



ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΕΠΙΣΤΗΜΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία Εικόνων με Βαθιά Μάθηση

Υποψήφιος: Σωκράτης Καραουστάς, (Π2017033)

Επιβλέπουσα: Κάτια-Λήδα Κερμανίδου (Αναπληρώτρια Καθηγήτρια)

Κέρκυρα

2023

Η πτυχιακή εργασία εγκρίθηκε από:

Έξαρχος Θεμιστοκλής (Αναπληρωτής Καθηγητής)
Βραχάτης Αριστείδης (Επίκουρος Καθηγητής)

Υπεύθυνη Δήλωση

Με το παρόν κείμενο βεβαιώνω ότι ο κάτωθι υπογράφων είμαι συγγραφέας της παρούσης πτυχιακής εργασίας, η οποία εκπονήθηκε στο πλαίσιο των απαιτήσεων του προγράμματος σπουδών του Τμήματος Πληροφορικής του Ιόνιου Πανεπιστήμιου και η οποία παραδόθηκε, μετά από έγκριση του επιβλέποντας καθηγητή μου, σε έντυπη και ψηφιακή μορφή στη Γραμματεία του Τμήματος. Επίσης δηλώνω πως κάθε πηγή που χρησιμοποίησα (βιβλιογραφία, αρθρογραφία, δικτυογραφία), για την υποστήριξη των υποθέσεων της μελέτης και της ερευνάς μου, είναι πλήρως συμβατή με τα ακολουθούμενα επιστημονικά πρότυπα και, επιπλέον, αναφέρεται ρητά, υπό μορφή αναφοράς-παραπομπής, σε όλο το φάσμα κειμένων της παρούσης εργασίας. Το αυτό ισχύει για τη χρήση δευτερογενών δεδομένων (πινάκων, διαγραμμάτων και εικόνων), ιδεών και λέξεων, τα οποία και αναφέρονται είτε ακριβώς όπως υπάρχουν στις πηγές είτε μεθερμηνεύονται από εμένα.

Επώνυμο	Καραουσάς
Όνομα	Σωκράτης
Αριθμός Μητρώου	Π2017033
Ημερομηνία	Ιανουάριος 2023
Υπογραφή	Σωκράτης Καραουσάς

Ευχαριστίες

στην οικογένεια μου

Περίληψη

Στην παρούσα εργασία, θα αναλυθεί το θέμα της ταξινόμησης εικόνων με βαθιά μάθηση με την χρήση Python. Η ταξινόμηση εικόνας είναι όπου ένας υπολογιστής μπορεί να αναλύσει μια εικόνα και να αναγνωρίσει την «κατηγορία» στην οποία ανήκει η εικόνα. (Η μια πιθανότητα η εικόνα να είναι μέρος μιας «κατηγορίας».) Μια τάξη είναι ουσιαστικά μια ετικέτα, για παράδειγμα, «αυτοκίνητο», «ζώο», «οικοδόμημα» και στην περίπτωσή μας η αφίσα της ταινίας σε ποιό/ποιά είδη ανήκει. Αρχικά, στο 1ο κεφάλαιο θα γίνει αναφορά στην μηχανική μάθηση. Πιο αναλυτικά, για τους αλγόριθμους και τα μοντέλα της μηχανικής μάθησης, καθώς και τις εφαρμογές και τα προβλήματα που επιλύει. Επίσης, θα αναφερθούν οι αρχιτεκτονικές και εκπαίδευση των νευρωνικών δικτύων. Ακόμη, αναφορά στην υπερπροσαρμογή και υποπροσαρμογή.

Στην συνέχεια, στο 2ο κεφάλαιο θα αναλύσουμε την αναπαράσταση εικόνων, την τμηματοποίηση εικόνας και επεξεργασία εικόνων. Επιπρόσθετα, θα γίνει αναφορά στην βαθιά μάθηση και τα ρηχά και βαθιά δίκτυα, τα μοντέλα της βαθιάς μάθησης καθώς και τα συνελκτικά νευρωνικά δίκτυα με ιστορική αναδρομή. Στο 3 κεφάλαιο, θα γίνει ανάλυση στις αρχιτεκτονικές βαθιάς μάθησης όπως το LeNet-5, AlexNet, VGGNet, ResNet, Inception-GoogleNet καθώς και στο RMDL.

Επιπλέον, στο 4ο κεφάλαιο θα αναφερθούμε στην Python και τις βιβλιοθήκες της, στο TensorFlow, στο Keras και στους γράφους ροής δεδομένων. Τέλος, στο 5ο κεφάλαιο θα γίνει αναφορά στην πειραματική διαδικασία, στις εντολές και την αρχιτεκτονική του μοντέλου που χρησιμοποιήσα.

Λέξεις κλειδιά: Ταξινόμηση, Εικόνες, Βαθιά Μάθηση, Μηχανική Μάθηση, Νευρωνικά Δίκτυα, Τεχνητή Νοημοσύνη

Abstract

In this study, the topic of image classification with deep learning using Python will be analyzed. Image classification is where a computer can analyze an image and identify the "class" the image belongs to. (Or a possibility that the image is part of a 'category'.) A class is essentially a label, for example 'car', 'animal', 'building' and in our case the movie poster which genre(s) it belongs to. Initially, in the 1st chapter, reference will be made to machine learning. More in detail, about the algorithms and models of machine learning, as well as the applications and problems it solves. Also, the architectures and training of neural networks will be mentioned. Also, reference to overfitting and underfitting.

Then, in the 2nd chapter we will analyze image representation, image segmentation and image processing. In addition, reference will be made to deep learning and shallow and deep networks, deep learning models as well as convolutional neural networks with historical regress. In chapter 3, deep learning architectures such as LeNet-5, AlexNet, VGGNet, ResNet, Inception-GoogLeNet as well as RMDL will be analyzed.

Additionally, in Chapter 4 we will cover Python and its libraries, TensorFlow, Keras, and dataflow graphs. Finally, in the 5th chapter reference will be made to the experimental process, the commands and the architecture of the model I used.

Key words: Classification, Images, Deep Learning, Machine Learning, Neural Networks, Artificial Intelligence

Περιεχόμενα

	Ευχαριστίες	Σελ. 4
	Περίληψη	5
	Abstract	6
	Περιεχόμενα	8
	Λίστα Διαγραμμάτων	11
	Λίστα Εικόνων	12
	Εισαγωγή	14
	ΚΕΦΑΛΑΙΟ 1ο – Εισαγωγή	
1.1	Μηχανική μάθηση	15
	1.1.1 Αλγόριθμοι μηχανικής μάθησης	18
	1.1.2 Μοντέλα μηχανικής μάθησης	19
	1.1.3 Εφαρμογές και προβλήματα που επιλύει η μηχανική μάθηση	19
1.2	Νευρωνικά Δίκτυα	20
	1.2.1 Αρχιτεκτονικές και εκπαίδευση νευρωνικών δικτύων	21
1.3	Overfitting - underfitting	23
	ΚΕΦΑΛΑΙΟ 2ο – Αναπαράσταση εικόνων με Βαθιά μάθηση	
2.1	Αναπαράσταση εικόνων	26
	2.1.1 Image Segmentation (Τμηματοποίηση Εικόνας)	28
	2.1.2 Image Processing (Επεξεργασία Εικόνας)	29
2.2	Βαθιά μάθηση	30
	2.2.1 Ρηχά και Βαθιά δίκτυα	32

2.3	Μοντέλα βαθιάς μάθησης	33
2.4	CNN - Convolutional Neural Networks	36
	2.4.1 Ιστορική αναδρομή	37
	2.4.2 Stochastic Gradient Descent	38
	ΚΕΦΑΛΑΙΟ 3ο – Αρχιτεκτονικές βαθιάς μάθησης	
3.1	Γνωστές αρχιτεκτονικές deep neural networks	40
3.2	Αρχιτεκτονικές βαθιάς μάθησης	42
	3.2.1 LeNet-5	42
	3.2.2 AlexNet	43
	3.2.3 VGGNet	44
	3.2.4 ResNet	45
	3.2.5 Inception-GoogLeNet	46
3.3	RMDL και SimpleNet	48

	ΚΕΦΑΛΑΙΟ 4ο – Python	
4.1	Εισαγωγή	50
4.2	Χρήσιμες βιβλιοθήκες της Python	50
4.3	TensorFlow, Keras	51
	4.3.1 Γράφοι Ροής Δεδομένων	52
4.4	Εξόρυξη και ανάλυση κειμένου με χρήση της Python	54
	ΚΕΦΑΛΑΙΟ 5ο – Πειραματική Διαδικασία	
5.1	Περιγραφή - Στόχοι	55
5.2	Υλοποίηση	55
	5.2.1 Προετοιμασία συνόλου δεδομένων	55
	5.2.2 Εξόρυξη και αποθήκευση εικόνων	55
	5.2.3 Επιλογή αρχιτεκτονικής του δικτύου	56
	5.2.4 Σύνθεση του μοντέλου	62
	5.2.5 Εκπαίδευση του μοντέλου	62

	5.2.6 Οπτικοποίηση με matplotlib	63
5.3	Συμπεράσματα - Προβλήματα	64
	5.3.1 Προβλήματα	64
	5.3.2 Συμπεράσματα	65
	5.3.3 Μελλοντική δουλειά	66
	Παράρτημα 1) get_data.py	67
	Παράρτημα 2) create_db.py	68
	Παράρτημα 3) movies_dataset.py	69
	Παράρτημα 4) movies_genre_model.py	76
	Παράρτημα 5) movies_milix_model.py	79
	Παράρτημα 6) prepare_dcgan_dataset.py	82
	Παράρτημα 7) tests.py	84
	Παράρτημα 8) train_model.py	88
	Πηγές – Αναφορές	
	Δικτυογραφία	90
	Εικόνες	92
	Διαγράμματα	93

Λίστα Διαγραμμάτων

Διάγραμμα 1.1	Αναδρομικό νευρωνικό δίκτυο	Σελ. 23
Διάγραμμα 1.2	Υποπροσαρμογή-Υπερπροσαρμογή	Σελ. 25
Διάγραμμα 2.1	Faster RCNN Δίκτυο	Σελ. 34
Διάγραμμα 2.2	CNN	Σελ. 37
Διάγραμμα 2.3	Stochastic Gradient Descent	Σελ. 39
Διάγραμμα 3.1	LeNet-5	Σελ. 43
Διάγραμμα 3.2	ResNet	Σελ. 46
Διάγραμμα 3.3	Inception	Σελ. 47
Διάγραμμα 3.4	GoogleNet τελική μορφή	Σελ. 48
Διάγραμμα 3.5	RMDL	Σελ. 49
Διάγραμμα 4.1	Γράφοι Ροής Δεδομένων	Σελ. 53
Διάγραμμα 5.1	Απώλεια εκπαίδευσης/επικύρωσης	Σελ. 61
Διάγραμμα 5.2	Ακρίβεια εκπαίδευσης/επικύρωσης	Σελ. 62
Διάγραμμα 5.3	Απώλεια εκπαίδευσης/επικύρωσης (1)	Σελ. 64
Διάγραμμα 5.4	Ακρίβεια εκπαίδευσης/επικύρωσης (1)	Σελ. 64
Διάγραμμα 5.5	Η συνάρτηση ενεργοποίησης softmax	Σελ. 65

Λίστα Εικόνων

Εικόνα 1.1	Φάσεις Μηχανικής μάθησης	Σελ. 16
Εικόνα 1.2	Εξέλιξη μηχανικής μάθησης	Σελ. 18
Εικόνα 1.3	Νευρωνικά δίκτυα	Σελ. 21
Εικόνα 1.4	Δίκτυα Εμπρόσθιας τροφοδότησης	Σελ. 22
Εικόνα 2.1	Αναπαράσταση Εικόνων (απρόμαυρη)	Σελ. 27
Εικόνα 2.2	Αναπαράσταση Εικόνων (έγχρωμη)	Σελ. 28
Εικόνα 2.3	Image segmentation	Σελ. 29
Εικόνα 2.4	Image Processing	Σελ. 30
Εικόνα 2.5	Μηχανική Μάθηση-Βαθιά Μάθηση	Σελ. 31
Εικόνα 2.6	Αρχιτεκτονική Faster RCNN	Σελ. 35
Εικόνα 3.1	AlexNet	Σελ. 44
Εικόνα 3.2	VGGNet	Σελ. 45
Εικόνα 5.1	MaxPooling2D	Σελ. 57
Εικόνα 5.2	Δομή του Sequential δικτύου	Σελ. 58
Εικόνα 5.3	Εκτέλεση κώδικα για 6x6 πυρήνα	Σελ. 59
Εικόνα 5.4	Εκτέλεση	Σελ. 61

Εισαγωγή – Προλεγόμενα

Η Βαθιά Μάθηση είναι μοντέλα και αλγόριθμοι οι οποίοι επιτρέπουν στους υπολογιστές να μαθαίνουν από πολύπλοκα δεδομένα. Τα Βαθιά Συνελικτικά Νευρωνικά Δίκτυα, ανατροφοδοτούμενα νευρωνικά δίκτυα, στοχαστικοί αλγόριθμοι εκπαίδευσης από σύνολα δεδομένων μεγάλης κλίμακας είναι κάποια δίκτυα στην Βαθιά Μάθηση. Τεχνικές μη επιβλεπόμενης βαθιάς μάθησης χρησιμοποιώντας μεταβολικούς αυτόματους κωδικοποιητές και παραγωγικά δίκτυα αμφισβήτησης. Τεχνικές βαθιάς ενισχυτικής μάθησης με εφαρμογές στην ρομποτική και την αυτόματη εκμάθηση παιγνίων. Στην παρούσα εργασία, θα ήθελα να καλύψω ένα κομμάτι της ταξινόμησης εικόνων χρησιμοποιώντας Βαθιά Μάθηση. Έτσι, θα γίνει αναφορά στην έννοια της Βαθιάς Μάθησης, ανάλυση των Νευρωνικών Δικτύων και περιγραφή γνωστών μοντέλων Βαθιάς Μάθησης. Στη συνέχεια, θα γίνει αναφορά στο πώς θα γίνει η δημιουργία ενός ταξινομητή εικόνων και μετά πηγαίνουμε σε υλοποίηση με την χρήση της γλώσσας Python, δηλαδή στην δημιουργία των μοντέλων για την ταξινόμηση των συνόλων δεδομένων των εικόνων. Συγκεκριμένα, στην εργασία αυτή θα εξετασθεί το σύνολο Movie Posters. Επίσης, θα αναφερθούμε σε κάποιες αρχιτεκτονικές Βαθιάς Μάθησης όπως το LeNet-5, AlexNet, VGGNet, ResNet και Inception-GoogLeNet.

ΚΕΦΑΛΑΙΟ 1^ο

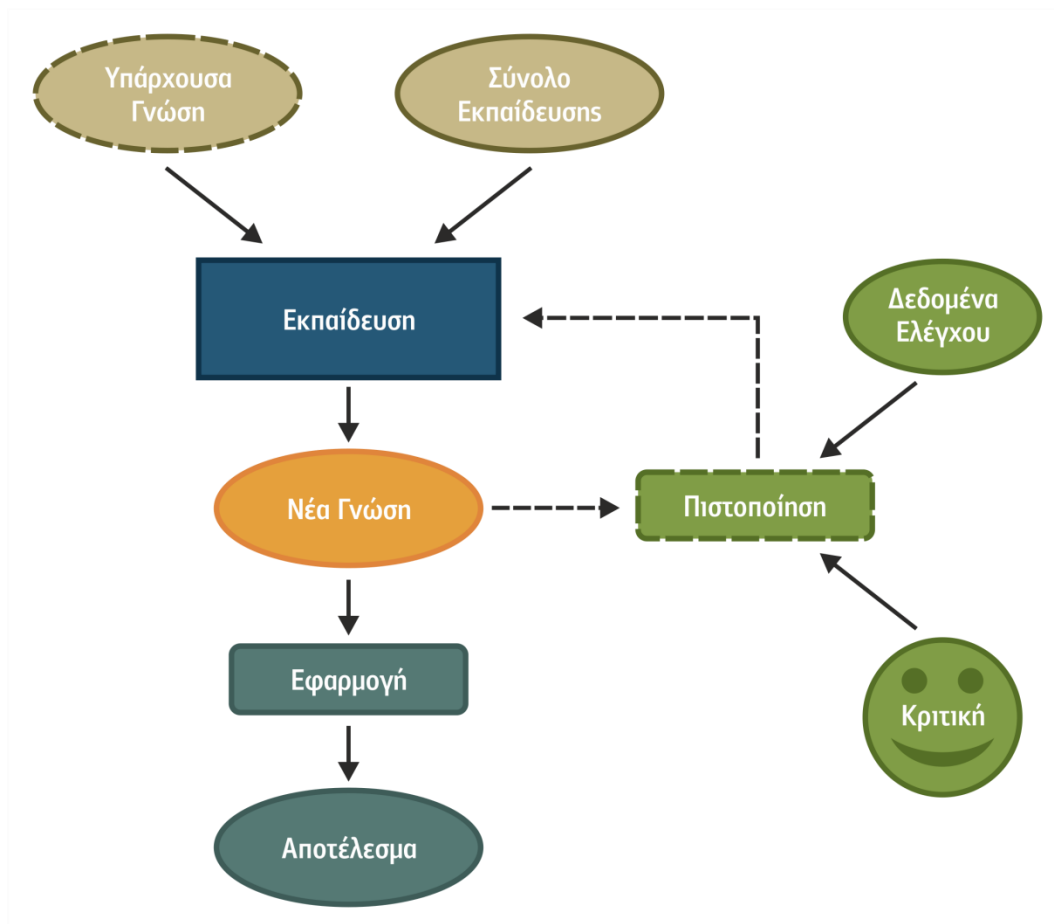
Εισαγωγή

1.1. Μηχανική Μάθηση

Μηχανική Μάθηση (Machine Learning - ML) ονομάζεται η επιστήμη η οποία αναφέρεται στην ανάπτυξη προγραμμάτων τα οποία έχουν τη δυνατότητα να μαθαίνουν, χωρίς τη χρήση ρητών οδηγιών από τον προγραμματιστή, βελτιώνοντας την ακρίβεια τους σταδιακά. Συχνά υπάρχει σύγχυση ανάμεσα στον όρο Μηχανική Μάθηση και Βαθιά Μάθηση (Deep Learning), ωστόσο οι δύο όροι, παρ'ότι συγγενεύουν, αναφέρονται σε διαφορετικά πεδία της Τεχνητής Νοημοσύνης (Artificial Intelligence)[1].

Οι αλγόριθμοι, οι οποίοι περιλαμβάνονται στη μηχανική μάθηση ως αντικείμενο έρευνας, λαμβάνουν και αναλύουν δεδομένα εισόδου. Με αυτό τον τρόπο μπορούν να κάνουν προβλέψεις σχετικά με το αποτέλεσμα μέσα στα πλαίσια ενός αποδεκτού εύρους σφάλματος. Ανάλογα με την ποσότητα των δεδομένων που δέχεται ο αλγόριθμος σαν είσοδο, αποκτά και την αντίστοιχη ακρίβεια προβλέψεων μετά την εκπαίδευσή του. Όσο περισσότερα είναι τα δεδομένα εισόδου τόσο πιο ακριβής θα γίνει ο αλγόριθμος πρόβλεψης. Κατά την εκπαίδευσή του επεξεργάζεται και τροποποιεί διάφορες λειτουργίες του, βελτιώνοντας την απόδοση του με αποτέλεσμα την ύπαρξη μιας μορφής ευφυΐας.[2] Η Μηχανική Μάθηση εμφανίζεται ως επί το πλείστον σε υπολογιστικά προβλήματα, στις οποίες ο ακριβής σχεδιασμός και ο ρητός προγραμματισμός των αλγορίθμων είναι ανέφικτος. Τέτοια παραδείγματα είναι το φιλτράρισμα ανεπιθύμητων μηνυμάτων (spam filtering), οι μηχανές αναζήτησης, η οπτική αναγνώριση χαρακτήρων (OCR), η υπολογιστική όραση κ.α.

Στην συνέχεια, σκοπός της μηχανικής μάθησης είναι η ανάπτυξη πολύπλοκων μοντέλων, τα οποία θα κατευθύνουν την μάθηση προς την πρόβλεψη. Οι ερευνητές, μέσω των συγκεκριμένων μοντέλων, είναι σε θέση να παράγουν αξιόπιστα αποτελέσματα για τη λήψη αποφάσεων. Υπάρχουν τέσσερις βασικές κατηγορίες που χωρίζονται οι αλγόριθμοι της Μηχανικής Μάθησης. Η διάκριση σε αυτές τις κατηγορίες πραγματοποιείται με κριτήριο τα δεδομένα εισόδου και το είδος τους.



Εικόνα 1.1 Φάσεις Μηχανικής μάθησης
(http://repfiles.kallipos.gr/html_books/93/04a-main.html)

Σε σχέση με τον τύπο εργασίας που πραγματοποιούν , καθώς επίσης και με το είδος δεδομένων εισόδου και εξόδου , οι τέσσερις βασικές κατηγορίες οι οποίες αποτελούν την Μηχανική Μάθηση είναι οι εξής : [3]

1. Επιβλεπόμενη Μάθηση (Supervised Learning)

Η επιβλεπόμενη μάθηση χρησιμοποιεί παραδείγματα με ετικέτες (labeled examples) $(x_i, y_i) \in N$, όπου κάθε x_i είναι ένα διάνυσμα χαρακτηριστικών (feature vector). Αυτές οι ετικέτες αποτελούν το σύνολο δεδομένων. Κάθε διάνυσμα δίνει ένα παράδειγμα για κάθε διάσταση $x(j)$, $j = 1, \dots, D$. Λαμβάνοντας ως παράδειγμα το ανθρώπινο σύνολο δεδομένων, κάθε x θα μπορούσε να αντιπροσωπεύει το ύψος σε $x(1)$, το βάρος σε $x(2)$ και ούτω καθεξής. Η ετικέτα y_i είναι ένα στοιχείο που ανήκει στο πεπερασμένο σύνολο της κλάσεων $(1, 2, \dots, C)$, ή έναν πραγματικό αριθμό ή πιο σύνθετες δομές. Επομένως, η ταξινόμηση των email έχει δύο ετικέτες: ανεπιθύμητη

και μη ανεπιθύμητη. Ο αλγόριθμος επιβλεπούμενης μάθησης στοχεύει στη χρήση ενός συνόλου δεδομένων για την κατασκευή ενός μοντέλου. Το μοντέλο αυτό παίρνει ένα x διάνυσμα χαρακτηριστικών ως είσοδο και επιχειρεί να κάνει σωστά την αντιστοίχιση των μη ετικετοποιημένων παραδειγμάτων με τις επιθυμητές εξόδους. Ουσιαστικά, προσπαθεί να προβλέψει τη σωστή κλάση. Στην περίπτωση που έχουμε διακριτά δεδομένα η επιβλεπόμενη μάθηση διακρίνεται σε προβλήματα ταξινόμησης (classification), ενώ όταν έχουμε συνεχή δεδομένα σε παλινδρόμησης (regression). [4]

2. Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

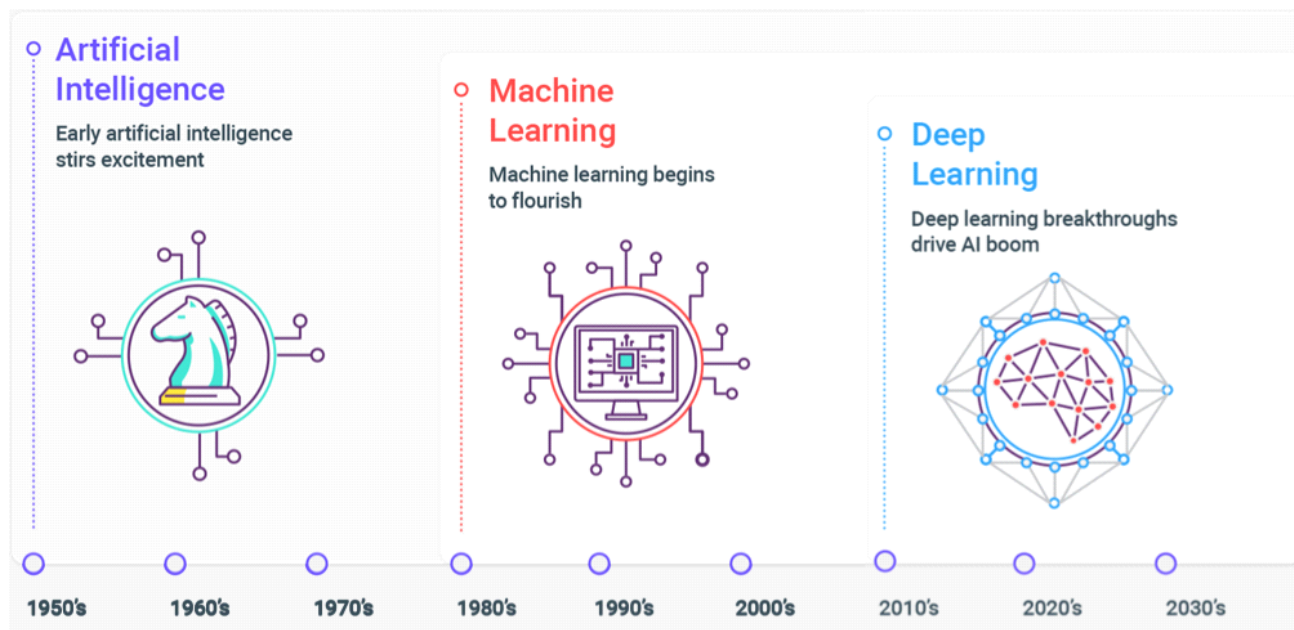
Στην κατηγορία μάθησης αυτή, μη ετικετοποιημένα παραδείγματα (x_i)N αποτελούν το σύνολο δεδομένων. Το x και πάλι είναι διάνυσμα χαρακτηριστικών, το οποίο λαμβάνεται από τον αλγόριθμο μάθησης ως είσοδος και παράγει ένα μοντέλο προκειμένου να βρεθεί κάποια δομή για τα δεδομένα, όπως μια ομάδα ή μια συστάδα. Ουσιαστικά, ο αλγόριθμος εντοπίζει τα κοινά στοιχεία των δεδομένων ώστε να τα τοποθετήσει σε ομάδες και να τα κατηγοριοποιήσει. Επιπλέον, δεν υπάρχει σφάλμα στην αξιολόγηση του αποτελέσματος διότι υπάρχει έλλειψη προηγούμενης γνώσης σχετικά με τις κατηγορίες δεδομένων.

3. Ημι-Επιβλεπόμενη Μάθηση (Semi-supervised Learning)

Η ημι-επιβλεπόμενη μάθηση βρίσκεται ανάμεσα στην επιβλεπόμενη και στη μη επιβλεπόμενη, διότι αποτελείται από ετικετοποιημένα και μη ετικετοποιημένα παραδείγματα. Τα μη ετικετοποιημένα παραδείγματα υπερτερούν σε αριθμό έναντι των ετικετοποιημένων. Οι αλγόριθμοι της ημι-επιβλεπόμενης μάθησης έχουν τον ίδιο στόχο με αυτόν της επιβλεπόμενης. Η διαφορά έγκειται στο γεγονός ότι το πλήθος των μη ετικετοποιημένων παραδειγμάτων ενδέχεται να δημιουργήσει ένα καλύτερο μοντέλο.

4. Ενισχυτική Μάθηση (Reinforcement Learning)

Η ενισχυτική μάθηση χρησιμοποιεί έναν αλγόριθμο, ο οποίος συνδέεται άμεσα με το περιβάλλον και αναπτύσσει τεχνικές με σκοπό τη μεγιστοποίηση της συνάρτησης ανταμοιβής. Ο αλγόριθμος σε αυτή την περίπτωση βασίζεται στις έννοιες της επιβράβευσης και τιμωρίας. Το σύστημα ανακαλύπτει μόνο του το είδος των ενεργειών που θα του προσφέρουν μεγαλύτερο όφελος χωρίς την καθοδήγηση από εξωτερικό επιβλέποντα. Παραδείγματα αυτής της κατηγορίας μπορούν να βρεθούν σε αυτόνομα οχήματα και παιχνίδια εκπαίδευσης ενάντια σε ανθρώπινους αντιπάλους και άλλα.



Εικόνα 1.2 Εξέλιξη μηχανικής μάθησης
(<https://medium.com/swlh/5-ways-deep-learning-is-changing-our-day-to-day-life7c17875b720c>)

1.1.1 Αλγόριθμοι μηχανικής μάθησης

Ενδέχεται να απαιτούνται διάφοροι παράγοντες για την επιλογή του σωστού αλγόριθμου μηχανικής μάθησης, όπως το μέγεθος των δεδομένων, η ποιότητα, η ποικιλία και ο τύπος των αναζητούμενων απαντήσεων. Η ακρίβεια του αλγορίθμου, ο ρυθμός εκμάθησης κ.λπ., διαδραματίζουν τον σημαντικότερο ρόλο στις μέρες μας εξαιτίας του μεγάλου όγκου δεδομένων. Η επιλογή του σωστού αλγορίθμου, ωστόσο, δεν είναι εύκολη. Χρειάζεται αναλυτική έρευνα και πειραματισμός και με άλλους αλγορίθμους ώστε, ακόμα και οι αναλυτές δεδομένων με περισσότερη εμπειρία, να είναι σε θέση να προτείνουν κάποιον αλγόριθμο με βεβαιότητα. Ουσιαστικά, εφαρμόζεται η μέθοδος trial and error. Το trial and error είναι η κύρια μέθοδος επίλυσης προβλημάτων. Χαρακτηρίζεται από επαναλαμβανόμενες ποικίλες προσπάθειες που συνεχίζονται μέχρι να επιτευχθεί η επιτυχία, ή έως ότου ο ασκούμενος σταματήσει να προσπαθεί. Τα προβλήματα διακρίνονται σε διάφορες κατηγορίες και με βάση αυτές χρησιμοποιούνται οι κατάλληλοι αλγόριθμοι που μπορούν να τα λύσουν. Υπάρχουν βέβαια και αλγόριθμοι, οι οποίοι χρησιμοποιούνται στις περισσότερες περιπτώσεις.[5]

1.1.2 Μοντέλα μηχανικής μάθησης

Στη Μηχανική Μάθηση, οι κατηγορίες προβλημάτων και η μορφή των δεδομένων αποτελούν κριτήρια βάση των οποίων επιλέγονται οι αλγόριθμοι. Τα δεδομένα διακρίνονται στο σύνολο εκπαίδευσης (training set) και το σύνολο ελέγχου (test set). Το σύνολο εκπαίδευσης, το οποίο αποτελείται από γνωστά παραδείγματα, λειτουργεί ως είσοδος για τον αλγόριθμο. Το σύνολο αυτό χρησιμοποιείται ως μέσο εκπαίδευσης κάποιου μοντέλου (model fit on training dataset). Το μοντέλο αυτό αναγνωρίζει πρότυπα και σχέσεις ανάμεσα στα δεδομένα. Με αυτό τον τρόπο μπορεί αργότερα να προβλέψει για τα άγνωστα παραδείγματα. Η μηχανική μάθηση στοχεύει στο να μπορεί το μοντέλο να αποδώσει ικανοποιητικά στα νέα δεδομένα και όχι τόσο στα δεδομένα εκπαίδευσης. Στόχος, δηλαδή, είναι η γενίκευση. Στις περισσότερες περιπτώσεις τα 2/3 των δεδομένων χρησιμοποιούνται για την εκπαίδευση και το 1/3 στον έλεγχο.

