target variable is continuous. The aim is to estimate the relationship between input features and a numerical output. Learned relationships are often visualized as lines or curves in feature space and are used for numerical predictions on unseen data [13].

Unsupervised Learning

Unsupervised learning tasks a model with uncovering inherent structure, patterns, or relationships within unlabeled data [15]. Unlike supervised learning, models do not map inputs to predefined labels. Instead they reveal hidden patterns, detect anomalies, group similar objects, or discover lower-dimensional representations of the data [13].

Two prominent categories are *clustering* and *association*.

Clustering. Clustering partitions data points into groups,clusters,so that observations in the same cluster are more similar to each other than to those in other clusters, using distance metrics or probabilistic criteria [61]. Clustering Algorithms can be divided into two categories, *hard clustering* where each data point is assigned exclusively into a single cluster and *soft clustering* where each data point is assigned a probability of belonging in each cluster.

Association. Association methods discover relationships and co-occurrence patterns among variables of a dataset. These methods identify rules describing how the presence of certain items/features relates to others, often as *association rules* [62]. They are often used to understand commonalities between variables and identify outliers.

Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning in which an agent learns to make sequential decisions by interacting with an environment. The models learn through a trial and error process, where at each step the agent observes the current state, selects an action, receives a reward (or penalty) indicating the desirability of that action, and transitions to a new state. The repetition of this interaction-feedback process over time enables the agent to discover an optimal policy that maximizes the cumulative reward throughout the decision-making process [16, 17].

2.3 Neural Networks

2.3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a cornerstone in modern machine learning and are the foundation of modern Deep Learning [63]. Inspired by the biological neural networks found in the human brain, which are responsible for learning, recognizing, and organizing information, ANNs were introduced as a mathematical formulation to simulate this process [18, 19]. ANNs consist of computational units called neurons, organized into groups called layers, with an initial input layer that receives the data, one or more hidden layers that process the data, and a final output layer with the final result. Neurons are connected to other neurons through learnable weights that change through training to approximate the underlying function of the data with the minimal possible error.

Each neuron receives a set of inputs,calculates a weighted sum of them, adds a bias term (also referred to as an activation threshold), and applies a non-linear activation function, as expressed in the following equation:

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

where x_i are the inputs, w_i the learnable weights, b the bias term, ϕ the activation function, and y the output of the neuron.

Several different activations ϕ are used to introduce non-linearity in a variety of ways, in order to capture more complex relationships within the data. Some of the most important activation functions are the following.

Sigmoid. Transforms input values into the range $[0, 1]$, often used for binary classification [64].

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Tanh. Transforms input values into the range $[-1, 1]$, often used for zero-centered data [21].

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}.$$

Softmax. Converts a vector of inputs into a normalized probability distribution, making it ideal for multiclass classification.

$$\text{softmax}(\mathbf{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}}, \qquad j = 1, \ldots, K.$$

ReLU. Outputs the input as is if it is positive and outputs 0 if it is negative. It introduces sparsity and reduces vanishing gradients, making it significant for deep networks [65].

$$\text{ReLU}(z) = \max(0, z).$$

SeLU. SeLU applies an exponential activation for negative inputs while linearly scaling positive inputs. It uses two preselected variables $a \approx 1.6733$ and $ȷ \approx 1.0507$ and enables neural networks to self-normalize without the need for normalization layers, thus improving stability in deep architectures. [66].

$$\text{SeLU}(z) = \begin{cases} ȷ\, z, & z > 0, \\ ȷ\, a\,(e^{z} - 1), & z \leq 0\ . \end{cases}$$

GELU. Applies a probabilistic gating mechanism where inputs are weighted by the cumulative distribution function (CDF) of a Gaussian distribution, providing smoother and often more expressive non-linearity than ReLU, particularly effective in modern Transformer architectures [67].

$$\text{GELU}(z) = z\,\Phi(z),$$

where $\Phi(z)$ is the standard normal CDF. A common tanh-based approximation is

$$\mathrm{GELU}(z) \approx \tfrac{1}{2}\, z\Big(1 + \tanh\Big(\sqrt{\tfrac{2}{\pi}}\,(z + 0.044715\, z^3)\Big)\Big).$$

2.3.2 Feedforward Neural Networks

Feedforward Neural Networks represent the simplest form of neural network in terms of connectivity: neurons are arranged in a sequence of layers, and connections flow strictly in one direction,from the input layer toward the output layer,with no backward connections or feedback loops [9]. Each layer produces a new representation of the data, with the input being gradually mapped to increasingly abstract and useful features.

A common and widely used feedforward architecture is the Multilayer Perceptron (MLP), in which every neuron in a layer is fully connected to every neuron in the next layer [68].

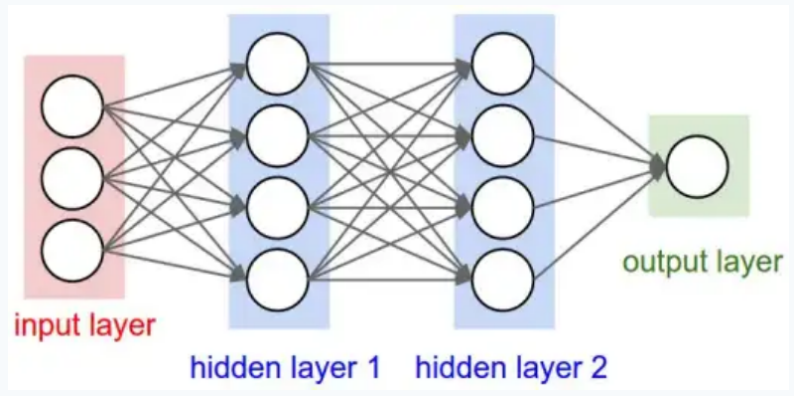


Figure 2.2. *A schematic of a feedforward neural network layer. Picture taken from [2].*

2.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a specialized class of neural networks designed to process data with an inherent grid-like structure, such as images, video frames, or spectrograms [20, 21]. Unlike Feedforward Neural Networks, where each neuron is fully connected to all inputs, CNNs employ the operation of convolution, where a small learnable filter (or kernel) slides across the input and computes localized feature responses. The motivation behind this choice is an important characteristic of natural images, that patterns such as edges, corners and textures tend to be local and appear repeatedly in different spatial positions that do not need to be detected independently. Consequently, weight sharing, the reuse of the same kernel across all spatial locations, enables CNNs to detect a given feature wherever it appears while significantly reducing the number of trainable parameters.

A typical CNN employs a sequence of convolutional layers that learn increasingly abstract representations of the input. Early layers typically respond to simple geometric patterns like edges and textures, while deeper layers integrate broader spatial information to recognize more complex structures, such as object parts or entire objects [22].

The convolutional layers are interleaved with activation functions and pooling operations, like max or average pooling that reduce spatial resolution by summarizing local neighborhoods, which lowers computational cost and encourages robustness to small geometric variations in the input. The behavior of convolutional and pooling operations is controlled by hyperparameters such as stride, which determines how far the filter moves across the input at each step, and padding, which dictates how image borders are handled. These two settings control the amount of spatial detail that is preserved or compressed through the network. After several stages of convolution and pooling, the extracted features are fed into one or more fully connected layers,similar to those in Feedforward Neural Networks,which produce the final predictions for the task.

CNNs were instrumental in the modern resurgence of deep learning, especially in computer vision, due to the success of AlexNet which achieved a breakthrough performance on the ImageNet competition in 2012 and outperformed traditional machine learning approaches [10]. (image showcasing a CNN from here [3])

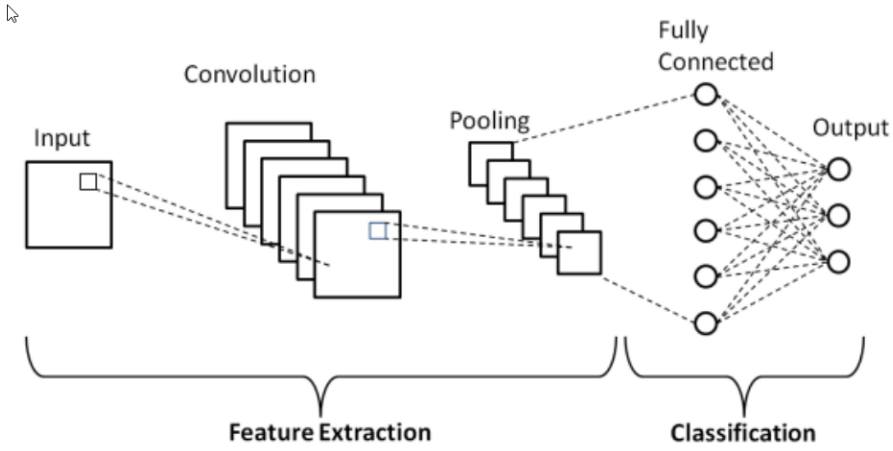


Figure 2.3. *A schematic illustration of a Convolutional Neural Network architecture. Picture taken from [3].*

2.3.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data, where the order data appear in is meaningful and information must be retained across time steps [23]. RNNs differ from feedforward networks by incorporating recurrent connections that return the unit's output back to itself and allow information to persist across time, making them ideal for tasks like language modeling, speech recognition, and time-series forecasting.

At each time step t, the RNN receives an input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$, which contains information from all previous time steps, and uses them to calculate the current hidden state $\mathbf{h}_t$ and an output $\mathbf{o}_t$ as seen in the equations:

$$\mathbf{h}_t = \varphi_h(W_{xh}\,\mathbf{x}_t + W_{hh}\,\mathbf{h}_{t-1} + \mathbf{b}_h),$$

$$\mathbf{o}_t = \varphi_o(W_{ho}\,\mathbf{h}_t + \mathbf{b}_o),$$

where W_{xh}, W_{hh}, and W_{ho} are the input-to-hidden, hidden-to-hidden, and hidden-to-output matrices respectively, $\mathbf{b}_h$ and $\mathbf{b}_o$ are the bias terms, and ϕ_h, ϕ_o the activation functions, with ϕ_h usually being sigmoid or tanh [69]. Similarly to CNNs, RNNs employ weight sharing, with the same weights being used across all timesteps, allowing them to process sequences of arbitrary length and learn temporal patterns that generalize across different positions in a sequence.

While RNNs are able to retain information from long sequences in the hidden state vector, in practice the amount of information is limited due to the fixed dimension of the vector. This results in relevant information being lost over time as well as vanishing and exploding gradients problems during training, with gradients either shrinking to zero or growing uncontrollably respectively as they propagate through many time steps [70].

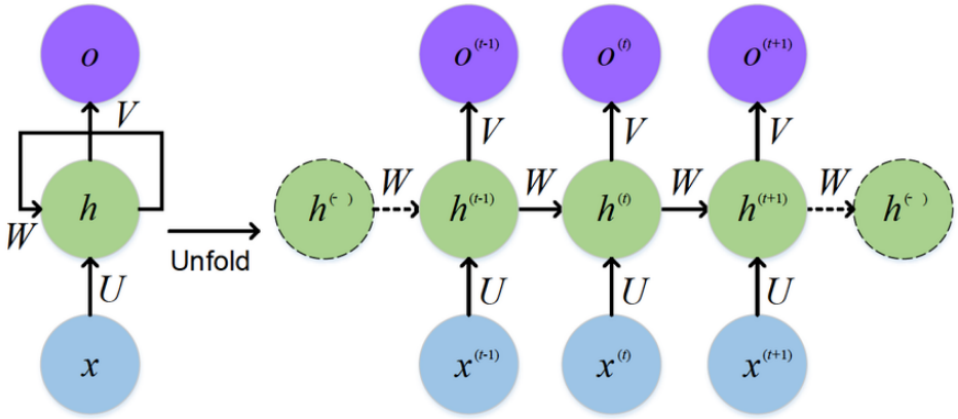


Figure 2.4. *A standard RNN and its unfolded (time-unrolled) representation. Picture taken from [4].*

While several gated RNN variants were introduced like Long Short-Term Memory (LSTM) [71] and Gated Recurrent Units (GRUs) [72], which control how information is stored, updated, and discarded to address these challenges, these models were eventually surpassed in their ability to model long-range dependencies by Transformer architectures described in the next section. Nonetheless RNNs remain a foundational concept in sequential modeling and have significantly influenced the development of modern neural architectures.

2.3.5 Transformers and Attention

Attention The attention mechanism was introduced as a solution to address the limitations RNNs faced when processing long input sequences. Instead of compressing all past information into a single hidden state, attention directly captures dependencies between inputs at different positions in the sequence [24]. The mechanism operates by projecting each token into three learned vectors: a query vector $\mathbf{q}$, which represents the information the token seeks in others, a key vector $\mathbf{k}$, which encodes the relevant information of that token that others may want to attend to and a value vector $\mathbf{v}$, which carries the actual information to be retrieved.

Attention computes a pairwise similarity score between each query and key vector. These scores are scaled by the dimensionality of the key vectors, d_k, and then passed through a softmax function to produce a distribution of attention weights. Finally these weights are used to create a weighted average of value vectors, creating for each token a new context representation of the other relevant tokens that has integrated information from the whole sequence. This can be mathematically formulated as:

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

where $\mathbf{X}$ are all the tokens in the sequence, $\mathbf{Q} = \mathbf{X}W_Q$, $\mathbf{K} = \mathbf{X}W_K$, $\mathbf{V} = \mathbf{X}W_V$ are the query, key, and value projections of all tokens respectively and W_Q, W_K, W_V are the corresponding learned weight matrices for the projections. This formulation allows both short- and long-range dependencies to be captured the same way without the structural constraints or memory bottlenecks found in recurrent architectures.

Multi-head attention To further enhance the ability of attention to capture relevant relationships between tokens, multi-head attention replaces the singular query, key, and value projection of the attention mechanism with several ones called *heads* that learn in parallel. The goal is for each head to learn different kinds of relationships between tokens, such as syntactic dependencies, semantic similarity, global patterns, or more localized ones. The outputs of all heads are concatenated, creating a final more rich representation of the correlations between tokens [5]. Formally,

$$\text{head}_i = \text{Attention}\left(\mathbf{Q}W_Q^{(i)}, \mathbf{K}W_K^{(i)}, \mathbf{V}W_V^{(i)}\right), \qquad \text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]W_O.$$

Transformer The Transformer architecture is built upon the attention mechanism and is composed of a series of layers known as Transformer blocks. Each block contains two main submodules: a Multi-Head Attention mechanism and a Feed-forward network, each followed by a residual (skip) connection, which stabilizes gradient flow by adding the module's input to its output, and a Layer Normalization (LayerNorm) operation, which further enhances training stability in deep architectures [5]. In the original full Transformer, these blocks are organized into two distinct components: an encoder and a decoder. While both consist of stacked Transformer blocks, their functions differ. The encoder applies self-attention over the entire input sequence, producing contextualized representations of the input. The decoder, in contrast, includes a masked self-attention mechanism that prevents each position from attending to future tokens,an essential property for autoregressive generation,as well as an encoder-decoder attention module that conditions the decoder's representations on the encoder's output.

Because attention itself is permutation-invariant and treats all tokens independently of their position, the model augments tokens with positional embeddings. The original Transformer employs sinusoidal positional embeddings, which use deterministic sine and cosine functions to encode position, while more recent variants such as Rotary Positional Embeddings (RoPE) apply position-dependent rotations to queries and keys and

demonstrate improved performance [73]. Although the original Transformer architecture contains both an encoder and a decoder, many later models adopt encoder-only or decoder-only designs depending on the task. Encoder-only models such as BERT focus on bidirectional self-attention to learn contextual representations suited for classification and language understanding [74], while decoder-only models such as GPT use solely stacked masked self-attention blocks for autoregressive text generation [75].

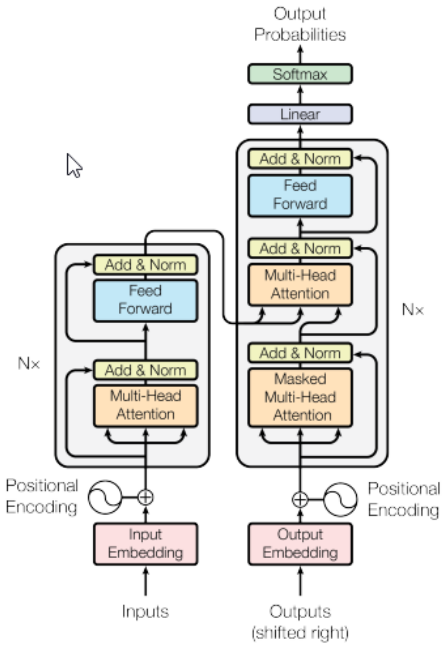


Figure 2.5. *Illustration of the Transformer encoder-decoder architecture. Picture taken from [5].*

Despite the fact that the original Transformer used both encoder and decoder, many influential architectures adopt encoder-only or decoder-only designs depending on the task. Encoder-only models like BERT [74] focus on bidirectional contextual representation learning for classification and understanding tasks, while decoder-only like GPT rely solely on stacked masked self-attention blocks for autoregressive generation [75].

Vision Transformer The Vision Transformer (ViT) is an extension of the Transformer architecture for image recognition tasks by reinterpreting images as sequences of tokens rather than relying on convolutional operations. The input image is split into a grid of fixed-size patches, usually 16×16, which are flattened and projected into an embedding space, allowing the patches to be treated analogously to word tokens in natural language processing. Similarly to the original Transformer, to preserve spatial structure, learnable positional embeddings are added to the patches. The architecture is encoder-only with bidirectional multi-head self-attention, allowing the model to integrate information across all patches, capturing both local interactions and long-range dependencies in parallel. A special learnable token, called the [CLS] token, is prepended to the sequence in order to aggregate global information and its final representation is used to make the final

prediction in classification tasks. The ViT architecture demonstrated that when trained on sufficiently large datasets such as ImageNet-21k or JFT-300M, it can achieve performance comparable to or exceeding that of state-of-the-art convolutional neural networks [25].

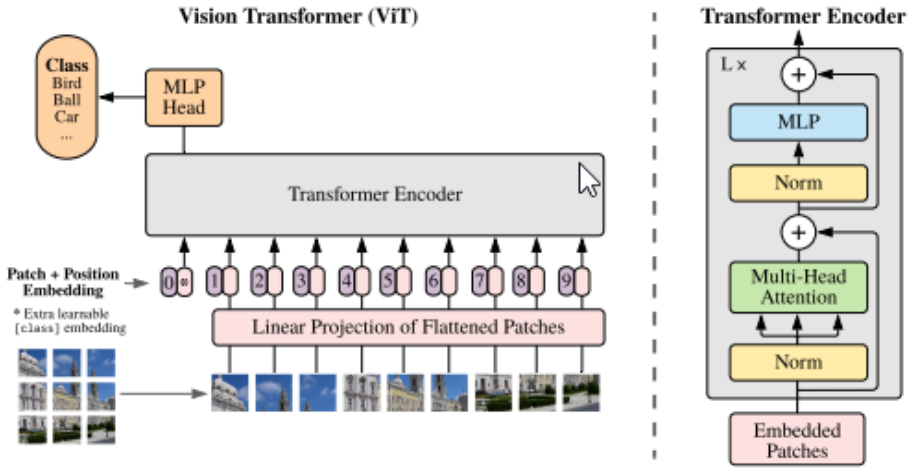


Figure 2.6. *Schematic of the Vision Transformer. Picture taken from[6].*

2.4 Training

2.4.1 Loss Function

A loss function is a measure that quantifies a model's error and acts as a guide to find the optimal model parameters to minimize said error. In supervised learning, it measures the discrepancy between the model's prediction $\hat{y}$ and the true label y, $L(y, \hat{y})$ and based on it a gradient signal is created that updates the model's parameters in a direction that reduces said value, ideally guiding the model towards an optimal state of zero loss [9]. Different loss functions are used for different tasks, with regression problems commonly employing Mean Squared Error (MSE) or Mean Absolute Error(MAE), which penalize large deviations from the target value, while classification tasks typically using Cross-Entropy(CE) loss which encourages high confidence in the correct classes [26].

