

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



- Πτυχιακή Εργασία -

Αναγνώριση Καλλιτέχνη με την χρήση Convolutional Neural Network

Artist Identification using Convolutional Neural Networks

Ευφραΐμ Αναστασιάδης

A.M : Π2016157

Επιβλέπων: Μιχαήλ Στεφανιδάκης

10 Φεβρουαρίου 2022

Επιβλέπων

Μιχαήλ Στεφανιδάκης, Επίκουρος Καθηγητής

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Τριμελής Επιτροπή

Μ. Στεφανιδάκης, Επίκουρος Καθηγητής

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Θ. Ανδρόνικος, Αναπληρωτής Καθηγητής

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Κ. Κερμανίδου, Αναπληρωτής Καθηγητής

Ιόνιο Πανεπιστήμιο, Τμήμα Πληροφορικής

Περίληψη.....	5
Summary.....	5
1ο	
Introduction.....	6
2ο	
Θεωρία.....	7
2.1 A .I .(Artificial Intelligence).....	7
2.2 Μηχανική Μάθηση.....	8
2.3 Βαθιά Εκμάθηση.....	10
2.4 CNNs.....	11
2.5 Transfer Learning.....	14
2.6 ImagiNet.....	15
3ο Δημοφιλείς αρχιτεκτονικές.....	16
3.1 ResNet50.....	16

3.2 Efficient	
Net.....	17
.....	
3.3 Vision	
Transformer.....	18
.....	
3.4	
Xception.....	19
.....	
4ο Εργαλεία και	
Βιβλιοθήκες.....	21
.....	
4.1	
Pytorch.....	21
.....	
4.2 Weights And	
Biases.....	23
.....	
4.3	
Django.....	24
.....	
5ο Μεθοδολογία εκπαίδευσης και	
προγραμματισμός.....	24
.....	
6ο Αποτελέσματα και	
Εκπαίδευση.....	37
.....	
6.1 Αρχική επιλογή	
μοντέλου.....	37
.	
6.2 Transfer Learning -	
Pre-trained.....	38
.....	
6.3 Fine	
Tuning.....	40
.....	
6.4	
Overfitting.....	41
.....	

6.4.1	
Augmentations.....	44
6.4.2 Άλλες	
Αρχιτεκτονικές.....	45
6.4.3	
Early-Stopping.....	46
6.4.4 Τεχνικές	
Κλαδέματος.....	47
6.4.5	
Κανονικοποίηση.....	48
6.4.5.1 L1	
regularization.....	48
6.4.5.2 L2	
regularization.....	48
6.4.5.3	
Dropout.....	49
6.5 Προ-επεξεργασία	
δεδομένων.....	51
6.6 Τελικά	
συμπεράσματα.....	52
7ο Δημιουργία και χρήση Web	
Εφαρμογής.....	53
8ο Συμπεράσματα και Μελλοντικές	
Επεκτάσεις.....	70

Περίληψη

Η επιστήμη της βαθιάς εκμάθησης (Deep Learning) έχει αρχίσει να διαδραματίζει μείζονα ρόλο στην Πληροφορική και έχει οδηγήσει στην δημιουργία συστημάτων αναγνώρισης μεγάλης ακρίβειας για πληθώρα προβλημάτων ξεπερνώντας σε ορισμένες περιπτώσεις τις ανθρώπινες ικανότητες. Η χρήση τους στο τομέα αναγνώριση τέχνης είναι ένα σχετικά καινούργιο ζήτημα το οποίο έχουν προσπαθήσει να λύσουν επιστήμονες της πληροφορικής ακολουθώντας πληθώρα διαφορετικών προσεγγίσεων. Στην παρούσα πτυχιακή εργασία θα γίνει μια προσπάθεια να δημιουργηθεί ένα μοντέλο το οποίο έχει εκπαιδευτεί χρησιμοποιώντας Convolutional Neural Networks (CNNs) που έχει σαν στόχο να επιτρέπει στον χρήστη, να ανεβάσει μία ζωγραφιά και να πληροφορηθεί από ποιους καλλιτέχνες που υπάρχουν στην βάση δεδομένων της εφαρμογής έχει επηρεαστεί ή απλά με ποιους το έργο του έχει τις περισσότερες ομοιότητες.

Summary

The science of Deep Learning has begun to play a fairly large role in Informatics and there are highly accurate identification systems for a wealth of information, some of which have already exceeded human capabilities. Using them for art recognition is a relatively new optic that several researchers have recently tried to address with a myriad of different approaches. In this dissertation an attempt will be made to create a model that has been trained using Convolutional Neural Networks (CNNs) that will aim to allow anyone either an artist or not to create a painting and to learn which artists in the database has influenced them or simply with whom their work has most similarities.

1^o Introduction

Οι άνθρωποι γενικά είναι αρκετά καλοί στην αναγνώριση εικόνων καθώς ανακαλύπτουν με ευκολία τα μοτίβα και κάνοντας αυτόματα τις συνδέσεις που απαιτούνται. Η αναγνώριση έργων ενός καλλιτέχνη είναι μία διαδικασία στην οποία ακόμα και οι άνθρωποι δυσκολεύονται αρκετά. Για την σωστή κατηγοριοποίηση ενός πίνακα χρειάζεται μία ομάδα ανθρώπων για παράδειγμα ιστορικών, καλλιτεχνών και γενικά προσώπων τα οποία έχουν μελετήσει και κατανοήσει μεγάλο αριθμό έργων τέχνης. Ακόμα όμως και μια τέτοια ομάδα ανθρώπων δεν θα μπορούσε με απόλυτη σιγουριά να αποδείξει ότι ένα άγνωστο έργο ανήκει σε έναν καλλιτέχνη.

Αυτό το πρόβλημα μπορεί να προσεγγιστεί από πολλές διαφορετικές οπτικές, γιατί η αναγνώριση καλλιτέχνη δεν εξαρτάται από το περιεχόμενο της εικόνας. Ένας ζωγράφος σχεδόν ποτέ δεν περιορίζεται σε ένα μόνο θέμα, γι' αυτό και το πρόβλημα σταματάει να είναι η αναγνώριση συγκεκριμένων μοτίβων στη θεματική των έργων. Υπάρχουν και πιο πρακτικές διαφορές όπως η τεχνοτροπία , οι διαφορετικές μπογιές, οι καμβάδες, μέχρι και οι ρωγμές που δημιουργούνται στην ξεραμένη μπογιά με την πάροδο του χρόνου. Όλες αυτές οι διαφορές μπορούν να αναλυθούν από ένα μοντέλο αναγνώρισης. Ακόμα και με όλες αυτές τις πληροφορίες είναι πολύ δύσκολο να γίνει μία σίγουρη σύνδεση με τους καλλιτέχνες, γιατί ορισμένοι αλλάζουν τους τρόπους που ζωγραφίζουν μέσα στα χρόνια, αλλάζουν χώρες και αλλάζουν τα υλικά και τον τρόπο που τα χρησιμοποιούσαν.

Σε αυτή την πτυχιακή θα γίνει μία προσπάθεια να εκπαιδευτεί ένα μοντέλο, ώστε να αναγνωρίζει τον “χαρακτήρα” (όσο είναι δυνατόν) ενός καλλιτέχνη και να βρίσκει τις ομοιότητες του έργου το οποίο θα παρέχει ο χρήστης με κάποιους από τους καλλιτέχνες που υπάρχουν στη βάση

δεδομένων. Ταυτόχρονα θα δοκιμαστούν και θα αναλυθούν διάφορα μοντέλα και τεχνικές οι οποίες θεωρούνται state of the art αυτή την περίοδο.

Επειδή πολλοί καλλιτέχνες έχουν επηρεάσει ο ένας τον άλλον και πολλά κινήματα τέχνης αλληλοεπικαλύπτονται, το πρόβλημα θα προσεγγιστεί με K-neighbors λογική. Κατ' επέκταση οι πέντε πιο όμοιοι καλλιτέχνες θα επιστρέφονται ως η ομάδα που έχει τις περισσότερες ομοιότητες με την ζωγραφιά που θα δώσει ως είσοδο ο χρήστης.

Η σχέση της τέχνης με την βαθιά εκμάθηση παρουσιάζει μεγάλο ενδιαφέρον για πολλούς επιστήμονες της πληροφορικής καθώς εμπεριέχει και φιλοσοφικές ερωτήσεις για τον χαρακτήρα και το βάθος κατανόησης που ένα σύγχρονο μοντέλο μπορεί να φέρει εις πέρας. Η τάση αυτή ταυτόχρονα με τη ψηφιοποίηση μεγάλου αριθμού έργων τέχνης έχει δώσει την ευκαιρία να εξελιχθεί η χρήση αλγορίθμων μάθησης στη τέχνη.

Για την εκπαίδευση θα χρησιμοποιηθούν Deep Convolutional Networks προεκπαιδευμένα στο γνωστό dataset [ImageNet](#).

Στα παρακάτω υποκεφάλαια θα αναλυθούν τα μοντέλα που χρησιμοποιήθηκαν, η έννοια της βαθιάς εκμάθησης, της μηχανικής μάθησης και μέχρι ποιο σημείο έχει φτάσει η τεχνολογία, τα εργαλεία που χρησιμοποιούνται, η διαδικασία εκπαίδευσης αναλυτικά με αποτελέσματα και τέλος ένα συμπέρασμα για το μέλλον της σχέσης της βαθιάς εκμάθησης και της τέχνης.

2^ο Θεωρία

2.1 Α.Ι.(Artificial Intelligence)

Από τη βιομηχανική επανάσταση, οι άνθρωποι έχουν αρχίσει να αντικαθιστούν με μηχανές πολλές χειρωνακτικές εργασίες. Αυτό συνέβη εξαιτίας της απότομης εξέλιξης της τεχνολογίας. Η ιδέα της τεχνητής νοημοσύνης εμφανίστηκε πρώτα σε ιστορίες επιστημονικής φαντασίας, ως αντίδραση στην απότομη επιρροή που οι μηχανές άρχισαν να έχουν στην καθημερινότητα των ανθρώπων.

Η τεχνητή νοημοσύνη είναι μια τεχνολογία η οποία αρχικά είχε σαν στόχο να δημιουργηθούν έξυπνοι υπολογιστές, δηλαδή υπολογιστές οι οποίοι θα μπορούν να κάνουν πράγματα που απαιτούν ανθρώπινη ευφυΐα. Όπως είναι λογικό η ιδέα της δημιουργίας τέτοιου επιπέδου

τεχνολογίας ήταν ένα μεγάλο βήμα για την ανθρωπότητα και υπάρχουν ακόμα και σήμερα πολλοί που θεωρούν ότι η συνεχόμενη αντικατάσταση των ανθρώπων με μηχανές μπορεί να έχει επιβλαβείς επιπτώσεις στην ανθρώπινη κοινωνία. Στην εποχή μας η τεχνητή νοημοσύνη έχει ήδη αλλάξει και σίγουρα θα συνεχίσει να αλλάζει την ζωή και των ειδικών και του μέσου ανθρώπου. Η ανάπτυξη αυτή γίνεται ξεκάθαρη τα τελευταία χρόνια, όπου τα κράτη και οι μεγαλύτερες εταιρείες τεχνολογίας είναι σε έναν αγώνα μεταξύ τους για το ποιος θα είναι ο επόμενος που θα πάρει έλεγχο αυτού του επικείμενου κύματος ανάπτυξης. Ενημερωτικά κάποιοι από τους τομείς στους οποίους εφαρμόζεται η τεχνητή νοημοσύνη είναι η βιομηχανοποίηση, υγειονομική περίθαλψη, ασφάλεια, εκπαίδευση και αλυσίδες εφοδιασμού αλλά σε όποια εμφάνιση υπολογιστών και να γίνει αναφορά, είναι ασφαλές να θεωρηθεί βοηθητική η πιθανή ενσωμάτωση κάποιας μορφής έξυπνου προγράμματος.[9]

Σε αυτήν την εποχή που αρχίζουν οι έννοιες και οι τρόποι με τους οποίους εφαρμόζεται η τεχνητή νοημοσύνη να γίνονται πιο διαδεδομένοι υπάρχει πολύ συχνά σύγχυση για τη σχέση της τεχνητής νοημοσύνης, της μηχανικής μάθησης και της βαθιάς εκμάθησης.

Καθώς και στη συνέχεια θα χρησιμοποιηθούν αρκετά συγκεκριμένες έννοιες όπως τα CNNs είναι σημαντικό να εντοπιστούν και να κατανοηθούν οι διαφορές και η ανάπτυξη που έχει υπάρξει στον τομέα.

Αρχικά ο πιο διαδεδομένος και εφ'όλης της ύλης όρος είναι η Τεχνητή Νοημοσύνη(Artificial Intelligence) και εμπεριέχει όλους τους τρόπους με τους οποίους ένας υπολογιστής ή ένα σύστημα μπορεί να εξελιχθεί, ώστε να ανταπεξέρχεται σε όλο και πιο περίπλοκα προβλήματα. Στα παραδείγματα τέτοιων συστημάτων συμπεριλαμβάνονται από προγράμματα που μαθαίνουν να παίζουν σκάκι και άλλα παιχνίδια, έως τεχνολογίες αναγνώρισης προσώπων (Face recognition) που χρησιμοποιούνται από κράτη για την επιτήρηση των πολιτών.

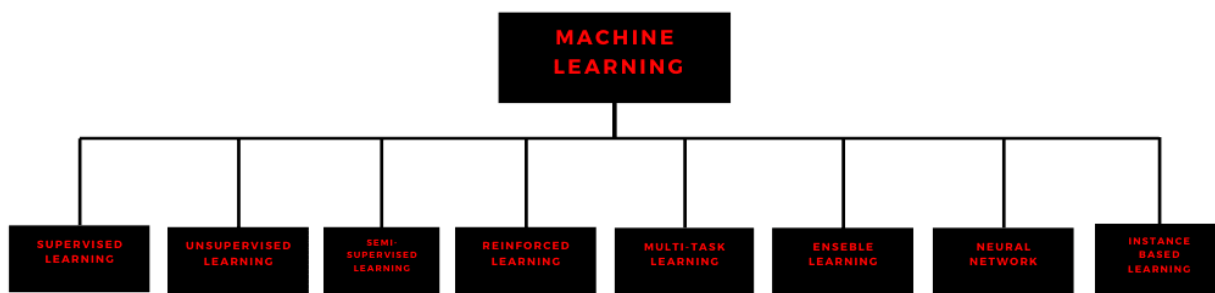
Οι τρεις κατηγορίες που χωρίζεται αυτή η τεχνολογία είναι Narrow AI, artificial general intelligence (AGI), and super-intelligent AI. Αυτές οι τρεις ομάδες με την σειρά που αναφέρθηκαν αντιστοιχούν σε σταδιακά εξυπνότερα και καλύτερα συστήματα. Αρχικά το Narrow AI το οποίο έχει δημιουργηθεί για έναν συγκεκριμένο σκοπό, στη συνέχεια τα AGI τα οποία θεωρείται ότι έχουν ικανότητες πανομοιότυπες με ανθρώπους και τέλος τα super-intelligent AI τα οποία έχουν ξεπεράσει τις ικανότητες των ανθρώπων.[2]

2.2 Μηχανική Μάθηση

Κατεβαίνοντας το γενεαλογικό δέντρο της τεχνολογίας της τεχνητής νοημοσύνης σε ένα από τα κλαδιά ανήκει το Machine Learning(ML). Στο επίκεντρο όλων των προγραμμάτων που έχουν σαν βάση το ML υπάρχει η ιδέα ότι το πρόγραμμα πρέπει να έχει την ικανότητα να του

διοχετεύονται πληροφορίες και αυτό με την σειρά του να “μαθαίνει” μόνο του. Αυτή είναι και η λογική η οποία ταιριάζει στη πιο γενική ιδέα της τεχνητής νοημοσύνης καθώς και το πιο υποσχόμενο παρακλάδι αυτής της τεχνολογίας. Ειδικά στην εποχή των big data, που η ανθρωπότητα πλέον παράγει τόσο μεγάλη ποσότητα πληροφορίας, πλέον υφίσταται αδύνατον για το ανθρώπινο δυναμικό να εξετάσει διεξοδικά το μεγαλύτερο ποσοστό των διαθέσιμων πληροφοριών. Μέσα σε αυτό το περιβάλλον, πολλές χρήσιμες πληροφορίες μένουν στην αφάνεια, ενώ ταυτόχρονα πολλές επιβλαβείς ή ανούσιες πληροφορίες κυριαρχούν συχνά τις οθόνες των χρηστών. Βλέποντας πλέον την έκταση των πληροφοριών που θέλουν να χρησιμοποιήσουν οι σύγχρονοι άνθρωποι, το ML φαίνεται να έχει έρθει για να μείνει στις ζωές όλων. Αυτό συμβαίνει είτε συνειδητά με την χρήση προγραμμάτων, είτε ασυναίσθητα μέσω ιστοσελίδων και εφαρμογών που το χρησιμοποιούν για να προωθήσουν σε κάθε χρήστη περισσότερο υλικό που θα τους κρατήσει αφοσιωμένους, είτε από κάποιο κράτος που θέλει να πάρει πληροφορίες για τον πληθυσμό του.

Το ML μπορεί να χωριστεί στις παρακάτω κατηγορίες, για μια πιο αναλυτική άποψη βλ. στο [5].



Σχήμα 1: Οι κατηγορίες του ML

Supervised Learning, οι αλγόριθμοι οι οποίοι χρειάζονται εξωτερική βοήθεια για την σωστή λειτουργία τους. Αυτό σημαίνει ότι ο αλγόριθμος έχει και τα δεδομένα εισόδου και τα επιθυμητά αποτελέσματα. Τα δεδομένα χωρίζονται σε εκπαίδευσης (train), επιβεβαίωσης (validation) και ελέγχου (test). Σκοπός τους είναι να μπορούν να γενικοποιήσουν την γνώση και να αποδώσουν σωστά αποτελέσματα όταν τους παρέχονται καινούργια δεδομένα χωρίς το αποτέλεσμα. Οι επιβλεπόμενοι αλγόριθμοι διαπρέπουν σε προβλήματα στα οποία υπάρχει μικρός αριθμός διαθέσιμων πληροφοριών. Μερικά παραδείγματα (Decision Tree, Naive Bayes)

Unsupervised Learning, όπως δηλώνει και το όνομα οι αλγόριθμοι αυτοί εκπαιδεύουν τον εαυτό τους, δηλαδή τα δεδομένα εισόδου δεν συνοδεύονται από τα αποτελέσματα. Τους δίνονται τα δεδομένα και από μόνοι τους εξάγουν χαρακτηριστικά δημιουργώντας ομάδες, μέσα στις οποίες στη συνέχεια προσπαθούν να κατηγοριοποιήσουν τα καινούργια δεδομένα. Η χρήση τέτοιων προγραμμάτων προτείνεται να συνοδεύεται από μεγάλο αριθμό δεδομένων για είσοδο. Μερικά παραδείγματα (K-means Clustering, Principal Component Analysis)

Semi Supervised Learning, ένας συνδυασμός των προηγούμενων δύο κατηγοριών, δείχνει την αξία του σε περιπτώσεις που υπάρχει μεγάλος αριθμός δεδομένων αλλά η διαδικασία του να κατηγοριοποιηθούν είναι δύσκολη, καθώς εκπαιδεύεται σε ένα συγκεκριμένο αριθμό δεδομένων που είναι labeled και στη συνέχεια στα δεδομένα που δεν είναι. Μερικά παραδείγματα (Transductive SVM, Self Learning)

Reinforcement Learning, μια από τις 3 βασικές δομές των ML συστημάτων μαζί με το Supervised Learning και το Unsupervised Learning. Βασίζεται στην ιδέα του να επιβραβεύεται το λογισμικό για τις σωστές ενέργειες και να τιμωρείται αντίστοιχα για τις λάθος, φτάνοντας σε ένα επιθυμητό στάδιο μέσω της διαδικασίας πειραματισμού και σφάλματος.