Προκειμένου να υλοποιηθεί η Μηχανική Μάθηση, χρησιμοποιούνται τα κατάλληλα μοντέλα, αυτά δηλαδή που έχουν εκπαιδευτεί στα δεδομένα εκπαίδευσης και έπειτα έχουν την ικανότητα επεξεργασίας των νέων δεδομένων και την χρήση τους για προβλέψεις. Τα συστήματα μηχανικής μάθησης έχουν χρησιμοποιήσει διάφορους τύπους μοντέλων μεταξύ των οποίων είναι οι εξής :

- Δέντρο Απόφασης - Decision Tree
- Naive Bayes
- Νευρωνικά Δίκτυα
- Μηχανές διανυσμάτων υποστήριξης - SVM (Support vector machines)
- Κ-πλησιέστεροι γείτονες - k-Nearest Neighbors (kNN)

Η συγκεκριμένη διάκριση των κυριότερων μοντέλων Μηχανικής Μάθησης έγινε αναφορικά με την ταξινόμηση.

1.1.3 Εφαρμογές και προβλήματα που επιλύει η μηχανική μάθηση

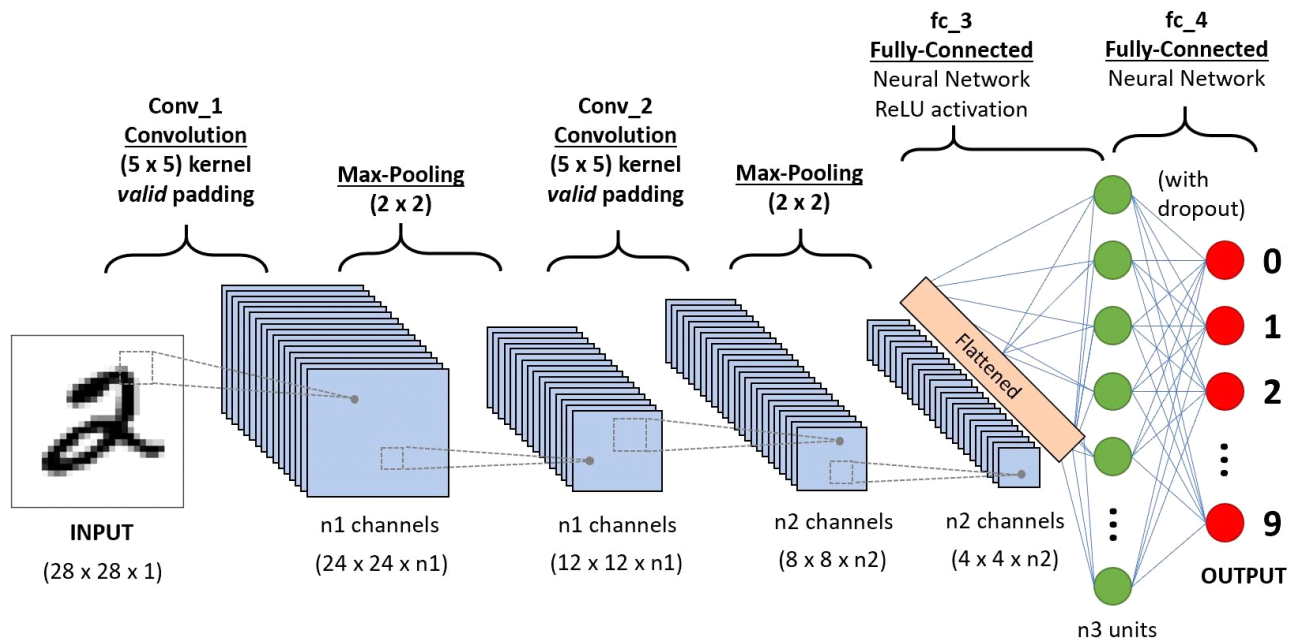
Οι εφαρμογές της Μηχανικής Μάθησης πληθαίνουν όλο και περισσότερο σε διάφορα επιστημονικά πεδία, μεταξύ των οποίων η Ιατρική, η Οικονομία και η Επιστήμη Υπολογιστών. Αναλυτικότερα, μερικές από τις εφαρμογές της μηχανικής μάθησης είναι η υπολογιστική όραση, η αυτόνομη οδήγηση, η αναγνώριση ομιλίας, προσώπου και προτύπων και η επεξεργασία φυσικής γλώσσας. Ακόμη, η μηχανική μάθηση έχει εφαρμογές σε ιατρικές διαγνώσεις, ανίχνευση απάτης, για παράδειγμα στην περίπτωση των πιστωτικών καρτών και σε ηλεκτρονικά παιχνίδια. Οι αλγόριθμοι Μηχανικής Μάθησης είναι απαραίτητοι για την επίλυση

προβλημάτων όπως η ταξινόμηση, η παλινδρόμηση (regression), η ομαδοποίηση (clustering) και η μείωση διαστάσεων.

1.2 Νευρωνικά δίκτυα

Κατά παράδοση , ο όρος νευρωνικό δίκτυο αναφέρεται σε ένα δίκτυο ή κύκλωμα των βιολογικών νευρώνων. Το νευρωνικό δίκτυο , στη σύγχρονη χρήση του, εστιάζει σε τεχνητά νευρωνικά δίκτυα. Τα τεχνητά νευρωνικά δίκτυα , τα οποία αποτελούνται από τεχνητούς νευρώνες ή κόμβους, διακρίνονται σε βιολογικά νευρωνικά δίκτυα και σε τεχνητά νευρωνικά δίκτυα. Τα πρώτα κατασκευάζονται από πραγματικούς βιολογικούς νευρώνες , που συνδέονται με ένα νευρωνικό σύστημα. Τα τεχνητά νευρωνικά δίκτυα περιλαμβάνουν διασυνδεδεμένους τεχνητούς νευρώνες.

Σύμφωνα με τον Haykin, ο εγκέφαλος ως πολύπλοκος , μη γραμμικός και παράλληλος επεξεργαστής χρησιμοποιεί τους Νευρώνες , που αποτελούν και τα δομικά του στοιχεία, με κατάλληλο τρόπο ώστε να μπορούν να κάνουν υπολογισμούς σε σχέση με την αναγνώριση προτύπων, τον έλεγχο, την αντίληψη, την κίνηση κ.α.[6] Οι νευρώνες του εγκεφάλου λειτουργούν με ταχύτητα πολλαπλάσια ενός υπολογιστή με τις σημερινές δυνατότητες. Ήδη από την γέννηση του ανθρώπου, ο εγκέφαλος , ο οποίος έχει σημαντική δομή, δημιουργεί δικούς του κανόνες συμπεριφοράς (εμπειρία). Μια σημαντική ικανότητα που διαθέτει το νευρικό σύστημα του ανθρώπου είναι η ρύθμιση και έλεγχος της λειτουργίας όλων των οργάνων του ανθρώπινου σώματος. Επιπλέον, είναι υπεύθυνο και για την ομαλή συνύπαρξη των οργάνων. Ακόμη, το νευρικό σύστημα βρίσκεται στο επίκεντρο των ψυχικών λειτουργιών. Επίσης, μέσω του ματιού, του αυτιού, και των υπόλοιπων αισθητήριων οργάνων, το νευρικό σύστημα βοηθά τον άνθρωπο να αντιληφθεί το περιβάλλον.



Εικόνα 1.3 Νευρωνικά δίκτυα
<https://towardsdatascience.com/reducing-the-carbon-foot-prints-of-cnns-at-the-cost-of-interactions-depthwise-pointwise-conv-5df850ea33a4>

1.2.1 Αρχιτεκτονικές και εκπαίδευση νευρωνικών δικτύων

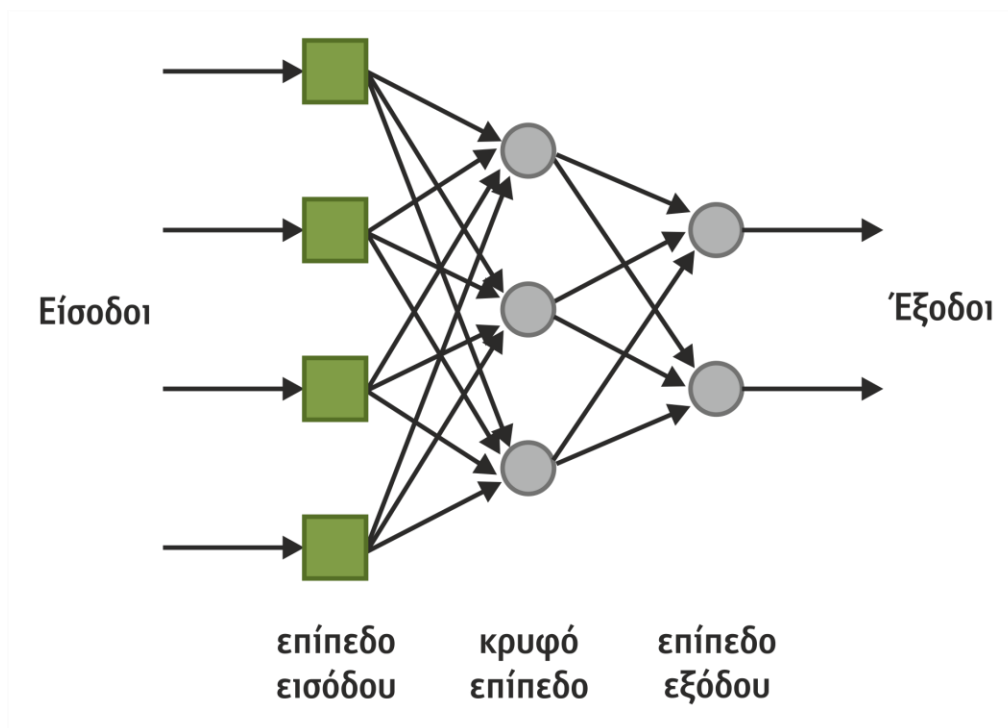
Ο τρόπος διάταξης και σύνδεσης των κόμβων μεταξύ τους έχει σχέση με την αρχιτεκτονική των νευρωνικών δικτύων. Οι κόμβοι διασυνδέονται σε διαφορετικά επίπεδα. Τα επίπεδα αυτά είναι ένα επίπεδο εισόδου (input layer), ένα ή περισσότερα κρυφά επίπεδα (hidden layers) και ένα επίπεδο εξόδου (output layer).

Η επιλογή της πιο κατάλληλης αρχιτεκτονικής είναι μια βασική απόφαση κατά τον σχεδιασμό ενός νευρωνικού δικτύου, καθώς καθορίζει την ικανότητα του να επιλύει ένα συγκεκριμένο πρόβλημα και σχετίζεται στενά με τον αλγόριθμο εκπαίδευσης που χρησιμοποιείται.

Κρίνεται ιδιαίτερα σημαντικό, για την σχεδίαση ενός νευρωνικού δικτύου, να επιλεγεί η σωστή αρχιτεκτονική. Η αρχιτεκτονική ενός νευρωνικού δικτύου είναι υπεύθυνη για την ικανότητά του να λύνει ορισμένα προβλήματα. Επίσης, σχετίζεται άμεσα με τους αλγόριθμους εκπαίδευσης. Τα δίκτυα εμπρόσθιας τροφοδότησης (Feed Forward Neural Networks) και τα αναδρομικά νευρωνικά δίκτυα (Recurrent Neural Networks-RNN) αποτελούν τις βασικότερες αρχιτεκτονικές που υπάρχουν.

1. Δίκτυα Εμπρόσθιας τροφοδότησης (Feed Forward Neural Networks)

Πιο αναλυτικά, τα δίκτυα εμπρόσθιας τροφοδότησης αποτελούν έναν από τους πιο απλούς τύπους των τεχνητών νευρωνικών δικτύων. Τα δεδομένα, σε ένα νευρωνικό δίκτυο εμπρόσθιας τροφοδότησης, μέσω των διάφορων κόμβων εισόδων φτάνουν στον κόμβο εξόδου. Η μεταφορά πληροφοριών και η εκτέλεση υπολογισμών πραγματοποιούνται σε ένα ή περισσότερα κρυμμένα επίπεδα. Η εκτέλεση υπολογισμών δεν είναι εφικτή στο επίπεδο εισόδου. Τα Perceptrons (δυναμικοί γραμμικοί ταξινομητές) και τα Perceptrons πολλών στρωμάτων (Multi-Layer) αποτελούν τα δύο βασικά είδη νευρωνικών δικτύων εμπρόσθιας τροφοδότησης. Τα τελευταία διαθέτουν περισσότερα από δύο κρυμμένα επίπεδα. Επιπλέον, οι σιγμοειδείς συναρτήσεις ενεργοποίησης και η μέθοδος εκμάθησης backpropagation (ανάστροφη διάδοση) που χρησιμοποιούνται ρυθμίζουν τα βάρη του δικτύου και ελαχιστοποιούν τη συνάρτηση σφάλματος.

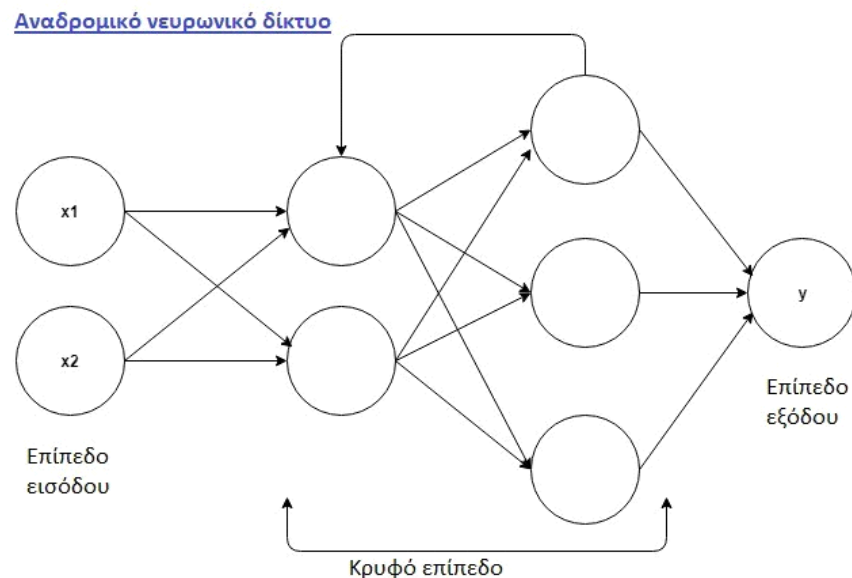


Εικόνα 1.4 Δίκτυα εμπρόσθιας τροφοδότησης
(http://repfiles.kallipos.gr/html_books/93/04a-main.html, σχ. 4.19)

2. Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks)

Από την άλλη πλευρά, τα Αναδρομικά Νευρωνικά Δίκτυα περιλαμβάνουν το λιγότερο έναν βρόγχο ανάδρασης. Ο κάθε νευρώνας που φεύγει από το δίκτυο ανατροφοδοτεί άλλους νευρώνες, οι οποίοι εισέρχονται στο ίδιο επίπεδο. Ακόμη, μπορεί να συμβεί αυτοανάδραση, δηλαδή η έξοδος του νευρώνα μπορεί να οδηγήσει στην είσοδο του ίδιου του νευρώνα στο ίδιο επίπεδο. Η λειτουργία του κάθε κόμβου είναι σημαντική, διότι αυτός δρα ως κύτταρο μνήμης

ενώ παράλληλα χρησιμοποιείται ως εργαλείο υπολογισμού και εκτέλεσης πράξεων. Ακόμη οι κόμβοι υπάρχουν , όπως θα δούμε και στη συνέχεια , και στη βαθιά μάθηση. Το LSTM (Long Short-Term Memory-Μακροπρόθεσμη Μνήμη) και Gated Recurrent Unit (GRU) αποτελούν υποκατηγορία των αναδρομικών νευρωνικών δικτύων.



Διάγραμμα 1.1 Αναδρομικό νευρωνικό δίκτυο
(<https://medium.com/@curiously/making-a-predictive-keyboard-using-recurrent-neural-networks-tensorflow-for-hackers-part-v-3f238d824218>)

1.3 Overfitting - underfitting

Σύμφωνα με την στατιστική , η υπερπροσαρμογή αναλύει ένα συγκεκριμένο σύνολο δεδομένων με σχετική ή απόλυτη ακρίβεια . Αυτό έχει ως αποτέλεσμα την αποτυχία να προστεθούν δεδομένα ή να γίνουν σωστές προβλέψεις για μελλοντικές παρατηρήσεις. Η υπερπροσαρμογή έχει ως σκοπό την ακούσια εξαγωγή μέρους της υπολειπόμενης διακύμανσης , δηλαδή του θορύβου. Η διαδικασία αυτή γίνεται έτσι ώστε, η παραλλαγή να εκπροσωπεί την υποκείμενη δομή του μοντέλου. Ένα μοντέλο υπερπροσαρμογής , το οποίο είναι στατιστικό μοντέλο , περιλαμβάνει αρκετές παραμέτρους , περισσότερες από όσες αιτιολογούνται από τα δεδομένα.

Όταν το κριτήριο της επιλογής ενός μοντέλου διαφέρει από το κριτήριο καταλληλότητάς του τότε συμβαίνει υπερβολική εκπαίδευση. Παραδείγματος χάρη , το κριτήριο επιλογής ενός μοντέλου είναι η αύξηση της απόδοσης του σε ένα σύνολο δεδομένων εκπαίδευσης ενώ το κριτήριο καταλληλότητάς του έχει σχέση με την ικανότητα ικανοποιητικής απόδοσης σε μη ορατά δεδομένα. Σε αυτή την περίπτωση , η υπερβολική εκπαίδευση συμβαίνει όταν ένα μοντέλο έχει αποστηθίσει δεδομένα εκπαίδευσης και όχι όταν έχει τις γνώσεις γενίκευσης μίας τάσης.

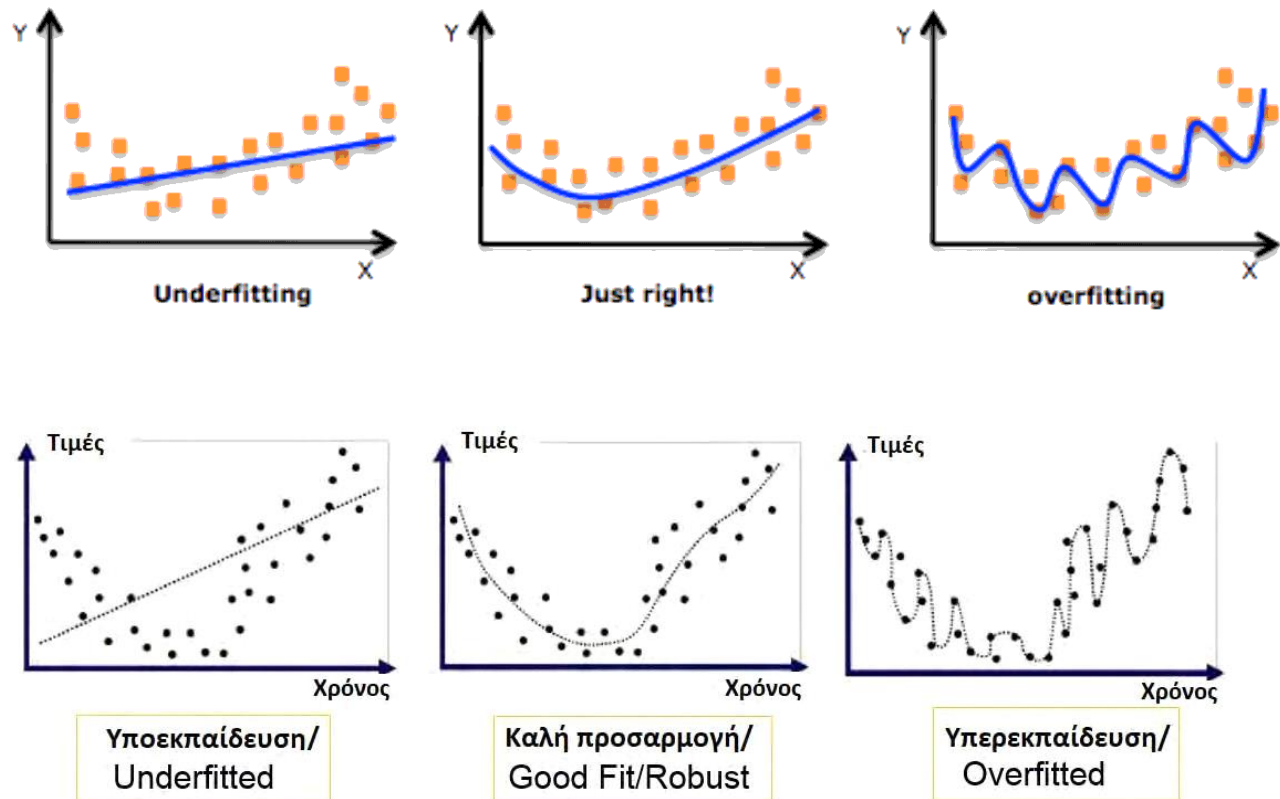
Όταν οι παράμετροι είναι ίσες ή μεγαλύτερες σε αριθμό από τις παρατηρήσεις , τότε το μοντέλο έχει την ικανότητα να δώσει μία πρόβλεψη των δεδομένων εκπαίδευσης που να είναι σωστή και ακριβής . Αυτό έχει επιτευχθεί χρησιμοποιώντας αποκλειστικά την απομνημόνευση των συνολικών δεδομένων. Όμως το συγκεκριμένο μοντέλο ενδέχεται να αποτύχει όταν πραγματοποιεί διάφορες προβλέψεις.

Εκτός από τον αριθμό των δεδομένων και των παραμέτρων, η υπερπροσαρμογή μπορεί να συμβεί και εξαιτίας της συμβατότητας της δομής του μοντέλου με το μέγεθος των σφαλμάτων και το σχήμα των δεδομένων σε σχέση με το επιθυμητό προσδοκώμενο επίπεδο σφάλματος ή θορύβου στα δεδομένα. Στις περιπτώσεις που οι παράμετροι του προσαρμοσμένου μοντέλου δεν είναι υπερβολικά πολλές , η προσαρμοσμένη σχέση είναι γνωστό πως δεν θα είναι τόσο λειτουργική σε ένα νέο σύνολο δεδομένων σε σχέση με ένα σύνολο δεδομένων που υπάρχει στην προσαρμογή . Αυτό το φαινόμενο συχνά αναφέρεται ως συρρίκνωση. Ειδικότερα, η τιμή του συντελεστή προσδιορισμού θα συρρικνωθεί σε σχέση με τα αρχικά δεδομένα.

Η σύγκριση μοντέλων, η πρόωρη διακοπή , το κλάδεμα , η διασταυρούμενη επικύρωση , η τακτοποίηση , οι μπεϋζιανές προκαταβολές ή η εγκατάλειψη είναι κάποιες από τις τεχνικές οι οποίες εφαρμόζονται για την μείωση της πιθανότητας ή της ποσότητας της υπερπροσαρμογής. Οι τεχνικές αυτές βασίζονται στην τιμωρία των υπερβολικά πολύπλοκων (σύνθετων) μοντέλων ή στην εξέταση της ικανότητας γενίκευσης του μοντέλου , κρίνοντας την λειτουργικότητά του σε ένα σύνολο δεδομένων που δεν χρησιμοποιούνται για την εκπαίδευση. Αυτό προσεγγίζει (ως ένα βαθμό) τα τυπικά αόρατα δεδομένα που ενδέχεται να συναντήσει ένα μοντέλο.

Το στατιστικό μοντέλο ή ο αλγόριθμος μηχανικής μάθησης στην περίπτωση της υποπροσαρμογής , δηλαδή του αντιστρόφου της υπερπροσαρμογής , είναι αρκετά απλός ώστε να γίνει ακριβής παρουσίαση των δεδομένων. Όταν υπάρχει μεγάλο ποσοστό μεροληψίας και μικρό ποσοστό διακύμανσης στο μοντέλο ή στον αλγόριθμο που εφαρμόζεται , τότε αυτό είναι μία ένδειξη υποπροσαρμογής. Αντιθέτως στην υπερπροσαρμογή , υπάρχει μικρό ποσοστό μεροληψίας και μεγάλο ποσοστό διακύμανσης. Η ανάλυση ενός μοντέλου ή αλγορίθμου για το σφάλμα μεροληψίας , το σφάλμα διακύμανσης και το μη αναγόμενο σφάλμα , επιτυγχάνεται μέσω της εξισορρόπησης μεροληψίας-διακύμανσης . Η μη ακριβής εμφάνιση των σημείων δεδομένων συμβαίνει όταν στο μοντέλο υπάρχει υψηλή προκατάληψη (μεροληψία) και χαμηλή διακύμανση. Όταν συμβεί αυτό , το μοντέλο αδυνατεί να κάνει προβλέψεις για μεταγενέστερα αποτελέσματα δεδομένων. Η γραμμική γραμμή , σύμφωνα με το Διάγραμμα 1.2 , δεν αναπαριστά όλα τα δεδομένα σημεία καθώς δεν υπάρχει ομοιότητα με την καμπυλότητα των σημείων. Το αναμενόμενο θα ήταν η ύπαρξη μιας γραμμής σε σχήμα παραβολής όπως φαίνεται στο Διάγραμμα 1.2 . Το Διάγραμμα , όπως προαναφέρθηκε , αν χρησιμοποιούνταν για την ανάλυση , τα προγνωστικά αποτελέσματα θα ήταν λανθασμένα και αντίθετα σε σχέση με τα αποτελέσματα όπως θα προέκυπταν μετά από ανάλυση του σχήματος.

Σε ένα μη προσαρμοσμένο μοντέλο , όπως επισημαίνουν οι Burnham & Anderson , δεν θα δινόταν ιδιαίτερη προσοχή σε κάποια αναπαραγωγήσιμη (δηλαδή, εννοιολογικά αναπαραγωγήσιμη στα περισσότερα άλλα δείγματα) δομή στα δεδομένα. Αυτό θα είχε ως αποτέλεσμα την αποτυχία προσδιορισμού των αποτελεσμάτων, τα οποία στηρίζονταν πραγματικά από τα δεδομένα . Η προκατάληψη (μεροληψία), σε αυτή την περίπτωση , διαδραματίζει σημαντικό ρόλο στους εκτιμητές παραμέτρων. Συχνά , λανθασμένα , δεν λαμβάνεται υπόψη η διακύμανση δειγματοληψίας και οι δύο παράγοντες συμβάλλουν σε μη ορθή κάλυψη διαστήματος εμπιστοσύνης. Τα μη εφαρμοσμένα μοντέλα έχουν την τάση απώλειας αξιόλογων αποτελεσμάτων θεραπείας στο πλαίσιο πειραματικών περιβάλλοντων.



Διάγραμμα 1.2 Υποπροσαρμογή-Υπερπροσαρμογή
(https://medium.com/@sarojthapa60_76302/what-is-overfitting-and-how-to-avoid-it-56d4b9ba0688)

ΚΕΦΑΛΑΙΟ 2^ο

Αναπαράσταση Εικόνων με Βαθιά Μάθηση

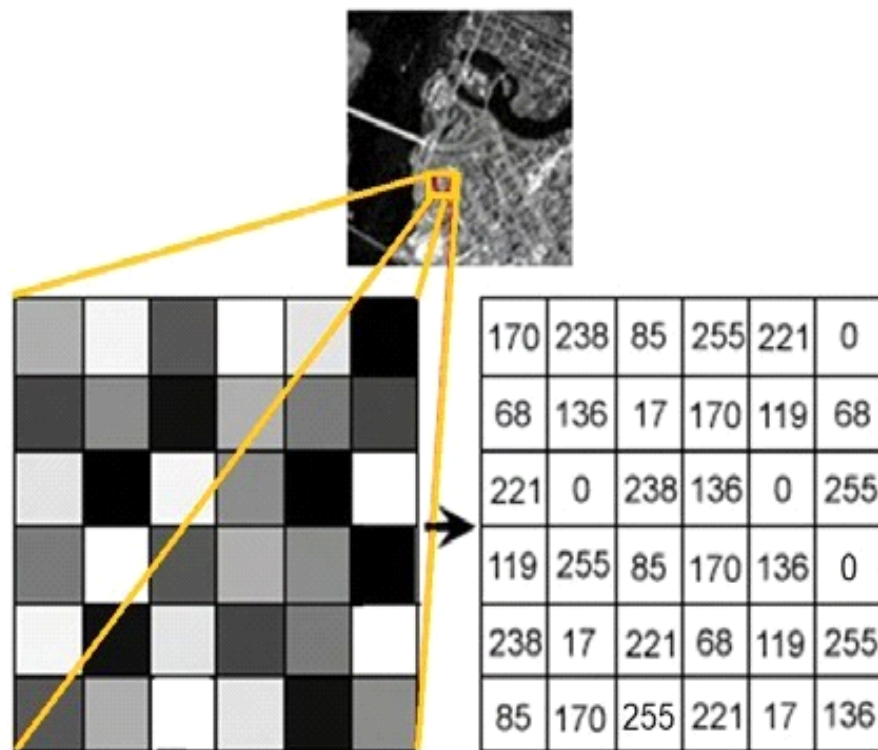
2.1 Αναπαράσταση Εικόνων

Αντικείμενο μελέτης του επιστημονικού κλάδου (πεδίου) της υπολογιστικής όρασης είναι η ικανότητα των υπολογιστών να κατανοούν σε υψηλό επίπεδο ψηφιακές εικόνες και βίντεο. Σημαντικό είναι να γίνει λόγος για τον ορισμό και την αναπαράσταση της εικόνας προτού αναφερθούμε στον τρόπο με τον οποίο ταξινομούνται οι εικόνες. Με άλλα λόγια, οποιαδήποτε εικόνα μπορεί να θεωρηθεί ότι έχει μία συνάρτηση που αντιστοιχεί σε μία συγκεκριμένη θέση, με μία τιμή για κάθε pixel (εικονοστοιχείο).