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$\mathcal{L}_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|$$

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\,\log \hat{y}_{i,k}$$

2.4.2 Optimization

Optimization is the iterative procedure of adjusting the network's parameters in order to find the optimal set of values that minimize the selected loss function. In neural

network architectures this process takes place using gradient based methods, where the gradient of the loss with respect to each parameter is calculated and guides how the parameter should change. The foundational algorithm for this process is Gradient Descent, which updates the parameters in the opposite direction of the loss gradient computed over the entire dataset. Since this approach is costly, most modern neural models use Stochastic Gradient Descent [27] or some of its more modern refinements like momentum [76], Adam [77] or AdamW [78] instead, which approximate the gradient using small subsets of the data called mini-batches. The training process is further influenced by the selection of key hyperparameters such as batch size, learning rate and momentum coefficients as well as strategies like learning rate warm-up, learning rate scheduling and gradient clipping that are critical for ensuring stable and efficient optimization.

Chapter 3

Explainability Metrics

3.1 Introduction

With machine learning models drastically increasing in size and complexity, understanding how a model makes its final decision has become really complicated. Especially in deep learning where, unlike traditional statistical models, neural networks often function as "black box" systems whose decision making process is not interpretable due to their deep, high-dimensional and nonlinear nature [11]. As understanding these models can be crucial especially when applied in critical fields like healthcare, autonomous vehicles or criminal justice where trust, transparency, fairness, robustness and regulatory compliance are essential many methods have been developed to solve this issue [28, 29]. These methods have been grouped under the term *Explainability Metrics* inside the general field of Explainable Artificial Intelligence (XAI) and are a central research topic in modern ML. A large subset of these methods can be formally described as attribution methods, which aim to assign importance scores to components of a mode input features, neurons, layers or learned representations contribute to the model's final prediction in a way that humans can interpret, not only for safety and accountability but also for scientific insight, debugging, and model refinement [30]. Formally, given a model $f : \mathcal{X} \to \mathcal{Y}$ and an input $x \in \mathcal{X}$, an attribution method is a functional $\Phi : \mathcal{F} \times \mathcal{X} \to \mathbb{R}^{|\mathcal{X}|}$, which returns $\gamma = \Phi(f, x) \in \mathbb{R}^{|\mathcal{X}|}$, (with $f \in \mathcal{F}$) that represents an attribution map that explains the prediction of f for input x, with larger γ indicates greater influence for a variable. In the following sections the main groups of attribution-based explainability metrics will be presented along with short explanations of key methods from each group.

3.2 Occlusion-Based Methods

Occlusion-based or Perturbation-based methods aim to identify important parts/regions of the input by modifying them or removing them and measuring the change in the model's prediction. They provide a model-agnostic way to create robust and interpretable attribution maps but require a multitude of forward passes to calculate.

3.2.1 Occlusion Sensitivity

Occlusion sensitivity is one of the earliest approaches in the field. It employs a sliding window square mask that zeroes a region of the input image and compares how much the confidence in the target class changed compared to the full input. Squares that cause the largest shift in prediction are considered the most influential and the values of the all masks are considered an indicator of the model's overall reliance on local image content [22].

3.2.2 RISE

RISE (Randomized Input Sampling for Explanation) uses a number of random masks that zero out pixels of the original input. For each mask it calculates how much the prediction shifted and creates a weighted average of all masks to see how much each pixel affected the final prediction. This results in a smoother and more robust final pixel-importance map [31].

3.3 Gradient-Based Methods

Gradient-based explainability metrics aim to measure feature importance by using the gradient of the model with respect to the input as a basis. The key idea behind such metrics is that high gradient values for a certain feature would imply that minuscule changes in that feature greatly affect the final output, thus making that feature important. They are computationally efficient methods as deep learning architectures calculate gradients through back-propagation, and various refinements to the core ideas have been suggested to combat issues such as noise, saturation, and lack of local sensitivity, as well as to provide more theoretical guarantees [32].

3.3.1 Vanilla Gradients (Saliency Maps).

The simplest gradient-based method computes the absolute value of the gradient of the model's output with respect to each input feature, creating a saliency map that showcases feature influence. However, this formulation obscures information about the sign of the gradients as well as being susceptible to noise, something that motivated the development of better alternatives [33].

3.3.2 Input x Gradient.

Input × Gradient was introduced as a way to produce sharper and more interpretable attribution maps than saliency maps. This formulation captures both local sensitivity and the sign of the gradient of each feature and acts as a form of first-order Taylor decomposition. Nonetheless, it still faces issues with gradient noise and saturation [34]. The attribution is:

$$S_{\text{IxG}}(x) = x_i \frac{\partial F(x)}{\partial x_i}$$

where $\odot$ denotes elementwise multiplication.

3.3.3 SmoothGrad.

SmoothGrad is a technique with the goal of reducing the noise present in saliency maps that sometimes makes them difficult to interpret. The cause of such noise was attributed to meaningless local variations in partial derivatives, and the suggested solution was creating a local average of gradient values. This was achieved by calculating the gradient of multiple noisy versions of the input and taking their average, which smoothened out the mentioned artifacts and resulted in more stable and coherent attribution maps [79]. With noise $\epsilon_k \sim \mathcal{N}(0, \sigma^2 I)$, the estimate is:

$$S_{\mathrm{SG}}(x) \;=\; \frac{1}{N}\sum_{k=1}^{N} \frac{\partial F(x+\epsilon_k)}{\partial x_i}.$$

3.3.4 Integrated Gradients.

Integrated Gradients (IG) were proposed as a solution to some key limitations of vanilla gradients such as their susceptibility to gradient saturation, as well as to provide theoretical grounding to the positive traits of the metric. Instead of a single gradient, IG calculates influence by accumulating gradient information across a linear continuous path from a baseline x', usually a black image, to the actual input x [35]. This interpolation helps alleviate the saturation problems by avoiding flat points caused by ReLU activations that would be an issue when using a single gradient. Moreover, this formulation fulfills desirable axioms such as:

Sensitivity: For every input and baseline that differ in at least one feature and produce different outputs, the feature should have a non-zero attribution.

Implementation Invariance: Two networks that produce equal outputs for all inputs should produce the same attributions despite possibly different implementations.

Completeness: The sum of all feature attributions is equal to the difference between the model's output for the input and the baseline.

Formally, the IG attribution for feature i is

$$\mathrm{IG}_i(F, x; x') \;=\; (x_i - x_i') \int_0^1 \frac{\partial F(x' + a\,(x - x'))}{\partial x_i}\, da.$$

These features made IG a robust and efficient attribution method with a deeper theoretical grounding than most methods. Many papers have proposed even further improvements on the method. Different baselines have been proposed, like a Gaussian baseline and a Gaussian-blurred image baseline [80], maximum-distance and uniform baselines [81], as well as Expected Gradients that propose using a variety of baselines rather than using a single reference [82]. Other researchers have focused on improving upon the path between the baseline and input, like geodesic integrated gradients that treat the input space as a Riemannian manifold and integrate along a learned "path of least resistance" [83], IG^2, which adjusts the integration path and baseline in order to use both the explicand and counterfactual gradients to reduce attribution noise and arbitrary baseline selection [84] and Guided Integrated Gradients that adaptively select integration

paths shaped by gradient information [85]. Lastly, Transformer-specific techniques like Sequential Integrated Gradients suggest integrating one token at a time while keeping the remaining tokens fixed to obtain more semantically valid attributions [86].

3.3.5 Conductance

Conductance is a metric introduced by Dhamdhere, Sundararajan, and Yan [12] as an extension of the ideas presented by Integrated Gradients [35] as a way to attribute a model's prediction to its hidden units, not just the input features.

Conceptually, conductance can be interpreted as measuring the "flow" of attribution that passes through a neuron when calculating IG. Similarly to IG, it attributes the model's prediction by integrating along a continuous path from a baseline to the actual input but it decomposes the IG computation via chain rule to propagate attributions across the network's layers and assigns a value to each neuron representing its contribution.

Let $F : \mathbb{R}^n \to [0, 1]$ denote the model's output, $x \in \mathbb{R}^n$ an input, and $x' \in \mathbb{R}^n$ a baseline.

For a hidden neuron k in a layer l, it's conductance *with respect to feature* i is:

$$\mathrm{Y}^{(l)}_{k,i}(F, x; x') \;=\; (x_i - x'_i) \int_0^1 \frac{\partial F(x' + a\,(x - x'))}{\partial y^{(l)}_k} \, \frac{\partial y^{(l)}_k}{\partial x_i} \, da, \tag{3.1}$$

where $y^{(l)}_k$ is the output of neuron k in layer l given input $x + a\,(x - x')$ and $Y^{(l)}_{k,i}(F, x; x')$ is its conductance with respect to feature i.

The *total conductance* of neuron k is the summation over all input features:

$$\mathrm{Y}^{(l)}_{k}(F, x; x') \;=\; \sum_{i=1}^{n} (x_i - x'_i) \int_0^1 \frac{\partial F(x' + a\,(x - x'))}{\partial y^{(l)}_k} \, \frac{\partial y^{(l)}_k}{\partial x_i} \, da. \tag{3.2}$$

This formulation inherits desirable properties of IG such as completeness, linearity, and sensitivity, while also helping to alleviate problems caused by gradient saturation and ensuring correct polarity of the attribution [35, 12].

Conductance values can be positive or negative, with positive values indicating that a neuron contributes positively to the output prediction, negative values indicating a negative contribution, and values close to zero suggesting minimal influence.

Practical approximation. To make computation practical, an approximation showcased in [87] is used. The continuous path integrals are approximated with a Riemann sum and the fact that total conductance is equivalent to path integrated gradients on the neuron itself is showcased. With those approximations, they present an estimator that avoids summing over input dimensions and accumulates, at each step, the product of the gradient of the output with respect to the neuron and the change in the neuron's activation from the previous step. Let $x^{(j)} = x' + \frac{j}{n_c}(x - x')$ be n_c evenly spaced points on the straight-line path from baseline to input, and let $F^{(l)}_k(x)$ be the activation of neuron k in layer l at input x.

With these alterations, conductance can be approximated as:

$$\mathrm{Y}_k^{(l)}(F, x; x') \approx \sum_{j=1}^{n_c} \frac{\partial F(x^{(j)})}{\partial k} \left(F_k^{(l)}(x^{(j)}) - F_k^{(l)}(x^{(j-1)}) \right). \tag{3.3}$$

From this point onward, we simplify the notation by omitting the baseline x' and model F when clear from context, signifying conductance of neuron k in layer l as $\mathrm{Y}_k^{(l)}(x)$.

3.4 Propagation/Decomposition-Based Methods

Propagation-based or decomposition-based attribution methods work by redistributing the output relevance backward through the network, assigning each neuron a score all the way up to the initial input feature. For each architecture and layer type they employ custom backpropagation rules to simulate how information flows through the network instead of relying on the standard gradient rules while ensuring that the total relevance is the same as in the output, also known as *relevance conservation*. These methods have gained popularity due to the custom propagation rules being able to avoid issues plaguing gradient-based methods like gradient saturation. In this section a brief overview of some key decomposition methods will take place [36].

3.4.1 Layer-wise Relevance Propagation (LRP).

Layer-wise Relevance Propagation (LRP) is one of the most important decomposition techniques. It is used to propagate the model's prediction down to initial features through various redistribution rules that all comply with the relevance conservation property [37]. The conservation reads

$$\sum_i R_i^{(l)} = \sum_j R_j^{(l+1)},$$

where $R_i^{(l)}$ is the relevance assigned to neuron i at layer l.

There are many options of propagation rules that can be assigned like the core epsilon rule (LRP-ε) or the gamma rule (LRP-γ) that favors positive contributions over negative ones, among many others. Let $a_i^{(l)}$ be the activation of neuron i at layer l, w_{ij} the weight to neuron j at layer $l+1$, and $z_{ij} = a_i^{(l)} w_{ij}$. Two widely used rules are:

$$\text{LRP-}\varepsilon: \quad R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \varepsilon \cdot \mathrm{sign}(\sum_{i'} z_{i'j})} R_j^{(l+1)},$$

$$\text{LRP-}\gamma: \quad R_i^{(l)} = \sum_j \frac{a_i^{(l)}(w_{ij} + \gamma\, w_{ij}^+)}{\sum_{i'} a_{i'}^{(l)}(w_{i'j} + \gamma\, w_{i'j}^+)} R_j^{(l+1)}, \quad w^+ = \max(w, 0),$$

with hyperparameters $\varepsilon \geq 0$ and $\gamma \geq 0$ controlling numerical stability and emphasis on positive evidence, respectively. The amount of possible rules as well as the selection of hyperparameters controlling them make LRP a flexible method that can be customized for every architecture and allows for sharp and stable attribution maps, especially in computer vision [88]. Many works have proposed custom rules to improve upon the basic

LRP formulation, with notable works like Transformer-specific LRP or Attn-LRP extending the formulation to attention architectures [89, 90, 91].

3.4.2 DeepLIFT.

DeepLIFT is a decomposition technique that compares the relevance of neurons to ones of a baseline input. Specifically, it uses chain rules analogous to the gradient chain rules but instead uses the *differences* between the neuron activations (or outputs) between the input and the baseline, while ensuring that the sum of contributions to the output is equal to the difference of output and the baseline. This formulation allows DeepLIFT to avoid gradients and all the problems that come with them like saturation and flat areas [34]. Let x' be a reference (baseline) input, and define $\Delta x = x - x'$, $\Delta y = y(x) - y(x')$. DeepLIFT assigns contributions $C_{\Delta x_i \to \Delta y}$ such that

$$\sum_i C_{\Delta x_i \to \Delta y} \;=\; \Delta y,$$

and propagates *multipliers* $m_{i \to j} = \frac{C_{\Delta x_i \to \Delta y_j}}{\Delta x_i}$ through the network via a chain rule analogous to backpropagation.

Chapter 4

Generalization and Regularization

4.1 Introduction

The main objective when training a neural network is to ensure that it performs well not only on the data it has seen during training but also on previously unseen inputs. This capability is called generalization and showcases whether the model has learned meaningful underlying patterns in the data rather than just memorizing the training examples. To estimate this, in most cases, a small subset of the data is withheld from training and used as a test set, providing an unbiased sample for evaluating performance on unseen data. Generalization can fail in two ways, underfitting and overfitting. Underfitting occurs when a model is too simple or inappropriately trained, causing it to fail to capture the structure of the data and resulting in poor performance on both the training and test sets, also referred to as high bias. On the other hand, overfitting occurs when a model learns train-set-specific and noisy patterns, resulting in it performing well on the train set but struggling on the test set. This is common in more complex architectures and is also referred to as high variance. This interplay, known as the bias–variance trade-off [92], represents a central challenge in the training of machine learning models, particularly deep neural networks, and practitioners are tasked to achieve a balance between the two. To combat overfitting, several regularization techniques have been proposed to shape and constrain the optimization process, encouraging models to learn more robust and generalizable representations. This is especially critical in computer vision, where visual input showcases great variability due to factors such as viewpoint, illumination, occlusion, deformation, and noise in images, and it is crucial that the model does not memorize them. In the following section, several key regularization strategies for neural networks will be presented, with particular emphasis on techniques that have proven effective in modern computer vision systems.

4.2 Loss-Based Regularization

Loss-based regularization refers to techniques that add penalty terms to the loss function or change how it is calculated in order to discourage the model from certain behaviours and guide it to a more stable and generalizable state.

4.2.1 L2 Regularization

L2 regularization, also known as weight decay, adds the L2 norm of the parameters to the loss function in order to discourage large weights and create a smooth parameter space. This has been shown to improve stability and generalizability, especially in deep networks and has been one of the most common techniques in modern architectures [38].

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \|w\|_2^2$$

4.2.2 L1 Regularization

L1 regularization switches the L2 norm of weight decay with the L1 norm of all parameters to be added in the loss function. The key difference of this change is that it can lead certain weights to become exactly zero, thus encouraging sparser solutions and a form of feature selection [39].

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \|w\|_1$$

4.2.3 Label Smoothing

Label smoothing is a method to discourage overconfidence in classification tasks. It replaces the target label from being one-hot to a smoother distribution, where the correct class has a probability close to 1 and the other wrong classes are given a small probability. This was shown to improve calibration and generalization in large scale vision models [40]. Let K be the number of classes and $\varepsilon \in [0, 1)$ the smoothing factor. For class k^*,

$$y_k = \begin{cases} 1 - \varepsilon, & k = k^*, \\ \dfrac{\varepsilon}{K - 1}, & k \neq k^*, \end{cases} \qquad \mathcal{L}_{\text{CE}}^{\text{LS}} = -\sum_{k=1}^{K} y_k \log \hat{y}_k.$$

4.3 Normalization Techniques

Normalization techniques are used to smooth training by reducing the change in the distribution of activations, also known as internal covariate shift. These techniques enable higher learning rates and result in faster, more stable optimization processes that avoid problems like vanishing or exploding gradients.

4.3.1 Batch Normalization

Batch Normalization (BatchNorm) normalizes the activations within each mini-batch using the mean and variance of all the samples within it. Then, using learnable parameters γ, β, it allows the model to learn the optimal scaling and shifting of the normalized activations [41].

4.3.2 Layer Normalization

Layer Normalization normalizes the activations across features in each individual sample. Similarly to BatchNorm, the mean and variance of all features are calculated, as well

as learnable parameters γ, β for optimal scaling and shifting. This technique is key to ensure smooth gradient propagation in deeper architectures,especially in Transformer architectures due to their depth and use of skip connections [42].

4.4 Data Augmentations

Data Augmentation is a regularization technique that artificially increases the diversity of the dataset without having to collect new data. It is widely applied in computer vision where various transformations are applied to an image while keeping the original label, something that improves generalizability as the model becomes more invariant to small alterations in scale, geometry, color, and structure, etc. Traditional transformations such as rotation, flipping, scaling, translation, and contrast/brightness adjustment expand the training distribution and are critical for preventing overfitting and data memorization, especially in high-capacity models. In the following section, some commonly applied image data augmentation techniques will be presented [43].

4.4.1 AutoAugment

AutoAugment is a method that automatically learns an optimal augmentation policy to be used during training from a dataset using reinforcement learning. An LSTM is trained to sample augmentation policies (like translation, inversion, color shift, etc.) and a child model's performance on the validation set is used as a reward signal for the effectiveness of the policies. This process, through multiple iterations of training using the Proximal Policy Optimization algorithm [93], finds an optimal augmentation strategy for the dataset. This method showcased that augmentation can also be viewed as a form of optimization and although expensive, it produces improved training results across multiple datasets [44].

4.4.2 RandAugment

RandAugment is a simplified version of AutoAugment that removes the separate search phase and reduces the search space of possible augmentation strategies. Instead, only two hyperparameters should be tuned: the number of augmentations to apply per image N (sampled from a uniform set of transforms) and a global magnitude value M, which controls how severe the distortion caused by each transformation will be. This method showcased competitive results to AutoAugment across various vision benchmarks while being a lot more cost efficient [45].



Figure 4.1. *Illustrative examples of randomly selected transforms nn RandAugment. Picture taken from [6].*

4.4.3 Repeated Augmentation.

Repeated Augmentation is a method that suggests multiple differently augmented variants of the same image to be used in the same mini-batch, instead of one single augmented variant being in each training epoch. The model is forced more to simultaneously learn more invariances, something that encourages prediction consistency across augmentations. This approach has been shown to improve generalization and training stability, particularly in large-scale vision models [94].

4.4.4 Cutout

Cutout is a technique that helps increase a model's robustness to occlusion and missing information by randomly dropping a fixed-size square region of the image and replacing it with zeros. It was shown to decrease over-reliance on specific local features and encourage more distributed representations [95].

4.4.5 Random Erasing

Random Erasing is a method that generalizes the core ideas of Cutout by removing rectangular regions of variable sizes and aspect ratios and replacing them with random noise or constant values. This more flexible strategy showcased improved generalization compared to standard Cutout [96].

4.4.6 Mixup

Mixup is a data augmentation technique that creates new training samples by creating a linear interpolation of two other training samples. A weighted average of both the images as well as their respective labels is created, something that has been shown to result in smoother decision boundaries between labels as well as reducing the impact of corrupt labels. The method acts as a way to fill the gap between the original data points in the

feature space and promotes linear behavior between classes, resulting in more global and consistent patterns being discovered and in turn improved regularization [46].