Neural Network, βασίζονται στο νευρωνικό σύστημα των ανθρώπων προσπαθώντας να ανακαλύψουν τις σχέσεις που πιθανόν υπάρχουν ανάμεσα στα δεδομένα που παρέχονται. Όπως το ανθρώπινο νευρωνικό δίκτυο αποτελείται από νευρώνες οι οποίοι μεταφράζουν οποιαδήποτε πληροφορία με μεγάλη ελαστικότητα, τα νευρωνικά δίκτυα καταφέρνουν κάτι παρόμοιο χρησιμοποιώντας 3 επίπεδα. Το **επίπεδο εισόδου** στο οποίο παρέχονται οι πληροφορίες, το ενδιάμεσο ή **κρυφό επίπεδο** (hidden layer) στο οποίο τα βάρη τα οποία έχουν δημιουργηθεί για κάθε ένα από τα δεδομένα επεξεργάζονται και το **επίπεδο εξόδου** στο οποίο θα γίνει η ταξινόμηση. Υπάρχουν πολλοί τρόποι με τους οποίους μπορούν να χρησιμοποιηθούν τα NNs σε συνδυασμό με τις προαναφερόμενες μεθόδους του ML.

2.3 Βαθιά Εκμάθηση

Τέλος θα αναφερθεί η έννοια της βαθιάς εκμάθησης (DL), που αποτελεί υποσύνολο της μηχανικής μάθησης και συμπεριλαμβάνει την χρήση Νευρωνικών Δικτύων (Neural Networks). Οι υπολογιστές και τα όρια τους, ή καλύτερα η περιέργεια για το αν υπάρχουν όρια όσο αναφορά τι μπορεί ένας υπολογιστής να καταφέρει ήταν μία απορία που η ανθρωπότητα είχε πριν καν δημιουργηθεί ο πρώτος υπολογιστής. Μπορεί όταν έκανε την εμφάνιση της η τεχνολογία για πρώτη φορά να χρησιμοποιήθηκε μόνο για διαδικασίες οι οποίες ήταν εύκολες για έναν υπολογιστή, όπως μαθηματικές πράξεις, αλλά τα τελευταία χρόνια έχει ανακαλυφθεί το πραγματικό εύρος ικανοτήτων που η βαθιά εκμάθηση μπορεί να προσφέρει. [6]

Το πιο δύσκολο και το πιο μεγάλο εμπόδιο για έναν υπολογιστή και την τεχνολογία της τεχνητής νοημοσύνης είναι το να κατανοήσει την πληθώρα των αυτόματων διαδικασιών που ένας ανθρώπινος εγκέφαλος πραγματοποιεί καθημερινά, σε μυριάδα καταστάσεων, αλλά δεν μπορεί

να αποτυπωθεί με μαθηματική ακρίβεια. Κάθε λέξη και κάθε πρόσωπο που ένας άνθρωπος αναγνωρίζει χρησιμοποιεί μια αυτόματη διαδικασία η οποία είναι πολύ δύσκολο να μεταφερθεί στον υπολογιστή. Η λύση σε αυτό το πρόβλημα ήρθε με την μορφή της βαθιάς εκμάθησης, χρησιμοποιώντας μια ιεραρχία από έννοιες διασυνδεδεμένες η μία με την άλλη πηγαίνοντας όλο και πιο “βαθιά” από τις πιο περίπλοκες στις πιο απλές, επιτρέποντας στον υπολογιστή να καταλάβει έως ένα σημείο τους κανόνες του κόσμου που του ζητείται. Μεγάλο μέρος της αξίας της βαθιάς εκμάθησης είναι η ικανότητα να παίρνει μεγάλο αριθμό πληροφοριών και μετά από την απαραίτητη εκπαίδευση αυτόματα να κατηγοριοποιεί ορισμένα χαρακτηριστικά, ελαφρύνοντας αισθητά το φορτίο από το ανθρώπινο δυναμικό. Αξίζει να σημειωθεί ότι παρόλο που η γενική ιδέα είναι ότι τα DL συστήματα είναι το μέλλον υπάρχουν διαφορούμενες απόψεις που θεωρούν ότι έχουν υπερεκτιμηθεί οι αξίες και οι δυνατότητες τους.

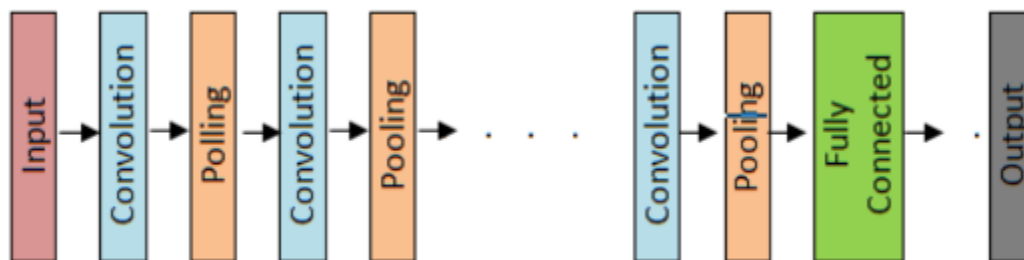
2.4 CNNs

Τα CNNs ή αλλιώς Convolutional Neural Networks αποτελούν την τελευταία λέξη της τεχνολογίας όσον αφορά την ταξινόμηση εικόνων. Η όραση του υπολογιστή έχει πολλές κατηγορίες στις οποίες κάθε έρευνα μπορεί να ειδικευτεί. Η ταξινόμηση των εικόνων είναι η πιο σημαντική από αυτές, καθώς οι υπόλοιπες μπορούν να στηριχθούν πάνω σε αυτή για να λυθεί το συγκεκριμένο πρόβλημα. Στο παρελθόν η μόνη δυνατή λύση ήταν η δημιουργία συγκεκριμένων χειροποίητων χαρακτηριστικών, κάτι το οποίο είχε πολλούς περιορισμούς και ήταν απίστευτα χρονοβόρο. Με την εξέλιξη της επιστήμης, την επανεμφάνιση των Neural Networks ή αλλιώς βαθιάς εκμάθησης σε συνδυασμό με την χρήση του αλγόριθμου οπίσθιου πολλαπλασιασμού (back-propagation) οι συνδέσεις σε μεγάλο αριθμό δεδομένων έγιναν πολύ πιο εφικτές. Ανάμεσα στα βαθιά νευρωνικά δίκτυα, τα CNNs έχουν τα καλύτερα αποτελέσματα όπως φαίνεται από τις έρευνες που έχουν πραγματοποιηθεί.

Ξεκινώντας από την δημιουργία τους, τα CNNs παίρνουν την μορφή τους αντιγράφοντας για ακόμα μια φορά την φύση και πιο συγκεκριμένα την όραση, δηλαδή νευρωνικά δίκτυα με πολλαπλά επίπεδα τα οποία μπορούν να αναγνωρίσουν, μόνο από τα εικονοστοιχεία, οπτικά μοτίβα χωρίς ανθρώπινη παρέμβαση. Η πρώτη λειτουργική μορφή είναι το LeNet-5. Ακόμα και στην αρχική τους μορφή τα CNNs είχαν πολλά πλεονεκτήματα συγκριτικά με άλλα νευρωνικά

δίκτυα, όπως τον μικρότερο αριθμό παραμέτρων και συνδέσεων, πράγμα που έκανε πιο εύκολη την εκπαίδευση. Ο περιορισμός που υπήρχε όμως όσον αφορά την κατηγοριοποίηση εικόνων που μπορεί να χαρακτηριστεί ως ένα αρκετά πολύπλοκο πρόβλημα, παρέμενε, γιατί υπήρχε έλλειψη δεδομένων, υπολογιστικής δύναμης και τρόπων για καλύτερη κανονικοποίηση (regularization). Στην τωρινή εποχή όλα αυτά τα προβλήματα έχουν λυθεί και το 2012 το AlexNet χρησιμοποιώντας το ImageNet είχε ιδιαίτερα καλά αποτελέσματα οδηγώντας στην δημιουργία όλο και πιο δυνατών CNNs όπως το ResNet. [10]

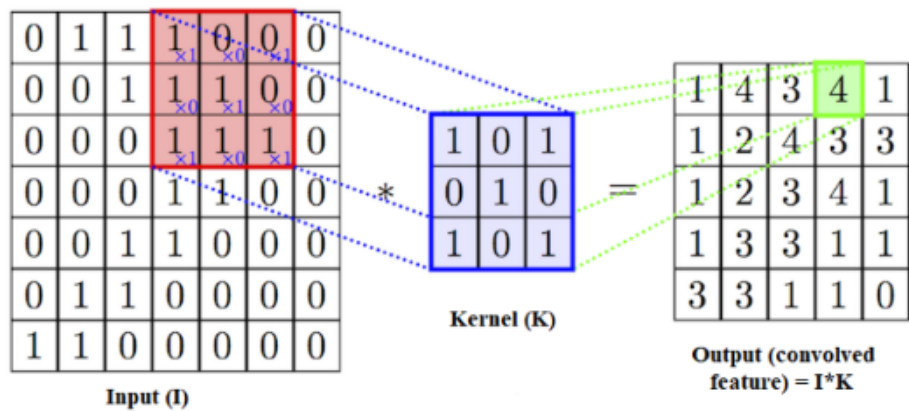
Τα CNNs αποτελούνται από ένα ή πολλά επίπεδα συστροφών (convolutions) και υποδειγματοληψίας (sub-sampling), ένα ή περισσότερα επίπεδα πλήρως συνδεδεμένων στρωμάτων και τέλος ένα επίπεδο εξόδου.



Εικόνα 1: Δομικό στοιχείο για ένα τυπικό CNN

Πηγή: [10]

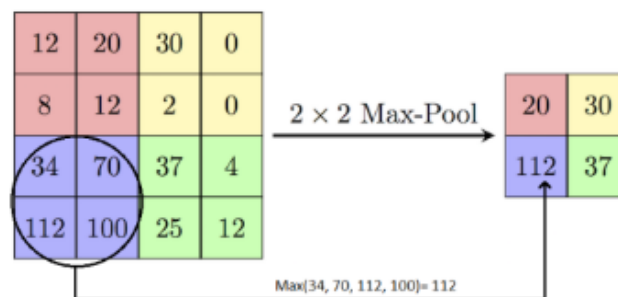
Το συνελκτικό στρώμα (Convolutional layer) είναι το κεντρικό τμήμα ενός CNN, επειδή οι εικόνες είναι γενικά σταθερές, ένα μέρος της εικόνας είναι ίδιο με οποιοδήποτε άλλο. Με αυτή τη λογική ένα χαρακτηριστικό που μαθαίνεται σε ένα σημείο μπορεί να ταιριάζει σε ένα άλλο μοτίβο σε μια άλλη περιοχή της εικόνας. Έτσι ένα συνελκτικό στρώμα παίρνει ένα μικρό κομμάτι της εικόνας, προσθέτει κάθε στοιχείο με τους γείτονες του και τα σταθμίζει με το kernel. Οι τιμές κάθε δεδομένου εικονοστοιχείου στην εικόνα εξόδου καθορίζονται πολλαπλασιάζοντας το κάθε kernel με τις αντίστοιχες τιμές κάθε εικονοστοιχείου της εικόνας εισόδου.



Εικόνα 2: Συνελικτικό στρώμα

Πηγή: [10]

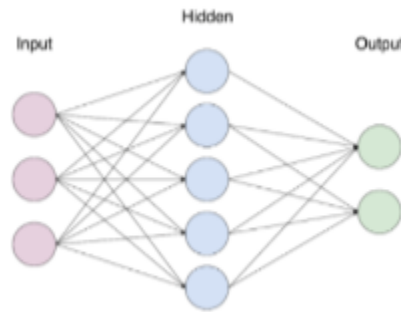
Υποδειγματοληψία (Sub-sampling) ή Επίπεδο συγκέντρωσης (Pooling Layer), ο σκοπός του οποίου είναι να “μικρύνει” την εικόνα παίρνοντας ένα μικρό μέρος της εικόνας εξόδου από το συνελικτικό στρώμα και “συγκεντρώνοντας” το σε μία τιμή. Υπάρχουν πολλές τεχνικές για pooling όπως max pooling, mean pooling, average pooling κλπ.



Εικόνα 3: Λειτουργία Max Pooling

Πηγή[10]

Τέλος, αν παραληφθεί το επίπεδο εξόδου το οποίο εξαρτάται από ανάγκες και τον στόχο του αλγορίθμου, απομένουν τα Πλήρως Συνδεδεμένα Δίκτυα (Fully-connected Layers), τα οποία παίρνουν τις πληροφορίες από όλους τους “νευρώνες” των προηγούμενων επιπέδων και τα επεξεργάζονται με τον νευρώνα του επιπέδου για να φτάσουν στο τελικό αποτέλεσμα.



Εικόνα 4: Πλήρως συνδεδεμένο επίπεδο

Πηγή: [10]

2.5 Transfer Learning

Ένα από τα πιο σημαντικά στοιχεία για τον κόσμο της μηχανικής μάθησης και της βαθιάς εκμάθησης είναι οι πληροφορίες που παρέχονται στο πρόγραμμα εκπαίδευσης και στη συνέχεια η δημιουργία συνδέσεων μεταξύ “έννοιών”. Υπάρχουν όμως ορισμένες καταστάσεις που ακόμα και στην εποχή των big data, είτε δεν υπάρχουν αρκετές πληροφορίες, είτε δεν είναι πολύ ακριβείς, είτε είναι πολύ χρονοβόρα η διαδικασία συλλογής και κατηγοριοποίησης τους. Οι καταστάσεις αυτές μπορούν να αντιμετωπιστούν με την μεταφορά μάθησης (transfer learning), μεταφέροντας την γνώση από ένα προηγούμενο εκπαιδευμένο μοντέλο στο νέο σύστημα αναγνώρισης στο οποίο δεν υπάρχει επαρκής αριθμός δεδομένων. Μεγάλες διαφορές στο πώς είναι διαμορφωμένα τα προαναφερόμενα δεδομένα στο testing και στο πώς είναι διαμορφωμένα τα καινούργια δεδομένα που θέλουμε να προσθέσουμε στην διαδικασία της εκπαίδευσης, μπορούν να εισφέρουν στον υποβιβασμό των αποτελεσμάτων.

Η λογική του Transfer Learning(TL) έχει τις ρίζες της στην πραγματικότητα, κάτι το οποίο για έναν άνθρωπο μπορεί να γίνει εύκολα αντιληπτό. Ο πιο διαδεδομένος τρόπος εκπαίδευσης στις ανθρώπινες κοινωνίες πάντα ήταν η ανταλλαγή πληροφοριών και η μεταλαμπάδευση γνώσεων από τους πιο έμπειρους στους αρχάριους. Ακολουθώντας το μοντέλο της φύσης το TL έχει σαν στόχο να μεταφέρει γνώση από ένα ήδη εκπαιδευμένο μοντέλο σε ένα το οποίο εκπαιδεύεται. Για να πετύχει όμως η μεταξύ τους επικοινωνία και να επιτευχθεί το καλύτερο αποτέλεσμα, οι “έννοιες” και τα δεδομένα που αυτά τα δύο πραγματεύονται πρέπει να έχουν πολλές παρόμοιες

ιδιότητες. Για παράδειγμα, ώστε να γίνει πιο κατανοητή η ιδέα των ομοιοτήτων, ένα πρόγραμμα το οποίο έχει εκπαιδευτεί στο να αναγνωρίζει πρόσωπα θα μπορούσε να χρησιμοποιηθεί στην αναγνώριση ανθρώπων. Και τα δύο είναι προβλήματα όρασης και η “έννοια” του ανθρώπου εμπεριέχει την “έννοια” του προσώπου, οπότε υπάρχει μια σύνδεση μέσω της οποίας θα μπορούσε το καινούργιο πρόγραμμα να βοηθηθεί. Ένα άλλο παράδειγμα είναι ένας αθλητής καλαθοσφαίρισης και ένας μέσος άνθρωπος οι οποίοι προσπαθούν να μάθουν και οι δύο ποδόσφαιρο. Στη διαδικασία αυτή θα ήταν πολύ πιο ήπια η εξέλιξη και η προπόνηση για τον αθλητή. Ο λόγος είναι ότι πριν αρχίσει η διαδικασία μάθησης ποδοσφαίρου, ο αθλητής θα είχε ήδη προπονήσει το σώμα του και θα είχε ήδη αναπτύξει το ομαδικό πνεύμα που χρειάζεται για τις ανάγκες της προηγούμενης δουλειάς του, οπότε οι πληροφορίες για την μάθηση του ποδοσφαίρου θα περνούσαν πιο εύκολα σε αυτόν.

Για την σωστή κατανόηση και ανάλυση της τεχνολογίας του TL δίνονται οι παρακάτω ορισμοί.

Ορισμός 1: Ένα πεδίο ορισμού (Domain) το οποίο αποτελείται από δύο υποκατηγορίες, ένα πεδίο από χαρακτηριστικά το οποίο θα ονομαστεί X και μία οριακή κατανομή $P(X)$. Ισχύει $D = \{X, P(X)\}$, το X δηλώνει ένα πεδίο χαρακτηριστικών $X = \{x | x_i \in X, i=1, \dots, n\}$.

Ορισμός 2: Ένα έργο (task) T αποτελείται από ένα πεδίο ταμπελών Y και μία συνάρτηση απόφασης f , όπου $T = \{Y, f(\cdot)\}$. Η συνάρτηση απόφασης θα μαθευτεί από τα δεδομένα.

Έχοντας πλέον έναν ορισμό για το TL, η διαδικασία μάθησης μπορεί να διατυπωθεί ως εξής: έστω ένα συγκεκριμένο πεδίο ορισμού D_s με ένα αντίστοιχο έργο (task) T_s ως πηγή και ένα πεδίο ορισμού του στόχου D_t με ένα αντίστοιχο έργο (task) T_t ως ο στόχος, η διαδικασία βελτίωσης της συνάρτησης απόφασης $F_t(\cdot)$ χρησιμοποιώντας τις σχετικές πληροφορίες από τις πληροφορίες των πηγών D_s και T_s . Ισχύει $D_s \neq D_t$ και $T_s \neq T_t$.

Από την εμφάνιση της τεχνολογίας του TL χρησιμοποιούνται δύο μεγάλες κατηγορίες, Homogeneous transfer learning και Heterogeneous transfer learning. Η διαφορά των δύο προέρχεται από τη διαφοροποίηση των πηγών (source) και των στόχων (target), καθώς στην ομοιογενή μεταφορά μάθησης η πηγή και ο στόχος συνυπάρχουν στον πεδίο χαρακτηριστικών, ενώ αντίθετα στην ετερογενή μεταφορά μάθησης εκπροσωπούνται από διαφορετικά πεδία χαρακτηριστικών.

Πιο αναλυτικά, αφού $D_s = \{X_s, P(X_s)\}$ και $D_t = \{X_t, P(X_t)\}$ η ετερογενή μάθηση χαρακτηρίζεται ως η περίπτωση που $X_s \neq X_t$ και αντιθέτως η περίπτωση που $X_s = X_t$ χαρακτηρίζεται ως ομοιογενή μάθηση.