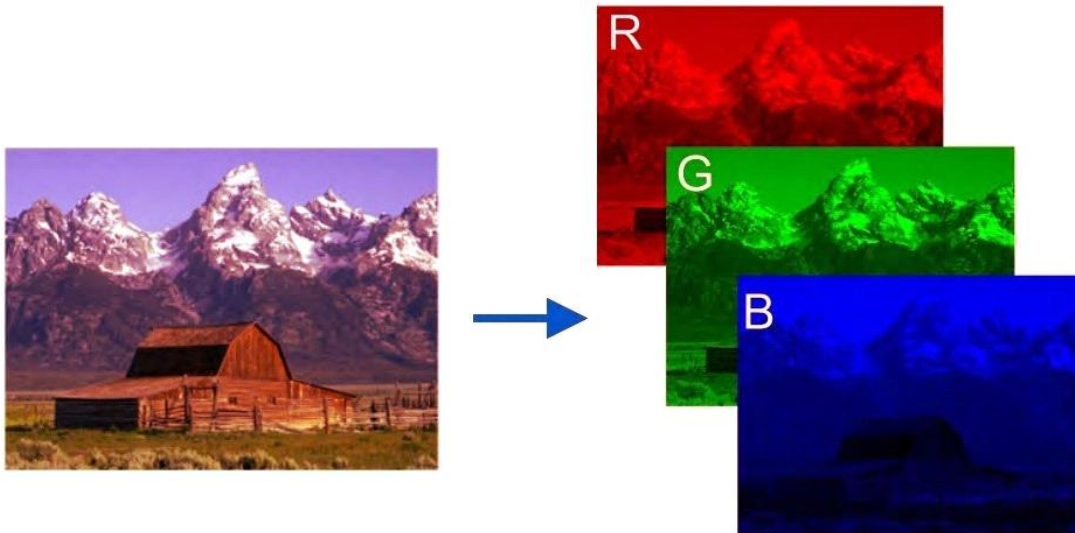
Έτσι, κάθε εικόνα αναπαρίσταται ως μια σειρά από τιμές των pixels. Κάθε εικόνα είναι, δηλαδή, μία συνάρτηση με συγκεκριμένη θέση και τιμή για κάθε pixel. Οπότε, συμπεραίνουμε πως κάθε εικόνα μπορεί να αναπαρασθεί ως ένας πίνακας με τιμές των εικονοστοιχείων. Αν για παράδειγμα, μια εικόνα έχει ανάλυση 1.000x750, τότε το πλάτος της είναι 1,000 εικονοστοιχεία και το ύψος της 750. Η ανάλυση της εικόνας, αν την σκεφτούμε ως πολυδιάστατο πίνακα, περιέχει 1,000 στήλες και 750 σειρές. Στην συγκεκριμένη εικόνα τα εικονοστοιχεία στο σύνολό τους είναι 750,000, δηλαδή 1,000 x 750. Όταν η εικόνα είναι έγχρωμη, τότε αναφερόμαστε στις τρεις εντάσεις των συστατικών: το κόκκινο, το πράσινο και το μπλε. Ένα χρώμα αντιπροσωπεύεται δηλαδή από τις προαναφερθείσες εντάσεις συστατικών. Το εύρος της κλίμακας RGB (Red-Green-Blue) είναι [0,255], με το 0 και το 255 να δηλώνουν καμία (μηδενική) και πλήρη αναπαράσταση αντίστοιχα. Ο καθορισμός ενός χρώματος σύμφωνα με το μοντέλο RGB εξαρτάται από τον καθορισμό της ποσότητας των τριών χρωμάτων που υπάρχουν στο κάθε εικονοστοιχείο, δηλαδή του κόκκινου, του μπλε και του πράσινου. Η έγχρωμη εικόνα αποτελείται από τον συνδυασμό τριών ανεξάρτητων (αυτόνομων) πινάκων W πλάτους και H ύψους. Ένας πίνακας αντιστοιχεί στο καθένα από τρία χρώματα. Από τους τρεις αυτούς πίνακες προκύπτει ένας πίνακας με πολλές διαστάσεις που έχει τη μορφή $W \times H \times D$. Το D ορίζεται ως ο αριθμός των καναλιών ή το βάθος της κλίμακας RGB, $D = 3$. [7]

Σε αντίθεση με τις έγχρωμες απεικονίσεις, οι ασπρόμαυρες αποτελούνται από εικονοστοιχεία με τιμές στην κλίμακα Greyscale. Η Greyscale ορίζεται ως η κλίμακα του γκρι, η οποία αποτελείται από διαβαθμίσεις του λευκού και του μαύρου και το εύρος των τιμών της

είναι από το 0 έως το 255. Στο 0 βρίσκεται το μαύρο ενώ το λευκό στο 255. Μία ασπρόμαυρη φυσική εικόνα απεικονίζεται ψηφιακά , όπως φαίνεται στην Εικόνα 2.1 . Αντίθετα μία έγχρωμη εικόνα απεικονίζεται , όπως παρατηρούμε στην Εικόνα 2.2 , με τα 3 κανάλια χρωμάτων. Σύμφωνα με τον T. Young (1802) , ο συνδυασμός των τριών βασικών χρωμάτων (μπλε , κόκκινο , πράσινο) μπορεί να δημιουργήσει (ή να οδηγήσει στη δημιουργία οποιουδήποτε χρώματος) οποιοδήποτε χρώμα. Η χρωματική αναπαράσταση στηρίζεται σε αυτή τη θεωρία.



Εικόνα 2.1 Αναπαράσταση Εικόνων (ασπρόμαυρη)

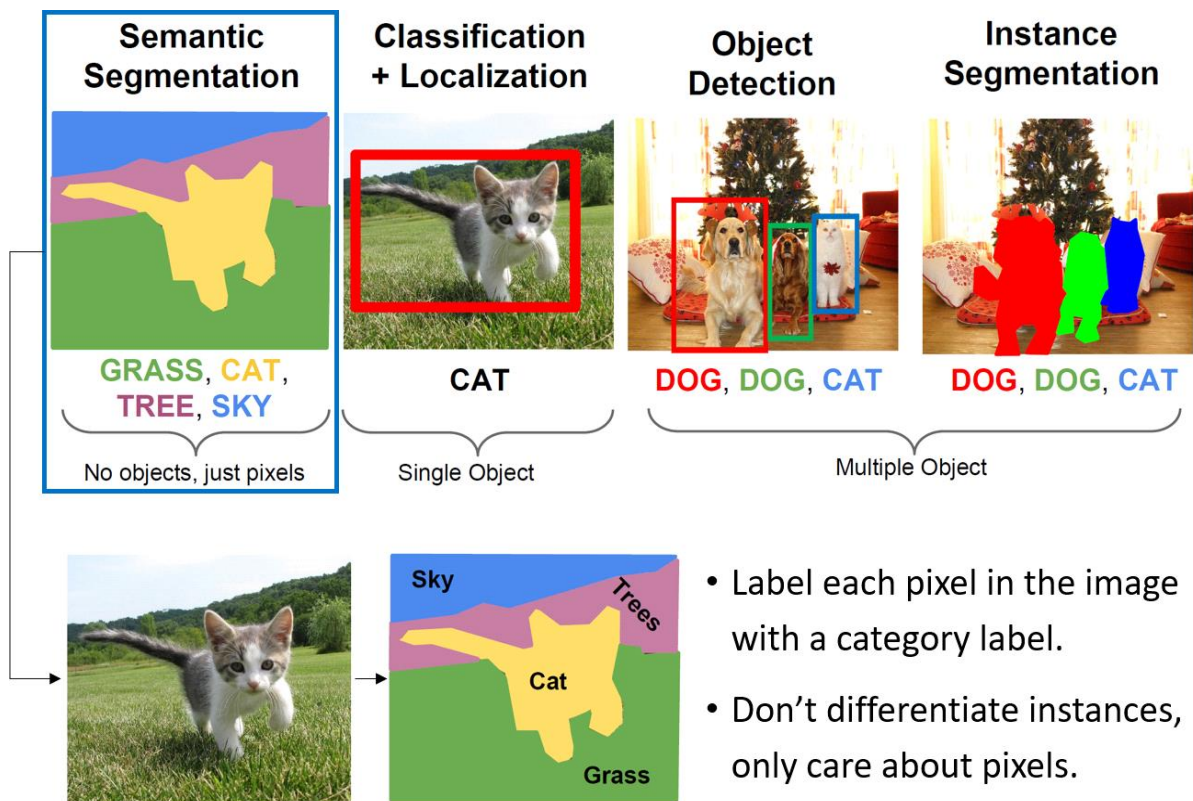


Εικόνα 2.2 Αναπαράσταση Εικόνων (έγχρωμη)

2.1.1 Image Segmentation (Τμηματοποίηση Εικόνας)

Η τμηματοποίηση είναι η διαδικασία διαίρεσης μιας ψηφιακής εικόνας σε πολλαπλά τμήματα ή περιοχές. Ο στόχος της τμηματοποίησης είναι να απλοποιήσει ή/και να αλλάξει την αναπαράσταση μιας εικόνας ώστε να γίνει πιο ουσιαστική και πιο εύκολη στην ανάλυση. Η τμηματοποίηση εικόνας χρησιμοποιείται συνήθως για την ανίχνευση αντικειμένων και ορίων (γραμμές, καμπύλες κ.λπ.) σε εικόνες. Συγκεκριμένα, η τμηματοποίηση εικόνας είναι η διαδικασία εκχώρησης ετικετών σε κάθε εικονοστοιχείο μιας εικόνας, έτσι ώστε τα εικονοστοιχεία με την ίδια ετικέτα να έχουν ορισμένες ιδιότητες.

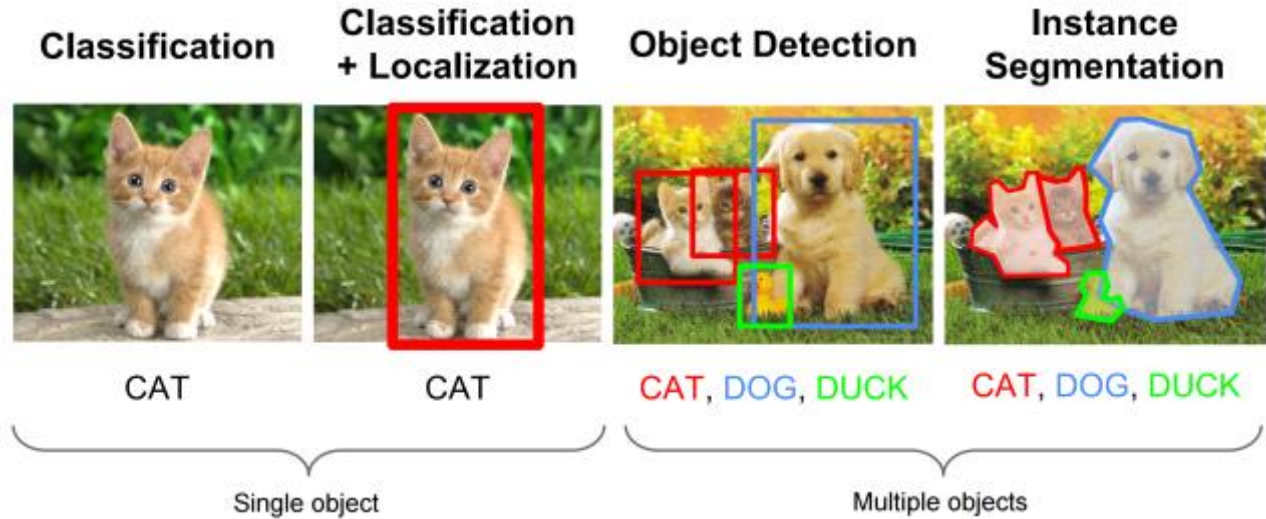
Η τμηματοποίηση μιας εικόνας έχει ως αποτέλεσμα ένα σύνολο τμημάτων που εκτείνονται σε ολόκληρη την εικόνα ή ένα σύνολο περιγραμμάτων που εξάγονται από την εικόνα. Κάθε εικονοστοιχείο σε μια περιοχή είναι παρόμοιο σε σχέση με κάποια χαρακτηριστική ή υπολογισμένη ιδιότητα, όπως το χρώμα, η ένταση ή η υφή. Οι γειτονικές περιοχές έχουν σημαντικά διαφορετικό χρώμα σε σχέση με τα ίδια χαρακτηριστικά. Όταν εφαρμόζεται σε μια στοίβα εικόνων, συνήθως στην ιατρική απεικόνιση, τα προκύπτοντα περιγράμματα μετά την τμηματοποίηση μπορούν να χρησιμοποιηθούν για τη δημιουργία τρισδιάστατων ανακατασκευών της εικόνας με τη βοήθεια αλγορίθμων παρεμβολής, όπως οι τρισδιάστατοι κύβοι. [8]



Εικόνα 2.3 Image segmentation
https://oi.readthedocs.io/en/latest/computer_vision/segmentation/segmentation.html

2.1.2 Image Processing (Επεξεργασία Εικόνας)

Η ψηφιακή επεξεργασία εικόνας είναι η χρήση των υπολογιστών για την επεξεργασία εικόνων χρησιμοποιώντας αλγορίθμους. Αυτό μπορεί να είναι κατάπτωση ή πεδίο επεξεργασίας ψηφιακών γραφικών, καθώς η ψηφιακή επεξεργασία αρχείων παρτίδας έχει πολλά πλεονεκτήματα σε σχέση με το αναλογικό αρχείο. Επιτρέπει την εφαρμογή ενός πολύ ευρύτερου φάσματος αλγορίθμων στα δεδομένα εισόδου και μπορεί να αποφύγει προβλήματα όπως η δημιουργία θορύβου και παραμόρφωσης κατά την επεξεργασία. Δεδομένου ότι οι εικόνες ορίζονται σε δύο διαστάσεις (ίσως περισσότερες), η ψηφιακή επεξεργασία εικόνας μπορεί να μοντελοποιηθεί με τη μορφή πολυδιάστατων συστημάτων. Η δημιουργία και η ανάπτυξη της ψηφιακής επεξεργασίας εικόνας επηρεάζεται κυρίως από τρεις παράγοντες: πρώτον, την ανάπτυξη των υπολογιστών. Δεύτερον, η ανάπτυξη των μαθηματικών (ιδιαίτερα η δημιουργία και η βελτίωση της μαθηματικής διακριτής θεωρίας). Τρίτον, υπάρχει αύξηση της ζήτησης για ένα ευρύ φάσμα εφαρμογών σε τομείς όπως το περιβάλλον, η γεωργία, ο στρατός, η βιομηχανία και η ιατρική.



Εικόνα 2.4 Image Processing (<https://medium.com/analytics-vidhya/weakly-supervised-learning-for-object-localization-4b73d4f4f4a6>)

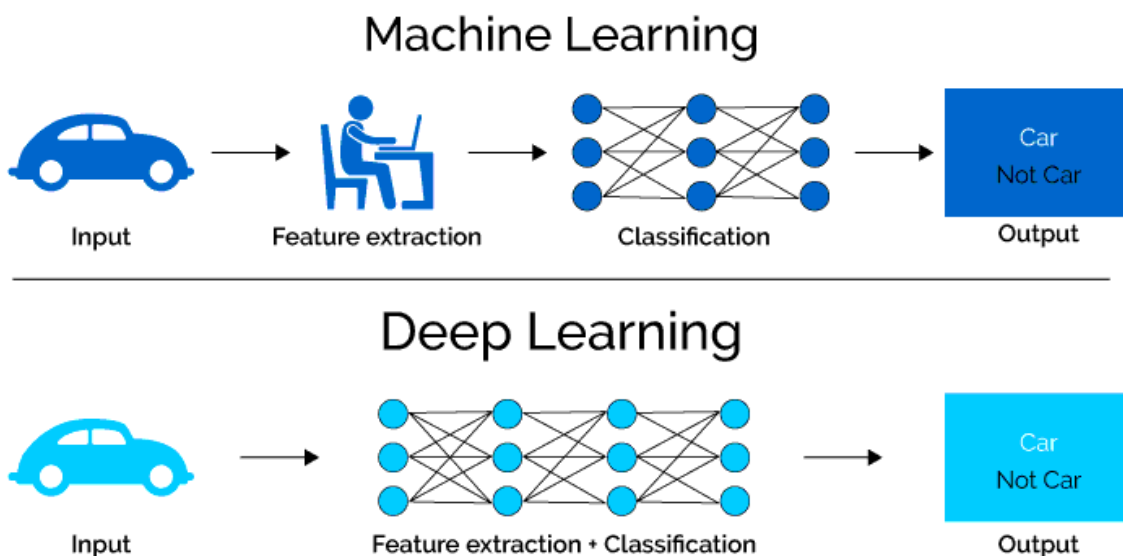
2.2 Βαθιά Μάθηση

Με την πάροδο του χρόνου, το κόστος του υλικού έχει μειωθεί σημαντικά και η ανάπτυξη των GPU (κάρτες γραφικών) έχει βελτιωθεί σημαντικά. Αυτό οδήγησε στην ανάπτυξη μιας νέας προσέγγισης στα προβλήματα μηχανικής μάθησης που ονομάζεται Βαθιά Μάθηση (Deep Learning). Ουσιαστικά, αυτή η προσέγγιση επιχειρεί να μοντελοποιήσει πώς λειτουργεί ο ανθρώπινος εγκέφαλος. Δηλαδή, προσπαθεί να μοντελοποιήσει πώς ο ανθρώπινος εγκέφαλος επεξεργάζεται τον ήχο και το φως και τα μετατρέπει σε ακοή και όραση, αντίστοιχα.

Για αναλυτικές εργασίες, η βαθιά μάθηση προχωρά ακόμη περισσότερο, επιτρέποντας στους υπολογιστές να «σκέφτονται» λύσεις μόνοι τους, χωρίς ανθρώπινη βοήθεια. Ας ρίξουμε μια πιο προσεκτική ματιά στο τι ακριβώς σημαίνει αυτό. Η παραδοσιακή προσέγγιση στην ανάλυση είναι η χρήση των διαθέσιμων δεδομένων για τη δημιουργία μεταβλητών, την επιλογή μοντέλων εκπαίδευσης και, τέλος, την αξιολόγηση της απόδοσης του μοντέλου. Δεδομένου ότι η πληρότητα και η ακρίβεια του μοντέλου εξαρτώνται από την ποιότητα του μοντέλου και τις παραμέτρους του, αυτές οι μέθοδοι είναι αποτελεσματικές σε συστήματα πρόβλεψης που δεν απαιτούν γενίκευση. Για παράδειγμα, όταν αναπτύσσετε ένα μοντέλο ανίχνευσης απάτης, εκπαιδεύεται πρώτα στο διαθέσιμο σύνολο δεδομένων και, στη συνέχεια, προσθέτει νέα δεδομένα. Αν συνεχιστεί αυτό, το μοντέλο θα έχει χιλιάδες μεταβλητές, καθεμία από τις οποίες έχει νόημα και δεν επηρεάζει το μοντέλο. Επομένως, το μοντέλο θα πρέπει να ανακατασκευαστεί με βάση τα νέα δεδομένα. Για να προστεθούν νέα δεδομένα, πρέπει να εκτελεστεί ξανά η ίδια ενέργεια. [9]

Οι προσεγγίσεις βαθιάς μάθησης, από την άλλη πλευρά, αντικαθιστούν τυποποιήσεις στατικών μοντέλων μάθησης που βασίζονται σε ιεραρχικά επίπεδα που μαθαίνουν να αναγνωρίζουν κρυφά δεδομένα. Η βαθιά μάθηση είναι ένα δυναμικό μοντέλο μάθησης που οδηγεί σε συστήματα πρόβλεψης που μπορούν να γενικευθούν καθώς αυτοπροσαρμόζονται σε νέα δεδομένα και βελτιώνονται συνεχώς, καθιστώντας το μια πιο δυναμική προσέγγιση από τα συστήματα πρόβλεψης που βασίζονται σε κανόνες. Στην ουσία εκπαιδεύεται η διαδικασία και όχι το μοντέλο.

Οι προσεγγίσεις βαθιάς μάθησης βασίζονται σε μοντέλα μηχανικής μάθησης, τα νευρωνικά δίκτυα. Με άλλα λόγια, η βαθιά μάθηση είναι μέρος του τομέα της μηχανικής μάθησης και μέρος του τομέα της τεχνητής νοημοσύνης. Οι αλγόριθμοι βαθιάς μάθησης συνήθως χρησιμοποιούν πολλά επίπεδα με τη μορφή δικτύων τεχνητής νοημοσύνης. Οι μέθοδοι μάθησης βασίζονται σε αναπαραστάσεις δεδομένων. Ως μέθοδος, εμφανίστηκε για πρώτη φορά στο έργο του Ivakhnenko το 1965. Ο όρος προτάθηκε το 1986 από τους Fukushima και Jan LeCun το 1989 ως εφαρμογή στη Μηχανική Όραση. Ωστόσο, η μαθηματική κοινότητα το απέρριψε λόγω της πολυπλοκότητάς του, θεωρώντας αδύνατη την εφαρμογή του στην πράξη και άρχισε να προσελκύει ερευνητικό ενδιαφέρον. Αργότερα αναπτύχθηκαν γνωστοί αλγόριθμοι όπως οι CNN και RNN (Recurrent Neural Network, Convolutional Neural Network). Οι εφαρμογές του είναι πλέον ευρέως διαδεδομένες και αυτοί οι αλγόριθμοι έχουν επιτύχει πολύ υψηλές επιδόσεις. Τομείς όπου η βαθιά μάθηση υπερέχει είναι κυρίως η Επεξεργασία Φυσικής Γλώσσας, η Βιοπληροφορική και τα Συστήματα Συστάσεων. Οι αλγόριθμοι Βαθιάς Μάθησης ξεχωρίζουν για την απόδοσή τους και επιτυγχάνουν τις υψηλότερες αποδόσεις σε πρόσφατους διαγωνισμούς και δοκιμές. [10]



Εικόνα 2.5 Μηχανική Μάθηση-Βαθιά Μάθηση
(<https://www.csc.com.gr/machine-learning->

%CE%BC%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE-%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9/)

Η Βαθιά Μάθηση ταξινομείται σε τρεις διαφορετικές κατηγορίες με βάση τις μεθόδους μάθησης. [11]

- **Επιβλεπόμενη μάθηση:** Στην επιβλεπόμενη μάθηση, τα δεδομένα εισόδου περιέχουν ένα σύνολο χαρακτηριστικών και ετικετών. Υπάρχει συσχέτιση μεταξύ ιδιοτήτων και ετικετών. Χρησιμοποιώντας αυτόν τον συσχετισμό, ο αλγόριθμος σε αυτό το μοντέλο πρέπει να μάθει τους γενικούς κανόνες για να συσχετίσει νέα χαρακτηριστικά με τη σωστή ετικέτα.
- **Μη-Επιβλεπόμενη μάθηση:** Στη μη-επιβλεπόμενη μάθηση, τα δεδομένα δεν συνδέονται απαραίτητα με ετικέτες. Αυτά τα μοντέλα θα πρέπει να μπορούν να εξάγουν σημαντικές προβολές από τα ίδια τα δεδομένα χωρίς προηγούμενη εμπειρία με τα δεδομένα.
- **Ενισχυτική Μάθηση:** Στο μοντέλο αυτό, υπάρχει αλληλεπίδραση μεταξύ του υπολογιστικού συστήματος και του δυναμικού περιβάλλοντος. Ο στόχος του αλγορίθμου σε αυτό το μοντέλο είναι να πλησιάσει τον στόχο χωρίς καμία εμπειρία ή οδηγίες (για παράδειγμα, οδήγηση αυτοκινήτου, παιχνίδι εναντίον ενός αντιπάλου κ.λπ.).

Παραδείγματα μοντέλων Επιβλεπούμενης Μάθησης είναι τα συνελκτικά νευρωνικά δίκτυα (CNN), τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) και τα δίκτυα τροφοδοσίας, όπως η μακροπρόθεσμη βραχυπρόθεσμη μνήμη (LSTM). Ένα παράδειγμα ενός μοντέλου μάθησης χωρίς επίβλεψη είναι το Deep Belief Network (DBN), το οποίο είναι ουσιαστικά ένα σύνολο περιορισμένων μηχανών Boltzmann (RBM). Τέλος, η Ενισχυτική Μάθηση βασίζεται στη βελτιστοποίηση χαρακτηριστικών της συνάρτησης. Ένα παράδειγμα τέτοιου μοντέλου είναι το Deep Q-Network (DQN). [12] Με άλλα λόγια, ένα Νευρωνικό Δίκτυο μπορεί να συγκριθεί με ένα μαύρο κουτί στο ότι ο χρήστης εισάγει δεδομένα και το μοντέλο πέρνει αποφάσεις.

2.2.1 Ρηγά και Βαθιά δίκτυα

Αποδείχθηκε ότι ένα νευρωνικό δίκτυο με τουλάχιστον ένα κρυφό στρώμα μπορεί να θεωρηθεί ως γενική προσέγγιση. Για παράδειγμα, ένα τέτοιο νευρωνικό δίκτυο μπορεί θεωρητικά να αντιπροσωπεύει οποιαδήποτε λειτουργία. Ωστόσο, αυτό έρχεται σε αντίθεση με τους περιορισμούς των νευρωνικών δικτύων χωρίς κρυφά επίπεδα. Συγκεκριμένα, τα μονοστρωματικά νευρωνικά δίκτυα, όπως τα perceptrons, είναι απλοί γραμμικοί ταξινομητές και επομένως είναι αναποτελεσματικοί στην εκπαίδευση ενός ευρέος φάσματος εργασιών. Για παράδειγμα, τέτοια δίκτυα ενός επιπέδου δεν μπορούν να μάθουν να μοντελοποιούν απλά

χαρακτηριστικά όπως η συνάρτηση XOR ή άλλα προβλήματα ταξινόμησης που δεν μπορούν να διαχωριστούν γραμμικά. Στην πράξη, ωστόσο, τα δίκτυα με ένα μόνο κρυφό επίπεδο, ακόμη και τα πολύ μεγάλα δίκτυα, μπορούν να μάθουν να αναπαριστούν σύνθετα χαρακτηριστικά με τον ίδιο τρόπο όπως τα δίκτυα με πολλά κρυφά επίπεδα. Στην πραγματικότητα, η παρατήρηση ότι τα δίκτυα με πολλά κρυφά επίπεδα ή βαθιά δίκτυα επιτυγχάνουν υψηλότερη ακρίβεια από τα δίκτυα με λίγα κρυφά επίπεδα (όπως τα ρηχά δίκτυα) είναι μια από τις τελευταίες εξελίξεις στην εκπαίδευση νευρωνικών δικτύων.

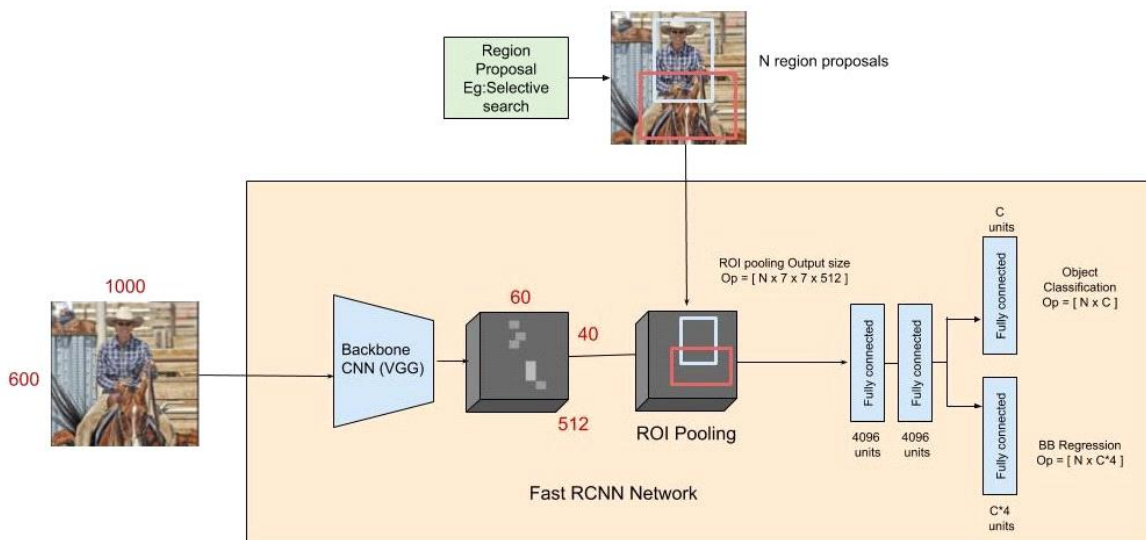
2.3 Μοντέλα βαθιάς μάθησης

Η Βαθιά Μάθηση ανήκει σε μια ευρύτερη οικογένεια μεθόδων μηχανικής μάθησης που βασίζονται σε τεχνητά νευρωνικά δίκτυα με λειτουργία εκμάθησης. Σε αυτό το πλαίσιο, η μάθηση μπορεί να ταξινομηθεί ως εποπτευόμενη, ημι-εποπτευόμενη και χωρίς επίβλεψη. Τα περισσότερα μοντέλα βαθιάς μάθησης βασίζονται σε τεχνητά νευρωνικά δίκτυα, κυρίως σε Συνελκτικά Νευρωνικά Δίκτυα (CNN), καθώς και σε κόμβους δικτύων βαθιάς εμπιστοσύνης (deep belief networks) και σε μηχανές βαθιάς Boltzmann. Στη βαθιά μάθηση, κάθε επίπεδο μαθαίνει πώς να μετατρέπει τα δεδομένα εισόδου σε πιο αφηρημένες και σύνθετες αναπαραστάσεις. Για εφαρμογές αναγνώρισης εικόνας, η ακατέργαστη είσοδος μπορεί να είναι μια σειρά pixel. Το πρώτο επίπεδο αναπαράστασης εξάγει τα εικονοστοιχεία και κωδικοποιεί τις γωνίες, το δεύτερο συνδυάζει και κωδικοποιεί μια σειρά γωνιών, το τρίτο κωδικοποιεί τη μύτη και τα μάτια και το τέταρτο επίπεδο περιέχει τα πρόσωπα της εικόνας. Είναι σημαντικό να σημειωθεί ότι η ίδια η διαδικασία βαθιάς μάθησης μπορεί να μάθει πού και σε ποιο επίπεδο να τοποθετεί τα χαρακτηριστικά με τον καλύτερο τρόπο. Φυσικά, αυτό δεν σημαίνει ότι δεν χρειάζεται σκληρός συντονισμός. Αυτό είναι επίσης εμφανές από το γεγονός ότι ο διαφορετικός αριθμός στρωμάτων και τα διαφορετικά μεγέθη στρώματος μπορεί να έχουν ως αποτέλεσμα διαφορετικές μειώσεις. Η λέξη "βάθος" στο "deep learning" αναφέρεται στον αριθμό των επιπέδων μέσω των οποίων μετασχηματίζονται τα δεδομένα. Συγκεκριμένα, τα συστήματα βαθιάς μάθησης έχουν ένα βάθος βάσης που καθορίζεται από την διαδρομή ανάθεσης πίστωσης (CAP). Το CAP είναι μια ακολουθία μετασχηματισμών από την είσοδο στην έξοδο. Το CAP αντιπροσωπεύει πιθανές αιτιώδεις σχέσεις μεταξύ εισροών και εκροών. Για τα δίκτυα πρόσθιας τροφοδότησης, το βάθος CAP είναι το βάθος του δικτύου συν ένα συν τον αριθμό των κρυφών επιπέδων (το επίπεδο εξόδου είναι διαμορφώσιμο). Για τα αναδρομικά νευρωνικά δίκτυα, όπου το σήμα μπορεί να περάσει μέσα από τα στρώματα πολλές φορές, το βάθος CAP είναι πιθανώς απεριόριστο. Η βαθιά μάθηση είναι η χρήση εκθετικών συναρτήσεων για την αύξηση της ισχύος των δεδομένων. Ο τρόπος που χρησιμοποιείται αυτή η τιμή είναι ο αλγόριθμος εκπαίδευσης, αλλά η ανάθεση πίστωσης είναι συνάρτηση της εργασίας με τα βάρη και χρησιμοποιείται για την ενημέρωση των βαρών αργότερα. Όταν το CAP είναι μικρότερο από 2 θεωρείται ρηχή μάθηση (shallow learning). Η βαθιά μάθηση ($CAP > 2$) έχει αποδειχθεί ως ένας εκτιμητής που μπορεί να χρησιμοποιηθεί για την αξιολόγηση αυθαίρετων χαρακτηριστικών καθώς και για την αναπαραγωγή ρηχών δικτύων μάθησης. Οι αλγόριθμοι βαθιάς μάθησης μπορούν επίσης να εφαρμοστούν σε προβλήματα μάθησης χωρίς επίβλεψη, όπως αυτά που χρησιμοποιούνται για την αποκάλυψη

φαινομένων φυσικής γλώσσας. Ένα χαρακτηριστικό παράδειγμα είναι το δίκτυο βαθιάς εμπιστοσύνης.