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \qquad \tilde{y} = \lambda y_i + (1-\lambda) y_j,$$

4.4.7 CutMix

CutMix, like Mixup, is a mixed-sample augmentation method where a rectangular region of an image is replaced with a same-sized region from another image. The newly created image is assigned a label that is a weighted average of the original labels, proportionally to the cut area. It has been shown to improve localization ability, encouraging the model to discover multiple discriminative regions, while also keeping spatial structure and improving robustness to outliers [47].

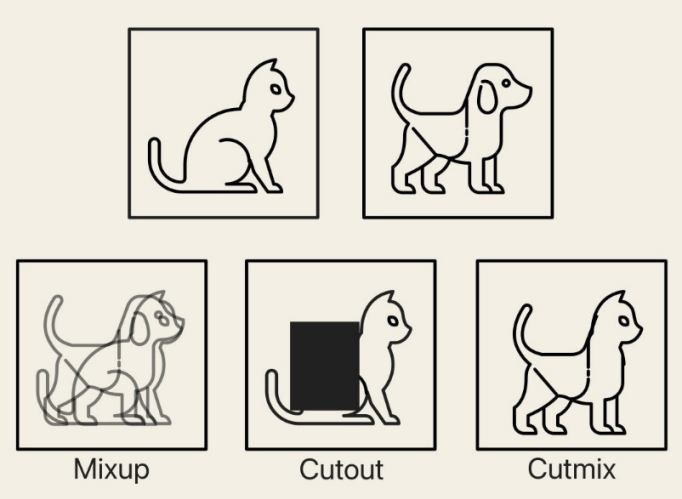


Figure 4.2. *Visual comparison of Cutout, Mixup, and CutMix augmentations. Picture taken from [7].*

4.5 Drop-Based Regularization

Drop-based techniques add stochasticity in the training process by randomly deactivating units, connections, or entire submodules of a neural network. This reduces feature co-adaptation within the network, thus reducing wastage of network resources as well as the influence of spurious features, resulting in a more robust and generalizable model.

4.5.1 Dropout

Dropout is a technique that randomly zeroes the values of a subset of activations during training, with each activation being independently dropped. Each activation has the same probability of being dropped, called dropout rate p, and the remaining activations are scaled by $1/(1-p)$ to retain the expected magnitude of the layer.

This process results in less co-adaptation of features between neurons and forces the network to distribute learned representations across multiple pathways rather than

relying excessively on a small subset. As a result, the model uses more of its capacity and overfitting is reduced by promoting more robust and redundant representations [48].

Another more probabilistic interpretation of the method, introduced by Gal and Ghahramani, demonstrates that dropout can be viewed as an approximate form of Bayesian inference, where the stochastic dropout masks implicitly define a variational distribution over the network's weights. This makes training using dropout a form of minimizing the divergence between an approximate posterior and the true Bayesian posterior, providing a theoretical justification for its regularization effect [97].

Generally, the mechanism is disabled during inference to produce deterministic predictions, although an alternative approach known as Monte Carlo Dropout keeps it active at inference and performs multiple stochastic predictions, enabling estimation of model uncertainty [97].

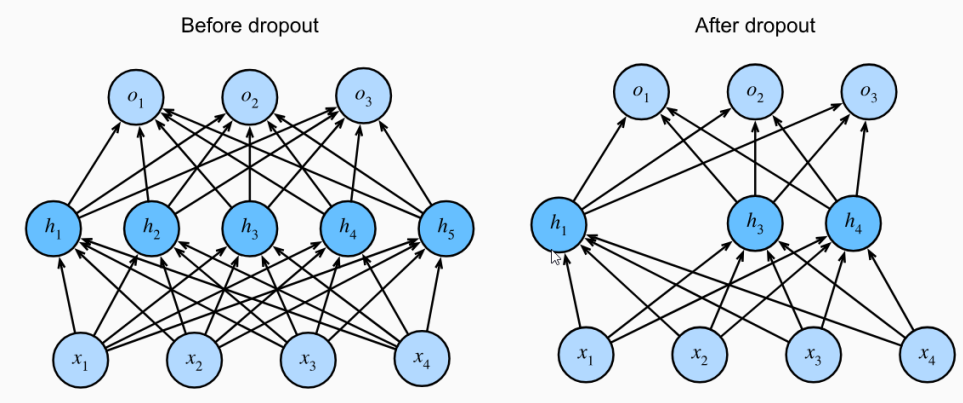


Figure 4.3. *Illustration of an MLP before and after applying dropout. Picture taken from Dive into Deep Learning (d2l.ai).*

Numerous dropout variants have been proposed to better match the properties of specific neural network architectures. In CNNs, DropBlock removes contiguous feature-map regions instead, addressing spatial correlation and preventing information from leaking through neighboring activations, while in RNNs Gal and Ghahramani suggest a fixed dropout mask across all time steps, producing more temporally consistent and stable training [98, 99].

Other variants focus on training stability and feature statistics. AlphaDropout maintains the self-normalizing properties of networks using SELU activations by preserving activation mean and variance [66] while DropCov adapts dropout rates based on feature dimensionality, importance, and correlation prior to global covariance pooling [100].

Scheduled approaches have also been proposed with Annealed dropout gradually reducing dropout rates during training, with the goal of taking advantage of a larger effective network during later training iterations, while Curriculum Dropout argues in favour of increasing dropout probability during training, as stronger regularization is needed later in the training to avoid overfitting [101, 102].

4.5.2 Standout.

Standout is one of the first proposed importance-aware and adaptive dropout mechanisms. Rather than applying a fixed dropout probability uniformly across neurons, it introduces learned, data-dependent dropout rates by gating each neuron's activation through a logistic function with trainable parameters optimized via gradient descent. This formulation allows dropout probabilities to evolve during training and reflect neuron importance dynamically. Standout demonstrated improved performance over fixed-rate dropout, but these gains come at the expense of increased computational complexity and additional trainable parameters.

Future works like Concrete Dropout expanded on the idea of learned adaptive dropout, enabling learning of dropout probabilities via a continuous relaxation of Bernoulli variables, providing a principled Bayesian interpretation as approximate variational inference. This methodology further enhanced generalization without manual tuning by adapting regularization strength to model uncertainty [58, 103].

4.5.3 R-drop.

With the goal of lessening the discrepancy between inference and training, R-drop employs two different forward passes of the same input with different random dropout and adds a KL-divergence penalty between predictions in the loss functions. This method enforces consistency under dropout stochasticity, resulting in less uncertainty and better performance in inference [104].

This idea was further enhanced in future works like LR-drop that includes additional MSE penalties between the attention heads and feedforward activations of the two forward passes, achieving additional consistency and performance gains [105].

4.5.4 Drop-Path (Stochastic Depth)

Drop-Path, also referred to as Stochastic Depth, extends the concepts of dropout from individual activations to entire residual branches or layers. During training, for each sample, whole residual paths are randomly skipped with probability p, causing each forward pass to sample sub-networks of varying depth. This alleviates difficulties in optimization found in deep architectures by reducing the effective depth during training while keeping the full-depth model at inference time [49].

4.5.5 Y-Drop

Y-Drop is a regularization method with the goal to convert the dropout algorithm from a completely random procedure into one that detects *important* neurons and drops them at a higher rate. Importance is estimated using *neuron conductance*, a task-agnostic metric that attributes a model's final prediction to the network's hidden units that was in section 3.3.5. Y-Drop assigns higher dropout rates to neurons with higher conductance in a layer, forcing the network to solve the task at hand without some of its most important units. This process encourages comparatively weaker neurons that may have been unused or

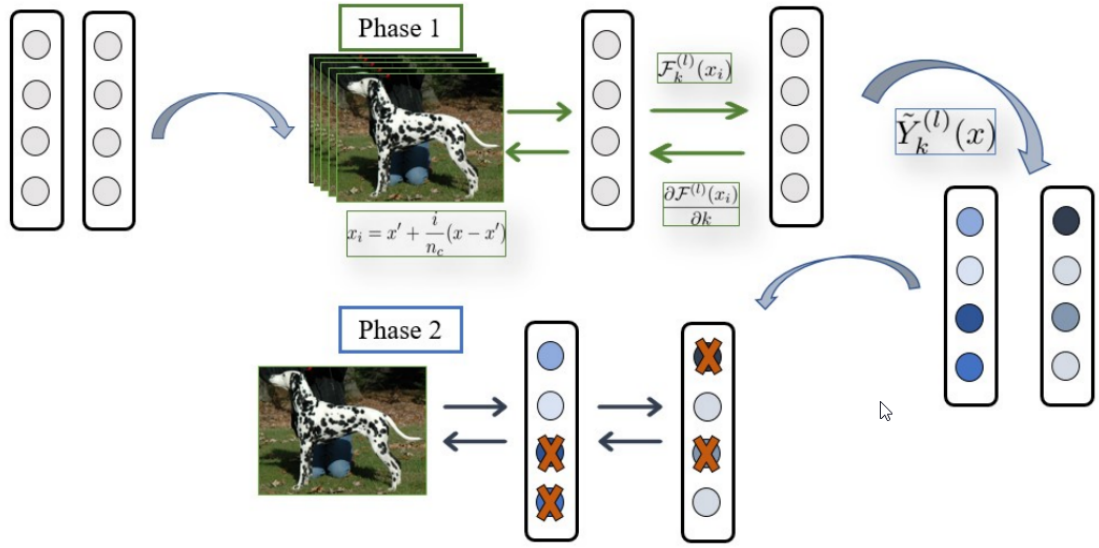


Figure 4.4. *Overview of the Y-drop procedure with conductance-based ranking and bucketed dropout. Taken from [8].*

underutilized to participate in the final prediction and learn critical information, with the goal of achieveing greater regularization and generalization [8].

To achieve this, the method is split in two phases. A conductance estimation/neuron ranking phase and a conductance guided-dropout phase. In the first phase a subset of the batch with B_c images is used to compute the conductance of all hidden units for each image in the selected linear layers. The mean conductance across the B_c images is computed to get a per-unit mean conductance value that reflects its overall contribution to the model's predictions. Using this value, the neurons in each layer l are ranked and split into two buckets, a strong bucket B_s containing the top $w_s \times 100\%$ of neurons with the highest mean conductance and a weak bucket containing the remaining $w_w \times 100\%$ of neurons, where $w_s, w_e \in [0, 1]$ and $w_s + w_w = 1$

In the second phase, these buckets are used to assign adaptive dropout probabilities to each neuron. Neurons in the strong bucket B_s are assigned higher dropout rates p_h, while neurons in the weak bucket B_w receive lower dropout rates p_l. Specifically, following the findings of [106], instead of using fixed dropout rates, the rates are sampled from the right half of a Gaussian distribution, which was shown to improve generalization. For each bucket, the per-neuron dropout rates are drawn from the distributions

$$p_l \sim \mathcal{N}^+(\mu_L, \sigma_L), \qquad p_h \sim \mathcal{N}^+(\mu_H, \sigma_H),$$

respectively, where μ_L, μ_H are the means and σ_L, σ_H the standard deviations for the each of the buckets distributions. The sampled values are then clipped by predefined maximum thresholds $p_l^{\max}$ and $p_h^{\max}$, since overly large dropout rates were shown to be damaging.

After the dropout rates have been determined, the model performs it's normal forward-backward pass and updates its weights. This process is repeated in every training iteration after the first $K\%$ training epochs. In the first $K\%$ of training epochs, standard

dropout is used since initially the weights are randomly initialized and the resulting activations and conductance values are not yet meaningful, making Y-drop useless or maybe harmful. These $K\%$ epochs are called annealing factor. An additional implementation detail is that because the dropout of each neuron changes at every iteration, the usual dropout rescaling trick cannot rely on a single predetermined p. To address this, Y-drop maintains the mean of each neuron's dropout rate using a exponential movemening average controlled by a parameter a called elasticity. The mean dropout rate in each neuron is initialized as the average value of the 2 dropout rate means of the buckets and is updated in every iteration.

The method was applied to the linear layers of multiple CNN architectures and was evaluated on various datasets showcasing substantial improvements in regularization and performance gains compared to standard dropout. Additionally, it was shown that the benefits tend to increase as the model size increases, indicating that Y-drop effectively leverages the additional capacity of larger networks. Lastly, a toy experiment on a single linear layer was conducted showcasing that the use of Y-drop resulted in more balanced conductance scores and a higher conductance for the whole layer, further indicating that the method encourages a more distributed representation across neurons.

Chapter 5

Generalized Y-drop

5.1 Introduction

This chapter will describe the main contributions of this thesis. It outlines the general motivation of investigating the field of regularization through explainability metrics and specifically the Y-drop method. We will describe what we consider the main limitations of the Y-drop method and describe the various methods we introduced to alleviate them and expand upon the original method. Each section describes in detail what was tested, the ideas behind the choice and a short mention of the results it produced, before showcasing them in a more detailed and descriptive way in section 6.

5.2 General Motivation

Explainability metrics are not solely meant to give human insight into a model's behaviour but can also serve as levers to *change and improve* that behaviour. A common application in previous works is *data debugging*, where attributions help trace erroneous predictions back to harmful, mislabelled, or outlier training examples, either at the example or the feature level. Those insights can then guide the removal, relabeling, or reweighting of data to yield a cleaner dataset that will allow models trained on it to improve downstream performance [51, 52, 53].

Another widely used application is *model compression and pruning*. Attribution- or saliency-based scores are used to identify weights, filters, layers, or attention heads that contribute negatively or only negligibly to the final prediction. Pruning these components yields smaller models that require less memory and have quicker inference times, often with minimal accuracy loss and sometimes even accuracy gains [54, 55, 56, 57].

While these uses mostly apply explainability *before* training (to curate data) or *after* training (to prune models), there is a significant opportunity to inject *explanation feedback during training* to steer optimization by penalizing undesirable behaviors or rewarding desirable ones according to information provided by attribution signals. Some prior work on the matter relies on external guidance (e.g., expert masks or human-defined concepts) [107, 108], but in this work we focus more on *model-intrinsic* metrics that provide direct feedback of the model's internal process.

A method that does something as described before is the Y-drop method [8], described

in detail in section 4.5.5 , which uses the Conductance metric to identify neurons that play a key role in the final prediction and assigns them higher dropout rates. This was shown to achieve improved regularization over standard dropout and incentivize higher neuron participation in the final prediction.

Despite the clear benefits showcased, several design choices leave room for refinement to make the method more easily applied across architectures with less hyperparameter tuning, while achieving larger performance gains at a reduced computational cost. In this chapter, we will examine some of the key limitations of the Y-drop framework as well as our proposed methods to alleviate them. Moreover, we will describe how the method was applied to the ViT architecture.

5.3 Issues with Y-drop

This section presents some of what we consider to be the key limitations of the Y-drop method.

Parameter Tuning. One of the most notable issues lies in the way dropout rates are assigned. Specifically, Y-drop relies on the creation of two fixed-size buckets, each with its own mean dropout rate. This formulation makes parameter optimization significantly more complex. Unlike standard dropout, which requires tuning a single dropout rate, Y-drop necessitates the adjustment of three key hyperparameters: the mean dropout rate for the strong bucket p_h , the dropout rate for the weak bucket p_l, and the proportion of neurons assigned to each group w_s, w_w, as well as secondary parameters that require tuning like the maximum dropout rates and deviation of each bucket. This added complexity makes tuning more difficult and computationally expensive.

Use of Conductance Information. The method does not fully leverage the information provided by conductance-based attributions. By ranking neurons and dividing them into only two groups, the approach may over-penalize certain neurons. For instance, neurons with only slightly higher conductance scores than their neighbors may still be placed in the "strong" bucket and subjected to disproportionately higher dropout rates. A more nuanced method, one that accounts for the relative differences in conductance between neurons and assigns dropout rates accordingly, could lead to smoother training dynamics and improved generalization.

Comparison with other methods. Another challenge lies in comparing Y-drop to alternative methods. Because Y-drop introduces Gaussian noise and max clipping into the dropout rates, the mean effective dropout rate across the model is not fixed or easily controlled. In certain architectures, the method's benefit may be a result of the larger effective dropout rate and not the method itself. This in combination with the effective rate's instability, makes comparison with other dropout variants difficult. Moreover, in different architectures such as transformers, which typically employ relatively low drop

rates (often around 0.1 in the literature) compared to CNNs (Y-drop's mean drop rate is ~0.35), Y-drop noise will require a lot more tuning.

Alignment with Optimization Objective Although the original Conductance metric captures each neuron's representational contribution to the final prediction, it does not directly reflect how that contribution influences the model's actual performance relative to the training objective. As a result, neurons that strongly influence the output, but in a way that increases the loss, may still be treated as highly important and assigned aggressive dropout rates. This results in them contributing and being trained less often despite being neurons that need it the most. Reconsidering the definition of neuron importance by reformulating the underlying attribution metric could offer a path toward improving Y-drop's regularization behaviour.

Computational Cost While Y-drop provides stronger regularization than standard dropout, its main drawback lies in the substantial computational overhead introduced by the additional forward–backward pass required to compute Conductance at every training step. Although an additional cost in a method like this is unavoidable since the calculation of the metric is essential, investigating computationally cheaper alternatives that achieve comparable regularization benefits is an important goal of this work.

5.4 Proposed Methods

5.4.1 Alternative Dropout Rate Assignments

To address these limitations and improve generalizability, alternative approaches are proposed, with the goal of designing a method where only one key hyperparameter needs tuning, namely a mean dropout rate p_{mean}, that the average dropout rate across a layer remains equal to. Dropout rates are then derived from conductance scores and normalized around this target. In this way, the method retains sensitivity to neuron-level importance while maintaining control over the overall dropout rate, enabling more stable training and easier and fairer comparisons with standard methods.

Many of the core ideas of Y-drop are maintained, such as computing conductance scores batch-wise and using a per-unit mean, an exponential moving average of dropout rates for the dropout rescaling trick and enforcing an upper limit on the dropout rate, as excessively high values have been observed to disrupt training.

It is important to note that in the code implementation keep rates are computed rather than dropout rates. Consequently, in each of the following methods, the sign of the conductance scores is inverted and normalized around the mean keep rate $1 - p_{\text{mean}}$. Nevertheless, the underlying logic is equivalent to the dropout-based formulation, which is described here to avoid unnecessary confusion for the reader.

To further explore how neuron conductance interacts with dropout rate assignment, two alternative formulations were developed: Ranking Rates and Relative Rates.

These approaches had the goal to be a comparison of how the conductance scores should be interpreted when assigning dropout rates. They were designed as an experimental framework to compare whether dropout should depend on the ordering of neurons or the relative difference in conductance scores. Simultaneously, they helped test the impact outliers of the conductance distribution have, as well as how sharp or smooth the final dropout distribution should be to get optimal results.

Ranking Rates This approach, like the original Y-drop paper, solely ranks the neurons based per unit conductance mean while discarding magnitudes entirely. This ranking is then used to create a dropout rate distribution around a pre-selected mean dropout rate p_{mean} while keeping the rate below a preselected maximum rate $p_{\max}$.

Procedure:

- **Ranking of neurons:**
  For each layer l, neurons are ranked by their per unit mean conductance values $Y^{(l)}$

$$r^{(l)} = \text{rank}(Y^{(l)})$$

  where $r \in [0, N^{(l)}]$ with $N^{(l)}$ being the number of neurons in the layer $N^{(l)}$.

- **Power-based mapping of ranks:** The ranks are transformed into preliminary dropout probabilities using a power-based mapping:

$$p'^{(l)} = p_{\max} \cdot \left(\frac{r^{(l)} - 0.5}{N^{(l)}} \right)^{\text{power}}$$

  In most experiments power is equal to 1 unless clearly mentioned. As a parameter it controls the sharpness of the dropout distribution with larger values emphasizing the difference between the top and bottom ranked neurons, while lower values produce a flatter distribution.

