ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βελτίωση ταξινομητών βαθιάς μάθησης με αυτόματη επαύξηση δεδομένων

Της φοιτήτριας
Οσμαντζικίδου Έλλη Ειρήνη
Αρ. Μητρώου: 154514

Επιβλέπων
Κωνσταντίνος Διαμαντάρας
Βαθμίδα Καθηγητής

Ημερομηνία Σεπτέμβριος 2023

Βελτίωση ταξινομητών βαθιάς μάθησης με αυτόματη επαύξηση δεδομένων

20233

Οσμαντζικίδου Έλλη Ειρήνη

Διαμαντάρας Κωνσταντίνος

15 Φεβρουαρίου 2022

10 Σεπτεμβρίου 2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αυτη η πτυχιακή είναι αφιερωμένη στην οικογένεια μου»

Πρόλογος

Τα τελευταία χρόνια η μηχανική μάθηση γνωρίζει ραγδαία ανάπτυξη με εφαρμογές που συναντούν οι άνθρωποι και στην καθημερινή τους ζωή. Ένας από τους πιο διαδεδομένους τομείς της μηχανικής μάθησης είναι ο τομέας της βαθιάς μάθησης, που τα μοντέλα του έχουν μεγάλη επιτυχία σε πάρα πολλούς τομείς, από την ταξινόμηση spam mail μέχρι και αναγνώριση πολύπλοκων εγκεφαλικών σημάτων. Ένα από τα κύρια προβλήματα που αντιμετωπίζει όμως είναι ότι πολλές φορές χρειάζονται πολύπλοκα ή ογκώδη dataset έτσι ώστε να βρεθούν τα σωστά αποτελέσματα.

Για την καταπολέμησή αυτού του προβλήματος, έχει αναπτυχθεί η αυτοματοποιημένη επαύξηση δεδομένων. Με αυτήν μπορούμε ακόμα και τα πιο μικρά dataset, να τα επαυξήσουμε, μεγαλώνοντας και το μέγεθος του dataset άλλα και την πολυπλοκότητα των εικόνων, με απώτερο σκοπό την επίτευξη καλύτερης γενίκευσης του μοντέλου. Αυτοματοποιημένη, γιατί πολλές φορές η εύρεση του σωστού συνδυασμού των αλλαγών που πρέπει να κάνουμε, είναι δύσκολο να βρεθούν ή είναι αρκετά πολύπλοκες για να βρεθούν με το χέρι.

Περίληψη

Στον τομέα της βαθιάς μάθησης, η επιτυχία ενός μοντέλου ταξινόμησης συχνά κρίνεται από την ποσότητα και την ποιότητα των δεδομένων στο οποίο εκπαιδεύεται. Πολλές φορές όμως λόγω των προβλημάτων που καλούμαστε να αντιμετωπίσουμε, δεν θα έχουμε επαρκή δείγματα για την ορθή εκπαίδευση του μοντέλου. Για να πετύχουμε ακόμα μεγαλύτερη ακρίβεια στις προβλέψεις του δημιουργήθηκε ένας τρόπος αυτόματης επαύξησης δεδομένων, έτσι ώστε να γίνεται γρήγορα και αποτελεσματικά η βελτίωση των ταξινομητών.

Παρόλο που οι τεχνικές αυτόματης επαύξησης είναι πάρα πολλές, αυτό το paper στηρίχθηκε σε μία καινοτόμα state of the art προσέγγιση, την Διαφορική Αυτόματη Επαύξηση Δεδομένων (Differential Automatic Data Augmentation). Η βασική καινοτομία του είναι ότι μετατρέπει τα augmentations σε κομμάτι του ταξινομητή και τα εκπαιδεύει μαζί με αυτόν, παίρνοντας τα αποτελέσματα του ταξινομητή σαν ανατροφοδότηση για τις αλλαγές που κάναμε. Στόχος του είναι να βρει την βέλτιστη πολιτική επαύξησης δεδομένων.

Σαν ταξινομητή επέλεξα να χρησιμοποιήσω ένα ResNet50 με μάθηση με μεταφορά έτσι ώστε να μη χρειαστεί να εκπαιδεύσω όλο το μοντέλο άλλα μόνο τα τελευταία στρώματα, χρησιμοποιώντας τον με αυτόν τον τρόπο σαν εξαγωγέα χαρακτηριστικών. Σαν δεδομένα επέλεξα ένα dataset δυαδικής ταξινόμησης με εικόνες από γάτες και σκύλους. Ήθελα κάτι που η δυσκολία του δεν θα προκύπτει μέσα από πλήθος κλάσεων αλλά θα προκύπτει από το πόσο δύσκολο ήταν να ταξινομηθούν οι εικόνες σωστά, είτε λόγω της θέσης του αντικειμένου, είτε λόγω του χρώματος και να βελτιώσω αυτές τις περιπτώσεις.

Μέσα από πέντε πειράματα καταφέρνω να παρουσιάσω την αποτελεσματικότητα των τεχνικών αυτόματης επαύξησης σε σχέση με την μη χρήση της ή την χρήση της με χειροκίνητο τρόπο.

Improving Deep Learning Classifiers with Automatic Augmentation

Osmantzikidou Elli Eirini

Abstract

In the field of machine learning, the success of a deep learning classification model is directly correlated with the dataset that it is being trained on. Due to the nature of the problems we are tasked with solving, many times we won't have sufficient data for the correct training and optimization of the problem. In order to achieve even better predictions with greater generalization results, various automatic data augmentation methods were created, so the classifiers could improve quickly and efficiently.

Although there are many automatic augmentation techniques, in this paper I relied on an innovative state-of-the-art approach, Differential Automatic Data Augmentation. The core idea behind it is to turn the augmentations into a part of the classifier and to train them along with it, using the results of the classifier as feedback on the model regarding the changes we made. Its goal is to find the optimal data augmentation policy for the given dataset.

As a classifier, I chose to use a ResNet50 with transfer learning so that I don't have to train the whole model from scratch but only the last layers, thus using the core ResNet50 as a feature extractor. Regarding the dataset, I chose a binary classification dataset with images of cats and dogs. I wanted the complexity of the problem to not be in a variety of classes, but in the difficulty of the classifications of the images and the correct augmentations that had to be done to achieve a better accuracy.

Through five experiments, I manage to present the effectiveness of automatic augmentation techniques in relation to not using any augmentation at all or using it with manual values.

Ευχαριστίες

Σε αυτό το σημείο, θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς την οικογένειά μου, η οποία με στήριξε και με ενέπνευσε καθ' όλη την διάρκεια των σπουδών μου, αποτελώντας ανέκαθεν μια πηγή δύναμης για μένα.

Επίσης, δεν μπορώ παρά να ευχαριστήσω τους εκπαιδευτικούς μου, οι οποίοι με κατεύθυναν και με συμβούλευαν καθόλη την διάρκεια τον σπουδών μου. Ειδικότερα θα ήθελα να ευχαριστήσω τον κύριο Διαμαντάρα, ο οποίος με την μεγάλη του υπομονή με υποστήριξε και συνέβαλε στην επιτυχή ολοκλήρωση αυτής της πτυχιακής εργασίας.

Τέλος, δεν μπορώ να παραλείψω να ευχαριστήσω τους φίλους μου, οι οποίοι μου στάθηκαν καθ' όλη την διάρκεια των σπουδών μου, προσφέροντας ανεκτίμητη υποστήριξη.

Περιεχόμενα

Εισαγωγή

Σε ότι αφορά δεδομένα εικόνων, η επαυξηση τους, είτε σε επιπεδο μετατοπίσεων, μετατροπών ή σε επίπεδο αλλαγής χρώματος, πραγματοποιείται πάντα με κύριο γνώμονα το πρόβλημα που καλούμε να λύσουμε και χρησιμοποιούνται σε μεγάλο βαθμό σε προβλήματα μηχανικής μάθησης. Ο λόγος χρησης τους είναι γιατί βοηθάει τα μοντέλα να μην κάνουν overfit, παρέχοντας στα μοντέλα ποικίλες εικόνες έτσι ώστε να μάθουν σε ένα μεγάλο εύρος και συνεπώς να γενικεύουν καλύτερα. Αυτό είναι ιδιαίτερα σημαντικό όταν έχουμε ένα μικρότερο dataset και θέλουμε να το μεγαλώσουμε, αλλα δεν διαθέτουμε τρόπο να συλλέξουμε περισσότερα δείγματα Η όταν θέλουμε να καταπολεμήσουμε την άνιση κατανομή δειγμάτων στο dataset, με αυτόν τον τρόπο μπορούμε να δημιουργήσουμε καινούργια απο την κλάση με τα λιγότερα δείγματα.

Όμως η επιλογή των σωστών μετατροπών και παραμέτρων για την αποτελεσματική επαύξηση των δεδομένων μπορεί να αποβεί αρκετα χρονοβόρα διαδικασία. Έτσι το 2018, οι Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le δημοσίευσαν το μοντέλο τους το AutoAugment. [86] Η καινοτομια του είναι ότι μετέτρεψαν την διαδικασία εύρεσης και επιλογής των μετατροπών που πρέπει να γίνουν σε μία ακολουθία που πρέπει να βελτιστοποιηθεί μετατρεποντας το ετσι σε πρόβλημα βελτιστοποιησης. Δυστυχώς όμως παρόλο που η ιδέα ήταν πάρα πολύ καλή, λόγω του ότι χρησιμοποιούσαν μάθηση με ενίσχυση για την εύρεση των παραμέτρων των μετατροπων, η εκπαίδευση ενός τέτοιου μοντέλου είναι ιδιαίτερα υπολογιστικά ακριβής και χρονοβόρα κάτι που την καθιστά μη πρακτική. Ένα χρόνο αργότερα και στην προσπάθεια να μειώσουν το υπολογιστικό κόστος οι Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, Sungwoong Kim υλοποίησαν το Fast AutoAugment. [87] Οι καινοτομία του είναι ότι αντί να χρησιμοποιεί μάθηση με ενίσχυση για την εύρεση των παραμέτρων, μετατρέπει το πρόβλημα της αναζήτησης σε πρόβλημα πυκνότητας και χρησιμοποιεί Bayesian optimization για την ευρεση των βέλτιστων μετατροπών. Την ίδια περίοδο, οι D Ho, E Liang, X Chen, I Stoica, P Abbeel δημοσίευσαν και το δικό τους μοντέλο το Population-based Augmentation. [88] Πάλι αντί να χρησιμοποιήσουν μάθηση με ενίσχυση χρησιμοποίησαν κάτι που μοιάζει με εξελικτικούς αλγορίθμους [89] δηλαδή χρησιμοποίησαν χρονοδιαγράμματα για να αναπαραστήσουν τις μετατροπές, οργανωμένες σε πληθυσμο και στην συνέχεια προσπαθούν να βρουν το βέλτιστο χρονοδιάγραμμα εξελίσοντας τον πληθυσμό. Παρόλα αυτά, και οι δύο παραπάνω τρόποι δεν είναι βέλτιστοι καθώς απαιτείται σημαντικός χρόνος. Έτσι, το 2020, δημοσιεύτηκε το DADA: Differentiable Automatic Data Augmentation. [78] Η καινοτομία του σε σχέση με τα προηγούμενα είναι ότι χρησιμοποιεί μια διαφορική συνάρτηση για να αναπαραστήσει τις μεταβλητές έτσι ώστε να καταφέρει να μειώσει τον χρόνο εκτέλεσης του μοντέλου, κάνοντας εκπαίδευση και στα model parameters και στα augmentation parameters την ίδια στιγμή.

Αποφάσισα λοιπόν να δοκιμάσω κατα πόσο το DADA μπορεί να χρησιμοποιηθεί σε ένα πρόβλημα ταξινόμησης και κατα πόσο μπορεί να αυξήσει τα αποτελέσματα ενός ταξινομητή. Το έκανα αυτό επειδή υπάρχουν πολλά προβλήματα που θέλουμε την μέγιστη ακρίβεια, και γι αυτό τον λόγο μια πιο αυτοματοποιημένη και βελτιωμένη τεχνική μπορεί να βοηθήσει πάρα πολύ.

Κεφάλαιο 1ο: Μηχανική Μάθηση

1.1 Ορισμός Μηχανικής Μάθησης

Η μάθηση είναι η διαδικασία που διαθέτουν όλοι οι νοήμονες ζωντανοί οργανισμοί και αναφέρεται στην ικανότητα τους να αποκτούν πληροφορίες και δεξιότητες μέσω του περιβάλλοντος.[1]

Μηχανική μάθηση είναι ένα υπο-πεδίο της επιστήμης των υπολογιστών που ασχολείται με την μελέτη και δημιουργία αλγορίθμων οι οποίοι "μαθαίνουν" χωρίς να έχουν προγραμματιστεί αποκλειστικά γι' αυτό. Αυτό το καταφέρνουν μέσω της αναγνώρισης προτύπων και συσχετίσεων ανάμεσα στα δεδομένα με σκοπό την λήψη αποφάσεων ή προβλέψεων(ανάλογα με το πρόβλημα).
Ο Tom M.Mitchell έδωσε έναν πιο επίσημο ορισμό[2]:

*Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία **E** ως προς μια κλάση εργασιών **T** και ένα μέτρο επίδοσης **P**, αν η επίδοσή του σε εργασίες της κλάσης **T**, όπως εκτιμάται από το μέτρο **P**, βελτιώνεται με την εμπειρία **E**.*

1.2 Ιστορική Αναδρομή

Ακολουθεί μία μικρή ιστορική αναδρομή από την αρχή της μηχανικής μάθησης έως και σήμερα: [3][4][5]

- Το **1950**, ο Alan Turing δημιουργεί ένα τεστ, "Turing Test", το οποίο εξετάζει κατα πόσο μια μηχανή επιδεικνύει ευφυή συμπεριφορά η οποία δεν έχει διαφορά από την συμπεριφορά ενός ανθρώπου.
- Το **1956**, οι Marvin Minsky, John McCarthy, Claude Shannon και Nathan Rochester σε μια διάσκεψη στο Dartmouth δημιουργούν τον όρο "Τεχνητή Νοημοσύνη".
- Το **1957**, ο ψυχολόγος Frank Rosenblatt, έχοντας ως πρότυπο το ανθρώπινο νευρικό σύστημα, δημιουργεί το πρώτο νευρωνικό δίκτυο, το Perceptron.
- Το **1967**, δημιουργείται ο αλγόριθμος κοντινών γειτόνων (Nearest Neighbour algorithm) και δίνεται στους υπολογιστές η ικανότητα τα αναγνωρίζουν επαναλαμβανόμενα μοτίβα.
- Το **1973**, η αγγλική κυβέρνηση και διάφοροι φορείς σταματούν να χρηματοδοτούν την έρευνα πάνω στην τεχνητή νοημοσύνη. Ιστορικά αυτό το γεγονός είναι γνωστό ως "χειμώνας της τεχνητής νοημοσύνης" ("AI winter").
- Το **1979**, μια ερευνητική ομάδα του πανεπιστημίου του Στάνφορντ καταφέρνει να δημιουργήσει ένα ρομπότ, το Καρτ ("the Cart") το οποίο μπορεί να κινείται αυτόνομα και να αποφεύγει εμπόδια μέσα σε ένα δωμάτιο.
- Το **1981**, ο Gerald Dejong παρουσίασε την ιδέα του για μάθηση βασισμένη στη μάθηση (Explanation-Based Learning). Σε αυτήν, ο υπολογιστής αναλύει τα δεδομένα που του έχουν δοθεί με σκοπό την δημιουργία γενικευμένων κανόνων.
- Το **1982**, ο John Hopfield δημιουργεί το δίκτυο του Hopfield (Hopfield Network), το οποίο είναι ένα από τα πρώτα αναδρομικά νευρωνικά δίκτυα
- Το **1986**, οι ψυχολόγοι David Rumelhart και James McClelland δημοσιεύουν ένα άρθρο το οποίο περιγράφει λεπτομερώς μία διαδικασία που ονομάζεται παράλληλη κατανεμημένη επεξεργασία (Parallel Distributed Processing) η οποία ουσιαστικά είναι η χρήση νευρωνικών δικτύων για να λυθούν προβλήματα μηχανικής μάθησης.
- Το **1989**, ο Christopher Watkins ανέπτυξε το Q-learning, έναν αλγόριθμο ενισχυτικής μάθησης χωρίς μοντέλο (Model-Free Reinforcement Learning algorithm).

- Το **1992**, ο Gerald Tesauro εφηύρε ένα πρόγραμμα το οποίο μπορούσε να παίξει τάβλι και το ονόμασε TD-Gammon. Το νευρωνικό αυτό μπορούσε να ανταγωνιστεί τους κορυφαίους παίκτες ταβλιού.
- Το **1995**, οι Vladimir Vapnik και Corinna Cortes δημοσίευσαν ένα άρθρο σχετικά με μηχανές διανυσμάτων υποστήριξης (Support Vector Machines - SVMs).
- Το **1996**, το Deep Blue, ένα πρόγραμμα που είχε εκπαιδευτεί στο σκάκι κατάφερε να νικήσει τον παγκόσμιο πρωταθλητή του σκακιού, Garry Kasparov.
- Το **2006**, ο Geoffrey Hinton επινοεί τον όρο "βαθιά μάθηση" ("Deep Learning") για να περιγράψει τις νέες αρχιτεκτονικές των νευρωνικών δικτύων ικανές να ανταπεξέλθουν σε πιο πολύπλοκα προβλήματα.
- Το **2009**, η Fei-Fei Li δημιουργεί ένα από τα μεγαλύτερα σύνολα δεδομένων (dataset), του οποίου ο κύριος στόχος ήταν η αναγνώριση αντικειμένων (object recognition), το ImageNet.
- Το **2011**, το Watson της IBM καταφέρνει να νικήσει 2 πρωταθλητές του τηλεοπτικού παιχνιδιού Jeopardy.
- Το **2012**, η ομάδα του Google Brain δημιουργεί ένα βαθύ νευρωνικό που μπορεί να αναγνωρίσει γάτες από βίντεο στο Youtube.
- Το **2014**, το Facebook αναπτύσσει το DeepFace, ένα βαθύ νευρωνικό ικανό να αναγνωρίσει άτομα από φωτογραφίες όσο αποτελεσματικά όσο και ένας άνθρωπος.
- Το **2015**, το DeepMind της Google καταφέρνει να παίξει 29 παιχνίδια Atari έχοντας εκπαιδευτεί σε ένα γενικό πλαίσιο, όχι στο κάθε παιχνίδι ξεχωριστά, από βίντεο.
- Το **2016**, το AlphaGo νικάει τον πρωταθλητή του Go, χρησιμοποιώντας τεχνικές αναζητήσεις δέντρων (tree search techniques).

1.3 Τρόποι Χρήσης Μηχανικής Μάθησης

Η μηχανική μάθηση έχει εφαρμογές σε πολλούς τομείς με σκοπό την επίτευξη διαφορετικών αποτελεσμάτων ανάλογα με τον τρόπο χρήσης [6].

- Μία από τις εφαρμογές της, που έχει αναπτυχθεί τα τελευταία χρόνια είναι η **αυτοματοποίηση διαδικασιών** που εκτελούνται στην καθημερινότητα μας. Στόχος της είναι η μείωση των σφαλμάτων που οφείλονται σε ανθρώπινους παράγοντες, όπως η έλλειψη γνώσεων ή η κούραση. Για παράδειγμα, ένα αυτο-οδηγούμενο αυτοκίνητο μπορεί να αναλύει δεδομένα πραγματικού χρόνου από διάφορους αισθητήρες που διαθέτει (π.χ. σήματα, κλήση δρόμου, καιρικές συνθήκες, θέση πεζών) και να πάρει αποφάσεις για την καλύτερη ενέργεια που πρέπει να ακολουθήσει προκειμένου να πετύχει τον στόχο του μέσα σε πάρα πολύ λίγο χρόνο, είτε είναι πλοήγηση είτε είναι παρκάρισμα, προσφέροντας έτσι αυτοματοποιημένη οδήγηση χωρίς των ανθρώπινο παράγοντα και μείωση των ατυχημάτων. Ένα πιο απλό παράδειγμα θα ήταν η αυτοματοποιημένη παροχή τροφής στα ζώα των κτηνοτροφικών μονάδων.
- Ένα ακόμα παράδειγμα που υπάρχει στην καθημερινή ζωή του ανθρώπου και είναι γνωστό σε όλους μας είναι η χρήση της μηχανικής μάθησης για την **δημιουργία συστάσεων**. Με πρωταρχικό στόχο την αύξηση των πωλήσεων και την επίτευξη της ικανοποίησης των πελατών, οι εταιρείες και οι ιστότοποι αναλύουν τα δεδομένα πλοήγησης και αγορών των χρηστών προκειμένου να δημιουργήσουν εξατομικευμένες διαφημίσεις και προτάσεις.
- Επίσης, μπορεί να χρησιμοποιηθεί για την αναγνώριση φωνής και φωνητικών εντολών. Αυτό μπορεί να βοηθήσει χιλιάδες συνανθρώπους που αντιμετωπίζουν προβλήματα όρασης αλλά

ταυτόχρονα το συναντάμε πολύ πιο συχνά με την μορφή των προσωπικών βοηθών που διαθέτουν πολλές ηλεκτρονικές συσκευές.

- Η μηχανική μάθηση μπορεί επίσης να παίξει σημαντικό ρόλο στη **λήψη αποφάσεων**, εκμεταλλευόμενη την δυνατότητά της να αναλύει μεγάλους όγκους δεδομένων και να παράγει συμπεράσματα βασιζόμενη σε αναλύσεις που πραγματοποίησε σε πάρα πολύ λίγο χρόνο με αυξημένη ακρίβεια. Στην **ιατρική** για παράδειγμα, οι γιατροί βοηθούνται από μοντέλα μηχανικής μάθησης στην ανάλυση ιατρικών δεδομένων και το ιστορικό των ασθενών, για την διάγνωση και την επιλογή της κατάλληλης θεραπείας. Επιπλέον, μπορεί να χρησιμοποιηθεί στον τομέα της **φαρμακολογίας** για την ανάπτυξη νέων φαρμάκων και τον προσδιορισμό των κατάλληλων δοσολογιών. Και στους δύο παραπάνω τομείς, όπου οι λάθος αποφάσεις μπορούν να έχουν σοβαρές συνέπειες, η εφαρμογή της μηχανικής μάθησης ως μια δευτερεύουσα γνώμη μπορεί να αποβεί σωτήρια.
- Επίσης, μηχανική μάθηση πλέον χρησιμοποιούμε για αυτόματη **αναγνώριση αντικειμένων** [6] μέσα σε οπτικά μέσα, όπως εικόνες και βίντεο. Αυτό μπορεί να βοηθήσει ιδιαίτερα σε τομείς όπως και την **ιατρική** με την αναγνώριση όγκων ή ασθενειών σε ακτινογραφίες αλλά και σε τομείς όπως η **ασφάλεια** με την αναγνώριση προσώπων και αναγνώριση ύποπτων μοτίβων που μπορεί να βοηθήσει σε κάποια έρευνα ή στην αποφυγή κάποιου ατυχήματος.
- Τα τελευταία χρόνια, η μηχανική μάθηση χρησιμοποιείται για την **πρόβλεψη αποτελεσμάτων και διακυμάνσεων** βασιζόμενη σε προηγούμενα δεδομένα από παρόμοια προϊόντα ή καταστάσεις. Στον τομέα της **οικονομίας**, τέτοια μοντέλα χρησιμοποιούνται για την πρόβλεψη των μετοχών στο χρηματιστήριο και την αύξηση των κερδών, καθώς και για την εισαγωγή νέων προϊόντων σε ένα κατάστημα με βάση παρόμοια προϊόντα, βοηθώντας στην επιλογή της κατάλληλης τιμής. Πολλές επιχειρήσεις χρησιμοποιούν μοντέλα μηχανικής μάθησης για την βελτιστοποίηση του **supply chain** τους, διασφαλίζοντας τον έγκαιρο εφοδιασμό των προϊόντων στα ράφια τους, ενώ παράλληλα αποτρέπουν τις ελλείψεις προϊόντων.

Αυτές είναι μόνο μερικές από τις χιλιάδες εφαρμογές που έχει η μηχανική μάθηση, και στα επόμενα χρόνια αναμένεται να συναντήσουμε ακόμα περισσότερα μοντέλα μηχανικής μάθησης με πολύ μεγαλύτερη συχνότητα στην καθημερινή μας ζωή. Ωστόσο, τι εννοούμε ακριβώς με τον όρο “αλγόριθμοι μηχανικής μάθησης”;

1.4 Αλγόριθμοι Μηχανικής Μάθησης

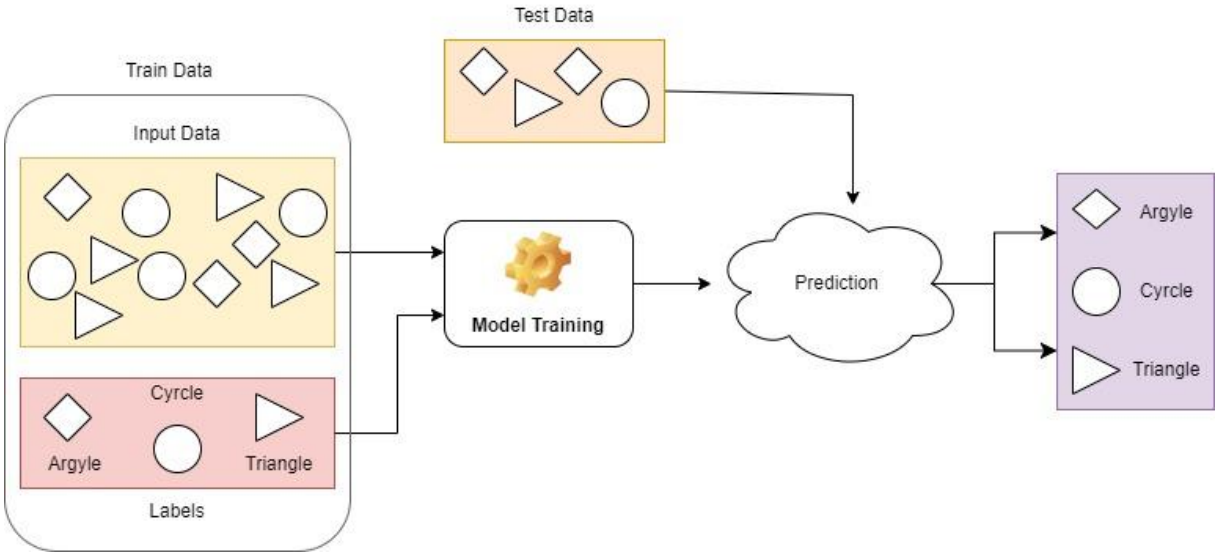
Οι αλγόριθμοι μηχανικής μάθησης χωρίζονται σε τρεις μεγάλες κατηγορίες: στους αλγόριθμους με επίβλεψη (**Supervised Learning**), στους αλγόριθμους χωρίς επίβλεψη (**Unsupervised Learning**) και στους αλγόριθμους με μάθηση με ενίσχυση (**Reinforcement Learning**). Κάπου ανάμεσα στη μάθηση με επίβλεψη και μάθηση χωρίς επίβλεψή υπάρχει και η **ημι-επιβλεπόμενη μάθηση**. Η κυρία διάφορα των κατηγοριών αυτών είναι ο τρόπος ανατροφοδότησης του μοντέλου.[8]

1.4.1 Επιβλεπόμενη Μάθηση (Supervised Learning)

Η επιβλεπόμενη μάθηση[9] είναι από τους πιο διαδεδομένους τύπους αλγορίθμων μηχανικής μάθησης σε ό,τι αφορά την αναγνώριση μοτίβων και την ταξινόμηση δεδομένων. Το μοντέλο σαν δεδομένα εισόδου δέχεται τις πληροφορίες πάνω στις οποίες θα εκπαιδευτεί και τα επιθυμητά αποτελέσματα, προσπαθώντας να βρει τον γενικό κανόνα που συνδέει αυτά τα δύο. Όσο τα δεδομένα τροφοδοτούνται στο μοντέλο, αυτό προσαρμόζει τα βάρη του κατάλληλα ανάλογα με τους στόχους που του έχουν δοθεί από τον άνθρωπο. Μετά την ολοκλήρωση της διαδικασίας εκπαίδευσης, ο αλγόριθμος

“προβλέπει” το αποτέλεσμα από ένα δείγμα δεδομένων χωρίς να έχει τους στόχους. Μερικά μοντέλα αυτής της κατηγορίας είναι:

- Νευρωνικά Δίκτυα (Neural Networks)
- Naive Bayes
- Γραμμικής Παλινδρόμησης (Linear Regression)
- Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines - SVMs)
- Κ-κοντινότεροι γείτονες (K-nearest neighbour)
- Δέντρα απόφασης (Decision Trees)



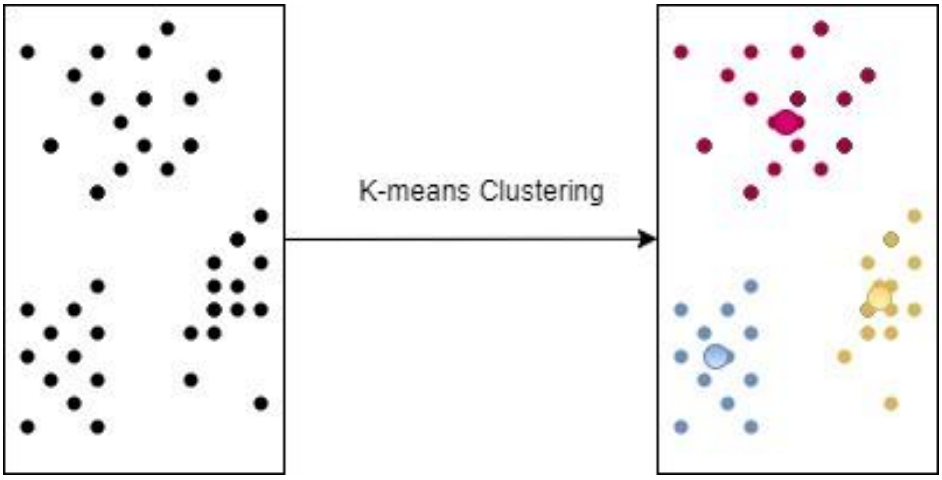
Εικ. 1.1 Παράδειγμα Supervised Learning

1.4.2 Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

Οι μην επιβλεπόμενοι αλγόριθμοι, σε αντίθεση με τους supervised, δεν έχουν τους στόχους σαν δεδομένα εισόδου. Αυτοί οι αλγόριθμοι καταφέρνουν να ανακαλύψουν μοτίβα και ομαδοποιήσεις δεδομένων (Clustering) χωρίς την ανθρώπινη παρέμβαση[10].