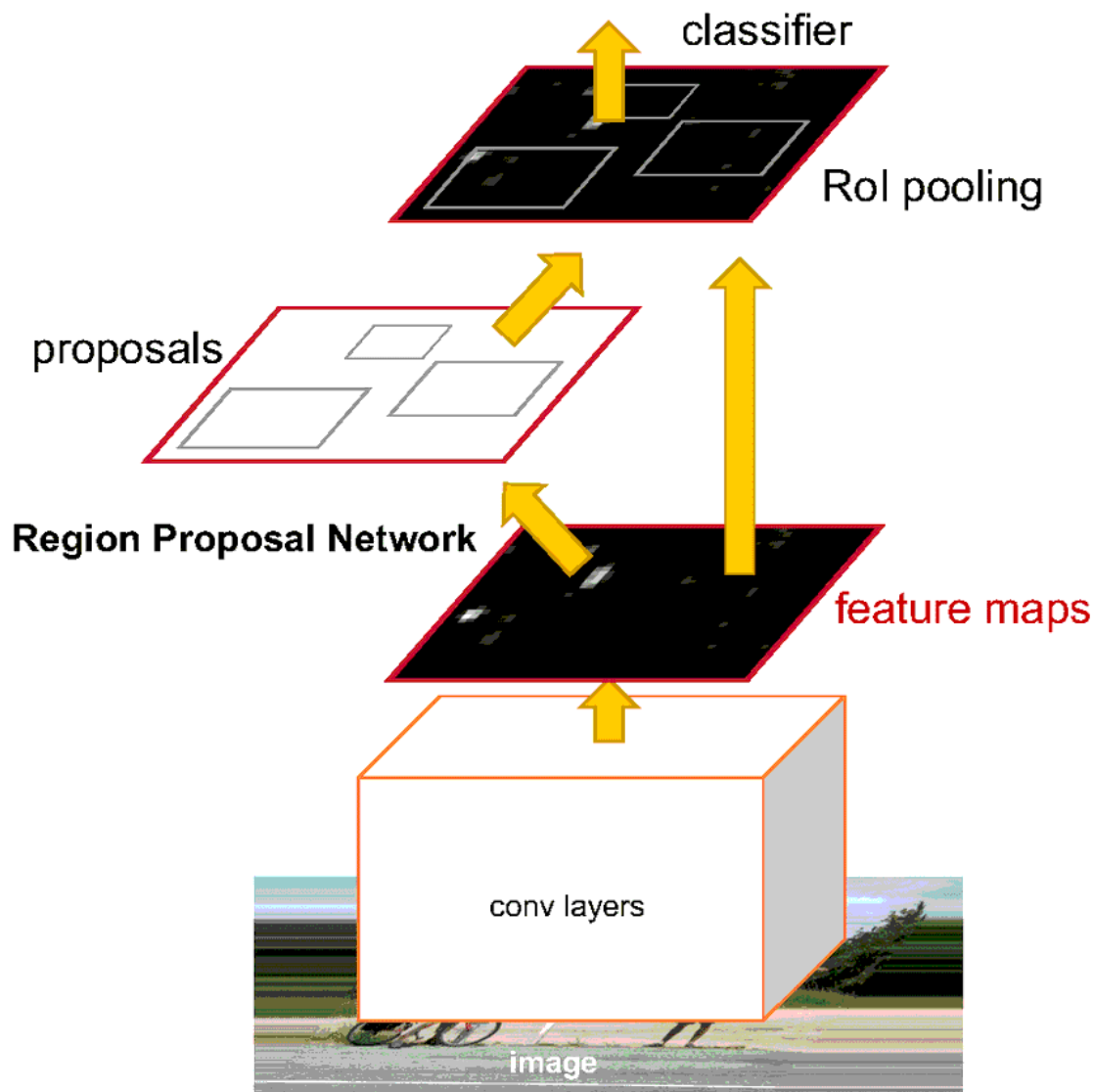
- **Fast RCNN**, τα Deep ConvNets χρησιμοποιήθηκαν για την αναγνώριση των αντικειμένων. Έχει λίγα μειονεκτήματα καθώς η εκπαίδευση είναι ακριβή στο διάστημα, η ανίχνευση αντικειμένων είναι αργή τη στιγμή της εξαγωγής συμπερασμάτων.

Εικόνα --> DeepConv (χάρτης δυνατοτήτων) --> διάνυσμα χαρακτηριστικών απόδοσης επένδυσης (ROI) --> παλινδρόμηση πλαισίου οριοθέτησης (SVM και DeepConv)



Διάγραμμα 2.1 Faster RCNN Δίκτυο
(<https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022>)

- **Faster RCNN**, συνδυασμός τεχνικής RCNN και Fast RCNN. Εισήχθη ένα νέο στοιχείο που ονομάζεται δίκτυο προτάσεων περιοχής (RPN).
Εικόνα --> συνέλιξη (χάρτης χαρακτηριστικών) --> RPN --> ROI --> οριοθέτηση αναδρομικού πλαισίου



Εικόνα 2.6 Αρχιτεκτονική Faster RCNN
<https://paperswithcode.com/lib/torchvision/faster-r-cnn>

- Το **Mask RCNN** είναι η επέκταση του Faster RCNN.

Το Masked R-CNN είναι απλό σαν έννοια. Το Faster R-CNN έχει δύο εξόδους για κάθε υποψήφιο αντικείμενο: μια ετικέτα κλάσης και μια μετατόπιση πλαισίου οριοθέτησης. Προσθέτουμε σε αυτό έναν τρίτο κλάδο που βγάζει τη μάσκα του αντικειμένου. Έτσι, το Mask R-CNN είναι μια φυσική και διαισθητική ιδέα. Ωστόσο, η έξοδος της πρόσθετης μάσκας διαφέρει από την έξοδο μιας κλάσης στο ότι πρέπει να ανακτήσει μια πιο ακριβή χωρική θέση του αντικειμένου. Στη συνέχεια, παρουσιάζουμε τα βασικά στοιχεία του Mask R-CNN,

συμπεριλαμβανομένης της ευθυγράμμισης pixel-to-pixel, ένα βασικό στοιχείο που λείπει από το Fast/Faster R-CNN.

Ξεκινήσαμε με μια σύντομη επισκόπηση του Faster R-CNN ανιχνευτή. Το Faster R-CNN αποτελείται από δύο στάδια. Το πρώτο στάδιο, ονομάζεται Δίκτυο Προτάσεων Περιοχής (RPN), προτείνει πλαίσια οριοθέτησης για υποψήφια χαρακτηριστικά. Το δεύτερο στάδιο είναι βασικά το Fast R-CNN που χρησιμοποιεί το RoIPool για την εξαγωγή χαρακτηριστικών από κάθε υποψήφιο πλαίσιο και την εκτέλεση ταξινόμησης και παλινδρόμησης. Οι συναρτήσεις που χρησιμοποιούνται και στα δύο βήματα μπορούν να χρησιμοποιηθούν μαζί για να επιταχυνθεί η εξαγωγή συμπερασμάτων. Το Mask R-CNN χρησιμοποιεί την ίδια διαδικασία δύο σταδίων με ένα πανομοιότυπο πρώτο βήμα (το οποίο είναι το RPN). Στο δεύτερο στάδιο, μαζί με την πρόβλεψη αλλαγής κλάσης και περιβάλλοντος, το Mask R-CNN εξάγει επίσης μια δυαδική μάσκα για κάθε περιοχή ενδιαφέροντος RoI. Αυτό είναι διαφορετικό από τα πρόσφατα συστήματα όπου η ταξινόμηση βασίζεται σε προβλέψεις μάσκας. Η προσέγγισή μας ακολουθεί το πνεύμα του Fast R-CNN, το οποίο εφαρμόζει την παλινδρόμηση και την ταξινόμηση οριοθέτησης παράλληλα (κάτι που διαπιστώσαμε ότι απλοποιεί πολύ τον αγωγό πολλαπλών σταδίων του αρχικού R-CNN). Επιπλέον, παρακάτω ορίζουμε την απώλεια πολλαπλών εργασιών για κάθε επιλεγμένη περιοχή ενδιαφέροντος RoI ως $L = L_{cls} + L_{box} + L_{mask}$. Η απώλεια ταξινόμησης L_{cls} και η απώλεια εικόνας L_{box} είναι τα ίδια με αυτά που ορίζονται. Ο κλάδος μάσκας έχει μέγεθος εξόδου $Km2$ για κάθε RoI, που κωδικοποιεί μια δυαδική μάσκα K με ανάλυση $m \times m$ για κάθε μια από τις κλάσεις K . Ορίζουμε το L_{mask} ως απώλεια εντροπίας και εφαρμόζουμε ένα σιγμοειδές ανά εικονοστοιχείο. Για κάθε RoI το οποίο σχετίζεται με την κλάση βασικής αλήθειας k , τότε η L_{mask} ορίζει την k -η mask (καμία άλλη έξοδος μάσκας δεν επηρεάζει την απώλεια). Ο ορισμός του L_{mask} επιτρέπει στο δίκτυο να δημιουργήσει μια μάσκα για κάθε κλάση χωρίς διενέξεις μεταξύ των κλάσεων. Βασιζόμαστε στις αποκλειστικές υποδιαιρέσεις της ταξινόμησης για να προβλέψουμε τις ετικέτες κλάσεων που χρησιμοποιούνται για την επιλογή της μάσκας εξόδου. Αυτό διαχωρίζει τη μάσκα και την πρόβλεψη τάξης. Αυτός είναι διαφορετικός από τον συνήθη τρόπο εφαρμογής του FCN (Fully Convolutional Network) στη σημασιολογική τμηματοποίηση. Η σημασιολογική τμηματοποίηση χρησιμοποιεί συνήθως softmax ανά pixel και πολυωνυμική απώλεια διασταυρούμενης εντροπίας. Σε αυτήν την περίπτωση, οι μάσκες όλων των τάξεων θα συγκρούονται. [13]

2.4 CNN - Convolutional Neural Networks

Τα Συνελικτικά Νευρωνικά Δίκτυα χρησιμοποιούνται κυρίως για εργασίες που σχετίζονται με εικόνες 2D, αλλά αυτό δεν επηρεάζει τη λειτουργικότητά τους σε τρισδιάστατες εικόνες. Οι κύριες λειτουργίες αυτού του δικτύου είναι: Πρώτον, στο Επίπεδο Εισόδου έχουμε ως είσοδο ένα στρώμα εισόδου, συνήθως ένα δισδιάστατο πίνακα που αντιστοιχεί σε pixel. Αυτή η είσοδος περνά μέσα από μια σειρά από κρυφά επίπεδα μέχρι να επιτευχθεί μια έξοδος.

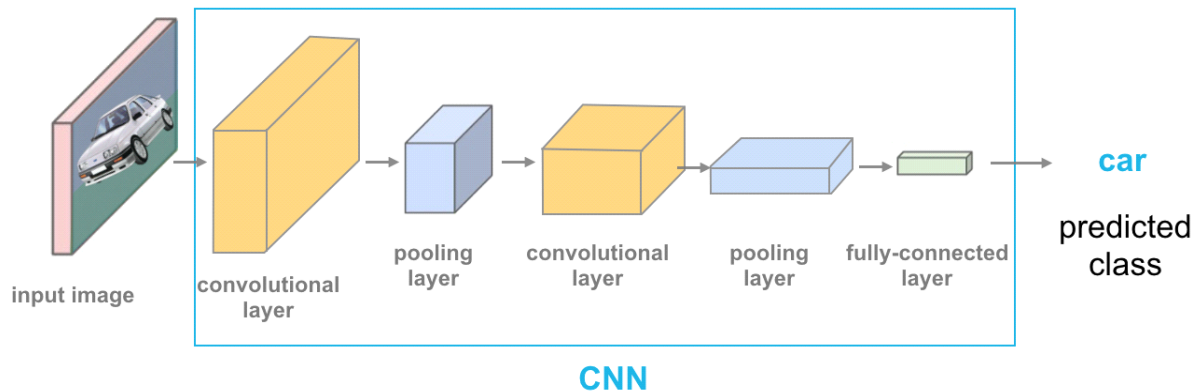
Το πρώτο κρυφό επίπεδο και το δεύτερο είναι το Συνελικτικό Επίπεδο. Εδώ έχουμε ένα κρυφό στρώμα όπου το άθροισμα υπολογίζεται από τα βάρη των νευρώνων. Αυτό το άθροισμα είναι η είσοδος στη συνάρτηση ενεργοποίησης, της οποίας η έξοδος είναι η έξοδος

του νευρώνα. Η πιο διάσημη και ευρέως χρησιμοποιούμενη συνάρτηση είναι η ReLu (Rectified Linear Unit). Υπάρχουν συνήθως πολλά επίπεδα συνέλιξης.

Το τρίτο επίπεδο είναι το Συγκεντρωτικό Επίπεδο, το οποίο στοχεύει στην απλοποίηση των πληροφοριών που εξάγονται από τα προηγούμενα επίπεδα. Τα επίπεδα συγκέντρωσης λειτουργούν με τον ίδιο τρόπο όπως τα συνελκτικά στρώματα και η πιο διάσημη συνάρτηση που χρησιμοποιείται είναι η συνάρτηση μέγιστου max pooling.

Το τελευταίο κρυφό στρώμα είναι συνήθως στο Πλήρως Συνδεδεμένο Επίπεδο. Εδώ υπάρχει ένας νευρώνας που αποτελείται από λειτουργίες ενεργοποίησης και είναι υπεύθυνος για τη συλλογή πληροφοριών και τη σωστή ταξινόμηση τους.

Τέλος, βρίσκεται το Επίπεδο Εξόδου. Υπάρχουν τόσοι νευρώνες σε αυτό το επίπεδο όσες και διαφορετικές κατηγορίες πληροφοριών. Βασικά, σε αυτό το επίπεδο, οι πληροφορίες μετατρέπονται σε πιθανότητες και το μοντέλο αξιολογείται με βάση αυτές τις πιθανότητες. [14]



Διάγραμμα 2.2 CNN
([https://cezannec.github.io/Convolutional Neural Networks/](https://cezannec.github.io/Convolutional_Neural_Networks/))

2.4.1 Ιστορική Αναδρομή

Τα Συνελκτικά Νευρωνικά Δίκτυα (LeCun, 1989), γνωστά και ως CNN, είναι ένας ειδικός τύπος νευρωνικού δικτύου για την επεξεργασία δεδομένων με τοπολογίες τύπου πλέγματος. Είναι εμπνευσμένο από τη βιολογία επειδή ο τρόπος που συνδέονται οι νευρώνες είναι παρόμοιος με τον οπτικό φλοιό των ζώων. Οι μεμονωμένοι νευρώνες ανταποκρίνονται σε ερεθίσματα μόνο σε μια περιορισμένη περιοχή του οπτικού πεδίου, γνωστή ως δεκτικό πεδίο. Οι Hubel και Wiesel ανακάλυψαν για πρώτη φορά τοπικά ευαίσθητους νευρώνες και δεκτικά πεδία στο οπτικό σύστημα της γάτας το 1960. Ανακάλυψαν δύο τύπους κυττάρων στον πρωτογενή οπτικό φλοιό, που ονομάζονται απλά κύτταρα και πολύπλοκα κύτταρα, και πρότειναν ένα μοντέλο για τους δύο τύπους κυττάρων σε εργασίες αναγνώρισης προτύπων.

Στη συνέχεια, το 1980, ο Fukushima εισήγαγε το Neocognitron, εμπνευσμένο από τις παραπάνω ιδέες. Είναι ένα ιεραρχικό πολυστρωματικό τεχνητό νευρωνικό δίκτυο. Το Neocognitron αποτελείται από διάφορους τύπους κυττάρων, μερικά από τα οποία είναι

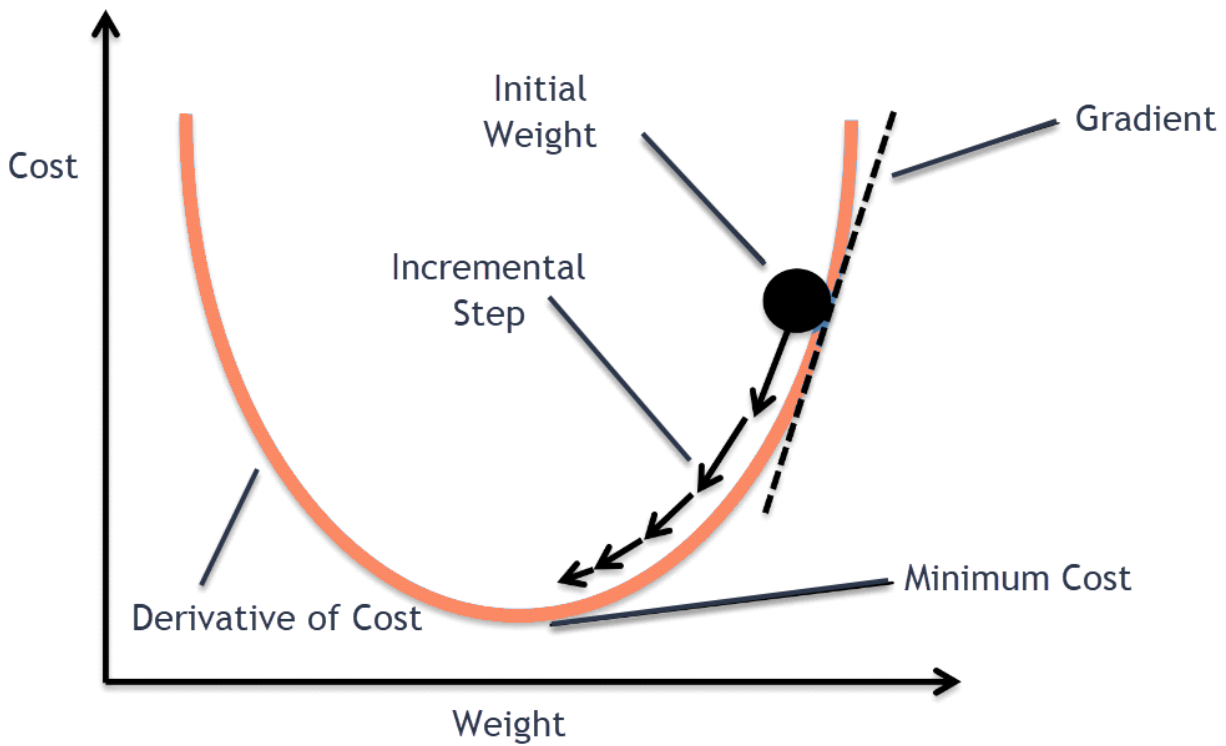
υπεύθυνα για την εξαγωγή τοπικών χαρακτηριστικών και άλλα για την παραμόρφωση χαρακτηριστικών, όπως η τοπική μετατόπιση. Τα τοπικά στοιχεία εισόδου ενσωματώνονται σταδιακά και ταξινομούνται στο υψηλότερο επίπεδο. Η ιδέα της ενσωμάτωσης τοπικών χαρακτηριστικών μπορεί επίσης να βρεθεί σε μοντέλα συνελκτικών νευρωνικών δικτύων.

Τα συνελκτικά νευρωνικά δίκτυα (CNN) είναι μια κατηγορία βαθιών νευρωνικών δικτύων που χρησιμοποιούνται συνήθως για την ανάλυση οπτικών εικόνων. Το όνομα ενός συνελκτικού νευρωνικού δικτύου υποδηλώνει ότι χρησιμοποιεί μια μαθηματική πράξη που ονομάζεται συνέλιξη. Η συνέλιξη είναι ένας ειδικός τύπος γραμμικής λειτουργίας. Ένα συνελκτικό νευρωνικό δίκτυο είναι ένα απλό νευρωνικό δίκτυο που χρησιμοποιεί συνέλιξη με τουλάχιστον ένα επίπεδο κοινού πολλαπλασιασμού. Τα συνελκτικά νευρωνικά δίκτυα είναι κανονικοποιημένες εκδόσεις πολυστρωματικών perceptrons και είναι συνήθως πλήρως διασυνδεδεμένα. Αυτό το χαρακτηριστικό τα καθιστά πιο ευάλωτα σε φαινόμενα υπερπροσαρμογής. Ένας τυπικός τρόπος αντιμετώπισης αυτού του φαινομένου περιλαμβάνει τη στάθμιση της συνάρτησης απώλειας (loss function). Ωστόσο, τα Συνελκτικά Νευρωνικά Δίκτυα ακολουθούν μια διαφορετική προσέγγιση και εκμεταλλεύονται τα ιεραρχικά πρότυπα στα δεδομένα αναλύοντας πιο σύνθετα πρότυπα σε μικρότερα, απλούστερα πρότυπα. Επομένως, τα CNN έχουν συνδεσιμότητα και πολυπλοκότητα χαμηλής κλίμακας.

Τα συνελκτικά νευρωνικά δίκτυα εμπνέονται από βιολογικές διεργασίες (τα πρότυπα των συνδέσεων μεταξύ των νευρώνων) και μοιάζουν με την οργάνωση του σημαντικού οπτικού φλοιού. Οι μεμονωμένοι νευρώνες ανταποκρίνονται σε μια περιορισμένη περιοχή του οπτικού πεδίου, το λεγόμενο πεδίο υποδοχής. Όλα τα πεδία υποδοχής των νευρώνων καλύπτουν ολόκληρο το οπτικό πεδίο. Τα CNN χρησιμοποιούν σχετικά λίγη προεπεξεργασία σε σύγκριση με άλλους αλγόριθμους ταξινόμησης εικόνων. Αυτό συμβαίνει επειδή το δίκτυο μαθαίνει τα ίδια τα φίλτρα, σε αντίθεση με άλλους αλγόριθμους όπου οι άνθρωποι καθορίζουν τα φίλτρα με βάση τη γνώση. Η ανεξαρτησία από την προηγούμενη γνώση και η ανθρώπινη προσπάθεια για την ανάπτυξη λειτουργιών νευρωνικών δικτύων είναι ένα σημαντικό πλεονέκτημα. [15]

2.4.2 Stochastic Gradient Descent

Η στοχαστική κάθοδος κλίσης (συνήθως συντομογραφείται ως SGD) είναι μια επαναληπτική μέθοδος βελτιστοποίησης αντικειμενικής συνάρτησης με κατάλληλες ιδιότητες ομαλότητας (όπως διαφοροποιήσιμη ή υποδιαφοροποιήσιμη). Αντικαθιστά την πραγματική κλίση (υπολογισμένη από ολόκληρο το σύνολο δεδομένων) με την εκτίμησή της (υπολογισμένη από ένα τυχαία επιλεγμένο υποσύνολο δεδομένων), καθιστώντας την μια προσέγγιση στοχαστικής βελτιστοποίησης gradient descent. Ειδικά για πολυδιάστατα προβλήματα βελτιστοποίησης, αυτό μειώνει το υπολογιστικό φορτίο και επιταχύνει τις επαναλήψεις λόγω του χαμηλότερου ρυθμού σύγκλισης. Αν και η βασική ιδέα των στοχαστικών προσεγγίσεων ανάγεται στον αλγόριθμο Robbins-Monro της δεκαετίας του 1950, η στοχαστική κάθοδος κλίσης έχει γίνει μια σημαντική τεχνική βελτιστοποίησης στη μηχανική μάθηση.



Διάγραμμα 2.3 Stochastic Gradient Descent
(https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1)

ΚΕΦΑΛΑΙΟ 3^ο

Αρχιτεκτονικές Βαθιάς Μάθησης

3.1 Γνωστές αρχιτεκτονικές Deep Neural Networks

Η παραδοσιακή μορφή της αρχιτεκτονικής συνελκτικών νευρωνικών δικτύων περιλαμβάνει πολλαπλά συνελκτικά επίπεδα με πολλαπλές λειτουργίες ενεργοποίησης, ακολουθούμενα από επίπεδα συγκέντρωσης που επαναλαμβάνονται μέχρι να μειωθεί το μέγεθος της εικόνας. Από εκεί, εισάγονται πλήρως συνδεδεμένα επίπεδα και το τελικό επίπεδο παρέχει προβλέψεις ανά κατηγορία. Ωστόσο, με την πάροδο του χρόνου, η αρχιτεκτονική των συνελκτικών δικτύων έχει γίνει πιο περίπλοκη, ειδικά με την εισαγωγή νέων μονάδων όπως τα υπολειμματικά μπλοκ και τα μπλοκ βαθιάς συνέλιξης.

Οι πιο γνωστές αρχιτεκτονικές συνελκτικού νευρωνικού δικτύου είναι :

1. LeNet (1998, Yann LeCun και άλλοι)
2. AlexNet (2012, Alex Krizhevsky)
3. GoogLeNet (2014, Christian Szegedy κ.α.)
4. VGG (2015, Karen Simoyan, Andrey Zisserman)
5. Inception-v3 (2015, Christian Szegedy κ.α.)
6. ResNet (2015, Kaiming He κ.α.)
7. Xception (2017, Francois Cholet)
8. ResNext (2017, Saining Xie, κ.α.)

Το νέο μοντέλο, που θα εμφανιστεί μετά το 2017, είναι ένας συνδυασμός των εξαρτημάτων των παραπάνω μοντέλων. Μέχρι το 2016, η γενική τάση ήταν να δημιουργηθούν ολοένα μεγαλύτερα συνελκτικά δίκτυα για τη βελτίωση της απόδοσης, αλλά ταυτόχρονα ξεκίνησαν οι πρώτες δοκιμές για να φέρουν τη βαθιά μάθηση σε συσκευές αιχμής και στις κινητές συσκευές. Ωστόσο, τα υπάρχοντα δίκτυα ήταν τεράστια σε σύγκριση με τις περιορισμένες δυνατότητες τέτοιων συσκευών, οπότε η χρήση του SqueezeNet (2016, Forrest N. Iandola κ.α.) και του MobileNets (2017, Andrew G. Howard) δημιούργησε την ανάγκη για ένα πιο ελαφρύ δίκτυο τα οποία και κέρδισαν.

Οι προσεγγίσεις βαθιάς μάθησης διαφέρουν σημαντικά τόσο στη δομή όσο και στην υπολογιστική απόδοση σε σύγκριση με προηγούμενες γενιές μεθόδων μηχανικής μάθησης (π.χ. ρηχές δομημένες αρχιτεκτονικές). Στο πλαίσιο της βαθιάς μάθησης, τα χαρακτηριστικά κατανέμονται σε διαφορετικά επίπεδα, χωρίζονται σε μπλοκ και συνδυάζονται με στρώματα μη γραμμικών μετασχηματισμών. Αυτή η χωρική διάταξη είναι μια ιεραρχική αρχιτεκτονική που

επιτρέπει σε αυτοματοποιημένα σχήματα να δημιουργούν αφαιρέσεις και σύνθετες αναπαραστάσεις από μεγάλες ποσότητες μη διαχειριζόμενων δεδομένων (Deng, 2013; Najafabadi κ.α., 2015).

Ρηχές και Βαθείς Δομημένες Αρχιτεκτονικές

Οι αρχιτεκτονικές ρηχών δομών συνήθως αποτελούνται από λίγα μόνο επίπεδα μη γραμμικών μετασχηματισμών χαρακτηριστικών. Τα συστήματα που βασίζονται σε ρηχές αρχιτεκτονικές έχουν περιορισμένη μοντελοποίηση και αναπαραστατική ισχύ (Deng & Yu, 2014). Έχει αναφερθεί ότι οι εφαρμογές που χρησιμοποιούν φυσικά σήματα μπορεί να είναι προβληματικές και αναποτελεσματικές όταν υλοποιούνται σε ρηχές αρχιτεκτονικές (Bengio & LeCun, 2009; Deng & Yu, 2014; SAP SE, 2015). Η ιεραρχική πολυεπίπεδη προσέγγιση των μεθόδων βαθιάς μάθησης βελτιώνει την αποτελεσματικότητα της εξαγωγής μη τοπικών και παγκόσμιων σχέσεων και προτύπων σε δεδομένα σε σύγκριση με ρηχές αρχιτεκτονικές δικτύου (Bengio & LeCun, 2009; Najafabadi κ.α., 2015). Η κύρια διαφορά μεταξύ των βαθιών προσεγγίσεων και άλλων μεθόδων μηχανικής μάθησης έγκειται στην ανεπαρκώς βαθιά αρχιτεκτονική τους, η οποία αναφέρεται στη διαδικασία που λαμβάνει χώρα σε μπλοκ όπου σχηματίζονται κρυφά επίπεδα (κόμβοι) και εσωτερικές αναπαραστάσεις. Ένας ρηχός ταξινομητής απαιτεί έναν κατάλληλο εξαγωγέα χαρακτηριστικών για να λειτουργήσει. Η καταλληλότητα ενός εργαλείου εξαγωγής χαρακτηριστικών είναι σημαντική για τη διαδικασία αναγνώρισης, καθώς ορίζεται ως η ικανότητα να σχηματίζει μια αναπαράσταση που επιλέγεται σύμφωνα με τα δεδομένα εισόδου (π.χ. εικόνες).

Επιπρόσθετα, οι παραγόμενες αναπαραστάσεις πρέπει να είναι αμετάβλητες σε σχέση με άσχετες πτυχές του ερεθίσματος εισόδου (LeCun κ.α., 2015). Η διαδικασία επιλογής των κατάλληλων χαρακτηριστικών για μια συγκεκριμένη εργασία είναι μια πολύ περίπλοκη διαδικασία. Το πρόβλημα γίνεται ακόμη πιο περίπλοκο όταν πρόκειται για σύνθετες εργασίες και πολυδιάστατα δεδομένα (Gudi, 2014). Ο παραδοσιακός τρόπος αφαίρεσης, δηλαδή ο διαχωρισμός των εννοιών υψηλού επιπέδου από τη λειτουργικότητα χαμηλού επιπέδου, είναι να προσδιορίζονται χειροκίνητα όλες οι απαραίτητες αφαιρέσεις, παρέχοντας παράλληλα μεγάλο αριθμό σχετικών παραδειγμάτων με σήμανση χειρός (Bengio, 2009). Η επιλογή της μη αυτόματης ανάπτυξης μιας αποτελεσματικής μηχανής εξαγωγής χαρακτηριστικών απαιτεί σημαντικές γνώσεις τομέα και προηγμένες δεξιότητες μηχανικής (LeCun κ.α., 2015). Επιπλέον, ο μη αυτόματος προσδιορισμός της σωστής λειτουργικότητας για μια συγκεκριμένη εργασία συνεπάγεται μια αλληλεξάρτηση μεταξύ της προτεινόμενης λύσης και της γνώσης του χρήστη για τον τομέα, ενώ αυτή η διαδικασία είναι εφαρμόσιμη σε σχεδόν οποιαδήποτε άλλη εργασία (Gudi, 2014). Όλες αυτές οι χειροποίητες διαδικασίες μπορούν να αποφευχθούν χρησιμοποιώντας μια διαδικασία μάθησης γενικού σκοπού που παρουσιάζει ιδιότητες παρόμοιες με τον νεοφλοιό (Arel & Rose, 2010), εστιασμένη στην εκμάθηση μοντέλων ιεραρχικών δεδομένων (Palm, 2012). Αυτός είναι ο στόχος των πλαισίων βαθιάς μάθησης όπου η κατασκευή ανιχνευτών χαρακτηριστικών γίνεται χρησιμοποιώντας μια διαδικασία μάθησης χωρίς επίβλεψη χωρίς την ανάγκη για επισημασμένα δεδομένα (LeCun κ.α., 2015). Το ενδιαφέρον σε αυτή την περίπτωση μετατοπίζεται σε διαδικασίες και στρατηγικές και, κατά συνέπεια, σε προσπάθειες

αποκάλυψης της δομής δεδομένων χρησιμοποιώντας αλγόριθμους μάθησης γενικού σκοπού (Bengio κ.α., 2012; LeCun κ.α., 2015).