2.6 ImagiNet

Συζητώντας για προβλήματα όρασης υπολογιστή, θα ήταν ανήκουστο να μην αναφερθεί το ImageNet. Η ιδέα του ImageNet πήρε μορφή το 2006 από την Fei-Fei Li η οποία αποφάσισε να επεκτείνει την επιλογή από δεδομένα που υπάρχουν για την εκπαίδευση των μοντέλων σε μία εποχή που το μεγαλύτερο ποσοστό της προσπάθειας των ερευνητών είχε επικεντρωθεί στην ανάπτυξη μοντέλων και αλγορίθμων. Το 2007 μετά την συνάντηση της με τον Christiane Fellbaum, ενός από τους δημιουργούς του WordNet, το ImageNet πήρε την τελική του μορφή. Με την τεράστια ροή οπτικών δεδομένων είτε σε μορφή βίντεο είτε κανονικών εικόνων, έχει δώσει την ευκαιρία να αναπτυχθούν αλγόριθμοι και μοντέλα πολύ πιο εξελιγμένα από ότι ήταν δυνατό στο παρελθόν. Για να γίνει όμως σωστή χρήση όλης αυτής της πληροφορίας από ένα σύστημα αναγνώρισης, είναι απαραίτητη η κατηγοριοποίηση αυτών των εικόνων. Το ImageNet είναι μία βάση δεδομένων που δημιουργήθηκε με ακριβώς αυτόν τον σκοπό, την συλλογή δεδομένων που μπορούν να χρησιμοποιηθούν για την ανάπτυξη του τομέα όσο πιο αποτελεσματικά και γρήγορα είναι δυνατόν. Είναι βασισμένο στην ιεραρχία του WordNet, το οποίο χρησιμοποιεί ορισμένες λέξεις ή φράσεις για να κατηγοριοποιήσει έννοιες, ονόματι “synonym set” η “synset”. Στην αρχή του ImageNet ο στόχος ήταν να παραχθούν 500-1000 εικόνες για κάθε synset με τη βοήθεια ανθρώπινου δυναμικού. Κάποιες από τις κατηγορίες είναι θηλαστικά, ψάρια, οχήματα και μουσικά όργανα. Παρόλο που ο σκοπός που δημιουργήθηκε ήταν από την αρχή η χρήση και διεύρυνση αυτής της βάσης δεδομένων, μετά την δημιουργία του έχει φέρει τεράστια κύματα προόδου στον κόσμο της όρασης υπολογιστών (computer vision). Πλέον αποτελείται από 14 εκατομμύρια εικόνες και 21 χιλιάδες WordNet synsets. [7]

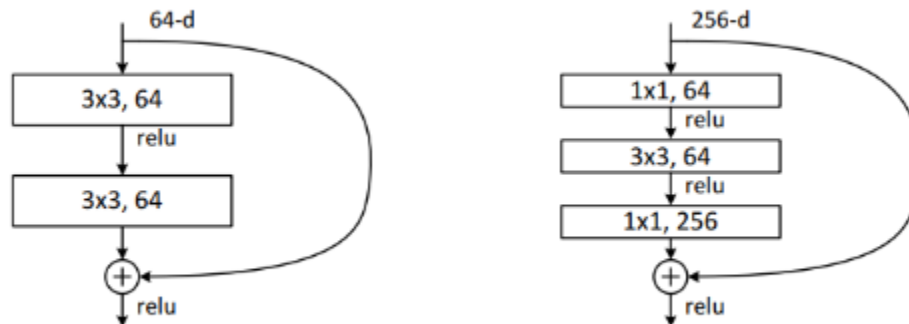
3^ο Δημοφιλείς αρχιτεκτονικές

Όσον αφορά τα προβλήματα όρασης υπολογιστή η διαδεδομένη λύση είναι η αύξηση των επιπέδων στα δίκτυα των μοντέλων, ώστε τα επιπλέον επίπεδα να μπορούν να βοηθήσουν σε πιο περίπλοκα προβλήματα, καθώς κάθε επιπλέον επίπεδο μπορεί να αναλάβει διαφορετικά χαρακτηριστικά. Αλλά όσο αυξάνεται το βάθος των επιπέδων τόσο πιο δύσκολη γίνεται η εκπαίδευση και σε ορισμένες περιπτώσεις το μοντέλο οδηγείται σε μείωση της ακρίβειας. Αυτό συμβαίνει λόγω της αναπόφευκτης υποβάθμισης (degradation), καθώς με την αύξηση των συνδέσεων υπάρχει κορεσμός και σταδιακή μείωση της ακρίβειας σε κάθε επίπεδο, με καταστροφικές συνέπειες στην εκπαίδευση και στις δοκιμές. Η υποβάθμιση δεν πρέπει να μπερδεύεται με την έννοια του overfitting και μπορεί να δημιουργηθεί από πληθώρα λόγων, με το πιο πιθανό να είναι το πρόβλημα εξαφάνισης των gradients. Τα gradients ορίζονται ως την παράγωγο μιας συνάρτησης που έχει περισσότερες από μία μεταβλητές εισόδου και χρησιμοποιούνται γενικά στον διανυσματικό λογισμό και πιο συγκεκριμένα στα CNNs στην υλοποίηση του back-propagation.

3.1 ResNet50

Τα Residual Neural Network (ResNet50) είναι τεχνητά νευρωνικά δίκτυα τα οποία δημιουργήθηκαν ως μια λύση σε αυτό το μεγάλο πρόβλημα μεγαλύτερων επιπέδων των Deep neural networks, δίνοντας επιτέλους την ευκαιρία να αξιοποιηθούν Νευρωνικά δίκτυα με περισσότερα επίπεδα. [6] Στο επίκεντρο αυτής της βελτίωσης είναι η λογική των Skip Connections, η οποία περιληπτικά επιτρέπει στα gradients μια γρήγορη μεταφορά, προσθέτοντας τα αποτελέσματα των προηγούμενων επιπέδων στα αποτελέσματα των στοιβαγμένων επιπέδων, αντιμετωπίζοντας το πρόβλημα της εξαφάνισης των gradients συνδυάζοντας επίσης και την χρήση των Identity functions ή Linear Activation Functions, τα οποία μειώνουν ακόμα περισσότερο την πιθανότητα λάθους.

Για την εκπαίδευση στην πτυχιακή αυτή εργασία χρησιμοποιείται το μοντέλο ResNet50 το οποίο κατατάσσεται στα Deep residual networks τα οποία με την σειρά τους ανήκουν στην οικογένεια των CNNs και αποτελείται από 50 επίπεδα. Η μεγάλη διαφορά των μοντέλων μετά το ResNet50 από τα προ του ResNet μοντέλα είναι η αύξηση των επιπέδων που υπάρχουν ανάμεσα σε κάθε δομικό στοιχείο συμφόρησης (congestion), αυξάνοντας περαιτέρω την ακρίβεια και μειώνοντας παράλληλα τον χρόνο που απαιτείται για εκπαίδευση. Υπάρχουν πολλά είδη από ResNets τα οποία χρησιμοποιούνται για τις ανάγκες μεγέθους που μπορεί να έχει ένας ερευνητής, από το ResNet-18, που χρησιμοποιείται και στα παρακάτω πειράματα, μέχρι το ResNet-200.



Εικόνα 5: Μία βαθύτερη Residual συνάρτηση F για το ImageNet. Αριστερά: ένα δομικό στοιχείο (56*56 χάρτης χαρακτηριστικών) για ResNet34. Δεξιά: ένα δομικό στοιχείο “bottleneck” για ResNet50/101/152

Πηγή:[6]

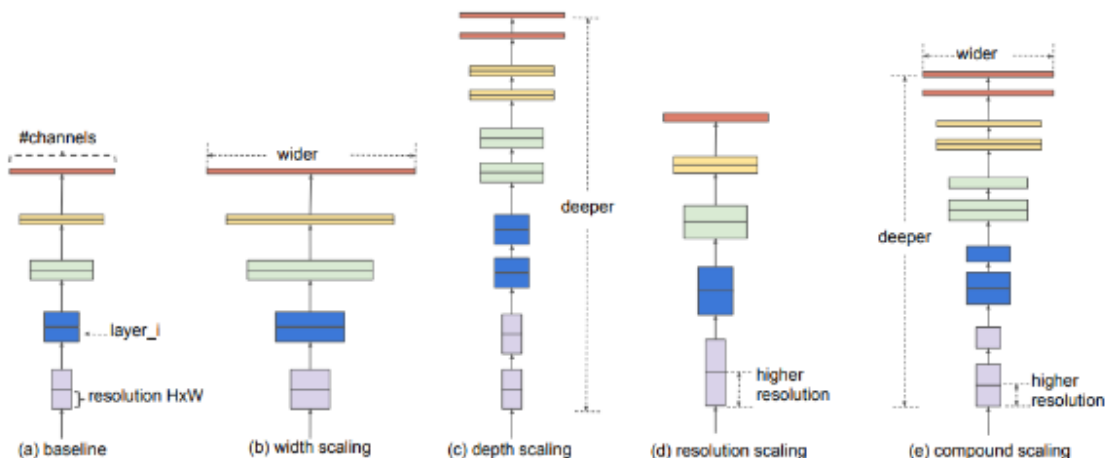
3.2 Efficient Net

Καθώς τα συνελκτικά νευρωνικά δίκτυα (Convolutional Networks) συνήθως αναπτύσσονται με σταθερούς πόρους και στη συνέχεια αναπτύσσονται περαιτέρω αν βρεθούν περισσότεροι πόροι, το EfficientNet είναι ένας τρόπος να υπάρχει έλεγχος των παραμέτρων όπως το βάθος, το πλάτος και η ανάλυση, με αποτέλεσμα να δημιουργείται μια αρχιτεκτονική η οποία μπορεί να φτάσει σε καλύτερες αποδόσεις[12].

Τα EfficientNets είναι μία πολύ καλή επιλογή καθώς φαίνεται να καταφέρνουν υψηλές αποδόσεις γρηγορότερα και με μικρότερο δίκτυο από τα ConvNets που υπήρχαν. Κάτι το οποίο ταιριάζει με την παρούσα πτυχιακή καθώς υπάρχει περιορισμός στη υπολογιστική δύναμη.

Οι πιο συχνές λύσεις για την βελτίωση ενός ConvNet είναι η αύξηση του βάθους και του πλάτους και πιο σπάνια η βελτίωση της ανάλυσης των εικόνων. Η αλλαγή αυτών των παραμέτρων μπορεί να γίνει συνδυαστικά ή ατομικά, η διαδικασία αυτή όμως παίρνει πολλές προσπάθειες και μικρές διορθώσεις μέχρι να βρεθεί η κατάλληλη ισορροπία και πολύ συχνά οδηγεί σε μοντέλα χειρότερης ακρίβειας.

Ο τρόπος που καταφέρνουν τα EffNets να φτάνουν σε καλύτερα αποτελέσματα πιο γρήγορα είναι χρησιμοποιώντας μία μέθοδο σύνθετης κλιμάκωσης (compound scaling method), με στόχο να κλιμακώνεται το δίκτυο με έναν πιο ομοιόμορφο τρόπο, μια αρκετά απλή αλλά αποδοτική λύση.



Εικόνα 6: Κλιμάκωση μοντέλου (a) ένα παράδειγμα δικτύου βάσης. (b)-(d) είναι μια συμβατική κλιμάκωση που μεγαλώνει μόνο μία διάσταση του δικτύου, είτε πλάτος, είτε βάθος, είτε ανάλυση δικτύου. (e) μία

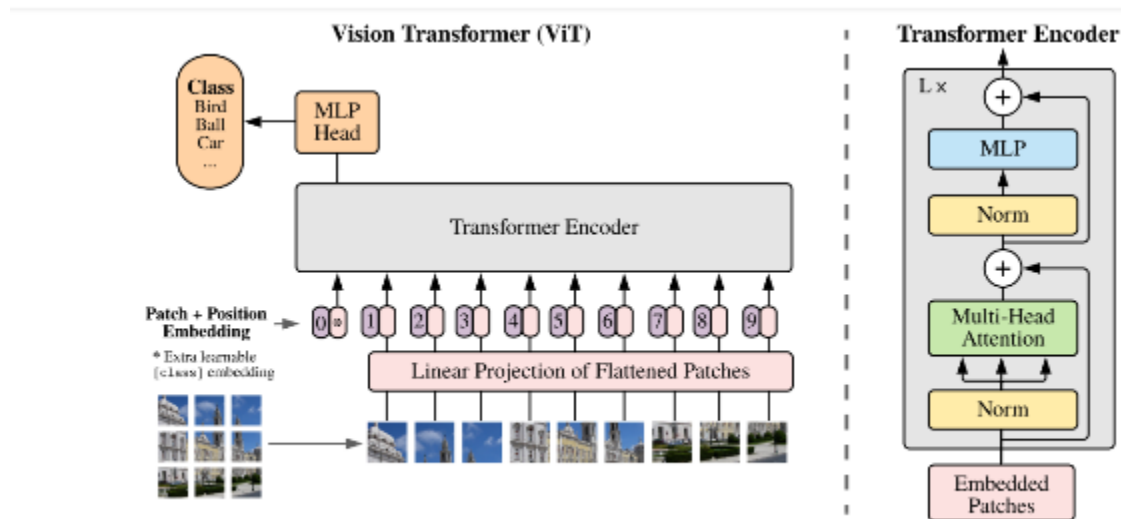
συνθετική μέθοδος κλιμάκωσης που κλιμακώνει ομοιόμορφα και τις τρεις διαστάσεις με σταθερή αναλογία.

Πηγή: [12]

3.3 Vision Transformer

Οι μετασχηματιστές ή αλλιώς transformers χρησιμοποιούνται κυρίως σε προβλήματα επεξεργασίας γλώσσας. Όσον αφορά τα προβλήματα όρασης οι μετασχηματιστές χρησιμοποιούνται συνήθως είτε σε συνδυασμό με κάποιο CNN, είτε για την μετατροπή ορισμένων στοιχείων χωρίς να επηρεάζουν όμως την δομή τους. Το ViT είναι απόδειξη ότι αυτή η έντονη σχέση που έχουν οι μετασχηματιστές και τα CNNs δεν σημαίνει ότι δεν μπορούν να χρησιμοποιηθούν ξεχωριστά.

Χρησιμοποιώντας τις μεγάλες βάσεις δεδομένων που υπάρχουν πλέον, όπως το ImageNet για την προ-εκπαίδευση, ένα σύστημα το οποίο χρησιμοποιεί την αρχιτεκτονική των μετασχηματιστών εφαρμόζοντας τους σε μέρη της εικόνας μπορεί να οδηγήσει σε αποτελέσματα παρόμοια ή και καλύτερα από τα υπερσύγχρονα CNNs.



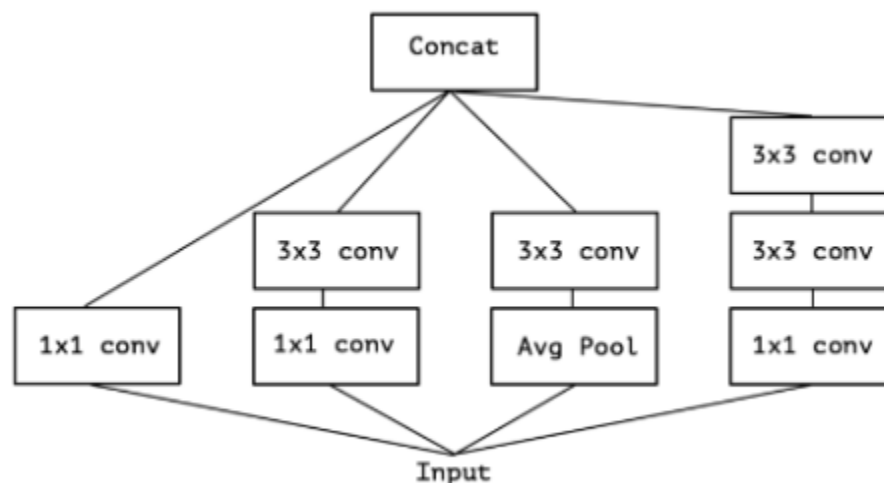
Εικόνα 7: Επισκόπηση μοντέλου. Χωρίζεται μια εικόνα σε τεμάχια σταθερού μεγέθους, ενσωματώνεται γραμμικά κάθε ένα από αυτά και πραγματοποιείται προσθήκη ενσωμάτωσης θέσεων και τροφοδότηση της ακολουθίας διανυσμάτων που προκύπτει σε ένα τυπικό μετασχηματιστή κωδικοποίησης. Για να πραγματοποιηθεί η ταξινόμηση, χρησιμοποιείται η τυπική προσέγγιση προσθήκη ενός επιπλέον εκπαιδευτικού "Classification Token" στην ακολουθία.

Πηγή: [13]

Πιο συγκεκριμένα η εικόνα χωρίζεται σε τμήματα και παρέχονται ακολουθίες γραμμικών ενσωματώσεων για κάθε τμήμα, δηλαδή μια προβολή των δεδομένων σε μικρότερη διάσταση διατηρώντας την “απόσταση” από τα γειτονικά τμήματα. Αυτές οι πληροφορίες παρέχονται σαν είσοδος σε έναν μετασχηματιστή και ακολουθεί εκπαιδευσιμε εποπτευόμενο τρόπο. Όταν εφαρμόζεται σε μέτριου μεγέθους βάσεις δεδομένων τα αποτελέσματα συνήθως είναι λίγο κατώτερα από τις αρχιτεκτονικές που έχουν σαν βάση τα ResNet. Οι μετασχηματιστές λόγω του χαρακτήρα της λειτουργίας τους μπορεί να χάσουν μερικά χαρακτηριστικά σε σχέση με τα αντίστοιχα CNNs αν δεν υπάρχει ικανοποιητική ποσότητα δεδομένων, καθώς πολύ πιθανό να μην μπορέσουν να γενικοποιήσουν τα δεδομένα ικανοποιητικά.

3.4 Xception

Το θεμελιώδες δομικό στοιχείο των μοντέλων τύπου Inception είναι η μονάδα Inception, της οποίας υπάρχουν πολλές διαφορετικές εκδόσεις, μία από αυτές βρίσκεται στο figure 1 και πιο συγκεκριμένα από το μοντέλο Inception V3.

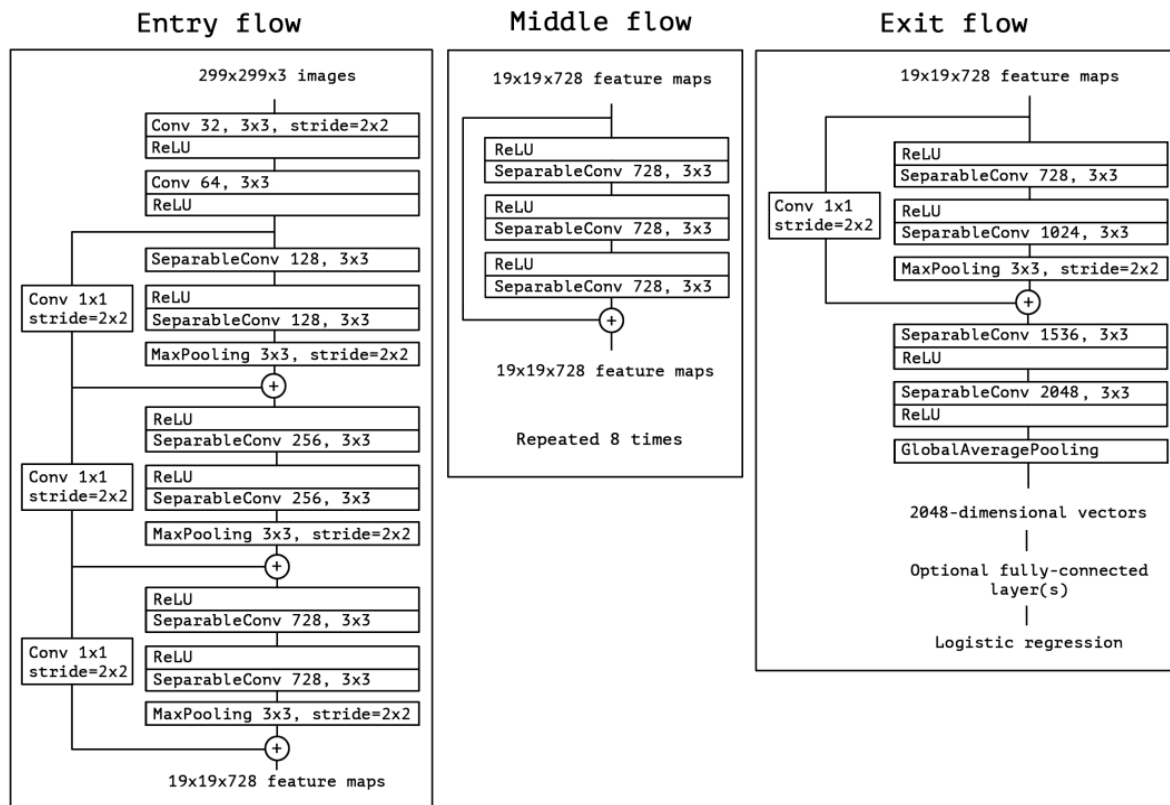


Εικόνα 8: Μια κανονική μονάδα Inception (Inception V3)

Πηγή: [14]

Ένα μοντέλο Inception μπορεί να γίνει κατανοητό ως μια στοίβα από πανομοιότυπες ενότητες. Αυτή είναι μια απόκλιση από το προηγούμενο στυλ των δικτύων που ήταν στοίβες απλών στρώματων συνέλιξης (convolution layers).