- **Centering around mean dropout rate:**
  The resulting probabilities are centered around the mean dropout rate p_{mean}, with a scaling factor τ, calculated using binary search, ensuring that all values remain within the valid range:

$$p^{(l)} = p_{\text{mean}} + \tau \cdot (p'^{(l)} - \text{mean}(p'^{(l)})), \qquad \tau = \max \text{ s.t. } p^{(l)} \in [0, p_{\max}]$$

Relative Rates This method explores the effect of using relative differences between the mean conductances per unit to assign an accordingly higher or lower dropout rate to a neuron. The goal is to preserve information about how much more important one neuron is compared to another, rather than only their order.

- **Outlier control (optional):** As this method would potentially be more vulnerable to outliers leading to certain neurons receiving extremely low or high dropout values, certain tests were conducted where the top and bottom x% of the Conductance

scores were clipped. This step is not performed in the default configuration and will be explicitly mentioned when used in the following results.

- **Normalization and scaling:** Conductance scores $Y^{(l)}$ are normalized using min–max scaling and rescaled to the interval $[0, p_{\max}]$

$$p'^{(l)} = p_{\max} \cdot \frac{Y^{(l)} - \min(Y^{(l)})}{\max(Y^{(l)}) - \min(Y^{(l)})}$$

- **Centering around mean dropout rate:** Similarly to the Ranking rates, the probabilities are centered around a the mean dropout rate p_{mean}, with a scaling factor τ keeping them in the valid range:

$$p^{(l)} = p_{\text{mean}} + \tau \cdot (p'^{(l)} - \text{mean}(p'^{(l)})), \qquad \tau = \max \text{ s.t. } p^{(l)} \in [0, p_{\max}]$$

These two methods provide alternative ways to leverage conductance scores for dropout rate assignment, each with its own advantages and potential drawbacks. By comparing their performance across various architectures and datasets, insights can be gained into the most effective strategies for conductance-based dropout regularization.

Moreover, both methods can function for every possible layer size and dropout rate without requiring any additional hyperparameter tuning and are compatible with alternative layer attribution methods beyond Conductance.

5.4.2 Loss Conductance

Another idea to improve the performance of Y-drop was redefining the Conductance metric from just attributing each neuron's contribution to the final prediction, to attributing each neuron's contribution to the training objective. To achieve this, we introduce *Loss Conductance* which instead attributes the contribution of each neuron to the final loss.

The new metric's formulation is the following where $\mathcal{L}$ represents the loss function and t the target class:

$$\text{Y}_k^{(l)}(F, x; x') \approx \sum_{j=1}^{n_c} \frac{\partial \mathcal{L}(F(x^{(j)}), t)}{\partial k} \left(F_k^{(l)}(x^{(j)}) - F_k^{(l)}(x^{(j-1)})\right). \tag{5.1}$$

where $x^{(j)} = x' + \frac{j}{n_c}(x - x')$.

In the original conductance formulation, conductance captured the representational influence of each neuron but not necessarily its role on optimizing the training objective. In contrast, Loss Conductance quantifies how variation in a neuron's activation affects the model's capacity to minimize the loss (e.g., a scalar objective $\mathcal{L}$).

This modification shifts Y-drop from a representational importance guided regularization method to an optimization relevance one, with neurons being assigned dropout rates that more directly reflect their contribution to the model's training dynamics.

Specifically positive Loss Conductance values indicate that the neuron contributes to

increasing the loss (i.e., harming performance), negative values indicate that the neuron contributes to decreasing the loss (i.e., helping performance) and values close to zero suggest minimal influence on the loss.

This distinction necessitates inverting the dropout assignment logic in Y-drop: neurons with low Loss Conductance values, those contributing the most to decreasing the final loss (i.e., those performing better), are assigned higher dropout rates, while ones with higher Loss Conductance values are assigned lower rates.

This new altered metric showcased larger improvements in performance and regularization, indicating that its attributions offered more substantive information with regards to the current work's objective.

5.4.3 Computationally Cheaper Variants

Since generalized Y-drop requires a lot more computational resources, primarily due to the double forward–backward passes required for Loss-Conductance calculation in every iteration, we wanted to explore cheaper alternatives. The main hypothesis guiding those efforts is that Generalized Y-drop's impact is lessened in latter stages of training.

There are two possible roots for such behaviour: (1) Generalized Y-drop's impact may be mostly concentrated in the initial stages of training, creating a more balanced neuron landscape that does not need further regularization after a certain point, or (2) the mean Loss-Conductance per-unit value, or at least the relative difference/ranking of neurons in the layer, may stabilize to a high extent after a certain point, making additional recalculations less meaningful. Based on these assumptions two strategies are proposed as a means to test the previous hypotheses as well as establishing beneficial less costly variants.

Early Termination This strategy proposes disabling Generalized Y-drop after a certain portion of training. After the termination point two alternative scenarios are proposed:

- Reverting to standard dropout, something based on the assumption that a sufficiently equalized neuron landscape has already been built.
- Freezing the dropout rates to the last computed values for the rest of training, something based on the assumption that the dropout rates would not change much as relative Loss-Conductance values would not change much.

In practice, both variants, and especially the freezing rates one, were proved to be too aggressive and resulted in substantial decrease in performance gains. This fact motivated the development of more nuanced methods described in the next section.

Infrequent Updates The second strategy, instead of completely disabling Generalized Y-drop, focuses on reducing the frequency of dropout rate updates. Instead of computing Loss Conductance at every iteration and updating rates, this process is done with an interval of x iterations, something that decreases the amount of updates being done proportionally to the chosen interval. Two variants of this strategy are proposed as well:

- **Fixed interval**, where a higher interval is chosen and maintained throughout training.
- **Scheduled interval**, where the interval starts from the value 1, as in the default Generalized Y-drop setup, and becomes higher in the latter stages of training.

The scheduled approach is a more smooth way of testing the hypothesis that relative Loss-Conductance values don't shift that much in latter stages of training, but instead of completely disabling the mechanism we accept that changes do take place but at a slower pace. This implies that some updates are needed but they can become more sparse without much performance loss.

Specifically a number of updates, n_{up}, is selected and the Generalized Y-drop portion of training is split in n_{up} equal fractions. At every fraction the interval is increased by 1, reaching a maximum value of $n_{\text{up}} + 1$.

While the fixed interval variant significantly decreased performance gains, due to what we consider to be its impact on early stages of training, the scheduled variant showcased the most promise out of all examined configurations. It achieved performance gains comparable to the ones of standard Generalized Y-drop while substantially decreasing the amount of additional passes required, something that proves the original hypothesis and proves that cheaper variants are possible and viable.

5.4.4 Application on Transformer Architectures

In Transformer architectures, dropout is typically applied in two main components within each block: (1) to the attention weights in the self-attention mechanism, and (2) to the linear layers in both the attention and feedforward submodules. When attempting to implement Y-drop in this setting, the method can be applied in a straightforward way to the linear layers as their neuron activations are analogous to the linear layers of CNNs. However, applying Y-drop to the attention weights introduces a fundamental difficulty.

Attention weights are not fixed activations but instead context dependent coefficients, representing pairwise relationships between the input's tokens. That implies that they can vastly differ between inputs, and averaging their conductance values across samples would have no semantic value, as the attention between two tokens in one input will have no correlation to that between 2 other tokens in another. Consequently, it is not meaningful to compute conductance across samples in the attention mechanism and therefore Y-drop will not be applied to it in the following tests.

Instead, standard dropout is going to be used in the attention mechanism and the Generalized Y-drop framework will only be applied in the linear layers.

Specifically, in the Vision Transformer architecture, Y-drop is applied to the model by replacing the standard dropout after the linear projection layer of the attention submodule and the two linear layers of the Feed-forward submodule.

Multiple Tokens. One key difference between Transformer architectures and CNNs is that transformers process inputs as a sequence of multiple tokens. Each token is propagated independently through linear layers, while the attention mechanism computes

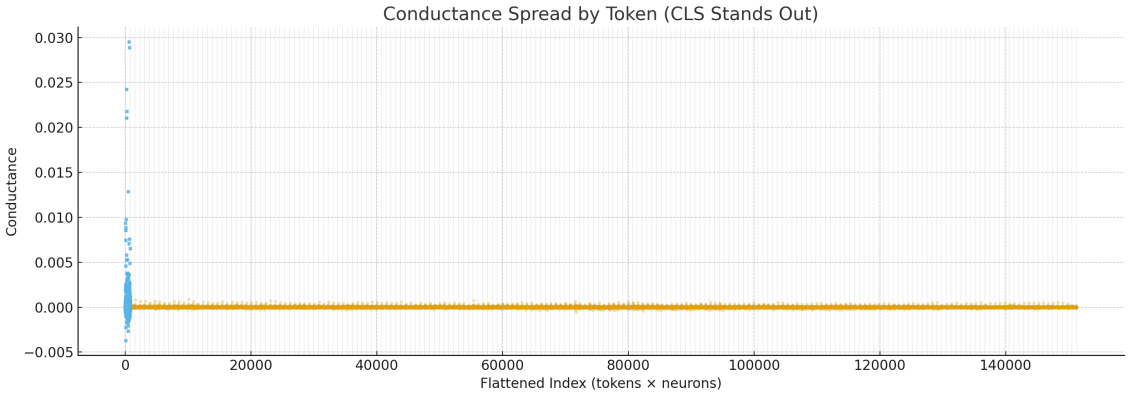


Figure 5.1. *Showcasing per token distribution,with cls token highlighted with blue*

pairwise relationships between them. Depending on the specific architecture and task, not all tokens are used to generate or contribute equally to the final prediction.

For instance, in the Vision Transformer (ViT), an additional learnable classification token (`[CLS]`) is appended to the start of the sequence. This token aggregates information from all the other tokens through the attention mechanism and is used to generate the final prediction.

Initially, when applying Y-drop in this architecture to compute the per-unit mean conductance, conductance was averaged across inputs and across tokens. However, after plotting the conductance values per token and per neuron in multiple layers and batches, it was observed that the `[CLS]` token showcased a much richer and diverse conductance distribution, while the rest of the tokens were relatively muted and had a distribution closer to the uniform one. This could imply that the `[CLS]` token carries most of the relevant gradient flow related to the output. Based on this observation, Y-drop was altered to only use the `[CLS]` token conductance to calculate the per-unit mean conductance, something that led to a measurable improvement in performance.

Residual Connections and Layer Normalization. Another key distinction between Transformer architectures and CNNs used in the original Y-drop paper is the use of residual connections and Layer Normalization (LayerNorm). Many of the layers where Y-drop is applied are immediately followed by a residual connection followed immediately by a LayerNorm operation.

In early experiments, conductance scores were computed after the LayerNorm layer but after testing it was revealed that calculating conductance before it and before the residual connection led to better performance. This improvement is likely due to the residual connection blending the current layer's activation with ones from previous layers. This can in turn obscure the contribution of each individual neuron contribution and worsen Y-drop's rate assignment.

5.4.5 Scaling Generalized Y-Drop/Dropout

In experiments with a lot of data augmentation, it was noticed that both Dropout and Generalized Y-drop had a severe negative impact on performance. This issue became particularly pronounced in larger Transformer variants, where using either of the methods caused the model to fail training altogether, with training collapsing after the warmup period, training loss diverging, and validation accuracy approaching zero. We hypothesized this was caused by the combination of heavy data augmentation and high dropout in early layers, which resulted in too noisy inputs that disrupt representation formation.

Inspired by Drop-Path, we introduce *Scaling Generalized Y-drop* and *Scaling Dropout* as a solution, in which the dropout rate is no longer constant across the model depth. Instead, the rate starts from an initial value of 0 at the initial Transformer block and linearly increases across blocks, reaching a selected maximum rate in the final block. This alternate formulation made both methods viable in heavy-augmentation settings and produced more competitive results.

Chapter 6

Experiments

6.1 Introduction

In this chapter, we present a comprehensive empirical evaluation of the methods and modifications proposed in Chapter 5. We begin by giving a detailed description of the experimental setup by presenting the various architectures, hyperparameter values and training configurations that will be used in the various experiments.

We then analyze and compare the performances produced by each of our proposed configurations. Instead of just presenting numerical results we provide interpretations and explanations on why we think each method succeeds or fails that lead us to our final decisions that led to the highest performance gains. Additionally we analyze settings where our proposed method struggles and the cost behind each modification in regards to the resulting performance gains.

Lastly the chapter concludes with a deeper theoretical analysis that uses the Loss-Conductance metric and training dynamics to explain the impact Generalized Y-drop had on the network both during training as well as the resulting model in evaluation. Through it and some additional experiments provide valuable insights on what aspects of the method actually led to the previously mentioned gains, how behaviour changes in regards to model depth and the information that the Loss-Conductance metric provides on how deep-learning architectures function.

6.2 Experimental Setup

6.2.1 Introduction

This section explains the experimental setup used in this work, by giving a short description of the various architectures, datasets, hyperparameters, and additional training settings that will be used in the latter experiments. It provides the base setting for each combination of model/dataset that will be used, and any change to that baseline will be mentioned clearly.

6.2.2 Architectures

This work was mainly focused on Transformer architectures; preliminary experiments were conducted on CNNs to verify that the proposed regularization methods can be helpful during training, before moving on to the larger Transformer models that require a lot more resources and time to train. In this section we describe the architectures that were used.

CNN. Two different CNN architectures were used, referred to as M1 and M2. Both models share the same convolutional backbone consisting of three convolutional layers with 96, 128, and 256 filters respectively, each having a 5×5 receptive field and a stride of 1 pixel. After every convolutional layer, Batch Normalization and a ReLU activation are applied, followed by 3×3 max pooling with a stride of 2 pixels. The two models differ only in the structure of their feedforward layers. Model M1 has two fully connected layers with 2048 units each, while Model M2 has three fully connected layers with 4096 units each. All linear layers are followed by ReLU activations and a dropout module, which in these experiments is replaced by the generalized Y-drop module.

Transformers. Most experiments were conducted using the Vision Transformer (ViT) architecture. Three common model sizes were used: ViT-Tiny, ViT-Small, and ViT-Base, with differences in embedding dimension and number of heads, as summarized in Table 6.1.

Table 6.1. *ViT model sizes used in experiments.*

Model	ViT-Tiny	ViT-Small	ViT-Base
Embedding Dimension	192	384	768
Number of Heads	3	6	12
Number of Blocks	12	12	12
Parameters	5M	22M	86M
Resolution	224x224	224x224	224x224

Moreover, to assess generalized Y-drop compatibility with heavy data augmentation, some tests are performed on the widely used variant of ViT, the Data-Efficient Image Transformer (DeiT) [109]. DeiT preserves the underlying ViT architecture while including techniques like label smoothing and a variety of data augmentation techniques described in earlier sections such as Repeated Augmentation, RandAugment, Mixup, CutMix, and Random Erasing. The original paper also suggests an additional knowledge distillation mechanism that was not used in this work to isolate the impact of regularization methods. As DeiT retains the base ViT architecture, the same Tiny, Small, and Base variants will be used.

6.2.3 Datasets

Three different widely applied datasets are being used: the CIFAR-10, CIFAR-100 and ImageNet-1k datasets that allow evaluation of the proposed methods with data of different

scale and complexity.

CIFAR datasets. The CIFAR-10 and CIFAR-100 datasets consist of the same 60,000 colour images of size 32×32, with 50,000 of them in the training set and 10,000 in the test set. The difference between them is the amount of possible classes as CIFAR-10 contains 10 distinct classes, making it a relatively simple vision task, while CIFAR-100 divides the images into 100 fine-grained classes making it a more challenging task [110].

ImageNet-1k. ImageNet-1k is a large-scale dataset containing 1.28 million training images and 50,000 validation images, that are commonly used as a test set since the test images were never made public. The images are split into 1000 classes of diverse real-world scenes and have varying resolutions. ImageNet-1k is considered the gold standard benchmark for large vision classification models and is commonly used across bibliography [111].

The CIFAR datasets were used in the CNN experiments as well as in many of the ViT-Tiny tests, where their resolution was increased to 224×224. They provided the basis of many of our findings and ablation analysis due to the faster training times they require. Nonetheless, due to their limited size they quickly overfitted in larger architectures, making their results unreliable indicators of performance.

Instead, using the findings from the CIFAR test, larger ViT variants and DeiT models were trained on the ImageNet-1k dataset that provides the necessary scale and complexity for such architectures.

6.2.4 Training Details

This section summarizes the training settings used across all experimental runs in each architecture/dataset. The hyperparameters remain constant across tests except when clearly mentioned in Generalized Y-drop specific hyperparameters.

Amount of runs. All results showcased are from multiple runs with different seeds showcasing the average maximum accuracies as well as the standard deviation from it. Tests with the CIFAR-10 and CIFAR-100 datasets were done using 5 different seeds while ImageNet tests due to the increased cost were done using 3 seeds.

Generalized Y-drop Parameters. Across experiments Generalized Y-drop was configured with elasticity $a = 0.01$, annealing factor $K = 10\%$ of total epochs. Conductance and Loss Conductance were calculated using 5 linear interpolation steps ($n_{\text{steps}} = 5$) and based on this number the amount of samples being used for the per unit-mean was equal to one fifth of the selected batch size. This way the amount samples, n_{samples}, passing through the model on the conductance calculation step is equal to the one in the training step. Thus experiments using a batch size of 256 used $n_{\text{samples}} = 51$ and ones with batch size of 1024 used $n_{\text{samples}} = 204$ respectively.

For CNN experiments both standard dropout and Generalized Y-drop used a mean dropout rate $p_{\text{mean}} = 0.35$ and the latter a maximum rate of $p_{\text{max}} = 0.7$ while in Transformer experiments as is common in literature dropout rate is $p_{\text{mean}} = 0.1$ and the max rate was selected as $p_{\text{max}} = 0.5$.

CNN Training. As CNN tests mostly took place as sanity checks most hyperparameters were chosen arbitrarily without extensive tuning to achieve the optimal performance. The two CNN models were trained for 150 epochs using a batch size of 256 and a fixed learning rate of 0.001. All results showcased are from 5 runs with different seeds showing the average maximum accuracy of each run as well as the standard deviation from it.

ViT CIFAR Training. Experiments on ViT-Tiny with CIFAR-10 and CIFAR-100 were conducted with a batch size of 256, base learning rate 0.001 with cosine annealing, weight decay 0.05, and 10 warmup epochs. Models trained on CIFAR-10 ran for 400 epochs, while CIFAR-100 models were trained for 300 epochs. Similarly to the CNN tests all results showcased are from 5 runs with different seeds showing the average maximum accuracy of each run as well as the standard deviation from it.

ViT/DeiT ImageNet Training. ImageNet tests both on ViT and DeiT models used the same settings as the CIFAR–ViT configuration, except for an increased batch size of 1024. They were all trained for 300 epochs, matching standard practice for these architectures.

Comparison with Drop-Path. In Transformer tests, in addition to comparing Generalized Y-drop with standard dropout, a comparison with Drop-Path is done. Drop-Path is generally applied with a rate that is linearly increasing across blocks, starting from a rate of 0 in the first block up to max drop rate in the last block. To ensure a fair comparison the max rate was set to 0.2, to yield a model-wise average rate of 0.1 matching the one used in Dropout and Generalized Y-drop.

The only exception to this are the final DeiT tests where a max rate of 0.1 is used like in the original DeiT paper. This change is done since in those tests we compare Scaling Generalized Y-drop and Scaling Dropout to the default Drop-Path that are applied in the same block-wise manner and were made to match the default drop rate for the architecture.

6.3 Dropout Rate Assignment

6.3.1 Introduction

This section will test and compare the two proposed dropout rate assignment methods introduced in Section 5, *Ranking Rates* and *Relative Rates*, and examine their impact on model performance across both CNN and Vision Transformer (ViT) architectures. All experiments in this part use the standard Conductance metric like the original Y-drop to isolate the effect of rate-assignment. The methods will be compared to select the more

robust option across architecture. Specifically on the ViT architectures the methods will be tested with multiple max dropout rate values, $p_{\max}$, to further reduce ambiguity on the optimal method selection as well as discovering the optimal hyperparameter value for further tests.

Simultaneously the comparison is a way of comparing how Conductance values should be interpreted for regularization purposes, whether the magnitudes and relative distances contain meaningful information that should be integrated into the dropout rate selections, or if they are just ranking them is a better approach as gradient based attribution methods like Conductance are prone to noise due to gradient variability and artifacts.