3.2 Αρχιτεκτονικές βαθιάς μάθησης

Οι διαφορές μεταξύ των γενικών δικτύων βαθιάς μάθησης και των απλών πολυεπίπεδων δικτύων εμπρόσθιας τροφοδότησης είναι οι εξής:

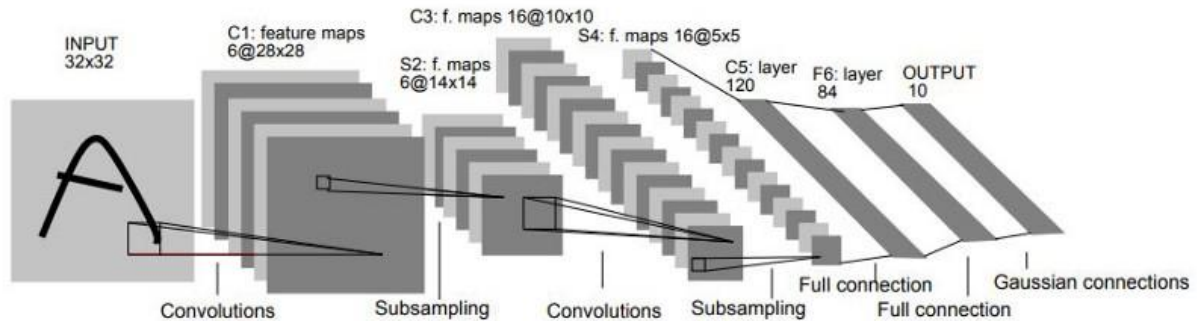
- Αύξηση του αριθμού των νευρώνων
- Ένας πιο περίπλοκος τρόπος σύνδεσης των επιπέδων
- Αυξημένη υπολογιστική ισχύς για μάθηση
- Αυτόματη εξαγωγή των χαρακτηριστικών

Η αύξηση του αριθμού των νευρώνων σημαίνει αύξηση του αριθμού των νευρώνων με την πάροδο του χρόνου για να αναπαραστήσουν πιο περίπλοκα μοντέλα. Οι δομές επιπέδων έχουν επίσης εξελιχθεί από πλήρως συνδεδεμένα δίκτυα σε δίκτυα πολλαπλών επιπέδων με σύνδεση νευρώνων τοπικά συνδεδεμένα μεταξύ των επιπέδων. Η βαθιά μάθηση μπορεί να οριστεί ως ένα νευρωνικό δίκτυο με πολλές παραμέτρους και επίπεδα. Οι θεμελιώδεις αρχιτεκτονικές των νευρωνικών δικτύων είναι τα Συνελικτικά Νευρωνικά Δίκτυα και τα Νευρωνικά Δίκτυα με Ανάδραση.

3.2.1 LeNet-5

Το LeNet-5 είναι η πιο απλή βασική αρχιτεκτονική με την οποία χτίζονται άλλα δίκτυα ταξινόμησης η οποία δημιουργήθηκε το 1998 από τους Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner. Διαθέτει 2 συνελκτικά και 3 συνδεδεμένα επίπεδα (εξου και 5). Η αρχιτεκτονική χρησιμοποιεί τη συγκέντρωση που γνωρίζουμε ως συγκέντρωση του μέσου όρου (μερικές φορές ονομάζεται υποδειγματοληψία με βάση την τιμή). Υπάρχει επίσης ένα υποσύνολο αρχιτεκτονικής που δημιουργεί ένα συνελκτικό και πλήρως

συνδεδεμένο στρώμα. Χρησιμοποιώντας αυτή τη δομή και αυτή τη συγκεκριμένη μεθοδολογία σχεδιασμού, υπάρχει νευρωνικό δίκτυο με περίπου 60.000 παραμέτρους που εκπαιδεύονται μέσω 8 εποχών. Στο Διάγραμμα 3.1 παρουσιάζεται η δομή του LeNet-5. Εάν αυτή η δομή επαναληφθεί μερικές φορές, λαμβάνονται πολλοί μικροί χάρτες χαρακτηριστικών. Στο τέλος, τροφοδοτούνται σε μια κλασική δομή αντίληψης πολλαπλών επιπέδων. Τέλος, οι χάρτες χαρακτηριστικών ευθυγραμμίζονται και τροφοδοτούνται με την κλασική δομή πολλαπλών επιπέδων.



Διάγραμμα 3.1 LeNet-5
(<https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/>)

3.2.2 AlexNet

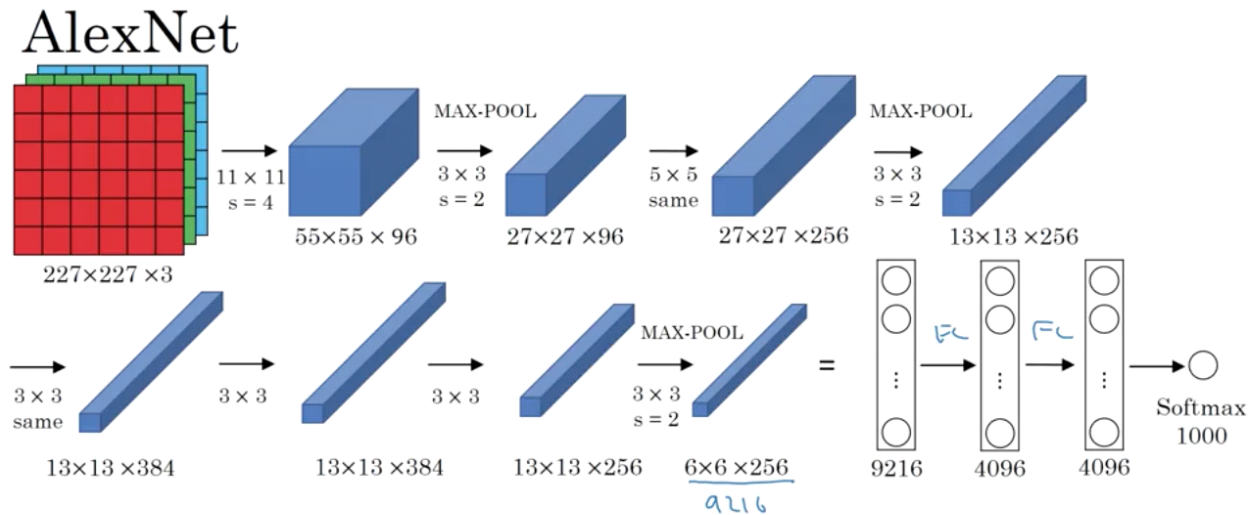
Το AlexNet είναι η συντομογραφία του δικτύου που κέρδισε ο Alex Krizhevsky το 2012 στον μεγάλης κλίμακας διαγωνισμό οπτικής αναγνώρισης ImageNet -ILSVRC-. Το ImageNet είναι ένα σύνολο δεδομένων που περιέχει πάνω από 15 εκατομμύρια εικόνων υψηλής ανάλυσης που ανήκουν σε 22.000 κατηγορίες.

Το AlexNet έχει 60 εκατομμύρια παραμέτρους, οι οποίοι αποτελούνται από 5 συνελκτικά επίπεδα, 3 max-pooling επίπεδα συγκέντρωσης και 3 πλήρως συνδεδεμένα επίπεδα. Το δίκτυο ακολουθεί τη γενική δομή ενός CNN, όπου όλα τα στρώματα ακολουθούνται από μια συνάρτηση ενεργοποίησης ReLU, ακολουθούμενη από την εφαρμογή ενός στρώματος κανονικοποίησης. Ωστόσο, το πλήρως συνδεδεμένο στρώμα ακολουθείται από μια συνάρτηση ενεργοποίησης softmax αντί για στρώματα κανονικοποίησης (normalization).

Χρησιμοποιώντας το ίδιο δίκτυο, η μέθοδος αυτή πέτυχε 45,7% βελτίωση σφάλματος top-1 σε σχέση με την μέθοδο νίκης του 2011. Το AlexNet είναι γνωστό όχι μόνο για το γεγονός ότι ανεβάζει τα αποτελέσματα της ταξινόμησης εικόνων σε ένα εντελώς νέο επίπεδο, αλλά και για τη μαζική παγίωση των χαρακτηριστικών drop-out σε όλα τα στρώματα του δικτύου. Αυτή η μαζική διαδικασία βελτίωσε επίσης τις επιδόσεις σε άλλες εργασίες, όπως η ανίχνευση σημασιολογικών χαρακτηριστικών υψηλού επιπέδου.

Στα τέλη του 2012, λόγω των περιορισμών στις κάρτες γραφικών GPU, οι νευροεπιστήμονες Alexey Krizhevsky και Ilya Sutskever από την Ιατρική Σχολή του

Πανεπιστημίου του Τορόντο χρησιμοποίησαν ένα ειδικό βαθύ νευρωνικό δίκτυο που ονομάζεται "AlexNet" για να μάθουν την αναγνώριση προσώπου. Η συμβολή του περιλαμβάνει: 1) Χρήση του ReLU για την αποφυγή κορεσμού και 2) Χρήση του Dropout για την αποφυγή της υπερπροσαρμογής.



[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

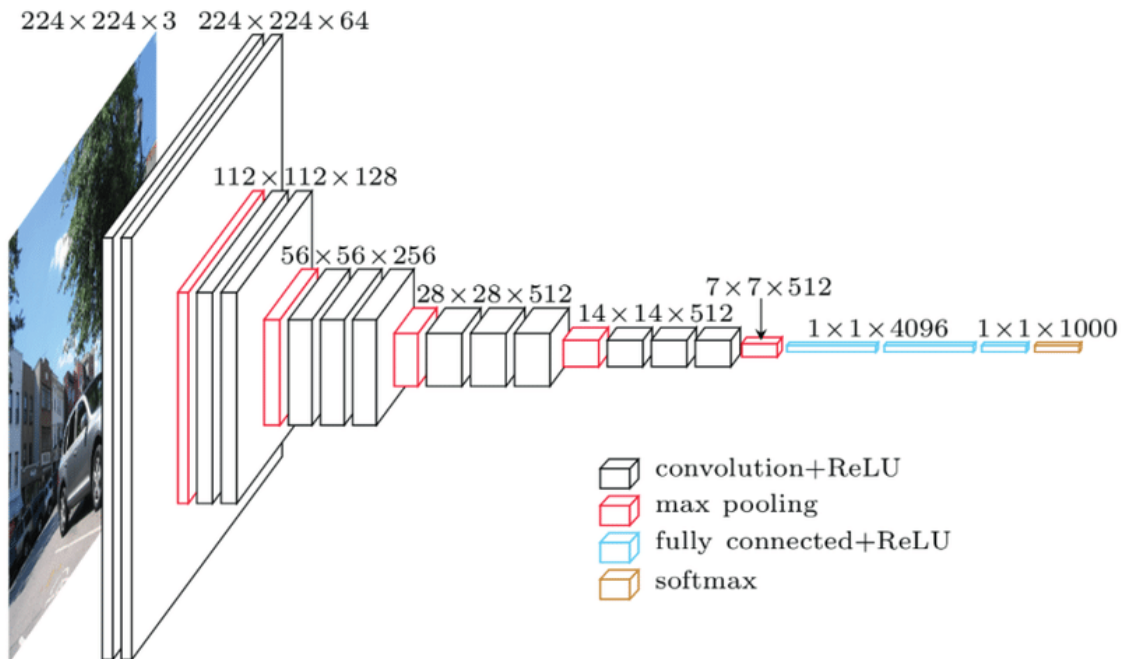
Andrew Ng

Εικόνα 3.1 AlexNet
(<https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848>)

3.2.3 VGGNet

Το VGGNet είναι ένας γενικός όρος για δομές δικτύου που δημιουργήθηκαν από την Ομάδα Οπτικής Γεωμετρίας (Visual Geometry Group) στο Πανεπιστήμιο της Οξφόρδης. Τα μεταγενέστερα δίκτυα VGGNet ξεχώρισαν τον Οκτώβριο του 2017, όταν κέρδισαν τον διαγωνισμό ILSVRC-2014 (μετά το GoogleNet). Προτάθηκε από τους Simonyan και Zisserman και χρησιμοποίησε 19 επίπεδα στα βαθιά συνελκτικά δίκτυα. Οι κύριες συνεισφορές των δικτύων VGGNet περιλαμβάνουν: 1) η απόδειξη ότι τα πολύ βαθιά δίκτυα βελτιώνουν τη γενίκευση και 2) η εύρεση μικρών φίλτρων πολλαπλών επιπέδων που είναι υπολογιστικά πιο αποτελεσματικά από την εκμάθηση. Για παράδειγμα, η χρήση τριών συνελκτικών στρώματων 3×3 είναι υπολογιστικά πιο αποτελεσματική από την εκπαίδευση ενός συνελκτικού στρώματος με ένα φίλτρο 7×7 .

Το VGG Net είναι μια εξέλιξη των δικτύων AlexNet, τα οποία χρησιμοποιούν πολύ μικρά φίλτρα συνελίξεων (3x3). Για το πρώτο μέρος του δικτύου υπάρχουν στρώματα συνελκτικής ανάλυσης με φίλτρα (3x3), ακολουθούμενα από τρία διαδοχικά στρώματα 3x3 max-pooling, ακολουθούμενα από πέντε στρώματα συνελκτικής ανάλυσης 3x3 και τέλος δύο ακόμη στρώματα 3x3 max-pooling. Επιπλέον, το VGGNet (Vector Graphics Group Net) χρησιμοποιεί μικρά μη επικαλυπτόμενα παράθυρα max-pooling (2x2). Αυτό το δίκτυο χρησιμοποιεί εκτεταμένες επεκτάσεις εκπαίδευσης. Αυτό το δίκτυο χρησιμοποιεί αύξηση εκπαίδευσης (training augmentation) και επεκτείνει την αύξηση που χρησιμοποιείται στο AlexNet.



Εικόνα 3.2 VGGNet
(https://www.researchgate.net/figure/VGGNet-architecture-19_fig2_333242381)

3.2.4 ResNet

Ο He κ.α. (2016) εισήγαγε τα υπολειμματικά δίκτυα (residual networks) για να συμβάλει στη βελτίωση της απόδοσης ισχυρών βαθιών δικτύων. Έχει αποδειχθεί ότι τα βαθύτερα δίκτυα βελτιώνουν τη γενίκευση, ειδικά σε μεγάλα σύνολα δεδομένων, αλλά σε

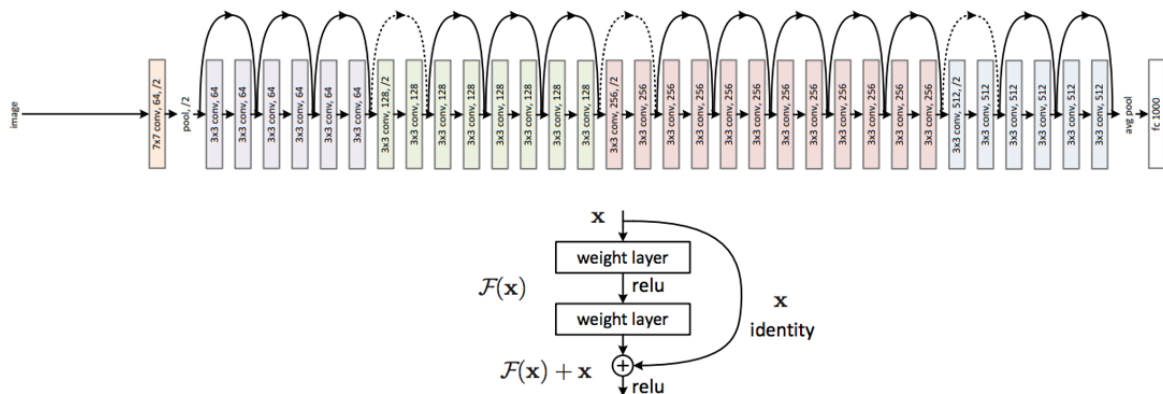
ένα αρκετά μεγάλο βάθος η εκπαίδευση γίνεται δύσκολη, ακόμη και με ομαλοποίηση παρτίδας (batch normalization) και σωστή αρχικοποίηση, η γενίκευση αρχίζει να μειώνεται. Τα βαθιά δίκτυα μπορούν να βελτιώσουν τη γενίκευση φτάνοντας σε ένα σημείο όπου η εκπαίδευση γίνεται δύσκολη, ακόμη και με ομαλοποίηση παρτίδων και κατάλληλη αρχικοποίηση, και έτσι η γενίκευση αρχίζει να μειώνεται ραγδαία. Πριν από την έλευση των νευρωνικών δικτύων, η εκπαίδευση των βαθιών δικτύων ήταν δύσκολη επειδή το πρόβλημα της εξαφανιζόμενης κλίσης επηρέαζε σημαντικά την απόδοση.

Οι καινοτομίες του He σε αυτό το θέμα μπορούν να συνοψιστούν στο Διάγραμμα 3.2 παρακάτω. Εκπαίδευοντας ένα βαθύ δίκτυο και χρησιμοποιώντας μια καλή γενίκευση του L1. Προσθέτοντας επίπεδα σε μια κατά τα άλλα πανομοιότυπη αρχιτεκτονική δικτύου (επανεκπαίδευση από τυχαία αρχικοποίηση), η νέα απώλεια εκπαίδευσης για ένα δίκτυο με $N + 1$ επίπεδα υπολογίζεται ως $L2 > L1$, δηλαδή αυξάνεται η απώλεια εκπαίδευσης.

Οπότε ο He πρότεινε μια "υπολειπόμενη σύνδεση" για να βοηθήσει στη βελτιστοποίηση. Αυτή η υπολειπόμενη σύνδεση προστίθεται στο συνελκτικό στρώμα για να λυθεί η ταυτότητα $L1 = L1$ (μετασχηματισμός ταυτότητας $H(x) = x$) και να γίνει εύκολη η εκμάθηση.

Το σημείο εκκίνησης της βασικής ιδέας του ResNet είναι ότι η εισαγωγή της λεγόμενης "ταυτονικής παρακαμπτήριας σύνδεσης", η οποία εγγυάται την προ-ενεργοποίηση πριν από την τελική. Η υπολειπόμενη σύνδεση αυτή μπορεί να θεωρηθεί ως συντόμευση. Υποθέτοντας ότι η χαρτογράφηση από το ένα επίπεδο στο άλλο είναι επιθυμητή αλλά δύσκολο να βελτιστοποιηθεί $H(x)$, τότε η υπολειπόμενη συνάρτηση είναι $H(x) + x$.

Residual Networks (ResNet50)



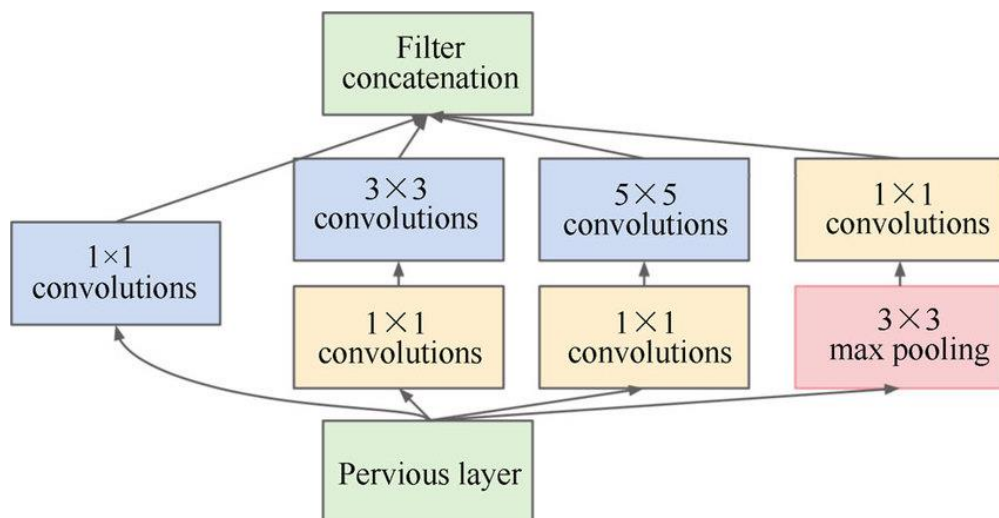
Διάγραμμα 3.2 ResNet
(https://jananisbabu.github.io/ResNet50_From_Scratch_Tensorflow)

3.2.5 Inception-GoogLeNet

Η αρχιτεκτονική Inception ή GoogLeNet των Szegedy, Liu, κ.α. έγινε ο νικητής του διαγωνισμού ILSVRC το 2014, όσον αφορά την ακρίβεια ταξινόμησης. Αυτή η αρχιτεκτονική είναι αρκετά ενδιαφέρουσα, επειδή σχεδιάστηκε ειδικά για να ελαχιστοποιεί τον υπολογισμό και εξερευνά πιο αποτελεσματικές αναπαραστάσεις.

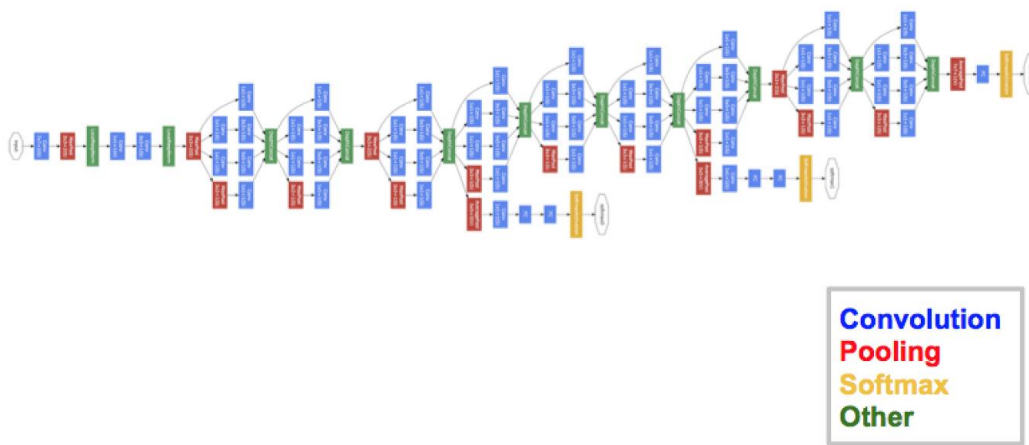
Το υπόβαθρο αυτής της αρχιτεκτονικής είναι ότι οι πιο σημαντικές συσχετίσεις στις φυσικές εικόνες δεν είναι παγκόσμιες, αλλά πολύ εντοπισμένες. Έτσι, τα φίλτρα, τα οποία υπολογίζουν κατά μέσο όρο τις πληροφορίες σε μεγάλες περιοχές, μαθαίνουν τα πιο σημαντικά χαρακτηριστικά, όπως οι κλίσεις και οι ακμές, αλλά όχι τις υφές ή τα χρώματα. Ωστόσο, με βάση τη μελέτη μας, ορισμένες συσχετίσεις, ανεξαρτήτως φίλτρων, γίνονται πιο εντοπισμένες και ευκολότερα αποτυπώσιμες σε φίλτρα 5×5 ή ακόμη και 7×7 . Ωστόσο, κατά την εκμάθηση βέλτιστων φίλτρων 9×9 , το Inception μαθαίνει συνήθως φίλτρα 1×1 και 3×3 , εκτός από λίγα φίλτρα 5×5 και ακόμη λιγότερα φίλτρα 7×7 . Με βάση αυτό, το Inception επιτυγχάνει αναπαράσταση και αποδοτικότητα.

Το δίκτυο περιέχει ένα στρώμα συνέλιξης 1×1 , όπως φαίνεται σε αυτό το Διάγραμμα 3.3. Στο τέλος του δικτύου χρησιμοποιείται η global average pooling για τον υπολογισμό των απωλειών. Η μονάδα εισόδου προκύπτει κάνοντας το πλάτος/ύψος εισόδου μια μεταβλητή μέσα σε μια συνέλιξη, όπως φαίνεται στο Διάγραμμα 3.3. Παράγει πολλές διαφορετικές εξόδους. Μπορούμε να στοιβάξουμε όλες τις εξόδους προκειμένου να ταξινομήσουμε τα εξαγόμενα χαρακτηριστικά. Μια ακόμη τεχνική, η οποία ονομάζεται inception module, είναι η χρήση συνελιξωνδιαφορετικού μεγέθους ή τύπου για την ίδια είσοδο και στοιβάγμα όλων των εξόδων.



Διάγραμμα 3.3 Inception

(https://www.researchgate.net/figure/nception-module-of-GoogLeNet-This-figure-is-from-the-original-paper-10_fig3_312515254)



Διάγραμμα 3.4 GoogleNet τελική μορφή
<https://medium.com/analytics-vidhya/cnns-architectures-lexnet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>

3.3 RMDL και SimpleNet

Η ανάπτυξη μεθόδων μηχανικής μάθησης για την κατηγοριοποίηση μεγάλων συνόλων δεδομένων απαιτεί τη συνεχή βελτίωση τόσο της επιλογής της μεθόδου όσο και της αρχιτεκτονικής. Ως εκ τούτου, θα διερευνήσουμε ένα νέο πλαίσιο βαθιάς μάθησης για την ταξινόμηση πολλαπλών τύπων δεδομένων το RMDL (Random Multimodel Deep Learning for Classification). Κάθε σύνθετο σύνολο δεδομένων μπορεί να αποδοθεί μόνο σε μία από ένα πεπερασμένο σύνολο κλάσεων και σε αυτή τη διαδικασία, τα δεδομένα αυτά χωρίζονται σε σύνολα εκπαίδευσης, επικύρωσης και δοκιμής. Το μοντέλο εφαρμόζεται σε όλους τους τύπους δειγμάτων δεδομένων, όπως κείμενο, εικόνες και βίντεο. Το RMDL δέχεται συνοδευόμενα σύνολα δεδομένων με κείμενο, εικόνα, βίντεο.

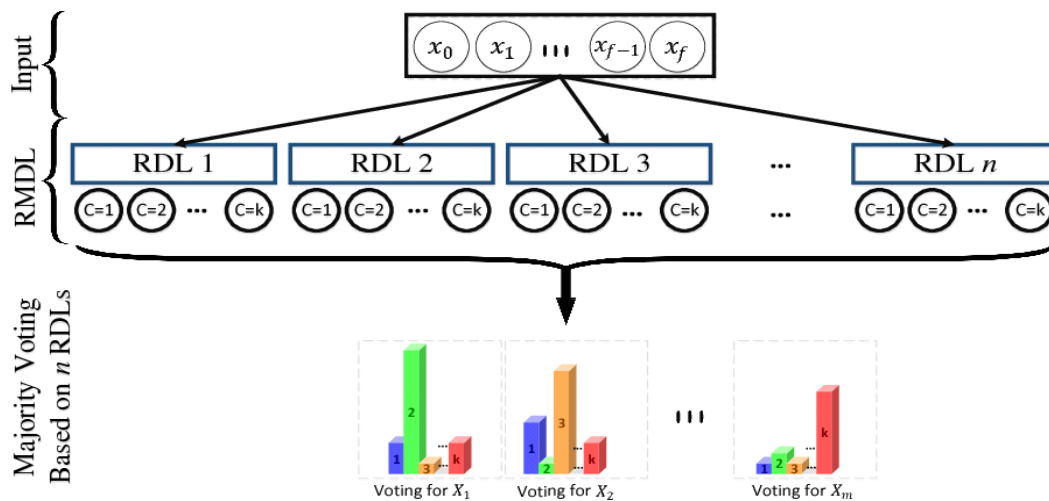
Στη φάση εκπαίδευσης της βαθιάς μάθησης, πρέπει να κατανοήσουμε πώς να χρησιμοποιήσουμε την αρχιτεκτονική βαθιάς μάθησης που είναι η καταλληλότερη για το πρόβλημα που αντιμετωπίζουμε. Η απόδοση των δικτύων βαθιάς μάθησης (βαθιά νευρωνικά δίκτυα, DNN, convolutional neural networks, CNN) περιλαμβάνει τη δοκιμή περισσότερων από μία ομάδα αρχιτεκτονικής βαθιών δικτύων. Μερικές από τις ανησυχίες των ανθρώπων που ασχολούνται με τη βαθιά μάθηση είναι ποια αρχιτεκτονική βαθιάς μάθησης προσεγγίζει το RMDL. Τα βαθιά νευρωνικά δίκτυα (DNN), τα συνεκτικά νευρωνικά δίκτυα (CNN) και τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN), τα οποία χρησιμοποιούνται για την επίλυση προβλημάτων ταξινόμησης εικόνων και κειμένων.

Οι γενιές των μοντέλων γίνονται τυχαία (π.χ. κατασκευάζονται 9 τυχαία μοντέλα RMDL, τα οποία παράγουν ένα μοναδικό CNN, RNN και DNN, λόγω της τυχαίας και μοναδικής τυχαίας δημιουργίας).

Πολλά γνωστά νευρωνικά δίκτυα βαθιάς μάθησης, όπως τα AlexNet, VGGNet, ResNet, GoogleNet, κυμαίνονται σε μέγεθος από 10 έως 100 εκατομμύρια παραμέτρους. Αυτό έχει προκαλέσει το ενδιαφέρον για αρχιτεκτονικές νευρωνικών δικτύων ελαφρού βάρους (όπως το LeNet, το GoogleNet και το NasNet, οι οποίες έχουν σχετικά πολύ χαμηλότερο υπολογιστικό/κόστος σε σύγκριση με τα CNN). Ωστόσο, η ακρίβειά τους δεν είναι εξίσου καλή με τα CNN..

Προτείνεται επίσης μια απλή αρχιτεκτονική, η οποία ονομάζεται SimpleNet και βασίζεται σε ένα σύνολο αρχών σχεδιασμού που δείχνουν εμπειρικά ότι μια καλά σχεδιασμένη και απλή αρχιτεκτονική μπορεί να αποδώσει εξίσου καλά με πολύπλοκα και βαθιά δίκτυα. Το SimpleNet παρέχει μια ισορροπία μεταξύ της αποδοτικότητας των υπολογισμών/της μνήμης και της ακρίβειας.

Η απλή αρχιτεκτονική που προτείνεται ξεπερνά την πιο σύνθετη αρχιτεκτονική βάθους μέχρι σήμερα, καθώς και την υπάρχουσα σύγχρονη δομή βάθους, προωθώντας περαιτέρω τον τομέα, ανοίγοντας δυνατότητες για συμπίεση μοντέλων, ελαφριές υλοποιήσεις και υποστήριξη μεγαλύτερης ποικιλίας εφαρμογών. Έχει μικρότερο αριθμό παραμέτρων από τις περισσότερες προηγούμενες βαθιές αρχιτεκτονικές όπως τα VGGNet, ResNet, GoogleNet.



Διάγραμμα 3.5 RMDL

(https://www.researchgate.net/figure/Overview-of-RDML-Random-Multimodel-Deep-Learning-for-classification-that-includes-n_fig1_324922651)

ΚΕΦΑΛΑΙΟ 4^ο

Python

4.1 Εισαγωγή

Η Python είναι μια διαδραστική, υψηλού επιπέδου, αντικειμενοστραφή γλώσσα προγραμματισμού. Χαρακτηρίζεται από απλότητα, αφού η σύνταξη είναι πολύ εύκολη στην εκμάθηση, γεγονός που οδηγεί σε μείωση του κόστους συντήρησης του προγράμματος.

4.2 Χρήσιμες βιβλιοθήκες της Python

❖ Numpy

Το Numpy είναι ένα βασικό πακέτο για την εκτέλεση υπολογισμών στη γλώσσα προγραμματισμού Python. Έχει ένα μεγάλο σύνολο μαθηματικών συναρτήσεων υψηλού επιπέδου και επομένως χρησιμοποιείται για εργασία με πολυδιάστατους πίνακες. Το Numpy είναι ένα ισχυρό εργαλείο. Αυτό συμβαίνει επειδή το TensorFlow χρησιμοποιεί πίνακες στις λειτουργίες του όπως περιγράφεται παρακάτω και συσχετίζει τα επεξεργασμένα δεδομένα με αυτούς.

❖ Pandas

Είναι μια βιβλιοθήκη της Python που χρησιμοποιείται ως εργαλείο ανάλυσης δεδομένων και μοντελοποίησης. Βασίζεται σε μια δομή δεδομένων που ονομάζεται DataFrame, η οποία μοιάζει με πίνακα.

4.3 TensorFlow, Keras

Η Python είναι μια δυναμική, υψηλού επιπέδου, γενικής χρήσης γλώσσα προγραμματισμού με πολύ καλή τεκμηρίωση και χιλιάδες βιβλιοθήκες, δηλαδή ελάχιστος κώδικας και αποτελεσματικός. Αυτή η εργασία χρησιμοποιεί τις βιβλιοθήκες Sklearn και matplotlib, οι οποίες θεωρούνται βασικά πακέτα για επιστημονικούς υπολογισμούς με χρήση Python. Μεταξύ άλλων, περιλαμβάνει ισχυρά αντικείμενα για N-διάστατους πίνακες που χειρίζονται ταχύτερα από τους πίνακες Python.