Ενώ οι μονάδες Inception είναι εννοιολογικά παρόμοιες με τις συνελίξεις (είναι συνελκτικοί εξαγωγείς χαρακτηριστικών), εμπειρικά φαίνεται να είναι ικανές να μάθουν πιο ουσιαστικά χαρακτηριστικά με λιγότερες παραμέτρους. Στην αρχιτεκτονική Xception, τα δεδομένα περνούν πρώτα από τη ροή εισόδου και μετά από τη μεσαία ροή που επαναλαμβάνεται οκτώ φορές και τέλος περνούν μέσω της ροής εξόδου. Οι λειτουργίες Inceptions μπορούν να χαρακτηριστούν σαν ένα ενδιάμεσο σημείο ανάμεσα στις συνελίξεις και στις σε βάθος διαχωρισμένες συνελίξεις (depthwise separable convolutions). Σε αυτή την παρατήρηση βασίστηκε η ιδέα να αντικατασταθούν κάποιες λειτουργίες Inception με εις βάθος διαχωρισμένες συνελίξεις σε αρχιτεκτονικές νευρωνικών δικτύων για την αντιμετώπιση προβλημάτων όρασης.



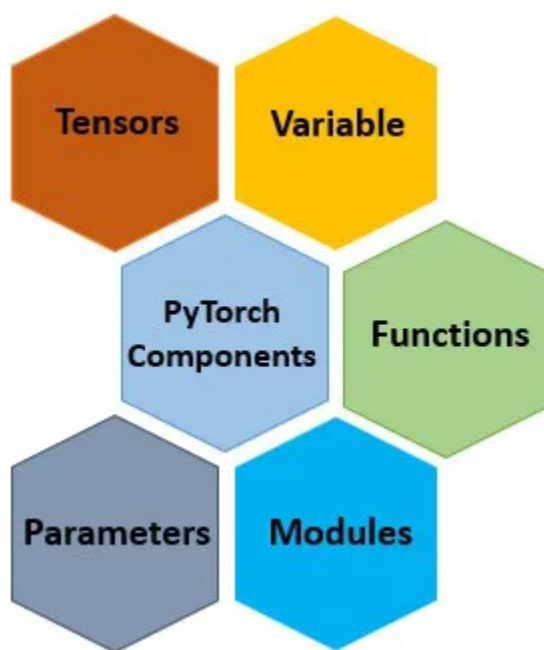
Εικόνα 9

Πηγή: [14]

4ο Εργαλεία και Βιβλιοθήκες

4.1 Pytorch

Για την δημιουργία του προγράμματος που πραγματοποιεί την εκπαίδευση χρησιμοποιήθηκε η προγραμματιστική γλώσσα Python και πιο συγκεκριμένα το framework PyTorch. Το PyTorch είναι μια βιβλιοθήκη μηχανικής εκμάθησης ανοιχτού κώδικα που δημιουργήθηκε χρησιμοποιώντας τη βιβλιοθήκη Torch της Python. Είναι μια δωρεάν βιβλιοθήκη ανοικτού κώδικα που δημιουργήθηκε από το τμήμα AI Research του Facebook και κυκλοφόρησε τον Ιανουάριο του 2016 για χρήση σε εφαρμογές υπολογιστικής όρασης, βαθιάς εκμάθησης και επεξεργασίας φυσικής γλώσσας. Το PyTorch περιέχει μια βασική δομή δεδομένων, τον ταυστή (tensor), ο οποίος είναι ένας πολυδιάστατος πίνακας παρόμοιος με τους πίνακες Numpy, που επιτρέπει στους προγραμματιστές να σχεδιάσουν εύκολα ένα περίπλοκο νευρωνικό δίκτυο. Το PyTorch γίνεται όλο και πιο δημοφιλές στους τρέχοντες τομείς και στην ακαδημαϊκή κοινότητα λόγω της ευελιξίας, της ταχύτητας και της ευκολίας στην εφαρμογή του, καθιστώντας το ένα από τα πιο δημοφιλή εργαλεία βαθιάς μάθησης. Ένας από τους λόγους για τους οποίους μπορεί κάποιος να θεωρήσει το PyTorch ως την καλύτερη επιλογή, είναι ότι σε αντίθεση με το TensorFlow και το Keras είναι τελείως βασισμένη στη Python και χρησιμοποιεί γραφήματα δυναμικών υπολογισμών (Dynamic Computation Graph), επιτρέποντας στο χρήστη να ελέγχει και να τρέχει κάθε κομμάτι κώδικα χωρίς να χρειάζεται να τελειώσει όλη διαδικασία. [15] [16]



Εικόνα 10

Πηγή: [15]

Τανυστές: Οι τανύστες ή αλλιώς Tensors, είναι ένα πολύ βασικό και σημαντικό μέρος του pytorch και της μηχανικής μάθησης γενικά. Για να αναπαρασταθούν τα δεδομένα στο ML χρειάζονται αριθμοί και οι τανυστές γενικά είναι μια προχωρημένη θεμελιώδη μονάδα που ταιριάζει στις ανάγκες της μηχανικής μάθησης και χρησιμοποιούνται ως δοχεία για τα δεδομένα μας. Πολλές από τις μορφές τις οποίες οι τανυστές μπορούν να πάρουν είναι, ένας αριθμός, μια μήτρα (matrix) και ένα διάνυσμα ή μία πολυδιάστατη συστοιχία όπως οι πίνακες Numpy. Υπάρχουν διάφοροι τύποι τανυστών, Float Tensor, Double Tensor, Half Tensor, Int Tensor, Long Tensor και δουλεύουν σε συνεργασία με την GPU και την CPU για πιο γρήγορες διεργασίες.

Μεταβλητές: Μια μεταβλητή ή Variable είναι ένα περιτύλιγμα γύρω από τους τανυστές που διατηρεί τη διαβάθμιση (gradient) στη θέση της.

Παράμετροι: Οι παράμετροι ή parameters είναι ουσιαστικά και πάλι ένα περιτύλιγμα γύρω από τις μεταβλητές και χρησιμοποιούνται όταν χρειάζεται ένας τανυστής ως παράμετρος μιας μονάδας (module) κάτι το οποίο δεν μπορεί να κάνει μια μεταβλητή ή οι τανυστές επειδή δεν έχουν διαβάθμιση (gradient).

Συναρτήσεις: Οι συναρτήσεις ή functions εκτελούν τις λειτουργίες μετασχηματισμού και δεν αποθηκεύουν στη μνήμη τις καταστάσεις. Θα δώσουν ως έξοδο την τιμή καταγραφής και το επίπεδο του γραμμικού στρώματος (linear layer) δεν μπορεί να λειτουργήσει ως συνάρτηση καθώς αποθηκεύει τις τιμές βάρους και προκατάληψης (weight and biases).

Μονάδες: Μονάδα ή module, είναι η βασική κλάση που χρησιμοποιείται από όλα τα νευρωνικά δίκτυα. Μία λειτουργική μονάδα έχει την ικανότητα να περιέχει περισσότερες από μία παραμέτρους, μονάδες ή λειτουργίες επιτρέποντας την δημιουργία πολύπλοκων συστημάτων. Μπορεί να αποθηκεύσει την κατάσταση (state) και τα βάρη εκμάθησης.

4.2 Weights And Biases

Ένα από τα αρνητικά της pytorch είναι η έλλειψη εργαλείων για την παρακολούθηση και οπτικοποίηση των πειραμάτων όπως το tensorboard, για αυτόν τον λόγο γίνεται η χρήση του Weights & Biases (WandB). Το WandB χρησιμοποιείται από όλους τους πρωτοπόρους ερευνητές και χρήστες AI, από εταιρείες όπως η SAMSUNG και η TOYOTA, μέχρι οργανισμούς σαν την OpenAI της Microsoft και την NVIDIA που έχουν σαν στόχο την μελέτη των συστημάτων AI. Είναι ένα πακέτο της Python που επιτρέπει στους χρήστες να

παρακολουθήσουν την διαδικασία της εκπαίδευσης, παρέχοντας στο χρήστη την δυνατότητα να το χρησιμοποιήσει σε συνδυασμό με την Pytorch, το TensorFlow ή το Keras. Ενσωματώνοντας το WandB απο την αρχή της διαδικασίας εκπαίδευσης ενός προγράμματος, στα παρασκήνια συλλέγει χρήσιμες πληροφορίες για το τι συμβαίνει. Οι μεγαλύτερες δυνάμεις του WandB βρίσκονται στο πόσο ξεκάθαρα αντικατοπτρίζει τα αποτελέσματα κάθε εκπαίδευσης, τόσο τα χαρακτηριστικά όπως ακρίβεια και λάθος, όσο και πιο τεχνικές λεπτομέρειες όπως χρήση GPU και CPU.

4.3 Django

Ένα web application framework το οποίο χρησιμοποιήθηκε επίσης είναι το Django, open source και στηριγμένο και αυτό στην python. Είναι μια ιδιαίτερα δημοφιλής επιλογή από πολλούς οργανισμούς και πολλές εταιρίες απο το Instagram μέχρι την NASA. Αποτελεί επίσης μια καλή επιλογή για πιο αρχάριους χρήστες καθώς υπάρχει πληθώρα πληροφορίας στο διαδίκτυο για κάθε πρόβλημα που μπορεί να αντιμετωπίσει ένας χρήστης. Το πιο αντιπροσωπευτικό του Django είναι αδιαμφισβήτητα το ότι ακολουθεί την λογική της python, συμπεριέχει δηλαδή πολλά ενσωματωμένα χαρακτηριστικά. Κάποια από αυτά είναι ένα επεκτάσιμο σύστημα ελέγχου ταυτότητας, servers για δοκιμές και μια πολύ χρήσιμη εφαρμογή διαχειριστή. Παρόλα αυτά το μεγαλύτερο πλεονέκτημα του Django δεν είναι οι ιδιότητες σαν framework αλλά η κοινότητα που έχει δημιουργηθεί γύρω από αυτό. Θα αναλυθεί περαιτέρω στο κεφάλαιο 7.

5^ο Μεθοδολογία εκπαίδευσης και προγραμματισμός

Για την εκπαίδευση του προγράμματος δημιουργήθηκε ένα πρόγραμμα train.py σε python,

```
import os
import albumentations
import timm
import torch
import wandb
```

```
from albumentations import augmentations as A
from albumentations.pytorch.transforms import ToTensorV2
from torch import nn
from glob import glob
```

Code snippet: train.py

Αρχικά γίνεται εισαγωγή των βιβλιοθηκών που χρειάζεται το πρόγραμμα. Δύο από τις εισαγωγές δημιουργήθηκαν για τις ανάγκες της παρούσας εργασίας (argparser.py, ArtistDataset.py) .

```
from ArtistDataset import ArtistData
from argparse import get_argparser
```

Code snippet: argparser.py

Το argparser.py χρησιμοποιείται για να γίνεται εύκολη η αλλαγή των παραμέτρων όταν εκτελείται το πρόγραμμα train.py το οποίο επιτρέπεται με την χρήση της λειτουργίας get_argparser και δημιουργώντας ένα λεξικό μέσα στην train.

```
parser = get_argparser()
args = parser.parse_args()
CONFIG = vars(args)
```

Code snippet: argparser.py

Οι μεταβλητές που μπορούν να αλλάξουν με την χρήση του parser είναι οι [model, learning rate, batch size, optimizer, loss, epochs, sample, folder και το artists number (κρατάει τους καλλιτέχνες που έχουν πάνω από τον αριθμό έργων που θα καθοριστεί)].

```
import argparse

def get_argparser():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-mc",
        "--model_conf",
        metavar='model_conf',
        type=str,
        default='resnet50'
    )
    parser.add_argument(
        "-lr",
        "--lr_conf",
        metavar='lr_conf',
        type=float,
        default=1e-4
    )
    parser.add_argument(
        "-b",
        "--batch_size",
        metavar='batch_size',
        type=int,
        default=32
    )
```

```

)
parser.add_argument(
    "-opt",
    "--optimizer",
    metavar='optimizer',
    type=str,
    default='Adam'
)
parser.add_argument(
    "-l",
    "--loss_conf",
    metavar='loss_conf',
    type=str,
    default='CrossEntropyLoss'
)
# add int argument for epochs
parser.add_argument(
    "-e",
    "--epochs",
    metavar='epochs',
    type=int,
    default=30
)
# add float argument for sample
parser.add_argument(
    "-s",
    "--sample",
    metavar='sample',
    type=float,
    default=1
)
)
parser.add_argument(
    "-f",
    "--folder",
    metavar='folder_config',
    type=str,
    default='0'
)
)
parser.add_argument(
    "-an",
    "--artists_number",
    metavar='artists_number',
    type=int,
    default='0'
)
)
return parser

```

Code snippet: argparse.py

Με την χρήση του parser και την δημιουργία ενός .bat type file υπάρχει η δυνατότητα να γίνουν πειράματα με διαφορετικές μεταβλητές αυτόματα. Το bat file έχει την παρακάτω μορφή,

call C:\Users\username\anaconda3\Scripts\activate.bat C:\Users\username\anaconda3

@echo off

cd "C:\Users\username\Desktop\Thesis"

python train.py -s 0.5 -lr 1e-5 -e 5 -f 1 (sample = 50% , learning rate = 1e-5 , epochs = 5 , folder = 1)

python train.py -s 0.5 -lr 1e-4 -e 5 -f 2 (sample = 50% , learning rate = 1e-4 , epochs = 5 , folder = 2)

python train.py -s 1 -lr 1e-5 -e 30 -f 4 (sample = 100% , learning rate = 1e-5 , epochs = 30 , folder = 4)

python train.py -s 1 -lr 1e-4 -e 30 -f 5 (sample = 100% , learning rate = 1e-4 , epochs = 30 , folder = 5)

python train.py -s 1 -lr 1e-3 -e 30 -f 6 (sample = 100% , learning rate = 1e-3 , epochs = 30 , folder = 6)

@pause

Το bat file εκτελείται στο Command Line, αρχικά καλεί την python μέσω του anaconda και με το cd πηγαίνει στο σημείο που βρίσκεται το πρόγραμμα. Αυτό δίνει την δυνατότητα να γίνουν πολλές προσπάθειες αυτόματα και με το συνδυασμό της ικανότητας του WandB να ελέγχει τη διαδικασία της εκπαίδευσης από σύνδεση στο κινητό, δόθηκε η ικανότητα να πραγματοποιούνται πολλά πειράματα από απόσταση. Με την παρακολούθηση της κάθε εποχής (epoch) από το WandB και τον τερματισμό της, το bat file ξεκινά αυτόματα την εκτέλεση του επόμενου πειράματος.

Στη συνέχεια στο ArtistDataset δημιουργείται μια κλάση μέσα στην οποία θα δοθούν τα datasets.

Τα imports τα οποία είναι απαραίτητα:

```
import cv2
import pandas as pd
import torch
import os
import numpy
from torch.utils.data import Dataset, DataLoader
from glob import glob
from sklearn.preprocessing import LabelEncoder
from random import shuffle
```

Code snippet: ArtistDataset.py

Στο ArtistData, γίνεται η συλλογή των εικόνων με την χρήση του glob.

```
class ArtistData:
    def __init__(
        self,
        base_path,
        metadata_csv,
        val_perc,
        batch_size,
        sample=1,
        label_encode=False,
        train_transforms=None,
```

```

        val_transforms=None,
    ):
        # DATA SET UP
        self.base_path = base_path
        self.train_path = os.path.join(base_path, "train", "*")
        self.test_path = os.path.join(base_path, "test", "*")
        self.train_data = glob(self.train_path)
        self.test_data = glob(self.test_path)

```

Code snippet: ArtistDataset.py

Αρχικοποιούνται όλες οι μεταβλητές οι οποίες θα χρειαστούν.

```

        self.val_perc = val_perc
        self.batch_size = batch_size
        self.metadata_csv = metadata_csv
        self.sample = sample
        self.label_encode = label_encode
        self.train_transforms = train_transforms
        self.val_transforms = val_transforms

```

Code snippet: ArtistDataset.py

Στη συνέχεια διαχωρίζεται το όνομα του κάθε αρχείου και διαβάζεται το κατάλληλο προ επεξεργασμένο metadata.csv που επιλέγεται σύμφωνα με την είσοδο του χρήστη για το artists number του argparser, για να επιβεβαιωθεί η ύπαρξη τους. Στο metadata αρχείο περιέχονται οι πληροφορίες που χρειάζεται το σύστημα για την εκπαίδευση και το testing, περιέχει τα ονόματα των αρχείων και σε ποιον καλλιτέχνη ανήκει το κάθε έργο. Περιέχει και άλλες πληροφορίες που μπορούν μελετηθούν παρακάτω [KaggleDataset](#), αλλά δεν είναι απαραίτητες για την υλοποίηση της παρούσας πτυχιακής.

```

self.metadata = pd.read_csv(os.path.join(self.base_path, self.metadata_csv))
    #to split data
    self.valid_imgs = self.metadata['new_filename'].apply(lambda x:
x.split('.')[0]).tolist()

    # load all images that are in the metadata column "new filename"
    self.train_data = list(filter(self.check_that_image_is_valid,
self.train_data))
    self.test_data = list(filter(self.check_that_image_is_valid,
self.test_data))

```

Code snippet: ArtistDataset.py

Χρησιμοποιείται το sample, ώστε να είναι δυνατή η λειτουργία του προγράμματος με μικρότερη ποσότητα δεδομένων λόγω του μεγάλου χρόνου που παίρνει η κάθε προσπάθεια. Το sample ορίζεται στην argparser και χωρίζει ανάλογα το train και validation set.

```

if sample != 1:
    #choice picks random
    s = int(len(self.train_data) * sample)
    s1 = int(len(self.test_data) * sample)

```

```

        self.train_data = numpy.random.choice(self.train_data, size= s,
replace=False)
        # do the same for test data
        self.test_data = numpy.random.choice(self.test_data, size=s1,
replace=False)

```

Code snippet: ArtistDataset.py

Διαβάζοντας το csv ή αλλιώς metadata, χρησιμοποιείται ο encoder στην στήλη που περιέχει τα ονόματα των καλλιτεχνών και την στήλη που περιέχει τα ονόματα με τα οποία είναι ανεβασμένα τα JPG files που περιέχουν τα έργα. Ακολουθεί η αντίστοιχη transform συνάρτηση, ανάλογα με το αν τα δεδομένα είναι στο train, στο eval ή στο test set και αρχικοποίηση των val, train και test dataset σύμφωνα με την κλάση ArtistDataset.

```

if self.label_encode == True:
    enc = LabelEncoder()
    self.metadata["artist_cat"] = enc.fit_transform(
        self.metadata["artist"].astype(str)
    )
    self.metadata["filename"] = enc.fit_transform(
        self.metadata["new_filename"].astype(str)
    )
    shuffle(self.train_data)
    shuffle(self.test_data)
    self.train_data, self.val_data = (
        self.train_data[int(len(self.train_data) * self.val_perc) :],
        self.train_data[: int(len(self.train_data) * self.val_perc)],
    )
    self.train_dataset = ArtistDataset(
        self.train_data, self.metadata, transforms=self.train_transforms
    )
    self.val_dataset = ArtistDataset(
        self.val_data, self.metadata, transforms=self.val_transforms
    )
    self.test_dataset = ArtistDataset(
        self.test_data, self.metadata, transforms=self.val_transforms
    )

```

Code snippet: ArtistDataset.py

Αρχικοποιούνται οι dataloaders πάλι ανάλογα με το αν τα δεδομένα είναι στο train, eval ή test. Γίνεται χρήση του shuffle για να είναι δίκαια πάντα η κατανομή.

```

def train_loader(self):
    return DataLoader(
        dataset=self.train_dataset, batch_size=self.batch_size, shuffle=True
    )

def val_loader(self):

```

```

        return DataLoader(
            dataset=self.val_dataset, batch_size=self.batch_size, shuffle=True
        )

    def test_loader(self):
        return DataLoader(
            dataset=self.test_dataset, batch_size=self.batch_size, shuffle=True
        )

```

Code snippet: ArtistDataset.py

Τέλος δημιουργείται μία συνάρτηση για να γίνεται έλεγχος του αν υπάρχει το όνομα του έργου μέσα στο csv που χρησιμοποιείται. Πραγματοποιείται ελέγχοντας αν το όνομα του αρχείου συμπεριλαμβάνεται στο metadata αρχείο που επιλεχθεί στην εκτέλεση του προγράμματος train.py.

```

def check_that_image_is_valid(self, img):
    img_num = img.split("\\")[-1].split('.')[0]
    return img_num in self.valid_imgs