Η ομαδοποίηση των δεδομένων μπορεί να γίνει σε πολλά επίπεδα και εξαρτάται από την δομή του dataset. Κάποια από αυτά είναι:

- Exclusive and Overlapping Clustering (παράδειγμα αλγόριθμος K-means)
- Hierarchical Clustering
- Probabilistic Clustering



Εικ. 1.2 Αλγόριθμος K-means για ομαδοποίηση των δεδομένων

Επίσης, unsupervised learning μπορεί να χρησιμοποιηθεί και για την μείωση των διαστάσεων των δεδομένων[11] (dimensionality reduction). Πολλές φορές τα χαρακτηριστικά (features) του dataset θα είναι πάρα πολλά, γεγονός που θα έχει επίπτωση στην απόδοση του μοντέλου και καθιστά δυσκολότερη την οπτικοποίηση των δεδομένων (data visualization). Γι' αυτό το λόγο με την μείωση των διαστάσεων των δεδομένων καταφέρνουμε να μειώσουμε τα data inputs κρατώντας την ακεραιότητα του dataset. Μερικές τεχνικές είναι:

- Principal Component Analysis
- Singular Value Decomposition
- Autoencoders

1.4.3 Μάθηση Με Μερική Επίβλεψη (Semi-Supervised Learning)

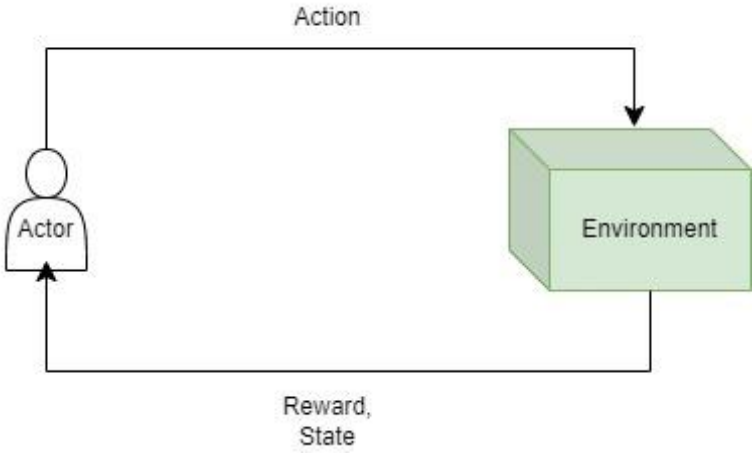
Όπως φανερώνει και το όνομα, αυτή η κατηγορία βρίσκεται ανάμεσα στο supervised και unsupervised learning, επειδή πολύ συχνά, στο dataset του προβλήματος δεν έχουμε όλους τους στόχους είτε γιατί έχουμε κάποιους περιορισμούς είτε γιατί δεν μπορούμε να τους ξέρουμε[12]. Συνήθως χρησιμοποιούμε έναν συνδυασμό μοντέλων από τις άλλες δύο κατηγορίες για την επίλυση των προβλημάτων.

1.5 Μάθηση Με Ενίσχυση (Reinforcement Learning)

Το Reinforcement Learning, διαφέρει πάρα πολύ από τις υπόλοιπες κατηγορίες. Η βασική ιδέα υλοποίησης του ξεκίνησε από τα παιχνίδια. Διαφέρει από τις άλλες κατηγορίες γιατί δεν χρειάζεται να του δοθούν δεδομένα που είναι ήδη γνωστός ο στόχος και δεν χρειάζεται να διορθωθούν όλες οι μη βέλτιστες αποφάσεις του. Αυτό γιατί το μοντέλο μαθαίνει από μόνο του αλληλεπιδρώντας με το περιβάλλον και προσπαθώντας να αποκτήσει την μέγιστη συνολική ανταμοιβή. Μια απόφαση που για μια συγκεκριμένη παρτίδα είναι λανθασμένη μπορεί σε μια επόμενη να είναι από τις καλύτερες κινήσεις. Γι αυτό το λόγο δεν αρκεί να επιλέγουμε κίνηση με βάση την πιο σωστή απόφαση φαινομενικά την δεδομένη χρονική στιγμή αλλά αφήνουμε ένα ποσοστό στο μοντέλο να κάνει explore

άλλες κινήσεις η οποίες με βάση προηγούμενες παρτίδες δεν ήταν βέλτιστες αλλά μπορεί να είναι τώρα. Ας επιστρέψουμε λίγο στα βασικά όμως.

Αρχικά πρέπει να οριστεί ένα περιβάλλον (**environment**) το οποίο έχει συγκεκριμένους κανόνες και στην συνέχεια ο πράκτορας (**agent**) αλληλεπιδρά με αυτό με βάση κάποιων κινήσεων (**actions**) για την επίτευξη κάποιου στόχου. Όταν ο agent "φτάσει" σε κάποιον στόχο τότε του δίνουμε την κατάλληλη ανταμοιβή (**reward**). Για να υπολογιστεί η ανταμοιβή πρέπει να παρατηρηθούν ποιες κινήσεις οδήγησαν τον πράκτορα πιο κοντά στον στόχο του, έτσι ώστε να προτιμήσει αυτά τα actions στο μέλλον. Το reward μπορεί να είναι είτε αρνητικό είτε θετικό ανάλογα με το τι actions επέλεξε ο agent και κατά πόσο τον βοήθησαν ή τον απομάκρυναν από τον στόχο του. Μαζί με το reward ο agent θα λάβει και το καινούργιο state, δηλαδή την εικόνα του περιβάλλοντος μετά από το αποτέλεσμα της κίνησης που έκανε. [13]



Εικ. 1.3 Μάθηση με ενίσχυση

Ας φέρουμε ως παράδειγμα ένα παιχνίδι σκακιού. Το enviroment θα είναι η σκακιέρα, ο agent θα είναι το μοντέλο μηχανικής μάθησης, τα actions θα είναι το πιο πιόνι θα κουνήσει καθώς και οι πιθανές θέσεις που μπορεί να πάει και το state θα είναι η τρέχουσα κατάσταση της σκακιέρας και ο στόχος θα είναι η νίκη. Ο πράκτορας θα κάνει evaluate κοιτώντας όχι μόνο την κίνηση που έκανε τώρα αλλά και το πόσο κοντά τον έφερε αυτή η κίνηση στην νίκη. Όταν τελειώσει η παρτίδα, θα επαναφέρουμε το enviroment στην αρχική του κατάσταση και θα αρχίσουμε καινούργια παρτιδα. [14]

Ένας πράκτορας ενισχυτικής μάθησης απαρτίζεται από επιμέρους κομμάτια που ορίζουν τον "τρόπο που ενεργεί". Αρχικά έχει κάτι που ονομάζεται συνάρτηση πολιτικής (**policy function** ή απλά **policy**), η οποία δεν είναι τίποτα άλλο από μία συσχέτιση ανάμεσα της τωρινής κατάστασης που βρίσκεται το περιβάλλον και σε μία κατανομή πιθανοτήτων όλων των πιθανών ενεργειών. Με λίγα λόγια είναι η στρατηγική που θα ακολουθήσει ο πράκτορας έτσι ώστε να πετύχει τους στόχους του. Μπορεί να είναι είτε ντετερμινιστική (**deterministic policy**) είτε στοχαστική (**stochastic policy**). Έχει επίσης μια συνάρτηση αξίας (**value function**), η οποία είναι μια αναπαράσταση για το πόσο καλό είναι ένα action/η και state. Στην πραγματικότητα είναι μια πρόβλεψη μελλοντικής ανταμοιβής και πόσο κοντά θα φέρει το μοντέλο στην επίτευξη του στόχου. Τέλος, υπάρχει και το μοντέλο (**model**) που είναι η αναπαράσταση του περιβάλλοντος από τον πράκτορα.[13]

Με βάση τις παραπάνω έννοιες οι πράκτορες χωρίζονται στις εξής υποκατηγορίες:

- **Value Based**: ο πράκτορας δεν έχει συγκεκριμένη πολιτική αλλά επιλέγει actions άπληστα και με κύριο κριτήριο την τωρινή κατάσταση του περιβάλλοντος,
- **Policy Based**: ο πράκτορας δεν έχει value function για να επιλέξει κινήσεις αλλά επιλέγει με βάση το policy,
- **Model Based**: ο πράκτορας έχει και χρησιμοποιεί πολιτική /ή και value function και έχει μοντέλο,
- **Model Free:** ο πράκτορας έχει και χρησιμοποιεί πολιτική/ή και value function αλλά δεν έχει μοντελο

1.6 Σύνολα Δεδομένων

Λόγω του ότι τα δεδομένα έχουν άμεση σχέση με την επιτυχία του μοντέλου, χωρίζονται σε τρία υποσύνολα για την καλύτερη αξιοποίηση τους:

- **Σύνολο Εκπαίδευσης (train set):** Το train set είναι το κύριο σύνολο δεδομένων και συνήθως πρόκειται για το πιο μεγάλο σετ. Χρησιμοποιείται για την εκπαίδευση του μοντέλου και βάση αυτού το μοντέλο θα αλλάξει τις παραμέτρους του.

- **Σύνολο Επικύρωσης (Validation Set):** Το validation set χρησιμοποιείται κατά την διάρκεια της εκπαίδευσης για την αξιολόγηση της μέχρι στιγμής απόδοσης του μοντέλου. Μπορεί να χρησιμοποιηθεί για την αξιολόγηση διαφορετικών τιμών υπερπαραμέτρων και την πιθανή ρύθμισή τους κατά την εκπαίδευση. Επίσης, χρησιμοποιείται για την αξιολόγηση, του κατά πόσο ένα μοντέλο έχει κάνει overfit στο σύνολο εκπαίδευσης, σταματώντας την εκπαίδευση εάν αυτό κριθεί απαραίτητο (εάν δηλαδή το σφάλμα εκπαίδευσης μειώνεται, ενώ το σφάλμα επικύρωσης αυξάνεται).

- **Σύνολο Δοκιμής (Test Set):** Το test set χρησιμοποιείται για την αξιολόγηση της τελικής απόδοσης του μοντέλου. Συνήθως είναι ένα σύνολο όπου το μοντέλο το συναντά πρώτη φορά και χρησιμοποιείται σαν τελικό τεστ της απόδοσης.

1.7 Underfitting - Overfitting

Κάθε μοντέλο μηχανικής μάθησης στην πραγματικότητα προσπαθεί να βρει την χρυσή τομή ανάμεσα σε αυτούς τους δύο όρους, επειδή αυτοί συνδέονται με την αποτελεσματικότητα του μοντέλου. Ο σκοπός των μοντέλων μηχανικής μάθησης είναι η εξαγωγή χαρακτηριστικών ή συμπερασμάτων από το **training set** και η εφαρμογή τους σε άγνωστα δεδομένα που τα βλέπει πρώτη φορά το μοντέλο, το **test set**. Πολλές φορές χρησιμοποιούμε και το **validation set** για να αξιολογήσουμε το μοντέλο και να αλλάξουμε τις παραμέτρους του.[16] Αν το μοντέλο δεν εκπαιδευτεί αρκετα και δεν μπορεί να εξάγει ένα σωστο αποτέλεσμα για το test set, τότε έχουμε το φαινόμενο του **underfitting**. Συνήθως underfitting έχουμε για μια πληθώρα από λόγους. Μερικοί από αυτούς είναι:

- Μεγάλο ποσοστό λάθους στο training set,
- Το dataset είναι μικρό ή έχουμε υπερβολικά πολλά δείγματα από την μία κλάση,
- Το μοντέλο που χρησιμοποιούμε είναι πολύ απλό και δεν μπορεί να γενικεύσει κατάλληλα,
- Το dataset έχει υπερβολικά πολλά features που αυξάνουν την πολυπλοκότητα του προβλήματος χωρίς να βελτιώνουν την αξιοπιστία.

Κάποιοι από τους τρόπους καταπολέμησης του underfitting είναι:

- Αύξηση της διάρκειας της εκπαίδευσης (**epoch**) για την βελτίωση τον αποτελεσμάτων,
- Πραγματοποίηση κατάλληλου preprocessing στο dataset έτσι ώστε να γίνουν οι απαραίτητες ενέργειες (noise reduction, data augmentations, etc) ,
- Αύξηση των κρυφών στρωμάτων (**hidden layers**) του μοντέλου.

Το overfitting αντίθετα, είναι όταν το μοντέλο εκπαιδευτεί πάρα πολύ στο training set, τότε θα αρχίσει να μαθαίνει και από τον θόρυβο και τα λανθασμένα data του training set. Αυτό θα έχει ως αποτέλεσμα μεγάλο ποσοστό λάθους στο test set επειδή έχει μάθει υπερβολικά καλά το training set. Μερικοί λόγοι για το overfitting είναι:

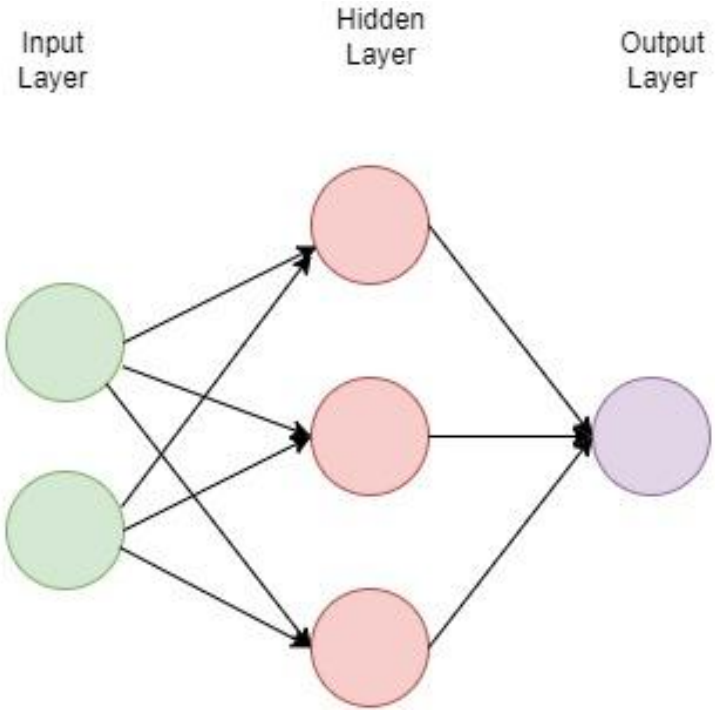
- Μικρό ποσοστό λάθους στο training set και μεγάλο στο test set,
- Το μοντέλο είναι υπερβολικά περίπλοκο,
- Το training set είναι μικρό.

Κάποιοι από τους τρόπους αντιμετώπισης του overfitting είναι:

- Μείωση της διάρκειας εκπαίδευσης,
- Μείωση των κρυφών στρωμάτων του μοντέλου,
- Αύξηση των δειγμάτων του training set [15]

1.8 Τεχνητά Νευρωνικά Δίκτυα

Τα Τεχνητά Νευρωνικά Δίκτυα είναι ένας από τους πυλώνες της μηχανικής μάθησης. Όπως φανερώνει και το όνομα τους, είναι “νευρωνικά” γιατί έχουν εμπνευστεί από το ανθρώπινο νευρικό σύστημα και δίκτυα γιατί έχουν πολλούς κόμβους που συνδέονται μεταξύ τους δημιουργώντας ένα δίκτυο. Καθένας από αυτούς τους κόμβους δέχεται μια η περισσότερες εισόδους, είτε από άλλους νευρώνες είτε σαν είσοδο στο μοντέλο, πραγματοποιεί κάποιους υπολογισμούς και έχει και μία έξοδο, που είτε με την σειρά του την τροφοδοτεί σε άλλους νευρώνες είτε αποτελεί την έξοδο του μοντέλου. Συνήθως τους νευρώνες τους ομαδοποιούμε σε στρώματα (**layers**).[17]



Εικ. 1.4 Απλό νευρωνικό δίκτυο

Γενικά θεωρούμε ότι τα νευρωνικά δίκτυα έχουν την παρακάτω δομή:

- Το στρώμα εισόδου (**input layer**) που αποτελείται από τους νευρώνες εισόδου. Οι νευρώνες αυτοί δεν πραγματοποιούν κανέναν υπολογισμό, απλά προωθούν τα δεδομένα εισόδου (data) στους υπολογιστικούς νευρώνες του επόμενου στρώματος. Οι νευρώνες αυτοί μπαίνουν αυστηρά και μόνο στην αρχή του νευρωνικού και κάθε μοντέλο έχει αυστηρα ένα στρώμα εισόδου.
- Το στρώμα εξόδου (**output layer**) που αποτελείται από τους νευρώνες εξόδου. Ομοίως, με τους νευρώνες εισόδου δεν πραγματοποιούν κανέναν υπολογισμό έτσι ώστε να δώσουν στο επόμενο στρώμα και οι τιμές τους "προκύπτουν¨ από τους υπολογισμούς στο προτελευταιο στρώμα. Η μόνη τους υποχρέωση είναι να εξάγουν τα αποτελέσματα του μοντέλου. Ομοίως, με το στρώμα εισόδου μπορεί να υπάρξει μόνο ένα σε κάθε μοντέλο. Συνήθως οι νευρώνες στο στρώμα εξόδου είναι ίσοι με τους πιθανούς στόχους. Πολλες φορές χρησιμοποιούμε και συναρτήσεις ενεργοποίησης σε αυτο το στρώμα.
- Το κρυφό στρώμα (**hidden layer**) που αποτελείται από τους κρυφούς νευρώνες ή νευρώνες υπολογισμών. Αυτοί οι νευρώνες παίρνουν την είσοδο που τους έχει δοθεί και την πολλαπλασιάζουν με μία τιμή που ονομάζεται βάρος (**weight**). Μετά περνάνε την τιμή που θα προκύψει από μια συνάρτηση που ονομάζεται συνάρτηση ενεργοποίησης (**activation function**) και ανάλογα δίνεται σαν είσοδος σε κάποιο άλλο νευρώνα. Δεν υπάρχει κάποιος περιορισμός στον αριθμό των κρυφών στρωμάτων που μπορεί να έχει ένα μοντέλο, αλλά τα περισσοτερα δεν είναι πάντα και καλύτερα καθώς αυξάνεται η πολυπλοκότητα του.[19]

Αν θέλαμε να περιγράψουμε μαθηματικά τον νευρώνα τότε ένας νευρώνας θα ήταν ένα μονοδιάστατο διάνυσμα από τα βάρη $\boldsymbol{w}$ του μαζί με ένα επιπλέον συναπτικό βάρος για την πόλωση $\boldsymbol{b}$ (**bias**). Θα είχε την μορφή:

$$w = \left[w_1 w_2 \ldots w_n \right]$$

Μαθηματικός τύπος 1.1

Για την έξοδο ενός νευρώνα θα χρησιμοποιούσαμε την παρακάτω εξίσωση:

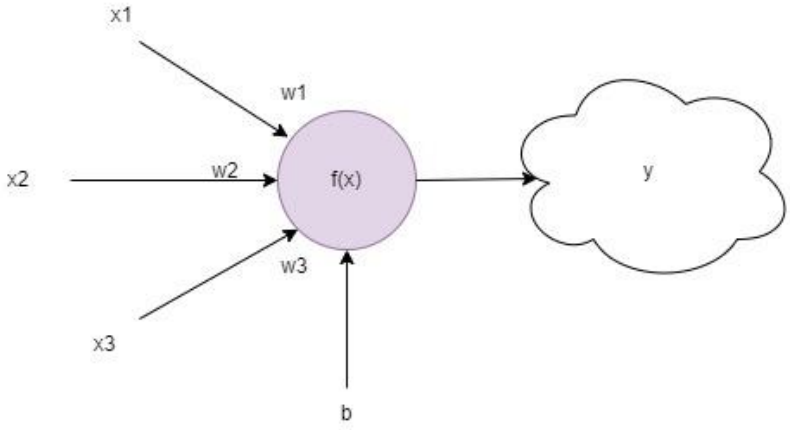
$$y_k = f\left(\sum_{i=1}^{N} x_i w_i + b \right)$$

Μαθηματικός τύπος 1.2

όπου *x* είναι η *i* είσοδος του προβλήματος, *w* είναι το βάρος της *i* εισόδου, *b* είναι η πόλωση του νευρώνα (συνήθως δεν εξαρτάται από το πρόβλημα) και η *f* είναι η συνάρτηση ενεργοποίησης.[19]

1.8.1 Perceptron

Το απλό Perceptron ή αλλιώς δυαδικός ταξινομητής είναι η πιο απλή μορφή που μπορεί να έχει ένα νευρωνικό δίκτυο. Perceptron αλλιώς ονομάζεται και ένας μόνο νευρώνας.



Εικ. 1.5 Perceptron

Αποτελείται από μόνο ένα στρώμα με μόνο ένα νευρώνα, την πόλωση, τα βάρη και την συνάρτηση ενεργοποίησης.

Μαθηματικα θα μπορούσε να αναπαρασταθεί απο την συνάρτηση:

$$f(x) = \begin{cases} 1, & wx + b > 0 \\ 0, & \sigma\varepsilon\ \kappa\acute{\alpha}\theta\varepsilon\ \acute{\alpha}\lambda\lambda\eta\ \pi\varepsilon\rho\acute{\iota}\pi\rho\omega\sigma\eta \end{cases}$$

Μαθηματικός τύπος 1.3

Γραφικά το perceptron χαράζει στο επίπεδο μια ευθεια γραμμη και το χωρίζει σε δυο μερη, χωρίζοντας τις δύο κλάσεις όσο πιο διακριτά γίνεται. [20]

1.9 Πολυστρωματικό Perceptron (Multilayer Perceptron - MLP)

Σε πραγματικά σενάρια όμως, τα dataset είναι πολύπλοκα και ως προς την κατανομή τους στον χώρο και ως προς τα επίπεδα στα οποία θα πρέπει να χωριστούν. Αυτο πολλές φορες καθιστά αδύνατον να χωριστούν γραμμικά γι' αυτο δημιουργήθηκαν τα Multilayer Perceptron.

Το Multilayer Perceptron σαν ορος χρησιμοποιείται για να περιγράψει ενα νευρωνικο δίκτυο με πολλά Perceptrons. Ένα MLP έχει την δομή που περιγράφεται στην εικ.1.4 αποτελείται δηλαδή από ένα στρώμα εισόδου, ένα τουλάχιστον κρυφό στρώμα και ένα στρώμα εξόδου. Εκτός από το στρώμα εισόδου όλοι οι υπόλοιποι νευρώνες στα υπόλοιπα στρώματα χρησιμοποιούν μια μη γραμμική συνάρτηση ενεργοποίησης, η οποία μπορεί να διαφέρει από στρώμα σε στρώμα. Λόγω της μη γραμμικής συνάρτησης ενεργοποίησης και των πολλών κρυφών στρωμάτων, τα MLPs καταφέρνουν με μεγάλη επιτυχία να διαχωρίζουν πολλά περισσότερα dataset, από ότι μπορεί να χωρίσει ένας απλός Perceptron, είτε είναι γραμμικά είτε όχι. Επίσης μια ακόμα ιδιαιτερότητα τους είναι ότι όλα τα στρώματα τους είναι πλήρως συνδεδεμένα στρώματα (**fully connected layers**) δηλαδή όλοι οι νευρώνες του προηγούμενου στρώματος συνδέονται με όλους τους νευρώνες του επόμενου στρώματος. [21]

Τα περισσότερα νευρωνικά δίκτυα είτε μιλάμε για απλα Perceptron είτε για MLPs, έχουν μια φορά προς τα μπροστά όπου τα δεδομένα εισέρχονται από το στρώμα εισόδου, περνάνε τα κρυφά στρώματα και παράγουν αποτελέσματα στο στρώμα εξόδου χωρίς κάποια επανάληψη ή κάποια κυκλική φορά. Αυτα ονομάζονται και δίκτυα πρόσθιας τροφοδότησης (**feedforward neural network**).[22]

1.10 Είδη Νευρωνικών Δικτύων

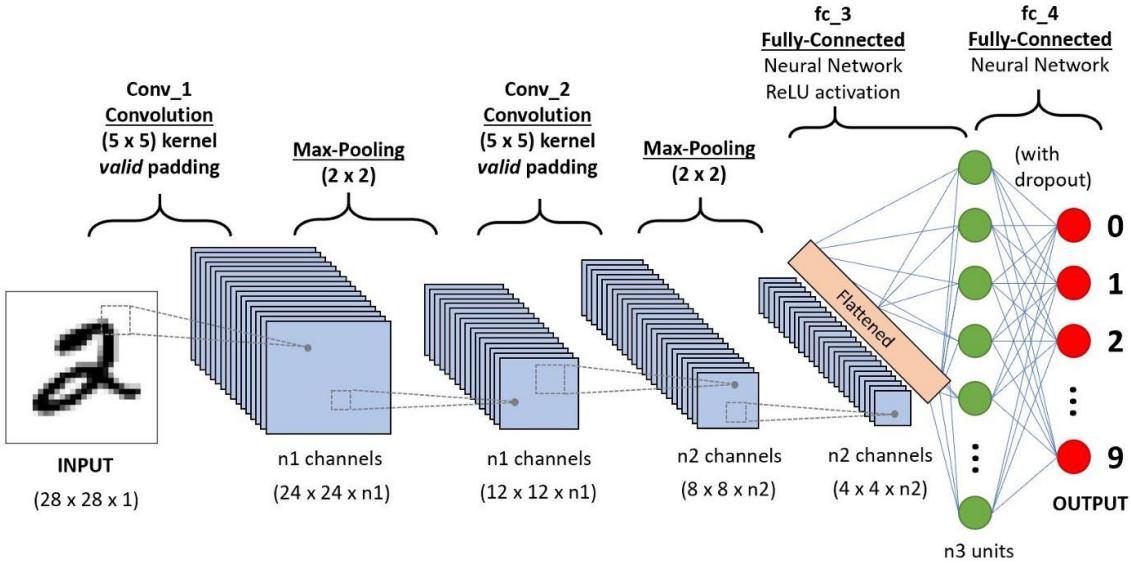
Αν όμως έχουμε ένα πολύ πολύπλοκο πρόβλημα, όπως για παράδειγμα η αναγνώριση φωνής, τότε μπορούμε να χρησιμοποιήσουμε και μια πιο σύνθετη μορφή νευρωνικών δικτύων, τα δικτυα ανατροφοδότησης (**recurrent neural network**). Η κυρια διαφορα τους με τα feedforward είναι ότι τα δεδομένα μπορούν να έχουν μια αμφίδρομη ροή. Στην πραγματικότητα, τα δίκτυα αυτά έχουν την ικανότητα να διατηρούν μια κρυφή κατάσταση η οποία αποθηκεύει πληροφορίες για τις προηγούμενες εισόδους.[23] Αυτο βοηθάει το δίκτυο να μαθαίνει και να αναγνωρίζει πιο εύκολα συσχετίσεις αυξάνει όμως κατα πολύ την πολυπλοκότητα του.