Το εργαλείο pickle υλοποιεί τη λειτουργικότητα για τη μετατροπή εικόνων ή αντικειμένων κειμένου σε ένα δυαδικό πρωτόκολλο για σειριοποίηση και αποσεροποίηση. Το "pickle" είναι η διαδικασία μετατροπής μιας ιεραρχίας αντικειμένων Python σε μια ροή byte, ενώ το "unpickling" είναι η αντίστροφη λειτουργία μετατροπής μιας ροής byte σε μια ιεραρχία αντικειμένων. Η βιβλιοθήκη Python TensorFlow, ακόμη και από το όνομά της, αντικατοπτρίζει τον τρόπο λειτουργίας της. Αποτελείται από τις λέξεις Tensor, που σημαίνει τάση-τανυστής, και Flow, που σημαίνει ροή. Τα δεδομένα στο TensorFlow αντιπροσωπεύονται ως τανυστές, οι οποίοι είναι πολυδιάστατοι πίνακες που λαμβάνουν τιμές 0 ή 1 (Nikhil Buduma, 2015). Το Flow ισχυρίζεται ότι το πρόγραμμα επιτρέπει στους χρήστες να εκφράζουν αυτούς τους υπολογισμούς ως γραφήματα ροής δεδομένων. Οι κόμβοι (νευρώνες) ενός γραφήματος είναι μαθηματικές πράξεις που υπολογίζουν τα δεδομένα που αποστέλλονται στον επόμενο κόμβο και τελικά εξάγουν τα δεδομένα που εξάγονται στον επόμενο κόμβο. Η παραπάνω ιδέα των υπολογιστικών λειτουργιών έχει εφαρμοστεί σε διάφορα επιστημονικά πεδία, αλλά το TensorFlow χρησιμοποιείται κυρίως για πολυεπίπεδα νευρωνικά δίκτυα ή βαθιά μάθηση. Η ευέλικτη αρχιτεκτονική του επιτρέπει την ανάπτυξη βαθιών νευρωνικών δικτύων και την βαθιά μηχανική μάθηση από κατανεμημένα δίκτυα υπολογιστών με έναν ή περισσότερους επεξεργαστές CPU ή GPU σε επιτραπέζιους υπολογιστές και κινητές συσκευές χρησιμοποιώντας ένα μόνο API.

Το Tensorflow είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα που χρησιμοποιείται για αριθμητικούς υπολογισμούς χρησιμοποιώντας γραφήματα ροής δεδομένων. Αναπτύχθηκε από την ομάδα της Google Brain ως μέρος της έρευνας Google Machine Intelligence για τη μηχανική μάθηση και την ανάλυση «βαθιών νευρωνικών δικτύων». Παρά τους αρχικούς λόγους δημιουργίας του, το σύστημα είναι πλέον αρκετά διαδεδομένο ώστε να χρησιμοποιείται για πολλούς άλλους σκοπούς. Έχοντας φτάσει στην έκδοση 1.0 τον Φεβρουάριο του 2017, συνεχίζει να εξελίσσεται ραγδαία, με πάνω από 21.000 αλλαγές κώδικα (commits) μέχρι σήμερα, πολλές από τις οποίες είναι από εξωτερικούς συνεργάτες. Αυτό είναι ένα εργαλείο cross-platform. Αυτό σημαίνει ότι μπορεί να λειτουργεί σε CPU, GPU (συμπεριλαμβανομένων φορητών συσκευών και ενσωματωμένων συστημάτων) και TPU (Tensor Processing Unit).

Το Tensorflow σας επιτρέπει να διανέμετε ροές εργασιών μηχανικής εκμάθησης σε πολλές συσκευές και χάρη στις διεπαφές υψηλού επιπέδου Python και C++, μπορείτε να δημιουργήσετε μηχανές ιδιαίτερα υψηλής απόδοσης που εκτελούν δημοφιλή API όπως Keras και Estimator API.

Google/Tensorflow:

Μια πλατφόρμα ανοιχτού κώδικα με μαθηματικούς υπολογισμούς σύμφωνα με το μοντέλο γραφικών του υπολογιστή. [16]

- Ευκολία υλοποίησης για πολύ βαθιά μοντέλα
- Μια βιβλιοθήκη βαθιών νευρικών μοντέλων χτισμένη πάνω από Tensorflow.
- Βιβλιοθήκες για Βαθιά νευρωνικά δίκτυα για το CNN.

Το TensorFlow είναι μια βιβλιοθήκη λογισμικού η οποία χρησιμοποιείται από τους προγραμματιστές της Google Brain Team για την απλοποίηση της εκπαίδευσης και της εξαγωγής συμπερασμάτων με νευρωνικά δίκτυα. Η βιβλιοθήκη βασίζεται σε ένα παράδειγμα προγραμματισμού ροής δεδομένων, παρόμοιο με το ομώνυμο του λογισμικού του. Το TensorFlow έχει σχεδιαστεί για να βοηθήσει στη δημιουργία μαθηματικών εκφράσεων που απαιτούν μεγάλους χρόνους εκτέλεσης λόγω της υψηλής επεξεργαστικής τους ισχύος. Συνδυάζει τεχνικές βελτιστοποίησης με υπολογιστική άλγεβρα προκειμένου να υπολογίζει γρήγορα και αποτελεσματικά συγκεκριμένους αλγορίθμους. [17]

4.3.1 Γράφοι Ροής Δεδομένων (Data Flow Graphs)

Οι εφαρμογές μηχανικής μάθησης είναι το αποτέλεσμα πολλαπλών υπολογισμών πολύπλοκων μαθηματικών τύπων. Το TensorFlow χρησιμοποιεί γραφήματα ροής δεδομένων για να περιγράψει υπολογισμούς. Κάθε κόμβος γραφήματος αντιπροσωπεύει ένα παράδειγμα μιας μαθηματικής πράξης και κάθε ακμή του γράφου αντιπροσωπεύει ένα πολυδιάστατο σύνολο δεδομένων ή πίνακα (τανυστής) στο οποίο εφαρμόζονται αυτές οι μαθηματικές πράξεις. Συγκεκριμένα, το TensorFlow διαχειρίζεται κόμβους και ακμές ως εξής:

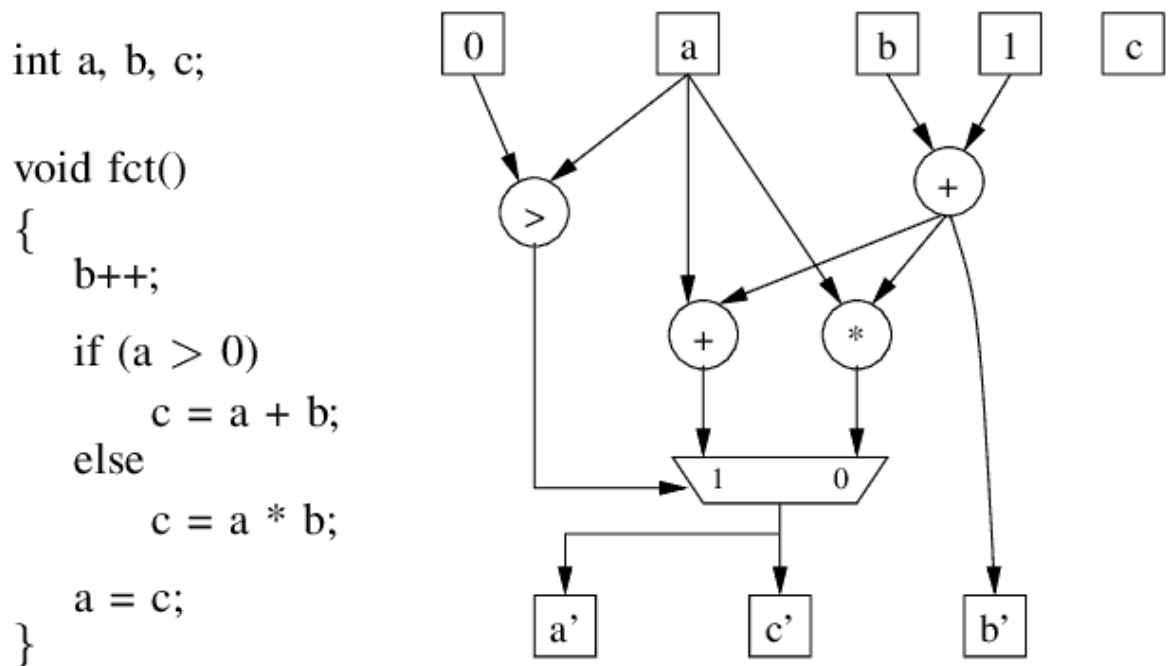
Κόμβος (node): Κάθε κόμβος αντιπροσωπεύει μια μαθηματική πράξη. Αυτή η διαδικασία έχει μια είσοδο και τουλάχιστον μια έξοδο.

Ακμή (edge): Υπάρχουν δύο τύποι ακμών στο TensorFlow:

Απλές Ακμές: Οι απλές άκρες επιτρέπουν στους τανυστές να καταρρέουν σε διαφορετικές κατευθύνσεις. Η έξοδος ενός βοηθητικού είναι η είσοδος ενός άλλου βοηθητικού.

Ειδικές Ακμές: Εδώ διαφορετικοί κόμβοι μπορούν να έχουν ακμές που τους συνδέουν. Αντιπροσωπεύει μια εξάρτηση μεταξύ κόμβων που χρειάζονται πρωτίστως ο ένας τις πληροφορίες του άλλου αντί για μρταγορά δεδομένων μεταξύ δύο κόμβων. Λαμβάνοντας υπόψη δύο κόμβους, τους κόμβους A και B, η ακμή μεταξύ των κόμβων B

και A υποδεικνύει ότι το B ξεκινά τη δουλειά του μετά την ολοκλήρωση του A. Αυτό σημαίνει ότι οι ακμές κωδικοποιούν τις εξαρτήσεις και τη σειρά αποσύνθεσης του συνόλου δεδομένων.



Διάγραμμα 4.1 Γράφοι Ροής Δεδομένων
(https://www.researchgate.net/figure/Data-flow-graph-example_fig2_224517165)

- **Keras**

Ένα ισχυρό εργαλείο που υπάρχει στην βιβλιοθήκη TensorFlow είναι το Keras.[18] Είναι γραμμένο σε γλώσσα python και ανήκει στη κατηγορία του ελεύθερου λογισμικού για νευρωνικά δίκτυα. Έχει δημιουργηθεί από τον Francois Chollet, ο οποίος είναι ένας μηχανικός που εργάζεται στην Google, με σκοπό την ταχεία δημιουργία, τροποποίηση, εκπαίδευση και αξιολόγηση νευρωνικών δικτύων. Για τον λόγο αυτόν είναι εφοδιασμένο με έτοιμες λειτουργίες που περιλαμβάνουν συνελκτικά επίπεδα, συναρτήσεις ενεργοποίησης και βελτιστοποίησης, ακόμη και εργαλεία για επεξεργασία συνόλων δεδομένων εικόνας και κειμένου. Τέλος, επιτρέπει

την εκπαίδευση νευρωνικών δικτύων βαθιάς μάθησης σε κάρτες γραφικών για μεγαλύτερη ταχύτητα και ευελιξία. [19]

4.4 Εξόρυξη και ανάλυση κειμένου με χρήση της Python

Η Python είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο, καθώς μπορεί να ενσωματωθεί με άλλες γλώσσες προγραμματισμού, καθιστώντας την ιδανική για καινοτόμες ιδέες για νέα έργα. Η χρήση της Python είναι ευρέως αναγνωρισμένη στον τομέα της τεχνητής νοημοσύνης, καθιστώντας την ιδανική για έργα σε τομείς όπως το λογισμικό υπολογιστών, η μηχανική μάθηση και ο NLP/Neuro Linguistic Programming. [20]

ΚΕΦΑΛΑΙΟ 5^ο

Πειραματική Διαδικασία

5.1 Περιγραφή – Στόχοι

Σκοπός του πειράματος μας είναι να εκπαιδεύσουμε ένα μοντέλο το οποίο δοθέντος μιας αφίσας μίας ταινίας θα μπορεί με ακρίβεια να προβλέψει το είδος της (π.χ. Θρίλερ, Δράση, Τρόμου, κλπ.). Συγκεκριμένα, θα εκπαιδεύσουμε το μοντέλο ώστε να είναι σε θέση να αναγνωρίζει ταινίες για τα είδη Κωμωδία, Δράμα, Δράση για λόγους απλότητας, ωστόσο είναι δυνατό να συμπεριλάβει κανείς όσες κατηγορίες επιθυμεί για την προπόνηση του μοντέλου. Με βάση σχετικά προγράμματα, θα θεωρήσουμε επιτυχημένη την εκπαίδευση όταν το μοντέλο μας θα είναι σε θέση να προβλέψει με ακρίβεια άνω του 75%, δεδομένου ότι το πρόβλημα το οποίο προσπαθούμε να λύσουμε έχει υψηλό βαθμό δυσκολίας (π.χ. δεν είναι απαραίτητο ότι δύο κωμωδίες έχουν κοινό μοτίβο στη δημιουργία του πόστερ τους). Ακόμη, θα εξετάσουμε τρόπους με τους οποίους μπορούμε να επιταχύνουμε τη διαδικασία της εκπαίδευσης μέσα από σωστές πρακτικές στη διαχείριση του συνόλου δεδομένων μας.

5.2 Υλοποίηση

5.2.1 Προετοιμασία συνόλου δεδομένων

Για την υλοποίηση των αλγορίθμων μας θα χρησιμοποιήσουμε το προγραμματιστικό περιβάλλον του Visual Studio Code 1.71 και την έκδοση 3.10.4 της Python. Το σύνολο δεδομένων πάνω στο οποίο θα προπονήσουμε και θα εξετάσουμε το μοντέλο μας έχει συλλεχθεί από το IMDB και βρίσκεται στο Kaggle, από όπου και θα το κατεβάσουμε σε μορφή .csv [21]. Το αρχείο MovieGenre.csv περιέχει στοιχεία από 40108 ταινίες και αποτελείται από 6 πεδία: imdbId, Imdb Link, Title, IMDB Score, Genre, Poster.

- *ImdbId* → Το αναγνωριστικό της ταινίας.
- *Imdb Link* → Ο υπερσύνδεσμος της ταινίας.
- *Title* → Ο τίτλος της ταινίας
- *IMDB Score* → Η βαθμολογία της ταινίας.
- *Genre* → Το είδος (ή τα είδη) της ταινίας
- *Poster* → Ο υπερσύνδεσμος που οδηγεί στο αρχείο εικόνας με κατάληξη .jpg και περιέχει το πόστερ της ταινίας.

5.2.2 Εξόρυξη και αποθήκευση εικόνων

Στη συνέχεια, θα πρέπει να δημιουργήσουμε έναν φάκελο που θα περιέχει το σύνολο των εικόνων πόστερ οι οποίες μας παρέχονται από το πεδίο Poster του MovieGenre.csv, καθώς και ένα αρχείο το οποίο θα λαμβάνει τις εικόνες από το διαδίκτυο και θα τις αποθηκεύει. Δημιουργούμε τον φάκελο posters και το αρχείο get_data.py αντίστοιχα. Το *get_data.py* δέχεται τρία ορίσματα: -download, -resize και -min_year.

get_data.py -download: Ελέγχει εάν τα αρχεία εικόνας υπάρχουν ήδη και αν όχι τα κατεβάζει και τα αποθηκεύει στον φάκελο posters.

get_data.py -resize: Τροποποιεί το μέγεθος των αρχείων εικόνας και τα ομαδοποιεί ως προς αυτό.

get_data.py -min_year: Θέτει την ελάχιστη χρονιά για την αναζήτηση εικόνων.

Δεν είναι απαραίτητο να τροποποιήσουμε το μέγεθος των εικόνων μας, ωστόσο η συγκεκριμένη τεχνική επιταχύνει τον αλγόριθμό μας αισθητά. Υπάρχουν περιπτώσεις στις οποίες η τροποποίηση του μεγέθους των εικόνων μπορεί να επηρεάσει την ακρίβεια προβλέψεων του μοντέλου, όπως για παράδειγμα εάν οι εικόνες που χρησιμοποιούμε περιέχουν σημαντικές λεπτομέρειες οι οποίες θα αλλοιωθούν με τη σμίκρυνσή τους. Αυτό βέβαια είναι κάτι που συνήθως δεν μπορούμε να γνωρίζουμε από την αρχή, αλλά ανακαλύπτουμε μέσα από πειραματισμούς.

Ενώ η αποθήκευση των αρχείων εικόνας jpg μας βοηθάει να ομαδοποιήσουμε το σύνολο δεδομένων μας, δεν είναι πολύ πρακτικό για τον αλγόριθμο που θα τα χειρίζεται. Για να χρησιμοποιήσουμε τις αποθηκευμένες εικόνες μέσα στον αλγόριθμο θα πρέπει να τις μετατρέψουμε σε δυαδικά αρχεία τα οποία περιέχουν στοιχεία της κλίμακας RGB.

Για την μετατροπή των εικόνων από jpg σε RGB θα χρειαστούμε την μέθοδο `convert()` της κλάσης `Image` από το πακέτο `PIL`. Η μετατροπή αποτελεί μία χρονοβόρα διαδικασία, επομένως θα κατασκευάσουμε μία ακόμη μέθοδο για την αποθήκευση των RGB αρχείων σε έναν νέο φάκελο `rgb_images` έτσι ώστε να μην χρειάζεται να επαναληφθεί.

Για την διαδικασία αυτή, θα χρησιμοποιήσουμε το πακέτο `Pandas`, το οποίο θα μας διευκολύνει να διαβάσουμε το csv αρχείο μας και να περιηγηθούμε στα στοιχεία του. Στη συνέχεια, θα αποθηκεύσουμε τις εικόνες που εξορύξαμε και θα τις μετατρέψουμε σε RGB, οι οποίες θα αποθηκεύονται με τη σειρά τους σε ένα μόνο δυαδικό αρχείο ώστε να αποφύγουμε το χρονοβόρο άνοιγμα/κλείσιμο αρχείων που θα αναγκαζόμασταν να υποστούμε εάν αποθηκεύαμε κάθε RGB εικόνα σε ένα ξεχωριστό δυαδικό αρχείο.

5.2.3 Επιλογή αρχιτεκτονικής του δικτύου

Έχοντας εξορύξει και αποθηκεύσει τα δεδομένα που μας παρέχει το σύνολο δεδομένων σε στιγμιότυπα της κλάσης `Monie`, επόμενο βήμα είναι η επιλογή της αρχιτεκτονικής που θα ακολουθήσουμε για την εκπαίδευση του μοντέλου. Επιλέξαμε να χωρίσουμε το σύνολο δεδομένων ως εξής: 75% εκπαίδευσης, 25% επαλήθευσης. Το αρχικό μέγεθος των εικόνων μας είναι 268x182 πίξελ. Επιλέγουμε να δουλέψουμε με εικόνες μεγέθους 30% του αρχικού, δηλαδή 80x55 πίξελ. Τελικά, μετά την μετατροπή τους σε RGB καταλήγουμε σε τρισεπίστας πίνακα 80x55x3 ως απεικόνιση μίας εικόνας.

Θα χρησιμοποιήσουμε το πακέτο `Keras` για να ορίσουμε την αρχιτεκτονική του μοντέλου και συγκεκριμένα την κλάση `Sequential` την οποία μας προσφέρει το πακέτο. Ο λόγος για τον οποίο επιλέξαμε τη συγκεκριμένη κλάση είναι η ικανότητα της να χειρίζεται επίπεδα νευρώνων εκ των οποίων ο καθένας συνδέεται με ακριβώς έναν νευρώνα εισόδου και έναν νευρώνα εξόδου.

Είσοδος: Εικόνες μεγέθους 80x55x3.

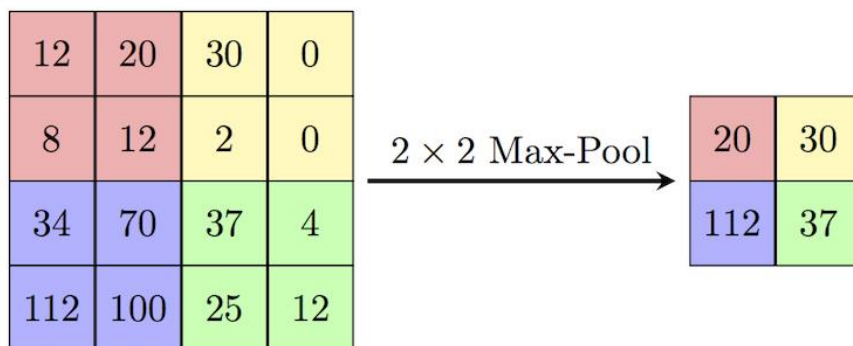
Conv2D: Αποτελείται από 32 πυρήνες μεγέθους 3x3 ίσης μετατόπισης και συνδέεται με 32 νευρώνες εξόδου. Ακόμη, χρησιμοποιεί την γραμμική συνάρτηση μετατόπισης `relu`.

MaxPooling2D: Εντοπίζει μοτίβα των πίξελ υπολογίζοντας τη μέγιστη τιμή σε ένα πεδίο 2x2.

Dropout: Επιλέγει τυχαία νευρώνες και ορίζει την τιμή 0 προς αποφυγή υπερπροσαρμογής. Επιλέξαμε 25% πιθανότητα για κάθε νευρώνα.

Τα πρώτα δύο επίπεδα του μοντέλου θα είναι της κλάσης Conv2D του πακέτου Keras. Η κλάση Conv2D σαρώνει την εικόνα με σκοπό να εξάγει πιθανά χαρακτηριστικά της. Απαραίτητα ορίσματα της Conv2D είναι οι διαστάσεις του πυρήνα (επιλέξαμε 3x3 πειραματικά), η οριζόντια μετατόπιση του πυρήνα κατά πλάτος της εικόνας (επιλέξαμε ίση μετατόπιση), η συνάρτηση ενεργοποίησης και ο αριθμός νευρώνων. Επιλέξαμε την relu σαν συνάρτηση ενεργοποίησης των νευρώνων η οποία απενεργοποιεί τους νευρώνες με αρνητική τιμή καθώς κρίνει ότι δεν είναι σημαντικοί, μειώνοντας έτσι την πολυπλοκότητα του αλγορίθμου. Τέλος, θέσαμε 32 νευρώνες για κάθε ένα από τα δύο συνελκτικά επίπεδα του μοντέλου μας καθώς είναι αρκετοί για την κάλυψη όλων των μοναδικών 3x3 επιφανειών μιας εικόνας.

Στη συνέχεια, θα προσθέσουμε ακόμη μία φορά το ίδιο σεντ επιπέδων και φίλτρων με ελάχιστες διαφορές στις ρυθμίσεις των συνελκτικών επιπέδων. Συγκεκριμένα, θα χρησιμοποιήσουμε 64 νευρώνες αντί για 32. Ο λόγος που χρησιμοποιούμε παραπάνω νευρώνες είναι αποτέλεσμα της εφαρμογής των φίλτρων, τα οποία παράγουν έναν πολλαπλάσιο δισδιάστατο πίνακα του αρχικού πίνακα εισόδου. Η εικόνα 5.1 απεικονίζει την παρακάτω διαδικασία. Εφαρμόζοντας το φίλτρο διαστάσεων 2x2 στον πίνακα εισόδου, παράγεται ένας νέος πίνακας εξόδου. Το γεγονός ότι το φίλτρο μετατοπίζεται ένα εικονοστοιχείο ανά εφαρμογή του έχει ως αποτέλεσμα ο πίνακας εξόδου να είναι διπλάσιος σε μέγεθος από τον πίνακα εισόδου. Εάν εφαρμόζαμε το φίλτρο ανά δύο εικονοστοιχεία (εφόσον το φίλτρο έχει διαστάσεις 2x2) ο πίνακας εξόδου θα είχε ίδιο μέγεθος με τον πίνακα εισόδου και συνεπώς δεν θα χρειαζόμασταν 64 νευρώνες στα συνελκτικά επίπεδα που ακολουθούν.



Εικόνα 5.1 – MaxPooling2D
(https://computersciencewiki.org/index.php/Max-pooling/_Pooling)

Τέλος, θα προσθέσουμε το φίλτρο Flatten, το οποίο μετατρέπει έναν πολυδιάστατο πίνακα σε μονοδιάστατο έτσι ώστε το επόμενο επίπεδο, Dense, να παράγει έναν μονοδιάστατο πίνακα σαν έξοδο. Ορίζουμε, λοιπόν, στο Dense επίπεδο μας την συνάρτηση ενεργοποίησης relu και αυτό με τη σειρά του παράγει 512 στοιχεία εξόδου, ένα από κάθε νευρώνα. Στη συνέχεια εφαρμόζουμε ακόμη μία φορά ένα Dropout με πιθανότητα εφαρμογής 50% και τέλος προσθέτουμε το επίπεδο εξόδου του δικτύου μας, Dense, το οποίο περιέχει τόσους νευρώνες όσες είναι και οι κατηγορίες ταινιών που μας ενδιαφέρουν. Ακόμη, θα ορίσουμε σαν συνάρτηση ενεργοποίησης την sigmoid, η οποία θα επιτρέψει στον αλγόριθμό μας να αποφασίζει εάν η εκάστοτε εικόνα αντιστοιχεί σε μια κατηγορία ταινίας. Πιο συγκεκριμένα, εάν η τιμή ενός νευρώνα εξόδου είναι μεγαλύτερη από 0.5, τότε η sigmoid θέτει στον ίδιο νευρώνα την τιμή 1, ενώ εάν η τιμή του είναι μικρότερη από 0.5, τίθεται η τιμή 0. Η Εικόνα 5.2 απεικονίζει την τελική πρόσοψη του κώδικα που χρησιμοποιήσαμε για την δομή της αρχιτεκτονικής του νευρωνικού δικτύου μας.

```

model = Sequential([
    Conv2D(32, (3, 3), padding='same', activation='relu'),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Conv2D(64, (3, 3), padding='same', activation='relu'),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='sigmoid')
])

```

Εικόνα 5.2 – Δομή του Sequential δικτύου

Στην συνέχεια, κάναμε πείραμα όπου οι διαστάσεις του πυρήνα της Conv2D είναι 6x6. Αλλάζοντας τον πυρήνα έχει νόημα να αλλάξουν και άλλες παράμετροι όπως το μέγεθος των εικόνων. Δεν θα υπήρχε μεγάλη διαφορά αλλάζοντας μόνο τον πυρήνα. Οπότε, πέρα από την τελική ακρίβεια του μοντέλου στο παρακάτω διάγραμμα δε υπάρχουν και άλλες διαφορές καθώς αυτό είναι και συγκυριακό εάν τρέχαμε το πρόγραμμα ξανά μπορεί να έβγανε άλλη ακρίβεια στα αποτελέσματα. Μετά από το πείραμα αυτό, διαπιστώθηκε ότι η χρήση μικρού μεγέθους πυρήνα οδηγεί στην μείωση πολλών παραμέτρων του μοντέλου και επίσης έχει σημαντική επίδραση στην ακρίβεια του μοντέλου καθώς και στον υπολογιστικό χρόνο και τη κατανομή του βάρους.

```

Creating binfiles...
Loading datasets...
Going to read 16509 posters of size 80x54
Done in 2.0844 seconds.

version: 1
min_year: 1977
max_year: 2017
genres: ['Comedy', 'Drama', 'Action']
ratio: 30

Train: X=(16509, 80, 54, 3), y=(16509, 3)
Test: X=(16509, 80, 54, 3), y=(16509, 3)
16509 validation samples
kernel_dimensions1: (6, 6)
kernel_dimensions2: (6, 6)
x_train.shape[1:] = (80, 54, 3)
2023-02-14 17:23:53.864994: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dLError: nvcuda.dll not found
2023-02-14 17:23:53.865513: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2023-02-14 17:23:53.869730: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-HRBS1D6
2023-02-14 17:23:53.870062: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-HRBS1D6
2023-02-14 17:23:53.872904: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
super(RMSprop, self)._init__(name, **kwargs)
Model: "sequential"

```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 80, 54, 32)	3488
conv2d_1 (Conv2D)	(None, 75, 49, 32)	36896
max_pooling2d (MaxPooling2D)	(None, 37, 24, 32)	0
dropout (Dropout)	(None, 37, 24, 32)	0
conv2d_2 (Conv2D)	(None, 37, 24, 64)	73792
conv2d_3 (Conv2D)	(None, 32, 19, 64)	147520
max_pooling2d_1 (MaxPooling2D)	(None, 16, 9, 64)	0
dropout_1 (Dropout)	(None, 16, 9, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 512)	4719104
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 3)	1539

```

=====
Total params: 4,982,339
Trainable params: 4,982,339
Non-trainable params: 0