```

Code snippet: ArtistDataset.py

Η κλάση ArtistDataset έχει σαν είσοδο τις λίστες με τις εικόνες που έχει δημιουργήσει η αρχική κλάση ArtistData μαζί με τα transforms και τα metadata. Με την χρήση της cv2 μετατρέπεται η κάθε εικόνα στη κατάλληλη μορφή για επεξεργασία. Αν έχει δηλωθεί ύπαρξη transform μετατρέπεται σε tensor και στην συνέχεια με την χρήση των metadata συνδέεται το id της κάθε εικόνας με τον καλλιτέχνη που της αντιστοιχεί.

```

class ArtistDataset(Dataset):
    def __init__(self, data, metadata, transforms=None):
        self.data = data
        self.metadata = metadata
        self.transforms = transforms

    def __getitem__(self, idx):
        img_path = self.data[idx]
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        if self.transforms is not None:
            img = self.transforms(image=img)["image"]
        return (
            img,
            torch.tensor(
                self.metadata[self.metadata.new_filename ==
img_path.split("\\")[-1]]["artist_cat"].to_numpy()[0]
            ),
        )

    def __len__(self):
        return len(self.data)

```

Code snippet: ArtistDataset.py

Επιστρέφοντας λοιπόν στο αρχικό πρόγραμμα train.py

Αρχικοποιούμε το WandB.

```
wandb.init(project="FinalRuns", entity="frem", save_code=True, config=CONFIG)
wandb.config = CONFIG
```

Code snippet: train.py

Δημιουργία των transforms. Τα κοινά μεταξύ των δύο transform για train και test/eval είναι η μετατροπή σε tensors με την χρήση του ToTensorV2, η αλλαγή του μεγέθους των εικόνων για να είναι όλες ίδιο μέγεθος και τέλος το normalization. Τα υπόλοιπα που υπάρχουν στο train transform είναι δημοφιλείς μετατροπές οι οποίες συχνά χρησιμοποιούνται κυρίως για να αντιμετωπιστεί το overfitting.

```
# TRANSFORMS
train_transforms = albumentations.Compose(
    [
        A.Downscale(p=0.3), # cuz overfit
        A.Blur(blur_limit=5, p=0.4), # cuz overfit
        A.Flip(),
        A.geometric.Resize(224, 224),
        A.geometric.transforms.Affine(scale=(1, 0.9), translate_percent=(0, 0.2),
rotate=(-10, 10), shear=(-10, 10), p=0.7),
        A.transforms.Emboss(alpha=(0.2, 0.9), strength=(0.2, 1.3),
always_apply=False, p=0.5),
        A.Normalize(),
        ToTensorV2(),
    ]
)

val_transforms = albumentations.Compose(
    [A.geometric.Resize(224, 224), A.Normalize(), ToTensorV2()]
)
```

Code snippet: train.py

Στη συνέχεια ανάλογα με τον ελάχιστο αριθμό έργων που είναι επιθυμητό να έχουν οι καλλιτέχνες στο κάθε πείραμα επιλέγεται ένα από τα csv, είτε για παραπάνω από 50 έργα, 100 έργα, 200 έργα, είτε όλοι οι καλλιτέχνες.

```
if CONFIG["artists_number"] == 200:
    csv_choice = metadata_choice =
r"C:\Users\User\Desktop\Thesis\metadata_preprocessed_200.csv"
elif CONFIG["artists_number"] == 100:
    csv_choice = metadata_choice =
r"C:\Users\User\Desktop\Thesis\metadata_preprocessed_100.csv"
elif CONFIG["artists_number"] == 50:
    csv_choice = metadata_choice =
r"C:\Users\User\Desktop\Thesis\metadata_preprocessed_50.csv"
```

```

else:
    csv_choice = metadata_choice =
r"C:\Users\User\Desktop\Thesis\metadata_preprocessed.csv"
Code snippet: train.py

```

Χρήση του προαναφερόμενου ArtistData για την εισαγωγή των δεδομένων.

```

data = ArtistData(
    base_path="D:\Art DataBase",
    metadata_csv=csv_choice,
    val_perc=0.2,
    batch_size=CONFIG["batch_size"],
    sample= CONFIG["sample"],
    label_encode=False,
    train_transforms=train_transforms,
    val_transforms=val_transforms,
)
Code snippet: train.py

```

Αρχικοποίηση του μοντέλου ανάλογα με την αρχιτεκτονική που επέλεξε ο χρήστης, με 6 επιλογές [Xception, EfficientNet, Vision Transformer, ResNet50, Resnet34, ResNet50d, Resnet18] και αρχικοποίηση των παραμέτρων (weights και biases) του μοντέλου.

Καθώς η κάθε αρχιτεκτονική έχει διαφορετική δομή, πρέπει να γίνει μετατροπή του τελευταίου επιπέδου, ώστε να ανταποκρίνεται στις ανάγκες των συγκεκριμένων δεδομένων.

Το out_features σε όλες τις περιπτώσεις αλλάζει σε 2319, το οποίο είναι ο αριθμός των καλλιτεχνών που υπάρχουν στο dataset. Το in_features είναι διαφορετικό σε κάθε μοντέλο και εξαρτάται από την έξοδο του προηγούμενου μοντέλου. Τα out_features παραμένουν 2319 και στις περιπτώσεις που γίνεται επιλογή μικρότερου αριθμού καλλιτεχνών, καθώς επιλέγουμε μόνο τα έργα από τους καλλιτέχνες που τηρούν τις προϋποθέσεις, οπότε δεν επηρεάζει τα αποτελέσματα.

```

# MODEL SET UP
gpu = torch.device("cuda:0")
model = timm.create_model(model_name=CONFIG["model_conf"], pretrained=True)
if (CONFIG["model_conf"] == 'efficientnet_b0'):
    last_fc = nn.Linear(in_features=1280, out_features=2319)
    model.classifier = last_fc
elif (CONFIG["model_conf"] == 'xception'):
    last_fc = nn.Linear(in_features=2048, out_features=2319)
    model.fc = last_fc
elif (CONFIG["model_conf"] == 'vit_base_patch16_224', pretrained=True):
    last_fc = nn.Linear(in_features=768, out_features=2319)
    model.head = last_fc
elif (CONFIG["model_conf"] == 'resnet34', pretrained=True):
    model.fc = nn.Sequential(
        #nn.Dropout(0.5),
        nn.Linear(in_features=512, out_features=2319)
    )

```

```

elif (CONFIG["model_conf"] == 'resnet50d', pretrained=True):
    model.fc = nn.Sequential(
        nn.Dropout(0.5),
        nn.Linear(in_features=2048, out_features=2319)
    )
elif (CONFIG["model_conf"] == 'resnet18', pretrained=True):
    model.fc = nn.Sequential(
        nn.Dropout(0.5),
        nn.Linear(in_features=2048, out_features=2319)
    )
else:
    model.fc = nn.Sequential(
        nn.Dropout(0.5),
        nn.Linear(in_features=2048, out_features=2319)
    )

```

Code snippet: train.py

Μεταφορά των παραμέτρων από τα μοντέλα, δήλωση του μοντέλου στην gpu ώστε να μειωθεί ο χρόνος που χρειάζεται για την προπόνηση. Όλες οι πληροφορίες για το μοντέλο δίνονται στο WandB για να κρατηθούν οι απαραίτητες.

```

params = model.parameters()
model.to(gpu)
wandb.watch(model)
print("\tMODEL SET UP DONE")

```

Code snippet: train.py

Γίνεται χρήση των αρχικοποιημένων loaders στα δεδομένα.

```

train_loader = data.train_loader()
val_loader = data.val_loader()
test_loader = data.test_loader()

```

Code snippet: train.py

Αρχικοποίηση του Adam, ενός αλγόριθμου βελτιστοποίησης για εκπαίδευση μοντέλων βαθιάς μάθησης, [18]

```

# TRAINING SET UP

optimizer = torch.optim.Adam(
    params,
    lr=CONFIG["lr_conf"],
    betas=(0.9, 0.999),
    eps=1e-08,
    weight_decay=1e-5,
    amsgrad=False,
)

```

Code snippet: train.py

Αρχικοποιείται ένας learning rate scheduler, ένα προκαθορισμένο πλαίσιο που προσαρμόζει το ρυθμό μάθησης μεταξύ των εποχών ή των επαναλήψεων κατά την διάρκεια εκπαίδευσης, πιο

συγκεκριμένα μειώνει το learning rate κατά gamma. Δηλώνεται η συνάρτηση λάθους, στη περίπτωση μας το CrossEntropyLoss.

```
scheduler = torch.optim.lr_scheduler.ExponentialLR(
    optimizer,
    gamma=0.1
)
criterion = nn.CrossEntropyLoss()
```

Code snippet: train.py

Γίνεται δημιουργία λιστών για να διατηρηθούν τα αποτελέσματα.

```
for epoch in range(CONFIG['epochs']):
    wandb.log({"epoch": epoch})

    # TRAIN LOOP
    model = model.train() # because of dropout in train we want it
    train_losses = list()
    train_accuracies = list()
```

Code snippet: train.py

Στο Training κομμάτι του κώδικα

- δηλώνεται χρήση της gpu στα δεδομένα που αντλήθηκαν από τις παραμέτρους
- αρχικοποίηση των gradients
- υπολογισμός του λάθους
- χρήση του backward() που αποτελεί το back propagation και είναι ένας αλγόριθμος για την επίβλεψη νευρωνικών δικτύων με την χρήση του gradient descent
- ένα “βήμα” για τον optimizer.

```
for i, data in enumerate(train_loader):
    inputs, labels = data
    inputs, labels = inputs.to(gpu), labels.to(gpu)

    labels = labels.long()

    # zero the parameter gradients
    optimizer.zero_grad()
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    outputs = torch.softmax(outputs, dim=1)
    loss.backward()
    optimizer.step()

    train_batch_loss = loss.cpu().item()
    train_losses.append(train_batch_loss)
```

```

        train_batch_acc =
labels.eq(outputs.detach().argmax(dim=1)).float().mean().cpu()
        train_accuracies.append(train_batch_acc)

        wandb.log(
            {"train_batch_loss": train_batch_loss, "train_batch_acc":
train_batch_acc}
        )

        wandb.log(
            {
                "epoch_train_loss": torch.tensor(train_losses).mean(),
                "epoch_train_acc": torch.tensor(train_accuracies).mean(),
            }
        )

```

Code snippet: train.py

Το ίδιο για το Evaluation, με την διαφορά ότι δεν χρησιμοποιείται back propagation και δεν χρειάζεται να κάνει βήμα ο optimizer.

```

# VALIDATION LOOP
model = model.eval() # because of dropout in eval we dont
val_losses = list()
val_accuracies = list()
with torch.no_grad():
    for i, data in enumerate(val_loader):
        inputs, labels = data
        labels = labels.long()
        inputs, labels = inputs.to(gpu), labels.to(gpu)
        outputs = model(inputs)

        # 2 compute objective function (how well the network does)
        loss = criterion(outputs, labels)
        outputs = torch.softmax(outputs, dim=1)
        batch_val_loss = loss.cpu().item()
        val_losses.append(batch_val_loss)
        batch_val_acc = (
            labels.eq(outputs.detach().argmax(dim=1)).float().mean().cpu()
        )
        val_accuracies.append(batch_val_acc)

```

Code snippet: train.py

Όσον αφορά την χρήση του softmax, είναι μία συνάρτηση η οποία δέχεται σαν είσοδο έναν vector με έναν αριθμό από τιμές και επιστρέφει έναν vector με τον ίδιο αριθμό τιμών μεταμορφωμένες ανάμεσα σε 0 και 1. Είναι πολύ σημαντικό να προστεθεί μετά από τον υπολογισμό και την χρήση των losses καθώς οι τιμές δεν πρέπει να αλλοιωθούν μέχρι τότε.

```

if (epoch > 12):
    scheduler.step()
wandb.log(
    {

```

```

        "epoch_val_loss": torch.tensor(val_losses).mean(),
        "epoch_val_acc": torch.tensor(val_accuracies).mean(),
    }
)

torch.save(
    model.state_dict(),
    os.path.join(
        "D:\Art DataBase\models\exception",
        f"{CONFIG['folder']}\{CONFIG['model_conf']}_epoch}.pth"
    ),
)

wandb.save(
    os.path.join("D:\Art DataBase\models\exception",
        f"{CONFIG['folder']}\{CONFIG['model_conf']}_epoch}.pth" )
)

```

Code snippet: train.py

Τέλος, για το testing όπως για το evaluation set η αρχικοποίηση γίνεται με `torch.no_grad` καθώς δεν χρειάζεται η συλλογή των gradients στα συγκεκριμένα βήματα. Αφού περάσουν οι πληροφορίες μέσα από το μοντέλο, δημιουργείται μία λίστα με τα ταξινομημένα outputs.

```

model.eval()

correct5 = 0
correct3 = 0
correct = 0
total = 0
acc5 = 0
acc3 = 0
acc = 0

with torch.no_grad():
    for data in test_loader:
        images, labels = data
        images, labels = inputs.to(gpu), labels.to(gpu)

        # calculate outputs by running images through the network
        outputs = model(images)

```

```

outputs = torch.softmax(outputs, dim=1)
sorted_outs = torch.argsort(outputs, dim=1, descending=True)
total += labels.size(0)
for i, label in enumerate(labels):
    if label in sorted_outs[i][:5]:
        correct5 += 1
for i, label in enumerate(labels):
    if label in sorted_outs[i][:3]:
        correct3 += 1
for i, label in enumerate(labels):
    if label in sorted_outs[i][:1]:
        correct += 1
acc5 = 100 * correct5 / total
acc3 = 100 * correct3 / total
acc = 100 * correct / total
print(f'Accuracy of the network test images top 1 : {acc} %')
print(f'Accuracy of the network test images top 3 : {acc3} %')
print(f'Accuracy of the network test images top 5: {acc5} %')

```

Αυτή η λίστα χρησιμοποιηθεί για να βρεθούν τα top-1, top-3 και top-5 accuracies αν ο σωστός καλλιτέχνης βρίσκεται στη πρώτη θέση για το top-1, στις πρώτες 3 για το top-3 και στις πρώτες 5 για top-5. Η προπόνηση έγινε αποκλειστικά με στόχο την βελτίωση του top-1 accuracy.

6^ο Αποτελέσματα και Εκπαίδευση

6.1 Αρχική επιλογή μοντέλου

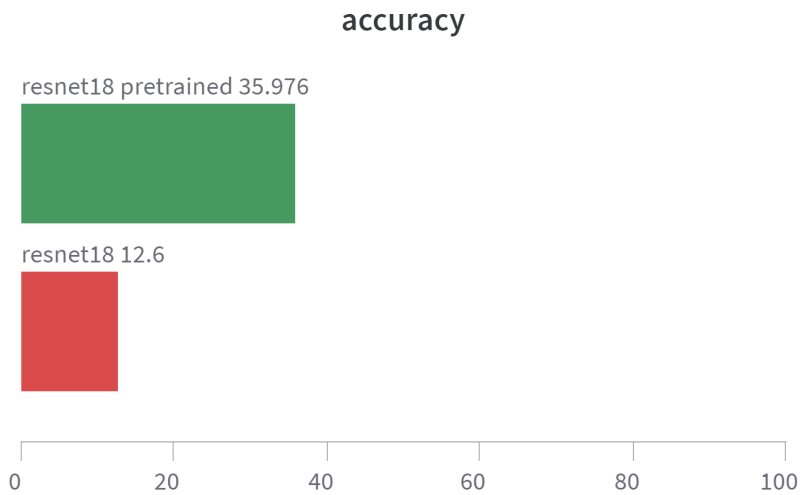
Η διαδικασία της εκπαίδευσης ξεκινάει με την επιλογή των αρχιτεκτονικών οι οποίες θα είναι κατάλληλες για το πρόβλημα αναγνώρισης που αντιμετωπίζεται. Μετά την μελέτη προσπαθειών από προηγούμενες έρευνες για την δημιουργία μοντέλου χαρακτηρισμού καλλιτέχνη ή καλλιτεχνικής τεχνοτροπίας (style), η έρευνα που ξεχωρίζει πραγματοποιήθηκε από το Stanford University με ακρίβεια **0.77** σε Top-1 και **0.89** σε Top-3 με την χρήση ResNet-18 σε μια μικρότερη βάση δεδομένων, για περισσότερα βλ. στο [17]. Στην παρούσα πτυχιακή τα πρώτα πειράματα πραγματοποιήθηκαν με την χρήση του ResNet-18 και στη συνέχεια δοκιμάστηκαν και πιο ισχυρά σύγχρονα μοντέλα και λειτουργίες. Λόγω του μεγάλου μεγέθους πληροφοριών, η διαδικασία της εκπαίδευσης ιδιαίτερα στα μεγαλύτερα και πιο πολύπλοκα μοντέλα αποδείχθηκε ιδιαίτερα χρονοβόρα. Για την λύση αυτού του προβλήματος χρησιμοποιήθηκε η τεχνική που ονομάζεται sampling, με την οποία τυχαία επιλέγονται ποσοστά από τα έργα τέχνης που βρίσκονται στα υπάρχοντα δεδομένα, ώστε να πραγματοποιηθούν πολλά πειράματα και τέλος αφού πραγματοποιηθεί η προσπάθεια να γίνει η εκπαίδευση με όλους τους καλλιτέχνες, να γίνει ένας καθαρισμός των δεδομένων ανάλογα με τον αριθμό των έργων του κάθε καλλιτέχνη.

6.2 Transfer Learning - Pre-trained

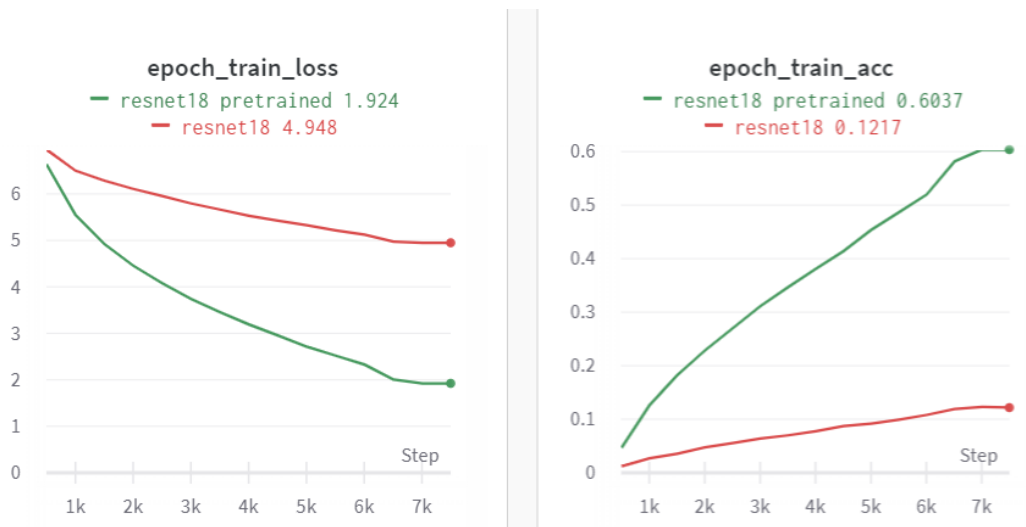
Τα αρχικά δύο μοντέλα θα προπονηθούν για 15 εποχές με τη μισή ποσότητα από τις εικόνες. Έγινε μία προσπάθεια με το μοντέλο να είναι προεκπαιδευμένα στο ImageNet και μια δεύτερη προπόνηση από την αρχή.

Ο τρόπος που χρησιμοποιείται το μοντέλο εκπαιδευμένο στο ImageNet, είναι φυσικά με διαδικασία μεταφοράς μάθησης. Αυτό το πείραμα πραγματοποιήθηκε για να αποδειχθεί, το αν τα χαρακτηριστικά που μεταφέρονται από το ImageNet βοηθούν την διαδικασία της εκπαίδευσης για την αναγνώριση καλλιτεχνών.