Finally we evaluate whether the two optional steps from the Relative and Ranking Rates formulations, to assess if they can improve performance while also giving us insights into the regularization effect of Generalized Y-drop. In Relative rates, we test the effect *winsorization*, where the top and bottom $x\%$ of values are clipped, allowing us to assess the influence of outliers on dropout rate assignment. For Ranking rates, we examine raising the exponent of the rank normalization above the default value of 1, thereby sharpening the rate distribution, to determine whether stronger emphasis on the highest- or lowest-ranked neurons yields further benefit.

6.3.2 Method Comparison

CNN. We first compare the two dropout-rate assignment methods using the CNN architectures M1 and M2. Across both models, Ranking Rates consistently provide larger performance gains than Relative Rates. The difference is especially clear in the smaller M1 model, where Relative Rates yield almost no improvement, while Ranking Rates showcase a noticeable boost in accuracy.

Table 6.2. *Test accuracy (%) of CNN models on the CIFAR-100 dataset reported as mean ± standard deviation across 5 runs.*

Method	**CIFAR-100**			
	M_1		M_2	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	60.35 ± 0.44	-	59.02 ± 0.38	-
Y-Relative	60.39 ± 0.46	+ 0.04	59.31 ± 0.58	+ 0.29
Y-Ranking	**60.67 ± 0.45**	**+ 0.32**	**59.73 ± 0.22**	**+ 0.71**

Another important observation is that in the larger M2 architecture, that showed clear signs of overfitting as it underperformed the smaller M1, the benefits of the method are a lot more pronounced, especially with Ranking Rates where the accuracy gain is double that of the M1 model. This showcases that the method can be a lot more effective and produce significant gains in models that require more regularization.

ViT. The evaluations on the ViT-Tiny architecture reveal similar trends. The highest performance is again achieved with Ranking Rates when $p_{\max} = 0.5$, mirroring the behaviour observed in CNNs. These results make us conclude that Ranking Rates, which are scale-invariant and less dependent on the exact distribution of Conductance values, are more robust to the noise inherent in gradient-based attribution metrics.

Table 6.3. *Test accuracy (%) of ViT-Tiny model on the CIFAR-100 dataset using the different generalized Y-drop methods across different p_{max} values reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

Method	**ViT-Tiny (CIFAR-100)**			
	Y-Ranking		Y-Relative	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	-	68.08 ± 0.34	-
Drop-Path	65.99 ± 0.05	- 2.09	65.99 ± 0.05	- 2.09
G. Y-drop p_{max} 0.3	68.24 ± 0.53	+ 0.16	68.23 ± 0.42	+ 0.15
G. Y-drop p_{max} 0.5	**68.46 ± 0.30**	**+ 0.38**	68.23 ± 0.23	+ 0.15
G. Y-drop p_{max} 0.7	68.29 ± 0.33	+ 0.21	68.30 ± 0.45	+ 0.22

When it comes to the different $p_{\max}$ values being tested we observe that too small values like $p_{\max} = 0.3$ lead to the worst and most unstable result, as can be seen the big increase in standard deviation. This implies that a broader range of dropout rates helps performance gains, likely due to it enabling stronger penalization of the most important neurons. On the other hand, despite large values like $p_{\max} = 0.7$ performing relatively well, it produces slightly worse and more unstable results than $p_{\max} = 0.5$ implying that overly aggressive dropout rate assignment can be detrimental. These findings together indicate that $p_{\max}$ is an important hyperparameter requiring careful tuning.

6.3.3 Winsorization.

To evaluate whether extreme outliers in the Conductance distribution skew the distribution and result in suboptimal dropout rate assignment that hurts performance, we test clipping the top and bottom 1% of values before rate assignment. Contrary to the hypothesis this change significantly degraded performance, indicating that detecting and punishing those extreme outliers is the key of the main benefit of the Generalized Y-drop method.

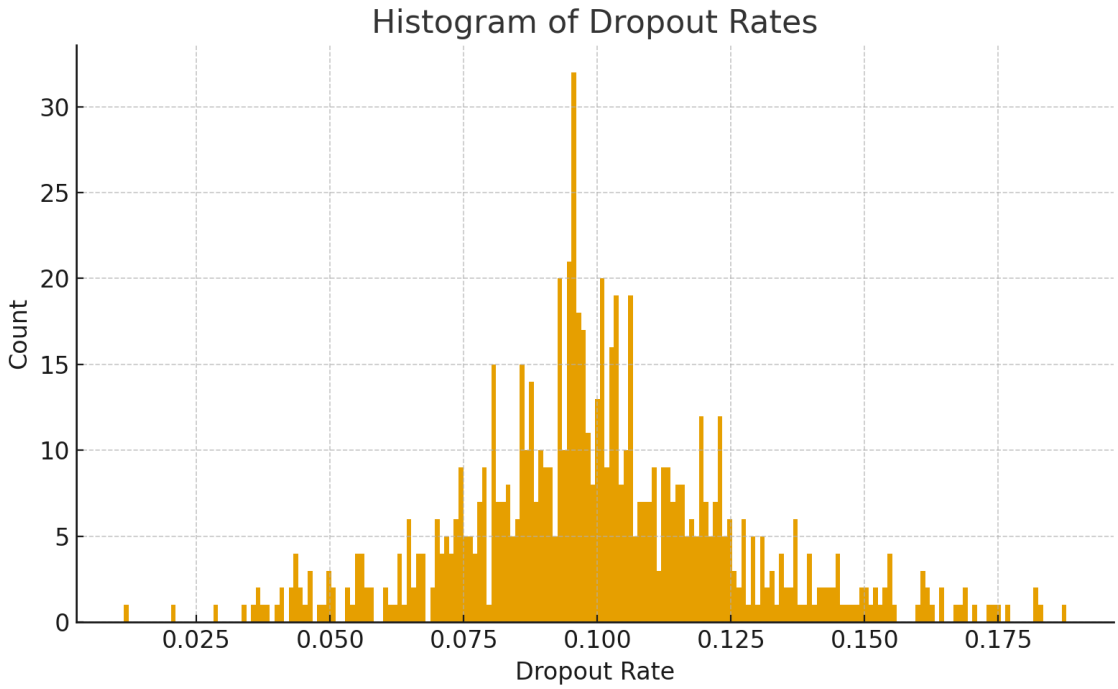


Figure 6.1. *A histogram of the mean dropout rate assigned to neurons of a layer across an epoch using Ranking Rates.*

Table 6.4. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-Relative with different $p_{\max}$ and winsorization settings (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
Y-Relative $p_{\max} = 0.5$	**68.23 ± 0.23**	**+0.15**
+1% winsorization	68.17 ± 0.22	+0.09
Y-Relative $p_{\max} = 0.7$	**68.30 ± 0.45**	**+0.22**
+1% winsorization	68.16 ± 0.37	+0.08

6.3.4 Distribution Sharpness.

For Ranking Rates, we test increasing the power exponent from the standard value of 1 to 2, thereby producing rates more sharply concentrated around extreme values. This change slightly degraded the performance as well and indicates there exists a delicate balance to punishing extreme values, and overly aggressive regularization may be damaging, similarly to higher $p_{\max}$ values.

Table 6.5. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-Ranking with different power settings. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
Y-Ranking power 1	**68.46 ± 0.30**	**+0.38**
Y-Ranking power 2	68.36 ± 0.43	+0.28

6.3.5 Conclusion.

Results in the tests showcased before indicate that the new more general and simplified formulations of dropout rate assignment result in improved regularization in both architectures tested with their effect being more pronounced when higher regularization is necessary. Ranking Rates especially, showcase superior performance across architectures indicating that methods less dependent on the exact conductance distribution are more stable and can avoid possible issues with the noise usually found in gradient based attribution metrics like Conductance.

Lastly the results with different $p_{\max}$ values as well as the Winsorization and Distribution Sharpness tests showcase that the primary advantage of Generalized Y-drop lies in its ability to appropriately penalize neurons with extreme Conductance values, but over amplifying the punishment of such neurons may harm performance thus a balanced approach should be selected. From this point onward all tests will take place using Ranking Rates and when noting something as Y-drop it will be the same as Y-Ranking.

6.4 Loss Conductance

6.4.1 Introduction

The aim of this section is to evaluate our introduced metric, *Loss Conductance*, on how well it performs in the Generalized Y-drop framework. The tests from the previous sections, both with CNNs as well as ViT-Tiny Transformers, will be repeated with the alternate formulation and in the ViT case the method will also be evaluated on the CIFAR-10 dataset.

These tests are done to compare this new metric compared to the standard Conductance and measure whether the more task-oriented formulation of attributing to the loss rather than the final prediction will produce a more appropriate neuron importance ranking that will result in a more targeted dropout rate assignment and more effective regularization.

Lastly we will compare the progress of training and validation loss during the training of the ViT-Tiny model when using Generalized Y-drop with Loss Conductance and standard dropout. This aims to visually showcase the regularization impact the method produces using the introduced metric.

6.4.2 Results

Across all architectures and datasets, Loss Conductance produces a slight but consistent improvement compared to standard Conductance. These results suggest that the loss-oriented attribution signal is more directly aligned with the model's optimization process, and thus provides more informative guidance for dropout rate assignment.

Table 6.6. *Test accuracy (%) of CNN models on the CIFAR-100 dataset using Y-drop with Loss Conductance vs standard Conductance, reported as mean ± standard deviation across 5 runs.*

Method	**CIFAR-100**			
	M_1		M_2	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	60.35 ± 0.44	-	59.02 ± 0.38	-
G. Y-drop	60.67 ± 0.45	+ 0.32	59.73 ± 0.22	+ 0.71
G. Y-drop Loss	**60.80 ± 0.39**	**+ 0.45**	**59.84 ± 0.41**	**+ 0.82**

Table 6.7. *Test accuracy (%) of ViT-Tiny model on the CIFAR-100 and CIFAR-10 datasets using Y-drop Loss Conductance vs standard Conductance reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

Method	**ViT-Tiny**			
	CIFAR-100		CIFAR-10	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	-	91.91 ± 0.32	-
Drop-Path	65.99 ± 0.05	- 2.09	90.8 ± 0.25	- 1.11
G. Y-drop	68.46 ± 0.30	+ 0.38	92.25 ± 0.17	+ 0.34
G. Y-drop Loss	**68.55 ± 0.28**	**+ 0.47**	**92.34 ± 0.22**	**+ 0.43**

6.4.3 Train-Validation Loss Comparison

To showcase how Generalized Y-drop using Loss Conductance affects training we compare training and validation curves it produces compared to the ones of standard dropout.

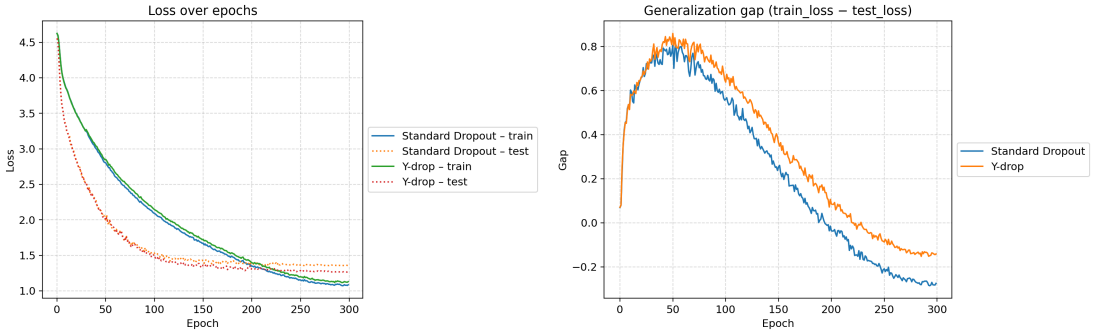


Figure 6.2. *Training and validation loss curves for ViT-Tiny on CIFAR-100 Dropout vs. Generalized Y-drop(left: actual values for both runs, right: difference between training and validation loss in both runs).*

We notice that the model trained with Generalized Y-drop produces a wider gap between training and test loss as training progresses. Training loss remains higher while validation loss keeps on decreasing, achieving a lower low at the end of training.

This is a clear pattern of improved regularization as the model avoids memorizing training data and overfitting on specific characteristics, instead learning more general and robust patterns about the task. This showcases that the penalization of neurons that influence the loss highly does indeed manage to regulate neurons from dominating prediction and learning superficial features that strongly lessen train loss but don't generalize in new data.

6.5 Token Selection and Layer Normalization

6.5.1 Introduction

In the previous ViT experiments, the dropout rate assignment relied exclusively on standard or Loss Conductance values from the [CLS] token, computed after Layer Normalization, in layers that are followed by residual connections. Since both these design choices may significantly shape the attribution signal, this section's experiments are meant to evaluate whether they produce the optimal results and explore how the metrics interact with the architectural elements specific to Transformers.

6.5.2 Token Selection

Plotting the Loss Conductance values of all tokens revealed that the [CLS] token attributions are the most varied while the rest remain more muted and uniform. As a result they were selected to be the sole token to dictate the dropout rate assignment. To make sure this assumption was correct, we repeat the previous tests using the mean of all tokens instead and compare the results to the ones produced with the [CLS] token.

Table 6.8. *Test accuracy (%) of ViT-Tiny model on the CIFAR-100 and CIFAR-10 datasets using different token selection methods for Generalized Y-drop reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

Method	**ViT-Tiny**			
	CIFAR-100		CIFAR-10	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	-	91.91 ± 0.32	-
G. Y-drop Loss Token Mean	68.46 ± 0.30	+ 0.38	92.25 ± 0.17	+ 0.34
G. Y-drop Loss CLS Token	**68.55 ± 0.28**	**+ 0.47**	**92.34 ± 0.22**	**+ 0.43**

The results in both datasets show that incorporating all tokens led to a consistent decrease in performance. This is likely caused by the nature of the [CLS] token, which aggregates information from all tokens through the attention mechanism, while the rest of the tokens contain more localized and muted activations in comparison. As a result, averaging these weaker signals effectively dilutes the more meaningful [CLS] attributions and degrades the quality of the importance ranking used for dropout rate assignment.

These results indicate that in multi-token architectures, the choice of attribution token is critical and tokens that serve as information sinks, like the [CLS] token, may provide the more reliable estimations.

6.5.3 Layer Normalization

In the ViT architecture, two of the three linear layers in each block to which Generalized Y-drop is applied are immediately followed by residual connections and a LayerNorm operation. This raises the question of whether it should be computed before LayerNorm, to isolate the contribution of the layer, or after LayerNorm, to capture the true output that is produced at this section of the architecture.

Table 6.9. *Test accuracy (%) of ViT-Tiny model on the CIFAR-100 and CIFAR-10 datasets using Y-drop with metrics calculated before Layer Normalization vs after reported as (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

Method	**ViT-Tiny**			
	CIFAR-100		CIFAR-10	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	-	91.91 ± 0.32	-
G. Y-drop af. Norm	68.46 ± 0.30	+ 0.38	92.25 ± 0.17	+ 0.34
G. Y-drop bf. Norm	68.36 ± 0.36	+ 0.28	92.27 ± 0.33	+ 0.36
G. Y-drop Loss af. Norm	68.55 ± 0.28	+ 0.47	92.34 ± 0.22	+ 0.43
G. Y-drop Loss bf. Norm	**68.85 ± 0.06**	**+ 0.77**	**92.38 ± 0.19**	**+ 0.47**

The tests are done both with standard and Loss Conductance and results differ across

metric. In standard Conductance, when calculating before layer normalization the performance slightly increases in CIFAR-10 but falls more sharply in CIFAR-100, and in both cases the standard deviation increases. On the other hand, with Loss Conductance, computing attributions before LayerNorm yields improved accuracy on both datasets, particularly on CIFAR-100, and results in lower variability across runs.

This indicates that avoiding residual information when calculating Loss Conductance led to a clearer and more meaningful attribution signal, resulting in an even greater performance gain from the method as well as greater stability in training. Moreover, with this change in both datasets, Generalized Y-drop with Loss Conductance achieved the best performance of every combination so far.

6.5.4 Conclusion

The experiments confirm that [CLS]-only attribution outperforms token-averaged attribution, a sign of the importance of selecting meaningful tokens in multi-token architectures to maximize Generalized Y-drop performance. Additionally, removing residual connections and Layer Normalization impact when computing Loss Conductance, by calculating it before LayerNorm, provides better attribution scores and leads to improved and more stable results.

Based on these findings we have understood more clearly the interaction of the mechanism with many aspects of the ViT architectures and to get the maximum benefit all further tests will be done using Loss Conductance values of the [CLS] tokens computed before LayerNorm.

6.6 Hyperparameter Impact

6.6.1 Introduction

This section evaluates the influence of several key hyperparameters on the effectiveness of Generalized Y-drop. As the values for the two primary hyperparameters, mean dropout rate p_{mean} and the max dropout rate p_{max}, have already been selected in previous sections, we want to evaluate many secondary hyperparameters in order to get insights on where the gains of the method are rooted and attempt to maximize them. The hyperparameters tested here are:

- **Elasticity** a
- **Annealing factor** $K(\%)$
- **Number of linear interpolation steps** n_{steps}
- **Number of samples used in conductance estimation** n_{samples}

Elasticity a is meant to help us understand how the weighted average Generalized Y-drop uses for the dropout rescaling trick impacts results and how sensitive the model is to quicker and slower change.

Annealing factor K is meant to help us understand how early in the training the method should be applied and at what point the Loss Conductance scores start providing meaningful attribution signals.

Lastly n_{steps} and n_{samples} both of which influence the quality and stability of the estimated mean Loss Conductance per unit and can have a significant impact in performance. Either through using more samples to get a more reliable estimate of neuron importance across the data distribution or using more linear steps to capture finer-grained loss sensitivity, we want to see if the additional cost these changes require produce better results.

6.6.2 Elasticity

To test the sensitivity of the method to the rescaling mechanism, we use double and half the original $a = 0.01$ elasticity, to see if faster or slower changes in each update can help stabilize the rescaling trick more. Results show minimal performance variation when either doubling or halving the value, indicating that the method does not heavily rely on the value and an arbitrary small value will produce stable results.

Table 6.10. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different elasticity values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
Elasticity a=0.005	68.77 ± 0.39	+0.69
Elasticity a=0.01	**68.85 ± 0.06**	**+0.77**
Elasticity a=0.02	68.83 ± 0.46	+0.75

6.6.3 Annealing Factor

Several switching points from dropout to Generalized Y-drop were tested to evaluate how early or late the method must be activated to be the most effective.

Table 6.11. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different annealing factor values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
Ann. Factor K=0(%)	68.89 ± 0.30	+0.81
Ann. Factor K=5(%)	**68.92 ± 0.25**	**+0.84**
Ann. Factor K=10(%)	68.85 ± 0.06	+0.77
Ann. Factor K=20(%)	68.45 ± 0.28	+0.37
Ann. Factor K=50(%)	68.37 ± 0.22	+0.29

Results indicate that an early activation is necessary to show significant gains in performance, and switching after 20% of training epochs more than halves the benefit.

Moreover, unlike the original bucket-based Y-drop, the generalized version, which does not cause as large shifts in dropout rates, can produce significant gains even when activated from the beginning, despite Loss Conductance values not being as meaningful at this stage. This is likely a result of the smoother rate assignments leading to less disruptive training dynamics.

Nonetheless, the optimal window of activation still lies around 5% of training and results in the highest accuracy more consistently.

6.6.4 Linear Interpolation Steps

In these tests, it was evaluated whether increasing or decreasing the number of interpolation steps, n_{steps}, in the conductance calculation procedure yields more or less informative metric estimates.

While increasing the number of steps might in principle capture finer-grained loss sensitivity with less noise, doing so greatly increases computational cost, since each additional step multiplies the number of samples requiring a forward–backward pass. This section aims whether that additional cost is justified by performance gains or if a smaller number of steps suffices.