```

Εικόνα 5.3 – Εκτέλεση κώδικα για 6x6 πυρήνα

```

Epoch 1/60
516/516 [=====] - 402s 778ms/step - loss: 0.6149 - accuracy: 0.4732 - val_loss: 0.6022 - val_accuracy: 0.5519
Epoch 2/60
516/516 [=====] - 397s 770ms/step - loss: 0.5940 - accuracy: 0.5369 - val_loss: 0.5796 - val_accuracy: 0.5840
Epoch 3/60
516/516 [=====] - 397s 770ms/step - loss: 0.5844 - accuracy: 0.5576 - val_loss: 0.5794 - val_accuracy: 0.5834
Epoch 4/60
516/516 [=====] - 397s 769ms/step - loss: 0.5792 - accuracy: 0.5638 - val_loss: 0.5656 - val_accuracy: 0.5985
Epoch 5/60
516/516 [=====] - 399s 773ms/step - loss: 0.5736 - accuracy: 0.5705 - val_loss: 0.5779 - val_accuracy: 0.6114
Epoch 6/60
516/516 [=====] - 397s 770ms/step - loss: 0.5689 - accuracy: 0.5730 - val_loss: 0.5555 - val_accuracy: 0.6034
Epoch 7/60
516/516 [=====] - 399s 774ms/step - loss: 0.5647 - accuracy: 0.5782 - val_loss: 0.5686 - val_accuracy: 0.6136
Epoch 8/60
516/516 [=====] - 397s 770ms/step - loss: 0.5618 - accuracy: 0.5850 - val_loss: 0.5488 - val_accuracy: 0.5960
Epoch 9/60
516/516 [=====] - 397s 769ms/step - loss: 0.5571 - accuracy: 0.5844 - val_loss: 0.5489 - val_accuracy: 0.5891
Epoch 10/60
516/516 [=====] - 399s 774ms/step - loss: 0.5545 - accuracy: 0.5839 - val_loss: 0.5519 - val_accuracy: 0.6091
Epoch 11/60
516/516 [=====] - 398s 772ms/step - loss: 0.5511 - accuracy: 0.5911 - val_loss: 0.5321 - val_accuracy: 0.6194
Epoch 12/60
516/516 [=====] - 398s 772ms/step - loss: 0.5470 - accuracy: 0.5905 - val_loss: 0.5356 - val_accuracy: 0.6224
Epoch 13/60
516/516 [=====] - 397s 770ms/step - loss: 0.5434 - accuracy: 0.5968 - val_loss: 0.5408 - val_accuracy: 0.5527
Epoch 14/60
516/516 [=====] - 398s 772ms/step - loss: 0.5409 - accuracy: 0.5976 - val_loss: 0.5209 - val_accuracy: 0.6309
Epoch 15/60
516/516 [=====] - 398s 772ms/step - loss: 0.5349 - accuracy: 0.6048 - val_loss: 0.5254 - val_accuracy: 0.6077
Epoch 16/60
516/516 [=====] - 398s 772ms/step - loss: 0.5333 - accuracy: 0.6037 - val_loss: 0.5384 - val_accuracy: 0.5768
Epoch 17/60
516/516 [=====] - 398s 772ms/step - loss: 0.5291 - accuracy: 0.6098 - val_loss: 0.5337 - val_accuracy: 0.5853
Epoch 18/60
516/516 [=====] - 398s 771ms/step - loss: 0.5238 - accuracy: 0.6091 - val_loss: 0.5182 - val_accuracy: 0.6064
Epoch 19/60
516/516 [=====] - 398s 772ms/step - loss: 0.5199 - accuracy: 0.6128 - val_loss: 0.5178 - val_accuracy: 0.6510
Epoch 20/60
516/516 [=====] - 398s 771ms/step - loss: 0.5143 - accuracy: 0.6156 - val_loss: 0.4932 - val_accuracy: 0.6347
Epoch 21/60
516/516 [=====] - 398s 771ms/step - loss: 0.5088 - accuracy: 0.6255 - val_loss: 0.4840 - val_accuracy: 0.6339
Epoch 22/60
516/516 [=====] - 398s 772ms/step - loss: 0.5072 - accuracy: 0.6228 - val_loss: 0.4840 - val_accuracy: 0.6344
Epoch 23/60
516/516 [=====] - 399s 774ms/step - loss: 0.5038 - accuracy: 0.6260 - val_loss: 0.4755 - val_accuracy: 0.6538
Epoch 24/60
516/516 [=====] - 398s 772ms/step - loss: 0.4963 - accuracy: 0.6347 - val_loss: 0.4800 - val_accuracy: 0.6677
Epoch 25/60
516/516 [=====] - 398s 772ms/step - loss: 0.4910 - accuracy: 0.6383 - val_loss: 0.4622 - val_accuracy: 0.6378
Epoch 26/60
516/516 [=====] - 398s 771ms/step - loss: 0.4859 - accuracy: 0.6393 - val_loss: 0.4546 - val_accuracy: 0.6533
Epoch 27/60
516/516 [=====] - 398s 771ms/step - loss: 0.4808 - accuracy: 0.6448 - val_loss: 0.4413 - val_accuracy: 0.6796
Epoch 28/60
516/516 [=====] - 398s 772ms/step - loss: 0.4747 - accuracy: 0.6495 - val_loss: 0.4377 - val_accuracy: 0.6790
Epoch 29/60
516/516 [=====] - 398s 771ms/step - loss: 0.4688 - accuracy: 0.6549 - val_loss: 0.4323 - val_accuracy: 0.6770
Epoch 30/60
516/516 [=====] - 399s 773ms/step - loss: 0.4623 - accuracy: 0.6574 - val_loss: 0.4229 - val_accuracy: 0.6871
Epoch 31/60
516/516 [=====] - 397s 770ms/step - loss: 0.4562 - accuracy: 0.6617 - val_loss: 0.4171 - val_accuracy: 0.7180
Epoch 32/60
Epoch 34/60
516/516 [=====] - 400s 775ms/step - loss: 0.4402 - accuracy: 0.6768 - val_loss: 0.4046 - val_accuracy: 0.7039
Epoch 35/60
516/516 [=====] - 399s 773ms/step - loss: 0.4323 - accuracy: 0.6808 - val_loss: 0.3818 - val_accuracy: 0.6999
Epoch 36/60
516/516 [=====] - 397s 770ms/step - loss: 0.4245 - accuracy: 0.6830 - val_loss: 0.3817 - val_accuracy: 0.7383
Epoch 37/60
516/516 [=====] - 398s 771ms/step - loss: 0.4170 - accuracy: 0.6952 - val_loss: 0.3725 - val_accuracy: 0.6794
Epoch 38/60
516/516 [=====] - 395s 766ms/step - loss: 0.4135 - accuracy: 0.6968 - val_loss: 0.3404 - val_accuracy: 0.7438
Epoch 39/60
516/516 [=====] - 395s 766ms/step - loss: 0.4106 - accuracy: 0.6972 - val_loss: 0.3349 - val_accuracy: 0.7481
Epoch 40/60

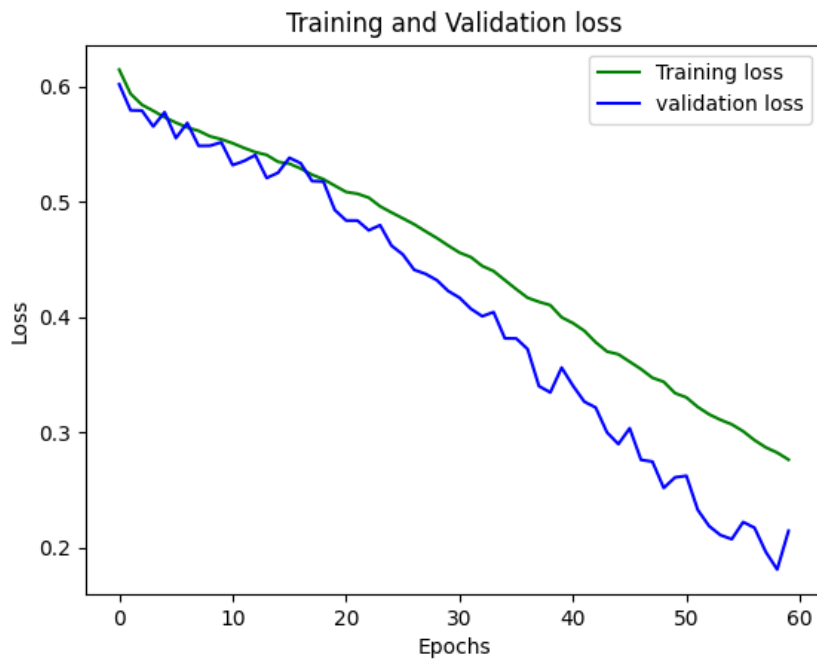
```

```

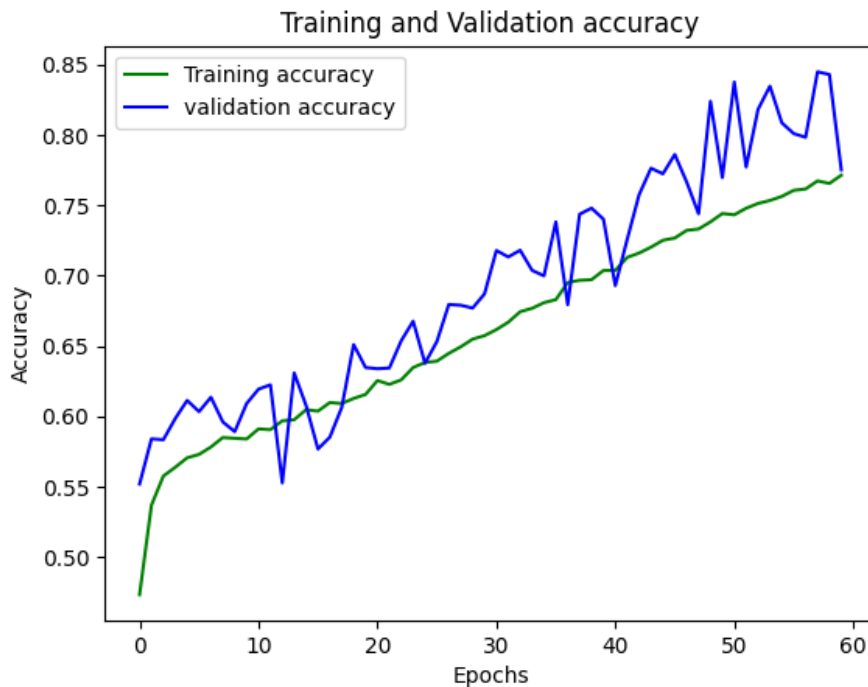
Epoch 41/60
516/516 [=====] - 393s 763ms/step - loss: 0.3950 - accuracy: 0.7038 - val_loss: 0.3407 - val_accuracy: 0.6930
Epoch 42/60
516/516 [=====] - 393s 762ms/step - loss: 0.3884 - accuracy: 0.7131 - val_loss: 0.3269 - val_accuracy: 0.7259
Epoch 43/60
516/516 [=====] - 394s 764ms/step - loss: 0.3784 - accuracy: 0.7162 - val_loss: 0.3218 - val_accuracy: 0.7576
Epoch 44/60
516/516 [=====] - 393s 762ms/step - loss: 0.3704 - accuracy: 0.7203 - val_loss: 0.3002 - val_accuracy: 0.7765
Epoch 45/60
516/516 [=====] - 393s 762ms/step - loss: 0.3680 - accuracy: 0.7252 - val_loss: 0.2901 - val_accuracy: 0.7725
Epoch 46/60
516/516 [=====] - 394s 764ms/step - loss: 0.3616 - accuracy: 0.7268 - val_loss: 0.3038 - val_accuracy: 0.7862
Epoch 47/60
516/516 [=====] - 394s 765ms/step - loss: 0.3551 - accuracy: 0.7323 - val_loss: 0.2764 - val_accuracy: 0.7666
Epoch 48/60
516/516 [=====] - 393s 762ms/step - loss: 0.3476 - accuracy: 0.7332 - val_loss: 0.2749 - val_accuracy: 0.7441
Epoch 49/60
516/516 [=====] - 394s 764ms/step - loss: 0.3441 - accuracy: 0.7383 - val_loss: 0.2521 - val_accuracy: 0.8240
Epoch 50/60
516/516 [=====] - 394s 763ms/step - loss: 0.3343 - accuracy: 0.7443 - val_loss: 0.2613 - val_accuracy: 0.7698
Epoch 51/60
516/516 [=====] - 394s 763ms/step - loss: 0.3306 - accuracy: 0.7434 - val_loss: 0.2626 - val_accuracy: 0.8378
Epoch 52/60
516/516 [=====] - 393s 763ms/step - loss: 0.3223 - accuracy: 0.7480 - val_loss: 0.2330 - val_accuracy: 0.7773
Epoch 53/60
516/516 [=====] - 393s 762ms/step - loss: 0.3159 - accuracy: 0.7514 - val_loss: 0.2190 - val_accuracy: 0.8183
Epoch 54/60
516/516 [=====] - 393s 762ms/step - loss: 0.3112 - accuracy: 0.7535 - val_loss: 0.2112 - val_accuracy: 0.8348
Epoch 55/60
516/516 [=====] - 393s 763ms/step - loss: 0.3072 - accuracy: 0.7564 - val_loss: 0.2077 - val_accuracy: 0.8088
Epoch 56/60
516/516 [=====] - 393s 762ms/step - loss: 0.3012 - accuracy: 0.7607 - val_loss: 0.2225 - val_accuracy: 0.8011
65 - val_accuracy: 0.8451
Epoch 59/60
516/516 [=====] - 394s 763ms/step - loss: 0.2827 - accuracy: 0.7656 - val_loss: 0.1814 - val_accuracy: 0.8431
Epoch 60/60
516/516 [=====] - 393s 763ms/step - loss: 0.2766 - accuracy: 0.7715 - val_loss: 0.2148 - val_accuracy: 0.7753

```

Εικόνα 5.4 - Εκτέλεση



Διάγραμμα 5.1 – Απώλεια εκπαίδευσης/επικύρωσης



Διάγραμμα 5.2 – Ακρίβεια εκπαίδευσης/επικύρωσης

5.2.4 Σύνθεση του μοντέλου

Για τη σύνθεση του μοντέλου επιλέξαμε να χρησιμοποιήσουμε την binary cross-entropy σαν συνάρτηση λάθους, ωστόσο υπάρχουν πηγές που συνιστούν τη χρήση categorical cross-entropy. [26] Παρότι η χρήση της binary cross-entropy μπορεί να μην θεωρείται η καθιερωμένη συνάρτηση λάθους για ένα πρόβλημα ταξινόμησης πολλαπλών κατηγοριών (multi-label), σκοπός είναι να χειριστούμε την κάθε κατηγορία εξόδου ως ανεξάρτητη διανομή Bernoulli, τιμωρώντας κάθε νευρώνα εξόδου ανεξάρτητα από τους υπόλοιπους. Επομένως, η binary cross-entropy ταιριάζει απόλυτα στις ανάγκες του προβλήματος μας.

Δεδομένου ότι σκοπός του αλγορίθμου είναι να μεγιστοποιήσει την ακρίβεια του (και δυνητικά να ελαχιστοποιήσει το μέγεθος λάθους), θα χρησιμοποιήσουμε μία ακόμη μέθοδο που μας προσφέρει το πακέτο Keras, έναν βελτιστοποιητή (optimizer). Δουλειά του βελτιστοποιητή είναι να ελαχιστοποιεί το μέγεθος λάθους προσαρμόζοντας τα βάρη (weights) και τον ρυθμό εκμάθησης (learning rate) του μοντέλου. Μετά από αρκετές δοκιμές καταλήξαμε στη χρήση του βελτιστοποιητή *RMSprop* με ρυθμό εκμάθησης 0.0001, καθώς ήταν ο μόνος συνδυασμός υπερπαραμέτρων ο οποίος κατάφερε σταδιακά να μειώσει το μέγεθος λάθους και να αυξήσει το ποσοστό ακρίβειας του αλγορίθμου χωρίς να προκαλέσει υπερπροσαρμογή (overfitting) ή εκθετικά αυξανόμενο μέγεθος λάθους.

5.2.5 Εκπαίδευση του μοντέλου

Τελικό βήμα είναι η εκπαίδευση του μοντέλου το οποίο δομήσαμε προσεκτικά και συνθέσαμε με τη βοήθεια υπερπαραμέτρων. Κατά την εκπαίδευση του μοντέλου καλούμαστε να ορίσουμε τον

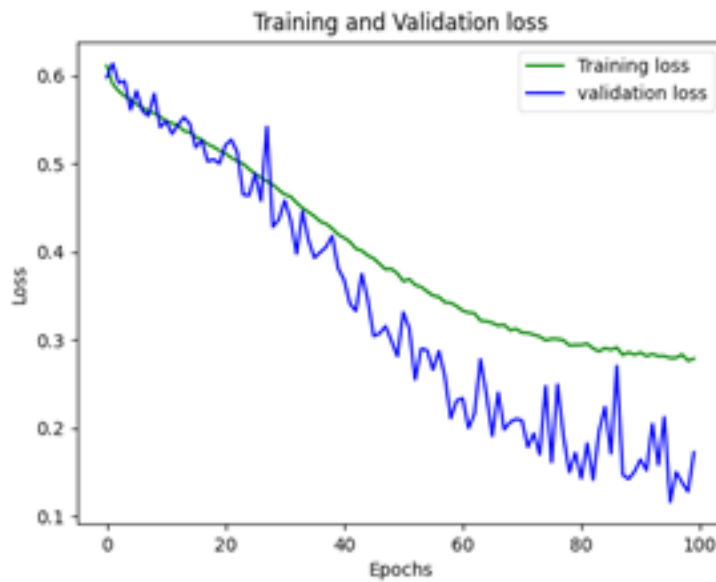
αριθμό των εποχών (epochs) και το μέγεθος δέσμης δεδομένων (batch size). Το δεύτερο δεν αποτέλεσε πρόβλημα καθώς η δοκιμή διαφορετικών μεγεθών δεν επηρέασε αισθητά τον αλγόριθμο, επομένως επιλέξαμε το μέγεθος το οποίο απέδιδε καλύτερα, δηλαδή `batch_size=32`. Ωστόσο, ο αριθμός εποχών μας προβλημάτισε σε μεγαλύτερο βαθμό αφού δοκιμάζοντας διαφορετικούς βελτιστοποιητές και ρυθμούς εκμάθησης, άλλαζε σημαντικά και η συμπεριφορά του αλγορίθμου ανά εποχή. Επομένως, χρειάστηκε πρώτα να ρυθμίσουμε τις υπερπαραμέτρους μας, και ύστερα μέσα από ελάχιστες δοκιμές ήταν σαφές ότι το μοντέλο έφτανε στο μέγιστο της απόδοσης του μετά από 85 εποχές.

Σημαντικό ρόλο στην λήψη αποφάσεων όσον αφορά την επιλογή υπερπαραμέτρων και εποχών του μοντέλου έπαιξε η οπτικοποίηση της απόδοσης του με το εργαλείο `matplotlib`.

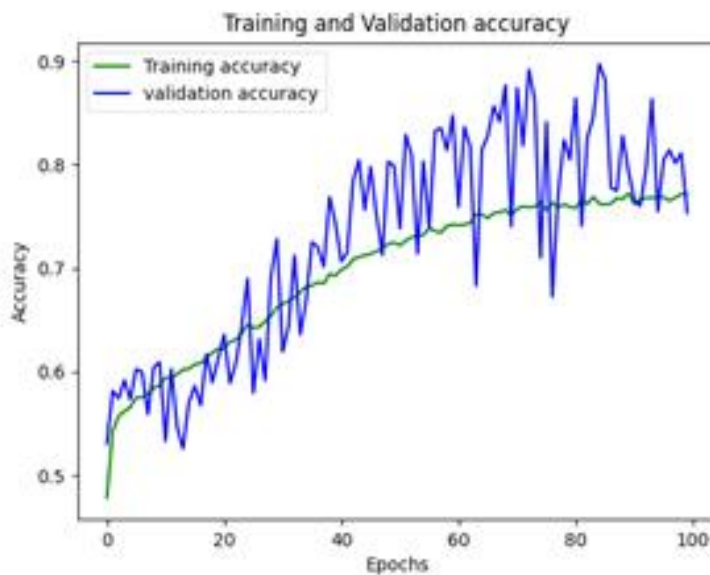
5.2.6 Οπτικοποίηση με `matplotlib`

Όταν ρυθμίζουμε ένα μοντέλο το οποίο αποσκοπεί στη λύση ενός προβλήματος είναι σημαντικό να γνωρίζουμε εάν πλησιάζουμε ή αποκλίνουμε από αυτήν. Πολλές φορές η εμπειρία του ερευνητή είναι αρκετή για τη σωστή ρύθμιση του μοντέλου, ωστόσο η οπτικοποίηση της απόδοσης του θα έπρεπε να αποτελεί αναπόσπαστο κομμάτι αυτής της διαδικασίας σε όλες τις φάσεις της. Το πιο δημοφιλές εργαλείο για την οπτικοποίηση δεδομένων και τον σχεδιασμό γραφημάτων είναι το πακέτο `matplotlib` το οποίο είναι γραμμένο σε `python`. Η κλάση `pyplot` του `matplotlib` μας επιτρέπει να σχεδιάσουμε γραφήματα απόδοσης για τις προβλέψεις του μοντέλου ώστε να οπτικοποιήσουμε τις μετρικές ακρίβειας και λάθους ανά εποχές. Τα γραφήματα των διαγραμμάτων 5.1 και 5.2 τα οποία παράχθηκαν με το `pyplot` αναπαριστούν την απόδοση ακρίβειας/εποχής και λάθους/εποχής του μοντέλου μας αντίστοιχα.

Μπορούμε ακόμη να οπτικοποιήσουμε τη μορφή των δεδομένων μας κατά την εφαρμογή φίλτρων ώστε να είμαστε σε θέση να επιβλέπουμε τον τρόπο με τον οποίο ο αλγόριθμος εξάγει χαρακτηριστικά από αυτά. Ακόμη, μέσα από την οπτικοποίηση των δεδομένων κατά την πρόβλεψη κατηγορίας από τον αλγόριθμο, είμαστε σε θέση να μελετήσουμε πιθανά μοτίβα τα οποία μπερδεύουν τον αλγόριθμο ώστε να προβλέψει λανθασμένα. Επομένως, η οπτικοποίηση των δεδομένων είναι εξαιρετικά σημαντική διαδικασία και θα έπρεπε να αποτελεί αναπόσπαστο κομμάτι της παραγωγής ενός μοντέλου καθ' όλα τα στάδιά της.



Διάγραμμα 5.3 – Απώλεια εκπαίδευσης/επικύρωσης

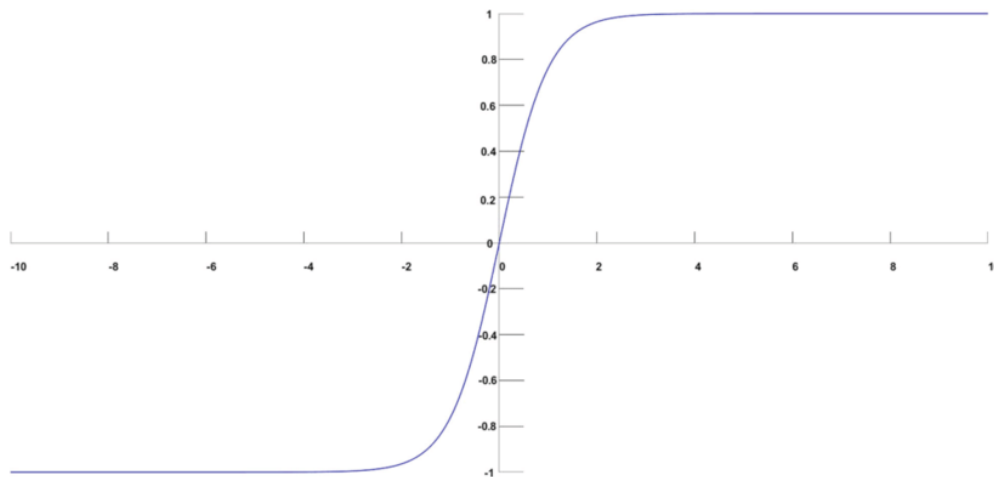


Διάγραμμα 5.4 – Ακρίβεια εκπαίδευσης/επικύρωσης

5.3.1 Προβλήματα

Κατά την διαδικασία σχεδίασης ενός νευρωνικού δικτύου υπάρχουν διάφορες παράμετροι και υπερπαράμετροι οι οποίες παρουσιάζουν σημαντικό αντίκτυπο στην απόδοση του αλγορίθμου. Επομένως, παρότι ενδεχομένως να υπάρχουν αρκετοί συνδυασμοί αυτών οι οποίοι να επιτυγχάνουν παρόμοιες αποδόσεις, σκοπός μας είναι να επιλέξουμε τον συνδυασμό ο οποίος αποδίδει τα μέγιστα αλλά επίσης να κατανοήσουμε τον λόγο για τον οποίο συμβαίνει αυτό.

Μία από τις μεγαλύτερες προκλήσεις που αντιμετωπίσαμε κατά την σχεδίαση του μοντέλου ήταν η επιλογή συνάρτησης ενεργοποίησης για το επίπεδο εξόδου του νευρωνικού μας δικτύου. Αρχικά, εμπνευσμένοι από τη βιβλιογραφία και από παρόμοια προβλήματα, επιλέξαμε τη μέθοδο softmax σαν μέθοδο ενεργοποίησης (Διάγραμμα 5.3). Η softmax σε συνδυασμό με categorical cross-entropy είναι η καθιερωμένη μέθοδος ενεργοποίησης για προβλήματα πολλαπλών κατηγοριών, ωστόσο το συγκεκριμένο δεν ήταν ένα κλασσικό πρόβλημα πολλαπλών κατηγοριών. Χρησιμοποιώντας τη softmax ο αλγόριθμος προσπαθούσε να προβλέψει το ποσοστό κατά το οποίο μία εικόνα ανήκει σε καθεμία από 3 κατηγορίες, με συνολική τιμή 1 (100%) μοιρασμένη σε αυτές. Δεδομένου, όμως, ότι μία εικόνα μπορεί να ανήκει σε περισσότερες από μία κατηγορίες, η softmax έπανε να είναι η κατάλληλη μέθοδος για το πρόβλημά μας. Αντιθέτως, χρειαζόμαστε μία μέθοδο η οποία θα χειρίζεται κάθε κατηγορία εξόδου ανεξάρτητα από τις υπόλοιπες. Ακόμη, ο αλγόριθμος θα πρέπει να τιμωρείται για κάθε λάθος πρόβλεψη ανεξάρτητα με την επιτυχία πρόβλεψης των υπόλοιπων κατηγοριών. Μετά από μελέτη, καταλήξαμε στην μέθοδο sigmoid η οποία, σε συνδυασμό με την binary cross-entropy, βελτίωσε την απόδοση του αλγορίθμου κατά 65%.



Διάγραμμα 5.5 – Η συνάρτηση ενεργοποίησης softmax
(<https://www.researchgate.net/publication/325856086/figure/fig1/AS:723221292789765@1549440801787/Softmax-function-image.png>)

5.3.2 Συμπεράσματα

Στην παρούσα πειραματική διαδικασία παρουσιάζεται η εκπαίδευση ενός νευρωνικού δικτύου και η χρήση του για την πραγματοποίηση προβλέψεων. Έπειτα από 100 εποχές, παρατηρήσαμε ότι ο αλγόριθμος επιτυγχάνει μέγιστη απόδοση στην 85η εποχή. Συγκεκριμένα, η ακρίβεια πρόβλεψης του μοντέλου έφτασε το 0.897 (~90%) ενώ ελαχιστοποίησε το μέγεθος λάθους στο 0.13. Συγκριτικά με αποτελέσματα άλλων μοντέλων πάνω στο ίδιο πρόβλημα, το συγκεκριμένο μοντέλο αποδίδει κατά περίπου 15% καλύτερα. Η διαφορά απόδοσης του μοντέλου μας σε σχέση με παρόμοια μοντέλα θεωρούμε ότι αποδίδεται στην επιλογή καταλληλότερης συνάρτησης λάθους σε συνδυασμό με τη συνάρτηση ενεργοποίησης του επιπέδου εξόδου του νευρωνικού δικτύου μας. Πράγματι, η επιλογή διαφορετικών συνδυασμών συναρτήσεων λάθους και ενεργοποίησης απέδιδε κατά 15-20% χαμηλότερη ακρίβεια προβλέψεων ενώ αποτύγχανε να

ελαχιστοποιήσει το μέγεθος λάθους στα επιθυμητά επίπεδα.

5.3.3 Μελλοντική Δουλειά

Για την αποτελεσματικότερη εκπαίδευση του νευρωνικού δικτύου, θα μπορούσαν να εφαρμοστούν επιπλέον τεχνικές. Για παράδειγμα, η χρήση εικόνων μεγέθους 30% του αρχικού παρότι βελτίωσε αισθητά την ταχύτητα εκπαίδευσης του μοντέλου, ενδεχομένως να επηρέασε την ακρίβειά του σε σχέση με τη χρήση εικόνων στο αρχικό τους μέγεθος. Επίσης, η χρήση εικόνων υψηλότερης ανάλυσης θα βοηθούσε στην πιο λεπτομερή ανάλυση τους βελτιώνοντας την ακρίβεια του μοντέλου. Ακόμη, το ήδη εκπαιδευμένο μοντέλο θα μπορούσε να χρησιμοποιηθεί για επιπλέον σκοπούς. Για παράδειγμα, δεδομένου ότι το νευρωνικό μας δίκτυο έχει εκπαιδευτεί ώστε να αναγνωρίζει τα χαρακτηριστικά τα οποία ορίζουν ένα είδος ταινίας, θα ήταν δυνατό να αντιστρέψουμε αυτή τη διαδικασία ώστε να παράγει πόστερ δεχόμενο ένα είδος ταινίας σαν είσοδο.