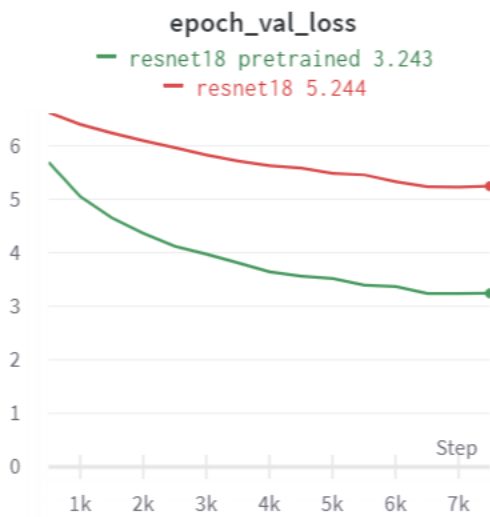
Στην αρχή ο υπολογισμός της ακρίβειας έγινε μόνο για top-1, καθώς όσον αφορά την διαδικασία εκπαίδευσης η βελτίωση σε top-1 θα οδηγήσει σε βελτίωση και για top-3 και top-5.



Είναι έντονα αισθητή η διαφορά ακρίβειας στα αποτελέσματα στις δύο περιπτώσεις με το προεκπαιδευμένο μοντέλο να ξεπερνά αισθητά το εκπαιδευμένο από την αρχή.



Η διαφορά είναι αισθητή και στο training και στο evaluation.

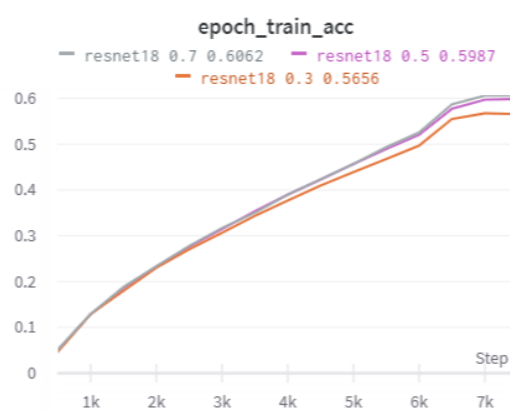
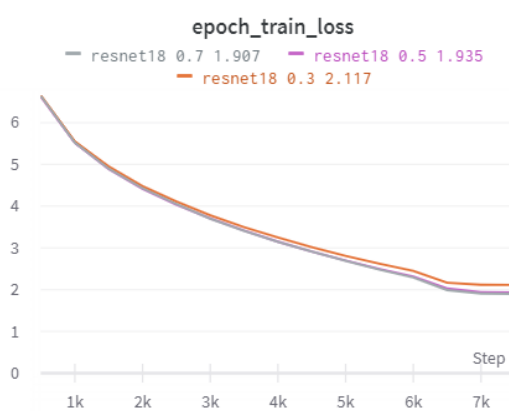
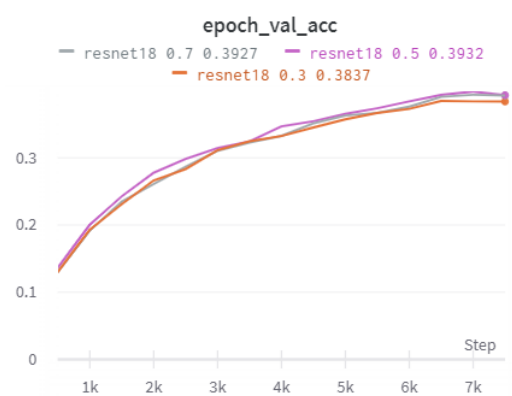
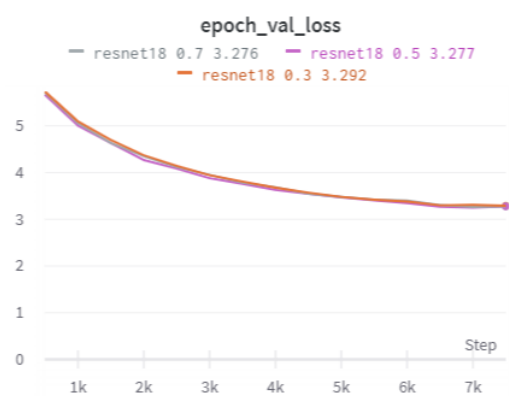
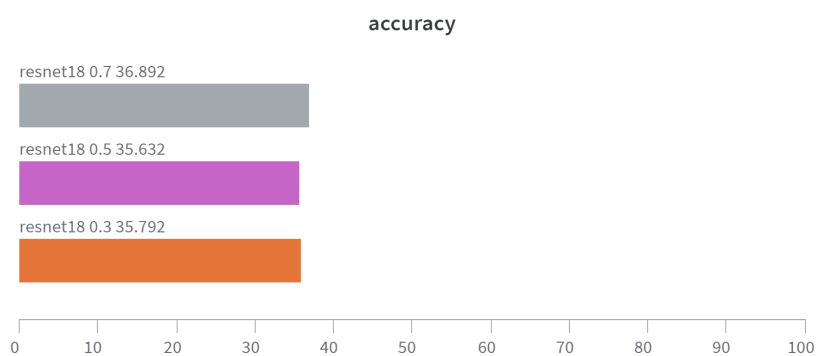


6.3 Fine Tuning

Στα 2 προηγούμενα παραδείγματα τα βάρη δεν αλλάζουν κατά την καινούργια εκπαίδευση και το μόνο που αλλάζει είναι το τελευταίο επίπεδο της αρχιτεκτονικής, για να γίνει δυνατή η κατανομή των αποτελεσμάτων στους καλλιτέχνες. Αυτή η τεχνική ονομάζεται fine tuning και τα βάρη είναι “ξεπαγωμένα” και εκπαιδεύονται μαζί με τον τελικό ταξινομητή.

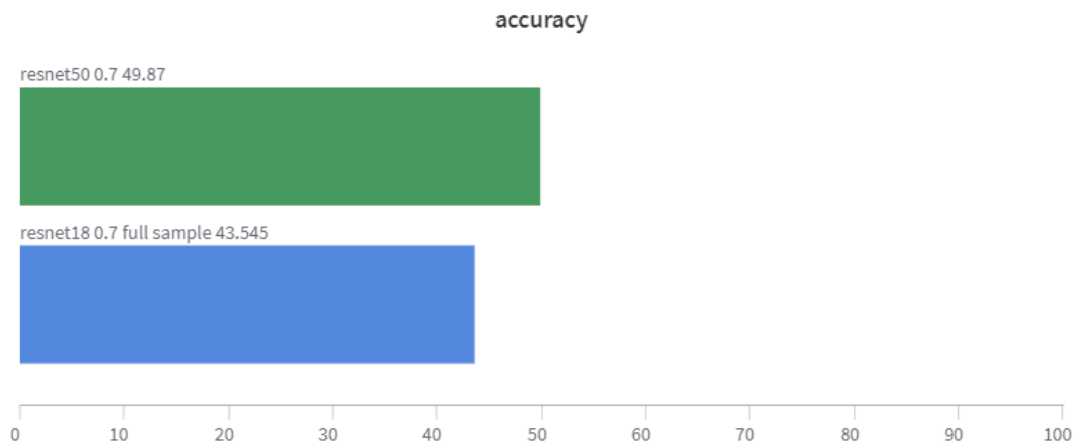
Στη συνέχεια έγινε μία προσπάθεια να παγώσουν ορισμένα επίπεδα του ταξινομητή, εκτός από τον τελικό ταξινομητή. Τα παρακάτω παραδείγματα έχουν και πάλι $\text{sample} = 0.5$ για την πραγματοποίηση πολλών πειραμάτων και χρησιμοποιείται το ResNet-18.

Έγιναν “παγώματα” στα βάρη για το 30%, το 50% και το 70% από αυτά. Τα καλύτερα αποτελέσματα τα έφερε το 70% με **37%** ακρίβεια αλλα με πολύ αμελητέα διαφορά.



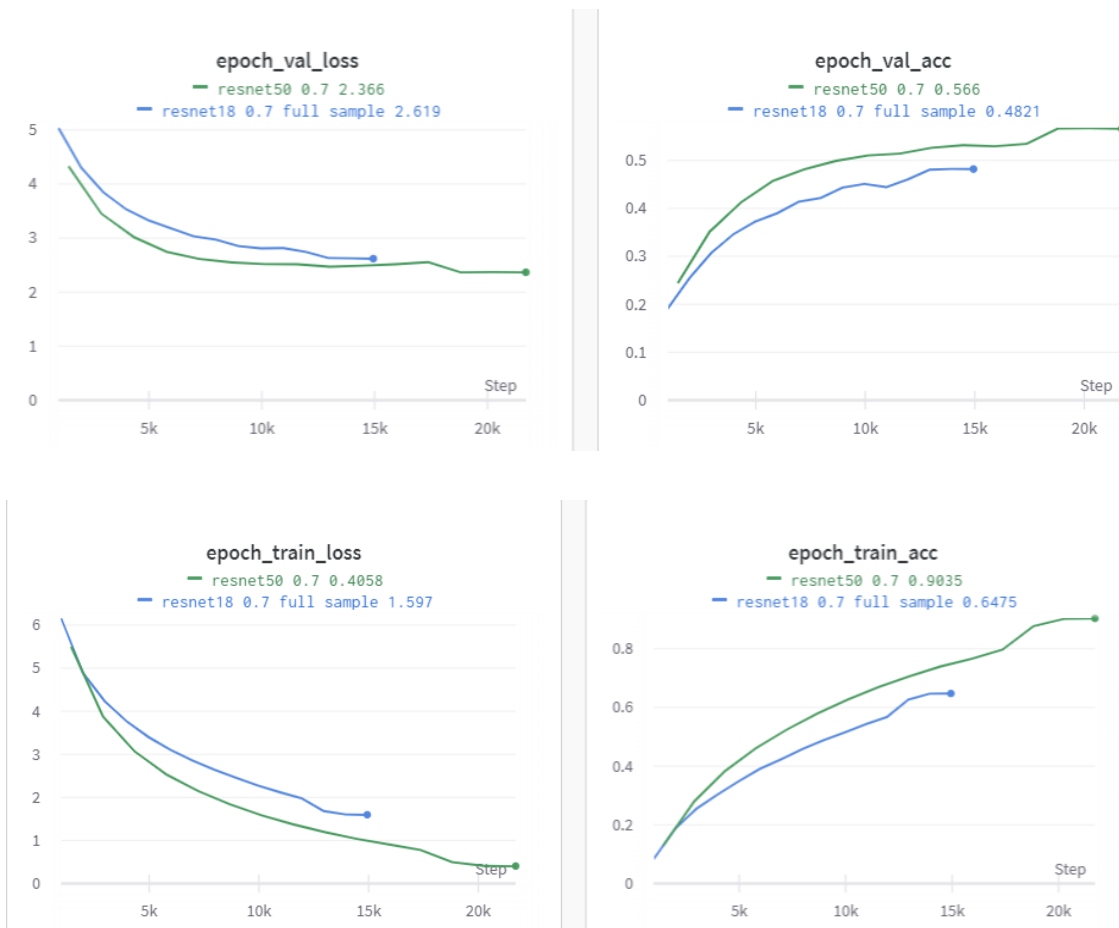
6.4 Overfitting

Έχοντας δοκιμάσει την προπόνηση μόνο με τα μισά δεδομένα, αξίζει να πραγματοποιηθεί μια προσπάθεια με ολόκληρο το dataset. Γνωρίζοντας ότι στο test set η ακρίβεια είναι μεγαλύτερη όταν παγώσει το 70% από τα βάρη πραγματοποιείται με αυτό τον τρόπο το fine-tuning. Τα δύο μοντέλα αρχιτεκτονικής που θα χρησιμοποιήσουμε είναι το ResNet-18 και ResNet-50.



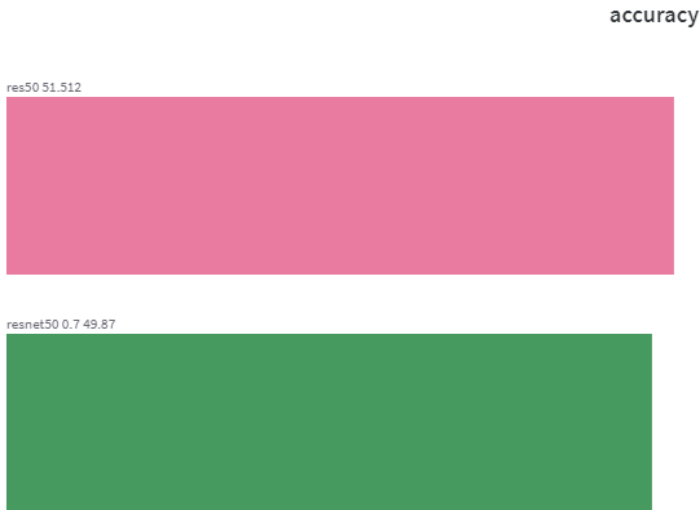
Κάποιες διαφορές που είναι αξιοσημείωτες είναι η διαφορά στο batch_size, καθώς λόγω των διαφορετικών διαστάσεων και της υπολογιστικής δύναμης οι μεγαλύτεροι αριθμοί από batches που μπορούσαν να χρησιμοποιηθούν ήταν batch_size = 64 για το ResNet-18 και batch_size = 44 για το ResNet-50.

Η διαφορά της ακρίβειας στα δύο μοντέλα, παρόλο που το ResNet-50 είναι ένα ισχυρότερο μοντέλο, είναι ένα αμελητέο ποσό του 6% παραπάνω με περίπου **50%** ακρίβεια για το ResNet50, με το ResNet-18 να φτάνει το **43.5%**. Βλέποντας τα διαγράμματα της εκπαίδευσης στο training και στο validation είναι αρκετά ξεκάθαρο ότι είναι μια περίπτωση overfitting.



Ενώ και στα δύο μοντέλα αντιμετωπίζεται το πρόβλημα του overfitting, στο ResNet-50 φαίνεται το μεγαλύτερο πρόβλημα, καθώς υπάρχει μεγάλη διαφορά με 0.90 στο training και 0.56 στο eval.

Τα καλύτερα αποτελέσματα μέχρι αυτό το σημείο τα έχει ένα ResNet50 μοντέλο που έτρεξε για 17 εποχές χωρίς κανένα παγωμένο επίπεδο, με ακρίβεια **51.5 %**. Μια ελάχιστη βελτίωση οφείλεται είτε στο μεγαλύτερο αριθμό εποχών, είτε στο ότι οι παράμετροι που παγώνουν όταν προπονούνται στο ImageNet δεν μεταφράζονται καλά στα δεδομένα, οπότε δεν υπάρχει λόγος να χρησιμοποιηθεί η τεχνική. Αλλά η διαφορά τους είναι πολύ μικρή οπότε δεν είναι δυνατό να βγει ένα σίγουρο συμπέρασμα.



Όσον αφορά το **overfitting**, αποτελεί ένα θεμελιώδες πρόβλημα για την μηχανική μάθηση. Εμφανίζεται στην περίπτωση που η διαδικασία μάθησης αποτυγχάνει στο να γενικοποιήσει ικανοποιητικά τα δεδομένα, με αποτέλεσμα να μην πραγματοποιείται σωστή αντιστοίχιση μεταξύ των δεδομένων που μαθαίνει στην εκπαίδευση και των συμπερασμάτων που παράγει στα δεδομένα ελέγχου (testing data). Η εκπαίδευση με το overfitting οδηγεί σε καλά αποτελέσματα στα δεδομένα εκπαίδευσης, αλλά η πληροφορία δεν βοηθάει στα δεδομένα ελέγχου. Αυτό συμβαίνει λόγω της ύπαρξης ήχου (noise) που δημιουργείται επειδή το μοντέλο “ταιριάζει” πολύ στενά στα δεδομένα εκπαίδευσης (training data) με αποτέλεσμα να μαθαίνει όλα τα δεδομένα, συμπεριλαμβανόμενων και χαρακτηριστικών που δεν χρειάζονται, αντί να συγκρατήσει τις αρχές που καθορίζουν και χαρακτηρίζουν τα δεδομένα. Αξίζει να σημειωθεί ότι ο χωρισμός των δεδομένων σε train και eval συμβαίνει για να μπορεί να εντοπιστεί το overfitting.

Τα βασικά προβλήματα που μπορούν να προκαλέσουν το φαινόμενο του overfitting μπορούν κατηγοριοποιηθούν σε 2 κατηγορίες.

- 1) Ο ήχος που μαθαίνεται από τα δεδομένα μάθησης παρατηρείται όταν ο αριθμός των δεδομένων είναι πολύ μικρός, όταν υπάρχουν λιγότερα αντιπροσωπευτικά δεδομένα ή υπάρχουν πολλοί θόρυβοι. Αυτό κάνει πιθανή την χρήση πληροφορίας από τους θορύβους για την επιλογή αποφάσεων, οπότε θα πρέπει ο αλγόριθμος να καταλαβαίνει ανάμεσα σε θόρυβο και αντιπροσωπευτικά δεδομένα.
- 2) Πολυπλοκότητα υπόθεσης: η αντιστάθμιση της πολυπλοκότητας, πιο συγκεκριμένα η ισορροπία ανάμεσα στο Bias και στο Variance που μεταφράζεται στην ακρίβεια και την συνοχή που χαρακτηρίζουν την σχέση ενός μοντέλου με τα δεδομένα.

6.4.1 Augmentations

Η αντιμετώπιση του προβλήματος του overfitting αλλάζει ανάλογα με πολλούς παράγοντες σε κάθε πρόβλημα εκπαίδευσης.

Μία από τις πιο σίγουρες και αποτελεσματικές τεχνικές για την αντιμετώπιση του overfitting είναι η συλλογή και ενσωμάτωση περισσότερων δεδομένων στη βάση δεδομένων. Αυτό μπορεί να υλοποιηθεί με πληθώρα τρόπων, είτε με την απόκτηση περισσότερων δεδομένων, είτε προσθέτοντας επίτηδες ήχο στα ήδη υπάρχοντα δεδομένα για τη δημιουργία καινούργιων δεδομένων σε ορισμένα προβλήματα. Καθώς στο συγκεκριμένο πρόβλημα αναγνώρισης καλλιτέχνη είναι πολύ δύσκολο να βρεθούν κατάλληλα δεδομένα με σωστές ετικέτες, κυρίως χρησιμοποιούνται τεχνικές για τη δημιουργία δεδομένων με διάφορα διάσημα augmentations στα δεδομένα εκπαίδευσης προσθέτοντας ήχο επίτηδες.

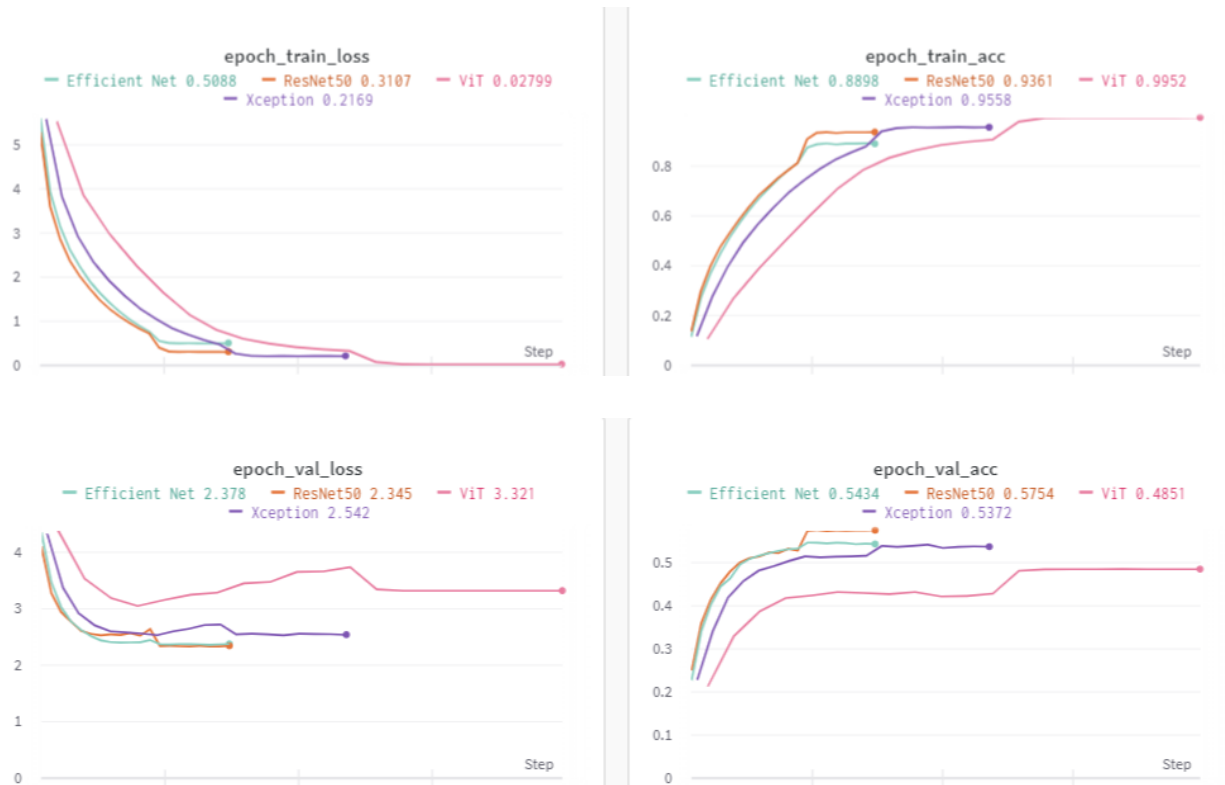
Βλέποντας ότι κατά την εκπαίδευση το δίκτυο κάνει overfit, έγινε σταδιακά ενσωμάτωση ορισμένων augmentations παραπάνω στο train. Αξίζει να σημειωθεί ότι το evaluation transform (χρησιμοποιείται και στο test set) δεν αλλάζει καθώς δεν θέλουμε να προσθέσουμε ήχο σε αυτό.