Table 6.12. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different interpolation step amount values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
n_{steps}=2	68.35 ± 0.28	+0.27
n_{steps}=3	68.53 ± 0.14	+0.45
n_{steps}=5	68.85 ± 0.06	+0.77
n_{steps}=10	68.90 ± 0.21	+0.82
n_{steps}=20	**68.92 ± 0.27**	**+0.84**

The results showcase that increasing the performance does improve with more interpolation steps, but the gains diminish rapidly beyond five steps. While moving from two to five steps yields a notable improvement, indicating that there is value in capturing more detailed conductance estimates, further increasing to ten or twenty steps only marginally enhances accuracy, making the additional computational cost unjustifiable. Therefore it is concluded that five interpolation steps strike a good balance between performance and efficiency.

6.6.5 Conductance Calculation Samples.

Finally, the number of samples used to compute per-neuron mean loss conductance is tested. Using more samples should, in theory, provide a more reliable estimate of neuron importance across the data distribution, but it also increases cost proportionally, similarly to the interpolation steps of the previous section.

Table 6.13. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different sample amount values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
$n_{\text{samples}} = 26$	68.64 ± 0.17	+0.56
$n_{\text{samples}} = 51$	68.85 ± 0.06	+0.77
$n_{\text{samples}} = 102$	68.98 ± 0.23	+0.90
$n_{\text{samples}} = 204$	**69.26 ± 0.34**	**+1.18**

The results indicate a strong positive correlation between number of samples and model performance. Increasing the number of samples produces significantly better results, implying that the current sample size (51 samples per update) does not fully capture which neurons consistently drive the loss.

However, the increased computational expense makes such large-sample variants impractical, especially in Transformer architectures. This highlights a limitation of Y-drop: more accurate conductance estimates can improve performance, but the cost of obtaining them grows quickly, something that may be even more apparent in larger datasets where a larger amount of samples is necessary to capture the image of the dataset.

6.6.6 Conclusion

The hyperparameter tests provided useful information on various aspects of the method.

- The method is insensitive to the choice of elasticity, allowing small default values without careful tuning.
- The annealing factor must remain small, with Generalized Y-drop being active for as many epochs as possible except for very few at the start of training.
- Increasing interpolation steps does not meaningfully improve performance, despite the theoretically more detailed estimates, and is not worth the additional cost.
- Increasing the number of samples used to compute Loss Conductance significantly enhances performance, indicating that a more general view of the dataset in each update is key to better results, but it comes with heavy additional computational cost.

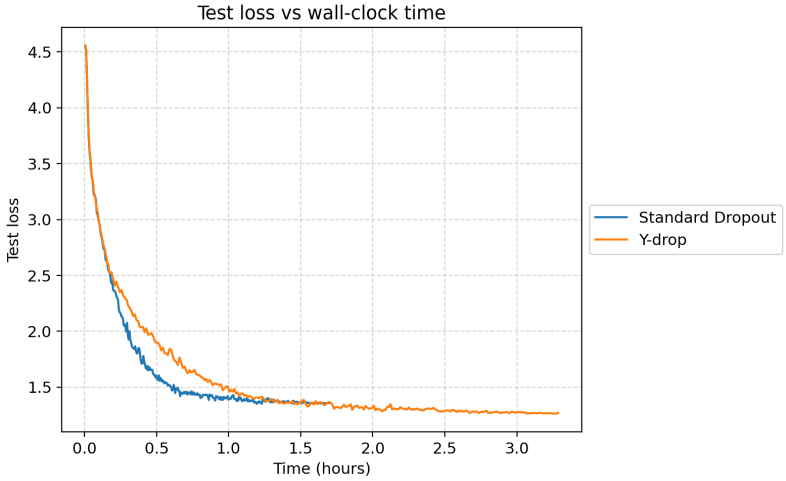


Figure 6.3. *Validation loss for ViT-Tiny on CIFAR-100 vs. wall-clock time for Dropout vs. Generalized Y-drop.*

Together these results indicate that to further enhance Generalized Y-drop one must focus mostly on applying it for more training epochs and with a larger sample of the dataset in each update, two changes that both further increase the cost of the method.

6.7 Computational Cost Reduction Attempts

6.7.1 Introduction

The biggest limitation of the Generalized Y-drop framework is the additional computational cost introduced by the second forward-backward pass required to calculate Loss Conductance at every iteration.

A visual representation of the additional cost can be seen when comparing the validation loss curve in regards to clock time as seen in image 6.3. Despite the fact that the model trained with generalized Y-drop achieves a smaller final validation loss, it requires a lot more time and resources, something that highlights the need for cheaper alternatives.

This section aims to evaluate the proposed computationally cheaper variants of Generalized Y-drop presented in section 5.4.3. Those include early termination and infrequent update strategies that plan to reduce the amount of Loss Conductance calculations and dropout rate updates while achieving analogous performances to the default formulation.

6.7.2 Early Termination

Early termination strategies propose deactivating Generalized Y-drop after a certain percentage of training by either switching back to standard dropout or freezing the rates to the one of the final updates for the rest of training.

Table 6.14. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different early termination strategies. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
G. Y-drop 50(%) of training then freeze	68.26 ± 0.28	+0.16
G. Y-drop 50(%) of training then dropout	68.35 ± 0.19	+0.27
G. Y-drop 75(%) of training then freeze	68.47 ± 0.28	+0.39
G. Y-drop 75(%) of training then dropout	68.62 ± 0.19	+0.54
G. Y-drop 100(%) of training	**68.85 ± 0.06**	**+0.77**

Results showcase that Generalized Y-drop provides substantial regularization across the entire training process. The earlier Y-drop is disabled the lesser are the performance gains. Especially when freezing the rates, the impact is greatly reduced, indicating that the relative importance of neurons keeps evolving throughout training and the frozen rates are outdated and are misaligned to the model's current state. This results in them being more harmful than simply having the equal rates of the standard dropout.

6.7.3 Infrequent updates

Infrequent update strategies propose raising the interval between updates to more iterations than the default one. These are applied both with a fixed larger interval across the whole training or a scheduled approach where the interval increases by one in nup steps.

Table 6.15. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop with different sample amount values. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
G. Y-drop interval 1	**68.85 ± 0.06**	**+0.77**
G. Y-drop interval 2	68.48 ± 0.32	+0.40
G. Y-drop interval 5	68.30 ± 0.37	+0.22
G. Y-drop scheduled n_{up}=1	68.65 ± 0.22	+0.57
G. Y-drop scheduled n_{up}=2	**68.79 ± 0.17**	**+0.71**

Results showcasing that a fixed larger interval leads to a consistent performance decline that grows along with the interval. Especially with interval equal to five the performance gains are almost eliminated.

On the other hand the scheduled approach showcases a lot more promise, as per-

formance gains remain substantial despite the lessening of updates. Especially for nup = 2 the performance is almost the same as the standard method with interval 1 despite requiring approximately 40% fewer conductance calculations.

These results prove the initial hypothesis behind the scheduled approach, that indeed relative neuron importance does continue to evolve in latter stages of training but more slowly, making more infrequent updates viable. Therefore, with appropriate scheduling, generalized Y-drop can indeed produce substantial performance gains without the excessive additional computational costs, something that can open up a path for future efficient implementations.

6.7.4 Conclusion

The previous experiments showcased that Generalized Y-drop continues to influence training throughout all phases, and prematurely disabling it leads to an analogous reduction in performance. Moreover, the relative importance of neurons within each layer continues to evolve even in the later stages of training, indicating that simply freezing dropout rates is ineffective.

However, the strong performance of the scheduled interval variant proves the hypothesis behind its conception, that although neuron importance does change over time, the rate of change decreases as training progresses. This observation opens the door to a range of scheduling strategies in which dropout rates are updated more frequently during the early, rapidly evolving stages of training and less frequently thereafter.

Strategies like these can be a key way of preserving the improved regularization methods while substantially decreasing the computational cost, making the Generalized Y-drop more practical for real world and large scale applications.

6.8 Imagenet, Larger Transformers and Data Augmentation

6.8.1 Introduction

As the previous tests were only conducted with the Tiny variant of ViT we wanted to assess whether the benefits benefits previously reported remain as model size increases, additional experiments were done using the ViT-Small and ViT-Base architectures. Since CIFAR-10 and CIFAR-100 that were previously used are too small for such high capacity models and would lead to quick overfitting regardless of the regularization strategies, the widely used ImageNet-1k dataset is selected.

The change of dataset also provides an additional challenge to the Y-drop model since now, the amount of samples used for each conductance update is smaller fraction of the whole train set compared to the CIFAR tests (6.4 smaller fraction) while also being smaller to the amount of classes in the dataset ($\sim$ 20% the amount of classes, compared to $\sim$ 50 in the CIFAR-100 tests). Since we know that the amount of samples is a vital parameter for final performance, this will test how the method performs with less information.

Lastly in this section some the Generalized Y-drop will be tested on the DeiT architecture to evaluate whether the framework can produce better regularization under heavy

data augmentation.

6.8.2 Larger Vision Transformers

We trained the ViT-Small and ViT-Base on Imagnet-1k using Generalized Y-drop and in both cases it resulted in improved final predictions with slightly less standard deviation. These results showcase that even with a more challenging dataset and larger models Generalized Y-drop can provide improved regularization and combat overfitting than standard dropout.

Table 6.16. *Test accuracy (%) of Larger Vit Architectures on the ImageNet dataset reported as mean ± standard deviation across 3 runs.*

Method	**ImageNet**			
	ViT – Small		*ViT – Base*	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Dropout	76.3 ± 0.26	-	76.14 ± 0.32	-
Drop-Path	74.51 ± 0.16	-1.79	73.93 ± 0.18	-2.11
G. Y-drop	**76.79 ± 0.21**	**+ 0.49**	**76.70 ± 0.20**	**+ 0.56**

The gains could be even more pronounced if the method was initialized earlier or more samples were used, to get a more representative image of the whole dataset in each update, nonetheless even under those more challenging settings the method proved effective.

6.8.3 DeiT Tests

Vision Transformers are *known* to benefit substantially from training with heavy data augmentation, a fact demonstrated prominently in the DeiT architecture [109]. DeiT preserves the underlying ViT architecture while including techniques such as RandAugment, Mixup, CutMix and Random Erasing which have shown to greatly improve accuracy and training stability on ImageNet. Subsequent popular transformer architecture, such as the Swin Transformer [112] have also adopted many of these augmentations and have achieved stronger results. Another key difference in those architectures is that they have replaced the standard dropout with Drop-Path (stochastic depth) in the Transformer blocks. Both DeiT and Swin report that Drop-Path yields superior results to Dropout in regularization in their architecture.

Nevertheless, it is important to evaluate whether the improved, Generalized Y-drop introduced in this work can match or surpass Drop-Path under the same conditions. In these tests the ΔY-drop metric will report the difference between Generalized Y-drop and Drop-Path is the used baseline, as opposed to Dropout in previous sections.

Initial DeiT tests. When first evaluating our method on DeiT models, regularization settings were kept the same as the ViT tests with drop rate being set at 0.1 across all

layers and drop-path being tested with a max value of 0.2.

Table 6.17. *Test accuracy (%) of DeiT-Tiny on the ImageNet dataset using Generalized Y-drop reported as mean ± standard deviation across 3 runs.*

DeiT-Tiny (ImageNet)		
Method	Accuracy	ΔY-drop
Drop-Path	**71.07 ± 0.18**	–
Dropout	68.80 ± 0.34	-2.27
G. Y-drop	69.20 ± 0.09	-1.87

Results from these runs were discouraging. In DeiT-Tiny, despite Generalized Y-drop outperforming standard dropout, they were severely inferior to Drop-Path. The situation was even worse for the DeiT-Base architecture where both dropout and Generalized Y-drop failed to train altogether. Training collapsed after the warmup period,training loss diverged and validation accuracy approached zero.

We hypothesized that this was caused due to the combination of heavy data augmentation and high dropout in early layers, which resulted in too noisy inputs that disrupt representation formation.

Scaling Generalized Y-drop tests. The previous results motivated the development of the Scaling Generalized Y-drop and Scaling Dropout shown in section 5.4.5. In these tests both these methods as well as Drop-Path are chosen to have a max drop rate of 0.1 like in the original DeiT paper.

Table 6.18. *Test accuracy (%) of DeiT Architectures on the ImageNet dataset using scaling Y-drop reported as mean ± standard deviation across 3 runs.*

Method	**ImageNet**			
	DeiT − Tiny		*DeiT − Base*	
	Accuracy	ΔY-drop	Accuracy	ΔY-drop
Drop-Path	72.00 ± 0.34	-	**81.79 ± 0.34**	-
Scaling Dropout	72.02 ± 0.09	0.02	80.92 ± 0.07	-0.87
Scaling G. Y-drop	**72.38 ± 0.26**	**+ 0.38**	80.95 ± 0.03	-0.84

This alteration made Generalized Y-drop more competitive with Drop-Path with it achieving higher accuracy in the DeiT-Tiny variant but Drop-Path remained superior on the DeiT-Base Model.

Another important observation is that even in the DeiT-Tiny model where scaling Y-drop *outperforms*, it no longer achieves a larger train-validation loss gap, with Drop-Path surpassing it. This showcases that under those settings Drop-Path achieves greater regularization.

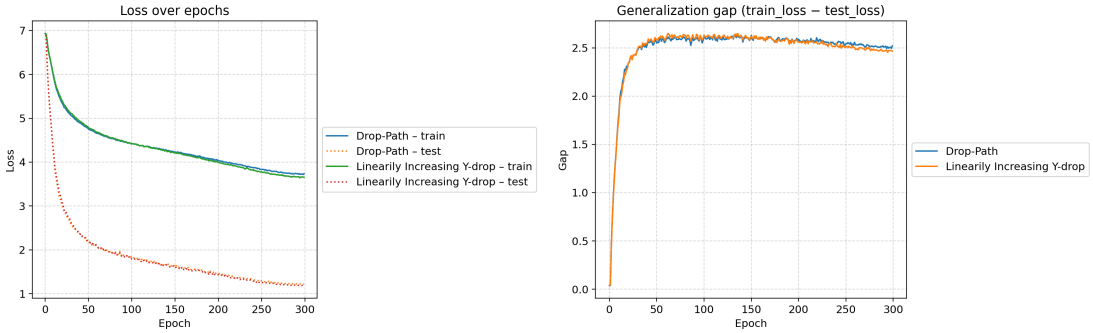


Figure 6.4. *Training and validation loss curves for DeiT-Tiny on CIFAR-100 Dropout vs. Generalized Y-drop(left: actual values for both runs, right: difference between training and validation loss in both runs).*

These results highlight an interaction reported in this paper [50], that when data augmentation is strong, the benefit of regularization methods decreases and can even become negative. As a result the Drop-Path, the milder and more structurally oriented method, outperforms Dropout in contrast to the previous low augmentation ViT tests, where Y-drop severely underperformed.

6.9 Loss Conductance a Deeper Analysis

6.9.1 Introduction

In this section we do a deeper dive into our introduced metric *Loss Conductance*. Specifically we want to examine how Generalized Y-drop impacts the metric's development during training to try to dive deeper into the way the method influences the model compared to using standard dropout. To this end, we employ several metrics and use them to explain and visualize the underlying factors that result in the observed performance improvements.

Moreover we showcase the different patterns Loss Conductance values on the test. We identify characteristic layer-wise patterns and study how these patterns vary due to the application of Generalized Y-drop or generally in better performing models. Based on these observations, we propose several hypotheses regarding the possible causes and meaning behind them, that through targeted experiments and comparisons with established metrics from the literature we come to our final conclusions. Through this process we further understand the relationship between Loss Conductance and Generalized Y-drop with training dynamics, model behaviour and performance as well as possible limitations of the Generalized Y-drop framework.

All results presented in this section are obtained from ViT-Tiny models trained on the CIFAR-100 dataset.

6.9.2 Loss Conductance during training

At selected epochs during training, we freeze training and calculate Loss Conductance across several batches, equal to the size and amount that is used in training, to calculate

the mean Loss-Conductance per unit for each neuron using attributions of the [CLS] token. Using those we calculate the mean and the variance, which we call within-epoch variance, of Loss Conductance for each neuron across all batches.

This setup matches the values that are used to update the model dropout rates during and the metrics derived can help understand how neuron contribution to the final loss evolves over training when using Generalized Y-drop instead of standard Dropout.

While the figures shown correspond to specific layers, they are representative of trends observed throughout the layers network and reflect systematic effects of generalized Y-drop.

Within-Epoch Variance In each layer independently, a mean of the within epoch variance of all its neurons is calculated and plotted. In both cases it can be seen that the value tends to increase as training progresses, showcasing the formation of specialization across neurons. However, the use of Generalized Y-drop seems to limit the growth of the metric, producing lower values in every stage of the training.

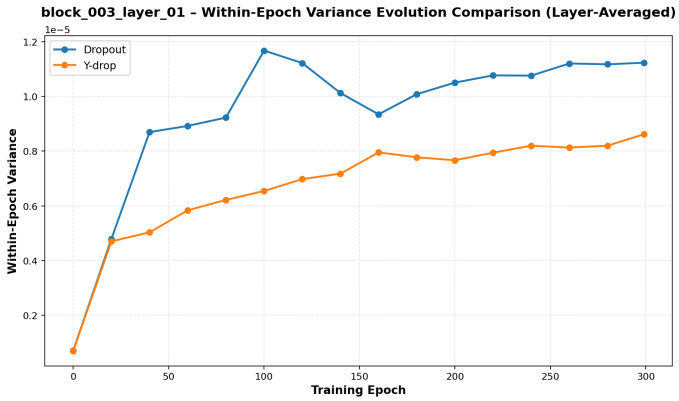


Figure 6.5. *Within-epoch variance across a layer during training. Comparison of Dropout vs. Generalized Y-drop.*

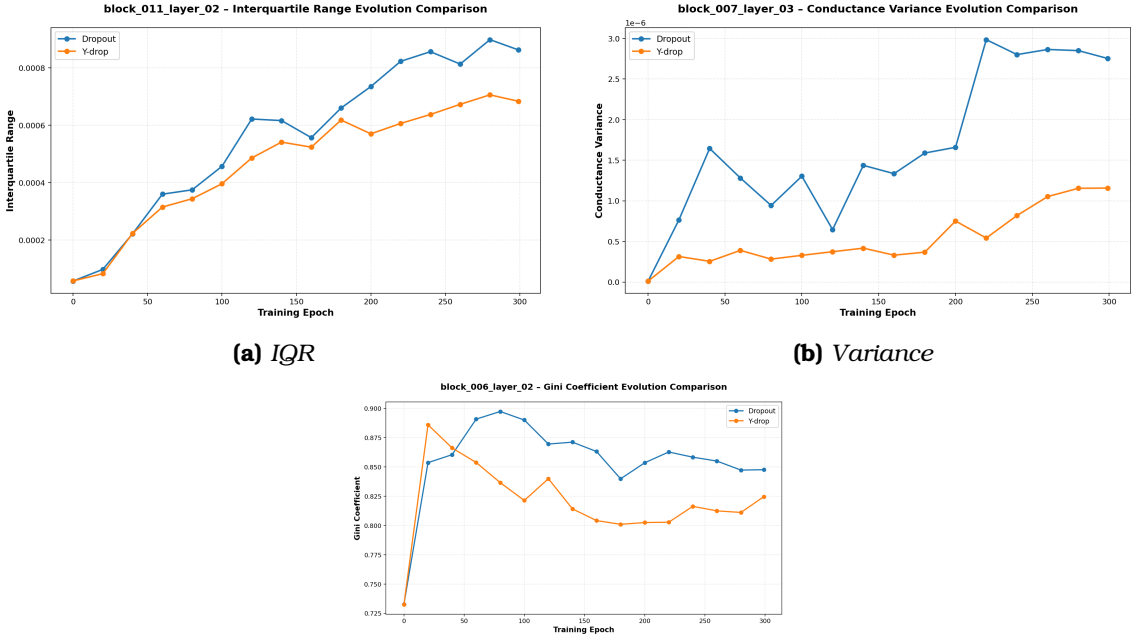
This indicates that neurons trained under generalized Y-drop contribute to the loss in a more stable and uniform manner across different batches. This implies that they learn more core task patterns and are less influenced by small variations and noise that appears in each sample. This stability is a strong indicator of improved regularization, as the model is not as heavily impacted by the aforementioned idiosyncratic features and can perform more appropriately with new data.