Μια άλλη μορφή feedforward network που χρησιμοποιείται κυρίως για εικόνες αλλά και ότι αφορά προβλήματα με οπτικά δεδομένα είναι τα συνελικτικά νευρωνικά δίκτυα **(Convolutional Neural Networks ή CNN)**.[24]

1.11 Συνελικτικά Δίκτυα (Convolutional Neural Networks - CNN)

Τα συνελικτικά νευρωνικά δίκτυα είναι μια υποκλάση των τεχνητών νευρωνικών δικτυων και συνήθως χρησιμοποιείται για την ανάλυση εικόνων και βίντεο. [24]

Τα συνελικτικά δίκτυα προσπαθούν να μιμηθούν τον ανθρώπινο εγκέφαλο και πιο συγκεκριμένα το πώς ο ανθρώπινος εγκέφαλος αναγνωρίζει και επεξεργάζεται οπτικές πληροφορίες. Στον ανθρώπινο εγκέφαλο, οι νευρώνες ανταποκρίνονται σε συγκεκριμένες περιοχές του οπτικού πεδίου, και οι απαντήσεις τους συνδυάζονται για να αναγνωρίσουν πολύπλοκα μοτίβα και αντικείμενα. [25]



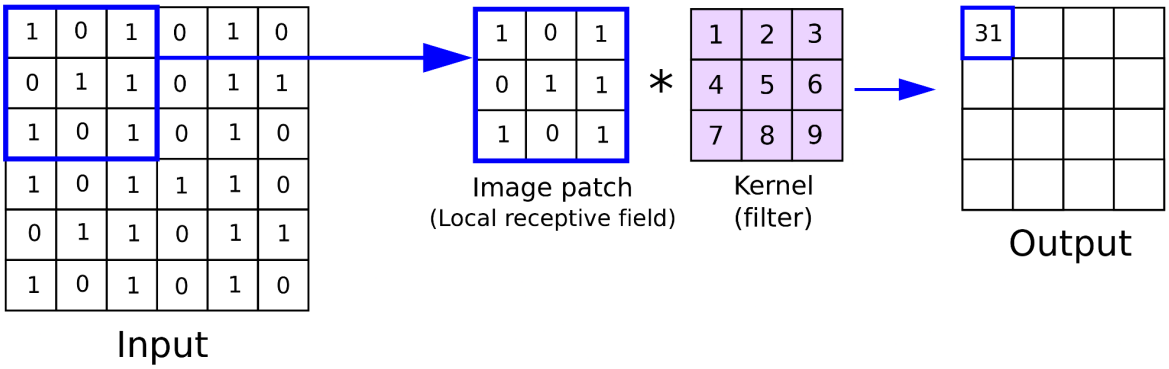
Εικ. 1.6 Παράδειγμα Συνελικτικού Δικτύου

Τα CNN εκμεταλλεύονται καλύτερα την διάταξη των πιξελ, αφού τα κοντινά πιξελ εχουν μεγαλυτερη σχέση απ' ότι τα μακρινά. Αυτο το πετυχαίνουν εφαρμόζοντας τοπικά "φίλτρα" στην εικόνα, χωριζοντας την ετσι σε μικρότερα μέρη και αναλύοντας ένα πολύ μικρό και εύκολο κομμάτι της. Σε αντίθεση με τα κλασικά μοντέλα μηχανικής μάθησης, η διαδικασία βελτιστοποίησης των φίλτρων εξαγωγής των χαρακτηριστικών γίνεται αυτοματοποιημένα και αποτελεί ένα από τα μεγαλύτερα πλεονεκτήματα των συνελικτικών δικτύων.[27]

Σε ότι αφορά τα δεδομένα εισόδου σε ένα τέτοιο νευρωνικό, η μόνη ιδιαιτερότητα τους βρίσκεται στις διαστάσεις της εικόνας που χρησιμοποιείται ως είσοδος που είναι τέσσερις και συνήθως είναι (μέγεθος batch, ύψος εικόνας, πλάτος εικόνας, κανάλιας εικόνας). Αυτό που μπορεί να αλλάξει είναι τα κανάλια εικόνας. Για μία εικόνα στην κλίμακα του γκρι (grayscale) το κανάλι θα είναι 1 ενω σε μία πολύχρωμη εικόνα RBG τα κανάλια θα είναι 3.

1.11.1 Συνελικτικά Στρώματα (Convolutional layers)

Ένα συνελικτικό στρώμα είναι ένα υπολογιστικό στρώμα που πραγματοποιεί την πράξη την συνέλιξης στην είσοδο του και στέλνει το αποτέλεσμα στο επόμενο στρώμα. Στόχος τους είναι η εξαγωγή των πιο σημαντικών πληροφοριών και χαρακτηριστικών από εικόνες ή βίντεο.



Εικ. 1.7 Πράξη συνέλιξης

Η εικόνα που θα εισέλθει ως είσοδος σε ένα τέτοιο στρώμα μετά την πράξη της συνέλιξης θα μετατραπεί σε ένα χάρτη χαρακτηριστικών (**feature map**). Στην συνέχεια αυτο θα περαστεί με την σειρά του σαν εισοδος στο επόμενο στρώμα του. Κάθε νευρώνας επεξεργάζεται την είσοδο μόνο του κομματιού που θα χωριστεί από το φίλτρο (**receptive field**).

Ειδικότερα, κάθε ένα συνελικτικό στρώμα δέχεται κάποια δεδομένα, συνήθως εικόνα ως είσοδο (**input**). Στην συνέχεια, ενεργούν τα φίλτρα (**kernels ή feature detectors)**. Τα φίλτρα συνήθως έχουν διαστάσεις 3x3 ή 5x5 έτσι ώστε να απομονώνουν και να αναλύουν μια μικρότερη περιοχή των δεδομένων, δημιουργώντας έτσι το πεδίο λήψης (**reception field**). Επίσης, περιέχουν βάρη που αλλάζουν και ρυθμίζονται με απώτερο σκοπό την εκμάθηση στην αναγνώριση συγκεκριμένων γεωμετρικών σχημάτων και χαρακτηριστικών. Όταν ενεργούν τα φίλτρα στην πραγματικότητα πραγματοποιείται η πράξη της συνέλιξης ανάμεσα στα δεδομένα εισόδου νευρώνα και στο φίλτρο. Η συνέλιξη είναι το πέρασμα του φίλτρου πάνω από τα δεδομένα και τον υπολογισμό του πολλαπλασιασμού στοιχείου προς στοιχείο μεταξυ των βαρών και των δεδομένων. Τα αποτελέσματα αυτα προστίθενται έτσι ώστε να δημιουργήσουν μια μόνο τιμή σαν έξοδο και στο τέλος να δημιουργηθεί το feature map (**output**). (Εικ. 1.7) [29]

Ένα επίπεδο δεν είναι απαραίτητο να έχει μόνο ένα φίλτρο, μάλιστα το πιο σύνηθες είναι να έχει πολλαπλά έτσι ώστε το feature map να είναι όσο πιο περιεκτικό γίνεται με τα χαρακτηριστικά της εικόνας μειώνοντας όμως τον θόρυβο.
Όλη η επιτυχία των συνελικτιλών δικτύων βασίζεται στο ότι σπάμε μια εικόνα σε πολλα μικροτερα γεωμετρικά σχήματα στα επιμέρους στρώματα και περνώντας διαδοχικά τις εικόνες απο πολλα τετοια, το δικτυο καταφέρνει να αναγνωρίσει πολυπλοκες μορφές όπως για παράδειγμα αναγνώριση συναισθημάτων από ένα πρόσωπο, με πολύ μεγαλύτερη ακρίβεια.[30]

1.11.2 Στρώματα Συγκέντρωσης (Pooling Layers)

Τα συνελικτικά δίκτυα περιλαμβάνουν και κάποια στρώματα συγκέντρωσης, τα οποία συνήθως τοποθετούνται αμέσως μετά από ένα συνελικτικό στρώμα. Στόχος τους είναι το **downsampling** των δεδομένων.[31] Το downsampling είναι τεχνική κατα την οποία μειώνουμε τις διαστάσεις των δεδομένων που θα εισαχθούν κρατώντας όμως όσο περισσότερα στοιχεία της εικόνας θα βοηθήσουν στην επίτευξη του στόχου. [32]

Συνήθως τα στρώματα συγκέντρωσης είναι δύο ειδών:

- Στρώμα Μέσης Συγκέντρωσης (**Average Pooling Layer**): Υπολογίζει και χρησιμοποιεί την μέση τιμή για κάθε ομάδα νευρώνων,
- Στρώμα Μέγιστης Συγκέντρωσης (**Maximum Pooling** ή **Max Pooling Layer**): Χρησιμοποιεί την μέγιστη τιμή. Προφανώς μιλάμε για τοπικό μέγιστο και όχι ολικο αφού η περιοχή μας είναι περιορισμένη στο feature map.[33]

1.12 Συναρτήσεις Ενεργοποίησης (Activation Functions)

Οι συναρτήσεις ενεργοποίησης είναι συναρτήσεις που επεξεργάζονται την έξοδο ενός νευρώνα, για να μπορέσει να αναπαραστήσει πιο πολύπλοκα προβλήματα. Αν δεν υπάρχουν τα activation functions σε κάθε στρώμα, τότε το μοντέλο θα ήταν απλά ένα μοντέλο γραμμικής παλινδρόμησης. Ανάλογα με το ποια function θα δράσουν στον νευρώνα, τότε θα αλλάξουν οι νευρωνες που θα ενεργοποιηθούν. Συνήθως οι συναρτήσεις ενεργοποίησης είναι μη γραμμικές. [34]

1.12.1 Βηματική Συνάρτηση

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Μαθηματικός τύπος 1.4

Η βηματική συνάρτηση (**Heaviside Step Function**) είναι μία μαθηματική συνάρτηση που δέχεται δεδομένα και παράγει ένα δυαδικό αποτέλεσμα. Ο νευρώνας ενεργοποιείται όταν το x έχει θετική τιμή αλλιώς μένει ανενεργός. [35]

1.12.2 Σιγμοειδής Λογιστική Συνάρτηση (Sigmoid Function)

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

Μαθηματικός τύπος 1.5

Η σιγμοειδης συνάρτηση είναι η πιό κατάλληλη συνάρτηση για το τελευταίο στρώμα σε προβλήματα δυαδικής ταξινόμησης (binary classification). Πολλές φορές αναφέρεται και ως logistic function.

1.12.3 Rectified Linear Unit - ReLU Function

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Μαθηματικός τύπος 1.6

Η ReLU, επίσης γνωστή και ώς συνάρτηση ράμπας, είναι μία από τις καλύτερες επιλογές για να χρησιμοποιηθεί ανάμεσα στα hidden layers του μοντέλου.

1.12.4 Softmax Function

$$f\left(x_i\right) = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}}$$

Μαθηματικός τύπος 1.7

Η softmax χρησιμοποιείται για την μετατροπή ενός διανύσματος με *k* πραγματικούς αριθμούς σε μια κατανομή πιθανοτήτων με *k* πιθανά αποτελέσματα. Συνήθως χρησιμοποιείται στο τελευταίο στρώμα όταν θέλουμε κατανομή πιθανοτήτων για το πρόβλημά μας και όχι κάποια συγκεκριμένη κλάση.

1.13 Υπολογισμός Σφαλμάτων - Συναρτήσεις κόστους

Για να γνωρίζουμε πόσο επιτυχές είναι το μοντέλο που έχουμε επιλέξει, πρέπει να ορίσουμε και μία μετρική έτσι ώστε να βλέπουμε πόσο μεγάλο ποσοστό λάθους (ή απλά σφάλμα) έχουμε. Αυτή η τιμή προκύπτει από μία συνάρτηση που λέγεται συνάρτηση κόστους ή συνάρτηση σφάλματος (**loss function or cost function)**. Αυτή ορίζεται ανάλογα με το πρόβλημα. [37]
Οι συναρτήσεις σφάλματος είναι ένα από τα πιο σημαντικά κομμάτια ενός μοντέλου μηχανικής μάθησης καθώς μέσω αυτών καταλαβαίνουμε πόσο αποδοτικό είναι ένα μοντέλο στα αποτελέσματα του σε σχέση με τα πραγματικά δεδομένα. Αυτό είναι πολύ σημαντικό στην μάθηση με επίβλεψη καθώς η εκπαίδευση του μοντέλου βασίζεται στην μείωση των σφαλμάτων. Τα σφάλματα για το οποία μιλάμε είναι η διαφορά μεταξύ των προβλέψεων του μοντέλου (**predictions**) και των πραγματικών τιμών των στόχων (**targets**). [38]

Παρακάτω θα αναλύσουμε μερικές από τις συναρτήσεις κόστους.

1.13.1 Mean Square Error (MSE)

Η συνάρτηση MSE είναι η πιο διαδεδομένη συνάρτηση που χρησιμοποιείται σε προβλήματα παλινδρόμησης, όπου τα αποτελέσματα είναι μια συνεχής μεταβλητή στο χώρο. Βρίσκει το σφάλμα υπολογίζοντας τον μέσο όρο των τετραγώνων των διαφορών μεταξύ των προβλέψεων και των πραγματικών τιμών. Σχηματικά, προσπαθεί να βρεί την καλύτερη καμπύλη που ελαχιστοποιεί την απόσταση μεταξύ των δύο τιμών ετσι ώστε να έχει ακόμα καλύτερα αποτελέσματα σε άγνωστα δεδομένα.

Ο τυπος της είναι:

$$MSE = \left(\frac{1}{n}\right) * \sum (yi - \bar{y})^2$$

Μαθηματικός τύπος 1.8

όπου:

- n είναι ο αριθμός των δειγμάτων στο dataset,
- yi είναι η προβλεπόμενη τιμή για το i-οστό δείγμα,

- ȳ είναι η πραγματική τιμή (target) για το i-οστό δείγμα.[39]

1.13.2 Mean Absolute Error (MAE)

Η συνάρτηση MAE είναι παρόμοια με την MSE και χρησιμοποιείται και αυτή σε προβλήματα παλινδρόμησης. Η κύρια διαφορά της με την MSE είναι ότι αντί να υπολογίζει μέσο όρο των διαφορών των τετραγώνων, η MAE υπολογίζει το μέσο όρο των απόλυτων τιμών των διαφορών. [40]

Ο τυπος της είναι:

$$MAE = \frac{\sum_{i=1}^{n} |yi - xi|}{n}$$

Μαθηματικός τύπος 1.9

1.13.3 Cross-Entropy Loss (Log Loss)

Η συνάρτηση Cross-Entropy Loss ή αλλιως Softmax loss χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης, όπου υπάρχουν περισσότερες από δύο κλάσεις σαν πιθανό αποτέλεσμα. Αυτή βρίσκει το σφάλμα μετρώντας την διαφορά μεταξύ των προβλεπόμενων τιμών και τις πραγματικές πιθανότητες των κλάσεων εκφρασμένες σαν one-hot encoding.

Ο τύπος της είναι:

$$Cross - Entropy = -\sum(yi * log(pi))$$

Μαθηματικός τύπος 1.10

όπου:

- Σ είναι το άθροισμα των κλάσεων,
- yi είναι η one hot encoded πραγματική πιθανότητα,
- pi είναι η προβλεπόμενη πιθανότητα [41]

1.13.3.1 One-hot encoding

Το one-hot encoding είναι μια τεχνική που χρησιμοποιούμε για την αναπαράσταση δεδομένων στα προβλήματα ταξινόμησης. Κάθε έξοδος-κατηγορία σε ένα τέτοιο πρόβλημα αναπαριστάται από ένα δυαδικό διάνυσμα οπου έχει ένα στην τιμή που είναι η κατηγορία και 0 σε όλες τις άλλες θέσεις. [42]

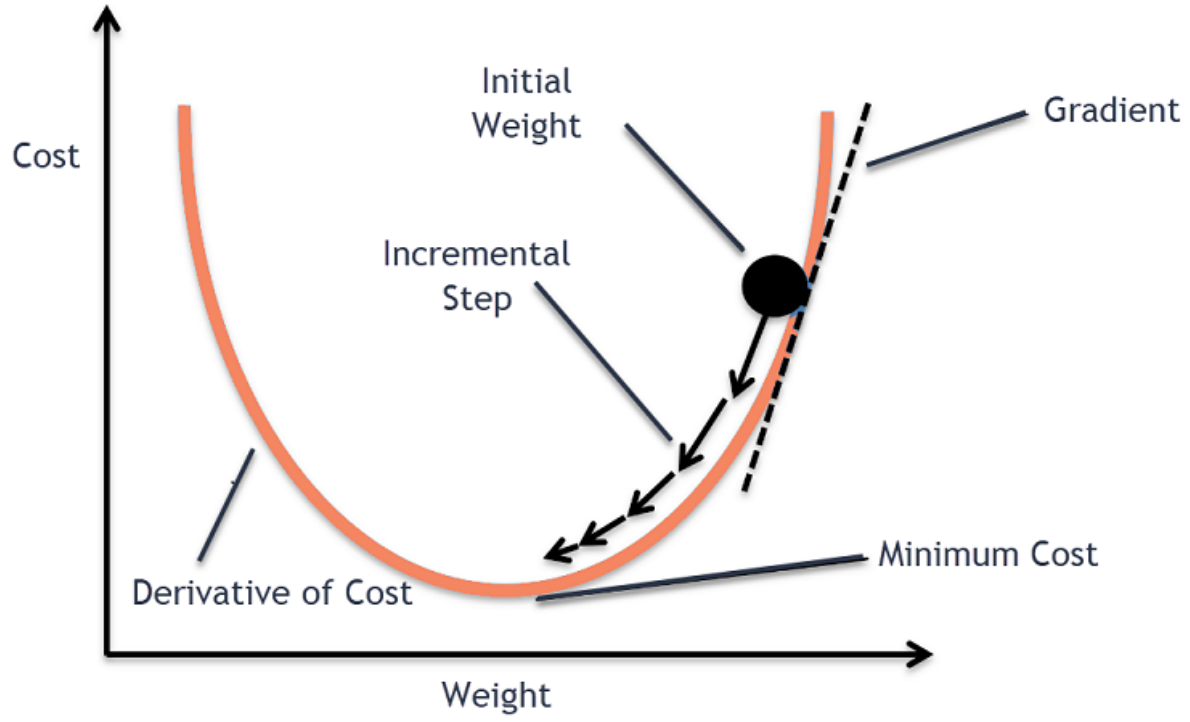
Για παράδειγμα, έστω ότι έχουμε να ταξινομήσουμε μια εικόνα ανάλογα με το ποιό φρούτο εμφανίζεται. Η κατηγορίες θα ήταν τρεις: μήλο, αχλάδι, βερίκοκο. Η παραπάνω κλάσεις θα αναπαρασταθούν κάπως έτσι:

- Μήλο => [1,0,0]
- Αχλάδι => [0,1,0]
- Βερίκοκο => [0,0,1]

1.14 Αλγόριθμοι Βελτιστοποίησης

Ο αλγόριθμος βελτιστοποίησης (**optimization algorithm**) είναι ένας αλγόριθμος που στόχος του είναι η βελτιστοποίηση της αποτελεσματικότητας του μοντέλου. Αυτο το πετυχαίνει προσπαθόντας να μειώσει όσο περισσότερο γίνεται το αποτέλεσμα του loss function που έχουμε επιλέξει, να μειώσει δηλαδή την απόκλιση μεταξύ τον πραγματικών τιμών και των προβλεπόμενων.

Ο αλγόριθμος βελτιστοποιησης αλλάζει τις παραμέτρους του μοντέλου βάση της κλίσης που έχει η συνάρτηση απώλειας. Σε κάθε επανάληψη, ο αλγόριθμος υπολογίζει την κλίση της συνάρτησης απώλειας, και στην συνέχεια αλλάζει τα βάρη και την πόλωση των νευρώνων κατα την αντίθετη κατεύθυνση της κλίσης. Αυτό γίνεται γιατί η κλίση είναι ένα διάνυσμα που "δείχνει" προς την πιο απότομη αύξηση της και άρα πρέπει να κατευθυνθούμε αντίθετα απο αυτην γιατι θεωρητικά εκεί βρίσκεται η μεγαλύτερη κατηφόρα έτσι ώστε η ο αλγόριθμος να συγκλίνει σταδιακά στο τοπικό ή ολικό μέγιστο που είναι ένα σύνολο τιμών που μειώνει το αποτέλεσμα της συνάρτησης απώλειας και αρα αυξάνει την αποδοτικότητα του μοντέλου. Η παραπάνω διαδικασία ονομάζεται και κατάβαση δυναμικού (**Gradient Descent**) . Αν δεν τα αλλάζαμε τις παραμέτρους προς την αντίθετη φορα, αλλα προς την κανονική τότε θα πηγαίναμε προς το τοπικό μέγιστο ή ολικό, θα ανεβαίναμε την ανηφόρα και θα μιλούσαμε και ανάβαση δυναμικού (**Gradient Ascend**). Προφανώς για να χρησιμοποιήσουμε gradient descent προϋποθέτει να υπάρχει η κλίση και για να υπάρχει η κλίση, πρέπει η συνάρτηση του κόστους να είναι διαφορίσιμη. [43]



Εικ.1.8 Γραφική αναπαράσταση Κατάβασης Δυναμικού

Το gradient descent θα μπορούσαμε να το χωρίσουμε σε τρεις κατηγορίες:

- **Batch Gradient Descent**: Σε αυτόν τον αλγόριθμο, πρώτα υπολογίζουμε την κλίση για ολόκληρο το πλήθος των δεδομένων και μετά ενημερώνουμε τις παραμέτρους με βάση την κλίση όλου του set. Παρόλο που με σχετική βεβαιότητα θα φτάσουμε στο ολικό μέγιστο, η χρήση του σε μεγάλα dataset απαιτεί πάρα πολλούς υπολογιστικούς πόρους και η εκπαίδευση

του μοντέλου θα διαρκέσει αρκετή ώρα. Συνεπώς, η χρήση του ενδύκνειται για μικρότερα dataset.

- **Stochastic Gradient Descent**:Σε αντίθεση με το batch εδω η ενημέρωση των παραμέτρων γίνεται μετά από κάθε δείγμα. Είναι πιο γρήγορος αλγόριθμος στο να φτάσει σε ένα αποτέλεσμα και με λιγότερους υπολογιστικούς πόρους αλλά λόγω του ότι κάθε δείγμα επηρεάζει τις μεταβλητές είναι πιο εύκολο να "φτάσει" σε ένα τοπικό ελάχιστο απ' ότι σε ένα ολικό ελάχιστο.
- **Mini-Batch Gradient Descent**: Είναι η μέση λύση των δύο παραπάνω κατηγοριων, δηλαδή χωρίζει τα δεδομένα σε μικρότερα κομμάτια και ενημερώνει τις παραμέτρους μετά απο κάθε κομμάτι. Αυτή η μέθοδος καθώς συνδυάζει τα θετικά των δύο παραπάνω μεθόδων, είναι δηλαδή πολύ πιο γρήγορη και απαιτεί λιγότερου πόρους και μπορεί πιο εύκολα να φτάσει στο ολικό μέγιστο χωρίς να κολλήσει κάπου. [44]

1.14.1 Adam (Adaptive Moment Estimation)

Ένας από τους πιό διαδεδομένους και πολυ χρησιμοποιημένους αλγορίθμους βελτιστοποίησης είναι ο **Adam** (**Adaptive Moment Estimation**). Ο Adam αποτελεί επέκταση του stochastic gradient descent, με πολύ πιο εύκολη υλοποίηση, συνδυάζοντας χαρακτηριστικά από άλλες υλοποιήσεις του stochastic gradient descent ώστε να μπορέσει να γίνει πιό αποδοτικός.

Τα χαρακτηριστικά που συνδυάζει είναι:

- **Adaptive Gradient Descent (AdaGrad)**: Είναι τεχνική που χρησιμοποιείται όταν έχουμε sparse κλίσης, όταν δηλαδή η παράγωγος μηδενίζεται σε πολλά σημεία, και αλλάζει τον ρυθμο μάθησης σε επίπεδο παραμέτρων. Κάνει μεγάλες ενημερώσεις στις τιμές των παραμέτρων που σχετίζονται με πιό σπάνια χαρακτηριστικά και μικρές σε αυτά που σχετίζονται με πιο συνηθισμένα καθώς αυτά τα χαρακτηριστικα θα εκπαιδευονται πιο συχνα και θα κάνουν περισσότερα βήματα. Μαθηματικά, αυτο το καταφέρνει αποθηκεύοντας και αξιοποιώντας τα αποτελεσματα των κλίσεων υψωμένα στο τετράγωνο για κάθε παράμετρο. Αυτο σημαίνει ότι το learning rate της κάθε παραμέτρου μειώνεται κατα την τετραγωνική ρίζα των τετραγωνισμένων κλίσεων. Το μεγαλύτερο πρόβλημα είναι ότι με αυτήν την μέθοδο μπορεί πολλά learning rate να μειωθούν πάρα πολύ με αποτέλεσμα να μην είναι βέλτιστα.
- **Root Mean Square Propagation (RMSProp)**: Πρόκειται για μία επέκταση του AdaGrad. Εκτος απο το να αλλάζει ρυθμό μάθησης ανάλογα με το τι χαρακτηριστικά επηρεάζουν οι παράμετροι, χρησιμοποιεί έναν μέσο όρο μερικών κλίσεων για να αλλάξει τον ρυθμό μάθησης. Αυτό επιτρέπει στον αλγόριθμο να έχει ένα είδος μνήμης που "ξεχνάει" τις πολύ παλιές κλίσεις και να δώσει περισσότερη βαρύτητα στις πιό καινουργιες. Με αυτόν τον τρόπο καταφέρνει να σταθεροποιήσει το learning rate και του δίνει την δυνατότητα να αλλάξει με βάση τις παραμέτρους αλλα ταυτόχρονα το αποτρέπει από το να πάρει πολύ μικρές τιμές, φτάνοντας έτσι το μοντέλο σε μια όχι βέλτιστη λύση.[45]

Ο Adam καταφέρνει να συνδυάζει τα θετικά των δύο παραπάνω υλοποιήσεων. Αυτό τον βοηθάει να συγκλίνει πολύ πιο γρήγορα και με μεγαλύτερη επιτυχία απο το άλλες μεθόδους. Προσαρμόζει τον ρυθμό απόδοσης της κάθε μεταβλητής με βάση τους μέσους όρους της κλίσης και της τετραγωνικής κλίσης των μεταβλητών. Επειδή έχει λιγότερες παραμέτρους που πρέπει να βρεθούν χειροκίνητα καθώς και οτι καταφέρνει να επιλύσει βέλτσιστα πολλά παραδείγματα, είναι ιδιαίτερα δημοφιλής και χρησιμοποιείται ευρέως σε πολλά προβλήματα σε διάφορους τομείς.

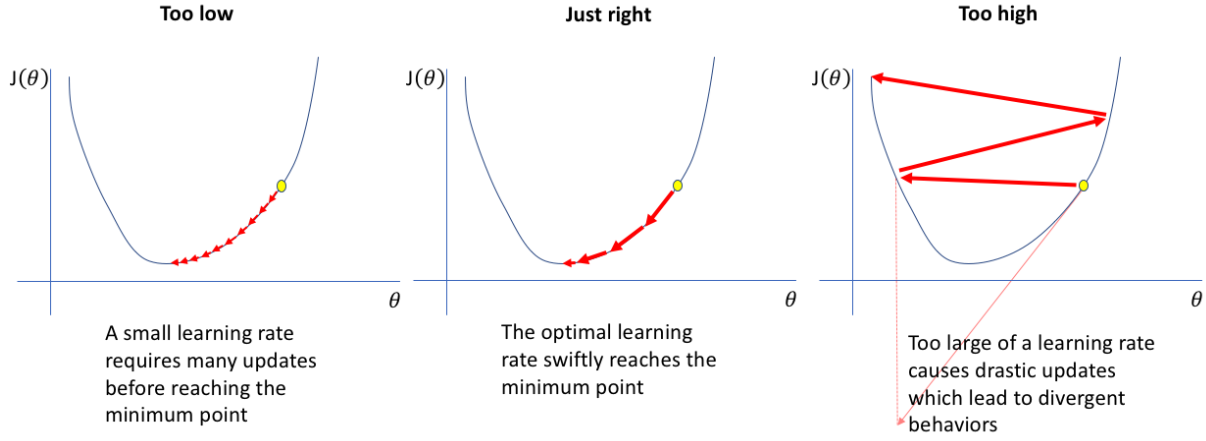
Ο Adam δουλεύει υπολογίζοντας δύο μετρικές, την πρώτη ροπή που είναι ο μέσος όρος της κλίσης και την δεύτερη ροπή που είναι η διακύμανση της κλίσης. Αυτοί οι δύο μέσοι όροι χρησιμοποιούνται για την ενημέρωση των παραμέτρων και μαζί με την διόρθωση του bias πραγματοποιουν την εκπαίδευση του μοντέλου.[46]

Αν πάμε ένα βήμα πίσω όμως, για να χρησιμοποιήσουμε κατάβαση δυναμικού για την ενημέρωση των παραμέτρων χρειαζόμαστε δύο πράγματα: μια κατεύθυνση και έναν ρυθμό μάθησης.