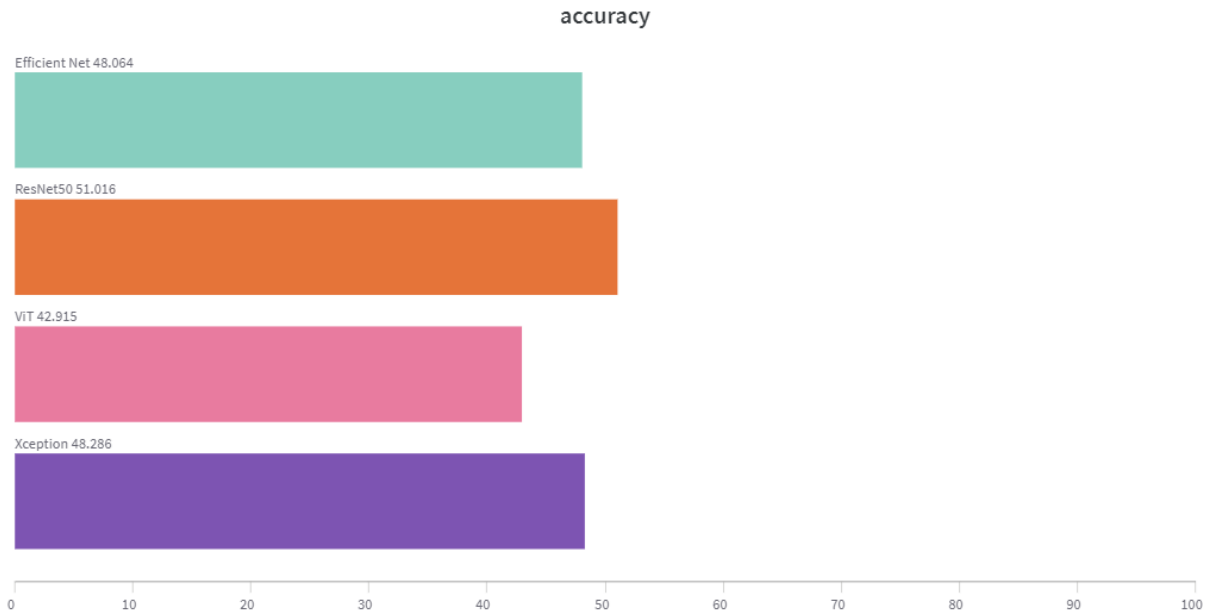
```
train_transforms = albumentations.Compose([
    A.Downscale(p=0.3),
    A.Blur(blur_limit=5, p=0.4),
    A.Flip(),
    A.geometric.Resize(224, 224),
    A.geometric.transforms.Affine(scale=(1, 0.9), translate_percent=(0, 0.2), rotate=(-10, 10), shear=(-10, 10), p=0.7),
    A.transforms.Emboss(alpha=(0.2, 0.9), strength=(0.2, 1.3), always_apply=False, p=0.5),
    A.Normalize(),
    ToTensorV2(),
])
```

Code snippet: train.py (new augmentations)

6.4.2 Άλλες Αρχιτεκτονικές

Στη συνέχεια έγιναν κάποιες προσπάθειες να χρησιμοποιηθούν και άλλα δίκτυα τα οποία θεωρούνται η τελευταία λέξη της τεχνολογίας στον κόσμο της βαθιάς εκμάθησης.



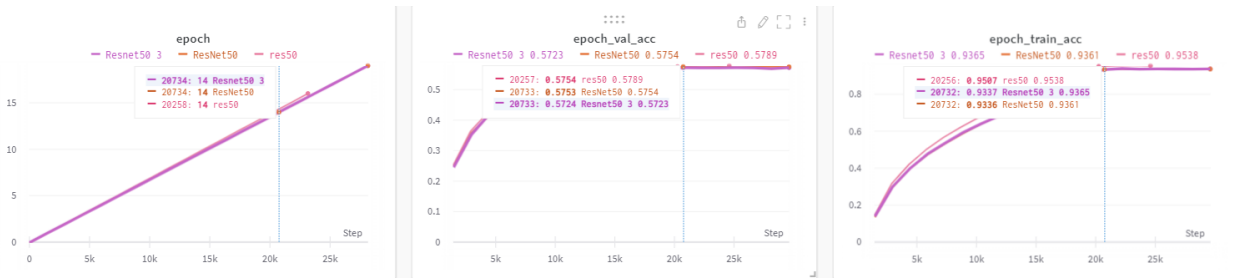


Βλέποντας τα αποτελέσματα το ResNet50 ακόμα κρατάει την πρώτη θέση με ακρίβεια **51%**. Επίσης γίνεται φανερό ότι ακόμα υπάρχει overfit σε όλα τα μοντέλα που χρησιμοποιήθηκαν.

Τέλος βρέθηκε από την βιβλιοθήκη timm ένα καινούργιο σχετικά ResNet50, το ResNet50-d το οποίο είναι αποτέλεσμα του συνδυασμού διάφορων βελτιώσεων της διαδικασίας, σύμφωνα με το ResNet-D trick. Περισσότερα στο [20].

6.4.3 Early-Stopping

Ένας άλλος διαδεδομένος τρόπος είναι η χρήση της τεχνικής του Early-stopping, που αποσκοπεί στην διακοπή της εκπαίδευσης μετά το σημείο που το evaluation σταματάει να βελτιώνεται και το training συνεχίζει να βελτιώνεται δίνοντας όμως χειρότερα αποτελέσματα στην ακρίβεια. Χάρη στο WandB μπορούν να εντοπιστούν τα σημεία στα οποία το overfit αρχίζει και επηρεάζει το δίκτυο και μπορούν να ελεγχθούν τα σημεία στα οποία το Early-stopping θα σταματούσε την εκπαίδευση.



Στο παραπάνω σχήμα παρουσιάζεται η διαδικασία εκπαίδευσης και το πώς σταματάει να αυξάνεται το accuracy τόσο στο train όσο και στο evaluation. Τρία μοντέλα resnet50 με διαφορετικό αριθμό augmentations φαίνεται ότι μετά την 14 εποχή το training έχει φτάσει το **90%** και το eval το **57%** και δεν βελτιώνεται, καθώς από την 13η εποχή μέχρι την 17η το accuracy φτάνει μέχρι το **51%** και πάλι. Συνοπτικά, ακόμα παρουσιάζεται overfitting.

6.4.4 Τεχνικές Κλαδέματος

Κάποιες άλλες τεχνικές που μπορούν να χρησιμοποιηθούν για την αντιμετώπιση του overfitting είναι η μείωση του δικτύου, καθώς ο ήχος σε οδηγεί σε ενσωμάτωση λάθος πληροφοριών.

Οι τεχνικές “κλαδέματος” του δικτύου. Χωρίζονται σε τεχνικές pre-pruning και post-pruning.

pre-pruning: χρησιμοποιούν κριτήρια διακοπής για να καθοριστεί πότε πρέπει να σταματήσει η προσθήκη συνθηκών σε έναν κανόνα ή την προσθήκη ενός κανόνα σε μια περιγραφή μοντέλου.

post-pruning: χωρίζει το σετ εκπαίδευσης σε δύο υποσύνολα: σετ καλλιέργειας και σετ κλαδέματος. Σε σύγκριση με τους αλγόριθμους pre-pruning, αγνοούν τα προβλήματα του overfitting κατά τη διάρκεια της μαθησιακής διαδικασίας. Αντίθετα, αποτρέπουν την overfitting μέσω της διαγραφής συνθηκών και κανόνων από το μοντέλο που δημιουργούνται κατά τη διάρκεια της μάθησης. Αυτή η προσέγγιση είναι πολύ πιο ακριβής αλλά ωστόσο, λιγότερο αποτελεσματική.

Δεν χρησιμοποιούνται τεχνικές κλαδέματος καθώς δεν ταιριάζουν στην θεματική της παρούσας εργασίας.

6.4.5 Κανονικοποίηση

Regularization - Κανονικοποίηση: το αποτέλεσμα ενός μοντέλου επηρεάζεται από έναν μεγάλο αριθμό χαρακτηριστικών και όσο αυξάνεται ο αριθμός τους, τόσο πιο πολύπλοκο γίνεται το μοντέλο. Ένα μοντέλο που επηρεάζεται από overfitting, παίρνει όλα τα χαρακτηριστικά υπόψη του, παρόλο που κάποια από αυτά έχουν περιορισμένη ή και μηδενική επιρροή στο επιθυμητό αποτέλεσμα. Η κανονικοποίηση έχει σαν στόχο να επιλέξει μόνο τα χρήσιμα χαρακτηριστικά απορρίπτοντας τα άχρηστα και να ελαχιστοποιήσει τα βάρη από τα χαρακτηριστικά που δεν έχουν ιδιαίτερη επιρροή στη τελική διαδικασία ταξινόμησης. Αυτό γίνεται εφικτό με την χρήση ενός regularizer. Θα αναλυθούν **L1 regularization**, **L2 regularization** ή αλλιώς Weight Decay και η τεχνική του **Dropout**.

6.4.5.1 L1 regularization

Στο L1 regularization αφαιρούνται ορισμένα χαρακτηριστικά από το μοντέλο μας και διατηρούνται μόνο τα πιο πολύτιμα σε αυτό χαρακτηριστικά, χρησιμοποιώντας το Lasso Regression. Έτσι, το μοντέλο καταλήγει να είναι πιο απλό και πιο εύκολο να ερμηνευθεί. Ωστόσο, την ίδια στιγμή, χάνονται και ορισμένα χρήσιμα χαρακτηριστικά που έχουν μικρότερη επίδραση στην τελική έξοδο.

6.4.5.2 L2 regularization

Στο L2 regularization χρησιμοποιείται το Ridge Regression που χρησιμοποιεί την ευκλείδεια απόσταση. Σε σύγκριση με την τακτοποίηση L1, αυτή η προσέγγιση κάνει τα δίκτυα να προτιμούν να μαθαίνουν και χαρακτηριστικά με μικρά βάρη. Δηλαδή αντί να απορρίπτονται αυτά τα λιγότερο πολύτιμα χαρακτηριστικά, τους δίνονται χαμηλότερα βάρη.

Γίνεται λοιπόν συσσώρευση όσο το δυνατόν περισσότερων πληροφοριών, δίνοντας μεγάλα βάρη μόνο στα χαρακτηριστικά που θεωρούνται σημαντικά για την βελτίωση της συνάρτησης αρχικού κόστους.

6.4.5.3 Dropout

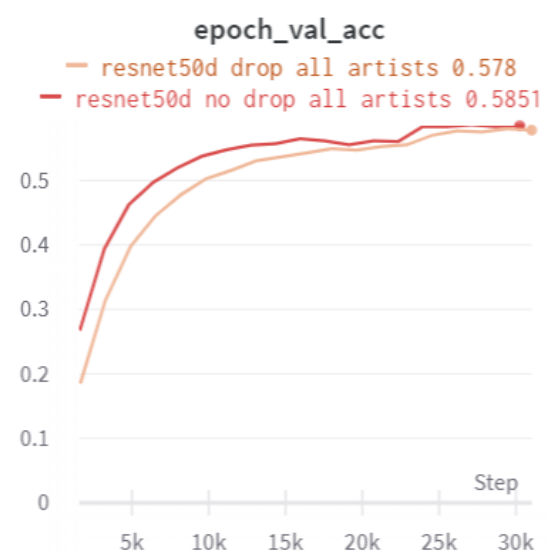
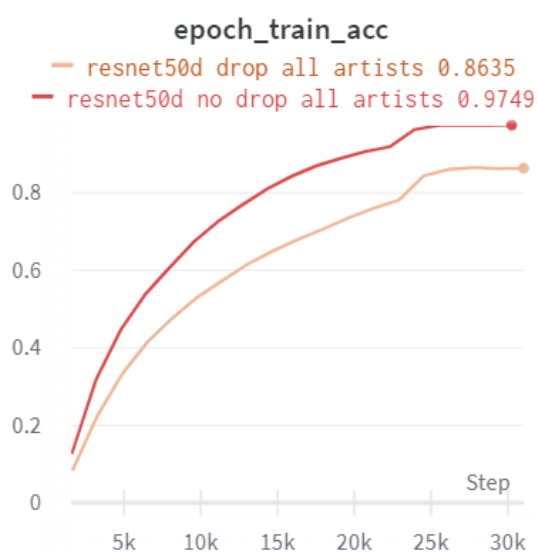
Καθώς το μοντέλο ακόμα αντιμετωπίζει overfit θα πραγματοποιηθεί ένα πείραμα με τη χρήση της τελευταίας τεχνικής Regularization που θα αναλυθεί, με τη χρήση dropout στο τελευταίο επίπεδο/layer της αρχιτεκτονικής που χρησιμοποιείται.

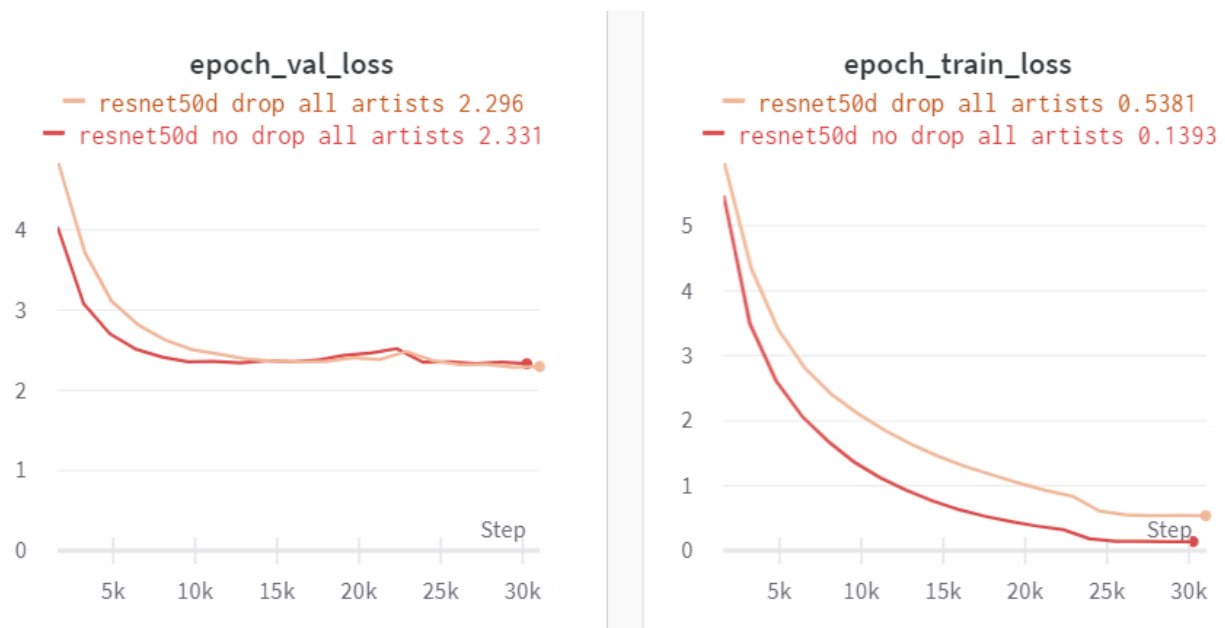
Γίνεται υπολογισμός πλέον των accuracies και για top-3 και top-5.

Πρόσθεση dropout.

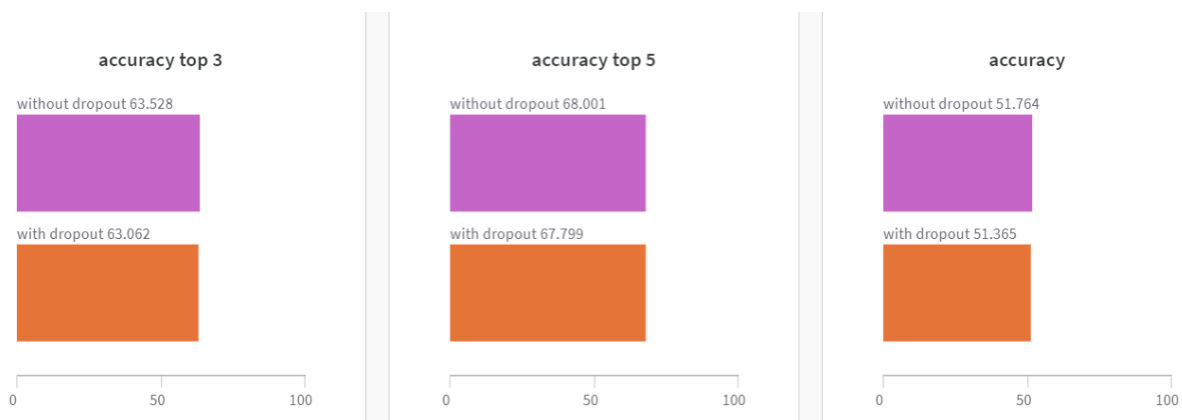
```
(CONFIG["model_conf"] == 'resnet50d'):\n\n    model.fc = nn.Sequential(\n\n        nn.Dropout(0.5),\n\n        nn.Linear(in_features=2048, out_features=2319))
```

Βλέποντας την διαδικασία εκπαίδευσης στο evaluation και στο training.





Αποτελέσματα με και χωρίς dropout:



Πάρθηκε η απόφαση να μην χρησιμοποιηθεί dropout στα επόμενα πειράματα.