Dispersion of neurons To further assess how evenly neurons within a layer contribute to the final loss, the dispersion of mean Loss Conductance values across neurons in a layer is measured using three complementary statistical metrics:

- Variance – measures the overall spread of conductance values. High variance implies that some neurons dominate while others remain inactive.
- Interquartile Range (IQR) – captures the central spread (difference between 75th and 25th percentile), robust to outliers. Lower IQR indicates more homogeneous neuron

importance.

- Gini Index – a widely used inequality measure. A Gini value of 0 denotes perfect equality (all neurons contribute identically), while higher values indicate stronger concentration of importance in a few neurons.



(a) *IQR* **(b)** *Variance*

(c) *Gini*

Figure 6.6. *Interquartile range, variance, and Gini coefficient of mean Loss Conductance across layers during training.Comparison of Dropout vs. Generalized Y-drop.*

Across all three metrics, the generalized Y-drop method consistently produces lower dispersion compared to standard dropout. This result demonstrates a more balanced contribution to the final loss across neurons, without letting a small subset dominate the decision making. This proves that like in the original Y-drop, the generalized version incentivizes more neuron participation.

6.9.3 Loss Conductance Test data

To better understand how generalized Y-drop shapes internal model representations, a post-hoc analysis of loss-based conductance is also performed in the final trained models.

Like before we calculate the mean Loss Conductance per unit for all neurons, using all the samples of the test set. The only exception is that instead of using only the [CLS] token attributions we use the sum of all tokens.

Dispersion of neurons For all linear layers where generalized Y-drop was applied, the interquartile range (IQR), variance, and Gini coefficient of the neuronwise Loss Conductance values.



(a) *IQR - Generalized Y-drop vs Dropout*

(b) *IQR - Difference*

(c) *Gini - Generalized Y-drop vs Dropout*

(d) *Gini - Difference*

(e) *Variance - Generalized Y-drop vs Dropout*

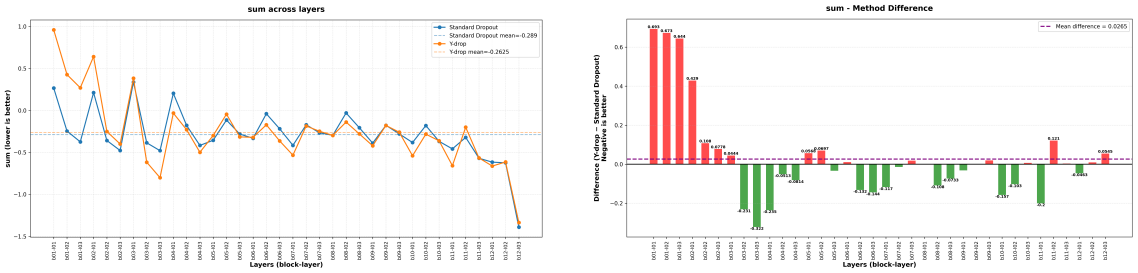
(f) *Variance - Difference*

Figure 6.7. *Dispersion metrics of mean Loss Conductance values for the whole test set in all layers. Comparison of Dropout vs. Generalized Y-drop: Actual values (left column) vs. differences between runs (right column).*

In contrast to the training set statistics, the test set exhibits higher dispersion across neurons: conductance values vary more substantially from neuron to neuron. This suggests that, after training with Y-drop, neurons behave less uniformly and specialize to a greater degree when generalizing to unseen data.

Layerwise Sums of Loss Conductance Initially, we expected the sum of conductance values to be lower in most layers, similarly to how Conductance was higher in the single layer toy example of Y-drop. But a different, more interesting nuanced pattern emerged.

Early layers when using Y-drop have a lot higher positive loss Conductance values while latter layers have slightly smaller negative loss Conductance values compared to the Dropout model.



(a) *LC Sum - Generalized Y-drop vs Dropout* **(b)** *LC Sum - Difference between runs*

Figure 6.8. *Sum of all neurons mean Loss Conductance values for the whole test set in all layers. Comparison of Dropout vs. Generalized Y-drop(left: actual values for both runs, right: difference between runs).*

Initially that was thought of as a problem due to the nature of Loss Conductance and the way it was used in Generalized Y-drop. Firstly some studies have indicated that Integrated Gradients struggle in deeper networks and produce noisier and less informative results [113, 114]. This could explain the different and more undesired behaviour in early layers as the metric is more unstable. Secondly the use of only the CLS token's attribution could be a problem in early layers, as without much cross-token integration that has not yet taken place in early layers, the CLS token's attribution does not provide general meaningful representations.

These ideas were disproven with experiments and evaluations in the next section.

6.9.4 Pattern validation Experiments

Comparison with better Y-drop models Both the dispersion and sum Loss Conductance patterns intensify in the model that was trained using Generalized Y-drop with 4 times the samples, a model that achieved greater final performance and regularization as can be seen in Figure 6.9. This showcases that these patterns are more a genuine property of well-performing Y-drop models and not an artifact of noisy attributions.

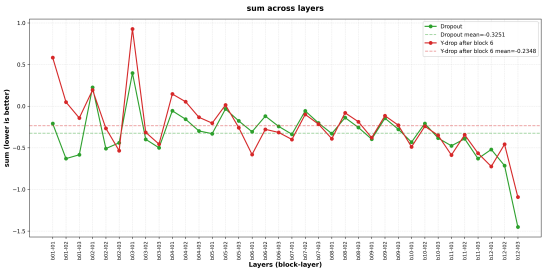
Alternative early transformer block regularizations To further prove the pattern is not a direct symptom of Generalized Y-drop, experiments are conducted where a certain amount of early blocks use standard dropout and Generalized Y-drop is limited to the rest of the deeper blocks. Additionally an experiment using the mean of all token attributions in early layers instead of the CLS token is done to assess that hypothesis.

Table 6.19. *Test accuracy (%) of ViT-Tiny on CIFAR-100 for Y-drop being applied only in deeper layers, while the rest use dropout. (mean ± std over 5 runs). ΔY-drop is the absolute change vs. Dropout.*

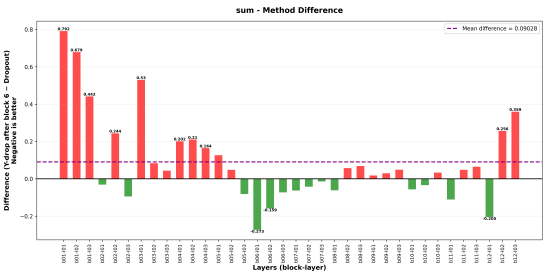
ViT-Tiny (CIFAR-100)		
Method	Accuracy	ΔY-drop
Dropout	68.08 ± 0.34	–
G. Y-drop 3 last blocks	68.61 ± 0.34	+0.53
G. Y-drop 6 last blocks	**68.78 ± 0.24**	**+0.70**
G. Y-drop 10 last blocks	**68.73 ± 0.32**	**+0.65**
G. Y-drop 12(all) blocks	**68.85 ± 0.06**	**+0.77**
G. Y-drop 12(all) blocks,first 2 token mean	68.60 ± 0.47	+0.52

Results showcase that when Generalized Y-drop is applied to only the last 3 blocks, indeed the performance gains reduce, but in all other tests it being applied in only a few deeper blocks showcased similar performance gains as when used in the full model. This demonstrates that the impact of the method is limited to the latter part of the network and early layers do not benefit as much. On the other hand using the mean of token's in early layers also reduced performance gains indicating that the CLS token's attributions were not the issue and even in the early layers, they are more informative that the aggregate of all the tokens.

Additionally when plotting the sum of all Conductance values of the model that was trained with Generalized Y-drop in only the last 6 blocks, the curve pattern across layers compared to the Dropout model remains the same, proving for certain that the intensification of the pattern is a sign of improved regularization and performance and not an unintended side effect of Generalized Y-drop.



(a) *LC Sum - Dropout vs Generalized Y-drop only last 6 blocks*



(b) *LC Sum - Difference between runs*

Figure 6.10. *Sum of mean Loss Conductance per nueuron values per layer for the whole test set in all layers.Comparison of Dropout vs. Generalized Y-drop in only the last 6 blocks (left: actual values for both runs, right: difference between runs).*

6.9.5 Conclusion and Implications

The observed rise and fall pattern of Loss Conductance mirrors a pattern that is described in research about intrinsic dimension. Intrinsic dimension counts the minimum

number of coordinates needed to describe the data and when applied in the hidden dimensions of deep networks it exhibits a “hunchback” profile. [115]

- Early layers: Intrinsic dimension increases, reflecting the network’s expanding of representation and pulling of features (“encoder-like” behavior)
- Deeper layers: Intrinsic dimension shrink sharply, reflecting the compression and refinement of information into task relevant representations to be used for the final prediction (“decoder-like” behavior)

It has been also shown that the better performing the model is the more steep is the curve, with higher Intrinsic Dimension in early layers and smaller values in later layers, something that also appears when comparing the Generalized Y-drop trained model to the Dropout trained one.

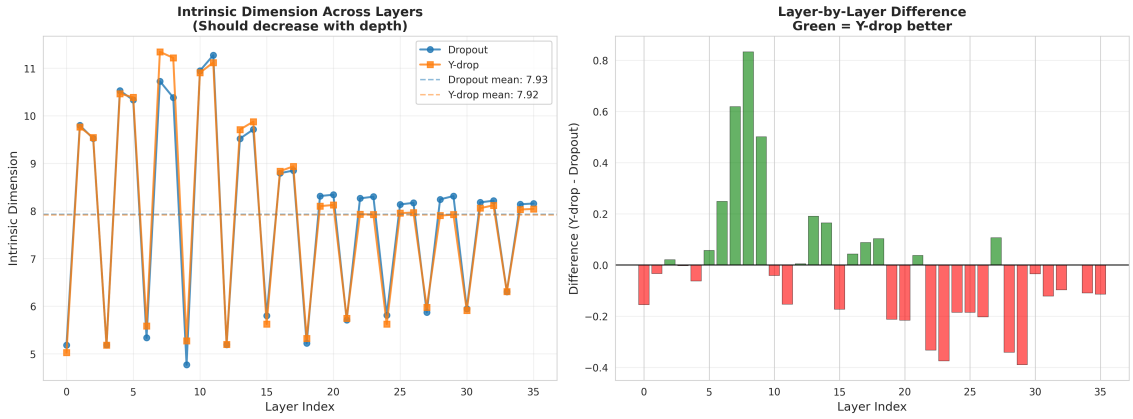
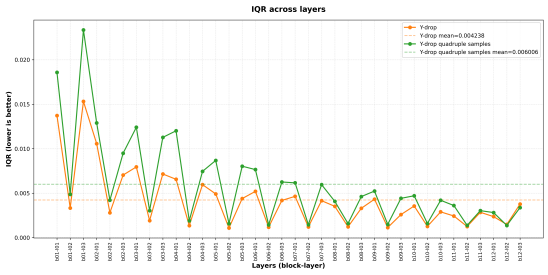


Figure 6.11. *Intrinsic Dimension: Comparison Dropout vs Generalized Y-drop (left: actual values for both runs, right: difference between runs).*

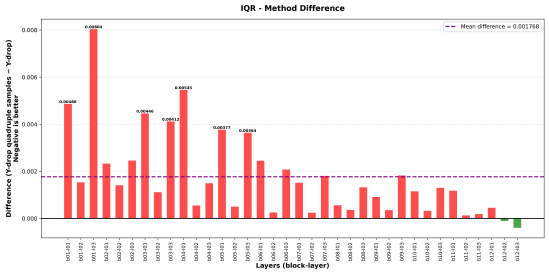
Based on these concepts, the presence of larger positive loss-conductance values in early layers can be interpreted as follows: in the better-performing Generalized Y-drop models, the early layers engage in more diverse and expansive feature extraction. This expansion phase enriches the representational space, but it does not immediately translate to a reduction in loss when viewed through the localized lens of neuron contributions. In fact, as the model explores a broader range of features, many of these early representations are not yet aligned with the downstream task and as a consequence yield positive Loss Conductance values.

In contrast, deeper layer that compress information into task-relevant representations, play a direct loss in the decrease of the loss, producing negative Loss-Conductance values, especially in well-trained models.

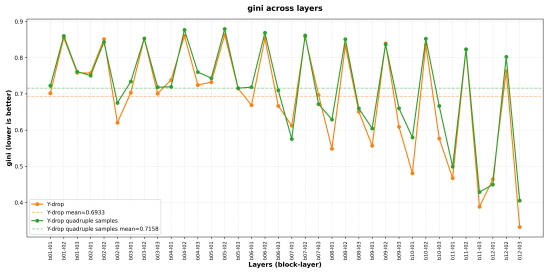
This insight has important implications for the design and application of Generalized Y-drop. The current formulation aligns greatly to deeper layers, something that explains the previous result that showcased most of the method’s impact being concentrated on the later layers. This in turn showcases that alternative approaches need to be formed for early layers to achieve more effective regularization.



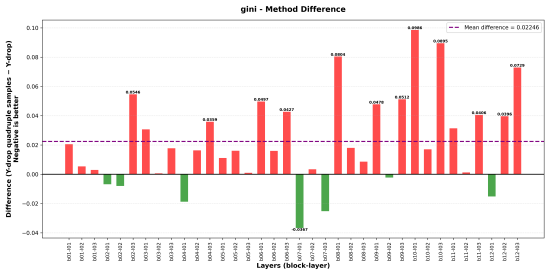
(a) *IQR - Generalized Y-drop vs Generalized Y-drop (4x samples)*



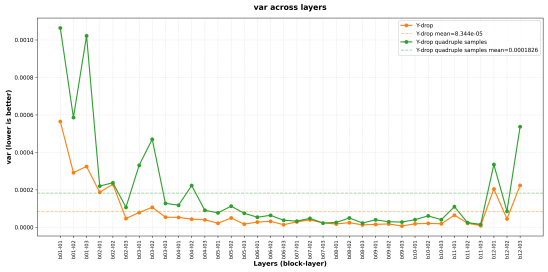
(b) *IQR - Difference*



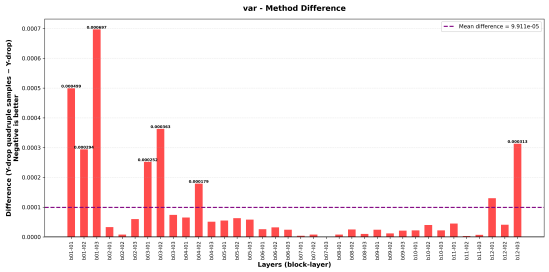
(c) *Gini - Generalized Y-drop vs Generalized Y-drop (4x samples)*



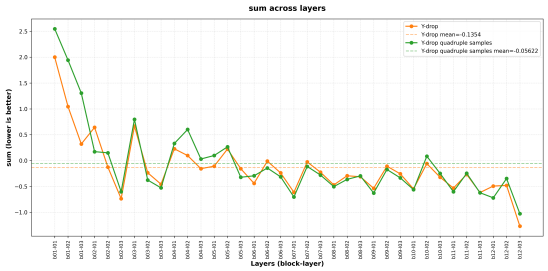
(d) *Gini - Difference*



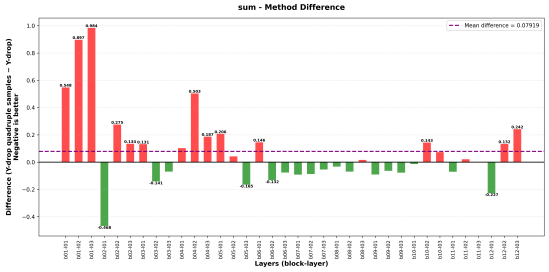
(e) *Variance - Generalized Y-drop vs Generalized Y-drop (4x samples)*



(f) *Variance - Difference*



(g) *LC Sum - GeneralizedY-drop vs Generalized Y-drop (4x samples)*



(h) *LC Sum - Difference*

Figure 6.9. *Dispersion metrics and Sum of mean Loss Conductance values for the whole test set in all layers. Comparison of Generalized Y-drop vs. Generalized Y-drop with 4x samples in Conductance calculation: Actual values (left column) vs. differences between runs (right column).*

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we focused on improving and expanding the Y-drop method, a Conductance based regularization framework, that assigns higher dropout rates to neurons deemed important according to the Conductance attribution metric. While the original method demonstrated improved regularization compared to standard dropout, it also exhibited several limitations related to lack of task alignment, increased computational costs and excessive amount of hyperparameters making results not optimal and practical application challenging.

Through systematic analysis and extensive experimentation, we addressed those issues and proposed Generalized Y-drop, a more practical and improved version of the method.

We introduce a simplified and more dynamic way to assign dropout rates, noted as Ranking Rates, that requires less hyperparameter tuning, can more easily be applied across architectures and appropriately handles noise present in gradient based methods like Conductance.

Most importantly we introduce Loss Conductance, an alternative attribution metric that measures each neuron's contribution to the training loss rather than the final prediction. By aligning neuron importance directly with the optimization objective, Loss Conductance provides a more meaningful signal for regularization, producing consistent performance improvements across architectures and datasets compared to the original Conductance.

Our results suggest that assigning moderately higher dropout rates to neurons that excessively reduce train loss, results in less superficial train set specific patterns and noise being learned. Moreover it encourages a more balanced distribution of loss contributions both across neurons within a layer and across training samples, creating a more uniform representational basis that better transfers to unseen data.

Through extensive ablation studies, we discover that the model thrives when being applied across all stages of training with large update batches that more accurately capture the data distribution larger to calculate conductance without necessarily requiring overly precise attribution estimates.

Since the previously described optimal setting further increase the already substan-

tial computational overhead the method introduces, we explored several cheaper variants, with the most successful being a scheduled dropout-rate update frequency strategy, which exploits the decreasing rate of change in relative Loss Conductance during later training stages, resulting in comparable performance while significantly reducing the computational cost.

Furthermore we adapted Generalized Y-drop to Transformer architectures, identifying design choices related to token selection and layer normalization that are critical for maximizing effectiveness. The method improved regularization even in higher capacity Vision transformers trained on more complex datasets, confirming its scalability.

Nonetheless Generalized Y-drop, like standard dropout, struggles when coupled with heavy data augmentation and even depth-scaled variants like Scaling Generalized Y-drop, that alleviates some instability, are not competitive to alternative regularization strategies.

Finally, we do a deeper examination of Loss Conductance in Transformer models that reveals distinct patterns across network depth that intensify further in better performing variants. Early layers that are associated with feature expansion showcase positive Loss Conductance, as their behaviour does not directly decrease loss and may in isolation increase it, whereas deeper layers show stronger alignment with loss minimization and have negative Loss Conductance values.

This analysis coupled with further experiments showcase that the current formulation of Generalized Y-drop is most effective in later, decoder-like layers, highlighting potential method limitations and motivating future work on layer-specific or depth-aware regularization strategies.

All in all, this thesis demonstrates that Generalized Y-drop, coupled with Loss Conductance, constitutes a meaningful advancement in conductance-based regularization. By improving alignment with optimization objectives, reducing the need for hyperparameter tuning, and offering practical strategies to manage computational cost, this work lays the foundation for more adaptive and interpretable regularization methods in deep neural networks.

7.2 Future Work

Possible future works stemming from this thesis can be pursued along several directions. One of the most important insights of this work is that Loss Conductance exhibits different patterns across network depth, particularly between early, feature-expanding layers and deeper, loss-compressing layers. As a result, having the same rule across layers is suboptimal and depth-aware dropout assignment is required. In early layers, we observed that higher positive test set Loss-Conductance values correlate with higher test set performance. This implies there to be a benefit in suppressing higher Loss Conductance values while training, as in those layers those neurons are the ones that likely cause overfitting instead. On the other hand, as these layers are not directly meant to reduce loss, loss-based attributions may not be that valuable, thus reverting to standard prediction-based Conductance may be preferred. Additionally, methods to automatically select the point where encoder-like layers end and decoder-like ones start is important

to escape manual detection in each architecture. The simplest plan to do this may be to switch the policy depending on the sign of Loss Conductance but more advanced methods can be explored.