1.15 Ρυθμός μάθησης

Ο ρυθμός μάθησης (**Learning Rate**) είναι μία από τις πιό βασικές υπερπαραμέτρους στα μοντέλα μηχανικής μάθησης. Είναι η μονάδα που θα αλλάξει ο αλγόριθμος βελτιστοποίησης και καθορίζει το ρυθμό μεταβολής των παραμέτρων στην προσπάθεια τους να ελαχιστοποιηθεί η συνάρτηση απώλειας. Με λίγα λόγια, εάν η κλίση ελέγχει την κατεύθυνση του βήματος, τότε ο ρυθμός μάθησης θα προσδιορίσει το πόσο μεγάλο θα είναι το βήμα.

Όπως όμως και σε όλες τι παραμέτρους στην μηχανική μάθηση, πρέπει να προσέχουμε και να βρούμε τον σωστό αριθμό γιατί ένας πολύ μεγάλος ρυθμός μάθησης μπορεί να προκαλέσει μεγάλη απόκλιση στην βελτιστοποίηση και ουσιαστικά προσπερνάει το ελάχιστο την συνάρτησης απώλειας αλλα και ένας πολύ μικρός ρυθμός μάθησης μπορεί προκαλέσει πολύ μικρή σύγκληση ή ακόμα και να "κολλήσει" σε ένα τοπικό ελάχιστο με αποτέλεσμα να μην βρεθεί το ολικό ελάχιστο, δίνοντας όχι την βέλτιστη λύση αλλά μια χειρότερη.[47]



Εικ. 1.9 Αναπαράσταση περιπτώσεων Learning rate

Συνήθως δεν υπάρχει κάποιος χρυσός κανόνας για την επιλογή του learning rate, απλά κάνουμε πειράματα μέχρι να βρούμε το καλύτερο. Πολλές φορές είναι δύσκολο να βρεθεί το learning rate γιατι προφανώς στην αρχή θα θέλαμε να είναι μεγάλο για να μαθαίνει πιο γρήγορα κάνοντας μεγαλύτερα "βήματα" προς την σωστή κατευθυνση αλλα πιο μετά θα θέλαμε να κάνει πιο μικρά για να μπορέσει να "βρεί" το ολικό ελάχιστο. Αυτο ακριβώς υλοποιήθηκε με μια τεχνική που ονομάζεται προγραμματισμός ρυθμου μάθησης (**learning rate scheduler**). Ένας learning rate scheduler αλλάζει τον ρυθμό μάθησης κατά την διάρκεια των εποχών συνήθως ανάλογα με τον χρόνο ή τα βήματα που έχουν γίνει. Αυτο το πετυχαίνει αλλάζοντας δύο παραμέτρους του, το decay και το momentum. Το decay είναι ο ρυθμός που θα αλλάζει το learning rate και το momentum που είναι υπεύθυνο για την επιτάχυνση της μάθησης ενώ ταυτόχρονα αποφεύγει το να κολλήσει σε τοπικά ελάχιστα. Αυτο το πετυχαίνει κρατώντας ένα ιστορικό του loss.[48]

1.16 Βαθιά Μάθηση (Deep Learning)

Τα τελευταία χρόνια γίνονται πολλές αναφορές στον όρο βαθιά μάθηση καθώς και όλα τα μοντέλα που αποτελούν την κορυφή των εκάστοτε υποκλάδων της μηχανικής μάθησης αυτοαποκαλούνται βαθιά μοντέλα. Τι είναι όμως το ένα μοντέλο Deep Learning?
Δεν είναι τίποτα παραπάνω από ένα νευρωνικό δίκτυο με τρία ή περισσότερα κρυφά στρώματα. Το ανθρώπινο νευρικό σύστημα είναι πολύπλοκο και για να προσομοιωθεί κατάλληλα θα χρειαζόταν πάρα πολλά στρώματα. Παρόλο που ένα απλό MLP με 3 στρώματα μπορεί να παράξει αποτελέσματα προσθέτοντας και άλλα στρώματα στο κρυφο επιπεδο θα μπορουσε να επιτευχθεί ακόμα πιο υψηλή ακρίβεια στο επιθυμητό αποτέλεσμα του μοντέλου. [49]

Επίσης τα μοντέλα βαθιάς μηχανικής μάθησης μπορούν πολύ πιο εύκολα να επεξεργαστούν και να αναγνωρίσουν συσχετίσεις ανάμεσα σε δεδομένα το οποία δεν έχουν ξεκάθαρη συσχέτιση ή δεν είναι στην κλασική δομή των δεδομένων, μαζί με τους στόχους, όπως για παράδειγμα εικόνες και βίντεο. Αυτο το καταφέρνουν επειδή λόγω των πολλών νευρώνων και στρωμάτων καταφέρνουν να δημιουργήσουν μια ιεραρχία χαρακτηριστικών και αφαιρούν αυτα που δεν κρίνουν αρκετά σημαντικά για τον στόχο. Για παράδειγμα, αν θέλουμε να κατηγοριοποιήσουμε φωτογραφιες γατων και σκυλων, το ότι έχουν και τα δύο ζώα τέσσερα πόδια η τρίχωμα είναι πληροφορίες που δεν βοηθούν το νευρωνικό και δεν θα τις λάβει υπόψη από μόνο του. Αντίθετα το οτι ο σκυλος εχει πλατια αυτια και η γάτα μουστάκια είναι πληροφορίες πάνω στις οποίες θα εκπαιδευτεί. Σε ένα απλό MLP για παράδειγμα θα έπρεπε αυτά τα features να προεπεξεργαστούν απο πρίν και να αφαιρεθούν. [50][51]

1.16.1 Κανονικοποίηση Βαθμίδας - Batch Normalization

Η κανονικοποίηση βαθμίδας είναι μια τεχνική που εφαρμόζεται κυρίως στα προβλήματα μηχανικής μάθησης και στόχος της είναι να κάνει την εκπαίδευση των νευρωνικών δικτύων πιο γρηγορη και πιο σταθερή χωρίς να απαιτεί μείωση των κρυφών στρωμάτων. Αυτό επιτυγχάνεται με την επανακεντρικοποίηση και επανακλιμάκωση των δεδομένων στην αρχή κάθε στρώματος. Προσπαθεί δηλαδή να τροποποιήσει τα δεδομένα εισόδου ετσι ώστε να έχουν μέση τιμή μηδέν και τυπική απόκλιση ένα. Συνήθως προσπαθεί να εντάξει όλες τις τιμές στο ίδιο εύρος τιμών έτσι ώστε το μοντέλο μαθαινει πιο ευκολα. Επίσης η κανονικοποίηση βοηθάει στο να μην έχουμε underfitting σε ένα μοντέλο. [52]

Κεφάλαιο 2ο: Κατηγοριοποίηση Εικόνων

Το πρώτο κομμάτι της πτυχιακής μου αφορά τα μοντέλα κατηγοριοποίησης. Πρίν όμως φτάσω σε αυτό το κομμάτι έπρεπε να επιλέξω ένα dataset. Αποφάσισα να χρησιμοποιήσω ένα dataset που χρησιμοποιείται σε binary image classification, το cat-vs-dog dataset [54], το οποίο είχε επαρκή αριθμό εικόνων-δειγμάτων για κάθε κλάση (περιπου δωδεκαμιση χιλιαδες για καθε μια) με διαφορετικές δυσκολίες. Όσον αφορά το classification, απο το να δημιουργηθει κατι εξ ολοκλήρου καινούργιο αποφάσισα να χρησιμοποιήσω μάθηση με μεταφορα στο μοντέλο ResNet50. Ας πάρουμε όμως τα πράγματα από την αρχή.

2.1 Προεπεξεργασία Δεδομένων (Data Preprocessing)

Η προεπεξεργασία δεδομένων ή αλλιως data preprocessing είναι ένα από τα πιό σημαντικά βήματα στην μηχανική μάθηση καθώς μέσω αυτου διασφαλίζουμε και την επιτυχία του μοντέλου. Τα δεδομένα που θα παρθούν απο τον πραγματικό κόσμο μπορεί να έχουν θορυβους, ελλειψεις, διπλοεγγραφές και πολλά άλλα χαρακτηριστικά που θα δυσκολέψουν το μοντέλο να ανακαλύψει τις συσχετίσεις ανάμεσα στα δείγματα. Είναι πολύ σημαντικό συνεπώς να γίνει μια προεργασια στα δεδομένα, η οποία αφενός θα ελέγχει για πιθανά προβλήματα-ελλείψεις, αφετέρου θα αυξάνει ή θα μειώνει τα δεδομένα.
Με την σειρα του το data preprocessing μπορεί να χωριστεί σε τέσσερις επιμέρους υποκατηγορίες:

- Συνδυασμός Δεδομένων (**Data Integration**)
- Καθαρισμός Δεδομένων (**Data Cleaning**)
- Μετασχηματισμός Δεδομένων (**Data Transformation**)
- Μείωση Δεδομένων (**Data Reduction**) [55]

2.1.1 Συνδυασμός Δεδομένων (Data Integration)

Ο συνδυασμός δεδομένων είναι η διαδικασία κατα την οποία δεδομένα από πολλές πηγές ενώνονται σε ένα ενιαίο dataset. Τα δεδομένα που θα συλλεχθούν για την δημιουργία του dataset μπορεί να είναι αποθηκευμένα σε διαφορετικές μορφές, για να εισαχθούν όμως όλα στο μοντέλο πρέπει να βρίσκονται όλα στην ίδια μορφή, όπως για παράδειγμα να βρίσκονται όλα στην ίδια βάση δεδομένων. [56]

2.1.2 Καθαρισμός Δεδομένων (Data Cleaning)

Ο καθαρισμός δεδομένων είναι ένα πολύ σημαντικό στάδιο στην προεπεξεργασία δεδομένων και αφορά τις τιμές που λείπουν, τα θορυβώδη δεδομένα, και τις ακραίες τιμές. Οι ενέργειες που ακολουθούν είναι οι εξής:

- **Συμπλήρωση των τιμών που λείπουν**: Αν υπάρχουν στο dataset τιμες που λείπουν, τότε είτε αγνοούμε το δείγμα εξ' ολοκλήρου, με την προϋπόθεση ότι έχουμε ένα μεγάλο αριθμό δειγμάτων, είτε προσπαθούμε να συμπληρώσουμε τα κενά, συνήθως μέσω του μέσο όρου ή της διαμέσου.
- **Θορυβώδης δεδομένα**: Με τον όρο θορυβώδης δεδομένα (**noisy data**) αναφερόμαστε κυρίως σε δεδομένα που που οι τιμές τους είναι τυχαίες, είτε σε δεδομένα τα οποία έγινε κάποιο λάθος κατα την μέτρηση τους. Με λίγα λόγια δεδομένα που δεν λείπουν αλλά είναι προβληματικά και θα παράξουν αξιόπιστα αποτελέσματα. Συνήθως για να "καταπολεμήσουμε" αυτό το πρόβλημα, δημιουργούμε συστάδες (**clustering**).

    - **Clustering**: Η συσταδοποίηση είναι μια τεχνική κατα την οποία δεδομένα με παρόμοια χαρακτηριστικά ομαδοποιούνται σε συστάδες-ομάδες. Όσο πιο κοντά βρίσκονται τα δεδομένα μέσα σε ένα cluster τόσο πιο πολύ μοιάζουν. Είναι μια τεχνική που κυρίως χρησιμοποιείται στην μη επιβλεπόμενη μάθηση, γιατι δεν έχουμε targets οπότε κάπως πρέπει να βρεθεί το label που θα μπορούσε να έχει.
- **Αφαίρεση ακραίων στοιχείων**: Για να μελετηθει κατα πόσο υπάρχουν ακραία σημεία (**outliers**) πρέπει να γίνει η διαδικασία του clustering. Τα σημεία που βρίσκονται έξω από όλες τις συστάδες θεωρούμε ότι είναι ακραίες τιμές και τις αφαιρούμε. [57]

2.1.3 Μετασχηματισμός Δεδομένων (Data Transformation)

Αφου έχουμε σχηματίσει και καθαρίσει το dataset, μπορούμε να μετασχηματίσουμε τα δεδομένα σε καινουργιες μορφές, με στόχο να γίνουν πιο κατάλληλα για το μοντέλο μηχανικής μάθησης. Ανάλογα με το πρόβλημα αλλα και τα δεδομένα, εφαρμόζουμε και την κατάλληλη τεχνική για να επιτευχθεί καλύτερα ο στόχος. Οι τεχνικές μετασχηματισμού δεδομένων (**Data Transformation**) μπορούν να αλλάξουν τις τιμές, την μορφή αλλα ακόμα και την δομή των δεδομένων δημιουργώντας καινουργια ή βελτιώνοντας τα ήδη υπάρχοντα.

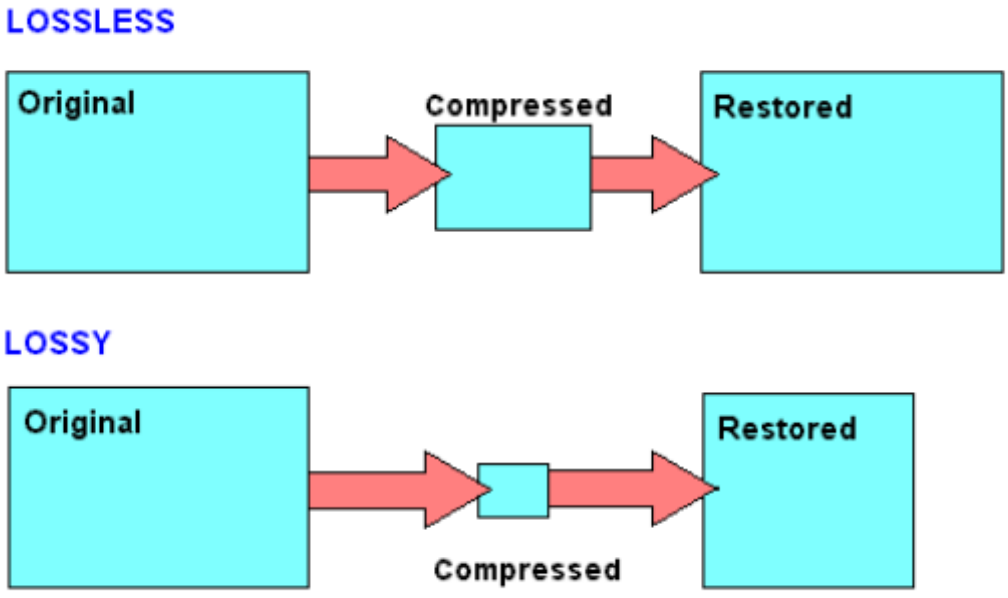
Μερικές μέθοδοι μετασχηματισμού δεδομένων είναι:

- **Κανονικοποίηση - Normalization**: Η κανονικοποίηση είναι απο τις πιο ευρέως διαδεδομένες τεχνικές στον μετασχηματισμό δεδομένων. Σε αυτή όλα τα αριθμητικά δεδομένα κλιμακώνονται προς τα πάνω ή προς τα κάτω, ανάλογα για να χωρέσουν σε ένα συγκεκριμένο εύρος. Ο στόχος που προσπαθούμε να πετύχουμε με το normalization είναι διαφορετικά χαρακτηριστικα να έχουν μια παρόμοια επίδραση στο μοντέλο, μειώνοντας έτσι την πιθανότητα κάποιο χαρακτηριστικό να έχει πολύ μεγαλύτερη επίδραση και να υπερισχύσει, αυξάνοντας όμως το πόσο εύκολα μπορεί να αναγνωρίσει το μοντέλο τις συσχετίσεις ανάμεσα στα δεδομένα. Απο τις πιο συνηθισμένες μορφές είναι η κανονικοποίηση στο εύρος (0,1).
- **Γενίκευση - Generalization**: Τα χαρακτηριστικά δεδομένων που θα μπορούσαν να περιγραφούν ώς χαρακτηριστικά χαμηλού επιπέδου μέσω αυτης της τεχνικής μπορούν να μετατραπούν σε χαρακτηριστικά υψηλότερου επιπέδου, μέσω της δημιουργίας ιεραρχιών στα δεδομένα. Αυτο βοηθάει στην καλύτερη κατανόηση των δεδομένων. Για παράδειγμα, έστω ότι τα δεδομένα έχουν ένα πεδίο το οποίο ονομάζεται “Ηλικία” και έχει τιμές στο (20,30). Μετά την τεχνική αυτή, η μεταβλητή θα μετασχηματιστεί σε κατηγορική με τιμές (νέοι,γέροι).
- **Διακριτοποίηση - Discretization**: Το discretization αφορά την μετατροπή των συνεχών τιμών των δεδομένων σε σύνολα διαστημάτων δεδομένων με συγκεκριμένες τιμές. Αυτο καθιστά πολύ πιο εύκολη την μελέτη και ανάλυση των δεδομένων καθώς οι διακριτές τιμές είναι πολύ πιο εύκολα διαχειρίσιμες απο τις συνεχόμενες. Αυτή η τεχνική είναι γνωστή και ώς τεχνική μείωσης δεδομένων καθώς ουσιαστικά μετατρέπει ένα μεγάλο συνεχές dataset, σε μία σειρα απο κατηγορικά δεδομένα. Τα νέα δεδομένα είναι πολύ πιο ευκολο να παράγουν ακριβή αποτελέσματα χωρίς όμως να έχουμε σοβαρή απώλεια πληροφοριών.
- **Κατασκευή Χαρακτηριστικών - Attribute Construction**: Με αυτήν την τεχνική δημιουργούμε καινούργια χαρακτηριστικά από υπάρχοντα χαρακτηριστικα. Για παράδειγμα, ένα πεδίο “Ονοματεπώνυμο”, με αυτην την τεχνική θα μπορούσε να δημιουργήσει δύο νέα πεδία το “Όνομα” και το “Επώνυμο”.
- **Ομαδοποίηση - Aggregation**: Είναι μια μέθοδος αποθήκευσης και παρουσίασης δεδομένων σε συνοπτική μορφή με βάση ένα πεδίο του dataset.. Για παράδειγμα, τα δεδομένα μπορούν να συγκεντρωθούν και να εμφανιστούν ανάλογα με το όνομα ή την ηλικία. [58]

2.1.4 Μείωση Δεδομένων (Data Reduction)

Η μείωση δεδομένων είναι ένα πολύ σημαντικό βήμα στην προεπεξεργασία δεδομένων. Συνήθως έχουμε έναν υπερβολικά μεγάλο όγκο δεδομένων που καθιστούν πολύ δύσκολη και χρονοβόρα την ανάλυση του. Γι' αυτό το λόγο, χρησιμοποιούμε τεχνικές για να μειώσουμε τον όγκο των δεδομένων αλλα να κρατήσουμε όσο το δυνατόν περισσότερη πληροφορία γίνεται. Δυστυχώς όμως πολλές φορές λόγω της δομής των δεδομένων, δεν μπορούμε να μειώσουμε το μέγεθος χωρίς να χάσουμε σημαντικές πληροφορίες. Γι αυτό τον λόγο υπάρχουν δύο ειδών τεχνικές μείωσης δεδομένων: [60]

- **Συμπίεση χωρίς απώλειες - Lossless Compression**: Όπως φανερώνει και το όνομα, σε αυτήν την περίπτωση είναι εφικτό να ξαναδημιουργησουμε τα δεδομένα σε μια καινούργια πιο συμπυκνωμένη μορφή χωρίς να χάσουμε σημαντικές πληροφορίες. Συνήθως χρησιμοποιείται για πιο απλές μορφές δεδομένων όπως αρχεία κειμένου. Αυτο το καταφέρνουμε μέσω τεχνικών εξάλειψης πλεονασμού (**redundancy elimination techniques**). Εξ ορισμού, δεν υπάρχει μία τεχνική που να μπορεί να συμπιέσει τα δεδομένα κατα τον τέλειο τρόπο. Γι αυτό το λόγο ανάλογα με τα δεδομένα και τον πιθανό πλεονασμό που έχουν, επιλέγουμε την κατάλληλη τεχνική. Συνήθως τεχνικές συμπίεσης αυτού του είδους χρησιμοποιούνται όταν έχουμε να τροποποιήσουμε δεδομένα που δεν μας επιτρέπεται να "χάσουμε" πληροφορία και θέλουμε τα συμπιεσμένα δεδομένα να είναι όσο γίνεται παρόμοια με τα αρχικά. Όταν προσπαθήσουμε να ξαναδημιουργησουμε τα αρχικά δεδομένα από τα τελικά, τότε θα μπορούσαμε να το κάνουμε με απόλυτη επιτυχία. Μερικές από τις μεθόδους είναι η Lempel-Ziv συμπίεση, η Run Length συμπίεση και η συμπίεση Huffman.



Εικ. 2.1 Lossless vs Lossy

- **Συμπίεση με απώλειες - Lossy Compression**: Σε αυτήν την περίπτωση, κατα την συμπίεση δεν μπορούμε να αποφύγουμε την απώλεια πληροφορίας των δεδομένων. Συνήθως χρησιμοποιείται για πιό πολύπλοκες μορφές δεδομένων όπως εικόνες και βίντεο. Σε αντίθεση με την συμπίεση χωρίς απώλειες όμως, εδώ τα τελικά δεδομένα λόγω της απώλειας πληροφοριών θα είναι χειρότερα από τα αρχικά. Όμως η συμπίεση με απώλειες μπορεί να χρησιμοποιηθεί σε ένα μεγαλύτερο εύρος δεδομένων. Ουσιαστικά, προσπαθούμε να

αφαιρέσουμε μικρές πληροφορίες που πολλές φορές δεν είναι ούτε καν ορατες στο ανθρώπινο μάτι με στόχο την πιο αποτελεσματική συμπίεση. Σε αυτήν την περίπτωση, αν προσπαθήσουμε να ξαναδημιουργησουμε τα αρχικά δεδομένα, δεν θα τα καταφέρουμε τόσο καλά επειδή έχουμε χάσει ένα κομμάτι της αρχικής πληροφορίας. Στο παρακάτω παράδειγμα μπορούμε να διακρίνουμε με ευκολία οτι στη εικόνα αναπαριστάται το πρόσωπο ενός σκύλου ενω η μόνη ευδιάκριτη διαφορά που μπορεί να παρατηρηθεί με το μάτι είναι ένα πιξελάρισμα στο φόντο στο πάνω δεξιά μέρος της φωτογραφίας. [61]

Εικ. 2.2 Παράδειγμα Lossy Compression

2.1.5 Dataset Cat-Vs-Dogs

Στο δικό μου dataset, τα περισσότερα απο αυτά τα βήματα έχουν ήδη γίνει από τους δημιουργούς του, το μοναδικό που εντόπισα και έπρεπε όντως να γίνει κάποια αλλαγή ήταν ότι μερικές εικόνες ήταν corrupted και έπρεπε να αφαιρεθούν.

2.2 Μοντέλα Κατηγοριοποίησης

Για την κατηγοριοποίηση των εικόνων χρησιμοποιούν κάποια μοντέλα επιβλεπόμενης μάθησης που ονομάζονται μοντέλα κατηγοριοποίησης (**classification models**). Στόχος αυτών των μοντέλων είναι να αναγνωρίζουν μοτίβα από τα δεδομένα εισόδου και να χρησιμοποιούν αυτά τα μοτίβα για να ταξινομήσουν νέα δεδομένα που δεν έχουν ξανασυναντήσει σε μια από τις κλάσεις. Αυτού του είδους τα μοντέλα χρησιμοποιούνται σε διάφορους τομείς, όπως η αναγνώριση εικόνων, η ανίχνευση ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου, η ιατρική διάγνωση. Όπως φαίνεται και απο τους τομείς, δεν μας περιορίζει το είδος των δεδομένων εισόδου αφου μπορεί να λειτουργήσει το ίδιο αποδοτικά και με απλά δεδομένα όπως αρχεία κειμένου αλλα και με ποιο πολύπλοκα όπως ταξινόμηση εικόνων. [63]

Μερικά μοντέλα είναι:

- **Naive Bayes:** Πρόκειται για έναν από τους απλούστερους αλλά πιο αποτελεσματικούς αλγόριθμους ταξινόμησης, ο οποίος εφαρμόζει το θεώρημα του Bayes. Το θεώρημα του Bayes είναι ένα μαθηματικό θεώρημα που ανήκει στη θεωρία των πιθανοτήτων και ασχολείται με τις υπό συνθήκη πιθανότητες. Ο τύπος του είναι:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Μαθηματικός τύπος 2.1

και ερμηνεύεται ώς η πιθανότητα να συμβεί το ενδεχόμενο Α, δεδομένου του Β, ισούται με την πιθανότητα να συμβεί το Β δεδομένου του Α, επί την πιθανότητα να συμβεί το Α, διά την πιθανότητα να συμβεί το Β.
Είναι ιδιαίτερα χρήσιμος για ταξινόμηση κειμένου και κατηγορικών δεδομένων.

- **Δέντρα αποφάσεων**:Ένας άλλος δημοφιλής αλγόριθμος είναι τα δέντρα αποφάσεων **decision trees**. Τα δέντρα αποφάσεων είναι ένα ιεραρχικό μοντέλο που μοιάζει με δέντρο στο οποίο κάθε φύλλο αντιπροσωπεύει τα labels-κλάσεις και τα κλαδιά αντιπροσωπεύουν τις συνδέσεις των χαρακτηριστικών που οδήγησαν στα φύλλα. Μερικά από τα πλεονεκτήματα τους ειναι οτι ειναι ευκολα στην κατανοηση, δεν θέλουν πάρα πολύ προεπεξεργασία δεδομένων και μπορεί να επαληθευτεί και απο μαθηματικά και πιο συγκεκριμένα την στατιστική.
- **Τυχαίο Δάσος και Ενίσχυση Κλίσης**: Πρόκειται για δύο μεθόδους που συνδυάζουν πολλαπλά δέντρα αποφάσεων για να επιτύχουν μεγαλύτερη ακρίβεια. Το **Random Forest** δημιουργεί πολλά δέντρα και υπολογίζει τον μέσο όρο των αποτελεσμάτων τους, ενώ το **Gradient Boosting** δημιουργεί διαδοχικά δέντρα, με την λογική ότι κάθε επόμενο εστιάζει στη βελτίωση των αποτελεσμάτων του προηγούμενου δέντρου.
- **Μηχανές Υποστήριξης Διανύσματος**: Τα **Support Vector Machines (SVM)** αποτελούν τις πιό ισχυρές μεθόδους πρόβλεψης και στόχος τους είναι να βρουν το υπερεπίπεδο που διαχωρίζει καλύτερα τις διαφορετικές κλάσεις, ενώ μεγιστοποιεί την απόσταση μεταξύ τους. Τέλος, για την πρόβλεψη των άγνωστων δεδομένων τα χαρτογραφεί στο ίδιο επίπεδο με τα γνωστά και ανάλογα με την μεριά που θα βρίσκεται στο χώρο, θα προβλέψει και την αντίστοιχη κλάση. Ένα από τα μεγαλύτερα πλεονεκτήματα των SVM είναι, ότι, είναι το ίδιο αποτελεσματικά και σε προβλήματα γραμμικής και μη γραμμικής ταξινόμησης.
- **Κ-Κοντινότεροι Γείτονες**: Το **K-Nearest Neighbors (KNN)** είναι μια ιδιαίτερη περίπτωση οπου το αποτέλεσμα εξαρτάται από τα υπόλοιπα data. Πιο συγκεκριμένα ένα δεδομένο εισόδου θα ταξινομηθεί με βάση την πιό κοινή κλάση ανάμεσα στους Κ κοντινότερους γείτονες του. Προφανώς το Κ μπορεί να πάρει μόνο θετικές ακέραιες τιμές και αν το Κ παρει την τιμή ένα τότε το δεδομένο θα ταξινομηθεί με βάση τον κοντινότερο γείτονά του.
- **Βαθιά μάθηση**: Το **Deep Learning** έχει επίσης φέρει σημαντικές προόδους στην ταξινόμηση με τα νευρωνικά δίκτυα. Τα **συνελικτικά νευρωνικά δίκτυα (CNN)** κυριαρχούν στην ταξινόμηση εικόνων μαθαίνοντας αυτόματα ιεραρχικά χαρακτηριστικά από τα δεδομένα. [64]

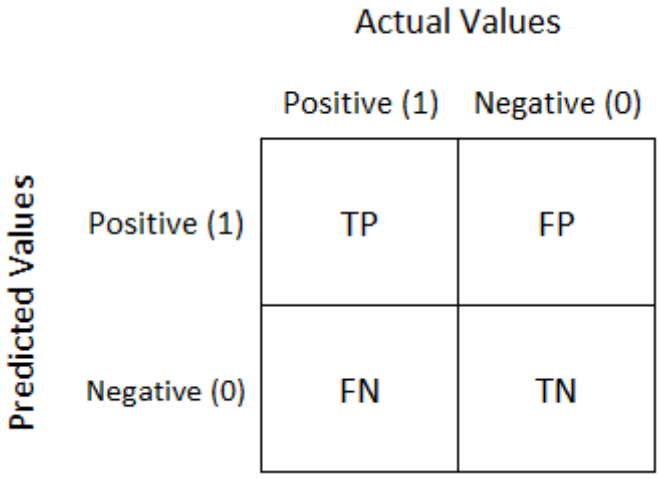
2.2.1 Τρόποι Αξιολόγησης Μοντέλων

Πρέπει όμως να μπορούμε να αξιολογήσουμε τα μοντέλα και την αποτελεσματικότητα τους σε σχέση με το πρόβλημα. Καθώς ανάλογα με το πρόβλημα υπάρχουν πιο κατάλληλα μοντέλα και περισσότερο κατάλληλες τροποποιήσεις και υπερ-παράμετροι που μπορούν να βρεθούν.Υπάρχουν πολλές διαφορετικές μετρικές για να μετρήσουν την αποτελεσματικότητα ενός μοντέλου. Για να καταλάβουμε καλύτερα τι μετράει η κάθε τεχνική κάλο θα ήταν να ορίσουμε πρώτα τον πίνακα συσχέτισης.