Παράρτημα

1) get_data.py

```
import os
import sys

import movies_dataset as movies

download = '-download' in sys.argv
resize = '-resize' in sys.argv

min_year = 0
for arg in sys.argv:
    if arg.startswith('-min_year='):
        min_year = int(arg.split('=')[1])

ratios = [30, 40, 50, 60, 70]
images_path = 'data/images/'
base_images_path = images_path + '100/'

# create images directory if not exists
if not os.path.isdir(base_images_path):
    os.makedirs(base_images_path)

# download posters
if download:
    movies.download_posters(min_year=min_year)

# resize images (so we can build test models more quickly)
if resize:
    for ratio in ratios:
        directory_path = images_path + str(ratio)
        # directory = os.path.join(os.getcwd(), directory_path)
        if not os.path.isdir(directory_path):
            os.makedirs(directory_path)
            command = 'mogrify -path "' + directory_path + '/' -resize ' \
                + str(ratio) + '% ' + base_images_path + '*.jpg -verbose'
            print(command)
            os.system(command)
```

2) create_db.py

```
import movies_dataset as movies

import time
from matplotlib import pyplot as plt

min_year = 1977
max_year = 2017
epochs = 50
ngenres = 7
genres = movies.list_genres(ngenres)

print('Creating file...')

t1 = time.perf_counter()
for ratio in [30]:
    # we load the data once for each ratio, so we can use it for multiple versions, epochs, etc.
    xfile = "data/posters"+str(ratio)+".bin"
    yfile = "data/posters"+str(ratio)+"_"+str(ngenres)+"genres.bin"
    movies.milix_create_binfiles(xfile, yfile, ratio, genres)
t2 = time.perf_counter()
print(f"Done in {t2-t1:0.4f} seconds.")

# load and show images
ratio = 30
xfile = "data/posters"+str(ratio)+".bin"
yfile = "data/posters"+str(ratio)+"_"+str(ngenres)+"genres.bin"
t1 = time.perf_counter()
X, ids = movies.milix_load_posters(xfile)
y = movies.milix_load_genres(yfile)
t2 = time.perf_counter()
print(f"Done in {t2-t1:0.4f} seconds.")
print(f"X.shape: {X.shape}")
print(f"y.shape: {y.shape}")
print(ids[2])
print(y[2])
pixels = X[2]
plt.imshow(pixels) #, cmap='gray')
plt.show()

print('Done.')
```

3) movies_dataset.py

```
"""
Manage movies data (extracted from /data/MovieGenre.csv).
"""
```

```
import io
import os.path
import urllib.request
```

```
import numpy as np
import pandas as pd
from PIL import Image
```

```
from matplotlib import pyplot as plt
import struct
```

```
images_folder = 'data/images/'
test_data_ratio = 7 # 14.3%
validation_data_ratio = 6 # 14.3%
parsed_movies = [] # cache
```

```
class Movie:
    imdb_id = 0
    title = ""
    year = 0
    genres = []
    poster_url = ""
```

```
# milix: added size in parameters
def poster_file_exists(self,size=100) -> bool:
    return os.path.isfile(self.poster_file_path(size))
```

```
def download_poster(self):
    try:
        response = urllib.request.urlopen(self.poster_url)
        data = response.read()
        file = open(self.poster_file_path(), 'wb')
        file.write(bytearray(data))
        file.close()
        return data
    except:
```

```

print('-> error')

def poster_file_path(self, size=100) -> str:
return images_folder + str(size) + "/" + self.poster_file_name()

def poster_file_name(self):
return str(self.imdb_id) + '.jpg'

def is_valid(self) -> bool:
return self.poster_url.startswith('https://') \
and 1900 <= self.year <= 2018 \
and len(self.title) > 1 \
and len(self.genres) > 1

def to_rgb_pixels(self, poster_size):
data = open(images_folder + str(poster_size) + '/' + str(self.imdb_id) + '.jpg', "rb").read()
image = Image.open(io.BytesIO(data))
rgb_im = image.convert('RGB')
pixels = []
for x in range(image.size[0]):
row = []
for y in range(image.size[1]):
r, g, b = rgb_im.getpixel((x, y))
pixel = [r / 255, g / 255, b / 255]
row.append(pixel)
pixels.append(row)
return pixels, image #new (image)

# milix
def read_image_rgb(self, poster_size):
data = open(images_folder + str(poster_size) + '/' + str(self.imdb_id) + '.jpg', "rb").read()
image = Image.open(io.BytesIO(data))
rgb_im = image.convert('RGB')
return rgb_im

def get_genres_vector(self, genres):
if len(genres) == 1:
has_genre = self.has_genre(genres[0])
return [int(has_genre), int(not has_genre)]
else:
vector = []
if self.has_any_genre(genres):
for genre in genres:
vector.append(int(self.has_genre(genre)))

return vector

```

```

def short_title(self) -> str:
    max_size = 20
    return (self.title[:max_size] + '..') if len(self.title) > max_size else self.title

def is_test_data(self) -> bool:
    return self.imdb_id % test_data_ratio == 0

def has_any_genre(self, genres) -> bool:
    return len(set(self.genres).intersection(genres)) > 0

def has_genre(self, genre) -> bool:
    return genre in self.genres

def __str__(self):
    return self.short_title() + ' (' + str(self.year) + ')'

def download_posters(min_year=0):
    for movie in list_movies():
        print(str(movie))
        if movie.year >= min_year:
            if not movie.poster_file_exists():
                movie.download_poster()
            if movie.poster_file_exists():
                print('-> downloaded')
            else:
                print('-> could not download')
            else:
                print('-> already downloaded')
            else:
                print('-> skip (too old)')

def load_genre_data(min_year, max_year, genres, ratio, data_type, verbose=True):
    xs = []
    ys = []
    for year in reversed(range(min_year, max_year + 1)):
        if verbose:
            print('loading movies', data_type, 'data for', year, '...')
        xs_year, ys_year = _load_genre_data_per_year(year, genres, ratio, data_type)
        _add_to(xs_year, xs)
        _add_to(ys_year, ys)
        if verbose:
            print('->', len(xs_year))
    return np.concatenate(xs), np.concatenate(ys)

```

```

def _load_genre_data_per_year(year, genres, poster_ratio, data_type):
    xs = []
    ys = []

    count = 1
    for movie in list_movies(year, genres):
        if movie.poster_file_exists():
            if (data_type == 'train' and not movie.is_test_data() and count % validation_data_ratio != 0) \
            or (data_type == 'validation' and not movie.is_test_data() and count % validation_data_ratio ==
            0) \
            or (data_type == 'test' and movie.is_test_data()):
                x, img = movie.to_rgb_pixels(poster_ratio) #new(img)
                y = movie.get_genres_vector(genres)
                xs.append(x)
                ys.append(y)
                count += 1

    xs = np.array(xs, dtype='float32')
    ys = np.array(ys, dtype='uint8')
    return xs, ys

def _add_to(array1d, array2d):
    if len(array1d) > 0:
        array2d.append(array1d)

'''
milix_create_binfiles
create fx (images) anf fy (genres=labels)

fx format
-----
nitems: int32
height: int32
width: int32
image(1): imdb_id[int32], rrrrggggbbbb
image(2): imdb_id[int32], rrrrggggbbbb
...,
image(nitems)

'''

def milix_create_binfiles(xfilename, yfilename, ratio, genres):

```

```

min_year = 1977
max_year = 2017
totcount = 0 # nitems (will be updated at the end)
fx = open(xfilename, "wb")
fy = open(yfilename, "wb")
fx.write((totcount).to_bytes(4, byteorder='big', signed=False))
fy.write((totcount).to_bytes(4, byteorder='big', signed=False))
for year in reversed(range(min_year, max_year+1)):
    print('adding movie data for ', year, '...')
    count = 0
    movies = list_movies(year, genres)
    for movie in movies:
        if movie.poster_file_exists(ratio):
            img = movie.read_image_rgb(ratio)
            gen = movie.get_genres_vector(genres)
            if (totcount==0):
                # first image, write height and width to the file
                fx.write((img.size[1]).to_bytes(4, byteorder='big', signed=False)) # height
                fx.write((img.size[0]).to_bytes(4, byteorder='big', signed=False)) # width
                # write genres size
                fy.write((len(gen)).to_bytes(4, byteorder='big', signed=False))

            # write imdb_id
            fx.write((movie.imdb_id).to_bytes(4, byteorder='big', signed=False))
            # write pixel data (first r then g and then b)
            red = []
            green = []
            blue = []
            for y in range(img.size[1]):
                for x in range(img.size[0]):
                    r, g, b = img.getpixel((x, y))
                    red.append(r)
                    green.append(g)
                    blue.append(b)

            fx.write(bytearray(red))
            fx.write(bytearray(green))
            fx.write(bytearray(blue))

            ""
            pixels = np.array(blue).reshape((img.size[1], img.size[0]))
            plt.imshow(pixels, cmap='gray')
            plt.show()
            ""

            # write genres
            #print(movie.imdb_id, gen)

```

```

fy.write(bytearray(gen))

count += 1
totcount += 1

if count > 0:
    print(f"Found {count} movies in year {year}.")

fx.seek(0)
fx.write((totcount).to_bytes(4, byteorder='big', signed=False))
fx.close()
fy.seek(0)
fy.write((totcount).to_bytes(4, byteorder='big', signed=False))
fy.close()


def milix_load_posters(fname):
    fx = open(fname, "rb")
    n = struct.unpack('>I', fx.read(4))[0]
    h = struct.unpack('>I', fx.read(4))[0]
    w = struct.unpack('>I', fx.read(4))[0]
    print(f"Going to read {n} posters of size {h}x{w}")
    X = []
    ids = []
    for i in range(n):
        imdb_id = struct.unpack('>I', fx.read(4))[0]
        ids.append(imdb_id)
        r = np.array(bytearray(fx.read(h*w))).reshape((h,w))
        g = np.array(bytearray(fx.read(h*w))).reshape((h,w))
        b = np.array(bytearray(fx.read(h*w))).reshape((h,w))
        image = np.dstack((r,g,b))
        # bottleneck in performance here
        ""
        image = np.zeros((h,w,3))
        for y in range(h):
            for x in range(w):
                image[y,x,0] = r[y,x];
                image[y,x,1] = g[y,x];
                image[y,x,2] = b[y,x];
            ""
        X.append(image/255.0)

    fx.close()
    X = np.array(X, dtype='float32')
    ids = np.array(ids, dtype='uint32')
    return X, ids

```

```

def milix_load_genres(fname):
    fy = open(fname, "rb")
    n = struct.unpack('>I', fy.read(4))[0]
    sz = struct.unpack('>I', fy.read(4))[0]
    y = []
    for i in range(n):
        g = []
        for j in range(sz): g.append(int.from_bytes(fy.read(1), 'big'))
        y.append(g)

    fy.close()
    y = np.array(y, dtype='uint8')
    return y

def list_movies(year=None, genres=None):
    if len(parsed_movies) == 0:
        data = pd.read_csv('data/MovieGenre.csv', encoding='ISO-8859-1')
        for index, row in data.iterrows():
            movie = _parse_movie_row(row)
            if movie.is_valid():
                parsed_movies.append(movie)

    parsed_movies.sort(key=lambda m: m.imdb_id)

    result = parsed_movies

    if year is not None:
        result = [movie for movie in result if movie.year == year]

    if genres is not None:
        result = [movie for movie in result if movie.has_any_genre(genres)]

    return result

def _parse_movie_row(row) -> Movie:
    movie = Movie()
    movie.imdb_id = int(row['imdbId'])
    movie.title = row['Title'][:7]
    year = row['Title'][-5:-1]
    if year.isdigit() and len(year) == 4:
        movie.year = int(row['Title'][-5:-1])

```

```

url = str(row['Poster'])
if len(url) > 0:
    movie.poster_url = url.replace("'", "")

genre_str = str(row['Genre'])
if len(genre_str) > 0:
    movie.genres = genre_str.split('|')

return movie

def search_movie(imdb_id=None, title=None) -> Movie:
    movies = list_movies()
    for movie in movies:
        if imdb_id is not None and movie.imdb_id == imdb_id:
            return movie
        if title is not None and movie.title == title:
            return movie

```

```

def list_genres(number):
    if number == 3:
        return ['Comedy', 'Drama', 'Action']
    if number == 7:
        return list_genres(3) + ['Animation', 'Romance', 'Adventure', 'Horror']
    if number == 14:
        return list_genres(7) + ['Sci-Fi', 'Crime', 'Mystery', 'Thriller', 'War', 'Family', 'Western']

```

4) movies_genre_model.py

```

import os
import time

from tensorflow.python.keras.layers import Conv2D, MaxPooling2D
from tensorflow.python.keras.layers import Dense, Dropout, Flatten
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.optimizer_v2 import rmsprop
from tensorflow.python.keras.optimizer_v2 import adam
import movies_dataset as movies
import matplotlib.pyplot as plt

def get_kernel_dimensions(version, shape, divisor):
    image_width = shape[1]

```

```

# original
if version == 1:
    return 3, 3

# square 10% width
if version == 2:
    return int(0.1 * image_width / divisor), int(0.1 * image_width / divisor)

# square 20% width
if version == 3:
    return int(0.2 * image_width / divisor), int(0.2 * image_width / divisor)

def build(version, min_year, max_year, genres, ratio, epochs,
          x_train=None, y_train=None, x_validation=None, y_validation=None):
    # log
    print()
    print('version:', version)
    print('min_year:', min_year)
    print('max_year:', max_year)
    print('genres:', genres)
    print('ratio:', ratio)
    print()

    # load data if not provided
    if x_train is None or y_train is None or x_validation is None or y_validation is None:
        begin = time.time()
        x_train, y_train = movies.load_genre_data(min_year, max_year, genres, ratio, 'train')
        x_validation, y_validation = movies.load_genre_data(min_year, max_year, genres, ratio,
        'validation')
        print('loaded in', (time.time() - begin) / 60, 'min.')
    else:
        print('data provided in arguments')

    print()
    print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_validation.shape[0], 'validation samples')

    # build model
    num_classes = len(y_train[0])
    kernel_dimensions1 = get_kernel_dimensions(version, x_train.shape, 1)
    kernel_dimensions2 = get_kernel_dimensions(version, x_train.shape, 2)
    print('kernel_dimensions1:', kernel_dimensions1)
    print('kernel_dimensions2:', kernel_dimensions2)

```

```

model = Sequential([
    Conv2D(32, kernel_dimensions1, padding='same', input_shape=x_train.shape[1:],
activation='relu'),
    Conv2D(32, kernel_dimensions2, activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Conv2D(64, kernel_dimensions2, padding='same', activation='relu'),
    Conv2D(64, kernel_dimensions2, activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='sigmoid')
])

opt = rmsprop.RMSProp(learning_rate=0.001, decay=1e-6)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
print(model.summary())

history = model.fit(x_train, y_train, batch_size=32, epochs=epochs,
validation_data=(x_validation, y_validation))

loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,51)
plt.plot(epochs, loss_train, 'g', label="Training loss")
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title("Training and Validation loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
if not os.path.isdir('saved_plots/'):
    os.makedirs('saved_plots/loss_plots')
    os.makedirs('saved_plots/accuracy_plots')
loss_fig = plt.gcf()
plt.draw()
loss_fig.savefig('saved_plots/loss_plots/' + str(ratio) + '_' +
                + str(version) +
                + '.png')

plt.clf()
loss_train = history.history['accuracy']
loss_val = history.history['val_accuracy']

```

```

epochs = range(1,51)
plt.plot(epochs, loss_train, 'g', label="Training accuracy")
plt.plot(epochs, loss_val, 'b', label='validation accuracy')
plt.title("Training and Validation accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
acc_fig = plt.gcf()
plt.draw()
acc_fig.savefig('saved_plots/accuracy_plots/' + str(ratio) + '_' \
                + str(version) \
                + '.png')

# create dir if none
save_dir = os.path.join(os.getcwd(), 'saved_models')
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)

# save model
model_file_name = 'genres' \
    + '_' + str(min_year) + '_' + str(max_year) \
    + '_g' + str(len(genres)) \
    + '_r' + str(ratio) \
    + '_e' + str(epochs) \
    + '_v' + str(version) + '.h5'

model_path = os.path.join(save_dir, model_file_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)

```

5) movies_milix_model.py

```

import os
import time

import tensorflow.keras
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.models import Sequential

import matplotlib.pyplot as plt

import movies_dataset as movies

```

```

def get_kernel_dimensions(version, shape, divisor):
    image_width = shape[1]

    # original
    if version == 1: return 3, 3

    # square 10% width
    if version == 2:
        return int(0.1 * image_width / divisor), int(0.1 * image_width / divisor)

    # square 20% width
    if version == 3:
        return int(0.2 * image_width / divisor), int(0.2 * image_width / divisor)

def build(version, min_year, max_year, genres, ratio, epochs,
          x_train=None, y_train=None, x_validation=None, y_validation=None):
    # log
    print()
    print('version:', version)
    print('min_year:', min_year)
    print('max_year:', max_year)
    print('genres:', genres)
    print('ratio:', ratio)
    print()

    # load data if not provided
    if x_train is None or y_train is None or x_validation is None or y_validation is None:
        print('no data provided in arguments')

    print()
    print("Train: X=%s, y=%s" % (x_train.shape, y_train.shape))
    print("Test: X=%s, y=%s" % (x_validation.shape, y_validation.shape))
    print(x_validation.shape[0], 'validation samples')

    # build model
    num_classes = len(y_train[0])
    kernel_dimensions1 = get_kernel_dimensions(version, x_train.shape, 1)
    kernel_dimensions2 = get_kernel_dimensions(version, x_train.shape, 2)
    print('kernel_dimensions1:', kernel_dimensions1)
    print('kernel_dimensions2:', kernel_dimensions2)
    #new
    print('x_train.shape[1:]=', x_train.shape[1:])

    model = Sequential([

```

```

Conv2D(32, kernel_dimensions1, padding='same', input_shape=x_train.shape[1:],
activation='relu'),
Conv2D(32, kernel_dimensions1, activation='relu'),
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.25),

Conv2D(64, kernel_dimensions2, padding='same', activation='relu'),
Conv2D(64, kernel_dimensions2, activation='relu'),
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.25),

Flatten(),
Dense(512, activation='relu'),
Dropout(0.5),
Dense(num_classes, activation='sigmoid')
])

'''
    compile the model using binary cross-entropy rather than categorical cross-entropy.
    This may seem counterintuitive for multi-label classification; however, the goal is to
    treat each output label as an independent Bernoulli distribution and we want to penalize
    each output node independently.
'''

opt = tensorflow.keras.optimizers.RMSprop(lr=0.0001, decay=1e-6)
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
print(model.summary())

history = model.fit(x_train, y_train, batch_size=32, epochs=epochs,
validation_data=(x_validation, y_validation))

# create dir if none
save_dir = os.path.join(os.getcwd(), 'saved_models')
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)

# save model
model_file_name = 'genres' \
+ '_' + str(min_year) + '_' + str(max_year) \
+ '_g' + str(len(genres)) \
+ '_r' + str(ratio) \
+ '_e' + str(epochs) \
+ '_v' + str(version) + '.h5'

loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(0,60)

```

```

plt.plot(epochs, loss_train, 'g', label='Training loss')
plt.plot(epochs, loss_val, 'b', label='validation loss')
plt.title("Training and Validation loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
if not os.path.isdir('saved_plots/'):
    os.makedirs('saved_plots/loss_plots')
os.makedirs('saved_plots/accuracy_plots')
loss_fig = plt.gcf()
plt.draw()
loss_fig.savefig('saved_plots/loss_plots/' + str(ratio) + '_' \
                + str(version) \
                + '.png')

plt.clf()
loss_train = history.history['accuracy']
loss_val = history.history['val_accuracy']
epochs = range(0,60)
plt.plot(epochs, loss_train, 'g', label='Training accuracy')
plt.plot(epochs, loss_val, 'b', label='validation accuracy')
plt.title("Training and Validation accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
acc_fig = plt.gcf()
plt.draw()
acc_fig.savefig('saved_plots/accuracy_plots/' + str(ratio) + '_' \
                + str(version) \
                + '.png')

model_path = os.path.join(save_dir, model_file_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)

```

6) prepare_dcgan_dataset.py

```

import os
import sys
from shutil import copyfile

import movies_dataset as movies

min_year = 0
max_year = 2017

```

```

ratio = 50
genres_white_list = []
genres_black_list = []
genres = []
path = 'dcgan_movies_posters'

# parse arguments
for arg in sys.argv:
    if arg.startswith('-min_year='):
        min_year = int(arg.split('=')[1])

    if arg.startswith('-max_year='):
        max_year = int(arg.split('=')[1])

    if arg.startswith('-ratio='):
        ratio = int(arg.split('=')[1])

    if arg.startswith('-include_genres='):
        genres_white_list = arg.split('=')[1].split(',')

    if arg.startswith('-exclude_genres='):
        genres_black_list = arg.split('=')[1].split(',')

# filter genres
if len(genres_white_list) > 0:
    genres = genres_white_list
else:
    genres = list(set(movies.list_genres(14)) - set(genres_black_list))

genres.sort()

# log
print()
print('min_year:', min_year)
print('max_year:', max_year)
print('genres:', genres)
print('ratio:', ratio)
print()

# collect movies
all_movies = []
for year in reversed(range(min_year, max_year + 1)):
    all_movies += movies.list_movies(year=year, genres=genres)

print('movies collected:', len(all_movies))

```

```

# create folder if doesn't exist
if not os.path.isdir(path):
    os.makedirs(path)

# delete content if any
for the_file in os.listdir(path):
    file_path = os.path.join(path, the_file)
    try:
        if os.path.isfile(file_path):
            os.unlink(file_path)
    except Exception as e:
        print(e)

# copy files
for movie in all_movies:
    if movie.poster_file_exists():
        copyfile(movie.poster_file_path(ratio), path + '/' + movie.poster_file_name())

# TODO: not very clean
# convert all to RGB
command = './convert_to_RGB.sh'
print(command)
os.system(command)

```

7) tests.py

```

import operator
from os import listdir
from os.path import isfile, join

import numpy as np
from keras.models import load_model

import movies_dataset as movies

saved_models_folder = 'saved_models/'
align = 33
my_format = "{:.0%}"
eval_models = True
print_summary = True
print_test_movies = True
crop_results = 3

class TransferModel:

```

```

min_year = 0
max_year = 0
genres = []
ratio = 0
epochs = 0
version = 1
file_path = ""
model = None

def eval(self):
    print('loading test data...')
    x_test, y_test = movies.load_genre_data(self.min_year, self.max_year, self.genres,
self.ratio, data_type='test',
                                         verbose=False)
    print('Evaluating model...')
    scores = self.model.evaluate(x_test, y_test, verbose=0)
    print('Test loss:', scores[0])
    print('Test accuracy:', scores[1])

def predict(self, movie):
    x = [movie.to_rgb_pixels(self.ratio)]
    x = np.array(x, dtype='float32')
    return self.model.predict(x)

def load(self):
    self.model = load_model(self.file_path)

def __str__(self):
    return 'Model v' + str(self.version) \
        + ' (' + str(self.min_year) + '-' + str(self.max_year) \
        + ' / g' + str(len(self.genres)) \
        + ' / r' + str(self.ratio) \
        + ' / e' + str(self.epochs) \
        + ')'

def parse_transfer_model(file_name):
    split = file_name.split('_')
    parsed = TransferModel()
    parsed.min_year = int(split[2])
    parsed.max_year = int(split[3])
    parsed.genres = movies.list_genres(int(split[4][1:]))
    parsed.ratio = int(split[5][1:])
    parsed.epochs = int(split[6][1:])
    parsed.version = int(split[7][1:].split('.')[0])
    parsed.file_path = file_name

```

```

return parsed

def list_model_files():
    return sorted(
        [f for f in listdir(saved_models_folder) if isfile(join(saved_models_folder, f)) and
        f.startswith('genres_')])

def repeat_to_length(string_to_expand, length):
    return (string_to_expand * (int(length / len(string_to_expand)) + 1))[:length]

def format_predictions(movie, genres, predictions):
    predictions_map = dict()
    for i in range(len(genres)):
        predictions_map[genres[i]] = predictions[0][i]

    sorted_predictions = sorted(predictions_map.items(), key=operator.itemgetter(1),
    reverse=True)

    predictions_str = []
    for genre, probability in sorted_predictions:
        if genre in movie.genres:
            is_present = "
        else:
            is_present = '[]'
        predictions_str.append(genre + is_present + ': ' + my_format.format(probability))

    spaces = repeat_to_length(' ', align - len(str(movie)))

    if crop_results is not None:
        return str(movie) + spaces + str(predictions_str[:crop_results])
    else:
        return str(movie) + spaces + str(predictions_str)

def main():
    for model_file_name in list_model_files():
        saved_model = parse_transfer_model(saved_models_folder + model_file_name)
        saved_model.load()
        print('-----')
        print('-----')
        print(saved_model)
        print('-----')
        if print_summary:

```

```

    print(saved_model.model.summary())
    if eval_models:
        saved_model.eval()

    test_movies = dict()

    test_movies['Comedy'] = ["Bienvenue chez les Ch'tis", "Frequently Asked Questions About
Time Travel",
                            "What We Do in the Shadows", "Hollywood Ending", "Whatever Works",
                            "The Mask",
                            "Liar Liar"]

    test_movies['Drama'] = ["No Country for Old Men", "The Martian", "Vanilla Sky"]

    test_movies['Action'] = ["The Matrix", "Man of Steel", "X-Men: Apocalypse", "Lara Croft:
Tomb Raider",
                            "Edge of Tomorrow", "Batman Forever", "Live Free or Die Hard"]

    test_movies['Horror'] = ["Dracula 2000", "The Blair Witch Project", "The Others",
                            "Aliens",
                            "Aliens vs. Predator: Requiem", "Alien: Resurrection"]

    test_movies['Animation'] = ["Paprika", "Castle in the Sky", "Spirited Away", "Zootopia",
                                "Trolls"]

    test_movies['Romance'] = ["Notting Hill", "Pretty Woman", "Bridget Jones's Diary"]

    test_movies['?'] = ["Pearl Harbor", "Twelve Monkeys", "The Truman Show", "Blade
Runner",
                        "Star Wars: Episode IV - A New Hope", "The Godfather", "A.I. Artificial
Intelligence",
                        "Enter the Void", "The Abyss", "Primer", "Coherence", "Pulp Fiction"]

    if print_test_movies:
        print()
        for expected_genre, movies_titles in sorted(test_movies.items()):
            print(' -- ' + expected_genre + ' -- ')
            for movie_title in movies_titles:
                movie = movies.search_movie(title=movie_title)
                if movie is not None:
                    predictions = saved_model.predict(movie)
                    print(format_predictions(movie, saved_model.genres, predictions))
                else:
                    print(movie_title + ' not found')
            print()

```

```
if __name__ == '__main__':  
    main()
```

8) train_model.py

```
import movies_dataset as movies  
import movies_milix_model  
  
import os  
import time  
from sklearn.model_selection import train_test_split  
  
min_year = 1977  
max_year = 2017  
epochs = 60  
ratio = 30  
ngenres = 3  
genres = movies.list_genres(ngenres)  
  
if not os.path.isdir('data/posters'):  
    os.makedirs('data/posters')  
    open(str(ratio)+".bin", 'w').close()  
    open(str(ratio)+"_"+str(ngenres)+"genres.bin", 'w').close()  
  
xfile = "data/posters"+str(ratio)+".bin"  
yfile = "data/posters"+str(ratio)+"_"+str(ngenres)+"genres.bin"  
  
print('Creating binfiles...')  
if not (os.path.isfile("data/posters"+str(ratio)+".bin") and  
os.path.isfile("data/posters"+str(ratio)+"_"+str(ngenres)+"genres.bin")):  
    movies.milix_create_binfiles(xfile, yfile, ratio, genres)  
  
#milix  
print('Loading datasets...')  
t1 = time.perf_counter()  
X, ids = movies.milix_load_posters(xfile)  
y = movies.milix_load_genres(yfile)  
t2 = time.perf_counter()  
print(f"Done in {t2-t1:0.4f} seconds.")  
  
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=49)  
X_train = X  
y_train = y  
X_test = X
```

```
y_test = y

# training version (1, 2 or 3)
version = 1
movies_milix_model.build(version, min_year, max_year, genres, ratio, epochs,
                          x_train=X_train,
                          y_train=y_train,
                          x_validation=X_test,
                          y_validation=y_test)

print('Done.')
```

Βιβλιογραφία – Αρθρογραφία

Δικτυογραφία

1. Μηχανική μάθηση, διαθέσιμο στο <https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7>
2. Aman Gupta, Feature Selection Techniques in Machine Learning, 2020, διαθέσιμο στο <https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/>
3. Avik Dutta, Stacking in Machine Learning, 2019, διαθέσιμο στο <https://www.geeksforgeeks.org/stacking-in-machine-learning>
4. Keith D. Foote, A Brief History of Machine Learning, 2021, διαθέσιμο στο <https://www.dataversity.net/a-brief-history-of-machine-learning/>
5. Machine Learning Tasks, διαθέσιμο στο <https://pythonprogramminglanguage.com/machine-learning-tasks/>
6. Chris Nicholson, A Beginner's Guide to Neural Networks and Deep Learning, διαθέσιμο στο <https://wiki.pathmind.com/neural-network>
7. François Chollet, Image classification from scratch, 2020, διαθέσιμο στο https://keras.io/examples/vision/image_classification_from_scratch/
8. Image classification with TensorFlow Lite Model Maker, 2022, διαθέσιμο στο https://www.tensorflow.org/lite/models/modify/model_maker/image_classification
9. Jason Brownlee, How To Improve Deep Learning Performance, 2016, διαθέσιμο στο <https://machinelearningmastery.com/improve-deep-learning-performance/>
10. Rodrigo Ceron, AI, machine learning and deep learning: What's the difference?, 2019, διαθέσιμο στο <https://www.ibm.com/blogs/systems/ai-machine-learning-and-deep-learning-whats-the-difference/>
11. Yoshua Bengio, Ian Goodfellow, Aaron Courville, Deep Learning, 2015, διαθέσιμο στο https://d1wqtxts1xzle7.cloudfront.net/62266271/Deep_Learning20200303-80130-1s42zvt-with-cover-page-v2.pdf?Expires=1661876523&Signature=JrWRql2SXAuK6tkvA1-7BYpTNYhdeNcFWUP5YnDhSU9WRuUwH0nirfeaqPtvGElhPC0iHLspx7gxrIY22xRZcOUZzWrm9uqyJ-HKbRvjF1d8it~-isyO-

- [KR0lpyLvlHcHXEdlWBcNq1zZUtaa8Vb4UBrjQkrkem26-1OcPHEX7Wq1dMp5ajAKHEC3b8GGcRZlyeXYKO9TW1Al14yvF8Ke5r7USoTowAPjSeUrMsAkoqyqNTWGLvzUdZU00eNCoXj8DIVRnZdYkv65NG2y0bPUhvIQ7iEVv9sqvf1JsENWgsxYMkZ0cgqsdw9Nmzc0Wz~PkP35x-COMFFopytElBCA &Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA](https://books.google.gr/books?hl=el&lr=&id=b06qDwAAQBAJ&oi=fnd&pg=PP9&dq=deep+learning&ots=oFTXNpVUQ&sig=B-7l49rx557FNV1qYOE8UYBnJKU&redir_esc=y#v=onepage&q&f=false)
12. John D. Kelleher, Deep Learning, 2019, διαθέσιμο στο https://books.google.gr/books?hl=el&lr=&id=b06qDwAAQBAJ&oi=fnd&pg=PP9&dq=deep+learning&ots=oFTXNpVUQ&sig=B-7l49rx557FNV1qYOE8UYBnJKU&redir_esc=y#v=onepage&q&f=false
 13. Timo Böhm, How to download Movie Posters as a Deep Learning Dataset, 2018, διαθέσιμο στο <https://towardsdatascience.com/movie-posters-81af5707e69a>
 14. Aravindpai Pai, CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning, 2020, διαθέσιμο στο <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/>
 15. Juergen Schmidhuber, Deep Learning, 2015, διαθέσιμο στο http://www.scholarpedia.org/article/Deep_Learning
 16. TFLearn, διαθέσιμο στο <http://tflearn.org/>
 17. TensorFlow, διαθέσιμο στο <https://www.tensorflow.org/>
 18. Keras, διαθέσιμο στο <https://keras.io/>
 19. Antonio Gulli, Sujit Pal, Deep Learning with Keras, 2017, διαθέσιμο στο https://books.google.gr/books?hl=el&lr=&id=20EwDwAAQBAJ&oi=fnd&pg=PP1&dq=deep+learning&ots=lIewcmgRO5&sig=WC8gSRxHbdVBXYTMCwPVFBggLfo&redir_esc=y#v=onepage&q=deep%20learning&f=false
 20. Python, διαθέσιμο στο <https://www.python.org/>
 21. Neha, Movie Genre from its Poster, 2018, διαθέσιμο στο <https://www.kaggle.com/datasets/neha1703/movie-genre-from-its-poster>
 22. Jason Brownlee, Train-Test Split for Evaluating Machine Learning Algorithms, 2020, διαθέσιμο στο <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>
 23. Shivam Bansal, Ultimate Guide to Understand and Implement Natural Language Processing (with codes in Python), 2017, διαθέσιμο στο <https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/>
 24. Dhruvil Shah, CNN Approach for Predicting Movie Genre from Posters, 2020, διαθέσιμο στο <https://towardsdatascience.com/cnn-approach-for-predicting-movie-genre-from-posters-95f122f88bc2>

25. Akhiljanardhanan, Multi-Label Image Classification using CNN (python), 2020, διαθέσιμο στο <https://medium.com/@akhiljanardhanan525/multi-label-image-classification-using-cnn-python-9c5dbddf7dc7>
26. https://github.com/benckx/dnn-movie-posters/blob/master/movies_genre_model.py

Εικόνες

Εικόνα 1.1 http://repfiles.kallipos.gr/html_books/93/04a-main.html

Εικόνα 1.2 <https://medium.com/swlh/5-ways-deep-learning-is-changing-our-day-to-day-life-7c17875b720c>

Εικόνα 1.3 <https://towardsdatascience.com/reducing-the-carbon-footprints-of-cnns-at-the-cost-of-interactions-depthwise-pointwise-conv-5df850ea33a4>

Εικόνα 1.4 http://repfiles.kallipos.gr/html_books/93/04a-main.html , σχ. 4.19

Εικόνα 2.3 https://oi.readthedocs.io/en/latest/computer_vision/segmentation/segmentation.html

Εικόνα 2.4 <https://medium.com/analytics-vidhya/weakly-supervised-learning-for-object-localization-4b73d4f4f4a6>

Εικόνα 2.5 <https://www.csc.com.gr/machine-learning-%CE%BC%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE-%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9/>

Εικόνα 2.6 <https://paperswithcode.com/lib/torchvision/faster-r-cnn>

Εικόνα 3.1 <https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848>

Εικόνα 3.2 https://www.researchgate.net/figure/VGGNet-architecture-19_fig2_333242381

Εικόνα 5.1 https://computersciencewiki.org/index.php/Max-pooling/_Pooling

Διαγράμματα

Διάγραμμα 1.1 <https://medium.com/@curiously/making-a-predictive-keyboard-using-recurrent-neural-networks-tensorflow-for-hackers-part-v-3f238d824218> , αρχικό σχέδιο από εδώ

Διάγραμμα 1.2 https://medium.com/@sarojthapa60_76302/what-is-overfitting-and-how-to-avoid-it-56d4b9ba0688

Διάγραμμα 2.1 <https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022>

Διάγραμμα 2.2
https://cezannec.github.io/Convolutional_Neural_Networks/

Διάγραμμα 2.3 https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1

Διάγραμμα 3.1 <https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/>

Διάγραμμα 3.2
https://jananisbabu.github.io/ResNet50_From_Scratch_Tensorflow/

Διάγραμμα 3.3 https://www.researchgate.net/figure/nception-module-of-GoogLeNet-This-figure-is-from-the-original-paper-10_fig3_312515254

Διάγραμμα 3.4 <https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>

Διάγραμμα 3.5 https://www.researchgate.net/figure/Overview-of-RDML-Random-Multimodel-Deep-Learning-for-classification-that-includes-n_fig1_324922651

Διάγραμμα 4.1 https://www.researchgate.net/figure/Data-flow-graph-example_fig2_224517165

Διάγραμμα 5.3
<https://www.researchgate.net/publication/325856086/figure/fig1/AS:723221292789765@1549440801787/Softmax-function-image.png>