Another possible research path is to expand from CNNs and Vision Transformers as Generalized Y-drop is inherently architecture- and modality-agnostic. Future studies should evaluate its effectiveness in other domains, including natural language processing, speech and audio modeling, and reinforcement learning, where an improved dropout method may be more helpful and possible differences in the attribution signals could be explored.

Additionally, the current implementation applies dropout exclusively to linear layers. Extending the framework to other architectural components, like attention mechanisms, might be a promising path. Rather than aggregating attention weight attributions directly, which lacks semantic meaning, attribution signals from the query, key, and value projections could be combined to estimate head-level importance, enabling importance-aware regularization across attention heads and encouraging more balanced head utilization. Although scheduled dropout-rate frequency updates already reduce computational overhead substantially, further work is needed to make attribution-guided regularization more scalable. One promising direction is the development of adaptive update schedules, where the interval between dropout-rate updates is determined automatically based on how much relative neuron importance changes over time. As training progresses and importance rankings stabilize, updates could become increasingly sparse.

Moreover, the Generalized Y-drop is compatible with any neuron-level attribution method. Exploring improved, more recent Integrated Gradient variants or alternative gradient- and propagation-based attribution techniques could either improve regularization effectiveness, reduce computational requirements, or be more compatible in certain architectures. Importantly, many of these methods could be reformulated to perform loss-based attribution, aligning neuron importance directly with the optimization objective and potentially yielding stronger and more stable regularization signals.

Beyond dropout, Loss Conductance, or other possible loss-based attribution methods, could also be incorporated directly into the training loss as penalization terms, analogous to weight decay, discouraging neurons from exhibiting excessively low train loss contributions. This formulation could be particularly valuable in regimes where dropout is less effective or incompatible, such as settings with heavy data augmentation.

Finally, loss-based attribution methods have potential applications beyond training-time regularization. They could be used for model pruning, by identifying neurons that consistently increase test loss or contribute minimally to performance, enabling the construction of smaller and more efficient models without sacrificing accuracy. Similarly, loss-based attribution could guide efficient fine-tuning, where only the most loss-relevant neurons are updated for a new task, reducing computational cost while preserving performance.

Bibliography

[1] TheDataScientistKiran. *Regression vs Classification in Machine Learning: What's the Difference?* https://python.plainenglish.io/regression-vs-classification-in-machine-learning-whats-the-difference-19c57837c203, 2025.

[2] Mukul Rathi. *Feed Forward Neural Network*. https://mukulrathi.com/demystifying-deep-learning/feed-forward-neural-network/.

[3] Kaggleuser community. *CNN diagram (discussion thread image)*. https://www.kaggle.com/discussions/getting-started/171198.

[4] Weijiang Feng. *The standard RNN and unfolded RNN (figure)*. https://www.researchgate.net/figure/The-standard-RNN-and-unfolded-RNN_fig1_318332317.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser και Illia Polosukhin. *Attention Is All You Need*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[6] imgaug Contributors. *imgaug: Example Augmentation Collections*. https://imgaug.readthedocs.io/en/latest/source/overview/collections.html, 2018.

[7] Oleksii Trekhleb. *Cutout, Mixup and CutMix: Implementing Modern Image Augmentations in PyTorch*. https://towardsdatascience.com/cutout-mixup-and-cutmix-implementing-modern-image-augmentations-in-pytorch-a9d7db3074ad/, 2020.

[8] Efthymios Georgiou, Georgios Paraskevopoulos και Alexandros Potamianos. *Y-Drop: A Conductance based Dropout for Fully Connected Layers*. *arXiv preprint arXiv:2409.09088*, 2024.

[9] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016. Comprehensive reference on modern deep learning.

[10] Alex Krizhevsky, Ilya Sutskever και Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Advances in Neural Information Processing Systems (NeurIPS)*, τόμος 25, 2012.

[11] Davide Castelvecchi. *Can we open the black box of AI?* *Nature*, 538(7623):20–23, 2016.

[12] Kedar Dhamdhere, Mukund Sundararajan και Qiqi Yan. *How Important Is a Neuron?* *arXiv preprint arXiv:1805.12233*, 2018.

[13] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[14] Pádraig Cunningham, Matthieu Cord και Sarah Jane Delany. *Supervised Learning*. *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*Matthieu Cord και Pádraig Cunningham, επιμελητές, Cognitive Technologies, σελίδες 21–49. Springer, Berlin, Heidelberg, 2008.

[15] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani και Jonathan Taylor. *Unsupervised Learning*. *An Introduction to Statistical Learning: with Applications in Python*Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani και Jonathan Taylor, επιμελητές, σελίδες 503–556. Springer International Publishing, Cham, 2023.

[16] Yuxi Li. *Deep Reinforcement Learning: An Overview*. *arXiv preprint arXiv:1701.07274*, 2017.

[17] Richard S. Sutton και Andrew G. Barto. *Reinforcement Learning: An Introduction (2nd ed.)*. MIT Press, 2018.

[18] Warren S. McCulloch και Walter Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[19] Frank Rosenblatt. *The Perceptron: A Perceiving and Recognizing Automaton*. Τεχνική Αναφορά με αριθμό, Cornell Aeronautical Laboratory, 1957.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio και Patrick Haffner. *Gradient-Based Learning Applied to Document Recognition*. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Yann LeCun, Léon Bottou, Genevieve B. Orr και Klaus Robert Müller. *Efficient BackProp*. *Neural Networks: Tricks of the Trade*, σελίδες 9–50. Springer, 1998.

[22] Matthew D. Zeiler και Rob Fergus. *Visualizing and Understanding Convolutional Networks*. *Proceedings of the European Conference on Computer Vision (ECCV)*, σελίδες 818–833. Springer, 2014.

[23] Jeffrey L. Elman. *Finding Structure in Time*. *Cognitive Science*, 14(2):179–211, 1990.

[24] Dzmitry Bahdanau, Kyunghyun Cho και Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. *International Conference on Learning Representations (ICLR)*, 2015.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit και Neil Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. *International Conference on Learning Representations (ICLR)*, 2021.

[26] *A Comprehensive Survey of Loss Functions in Machine Learning*. arXiv / survey, 2020. Update authors/year/venue to your preferred survey source.

[27] Herbert Robbins και Sutton Monro. *A Stochastic Approximation Method*. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[28] Finale Doshi-Velez και Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. *arXiv preprint arXiv:1702.08608*, 2017.

[29] Zachary C. Lipton. *The Mythos of Model Interpretability*. *arXiv preprint arXiv:1606.03490*, 2016.

[30] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt και Been Kim. *Sanity Checks for Saliency Maps*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[31] Vitali Petsiuk, Abir Das και Kate Saenko. *RISE: Randomized Input Sampling for Explanation of Black-box Models*. *British Machine Vision Conference (BMVC)*, 2018.

[32] Marco Ancona, Enea Ceolini, Cengiz Öztireli και Markus Gross. *Towards Better Understanding of Gradient-Based Attribution Methods for Deep Neural Networks*. *International Conference on Learning Representations (ICLR), Workshop Track*, 2018.

[33] Karen Simonyan, Andrea Vedaldi και Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. *arXiv preprint arXiv:1312.6034*, 2013.

[34] Avanti Shrikumar, Peyton Greenside και Anshul Kundaje. *Learning Important Features Through Propagating Activation Differences*. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[35] Mukund Sundararajan, Ankur Taly και Qiqi Yan. *Axiomatic Attribution for Deep Networks*. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, τόμος 70 στο *PMLR*, σελίδες 3319–3328, 2017.

[36] Grégoire Montavon, Wojciech Samek και Klaus Robert Müller. *Methods for Interpreting and Understanding Deep Neural Networks*. *Digital Signal Processing*, σελίδες 1–10. Elsevier, 2018.

[37] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus Robert Müller και Wojciech Samek. *On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation*. *PLOS ONE*, 10(7):e0130140, 2015.

[38] Arthur E. Hoerl και Robert W. Kennard. *Ridge Regression: Biased Estimation for Nonorthogonal Problems*. *Technometrics*, 12(1):55–67, 1970.

[39] Robert Tibshirani. *Regression Shrinkage and Selection via the Lasso*. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens και Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 2818–2826, 2016.

[41] Sergey Ioffe και Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, σελίδες 448–456, 2015.

[42] Jimmy Lei Ba, Jamie Ryan Kiros και Geoffrey E. Hinton. *Layer Normalization*. *arXiv preprint arXiv:1607.06450*, 2016.

[43] Alhassan Mumuni και Fuseini Mumuni. *Data augmentation: A comprehensive survey of modern approaches*. *ScienceDirect - Artificial Intelligence Review*, 2022.

[44] Ekin D. Cubuk, Barret Zoph, Jonathon Mane, Vijay Vasudevan και Quoc V. Le. *AutoAugment: Learning Augmentation Strategies from Data*. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 113–123, 2019.

[45] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens και Quoc V. Le. *RandAugment: Practical Automated Data Augmentation with a Reduced Search Space*. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, σελίδες 3008–3017, 2020.

[46] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin και David Lopez-Paz. *mixup: Beyond Empirical Risk Minimization*. *International Conference on Learning Representations (ICLR)*, 2018.

[47] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe και Youngjoon Yoo. *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, σελίδες 6023–6032, 2019.

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever και Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[49] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra και Kilian Q. Weinberger. *Deep Networks with Stochastic Depth*. *European Conference on Computer Vision (ECCV)*, σελίδες 646–661, 2016.

[50] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit και Lucas Beyer. *How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers*. *arXiv preprint arXiv:2106.10270*, 2021.

[51] Curtis G. Northcutt, Lu Jiang και Isaac L. Chuang. *Confident Learning: Estimating Uncertainty in Dataset Labels*. *arXiv preprint arXiv:1911.00068*, 2021.

[52] Pang Wei Koh και Percy Liang. *Understanding Black-box Predictions via Influence Functions*. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, τόμος 70 στο *PMLR*, σελίδες 1885–1894, 2017.

[53] Amirata Ghorbani και James Zou. *Data Shapley: Equitable Valuation of Data for Machine Learning*. *arXiv preprint arXiv:1904.02868*, 2019.

[54] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li και Jan Kautz. *Global Vision Transformer Pruning with Hessian-Aware Saliency*. *arXiv preprint arXiv:2110.04869*, 2021.

[55] Qihuang Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen και Tuo Zhao. *PLATON: Pruning Large Transformer Models with Upper Confidence Bound of Weight Importance*. *arXiv preprint arXiv:2206.12562*, 2022.

[56] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila και Jan Kautz. *Pruning Convolutional Neural Networks for Resource Efficient Inference*. *arXiv preprint arXiv:1611.06440*, 2016.

[57] Paul Michel, Omer Levy και Graham Neubig. *Are Sixteen Heads Really Better than One?* *arXiv preprint arXiv:1905.10650*, 2019.

[58] Jimmy Ba και Brendan Frey. *Standout: Variable Dropout for Deep Networks*. *arXiv preprint arXiv:1312.6116*, 2013.

[59] John McCarthy, Marvin Minsky, Nathaniel Rochester και Claude E. Shannon. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. 1956.

[60] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1η έκδοση, 1997. Classic introductory text defining the *E-T-P* formulation.

[61] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, 2006.

[62] Rakesh Agrawal, Tomasz Imieliński και Arun Swami. *Mining Association Rules Between Sets of Items in Large Databases*. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, σελίδες 207–216, 1993.

[63] Christopher M. Bishop και Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer, Cham, 2024.

[64] David E. Rumelhart, Geoffrey E. Hinton και Ronald J. Williams. *Learning representations by back-propagating errors*. *Nature*, 323:533–536, 1986.

[65] Vinod Nair και Geoffrey E. Hinton. *Rectified Linear Units Improve Restricted Boltzmann Machines*. *ICML*, 2010.

[66] Günter Klambauer, Thomas Unterthiner, Andreas Mayr και Sepp Hochreiter. *Self-Normalizing Neural Networks*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[67] Dan Hendrycks και Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. *arXiv preprint arXiv:1606.08415*, 2016.

[68] Kurt Hornik. *Approximation capabilities of multilayer feedforward networks*. *Neural Networks*, 4(2):251–257, 1991.

[69] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A Gentle Introduction and Overview*. *arXiv preprint arXiv:1912.05911*, 2019.

[70] Yoshua Bengio, Patrice Simard και Paolo Frasconi. *Learning Long-Term Dependencies with Gradient Descent is Difficult*. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

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

[72] Kyunghyun Cho, Bartvan Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk και Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1724–1734, 2014.

[73] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen και Yunfeng Liu. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021.

[74] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. *NAACL-HLT*, 2019.

[75] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei και Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. OpenAI Technical Report, 2019.

[76] Boris T. Polyak. *Some Methods of Speeding up the Convergence of Iteration Methods*. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[77] Diederik P. Kingma και Jimmy Ba. *Adam: A Method for Stochastic Optimization*. *International Conference on Learning Representations (ICLR)*, 2015.

[78] Ilya Loshchilov και Frank Hutter. *Decoupled Weight Decay Regularization*. *International Conference on Learning Representations (ICLR)*, 2019.

[79] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas και Martin Wattenberg. *SmoothGrad: Removing Noise by Adding Noise*. *arXiv preprint arXiv:1706.03825*, 2017.

[80] Ruth C. Fong και Andrea Vedaldi. *Interpretable Explanations of Black Boxes by Meaningful Perturbation*. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, σελίδες 3429–3437, 2017.

[81] Pascal Sturmfels, Scott Lundberg και Su In Lee. *Visualizing the Impact of Feature Attribution Baselines*. *Distill*, 2020.

[82] Gabriel G. Erion, Joseph D. Janizek, Pascal Sturmfels, Scott M. Lundberg και Su In Lee. *Improving performance of deep learning models with axiomatic attribution priors and expected gradients*. *Nature Machine Intelligence*, 3(7):620–631, 2021.

[83] Sina Salek και Joseph Enguehard. *Using the Path of Least Resistance to Explain Deep Networks*. *arXiv preprint arXiv:2502.12108*, 2025.

[84] Yue Zhuo και Zhiqiang Ge. *IG2: Integrated Gradient on Iterative Gradient Path for Feature Attribution*. *arXiv preprint arXiv:2406.10852*, 2024.

[85] Aviad Kapishnikov, Gil Veksler, Gal Elidan, Naftali Tishby και Benjamin I.P. Rubinstein. *Guided Integrated Gradients: An Adaptive Path Method for Removing Noise*. *arXiv preprint arXiv:2106.09788*, 2021.

[86] Joseph Enguehard. *Sequential Integrated Gradients: a simple but effective method for explaining language models*. *arXiv preprint arXiv:2305.15853*, 2023.

[87] Avanti Shrikumar, Jocelin Su και Anshul Kundaje. *Computationally Efficient Measures of Internal Neuron Importance*. *arXiv preprint arXiv:1807.09946*, 2018.

[88] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek και Klaus Robert Müller. *Layer-Wise Relevance Propagation: An Overview*. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, τόμος 11700 στο *LNCS*, σελίδες 193–209. Springer, 2019.

[89] Hila Chefer, Shir Gur και Lior Wolf. *Transformer Interpretability Beyond Attention Visualization*. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 782–791, 2021.

[90] Hila Chefer, Shir Gur και Lior Wolf. *Generic Attention-Model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers*. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, σελίδες 397–406, 2021.

[91] Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin και Wojciech Samek. *AttnLRP: Attention-Aware Layer-Wise Relevance Propagation for Transformers*. *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2021.

[92] Stuart Geman, Elie Bienenstock και René Doursat. *Neural Networks and the Bias/-Variance Dilemma*. *Neural Computation*, 4(1):1–58, 1992.

[93] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford και Oleg Klimov. *Proximal Policy Optimization Algorithms*. *arXiv preprint arXiv:1707.06347*, 2017.

[94] Elad Hoffer, Itay Hubara και Daniel Soudry. *Augment Your Batch: Improving Generalization Through Instance Repetition*. *International Conference on Learning Representations (ICLR)*, 2020.

[95] Terrance DeVries και Graham W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*. *arXiv preprint arXiv:1708.04552*, 2017.

[96] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li και Yi Yang. *Random Erasing Data Augmentation*. *Proceedings of the AAAI Conference on Artificial Intelligence*, τόμος 34, σελίδες 13001–13008, 2020.

[97] Yarin Gal και Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. *International Conference on Machine Learning (ICML)*, 2016.

[98] Golnaz Ghiasi, Tsung Yi Lin και Quoc V. Le. *DropBlock: A Regularization Method for Convolutional Networks*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[99] Yarin Gal και Zoubin Ghahramani. *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[100] Qilong Wang, Mingze Gao, Zhaolin Zhang, Jiangtao Xie, Peihua Li και Qinghua Hu. *DropCov: A Simple yet Effective Method for Improving Deep Architectures*. *Advances in Neural Information Processing Systems*, 2022.

[101] Jimmy Ba και Brendan Frey. *Adaptive Dropout / Annealed Dropout Training of Deep Networks*. *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2013.

[102] Pietro Morerio, Jacopo Cavazza, Riccardo Volpi, René Vidal και Vittorio Murino. *Curriculum Dropout*. *arXiv preprint arXiv:1703.06229*, 2017.

[103] Yarin Gal, Jiri Hron και Alex Kendall. *Concrete Dropout*. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[104] Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang και Tie Yan Liu. *R-Drop: Regularized Dropout for Neural Networks*. *arXiv preprint arXiv:2106.14448*, 2021.

[105] Shiwen Ni, Min Yang, Ruifeng Xu, Chengming Li και Xiping Hu. *Layer-wise regularized dropout for neural language models*, 2024.

[106] Shengjie Wang, Tianyi Zhou και Jeff Bilmes. *Jumpout : Improved Dropout for Deep Neural Networks with ReLUs*, 2019.

[107] Andrew Ross, Michael C. Hughes και Finale Doshi-Velez. *Right for the Right Reasons: Training Differentiable Models by Constraining Their Explanations*. *arXiv preprint arXiv:1703.03717*, 2017.

[108] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim και Percy Liang. *Concept Bottleneck Models*. *arXiv preprint arXiv:2007.04612*, 2020.

[109] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles και Hervé Jégou. *Training data-efficient image transformers & distillation through attention*. *arXiv preprint arXiv:2012.12877*, 2020.

[110] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Τεχνική Αναφορά με αριθμό, University of Toronto, 2009. CIFAR-10/100 dataset report.

[111] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li και Li Fei-Fei. *ImageNet: A Large-Scale Hierarchical Image Database*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 248–255, 2009.

[112] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin και Baining Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. *arXiv preprint arXiv:2103.14030*, 2021.

[113] Gary S. W. Goh, Sebastian Lapuschkin, Leander Weber, Wojciech Samek και Alexander Binder. *Understanding Integrated Gradients with SmoothTaylor for Deep Neural Network Attribution*. *arXiv preprint arXiv:2004.10484*, 2020.

[114] Yongjie Wang, Tong Zhang, Xu Guo και Zhiqi Shen. *Gradient based Feature Attribution in Explainable AI: A Technical Review*. *arXiv preprint arXiv:2403.10415*, 2024.

[115] Alessio Ansuini, Alessandro Laio, Jakob H. Macke και Davide Zoccolan. *Intrinsic dimension of data representations in deep neural networks*. *arXiv preprint arXiv:1905.12784*, 2019.

List of Abbreviations

AI	Artificial Intelligence
	Τεχνητή Νοημοσύνη
ML	Machine Learning
	Μηχανική Μάθηση
DL	Deep Learning
	Βαθιά Μάθηση
ANN	Artificial Neural Network
	Τεχνητό Νευρωνικό Δίκτυο
RNN	Recurrent Neural Network
	Αναδρομικό Νευρωνικό Δίκτυο
CNN	Convolutional Neural Network
	Συνελικτικό Νευρωνικό Δίκτυο
ViT	Vision Transformer
	Μετασχηματιστής Όρασης
DeiT	Data-efficient Image Transformer
	Αποδοτικός σε Δεδομένα Μετασχηματιστής Εικόνων
IQR	Interquartile Range
	Διατεταρτημοριακό Εύρος