2.2.1.1 Πίνακας Συσχέτισης

Ο **πίνακας συσχέτισης** (**confusion matrix)** είναι ένα από τα πιό σημαντικά εργαλεία που διαθέτουμε για να ορίσουμε τις μετρικές που μετράνε την αποτελεσματικότητα του μοντέλου. Είναι ένας πίνακας που οπτικοποιεί την αποτελεσματικότητα του μοντέλου στο να κατατάσσει τα δεδομένα στις διάφορες κλάσεις που έχουν ορισθεί. Ο πίνακας είναι δισδιάστατος με γραμμές και στήλες. Κάθε γραμμή έχει το σύνολο των δεδομένων στις πραγματικές τους κλάσεις και κάθε στήλη έχει το σύνολο των

δεδομένων που προβλέφθηκαν από το μοντέλο. Τα διαγώνια στοιχεία αντιπροσωπεύουν τα σωστά ταξινομημένα παραδείγματα για κάθε κλάση, ενώ τα στοιχεία εκτός διαγωνίου υποδηλώνουν τα λαθος. Ένας πίνακας συσχέτισης θα έχει την παρακάτω μορφή.



Εικ. 2.3 Confusion Matrix

Για να αναλύσουμε λίγο τις συντομογραφίες:

- **TP - True Positive**: Τα δεδομένα που προβλεφθηκαν απο το μοντέλο ως θετικά και είναι όντως θετικά.
- **TN - True Negative**: Τα δεδομένα που προβλεφθηκαν απο το μοντέλο ως αρνητικά και είναι όντως αρνητικά,
- **FN - False Positive:**Τα δεδομένα που προβλεφθηκαν απο το μοντέλο ως θετικά, ενώ στην πραγματικότητα είναι αρνητικά. Το False Positive είναι γνωστό και ώς Λάθος Τύπου I (**Type I Error**)
- **FP - False Negative:**Τα δεδομένα που προβλεφθηκαν απο το μοντέλο ως αρνητικά, ενώ στην πραγματικότητα είναι θετικά. Το False Negative είναι γνωστό και ώς Λάθος Τύπου II (**Type II Error**) [65]

2.2.2 Μετρικές Αξιολόγησης Μοντέλων

Τώρα που έχουμε ορίσει τον πίνακα συσχέτισης, μπορούμε να ορίσουμε πιο εύκολα τις διαφορετικές μετρικές που υπάρχουν. Μερικές από αυτές είναι:

- **Ακρίβεια (Accuracy)**: Η ακρίβεια είναι από τις πιό διαδεδομένες μετρικές και μετράει τον λόγο όλων των σωστών προβλέψεων ως προς όλες τις προβλέψεις. Είναι ειδική όταν όλες οι κλάσεις είναι ισοπίθανες και το dataset έχει επαρκή δείγματα από όλες καθώς τοτε το false positive και το false negative έχουν παρόμοια βαρύτητα.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Μαθηματικός τύπος 2.2

- **Ακρίβεια (Precision)**: Η ακρίβεια - Precision μετράει τον λόγο των ταξινομημένων θετικών προβλέψεων ως προς όλες τις θετικές προβλέψεις. Χρησιμοποιείται κυρίως όταν θέλουμε να μειώσουμε τα false positives.

$$Precision = \frac{TP}{TP + FP}$$

Μαθηματικός τύπος 2.3

- **Ανάκληση (Recall)**: Η ανάκληση μετράει το λόγο των σωστά ταξινομημένων θετικών προβλέψεων ως προς όλες τις πραγματικά θετικές προβλέψεις. Είναι γνωστό και ώς **True Positive Rate** αφού ουσιαστικά μετράει το κατα πόσο το μοντέλο μπορεί να προβλέψει τις πραγματικά θετικές προβλέψεις ώς θετικές**.** Χρησιμοποιείται κυρίως όταν θέλουμε να μειώσουμε τα false negatives.

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$

Μαθηματικός τύπος 2.4

- **Ειδικότητα (Specificity)**: Το ακριβώς αντίθετο απο την ανάκληση αφού μετράει το σε τί βαθμό το μοντέλο μπορεί να προβλέψει τις πραγματικά αρνητικές τιμές. Ορίζεται ώς τον λόγο τον αρνητικών προβλέψεων ως προς τις πραγματικά αρνητικές προβλέψεις. Είναι γνωστό και ως **True Negative Rate**. Χρησιμοποιείται όταν στόχος είναι η αναγνώριση των αρνητικών τιμών, επειδή έχουν μεγαλύτερες επιπτώσεις.

$$Specificity = \frac{TN}{N} = \frac{TN}{TN + FP}$$

Μαθηματικός τύπος 2.5

- **F1- Score**: Το F1- Score βρίσκεται κάπου στην μέση του recall και του precision και κατα μία έννοια μετράει ταυτόχρονα και τα δύο. Πρόκειται για το διπλάσιο του λόγου recall επι precision προς το σύνολο recall και precision. Χρησιμοποιείται όταν είναι εξίσου σημαντικά και η ανάκληση και η ειδικότητα και ειδικά εάν οι κλάσεις δεν έχουν τον ίδιο αριθμό δειγμάτων.

$$F1 = 2 * \frac{recall * precision}{recall + precision} = \frac{2TP}{2TP + FP + FN}$$

Μαθηματικός τύπος 2.6

Υπάρχουν και άλλες μετρικές αλλα οι παραπάνω είναι οι πιό διαδεδομένες και χρησιμοποιούνται στα περισσότερα προβληματα. Όπως παρατηρούμε ανάλογα με τα δεδομένα και το πρόβλημα που θέλουμε να “λυθεί” με το μοντέλο επιλέγουμε και την κατάλληλη μετρική έτσι ώστε να διασφαλιστεί η σωστή αναπαράσταση της αποδοτικότητας του. Συχνά μπορεί να χρειαστεί να χρησιμοποιήσουμε και περισσότερες από μία τεχνικές γιά να έχουμε μια πιό ολοκληρωμένη εικόνα εάν το πρόβλημα είναι υπερβολικά πολύπλοκο. [66]

2.2.3 Μοντελο κατηγοριοποιησης και μετρικές που επέλεξα

Μετά από μια ανάλυση των δεδομένων που ειχα στην διάθεση μου και του προβλήματος κατηγοριοποίησης που είχα να επιλύσω, ο διαχωρισμός των εικόνων σε δύο κλάσεις, αποφασισα να χρησιμοποιήσω **Deep Learning** μοντέλα και συγκεκριμένα συνελικτικά δίκτυα αφού τα δεδομένα μου ήταν εικόνες, **accuracy** ως κύριο μέτρο της αποτελεσματικότητας του μοντέλου καθώς οι εικόνες έχουν ίδιο πλήθος άρα μιλάμε για balanced dataset (περιπου 12500 στην κάθε κλάση για ένα σύνολο σχεδόν των 25000). Τέλος αποφάσισα για loss function να χρησιμοποιήσω Cross Entropy Loss.

2.2.4 Μάθηση Με Μεταφορά (Transfer Learning)

Όπως αναφέρθηκε και πιό πάνω για να εκπαιδευτεί σωστά ένα βαθύ μοντέλο μηχανικής μάθησης, πρέπει να υπάρχει ένας αρκετά μεγάλος όγκος δεδομένων, που είναι όσο πιο ισόποσα κατανεμημένη ανάμεσα στις πιθανές κλάσεις, ετσι ώστε να μην υπάρχουν φαινόμενα underfitting και overfitting. Στην συνέχεια το νευρωνικό θα αναλύσει τα δεδομένα και θα αναπτύξει μαθηματικές συσχετίσεις που αντιπροσωπεύουν τα μοτίβα που παρατηρούνται ανάμεσα στα δεδομένα της ίδιας τάξης. Αυτό όμως δημιουργεί κάποια προβλήματα.

Ευτυχώς υπάρχουν πάρα πολλά μεγάλα dataset ανοιχτού τύπου που μπορούν εύκολα να βρεθούν στο διαδίκτυο και καλύπτουν ποικιλους τομείς και τύπους δεδομένων. Για παράδειγμα, το “ImageNet” είναι από τα πιο μεγάλα και γνωστά dataset με πάνω από δεκατεσσερα εκατομμυρια εικόνες και περίπου είκοσι χιλιάδες κλάσεις. Αντίστοιχα, για **Natural Language Processing (NLP)** μοντέλα υπάρχει το “IMDB reviews” που περιέχει πενήντα εκατομμύρια κριτικές από ταινίες, ή το “WordNet”, που περιέχει πάνω από εκατό χιλιάδες λέξεις και τις ετυμολογικές συνδέσεις μεταξύ τους.

Όμως για να εκπαιδευτεί ένα μοντέλο με ένα τόσο μεγάλο dataset χρειάζεται πάρα πολύ ισχυρούς πόρους σε επίπεδο hardware (πολλαπλές GPU, μεγάλες RAM, Google’s TPU) έτσι ώστε να εκπαιδευτούν σε ένα ικανοποιητικό χρόνο. Πολλές φορές αυτούς τους πόρους δεν μπορούμε να τους έχουμε οπότε δημιουργήθηκε η ανάγκη για την δημιουργία ενός τρόπου ετσι ώστε να μπορούμε να χρησιμοποιήσουμε τα ήδη υπάρχοντα μοντέλα για παρόμοια προβλήματα.

Η **μάθηση με μεταφορά** είναι ακριβώς αυτό. Αντι δηλαδή να κατασκευάσουμε απο την αρχη τα μοντέλα για κάθε νέο πρόβλημα, εφόσον έχουν την ίδια αρχιτεκτονική, μπορούμε να μεταφέρουμε τα ήδη προ-εκπαιδευμένα βάρη από τα μοντέλα, που έχουν εκπαιδευτεί σε ένα μεγαλύτερο και πιο περίπλοκο dataset και έχουν μάθει πολλά διαφορετικά μοτίβα. Με αυτόν τον τρόπο και αλλάζοντας μόνο το τελευταίο επίπεδο το επίπεδο εξόδου, ετσι ώστε να να έχουμε σαν έξοδο τις κλάσεις που έχουμε στο πρόβλημα, μπορούμε να πετύχουμε πολύ ταχύτερη σύγκληση και μεγαλύτερη αποτελεσματικότητα. Η μάθηση με μεταφορά είναι ιδιαίτερα χρήσιμη όταν το πρόβλημα που καλούμαστε να λύσουμε έχει περιορισμένο αριθμό δεδομένων και η ευρεση ή προσθήκη καινούργιων δεδομένων είναι αδύνατη ή όταν έχουμε περιορισμένους υπολογιστικούς πόρους. [67]

Υπάρχουν πάρα πολλά μοντέλα που βρίσκονται στην κορυφή του πεδίου τους και έχουν εκπαιδευτεί και μπορούν να χρησιμοποιηθούν για πληθώρα προβλημάτων. Μερικά απο αυτα είναι:

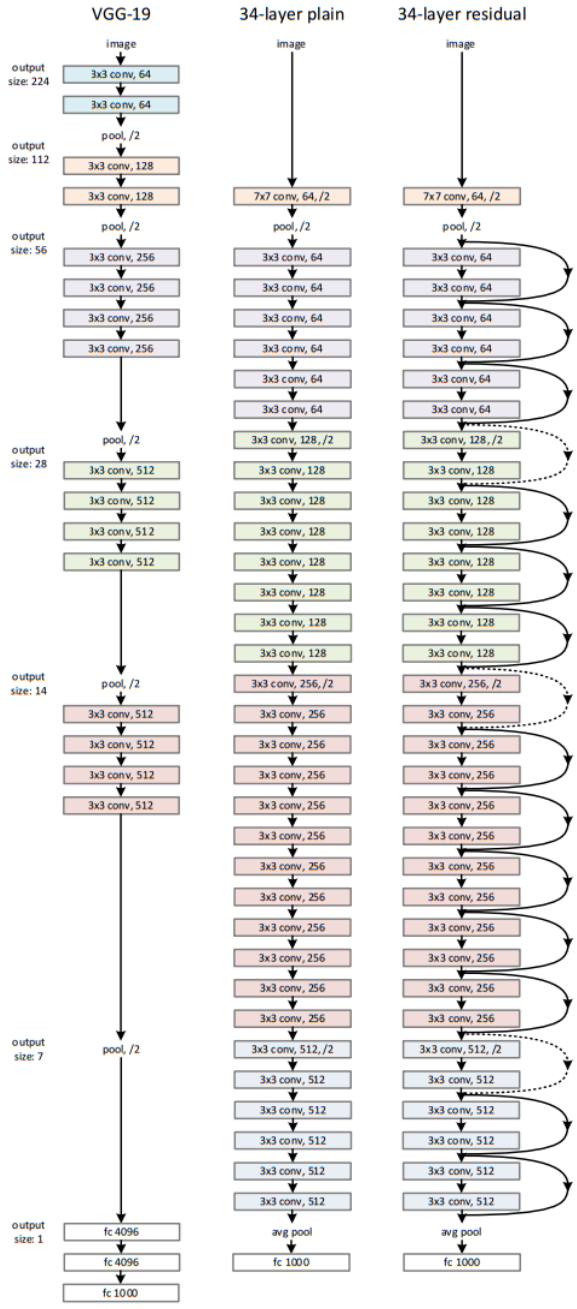
- Image Classification:
    - AlexNet
    - VGG
    - Inception-v3
    - ResNet
    - VGG [68]

- NLP:
    - BERT
    - GPT [69]
- Object detection:
    - YOLO
    - U-Net [70]

Υπάρχουν πολλά ακόμα σε διάφορους τομείς, αλλα το σημαντικό είναι πως βλέποντας την πληθώρα των μοντέλων καθως και του εύρος των πεδίων που καλύπτουν, πλέον η μάθηση με μεταφορά αποτελεί ένα πολύ συχνό φαινόμενο. Αυτο γιατι μπορούμε να πάρουμε μια γενική γνώση και να την τελειοποιήσουμε για το πρόβλημα που έχουμε να αντιμετωπίσουμε, κερδίζοντας έτσι χρόνο και πόρους.

Στο δικό μου πρόβλημα αποφάσισα να χρησιμοποιήσω ResNet και μάλιστα την αρχιτεκτονική με τα 50 στρώματα, το **ResNet-50**. [71]

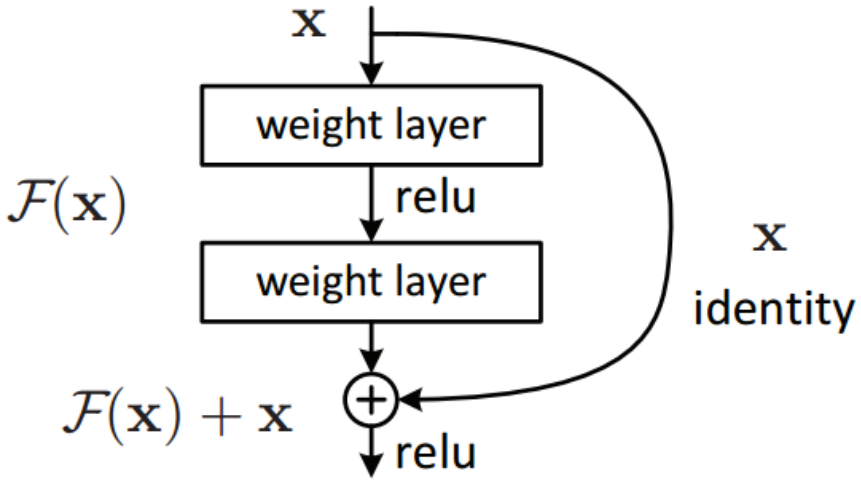
2.2.5 Residual Network - ResNet



Εικ. 2.4 ResNet-34 Architecture

Ένα Υπολειπόμενο Νευρωνικό Δίκτυο (**Residual Neural Network - ResNet**) είναι ένα μοντέλο βαθιάς μηχανικής μάθησης, που δημιουργήθηκε το 2015 στα πλαίσια του διαγωνισμού "ImageNet". Το έναυσμα για την δημιουργία του αποτέλεσε το πρόβλημα του υποβιβασμού (degradation) που παρουσιάζουν πολλά μοντέλα στην προσπάθεια τους για επίλυση ακόμα πιο πολύπλοκων προβλημάτων. Όταν τα δίκτυα γίνονται βαθύτερα, με περισσότερα κρυφά στρώματα δηλαδή, παρατηρείται μια γρήγορη μείωση στην ακρίβεια των μοντέλων με αποτέλεσμα το μοντέλο να χειροτερεύει.

Για να καταπολεμήσει αυτο ακριβώς το πρόβλημα το ResNet πρόσθεσε νέα σύνθετα μπλοκ στην αρχιτεκτονική του έτσι ώστε να του δίνεται η δυνατότητα να παραλείπει στρώματα μέσω μιας σύνδεσης που ονομάζεται **identity**.[72]



Εικ. 2.5 Basic Residual Block

2.2.5.1 Residual Blocks

Εκτός από το παραπάνω μπλοκ το οποίο είναι το κλασικό, στο ResNet χρησιμοποιούνται και καποιες παραλλαγές αυτού που έχουν ως στόχο την καλύτερη επίτευξη των στόχων.

- **Basic Block:** Πρόκειται για το πιο βασικό μπλοκ στο ResNet. Περιέχει δύο διαδοχικά 3x3 συνελικτικά στρώματα (Convolutional layers) και μια residual σύνδεση - identity (εικ.2.4). Η είσοδος και η έξοδος έχουν ίδιες διαστάσεις.
- **Bottleneck Block:** Περιέχει τρία διαδοχικά συνελικτικά στρώματα που έχουν διαφορετικές διαστάσεις μεταξύ τους και μια residual σύνδεση. Το πρώτο συνελικτικό στρώμα είναι ένα στρώμα συνέλιξης 1x1 με στόχο την μείωση των διαστάσεων, το δεύτερο είναι ένα συνελικτικό 3x3 και το τρίτο είναι πάλι 1x1 αλλα αυτην την φορα για αυξηση των διαστάσεων στην αρχική τους τιμή. Αυτά τα μπλοκς τα βλεπουμε κυρίως στα μοντέλα ResNet-50, ResNet-101 και ResNet-152. [72]
- **Pre-Activation Block**: Η μεγαλύτερη διαφορά του είναι ότι αλλάζει την σειρά που γίνονται ορισμένες πράξεις μέσα στο μπλοκ. Πιο συγκεκριμενα, εφαρμόζονται οι συναρτήσεις ενεργοποιήσης πριν τα συνελικτικά στρώματα. Αυτό έχει το πλεονέκτημα οτι δεν χρειάζονται τόσες πολλες κανονικές συνδέσεις μεταξύ των υπολοίπων μπλοκ, αυξάνοντας την σύγκληση του μοντέλου. Τα pre-activation μπλοκς χρησιμοποιούνται κυρίως σε πάρα πολύ μεγάλα και πολύπλοκα βαθιά νευρωνικά δίκτυα, με το πιο γνωστό παράδειγμα να είναι το GPT.[73]

Υπάρχουν πολλά ακόμα residual μπλοκς που θα μπορούσαμε να αναφέρουμε και να αναλύσουμε αλλα για τις ανάγκες την πτυχιακής θα ασχοληθούμε μονο με Bottleneck Blocks.

2.2.5.2 ResNet Variations

Όσο πιο διαδεδομένο και πετυχημένο γινόταν το αρχικό ResNet τόσο περισσότερες παραλλαγές ξεκίνησαν να εμφανίζονται, τροποποιημένες ανάλογα για τις ανάγκες του κάθε προβλήματος.

- **ResNet-18 και ResNet-34**: Πρόκειται για πιό “ρηχά” (όρος που χρησιμοποιείται για να υποδηλόσουμε το αντίθετο του βαθιά έχει δηλαδή λίγα κρυφά στρώματα) μοντέλα σε σχέση με το αρχικό ResNet. Οι αριθμοί δίπλα στο όνομα του ResNet υποδηλώνουν τον αριθμό των στρωμάτων, άρα μιλάμε για ένα ResNet με δεκα οκτω και τριαντα τέσσερα στρώματα αντίστοιχα. Πρόκειται για μοντέλα τα οποία προσπαθούν να έχουν την πιο αποδοτική εξαγωγή χαρακτηριστικών των δεδομένων με όσο το λιγότερο υπολογιστικούς πόρους γίνεται.
- **ResNet-101 και ResNet-152**: Πρόκειται για πιό βαθιά μοντέλα με εκατόν ένα και εκατόν πενήντα δύο κρυφά στρώματα αντίστοιχα. Είναι πολύ πιο αποτελεσματικά σε πολύ πιο πολύπλοκα προβλήματα και μπορούν να ανακαλύψουν πιο πολλές συσχετίσεις ανάμεσα στα δεδομένα με στόχο την σωστή ταξινόμηση τους. Στα μειονεκτήματά τους όμως, είναι ότι χρειάζονται πάρα πολλούς υπολογιστικούς πόρους και πολύ μεγάλα dataset για να εκπαιδευτούν σωστά.

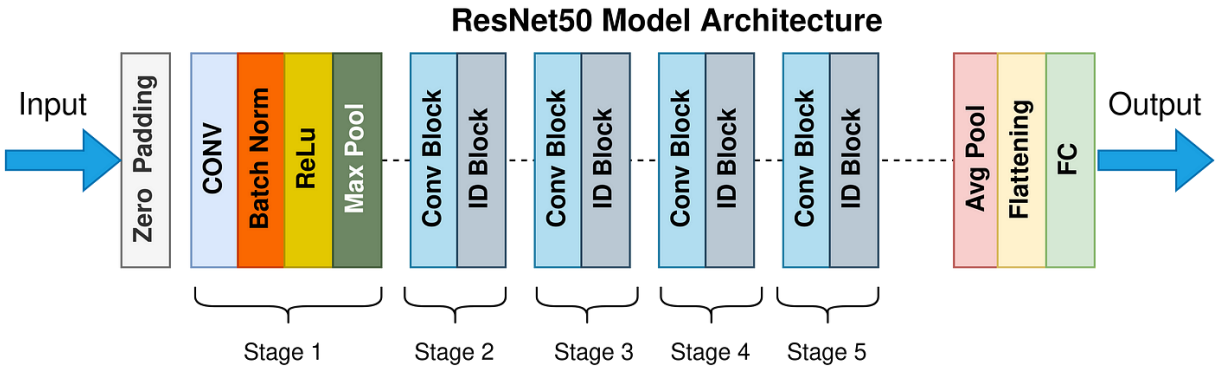
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Εικ. 2.6 Αρχιτεκτονικές διάφορων ResNet

- **Wide ResNet**: Η καινοτομία του βρίσκεται στο οτι αντι να χρησιμοποιεί κλασσικά ResNet blocks, χρησιμοποιεί Wide ResNet blocks. Η διαφορα τους με τα απλά είναι ότι χρησιμοποιούν περισσότερα φίλτρα, έτσι ώστε να αυξηθεί η ικανότητα του μοντέλου να μαθαίνει πιο πολύπλοκα μοτίβα χωρίς όμως να αυξήσουμε τα στρώματα και άρα να αυξήσουμε τις απαιτήσεις του μοντέλου. [76]
- **DenseNet**: Το DenseNet πήρε περισσότερο έμπνευση απο το ResNet παρά να αποτελεί παραλλαγή αυτου. Παρολα αυτα πρέπει να γίνει μία αναφορά καθώς αποτελεί πλέον μια απο τις βασεις για τα πιό σύγχρονα και διαδεδομένα μοντέλα-παραλλαγές αυτου. Το DensNet λοιπόν έχει την ιδιομορφία οτι όλα τα στρώματα συνδέονται με όλα τα επόμενα. Αυτό βοηθάει στο να είναι το μοντέλο πιο μικρο και να απαιτεί λιγότερους υπολογιστικούς πόρους χωρίς όμως να χάνει την δυνατότητα του να μαθαίνει πολύπλοκα μοτίβα.[77]

Στην επόμενη υποενότητα θα αναλύσουμε το ResNet-50 με περισσότερη λεπτομέρεια αφού πρόκειται για το μοντέλο που επέλεξα για την κατηγοριοποίηση των εικόνων.

2.2.5.3 ResNet-50



Εικ.2.7 Αρχιτεκτονική ResNet-50

Το ResNet-50 πρόκειται για μία παραλλαγή του αρχικού ResNet, με πενήντα στρώματα. Η κύρια χρήση του όπως και όλων των μοντέλων ResNet είναι για προβλήματα που έχουν να κάνουν με οπτικό υλικό, δηλαδή τα δεδομένα εισόδου συνήθως είναι εικόνες και βίντεο. Η καινοτομία του σε σχέση με τα υπόλοιπα, όπως αναφέρθηκε και παραπάνω, είναι τα redisual bottleneck blocks, που κάνουν την αναπαράσταση των δεδομένων πιο ικανοποιητική και άρα αυξάνουν την απόδοση του μοντέλου.[72] Ο αριθμός των στρωμάτων που διαθέτει είναι ιδανικός για προβλήματα με πολύπλοκα δεδομένα που διαθέτουν συνθετες συσχετίσεις μεταξύ τους αλλα πρεπει ταυτόχρονα να διαθέτουμε υπολογιστική ισχύ ετσι ώστε να μπορέσει να μάθει το μοντέλο. Με αυτόν τον τρόπο, το μοντέλο καταφέρνει να ξεπεράσει και τα μικρότερα variations του ResNet, αλλα μπορεί να συναγωνιστεί επάξια και τα πιο βαθια variations. Η επιλογή του μοντέλου πρέπει πάντα να γίνεται ανάλογα με το πρόβλημα. Εγω επέλεξα το ResNet-50 επειδη παρόλο που το πρόβλημα μου είναι ένα πρόβλημα δυαδικής ταξινόμησης, οι εικόνες που πρέπει να ταξινομήσω είναι αρκετά περίπλοκες και οι δύο κλάσεις έχουν αρκετά κοινά μεταξύ τους για να πρέπει το μοντέλο να αναζητήσει πιο εξειδικευμένα μοτίβα.

Ας αναλύσουμε όμως λίγο την αρχιτεκτονική του ResNet-50.

- **Στρώμα εισόδου**: Το ResNet-50 δέχεται ως είσοδο ένα δεδομένο με 3 κανάλια RGB και μέγεθος 224x224.
- **Κρυφά στρώματα**:
  - **Στάδιο 1 - Stage 1**:
    - Convolution layer, 7x7, με 64 filters με batch normalization και ReLU activation
    - Max Pooling layer 3x3
  - **Στάδιο 2 - Stage 2**:

    Πρόκειται για 3 επαναλαμβανόμενα Residual Bottleneck Blocks. Η δομή τους είναι:

    - Convolution layer, 1x1,64 filter με batch normalization και ReLU activation
    - Convolution layer, 3x3,64 filter με batch normalization και ReLU activation
    - Convolution layer, 1x1,256 filters με batch normalization
    - Skip Connection - Identity
  - **Στάδιο 3 - Stage 3**:

    Πρόκειται για 4 επαναλαμβανόμενα Residual Bottleneck Blocks. Η δομή τους είναι:

    - Convolution layer, 1x1,128 filter με batch normalization και ReLU

        - Convolution layer, 3x3,128 filter με batch normalization και ReLU
        - Convolution layer, 1x1,512 filters με batch normalization
        - Skip Connection - Identity
    - **Στάδιο 4 - Stage 4**:

      Πρόκειται για 6 επαναλαμβανόμενα Residual Bottleneck Blocks. Η δομή τους είναι:

        - Convolution layer, 1x1,256 filter με batch normalization και ReLU activation
        - Convolution layer, 3x3,256 filter με batch normalization και ReLU activation
        - Convolution layer, 1x1,1024 filters με batch normalization
        - Skip Connection - Identity
    - **Στάδιο 5 - Stage 5**:

      Πρόκειται για 3 επαναλαμβανόμενα Residual Bottleneck Blocks. Η δομή τους είναι:

        - Convolution layer, 1x1,512 filter με batch normalization και ReLU activation
        - Convolution layer, 3x3,512 filter με batch normalization και ReLU activation
        - Convolution layer, 1x1,2048 filters με batch normalization
        - Skip Connection - Identity
    - **Global Average Pooling layer**
- **Στρώμα εξόδου**:
    - Fully connected layer, Softmax συνάρτηση ενεργοποίησης, η έξοδος είναι ο αριθμός των κλάσεων [72]

2.2.6 Ταξινομητής ResNet-50 που χρησιμοποίησα

Η παραπάνω ιεραρχία όμως δεν θα με βοηθούσε καθόλου γιατί όταν χρησιμοποιώ ήδη εκπαιδευμένα μοντέλα για να γίνει η μεταφορά θα πρέπει να έχουν ίδια μορφή, δηλαδή ίδιο αριθμό στρωμάτων με τα ίδια χαρακτηριστικά. Τα βάρη που φόρτωσα όμως έχουν σαν βάση το ResNet-50 που έχει εκπαιδευτεί στο ImageNet dataset. Το τελευταίο layer δηλαδή έχει σαν έξοδο 1000 κλάσεις, ενω εγώ θα ήθελα να έχει σαν έξοδο 2 κλάσεις.
Οπότε αυτό που έκανα είναι να κάνω load το μοντέλο με τα pretrained βάρη απο το imagenet, πάγωσα το μοντέλο έτσι ωστε να το χρησιμοποιήσω σαν feature extractor και και να κρατήσω τα μοτίβα που έχει μάθει απο το πιο πολύπλοκο και δύσκολο dataset. Τέλος, άλλαξα το τελευταίο layer (fc-fully connected), κρατώντας τον αριθμο των features - νευρώνων του προτελευταίου layer για να μπορέσω να τα συνδεσω με το καινούργιο. Στην συνέχεια έβαλα δύο καινουργια fully connected layers, με τελική έξοδο δύο κλάσεις μία για τους σκύλους και μία για τις γάτες.

Ο λόγος που έβαλα δύο είναι γιατί παρόλο που μιλάμε για μόνο δύο κλάσεις, και θεωρείται σχετικά εύκολο για πρόβλημα κατηγοριοποίησης, υπάρχουν μερικές εικόνες που είναι δυσκολότερο να ταξινομηθούν σωστά λόγω διαφόρων παραγόντων όπως για παράδειγμα τις θέσεις των ζώων ή ο θόρυβος από το περιβάλλον. Έτσι μετά από πειράματα και με περισσότερα και με λιγότερα στρώματα, κατέληξα ότι το καλύτερο αποτέλεσμα το ειχα με την παραπάνω αρχιτεκτονική.

Η παρακάτω εικόνα είναι μία από τις εικόνες που δυσκολεύεται το μοντέλο να την ταξινομήσει σωστά με ένα fully-connected στρώμα αλλά όταν έβαλα δύο τοτε βελτιώθηκε κατα πολύ το ποσοστό σωστής ταξινόμησης.

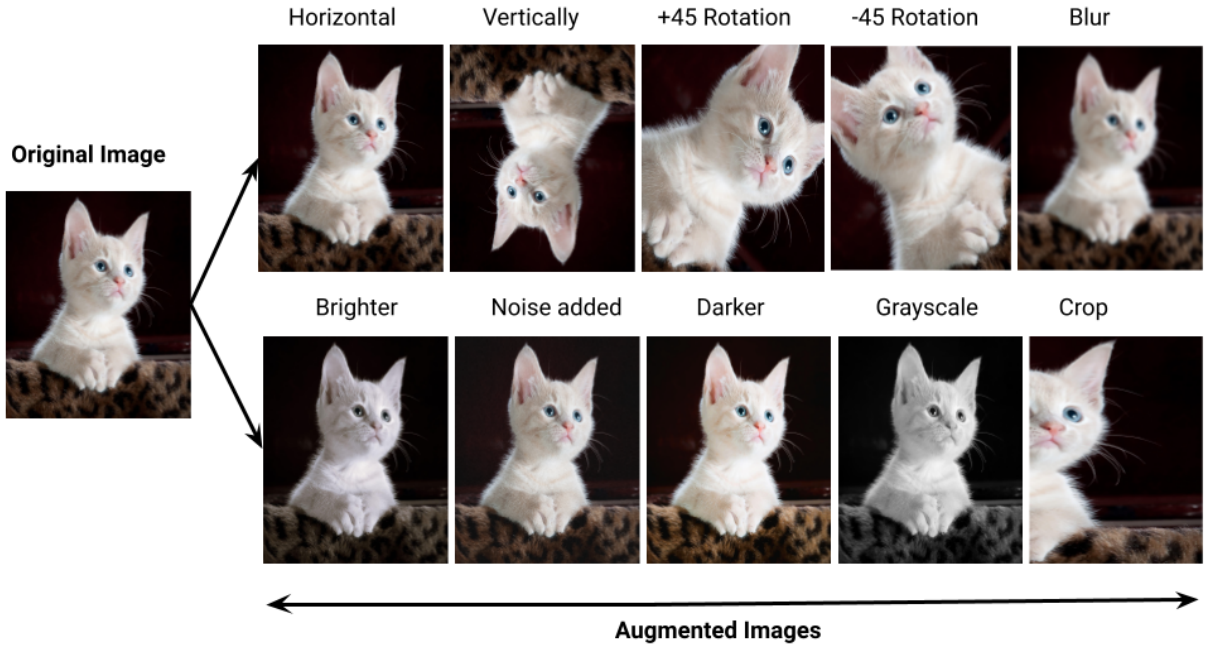
Εικ. 2.9 Παράδειγμα εικόνας σκύλου

Κεφάλαιο 3ο: Επαύξηση των δεδομένων

3.1 Differentiable Automatic Data Augmentation - DADA

Αποφάσισα λοιπόν να δοκιμάσω το μοντελο **D**ifferentiable **A**utomatic **D**ata **A**ugmentation. Αυτο το μοντέλο δημοσιεύτηκε το 2020, απο τους Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M. Robertson και Yongxin Yang. [78]

Η βασική ιδέα που υλοποιεί το DADA είναι ότι προσπαθεί να δημιουργήσει μια αυτοματοποιημένη διαδικασία για την ενίσχυση των δεδομένων. Συνήθως, όταν μιλάμε για ενίσχυση των δεδομένων (**data augmentation**), αναφερόμαστε στην τεχνική που γίνεται κατα την προετοιμασία των δεδομένων η οποία είναι η εφαρμογή διαφόρων μετατροπών και μετασχηματισμών στις εικόνες. Αυτο βοηθάει στο να έχουμε περισσότερα και πιο πολύπλοκα δείγματα, κάνοντας το μοντέλο μας πιο γενικό και αποτρέποντας το να overfitting. [79]



Εικ. 3.1 Παράδειγμα Data Augmentation

Το DADA προσπαθεί να πάει ένα βήμα παραπέρα την παραπάνω διαδικασία. Θεωρεί ότι οι μετατροπές που θα γίνουν στις εικόνες είναι κομμάτι της αρχιτεκτονικής του νευρωνικού δικτύου και άρα οι παράμετροι των μετατροπών μπορούν να εκπαιδευτούν όπως εκπαιδεύονται τα βάρη του νευρωνικού. Έτσι μέσω της όλης διαδικασίας εκπαίδευσης του μοντέλου δεν βελτιώνεται και ανακαλύπτει μοτίβα μόνο για την σωστή ταξινόμηση αλλά και για την σωστή παραμετροποίηση των τροποποιήσεων που πρέπει να γίνουν πριν για την επίτευξη καλύτερης απόδοσης του μοντέλου. Με αυτόν τον τρόπο το DADA μαθαίνει απευθείας τις καλύτερες τροποποιήσεις για τα δεδομένα που πρέπει να επεξεργαστεί. [78]

3.1.1 Μονάδες και Παράμετροι Ενισχυσης Δεδομένων

Οι μονάδες ενισχυσης δεδομένων (**data augmentation modules**) είναι μονάδες που υπάρχουν μέσα στο μοντέλο και στόχος τους είναι η εφαρμογή των μετατροπών στα δεδομένα εισόδου. Έχουν σχεδιαστεί έτσι ώστε να μιμούνται τις μετατροπές που θα χρησιμοποιούσε και ένας άνθρωπος, δηλαδή δεν θα υλοποιήσουν κάποια πολύπλοκη μετατροπή. Η καινοτομία τους βρίσκεται στις παραμέτρους των μετατροπών, που επιτρέπουν την εκπαίδευσή τους. Αυτές οι μονάδες εισάγονται μέσα στην αρχιτεκτονική του μοντέλου. Συνήθως τοποθετούνται ανάμεσα στο στρώμα εισόδου και το πρώτο στρώμα των κρυφών νευρώνων που λειτουργεί ώς εξαγωγέας χαρακτηριστικών. Τα τοποθετούμε εκεί γιατι με αυτόν τον τρόπο οι μετατροπές που έχουν γίνει θα επηρεάσουν την εκπαίδευση του μοντέλου από την αρχή. [78]

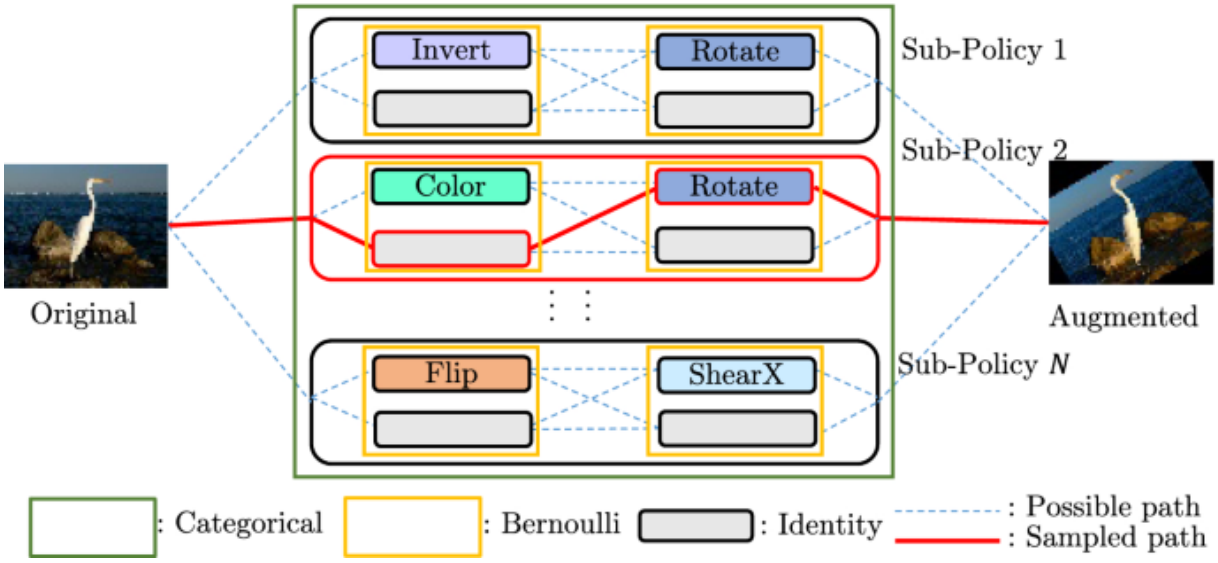
Οι παράμετροι της ενισχυσης δεδομένων είναι οι μαθηματικοί αριθμοί που έχουν οι μετατροπές. Αυτές οι παράμετροι ελέγχουν την συχνότητα που θα πραγματοποιηθεί μια μετατροπή καθώς και σε ποιο βαθμό θα πραγματοποιηθεί. Κατα την διαδικασία της εκπαιδευσης αυτοι οι αριθμοι αντιμετωπίζονται σαν να είναι παράμετροι του μοντέλου και άρα αλλάζουν βάση του ρυθμού μάθησης και τους αλγόριθμους βελτιστοποίησης με στόχο την μείωση της συνάρτησης απώλειας. Στόχος αυτών των παραμέτρων δεν είναι αλλος απο το να πραγματοποιήσουν όσο το δυνατόν περισσότερες μετατροπες, έτσι ώστε το μοντέλο να εκπαιδευτεί σε περισσότερα και πιο δύσκολα δεδομένα και να γενικευσει καλύτερα, χωρίς όμως να αλλοιώνει την αρχική εικόνα και να μπερδεύει το μοντέλο. [78]

3.2 Υλοποίηση DADA

Θα μπορούσαμε να ορίσουμε την αναζήτηση των βέλτιστων παραμέτρων για ενίσχυση δεδομένων ώς την αναζήτηση για το βέλτιστο policy function. Αυτό σημαίνει ότι αφού μιλάμε για ένα πρόβλημα που πρέπει να γίνουν αλλαγές οι οποίες έχουν μια συνέχεια μεταξύ τους και η ενέργεια που πραγματοποιείται επηρεάζει και αλλάζει το περιβάλλον, δηλαδή την εικόνα, και ο κύριος στόχος μας είναι η βελτιστοποίηση, τότε θα μπορούσαμε να το χαρακτηρίσουμε σαν ένα πρόβλημα μάθησης με ενίσχυση.

Πράγματι, το DADA είναι εμπνευσμένο από αυτήν αλλα δεν αποτελεί μοντέλο μάθησης με ενίσχυση με την παραδοσιακή έννοια καθώς δεν χρησιμοποιεί κανενα απο τα εργαλεία της. Αντ' αυτού το DADA χρησιμοποιεί την ιδέα της ευρεσης του βέλτιστου policy και του reward των reinforcement learning μοντέλων, δηλαδή την ικανότητα να αλλάζει τις παραμέτρους του, ανάλογα με το τι "παρατηρήθηκε" στο περιβάλλον και πόσο πιο κοντά το έφερε στο σωστό αποτέλεσμα. Το feedback στην περιπτωση με το classification που έχουμε είναι το ποσοστό επιτυχίας μετά τα augmentations. Επίσης χρησιμοποιεί κομμάτια απο την τεχνικη που υπάρχει και στο reinforcement learning, το δίλημμα της εξερεύνησης ή εκμετάλλευσης (**explorations vs exploitation dilemma**).[80] Το δίλημμα αυτό αναφέρεται σε ένα βασικό δίλημμα στα προβλήματα αποφάσεων που πραγματεύεται το αν όταν παίρνουμε μία απόφαση θα εξερευνήσουμε καινούργιες πιθανότητες ή εάν θα ακολουθήσουμε με βάση τα στοιχεία που ήδη έχουμε συλλέξει. Στην μάθηση με ενίσχυση εάν δεν υπάρχει αυτό το δίλημμα τότε πολύ πιθανόν να φτάσουμε σε μία sub-optimal λύση λόγω των επιλογών που θα γίνουν στην αρχή. Συνήθως ξεκινάμε έχοντας την παράμετρο ε, απο το exploration, με πολύ μεγάλο αριθμό έτσι ώστε να ψάχνει διαφορετικά actions και να βλέπει τι δουλεύει και τι όχι και σταδιακά μειώνεται το ε έτσι ώστε να βρεθεί η optimal ακολουθία actions για το πρόβλημα. Το DADA, εξερευνεί όλα τα πιθανά policies και την επίδραση τους στο μοντέλο και στην συνέχεια επιλέγει αυτή με την καλύτερη απόδοση.[81]

Το DADA χρησιμοποιεί έναν χώρο αναζήτησης (**search space**) που το policy έχει διάφορα sub-policies και το κάθε sub policy έχει δύο transformations με τις παραμέτρους τους (πόσο πιθανό είναι να γίνει και πόσο ισχυρό θα είναι). Αρχικά δημιουργεί μία ενιαία κατανομη με τον παραπάνω search space. Χρησιμοποιούμε Κατηγορική κατανομή για να επιλέξουμε απο τα sub-policies και κατανομή Bernoulli στα επιμέρους sub policies για να επιλέξουμε augmentation.

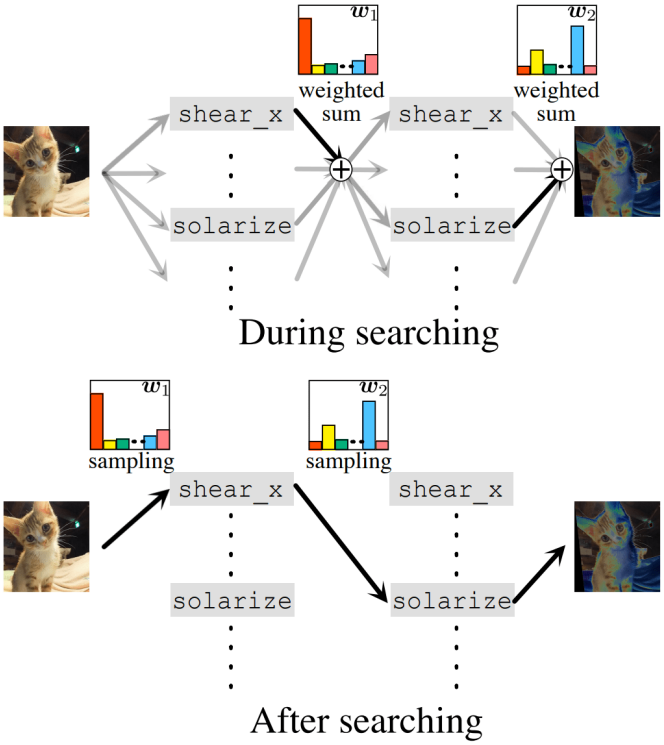


Εικ. 3.2 Η δομή του search space του DADA

Με αυτόν τον τρόπο, το πρόβλημα της εύρεσης του βελτιστου policy γίνεται ένα πρόβλημα εκτίμησης κλισης Monte Carlo. Το πρόβλημα εκτίμησης κλίσης Monte Carlo είναι η προσέγγιση της κλίσης μιάς συνάρτησης πολλών μεταβλητών, το policy στην συγκεκριμένη περίπτωση, με την μέθοδος της τυχαίας δειγματοληψίας Monte Carlo. Στην δειγματοληψία Monte Carlo, επιλέγεται τυχαία από το δειγματοχώρο ένα δείγμα και υπολογίζεται η τιμή του. Τέλος αυτές οι τιμές χρησιμοποιούνται για την δημιουργία της κλίσης.

Παρόλα αυτά, καμία από τις δύο παραπάνω κατανομές δεν είναι διαφοροποιήσιμες (differentiable). Για να μπορέσουν να εκπαιδευτούν οι παράμετροι, το DADA υλοποιεί και χρησιμοποιεί μια τεχνική που ονομάζεται Gumbel-Softmax εκτιμητής κλίσης (Gumbel-Softmax gradient estimator). Αυτη η τεχνική χρησιμοποιείται συνήθως σε προβλήματα μάθησης με ενίσχυση και συνδυάζει την κατανομή Gumbel με την συνάρτηση Softmax. Στόχος της είναι να προσεγγίζει μια διακριτή δειγματοληψία, όπως την επιλογή μιας ενέργειας, με διαφοροποιήσιμο τρόπο. Αρχικά έχουμε τα one-hot vectors που αναπαριστούν τα policies και για κάθε ένα από αυτά έχουμε κάποιες συνεχείς μη κανονικοποιημένες πιθανότητες που αναπαριστούν πόσο καλά τα πάει το κάθε policy στο μοντέλο. Στην συνέχεια για να μπορέσουμε να μετατρέψουμε την διαδικασία σε διαφοροποιήσιμη, εισάγουμε τον θόρυβο Gumbel με βάση την κατανομή Gumbel η οποία είναι συνεχής. Με αυτόν τον τρόπο εισάγουμε την έννοια της τυχαιότητας στην επιλογή. Στην συνέχεια εφαρμόζουμε την συνάρτηση softmax στις πιθανότητες. Η συνάρτηση μετατρέπει τις πιθανότητες σε μια καινούργια συνεχή κατανομή πιθανοτήτων η οποία είναι διαφοροποιήσιμη. Με αυτόν τον τρόπο μπορεί να υπολογιστεί και η κλίση για τις παραπάνω μεταβλητές αφου πλέον είναι και αυτές διαφοροποιήσιμες. Με λίγα λόγια κατα την διαρκεια της εκπαίδευσης, τα δείγματα τα λαμβάνει υπ' όψιν σαν να ήταν διακριτές τιμές και στην συνέχεια υπολογίζει την κλίση σαν να ήταν συνεχείς μεταβλητες. [90]

Επίσης, το DADA αντί να εκπαιδεύεται με βάση την μεγιστοποίηση της τιμής της ακρίβειας εκπαιδεύεται για να μειώσει το ποσοστό λάθους.



Εικ. 3.3 Παράδειγμα τελικού Policy

Ένα ακόμη απο τα θετικά του DADA είναι ότι εκπαιδεύει την ίδια στιγμή και τα policies και τον classifier στα policies, και έτσι γλιτώνει υπολογιστικό κόστος. Ταυτόχρονα όμως καταφέρνει να μειώσει κατα πολύ τον χρόνο εκπαίδευσης του μοντέλου αφου δεν χρειάζεται να περάσουμε τα δεδομένα δύο φορές από το μοντέλο. Η ταυτόχρονη εκπαίδευση των παραμέτρων όμως δεν βοηθάει μόνο σε αυτό το κομμάτι αλλα αυξάνει κατα πολύ και την αποδοτικότητα του μοντέλου. Το μοντέλο μαθαίνει ταυτόχρονα πώς να παράγει καινούργια δεδομένα με περισσότερες πληροφορίες έτσι ώστε να γενικευει καλύτερα. Επίσης μαθαίνοντας μαζί και τα δύο hyperparameters, το μοντέλο καταφέρνει να μειώσει την πιθανότητα του overfitting αφού αποτρέπει το μοντέλο από το να απομνημονεύσει τα δείγματα του train set και το αναγκάζει να προσπαθήσει να μάθει τις συσχετίσεις που έχουν αντ αυτού. [78]

3.3 Ροή DADA

Κατα την διαδικασία εύρεσης του optimal policy, γίνονται πάρα πολλές επαναλήψεις, ετσι ώστε να εκπαιδευτούν οι παράμετροι του augmentation. Η ροή του θα μοιάζει κάπως έτσι:

- **Αρχικοποίηση Μεταβλητών για τις πολιτικές επαύξησης**: Αρχικοποιούνται το μοντέλο , το search space και οι μεταβλητές των πολιτικών που έχουμε ορίσει,
- **Εκπαίδευση**: Εισάγουμε δεδομένα στο μοντέλο και πραγματοποιούνται και τα κατάλληλα augmentations με βάση το current policy που έχει επιλεχθεί στο training set.. Στην συνέχεια, τα δεδομένα περνάνε απο τον classifier και υπολογίζουμε την κλίση της συνάρτησης απώλειας και για τις παραμέτρους του μοντέλου και για τις παραμέτρους του augmentation.
- **Ενημέρωση παραμέτρων**: Αφού έχει γίνει ο υπολογισμός της κλίσης ενημερώνουμε τις παραμέτρους χρησιμοποιώντας κατάβαση δυναμικού (gradient descent) με βάση τις κλίσεις

που βρήκαμε. Κύριος στόχος αυτού του βήματος είναι η βελτίωση των πολιτικών που δοκιμάστηκαν. Επίσης η αλλαγή γίνεται με στόχο την επίτευξη μικρότερης απώλειας και όχι με στόχο την επίτευξη μεγαλύτερης ακρίβειας.

- **Validation set**: Τέλος περνάμε τα δεδομένα από το validation set για καθε policy, με στόχο την αξιολόγηση του μοντέλου στο πόσο καλά γενικεύει τα αποτελέσματα του σε δεδομένα που δεν έχει ξαναδεί.

Το μοντέλο συνεχίζει κάνοντας τα βήματα δύο,τρία και τέσσερα επαναληπτικά για όλη την διάρκεια τον εποχών που έχουμε ορίσει, με στόχο την βελτίωση των πολιτικών και του μοντέλου. Στο τέλος, το policy με το καλύτερο αποτέλεσμα επιλέγεται ώς το τελικό policy. Όπως μπορούμε να καταλάβουμε το μοντέλο για εύρεση του policy λειτουργεί το ίδιο με ένα κανονικό μοντέλο εκπαιδεύεται και στο τέλος θα βγάλει πολλαπλά variations του ίδιου policy.

Σαν έξοδο το policy έχει έξι μορφές, που διαφέρουν ανάλογα με την ποσότητα τον μετατροπών. Τα policy ονομάζονται genotype_105, genotype_5, genotype_10, genotype_15, genotype_20 και genotype_25, και ο αριθμός υποδηλώνει το πόσα ζευγάρια υλοποιούν. Στην συνέχεια θα πρέπει να πάρω το genotype και να μπεί σαν transformation στον classifier.

```
genotype_105: [[('AutoContrast', 0.5, 0.5), ('Sharpness', 0.5, 0.5)], [('TranslateX', 0.5, 0.5), ('Invert', 0.5, 0.5)],
genotype_5: [[('AutoContrast', 0.5, 0.5), ('Sharpness', 0.5, 0.5)], [('TranslateX', 0.5, 0.5), ('Invert', 0.5, 0.5)], [
genotype_10: [[('AutoContrast', 0.5, 0.5), ('Sharpness', 0.5, 0.5)], [('TranslateX', 0.5, 0.5), ('Invert', 0.5, 0.5)], [
genotype_15: [[('AutoContrast', 0.5, 0.5), ('Sharpness', 0.5, 0.5)], [('TranslateX', 0.5, 0.5), ('Invert', 0.5, 0.5)], [
genotype_20: [[('AutoContrast', 0.5, 0.5), ('Sharpness', 0.5, 0.5)], [('TranslateX', 0.5, 0.5), ('Invert', 0.5, 0.5)], [
genotype_25: [[('AutoContrast', 0.5, 0.5), ('Sharpness', 0.5, 0.5)], [('TranslateX', 0.5, 0.5), ('Invert', 0.5, 0.5)], [
```

Εικ. 3.4 Παράδειγμα αποτελέσματος genotypes

Παρόλο που ο όρος γενότυπο (genotype) συνήθως εμφανίζεται σε γενετικούς αλγόριθμους, το DADA δεν χρησιμοποιεί καθόλου γενετικούς αλγορίθμους. Οι γενετικοί αλγόριθμοι είναι αλγόριθμοι που έχουν εμπνευστει απο την διαδικασία της φυσικής επιλογής. Στους γενετικούς αλγόριθμους υπάρχει ένας πληθυσμός ο οποίος μέσω τον επιλογών και των μεταλλάξεων που ενεργούν πάνω του σε κάποια επόμενη γενιά θα βρεθεί η βέλτιστη λύση. Χρησιμοποιείται κυρίως σε προβλήματα βελτιστοποίησης (optimization problems) και ένα παράδειγμα γενετικού αλγόριθμου είναι τα decision trees. [82]

Κεφάλαιο 4ο: Εργαλεία που χρησιμοποίησα

4.1 Python

Για την επίλυση του προβλήματος στηρίχθηκα πάνω στην προγραμματιστική γλώσσα Python και τις διάφορες βιβλιοθήκες (**framework**) που διαθέτει. Η Python είναι μια γλώσσα υψηλού επιπέδου, ευκολη στη κατανόηση, η οποία χρησιμοποιείται σε πολλούς υποτομείς της επιστήμης των υπολογιστών, όπως είναι η ανάπτυξη εφαρμογών στο διαδίκτυο (web development) και η ανάλυση των δεδομένων (data analysis). Προφανώς, η python αποτελεί μία από τις πιό διαδεδομενες και αποτελεσματικές γλώσσες για την μηχανική μάθηση, αφου μπορούμε να έχουμε άμεση αλληλεπίδραση με τον τομέα της ανάλυσης δεδομένων, κάτι που μπορεί να βοηθήσει πάρα πολύ την επιτυχία το μοντέλου που θα επιλέξουμε. Επίσης, η κοινοτητα της είναι πάρα πολύ ενεργή και άρα υπάρχουν πάρα πολλες βιβλιοθήκες και πόροι που μπορούν να βρεθούν στο διαδίκτυο εύκολα.[53]

4.2 Pytorch

Πρόκειται για ένα από τα δύο πιο διαδεδομένα ανοιχτού κώδικα (**open-source**) framework για machine learning της python. Στην πραγματικότητα το pytorch είναι βασισμένο στο Torch, ένα άλλο open source framework γραμμένο στην γλώσσα LUA. [83]

Ένα από τα πιο σημαντικά χαρακτηριστικά του pytorch είναι ότι έχει παρέχει την δυνατότητα να κάνει πολλούς υπολογισμούς τένσορα τους οποίους δεν τους κανει στην CPU αλλα χρησιμοποιεί τους μικροεπεξεργαστές που έχει μέσα η GPU. Αυτοι οι επεξεργαστές παρόλο που δεν είναι ικανοί να κάνουν πολύπλοκους υπολογισμούς, είναι πάρα πολύ γρήγοροι, με αποτέλεσμα να μειώνεται κατα πολύ ο χρόνος εκπαίδευσης του μοντέλου. Επίσης σε αντίθεση με τους ανταγωνιστές της, η βιβλιοθήκη αυτή δίνει μεγάλη ελευθερία στους χρήστες γιατι χρησιμοποιεί έναν δυναμικό τρόπο δημιουργίας των μοντέλων και άρα έχει μεγαλύτερη δυνατότητα να δημιουργηθούν πολύ πιο πολύπλοκα μοντέλα. Αυτό όμως απαιτεί μια μεγαλύτερη περίοδο εκμάθησης της βιβλιοθήκης, αλλα λόγω της μεγάλης κοινοτητας που υπάρχει, μπορεί εύκολα κανείς να βρει οτι πληροφορίες και πηγές θέλει στο διαδίκτυο.

4.2.1 Data Loaders

Για την σωστή προεπεξεργασία και διαχείριση των δεδομένων πριν την εισαγωγή τους στο μοντέλο, μπορούμε να χρησιμοποιήσουμε ένα εργαλείο που ονομάζεται dataloaders.

Οι dataloaders είναι ένα εργαλείο που βοηθάει στην δημιουργία μικρότερων υποσυνόλων στο set δεδομένων (**batches**) ώστε να μπορούν να εισαχθούν σε μικρότερες ποσότητες στο μοντέλο και άρα να μπορούμε να χρησιμοποιήσουμε ακόμα μεγαλύτερα σύνολα δεδομένων. Επίσης ένας dataloader μπορεί και ανακατευει τα δεδομένα έτσι ώστε να βοηθήσουμε το μοντέλο να κάνει καλύτερες γενικεύσεις και όχι με βάση την διάταξη των δεδομένων εισαγωγής. [84]

Κεφάλαιο 5ο: Πειράματα & Αποτελέσματα

5.1 Πειράματα

Έκανα πέντε πειράματα. Όλα με τις ιδιες αρχιτεκτονικές μοντέλων και με τις ίδιες παραμέτρους, οι οποίες ήταν:

- Μοντέλο κατηγοριοποίησης : ResNet-50, με την παρακάτω αρχιτεκτονική (Εικ.5.1), χρησιμοποίησα έναν pretrained στο imagenet ResNet-50, το πάγωσα για να το χρησιμοποιήσω σαν feature extractor, και στην συνέχεια έβαλα δύο fully connected layers ετσι ώστε να ταξινομούνται τα δείγματα σε δύο κλάσεις.
- Δεδομένα: cat-vs-dog
- Batch: 64
- Epoch: 90
- Learning Rate: 0.1
- Learning Rate Scheduler: MultiStepLR
    - milestones = [30, 60, 80]
    - gamma = 0.1
- Optimizer: Adam
- Loss function: Cross-Entropy Loss

```
elif name == 'resnet_pet':
    model = models.resnet50(pretrained=True)
    model.trainable = False
    num_in_features = model.fc.in_features
    model.fc = nn.Sequential(nn.Linear(num_in_features, 256),
                             nn.ReLU(),
                             nn.Linear(256, num_class))
```

Εικ. 5.1 Αρχιτεκτονική ResNet50 που χρησιμοποίησα

Τα δεδομένα τα φόρτωσα με dataloaders και τα χώρισα ώς εξής:

- Train size: 14.998
- Val size: 6.249
- Test size: 3.750
- All images: 24.997

Τα augmentations που χρησιμοποίησα είναι αυτά που χρησιμοποίησαν και στο DADA οι δημιουργοί του δηλαδή τα:

'ShearX','ShearY','TranslateX','TranslateY','Rotate','AutoContrast','Invert','Equalize','Solarize','Posterize','Contrast','Color','Brightness','Sharpness','Cutout'

Χρησιμοποίησα αυτά γιατί θεωρώ ότι καλύπτουν ένα καλό ευρος απο augmentation σε πολλά επίπεδα της εικόνας και ταυτόχρονα καλύπτουν τα περισσότερα βασικά augmentations που θα εφαρμοστούν σε μία εικόνα.

5.1.1 ResNet_None

Το πρώτο πείραμα που έκανα είναι να τρέξω τον classifier χωρις καμία επαυξηση δεδομένων. Οι παράμετροι που είχα στο αρχείο configuration είναι αυτοι:

```yaml
model:
  type: resnet_pet
dataset: cat-vs-dog
aug: default
cutout: 0
batch: 64
epoch: 90
lr: 0.1
lr_schedule:
  type: 'resnet'
optimizer:
  type: Adam
```

Εικ. 5.2 ResNet-None Config File

Και η εντολή που χρησιμοποίησα ήταν αυτή:

```
echo "--------------------------Version with no augmentations--------------------------"
# TITAN
GPUS=0
SEED=0
DATASET=cat-vs-dog
CONF=confs/resnet50_pet_none.yaml
SAVE=weights/`basename ${CONF} .yaml`_${GENOTYPE}_${DATASET}_${SEED}/test.pth
CUDA_VISIBLE_DEVICES=${GPUS} python FastAutoAugment/train.py -c ${CONF} --dataset ${DATASET} --save ${SAVE} --seed ${SEED}   --only-eval
```

Εικ. 5.3 ResNet-None εντολή εκπαίδευσης - evaluation

Η παράμετρος -c είναι για να πάρει το config απο το path του CONF, η παράμετρος –save είναι για την αποθήκευση των βαρών που θα βρεθούν και η παράμετρος –only-eval είναι έτσι ώστε να φορτώσει τα βάρη που θα βρεθούν, αν βρει, και να τρέξει μία εποχή με τα καλύτερα σωσμένα βάρη έτσι ώστε να δούμε την καλύτερη απόδοση του. Αν δεν βρεθούν βάρη, τότε θα τρέξει η διαδικασία εκπαίδευσης του μοντέλου.

5.1.2 ResNet_Manual_Aug

Το δεύτερο πείραμα που έκανα ήταν στο ίδιο μοντέλο με manual preprocessing με ένα μικρό αριθμό augmentations που εγώ έβαλα με το χέρι.

```yaml
model:
  type: resnet_pet
dataset: cat-vs-dog
aug: manual_pet
cutout: 0
batch: 64
epoch: 90
lr: 0.1
lr_schedule:
  type: 'resnet'
optimizer:
  type: Adam
```

Εικ. 5.4 ResNet-Manual-Aug Config File

Μπορούμε να παρατηρήσουμε ότι στην παράμετρο aug αντι για default έχουμε το “manual_pet”. Αυτο είναι αυτές οι γραμμές στο αρχείο **data.py** όπου γίνεται και η φορτωση και προεπεξεργασία των δεδομένων:

```python
elif C.get()['aug'] == 'manual_pet':
    transforms_pet = [
        [('Brightness', 0.2, 7), ('Contrast', 0.2, 6)],
        [('Sharpness', 0.2, 9), ('Brightness', 0.2, 9)],
        [('Sharpness', 0.2, 1), ('Sharpness', 0.2, 3)],
        [('Solarize', 0.2, 5), ('AutoContrast', 0.2, 3)],
        [('ShearX', 0.2, 9), ('ShearY', 0.2, 9)],
        [('Sharpness', 0.2, 1), ('Brightness', 0.2, 3)],
        [('Solarize', 0.2, 5), ('Color', 0.2, 3)]
    ]
    transform_train.transforms.insert(0, Augmentation(transforms_pet))
```

Εικ. 5.5 ResNet-Manual-Aug Transformations

Παρατηρούμε ότι σε αυτό το πείραμα δεν χρησιμοποίησα εύρος τιμών για το τις πιθανές τιμές που θα πάρει το augmentation αλλά έκανα hard-code κάποιους αριθμούς.

Για να το τρέξουμε η εντολή θα μοιάζει κάπως έτσι:

```
echo "-------------------------Version with some manual random augmentations-------------------------"
# TITAN
GPUS=0
SEED=0
DATASET=cat-vs-dog
CONF=confs/resnet50_pet_manual_aug.yaml
SAVE=weights/`basename ${CONF} .yaml`_${GENOTYPE}_${DATASET}_${SEED}/test.pth
CUDA_VISIBLE_DEVICES=${GPUS} python FastAutoAugment/train.py -c ${CONF} --dataset ${DATASET} --save ${SAVE} --seed ${SEED} --only-eval
```

Εικ. 5.6 ResNet-Manual-Aug εντολή εκπαίδευσης - evaluation

5.1.3 ResNet_Manual_16

Το επόμενο πείραμα ήταν να βάλω πάλι manual τιμές αλλα αυτήν την φορά πήρα και τα 15 προτεινόμενα augmentations και έβαλα και εύρος τιμής. Αυτο σημαινει οτι κάθε φορά που θα έτρεχαν θα είχαν ένα ποσοστό να τρέξουν και θα έπαιρναν μια τυχαία τιμή από το εύρος. Το config είναι κάπως έτσι:

```
model:
  type: resnet_pet
dataset: cat-vs-dog
aug: manual_pet_105
cutout: 0
batch: 64
epoch: 90
lr: 0.1
lr_schedule:
  type: 'resnet'
optimizer:
  type: Adam
```

Εικ. 5.7 ResNet-Manual-16 Config File

Τα augmentations όλα έχουν probability 0,2 και ένα εύρος τιμών μοιάζουν κάπως έτσι:

```
elif C.get()['aug'] == 'manual_pet_105':
    transforms_pet = [
        [('ShearX', 0.2, (0.1, 0.25)),
         ('ShearY', 0.2, (0.1, 0.25))],
        [('TranslateX', 0.2, (-15, 15)),
         ('TranslateY', 0.2, (-15, 15))],
        [('Rotate', 0.2, (-20, 20)),
         ('AutoContrast', 0.2, 0.4)],
        [('Invert', 0.2, 0.4),
         ('Equalize', 0.2, 0.4)],
        [('Solarize', 0.2, 100),
         ('Posterize', 0.2, 4)],
        [('Contrast', 0.2, (0.8, 1.2)),
         ('Color', 0.2, (0.8, 1.2))],
        [('Brightness', 0.2, (0.8, 1.2)),
         ('Sharpness', 0.2, (0.8, 1.2))],
        [('Cutout', 0.2, (0.05, 0.15))]
    ]

    transform_train.transforms.insert(0, Augmentation105(transforms_pet))
```

Εικ. 5.8 ResNet-Manual-16 Transformations

Τέλος η εντολή για να τρέξει είναι η εξής:

```
# TITAN
GPUS=0
SEED=0
DATASET=cat-vs-dog
CONF=confs/resnet50_pet_manual_16_aug.yaml
SAVE=weights/`basename ${CONF} .yaml`_${GENOTYPE}_${DATASET}_${SEED}/test.pth
CUDA_VISIBLE_DEVICES=${GPUS} python FastAutoAugment/train.py -c ${CONF} --dataset ${DATASET} --save ${SAVE} --seed ${SEED} --only-eval
```

Εικ. 5.9 ResNet-Manual-16 εντολή εκτέλεσης

5.1.4 ResNet_Pet

Σε αυτό το πείραμα έτρεξα τον classifier με το policy (genotype) που προέκυψε μετά από το DADA. Δοκίμασα και τα έξι policy και βρήκα ότι το καλύτερο το είχα με το genotype 105, το οποίο έχει την μορφή:

```
pets = augment_pets = [[('TranslateX', 1.0, 0.0), ('Cutout', 0.25853341817855835, 0.45325741171836853)],
 ('Rotate', 0.6681274175643921, 0.0), ('Color', 0.4713013470172882, 0.09959087520837784)],
 [('Invert', 0.4519963562488556, 0.23395971953868866), ('Brightness', 0.5427407026290894, 0.5048942565917969)],
 [('Rotate', 0.685630202293396, 0.0), ('Brightness', 0.5903955101966858, 0.0)],
 [('ShearX', 0.6178597807884216, 0.0), ('TranslateX', 0.6691582798957825, 0.0)],
 [('ShearY', 0.4490097463130951, 0.14688481390476227), ('Solarize', 0.5363088846206665, 0.03307367488741875)],
 [('TranslateX', 1.0, 0.0), ('Color', 0.0, 0.5704734921455383)],
 [('Rotate', 0.48171672224998474, 0.19985082745552063), ('Invert', 1.0, 0.0)],
 [('TranslateX', 0.4294084310531616, 0.6250595450401306), ('Sharpness', 1.0, 0.17321433126926422)],
 [('Brightness', 1.0, 0.0), ('Sharpness', 0.23379294574260712, 0.3953215777873993)],
 [('TranslateX', 0.46468091011047363, 0.07018885761499405), ('TranslateY', 0.0, 0.4345453679561615)],
 [('AutoContrast', 0.9601126313209534, 0.017278339713811874), ('Color', 0.47007593512535095, 0.2681193947792053)],
 [('TranslateX', 1.0, 0.0), ('Brightness', 0.0, 0.5640402436256409)],
 [('Invert', 1.0, 0.34717366099357605), ('Sharpness', 0.6251532435417175, 0.378818541765213)],
 [('ShearY', 0.0, 0.49571454524993896), ('Brightness', 0.792211234569496, 0.0)],
 [('Color', 0.3098011910915375, 0.6004493832588196), ('Cutout', 1.0, 0.0)],
 [('Solarize', 0.6445430517196655, 0.14781059324741364), ('Brightness', 0.28754445910453796, 0.44469568133354187)],
 [('AutoContrast', 0.6342669725418091, 0.3393114507198334), ('Posterize', 0.1476307064294815, 0.7813881039619446)],
 [('ShearX', 0.001119075226597488, 0.40443193912506104), ('Equalize', 0.0, 0.7752625942230225)],
 [('ShearY', 0.1851600706577301, 0.7463002800941467), ('TranslateX', 0.5906282663345337, 0.2672671377658844)],
 [('Invert', 0.0, 0.5643936395645142), ('Posterize', 0.7747581005096436, 0.0)],
 [('AutoContrast', 0.7103358507156372, 0.0), ('Cutout', 0.5139594078063965, 0.10483657568693161)],
 [('ShearY', 0.44385042786598206, 0.685861349105835), ('AutoContrast', 0.36256420612335205, 0.8825321793556213)],
 [('ShearX', 0.1712750643491745, 0.285391241312027), ('Brightness', 1.0, 0.0)],
 [('Brightness', 0.26814815402030945, 0.6124013066291809), ('Cutout', 0.6749072670936584, 0.19064292311668396)],
 [('Contrast', 0.0, 0.5422906279563904), ('Brightness', 0.44715195894241333, 0.6214552521705627)],
 [('TranslateY', 0.9461399912834167, 0.0), ('Posterize', 0.9404996037483215, 0.0)],
 [('TranslateY', 0.5839270353317261, 0.22358831763267517), ('Contrast', 0.5230967402458191, 0.3824661672115326)],
 [('TranslateY', 0.39883536100387573, 0.9760780334472656), ('AutoContrast', 0.3366332948207855, 0.5608919262886047)],
 [('ShearY', 0.6326450109481812, 0.13954132795333862), ('Cutout', 1.0, 0.2145644575357437)],
 [('Rotate', 0.30208685994148254, 0.41960906982421875), ('Contrast', 0.6759893298149109, 0.5036547780036926)],
 [('Rotate', 0.5089541077613831, 0.22548530995845795), ('Cutout', 0.3701026439666748, 0.5258715748786926)],
 [('ShearY', 0.49445533752441406, 0.41933074593544006), ('TranslateY', 0.18727685511112213, 0.7543393969535828)],
 [('TranslateX', 0.6462928056716919, 0.5598364472389221), ('Solarize', 0.49661073088645935, 0.5784715414047241)],
 [('Invert', 1.0, 0.041735704988241196), ('Color', 0.7628924250602722, 0.1340072602033615)],
 [('AutoContrast', 0.9188946485519409, 0.21994252502918243), ('Brightness', 0.36281564831733704, 0.7421450614929199)],
 [('ShearX', 0.9050196409225464, 0.055310413241386414), ('Solarize', 0.0, 0.6355870962142944)],
 [('TranslateX', 0.348050594329834, 0.5647656321525574), ('Posterize', 0.36704832315444946, 0.8093982338905334)],
 [('Invert', 0.0, 0.6857280135154724), ('Cutout', 1.0, 0.7919554114341736)],
 [('TranslateX', 0.6111941933631897, 0.27800416946411133), ('AutoContrast', 0.0, 0.6468626260757446)],
```

Εικ. 5.10 Genotype-105 (part 1)

```
[('TranslateY', 0.46149730682373047, 0.6151329874992371), ('Solarize', 0.32081082463264465, 0.70698082447052)],
[('ShearX', 0.10545440018177032, 0.7439426183700562), ('Cutout', 0.45046746730804443, 0.765531599521637)],
[('TranslateY', 0.646645724773407, 0.8207805752754211), ('Cutout', 0.4010837972164154, 0.2189123034477234)],
[('Color', 0.2938053011894226, 0.25732356309890747), ('Sharpness', 0.0, 0.7204004526138306)],
[('TranslateY', 0.7623075842857361, 0.4620541036128998), ('Equalize', 0.7817502617835999, 0.6401883959770203)],
[('Equalize', 0.9616318345069885, 0.36528462171554565), ('Posterize', 0.3461883068084717, 1.0)],
[('Color', 1.0, 0.07381792366504669), ('Brightness', 0.0, 0.6325204968452454)],
[('ShearY', 0.45563238859176636, 0.5372594594955444), ('Invert', 0.33930137753486633, 1.0)],
[('Solarize', 0.780267059803009, 0.08899469673633575), ('Color', 0.08886528015136719, 0.7674002647399902)],
[('Equalize', 0.0, 0.4701734781265259), ('Cutout', 0.45463109016418457, 0.3331865072250366)],
[('AutoContrast', 0.6704162359237671, 0.22042988240718842), ('Invert', 0.55265212059021, 0.6221131682395935)],
[('Rotate', 0.8145011067390442, 0.32156437635421753), ('Equalize', 0.4632611572742462, 0.12831294536590576)],
[('Contrast', 0.30935513973236084, 0.35139182209968567), ('Sharpness', 0.5449407696723938, 0.252318799495697)],
[('ShearX', 0.6179948449134827, 0.5588110089302063), ('ShearY', 1.0, 0.5852112770080566)],
[('AutoContrast', 0.579543948173523, 0.6822150349617004), ('Sharpness', 1.0, 0.3174460828304291)],
[('Solarize', 0.1822802722454071, 0.4744413495063782), ('Contrast', 0.6041771769523621, 0.24254098534584045)],
[('ShearX', 0.7666686177253723, 0.5121914148330688), ('Color', 0.0, 0.4245493412017822)],
[('Rotate', 0.7338182926177979, 0.7676923871040344), ('AutoContrast', 0.0, 1.0)],
[('Rotate', 0.5617698431015015, 0.17968061566352844), ('Posterize', 0.4686174690723419, 0.5264806747436523)],
[('ShearY', 0.25325721502304077, 0.9536375403404236), ('Contrast', 1.0, 0.36505448818206787)],
[('TranslateX', 0.6437102556228638, 0.20982609689235687), ('Equalize', 1.0, 0.0)],
[('Equalize', 1.0, 0.00800690520554781), ('Solarize', 0.1636039912700653, 0.7304332852363586)],
[('ShearX', 0.6786504983901978, 0.732772171497345), ('Sharpness', 0.2306768298149109, 1.0)],
[('Equalize', 0.8211025595664978, 0.280110627412796), ('Color', 0.45983242988586426, 0.723436176776886)],
[('ShearX', 0.6785341501235962, 0.43091535568237305), ('TranslateY', 0.6938596963882446, 0.5738105773925781)],
[('AutoContrast', 0.3696613907814026, 0.7317968606948853), ('Equalize', 0.6942258477210999, 1.0)],
[('ShearY', 1.0, 0.41372397541999817), ('Posterize', 0.3244253396987915, 0.6755602955818176)],
[('TranslateY', 0.4770716428756714, 0.18641220033168793), ('Sharpness', 0.6888978481292725, 0.1603122055530548)],
[('Posterize', 1.0, 0.4602999687194824), ('Sharpness', 0.575628936290741, 0.655129015457092)],
[('TranslateX', 0.866016149520874, 0.3112044334411621), ('Rotate', 0.4021385908126831, 0.4261559545993805)],
[('TranslateY', 0.7545305490493774, 0.6602147221565247), ('Rotate', 0.34193578362464905, 0.7504627108573914)],
[('Sharpness', 0.6680079698562622, 0.4251497685909271), ('Cutout', 0.6108381152153015, 0.8058568835258484)],
[('TranslateX', 0.7612582445144653, 0.5497259497642517), ('Contrast', 1.0, 0.30741649866104126)],
[('Invert', 0.7901217937469482, 0.8141630291938782), ('Equalize', 0.6045821905136108, 0.8875864744186401)],
[('Contrast', 0.31384533643722534, 0.6387088298797607), ('Color', 0.18555063009262085, 0.7740585207939148)],
[('ShearX', 0.2881835401058197, 0.7265741229057312), ('Invert', 0.7664104700088501, 0.7822320461273193)],
[('Invert', 0.5991548895835876, 0.7997307181358337), ('Contrast', 0.0, 0.6958823204040527)],
[('Posterize', 0.6612584590911865, 0.28317293524742126), ('Brightness', 1.0, 0.0)],
[('TranslateY', 0.5428056120872498, 0.41568368673324585), ('Color', 0.975965917104431, 0.47479212284088135)],
```

Εικ. 5.11 Genotype-105 (part 2)

```
79  [('TranslateY', 0.5428056120872498, 0.41568368673324585), ('Color', 0.9759659171104431, 0.47479212284088135)],
80  [('ShearY', 0.5123565196990967, 0.7710933089256287), ('Rotate', 0.5464829206466675, 0.6727776527404785)],
81  [('ShearX', 0.7961742281913757, 0.4676613211631775), ('Posterize', 0.5614081025123596, 0.7473176121711731)],
82  [('AutoContrast', 0.7002123594284058, 0.6993647217750549), ('Contrast', 0.0, 0.7025669813156128)],
83  [('AutoContrast', 0.5268998146057129, 0.5422429442405701), ('Solarize', 0.42168325185775757, 0.6137245297431946)],
84  [('TranslateX', 0.5206902623176575, 1.0), ('Invert', 0.0, 0.820459127426147 5)],
85  [('Solarize', 0.5998654961585999, 0.3833093047142029), ('Cutout', 0.4866139888763428, 0.8870729804039001)],
86  [('ShearY', 0.8433416485786438, 0.9357380867004395), ('Equalize', 0.4940067231655121, 0.9639654755592346)],
87  [('Posterize', 0.33502769470214844, 0.8089266419410706), ('Contrast', 0.5782940983772278, 0.5413616895675659)],
88  [('TranslateY', 0.11877474188804626, 0.5403563976287842), ('Brightness', 0.33954915404319763, 0.7540316581726074)],
89  [('ShearY', 0.19222703576087952, 1.0), ('Sharpness', 0.7066844369888306, 0.4492713510990143)],
90  [('ShearX', 0.7650814652442932, 0.3703819811344147), ('Rotate', 0.7514920234680176, 0.29676803946495056)],
91  [('Rotate', 0.3960924446582794, 0.9882978200912476), ('Solarize', 0.6294889450073242, 0.9104700684547424)],
92  [('Solarize', 0.795807957649231, 0.6166693568229675), ('Sharpness', 0.22724345326423645, 1.0)],
93  [('Contrast', 0.4274555444717407, 0.8510411977767944), ('Cutout', 0.3554466962814331, 0.9444129467010498)],
94  [('Equalize', 0.0, 0.9223722219467163), ('Contrast', 0.7521807551383972, 0.8550173044204712)],
95  [('Equalize', 0.3446041941642761, 0.6948807835578918), ('Sharpness', 0.6538056135177612, 0.7804533729553222)],
96  [('Solarize', 0.3682363033294678, 0.6415891647338867), ('Posterize', 0.5486125349998474, 0.6131364703178406)],
97  [('Rotate', 0.7273991107940674, 0.7370281219482422), ('Sharpness', 0.41577088832855225, 0.9185465574264526)],
98  [('Equalize', 0.3039906919002533, 1.0), ('Brightness', 0.6532875299453735, 0.6612482070922852)],
99  [('TranslateY', 0.5717595219612122, 0.9974934458732605), ('Invert', 0.17359979450702667, 0.8569129705429077)],
100 [('ShearX', 0.0, 0.8131055235862732), ('AutoContrast', 0.3639216125011444, 0.5495970845222473)],
101 [('Invert', 0.42981240153312683, 0.45845869183540344), ('Solarize', 0.34817108511924744, 0.5835264325141907)],
102 [('Posterize', 0.5566455125808716, 0.9642588496208191), ('Cutout', 0.37985479831695557, 1.0)],
103 [('Posterize', 0.0938010886311531, 1.0), ('Color', 0.40751656889915466, 1.0)],
104 [('ShearX', 0.32394155859947205, 0.8099690079689026), ('Contrast', 0.3650970757007599, 1.0)],
105 [('ShearY', 0.09680172801017761, 1.0), ('Color', 0.44890281558036804, 1.0)]]
```

Εικ. 5.12 Genotype-105 (part 3)

Όπως μπορούμε να παρατηρήσουμε και στα παραπάνω screenshots πρόκειται για 105 σειρές απο ζεύγη μετατροπών, οι οποίες έχουν γίνει optimised με μεγάλο αριθμό δεκαδικών στοιχείων έτσι ώστε να πετύχουν ακόμα μεγαλύτερο αποτέλεσμα.

Η εντολή για να τρέξει είναι όμοια με τα παραπάνω και το config file είναι:

```
model:
  type: resnet_pet
dataset: cat-vs-dog
aug: d
genotype: augment_pets
cutout: 0
batch: 64
epoch: 90
lr: 0.1
lr_schedule:
  type: 'resnet'
optimizer:
  type: Adam
```

Εικ. 5.13 ResNet_Pet Config File

5.1.5 ResNet_Pet_Autoaug_Extend

Το τελευταίο πείραμα που έκανα ήταν να τρέξω τον classifier με ένα ήδη policy που είχαν βρει μεσω του DADA οι δημιουργοί του, το έκαναν optimize οργανώνοντας το σε ομάδες και είχε ήδη εκπαιδευτεί για παρόμοιο task . Το συγκεκριμένο είχε βρεθεί στο cifar-10 dataset που πρόκειται για ένα dataset με εικόνες που ανήκουν σε 10 διαφορετικές κλάσεις. Θεωρείται ένα από τα benchmark σε ότι αφορά ταξινόμηση εικόνων και έχει χρησιμοποιηθεί από πάρα πολλά papers για την αξιολόγηση των αποτελεσμάτων των μοντέλων τους. Οι κλάσεις που διαθέτει είναι: αεροπλάνο, αυτοκίνητο, πουλί, γάτα ,ελάφι, σκύλος, βάτραχος, άλογο, πλοίο, φορτηγό. [85]

```
('Invert', 0.1, 7), ('Contrast', 0.2, 6)],
('Rotate', 0.7, 2), ('TranslateXAbs', 0.3, 9)],
('Sharpness', 0.8, 1), ('Sharpness', 0.9, 3)],
('ShearY', 0.5, 8), ('TranslateYAbs', 0.7, 9)],
('AutoContrast', 0.5, 8), ('Equalize', 0.9, 2)]]
_1 = [
('Solarize', 0.4, 5), ('AutoContrast', 0.9, 3)],
('TranslateYAbs', 0.9, 9), ('TranslateYAbs', 0.7, 9)],
('AutoContrast', 0.9, 2), ('Solarize', 0.8, 3)],
('Equalize', 0.8, 8), ('Invert', 0.1, 3)],
('TranslateYAbs', 0.7, 9), ('AutoContrast', 0.9, 1)]]
_2 = [
('Solarize', 0.4, 5), ('AutoContrast', 0.0, 2)],
('TranslateYAbs', 0.7, 9), ('TranslateYAbs', 0.7, 9)],
('AutoContrast', 0.9, 0), ('Solarize', 0.4, 3)],
('Equalize', 0.7, 5), ('Invert', 0.1, 3)],
('TranslateYAbs', 0.7, 9), ('TranslateYAbs', 0.7, 9)]]
_3 = [
('Solarize', 0.4, 5), ('AutoContrast', 0.9, 1)],
('TranslateYAbs', 0.8, 9), ('TranslateYAbs', 0.9, 9)],
('AutoContrast', 0.8, 0), ('TranslateYAbs', 0.7, 9)],
('TranslateYAbs', 0.2, 7), ('Color', 0.9, 6)],
('Equalize', 0.7, 6), ('Color', 0.4, 9)]]
_0 = [
('ShearY', 0.2, 7), ('Posterize2', 0.3, 7)],
('Color', 0.4, 3), ('Brightness', 0.6, 7)],
('Sharpness', 0.3, 9), ('Brightness', 0.7, 9)],
('Equalize', 0.6, 5), ('Equalize', 0.5, 1)],
('Contrast', 0.6, 7), ('Sharpness', 0.6, 5)]]
```

```
    [('Invert', 0.1, 7), ('Contrast', 0.2, 6)],
    [('Rotate', 0.7, 2), ('TranslateXAbs', 0.3, 9)],
    [('Sharpness', 0.8, 1), ('Sharpness', 0.9, 3)],
    [('ShearY', 0.5, 8), ('TranslateYAbs', 0.7, 9)],
    [('AutoContrast', 0.5, 8), ('Equalize', 0.9, 2)]]
exp0_1 = [
    [('Solarize', 0.4, 5), ('AutoContrast', 0.9, 3)],
    [('TranslateYAbs', 0.9, 9), ('TranslateYAbs', 0.7, 9)],
    [('AutoContrast', 0.9, 2), ('Solarize', 0.8, 3)],
    [('Equalize', 0.8, 8), ('Invert', 0.1, 3)],
    [('TranslateYAbs', 0.7, 9), ('AutoContrast', 0.9, 1)]]
exp0_2 = [
    [('Solarize', 0.4, 5), ('AutoContrast', 0.0, 2)],
    [('TranslateYAbs', 0.7, 9), ('TranslateYAbs', 0.7, 9)],
    [('AutoContrast', 0.9, 0), ('Solarize', 0.4, 3)],
    [('Equalize', 0.7, 5), ('Invert', 0.1, 3)],
    [('TranslateYAbs', 0.7, 9), ('TranslateYAbs', 0.7, 9)]]
exp0_3 = [
    [('Solarize', 0.4, 5), ('AutoContrast', 0.9, 1)],
    [('TranslateYAbs', 0.8, 9), ('TranslateYAbs', 0.9, 9)],
    [('AutoContrast', 0.8, 0), ('TranslateYAbs', 0.7, 9)],
    [('TranslateYAbs', 0.2, 7), ('Color', 0.9, 6)],
    [('Equalize', 0.7, 6), ('Color', 0.4, 9)]]
exp1_0 = [
    [('ShearY', 0.2, 7), ('Posterize2', 0.3, 7)],
    [('Color', 0.4, 3), ('Brightness', 0.6, 7)],
    [('Sharpness', 0.3, 9), ('Brightness', 0.7, 9)],
    [('Equalize', 0.6, 5), ('Equalize', 0.5, 1)],
    [('Contrast', 0.6, 7), ('Sharpness', 0.6, 5)]]
```

Εικ. 5.14 Genotype optimised for Cifar-10 dataset(part 1)

```
p2_0 = [
  [('Color', 0.7, 7), ('TranslateXAbs', 0.5, 8)],
  [('Equalize', 0.3, 7), ('AutoContrast', 0.4, 8)],
  [('TranslateYAbs', 0.4, 3), ('Sharpness', 0.2, 6)],
  [('Brightness', 0.9, 6), ('Color', 0.2, 8)],
  [('Solarize', 0.5, 2), ('Invert', 0.0, 3)]]
p2_1 = [
  [('AutoContrast', 0.1, 5), ('Brightness', 0.0, 0)],
  [('CutoutAbs', 0.2, 4), ('Equalize', 0.1, 1)],
  [('Equalize', 0.7, 7), ('AutoContrast', 0.6, 4)],
  [('Color', 0.1, 8), ('ShearY', 0.2, 3)],
  [('ShearY', 0.4, 2), ('Rotate', 0.7, 0)]]
p2_2 = [
  [('ShearY', 0.1, 3), ('AutoContrast', 0.9, 5)],
  [('TranslateYAbs', 0.3, 6), ('CutoutAbs', 0.3, 3)],
  [('Equalize', 0.5, 0), ('Solarize', 0.6, 6)],
  [('AutoContrast', 0.3, 5), ('Rotate', 0.2, 7)],
  [('Equalize', 0.8, 2), ('Invert', 0.4, 0)]]
p2_3 = [
  [('Equalize', 0.9, 5), ('Color', 0.7, 0)],
  [('Equalize', 0.1, 1), ('ShearY', 0.1, 3)],
  [('AutoContrast', 0.7, 3), ('Equalize', 0.7, 0)],
  [('Brightness', 0.5, 1), ('Contrast', 0.1, 7)],
  [('Contrast', 0.1, 4), ('Solarize', 0.6, 5)]]
p2_4 = [
  [('Solarize', 0.2, 3), ('ShearX', 0.0, 0)],
  [('TranslateXAbs', 0.3, 0), ('TranslateXAbs', 0.6, 0)],
  [('Equalize', 0.5, 9), ('TranslateYAbs', 0.6, 7)],
  [('ShearX', 0.1, 0), ('Sharpness', 0.5, 1)],
  [('Equalize', 0.8, 6), ('Invert', 0.3, 6)]]
p2_5 = [
  [('AutoContrast', 0.3, 9), ('CutoutAbs', 0.5, 3)],
  [('ShearX', 0.4, 4), ('AutoContrast', 0.9, 2)],
  [('ShearX', 0.0, 3), ('Posterize2', 0.0, 3)],
  [('Solarize', 0.4, 3), ('Color', 0.2, 4)],
  [('Equalize', 0.1, 4), ('Equalize', 0.7, 6)]]
```

```
exp2_6 = [
    [('Equalize', 0.3, 8), ('AutoContrast', 0.4, 3)],
    [('Solarize', 0.6, 4), ('AutoContrast', 0.7, 6)],
    [('AutoContrast', 0.2, 9), ('Brightness', 0.4, 8)],
    [('Equalize', 0.1, 0), ('Equalize', 0.0, 6)],
    [('Equalize', 0.8, 4), ('Equalize', 0.0, 4)]]
exp2_7 = [
    [('Equalize', 0.5, 5), ('AutoContrast', 0.1, 2)],
    [('Solarize', 0.5, 5), ('AutoContrast', 0.9, 5)],
    [('AutoContrast', 0.6, 1), ('AutoContrast', 0.7, 8)],
    [('Equalize', 0.2, 0), ('AutoContrast', 0.1, 2)],
    [('Equalize', 0.6, 9), ('Equalize', 0.4, 4)]]
exp0s = exp0_0 + exp0_1 + exp0_2 + exp0_3
exp1s = exp1_0 + exp1_1 + exp1_2 + exp1_3 + exp1_4 + exp1_5 + exp1_6
exp2s = exp2_0 + exp2_1 + exp2_2 + exp2_3 + exp2_4 + exp2_5 + exp2_6 + exp2_7

return exp0s + exp1s + exp2s
```

Εικ. 5.15 Genotype optimised for Cifar-10 dataset(part 2)

Οι εντολές για την εκπαίδευση και evaluation καθώς και το config file έχουν μορφή παρόμοια με των παραπάνω πειραμάτων.

5.2 Αποτελέσματα

Τα αποτελέσματα από το πρώτο πείραμα (ResNet_None) ήταν τα εξής:


```
"loss_train": 0.832908185182983,
"loss_valid": 1.0745806952112729,
"loss_test": 0.8542668746948242,
"top1_train": 0.9331597222222222,
"top1_valid": 0.8745167525773195,
"top1_test": 0.8669333333333333,
"top2_train": 1.0,
"top2_valid": 1.0,
"top2_test": 1.0,
"epoch": 0
```


Εικ. 5.16 Αποτέλεσμα ResNet_None

Παρατηρούμε ότι παρόλο που τα πηγαίνει αρκετά καλά έχει περιθώριο για βελτίωση. Χρησιμοποίησα απλό ResNet χωρίς κάποιον άλλον τρόπο που θα μπορούσε να βελτιώσει την απόδοση το όπως την χρήση πολλαπλών classifiers (Ensemble Learning). Οι τρεις πρώτες μετρικές αναφέρονται στο loss που έχουμε στο κάθε set, οι επόμενες τρεις στο accuracy που έχουμε στα set και οι “top2_set” αναφέρονται στο τι ποσοστό επιτυχίας είχαμε στις δύο πρώτες κλάσεις. Για το συγκεκριμένο παράδειγμα είναι άχρηστο αλλα για πιό πολύπλοκα παραδείγματα είναι χρήσιμο καθώς μπορούμε να δούμε σέ τί ποσοστό βρήκε την σωστή κλάση απο τις top X που προέβλεψε.

Το αποτέλεσμα από το δεύτερο πείραμα (ResNet_Manual_Aug) που αφορούσε κάποια manual augmentations ήταν το εξής:

```
"loss_train": 1.5251907838596046,
"loss_valid": 1.9012991483371282,
"loss_test": 1.6400656062920889,
"top1_train": 0.9943910256410257,
"top1_valid": 0.9115657216494846,
"top1_test": 0.9072,
"top2_train": 1.0,
"top2_valid": 1.0,
"top2_test": 1.0,
"epoch": 0
```


Εικ. 5.17 Αποτέλεσμα ResNet_Manual_Aug

Εδώ παρατηρούμε ότι παρόλο που τα πηγαίνει αρκετά καλά στο training set, στο test και στο validation δεν τα έχει πάει τόσο καλά. Σε συνδυασμό με το υψηλό loss και στα τρία set είναι ασφαλές να πούμε ότι είναι παράδειγμα overfitting.

Το αποτέλεσμα από το τρίτο πείραμα (ResNet_Manual_16) που αφορούσε κάποια manual augmentations που έκανα αλλα αυτήν την φορά τα augmentations είχαν εύρος τιμών που θα επαιρναν τυχαία τιμές και είχαν και τα 16 augmentations που θα χρησιμοποιηθούν αργότερα από το μοντέλο DADA ήταν το εξής:


```
"loss_train": 0.10391086387710693,
"loss_valid": 0.29729480019856974,
"loss_test": 0.3616820509011547,
"top1_train": 0.9732905982905983,
"top1_valid": 0.9072164948453608,
"top1_test": 0.9042666666666667,
"top2_train": 1.0,
"top2_valid": 1.0,
"top2_test": 1.0,
"epoch": 0
```


Εικ. 5.18 Αποτέλεσμα ResNet_Manual_16

Εδώ παρατηρούμε ότι παρόλο που τα πηγαίνει υπερβολικά καλά στο training set, στο test και στο validation δεν τα έχει πάει τόσο καλά αλλά έχει βελτίωση από το απλό ResNet_None. Σε αντίθεση επίσης με το ResNet_Manual_Aug παρατηρούμε ότι τα loss έχουν πέσει αισθητά.

Το αποτέλεσμα από το τέταρτο πείραμα (ResNet_Pet) που ήταν αυτό με την χρήση του DADA ήταν το εξής:

```
"loss_train": 0.16274527290390214,
"loss_valid": 0.23393585540584683,
"loss_test": 0.378535828063637,
"top1_train": 0.9674813034188035,
"top1_valid": 0.9194587628865979,
"top1_test": 0.9176,
"top2_train": 1.0,
"top2_valid": 1.0,
"top2_test": 1.0,
"epoch": 0
```


Εικ. 5.19 Αποτέλεσμα ResNet_Pet

Εδώ παρατηρούμε ότι πηγαίνει υπερβολικά καλά στο training set, αλλα και στο test και στο validation τα πηγαίνει αρκετά καλα. Επίσης και οι τιμές του loss και στα τρία set είναι αρκετά ικανοποιητικές.

Το αποτέλεσμα από το πέμπτο πείραμα (ResNet_Pet_Autoaug_Extend) που ήταν και αυτό με την χρήση του DADA αλλα είχε γίνει ήδη optimised απο τους δημιουργούς του DADA και εκπαιδεύτικε στο cifar dataset ήταν το εξής:


```
"loss_train": 0.06534910064317986,
"loss_valid": 0.196047166842468,
"loss_test": 0.21736011981125922,
"top1_train": 0.9751602564102564,
"top1_valid": 0.9246134020618557,
"top1_test": 0.9178666666666667,
"top2_train": 1.0,
"top2_valid": 1.0,
"top2_test": 1.0,
"epoch": 0
```


Εικ. 5.20 Αποτελέσματα ResNet_Pet_Autoaug_Extended

Παρατηρούμε ότι ενώ υπάρχει μια βελτίωση στα loss τα accuracy είναι πάρα πολύ παρόμοια. Λογικό γιατί έχει γίνει optimised για να δίνει τα καλύτερα αποτελέσματα στο cifar-10 οπότε περίμενα ότι θα έχει υψηλά ποσοστά και στο δικό μου dataset.

Αν βάλουμε όλα τα αποτελέσματα σε έναν πίνακα και τα στρογγυλοποιήσουμε στα τρία ψηφία τότε προκύπτει ο παρακάτω πίνακας:

	loss_train	loss_valid	loss_test	top1_train	top1_valid	top1_test	Epoch
None Augment	0,833	1,075	0,854	0,933	0,875	0,867	89
Manual Augment	1,389	2,151	1,395	0,988	0,908	0,905	90
Manual Augment_16	0,104	0,297	0,362	0,973	0,907	0,904	84
Augment-Pets by RL	0,163	0,234	0,379	0,967	0,919	0,918	90
ResNet_Pet_Autoaug_Extend	0,065	0,196	0,217	0,975	0,925	0,918	80

Πίνακας 5.1 Σύγκριση των αποτελεσμάτων των 5 πειραμάτων

Έβαλα και μια έξτρα τελευταία στήλη που περιγράφει σε ποιά εποχή βρέθηκε το συγκεκριμένο αποτέλεσμα. Μπορούμε να καταλάβουμε ότι εάν ορισμένα τα τρέχαμε για παραπάνω αριθμό εποχών πολύ πιθανόν να είχαμε ακόμα καλύτερα αποτελέσματα ενώ άλλα όπως το PesNet_Pet_Autoaug_Extend φαίνεται να έφτασαν πολύ νωρίς στην υψηλότερη τους ακρίβεια οπότε πιθανών αυτή να είναι και η μεγαλύτερη τιμή που μπορεί να φτάσει το accuracy.

Συμπεραίνοντας, η διαφορα της μη χρησης κανενός augmentation με την χρήση γενικά augmentation είναι πολύ μεγάλη. Τώρα σε ότι αφορά το manual και το automatic η διαφορα είναι αισθητά πιο μικρή αλλά υπάρχει και δεδομένου του ότι μπορούν να βελτιωθούν και άλλο τα αποτελέσματα με το DADA, θα μπορούσαμε να μιλάμε για μία ακόμα μεγαλύτερη αυξηση στην διαφορά τους.

Κεφάλαιο 6ο: Συμπεράσματα

Σε αυτήν την πτυχιακή ασχολήθηκα με την χρήση αυτοματοποιημένων μεθόδων επαύξησης δεδομένων για την βελτίωση ταξινομητών. Τα αποτελέσματα ήταν αρκετά ενδιαφέροντα καθώς αποδεικνύουν πόσο σημαντική είναι η χρήση τεχνικών επάξησης και πόσο μπορεί να βελτιώσουν τα μοντέλα, αλλά ταυτόχρονα οτι η αυτόματη επαυξηση μπορεί να είναι πιο αποδοτική από την χειροκίνητη. Πιο συγκεκριμένα το DADA αποτελεί ένα πάρα πολύ αξιόπιστο και αποτελεσματικό μοντέλο, αφου εκτός του ότι βοήθησε το μοντέλο στην ακρίβεια αλλα και στην γενίκευση του, είναι και πάρα πολύ εύκολο να χρησιμοποιηθεί σε πολλά προβλήματα ταξινόμησης ως εργαλείο data augmentation. Η βασική του ιδέα είναι πάρα πολύ ενδιαφέρουσα γεγονός που το καθιστά αρκετά ευκολο και πιθανό να εμφανιστούν και πολλά ακόμα papers που έχουν στηριχθεί σε αυτο ίσως και σε διαφορετικούς τομείς.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]*Getting started-What is Learning?* Available at: https://www.queensu.ca/teachingandlearning/modules/students/04_what_is_learning.html.

[2] MITCHELL, Tom M.; MITCHELL, Tom M. *Machine learning*. New York: McGraw-hill, 1997.

[3] FRADKOV, Alexander L. Early history of machine learning. *IFAC-PapersOnLine*, 2020, 53.2: 1385-1390

[4] Morris, R. (1997) *Deep Blue versus Kasparov: The significance for Artificial Intelligence*. Menlo Park, CA: American Association for Artificial Intelligence

[5] *History of artificial intelligence*. Queensland Brain Institute - University of Queensland. (2019, January 30). https://qbi.uq.edu.au/brain/intelligent-machines/history-artificial-intelligence

[6]*Real-world examples of machine learning (ML)* (no date) *Tableau*. Available at: https://www.tableau.com/learn/articles/machine-learning-examples.

[7]Goncharov, I. (2022) *The role of machine learning in Autonomous Vehicles*, *W&B*. Available at: https://wandb.ai/ivangoncharov/AVs-report/reports/The-Role-Of-Machine-Learning-In-Autonomous-Vehicles--VmlldzoyNTExMDE3.

[8] Burns, E. (2021, March 30). *What is machine learning and why is it important?*. Enterprise AI. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

[9]*What is supervised learning? IBM*. Available at: https://www.ibm.com/topics/supervised-learning

[10]*What is unsupervised learning? IBM*. Available at: https://www.ibm.com/topics/unsupervised-learning

[11]Brownlee, J. (2020) *Introduction to dimensionality reduction for machine learning*, *MachineLearningMastery.com*.Available.at: https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/.

[12]Brownlee, J. (2020) *What is semi-supervised learning*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/what-is-semi-supervised-learning/.

[13]SUTTON, Richard S.; BARTO, Andrew G. *Reinforcement learning: An introduction*. MIT press, 2018.

[14]SILVER, David, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[15]*ML: Underfitting and overfitting* (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/.

[16]Shah, T. (2020) *About train, validation and test sets in machine learning*, *Medium*. Available at: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

[17]MÜLLER, Berndt; REINHARDT, Joachim; STRICKLAND, Michael T. *Neural networks: an introduction*. Springer Science & Business Media, 1995.

[18]Malik, F. (2019) *Neural network layers*, *Medium*. Available at: https://medium.com/fintechexplained/neural-network-layers-75e48d71f392.

[19]Babs, T. (2022) *The Mathematics of Neural Networks*, *Medium*. Available at: https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05.

[20]Sharma, S. (2019) *What the hell is Perceptron?*, *Medium*. Available at: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53.

[21]*Multilayer Perceptron Multilayer Perceptron - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron.

[22]DeepAI (2019) *Feed Forward Neural Network*, *DeepAI*. Available at: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network.

[23]*What are recurrent neural networks? IBM*. Available at: https://www.ibm.com/topics/recurrent-neural-networks.

[24]*What are convolutional neural networks? IBM*. Available at: https://www.ibm.com/topics/convolutional-neural-networks.

[25]Agrawal, Pulkit, et al. "Convolutional Neural Networks Mimic the Hierarchy of Visual Representations in the Human Brain."

[26]Saha, S. (2022) *A comprehensive guide to Convolutional Neural Networks - the eli5 way*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[27]Mikhailiuk, A. (2022) *Convolutional Neural Networks - the essential summary*, *Medium*. Available at: https://towardsdatascience.com/cnn-cheat-sheet-the-essential-summary-for-a-quick-start-58820a14d3b4.

[28]Anh H. Reynolds (2017) *Convolutional Neural Networks (cnns)*, *Anh H. Reynolds*. Available at: https://anhreynolds.com/blogs/cnn.html

[29]Sedghi, Hanie, Vineet Gupta, and Philip M. Long. "The singular values of convolutional layers." *arXiv preprint arXiv:1805.10408* (2018).

[30]Tripathi, Samarth, et al. "Using deep and convolutional neural networks for accurate emotion classification on DEAP data." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. No. 2. 2017.

[31]*Pooling layer Pooling Layer - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/mathematics/pooling-layer.

[32]Ding-Xuan Zhou *et al.* (2020) *Theory of deep convolutional neural networks: Downsampling*, *Neural Networks*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0893608020300204?via%3Dihub.

[33]Brownlee, J. (2019) *A gentle introduction to pooling layers for Convolutional Neural Networks*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[34]Sharma, S. (2022) *Activation functions in neural networks*, *Medium*. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[35]Weisstein, Eric W. "Heaviside step function." *https://mathworld. wolfram. com/* (2002).

[36]Ezeafulukwe, Uzoamaka A., Maslina Darus, and O. Fadipe-Joseph. "On analytic properties of a sigmoid function." *Int. Journal of Mathematics and Computer Science* 13.2 (2018): 171-178.

[37]*Introduction to loss functions* (2023) *DataRobot AI Platform*. Available at: https://www.datarobot.com/blog/introduction-to-loss-functions/.

[38]Pere, C. (2020) *What are loss functions?*, *Medium*. Available at: https://towardsdatascience.com/what-is-loss-function-1e2605aeb904.

[39]Frost, J. (2023) *Mean squared error (MSE)*, *Statistics By Jim*. Available at: https://statisticsbyjim.com/regression/mean-squared-error-mse/.

[40]Stephanie (2020) *Absolute error & mean absolute error (MAE)*, *Statistics How To*. Available at: https://www.statisticshowto.com/absolute-error/.

[41]Koech, K.E. (2022) *Cross-entropy loss function*, *Medium*. Available at: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[42]Brownlee, J. (2020) *Why one-hot encode data in machine learning?*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

[43]Crypto1 (2023) *How does the gradient descent algorithm work in machine learning?*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/#h-what-is-gradient-descent.

[44]Dabbura, I. (2022) *Gradient descent algorithm and its variants*, *Medium*. Available at: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3.

[45]Brownlee, J. (2021) *Gentle introduction to the adam optimization algorithm for deep learning*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[46]Zhang, J. (2020) *Optimisation algorithm - adaptive moment estimation(adam)*, *Medium*. Available at:

https://towardsdatascience.com/optimisation-algorithm-adaptive-moment-estimation-adam-92144d75e232.

[47]*What is learning rate in machine learning* (2022) *Deepchecks*. Available at: https://deepchecks.com/glossary/learning-rate-in-machine-learning/.

[48]Brownlee, J. (2020) *Understand the impact of learning rate on neural network performance*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

[49]Zhou, Kun, et al. "Filter-enhanced MLP is all you need for sequential recommendation." *Proceedings of the ACM web conference 2022*. 2022.

[50]*What is deep learning?* (no date) *IBM*. Available at: https://www.ibm.com/topics/deep-learning.

[51]Hinton, Geoffrey, Yann LeCun, and Yoshua Bengio. "Deep learning." *Nature* 521.7553 (2015): 436-444.

[52]Brownlee, J. (2019) *A gentle introduction to batch normalization for Deep Neural Networks*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/.

[53]*Python introduction* (no date) *Introduction to Python*. Available at: https://www.w3schools.com/python/python_intro.asp.

[54]Sachin (2020) *Cats-vs-dogs*, *Kaggle*. Available at: https://www.kaggle.com/datasets/shaunthesheep/microsoft-catsvsdogs-dataset.

[55]Yang, Hui. "Data preprocessing." *Pennsylvania State University: Citeseer* (2018).

[56]Lenzerini, Maurizio. "Data integration: A theoretical perspective." *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2002.

[57]*Guide to data cleaning: Definition, benefits, components, and how to clean your data* (no date) *Tableau*. Available at: https://www.tableau.com/learn/articles/what-is-data-cleaning.

[58]Obaid, Hadeel S., Saad Ahmed Dheyab, and Sana Sabah Sabry. "The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning." *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*. IEEE, 2019.

[59]*Definition of lossless compression* (no date) *PCMAG*. Available at: https://www.pcmag.com/encyclopedia/term/lossless-compression.

[60]Gupta, T. (2022) *Data preprocessing in Data Mining & Machine Learning*, *Medium*. Available at: https://towardsdatascience.com/data-preprocessing-in-data-mining-machine-learning-79a9662e2eb.

[61]Sonal, Dinesh Kumar. "A study of various image compression techniques." *COIT, RIMT-IET. Hisar* 8 (2007): 97-102.

[62]John, Jacob. "Discrete cosine transform in JPEG compression." *arXiv preprint arXiv:2102.06968* (2021).

[63]Kotsiantis, Sotiris B., Ioannis Zaharakis, and P. Pintelas. "Supervised machine learning: A review of classification techniques." *Emerging artificial intelligence applications in computer engineering* 160.1 (2007): 3-24.

[64]*10 machine learning algorithms to know in 2023*. Coursera. (n.d.). https://www.coursera.org/articles/machine-learning-algorithms

[65]Narkhede, S. (2021) *Understanding confusion matrix*, *Medium*. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[66]Hossin, Mohammad, and Md Nasir Sulaiman. "A review on evaluation metrics for data classification evaluations." *International journal of data mining & knowledge management process* 5.2 (2015): 1.

[67]Torrey, Lisa, and Jude Shavlik. "Transfer learning." *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010. 242-264.

[68]Pérez-Pérez, Blanca Dalila, Juan Pablo Garcia Vazquez, and Ricardo Salomón-Torres. "Evaluation of convolutional neural networks' hyperparameters with transfer learning to determine sorting of ripe medjool dates." *Agriculture* 11.2 (2021): 115.

[69]Malte, Aditya, and Pratik Ratadiya. "Evolution of transfer learning in natural language processing." *arXiv preprint arXiv:1910.07370* (2019).

[70]Bai, Yuhao, et al. "Multi-network fusion algorithm with transfer learning for green cucumber segmentation and recognition under complex natural environment." *Computers and Electronics in Agriculture* 194 (2022): 106789.

[71]Koonce, Brett, and Brett Koonce. "ResNet 50." *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization* (2021): 63-72.

[72]He, K. et al. (2016) 'Deep residual learning for image recognition', 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [Preprint]. doi:10.1109/cvpr.2016.90.

[73]He, Kaiming, et al. "Identity mappings in deep residual networks." *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer International Publishing, 2016.

[74]Sahoo, S. (2022) *Residual blocks-building blocks of Resnet*, *Medium*. Available at: https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec.

[75]Ruiz, P. (2019) *Understanding and visualizing ResNets*, *Medium*. Available at: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8.

[76]Zagoruyko, S. and Komodakis, N. (2016) ‘Wide residual networks’, *Procedings of the British Machine Vision Conference 2016* [Preprint]. doi:10.5244/c.30.87

[77]Huang, G. *et al.* (2017) ‘Densely connected Convolutional Networks’, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [Preprint]. doi:10.1109/cvpr.2017.243

[78]Li, Yonggang, et al. "Dada: Differentiable automatic data augmentation." *arXiv preprint arXiv:2003.03780* (2020).

[79]Van Dyk, David A., and Xiao-Li Meng. "The art of data augmentation." *Journal of Computational and Graphical Statistics* 10.1 (2001): 1-50.

[80]Toyokawa, Wataru, Hye-rin Kim, and Tatsuya Kameda. "Human collective intelligence under dual exploration-exploitation dilemmas." *PloS one* 9.4 (2014): e95789.

[81]Tokic, Michel. "Adaptive ε-greedy exploration in reinforcement learning based on value differences." *Annual Conference on Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[82]Holland, John H. "Genetic algorithms." *Scientific american* 267.1 (1992): 66-73.

[83]*Scientific computing for luajit. Torch*. Available at: http://torch.ch/.

[84]*Datasets & dataloaders¶ Datasets & DataLoaders - PyTorch Tutorials 2.0.1+cu117 documentation*. Available at: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html.

[85]*CIFAR-10 and CIFAR-100 datasets*. Available at: https://www.cs.toronto.edu/~kriz/cifar.html.

[86]Cubuk, Ekin D., et al. "Autoaugment: Learning augmentation policies from data." *arXiv preprint arXiv:1805.09501* (2018).

[87]Lim, Sungbin, et al. "Fast autoaugment." *Advances in Neural Information Processing Systems* 32 (2019).

[88]Ho, Daniel, et al. "Population based augmentation: Efficient learning of augmentation policy schedules." *International conference on machine learning*. PMLR, 2019.

[89]Bartz-Beielstein, Thomas, et al. "Evolutionary algorithms." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4.3 (2014): 178-195.

[90]Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax." *arXiv preprint arXiv:1611.01144* (2016).