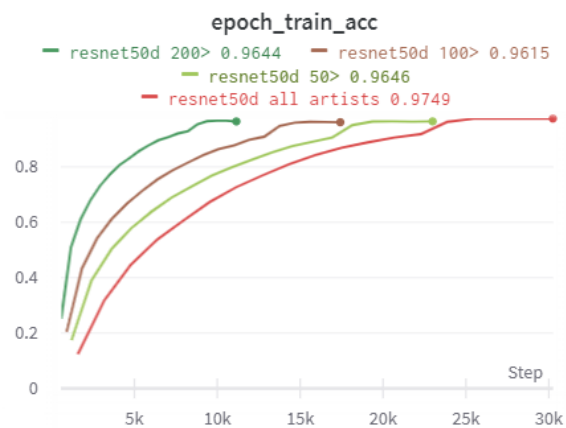
[19]

6.5 Προ-επεξεργασία δεδομένων

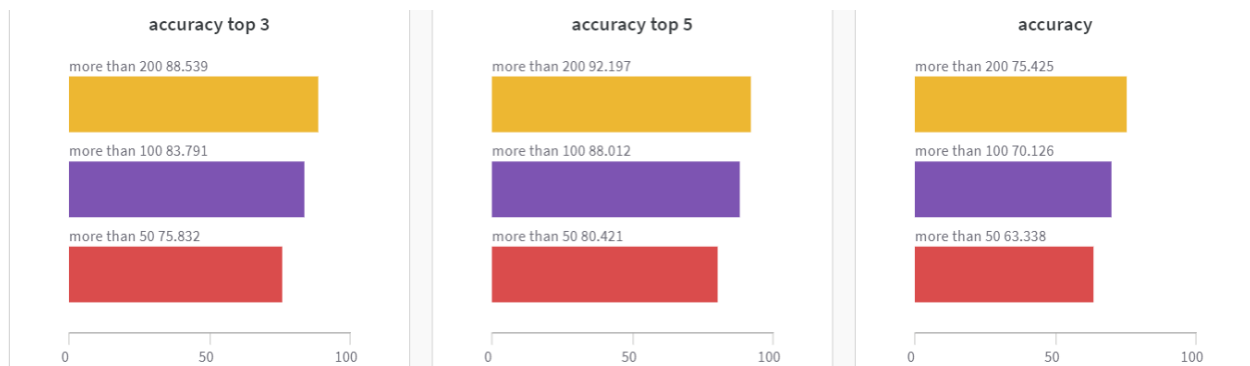
Στην αρχική του μορφή το dataset περιέχει 2319 καλλιτέχνες, οι οποίοι έχουν διαφορετικό αριθμό έργων ο καθένας. Αυτή η άνιση κατανομή πληροφορίες σε προβλήματα μηχανικής μάθησης μπορεί πολύ εύκολα να οδηγήσει σε κακή γενικοποίηση του συστήματος, καθώς δεν υπάρχει αρκετή πληροφορία για τον κατάλληλο διαχωρισμό του κάθε καλλιτέχνη. Η προεπεξεργασία των δεδομένων είναι από τα πιο σημαντικά βήματα για τη βελτίωση ενός μοντέλου. Στη συγκεκριμένη έρευνα καθώς ο σκοπός είναι μια γενική ιδέα των ομοιοτήτων μεταξύ μιας ζωγραφιάς του χρήστη και των καλλιτεχνών στη βάση δεδομένων, δεν αφαιρέθηκε κανένας καλλιτέχνης στα πειράματα μέχρι τώρα. Η κατανομή των έργων στους καλλιτέχνες όμως είναι άνιση στη συγκεκριμένη βάση δεδομένων και παρακάτω θα πραγματοποιηθούν πειράματα με τους καλλιτέχνες οι οποίοι έχουν πάνω από 50 έργα, πάνω από 100 έργα και τέλος πάνω από 200 έργα. Τα καλύτερα αποτελέσματα για το resnet50d χωρίς προεπεξεργασία να φτάνουν στο **51%** για top-1 και **68%** στο top-5

Καθαρίζοντας τα δεδομένα, ξεκινώντας υπήρχαν **2319** καλλιτέχνες , οι καλλιτέχνες που έχουν πάνω από 50 έργα είναι **510**, αυτοί που είχαν πάνω 100 είναι **255** και τέλος αυτοί που έχουν πάνω από 200 είναι **108**. Παρακάτω η διαδικασία εκπαίδευσης.





Μπορεί να παρατηρηθεί ότι με κάθε μείωση των καλλιτεχνών που θα δημιουργήσουν τις κλάσεις στην εκπαίδευση κρατώντας αυτούς που έχουν τα περισσότερα δεδομένα, διαφοροποιείται αισθητά και το αποτέλεσμα. Παρατηρείται μία σταθερή βελτίωση της εκπαίδευσης στο evaluation set με την μείωση των καλλιτεχνών, καθώς το μοντέλο μπορεί να γενικοποιήσει καλύτερα τα δεδομένα, μετριάζοντας τις επιπτώσεις του overfitting.



6.6 Τελικά συμπεράσματα

Τέλος, με τον μικρότερο αριθμό κλάσεων και με μεγαλύτερο αριθμό δεδομένων για κάθε κλάση το φαινόμενο του overfit μειώνεται δραστικά. Για τα αποτελέσματα για top-1, top-3 και top-5 υπάρχει μία αισθητή βελτίωση κάθε φορά που μικραίνει το dataset και κρατιούνται μόνο οι καλλιτέχνες που έχουν όλο και περισσότερα έργα, με το top-5 accuracy για τους καλλιτέχνες με

πάνω από 200 έργα να φτάνει σε ένα **92%**. Άξιο αναφοράς είναι ότι στην εφαρμογή που δημιουργείται, καθώς ο σκοπός είναι να υπάρχουν αρκετοί καλλιτέχνες με τους οποίους ίσως υπάρχουν ομοιότητες, θα πρέπει να γίνουν κάποιες δοκιμές για το αν με λιγότερους καλλιτέχνες βελτιώνεται ή χειροτερεύει η ικανότητα της εφαρμογής να βρει καλλιτέχνες που μοιάζουν με τα έργα του χρήστη. Αλλά αυτό δεν αποτελεί μέρος της διαδικασίας της εκπαίδευσης.

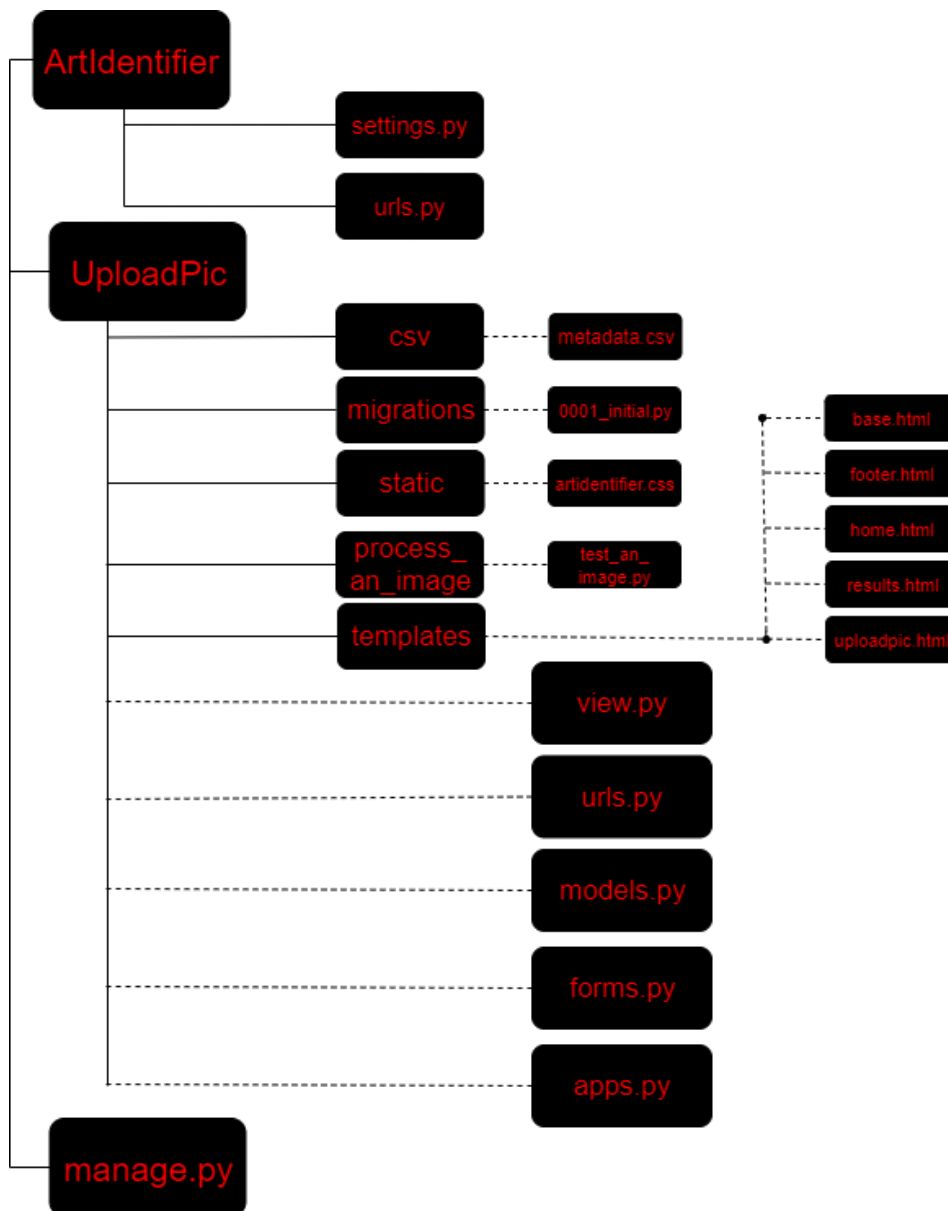
7ο Δημιουργία και χρήση Web Εφαρμογής

Για την δημιουργία της πλατφορμας που θα χρησιμοποιηθεί για την χρήση του Art Identifier, χρησιμοποιήθηκε το Django. Το Django είναι μια πολύ διαδεδομένη πλατφόρμα για την δημιουργία ιστοσελίδων και χρησιμοποιεί τη γλώσσα προγραμματισμού python. Μεγάλο ρόλο στη λήψη της απόφασης για την χρήση του Django, πέρα από τα έτοιμα χαρακτηριστικά που κάνουν κάποιες πτυχές της ανάπτυξης πιο απλές, διαδραμάτισε σίγουρα και η λειτουργικότητα στο περιβάλλον της python, μία γλώσσα που έχει συνοδεύσει σε έναν μεγάλο βαθμό την αναγέννηση που έχει έρθει τα τελευταία χρόνια στον τομέα της μηχανικής μάθησης, με αποτέλεσμα αφθονία από βιβλιοθήκες μηχανικής μάθησης να είναι διαθέσιμες στην python. Μία πολύ σημαντική λεπτομέρεια είναι επίσης ο μεγάλος αριθμός ανθρώπων που χρησιμοποιούν και την Python και το Django, καθώς έχει δημιουργηθεί μία πολύ καλή ατμόσφαιρα για καινούργιους χρήστες, δίνοντας την δυνατότητα να βρεθούν πληροφορίες για πληθώρα προβλημάτων στο διαδίκτυο χωρίς καμία οικονομική επιβάρυνση.

Όταν δημιουργήθηκε το Django στην αμερική το φθινόπωρο του 2003, οι Adrian Holovaty και Simon Willison άρχισαν να χρησιμοποιούν python με βασικό στόχο τους τον περιορισμό στην χρήση κώδικα που επαναλαμβάνεται. Το Django ακόμα και σήμερα δίνει μεγάλη έμφαση στον open-source χαρακτήρα του, για να μπορέσουν οι χρήστες να προσθέτουν και δικές του λειτουργικότητες στο οπλοστάσιο της πλατφόρμας, με την python να υπάρχει σε όλες τις όψεις της ανάπτυξης της ιστοσελίδας, από τα μοντέλα δεδομένων μέχρι και τις ρυθμίσεις. Πολύ χρήσιμη χαρακτηρίζεται και η έτοιμη διαχειριστική διεπαφή που προσφέρεται αυτόματα και χρησιμοποιείται για τον έλεγχο της βάσης δεδομένων, έχοντας τη δυνατότητα να δημιουργήσει, να ενημερώσει και να διαγράψει δεδομένα ανάλογα με τις επιλογές του διαχειριστή.

Ένα χαρακτηριστικό του Django που μπερδεύει πολλούς χρήστες είναι ο χαρακτηρισμός του ως MTV(Model, Template, View) η αλλιώς ένα αρχιτεκτονικό μοτίβο μοντέλου-προτύπου-προβολών που μοιάζει με αρχιτεκτονική του MVC. Η αρχιτεκτονική MVC(Model, View, Controller) είναι ένα μοτίβο αρχιτεκτονικής σχεδίασης λογισμικού για τη διαίρεση των λειτουργιών εφαρμογής και της λογικής του προγράμματος σε ξεχωριστά στοιχεία και μπορεί αλλιώς να χαρακτηριστεί ως αρχιτεκτονική μοντέλου-προβολών-ελεγκτή . Αρχικά, αυτή η αρχιτεκτονική χρησιμοποιήθηκε για τον σχεδιασμό διεπαφών χρήστη για επιτραπέζιους υπολογιστές (GUIs) αλλά σήμερα χρησιμοποιείται ευρέως στον σχεδιασμό εφαρμογών ιστού.

Παρακάτω ένα σχήμα με τα αρχεία τα οποία δημιουργήθηκαν για την ιστοσελίδα.



Σχήμα 2

Ο φάκελος **UploadPic** περιέχει τις κύριες λειτουργίες της ιστοσελίδας.

Στο **urls.py** καθορίζεται το όνομα της εφαρμογής και συνδέονται τα views με τα κατάλληλα urls.

```
from django.urls import path, include
from . import views
from django.conf import settings
from django.conf.urls.static import static
from .views import image_upload_view
from .views import results
app_name = 'uploadpic'
urlpatterns = [
    path('', views.index, name='index'),
    path('uploadpic/', views.image_upload_view, name="upload_pic"),
    path('results/', views.results, name="results"),
]
if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_URL)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Code snippet: urls.py

Το μοντέλο MTV που πλέον χρησιμοποιείται στη πλατφόρμα, όπως προαναφέρθηκε, έχει από λειτουργική άποψη πολλά κοινά με το μοντέλο MVC, χρησιμοποιώντας όμως διαφορετική ορολογία και βασίζεται στα τρία αυτά στοιχεία που θα αναλυθούν παρακάτω.

Μοντέλα: Τα μοντέλα(models) αποτελούν τον οδηγό για τη δομή των δεδομένων, δηλαδή την μορφή την οποία όλες οι πληροφορίες θα πρέπει να έχουν όταν αποθηκευτούν στην βάση δεδομένων και ορίζουν το πως δομείται η βάση δεδομένων της εφαρμογής.

Για τη δημιουργία ενός μοντέλου μπορεί να χρησιμοποιηθεί πληθώρα από βάσεις, με μερικά παραδείγματα να είναι η MariaDB, MySQL, Oracle και η SQLite η οποία χρησιμοποιείται και στην παρούσα πτυχιακή. Είναι δυνατή η δημιουργία πολλών βάσεων δεδομένων ανάλογα με τις ανάγκες της κάθε εφαρμογής.

Με τη χρήση του Django οι χρήστες δεν χρειάζεται να αλληλεπιδράσουν καθόλου με τη βάση δεδομένων μέσω της γλώσσας SQL για τη δημιουργία και τον χειρισμό των βάσεων δεδομένων.

Όπως βλέπουμε και στο κομμάτι κώδικα παρακάτω, με αυτόν τον τρόπο ορίζεται στο Django η μορφή της βάσης, ως κλάση της γλώσσας python και στη συνέχεια η πλατφόρμα μετατρέπει αυτές τις κλάσεις σε πίνακες των βάσεων όπως έχουν οριστεί από τον χρήστη στο αρχείο ρυθμίσεων της εφαρμογής.

```
from django.db import models

# Create your models here.
class Artist(models.Model):
    name = models.CharField(max_length=200, default="")
    img = models.ImageField(upload_to="images", default="")

    def __str__(self):
        return self.name
```

Code snippet: Models.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Code snippet: settings.py

Υπάρχει επίσης η δυνατότητα για την επιλογή των μορφών σχέσεων που οι συμβατικές SQL βάσεις προσφέρουν, όπως σχέσεις ένα-προς-πολλά (one-to-many relations) ή πολλά-προς-πολλά (many-to-many relations). Αλλά επειδή οι ανάγκες της συγκεκριμένης εφαρμογής είναι ένα όνομα της ζωγραφιάς του χρήστη, καθώς και ένα αρχείο εικόνας με την ζωγραφιά που θα περάσει μέσα από το εκπαιδευμένο μοντέλο, μια σχέση μια-προς-μια (one-to-one) είναι αρκετή.

Το ModelForm είναι μια κλάση που χρησιμοποιείται για την απευθείας μετατροπή ενός μοντέλου σε φόρμα Django.

```
# import form class from django

from django import forms

from django.db import models #now

from django.forms import fields #now

# import Artist from models.py

from .models import Artist

# create a ModelForm

class ArtistForm(forms.ModelForm):
```

```
# specify the name of model to use

class Meta:

    model = Artist

    fields = '__all__'

class QueryForm(forms.Form):

    item = forms.ModelChoiceField(

        widget=forms.Select,

        queryset=Artist.objects.all()

    )
```

Code snippet: forms.py

Όψεις: Οι όψεις ή αλλιώς views διαπραγματεύονται την αποστολή, επεξεργασία και αποθήκευση των δεδομένων από την βάση δεδομένων, μπορεί να χαρακτηριστεί πιο απλοϊκά και ως η διεπαφή επικοινωνίας ενός χρήστη με το σύστημα με την βοήθεια μιας αίτησης (request) και την επιστροφή μιας απάντησης (response).

Στην αρχική του μορφή το Django, πρόσφερε την διεργασία όψης (function view) για κάθε όψη της εφαρμογής που χαρακτηρίζεται ως μορφή των όψεων βασισμένη σε διεργασίες (function-based views). Αλλά νωρίς στην εξέλιξη του οι δημιουργοί και οι χρήστες συνειδητοποίησαν ότι αυτή η μορφή έχει κάποια έμφυτα μειονεκτήματα, με το βασικότερο το γεγονός ότι δεν υπάρχει η δυνατότητα να μην χρειάζεται η επανάληψη κώδικα για λειτουργίες που ήταν σταθερές σε κάθε όψη, κάτι το οποίο πήγαινε ενάντια στην αρχική λογική και ιδεολογία πίσω από την δημιουργία της πλατφόρμας.

Η λύση για αυτό το πρόβλημα ήταν η δημιουργία των όψεων βασισμένων σε διεργασίες (function-based generic views) με την κύρια διαφορά του να είναι ότι ορισμένες μέθοδοι οι οποίες είναι κοινές ανάμεσα στις όψεις και σε διαφορετικές εφαρμογές, πλέον παρέχονται έτοιμες.

Στη συνέχεια η ανάπτυξη της Django έδωσε στους χρήστες και την επιλογή των όψεων βασισμένες στις κλάσεις (Class Based Views, CBV), ώστε να δίνεται στον χρήστη η δυνατότητα της κληρονομικότητας και κατα συνέπεια η δυνατότητα της δημιουργίας συνδυαστικών όψεων. Στον πυρήνα τους, τα CBV είναι κλάσεις Python. Το Django συνοδεύεται από μια ποικιλία από CBV "πρότυπα" που έχουν προρυθμισμένες λειτουργίες που μπορούν να επαναχρησιμοποιηθούν και πολλές φορές να επεκταθούν. Σε αυτές τις όψεις δίνονται στη συνέχεια χρήσιμα ονόματα που περιγράφουν το είδος της λειτουργικότητας που παρέχουν. Αναφέρονται ως "γενικές προβολές" (Generic Views), επειδή παρέχουν λύσεις σε κοινές απαιτήσεις. Οι όψεις αυτές

διαθέτουν τεκμηρίωση στον ιστότοπο του Django που δείχνει κάθε λειτουργικότητα που προσφέρεται, ποιες ρυθμίσεις απαιτούνται ή είναι δυνατές και πώς να επεκταθούν.

Παρακάτω τα 3 views που χρησιμοποιούνται, η index η οποία δείχνει στο Home Page, η results η οποία είτε αφού ανεβάσει ο χρήστης μια νέα ζωγραφιά ή επιλέξει μια απο τις προηγούμενες ζωγραφιές που έχει ανεβάσει τον προωθεί στην σελίδα με τα αποτελέσματα και τέλος η image_upload_view η οποία είναι υπεύθυνη για τη διαδικασία του να ανεβάσει ο χρήστης την ζωγραφιά του.

```
from django.shortcuts import render
from django.http import HttpResponseRedirect
from .models import Artist
from django.shortcuts import render, redirect
from .forms import ArtistForm, QueryForm
from .process_image.test_an_image import results_pic

def index(request):
    return render(request, 'home.html')

def results(request):
    if request.method == 'POST':
        form = ArtistForm(request.POST, request.FILES)

        if form.is_valid():
            form.save()

            # Get the current instance object to display in the template
            img_object = form.instance

            res = results_pic(img_object.img)

            return render(request, 'results.html', {'img_obj': img_object, 'res':
res})

    if request.method == 'GET':
        if len(request.GET) > 0:
            form = QueryForm(request.GET)

            if form.is_valid():
                img_object = Artist.objects.filter(id=int(request.GET['item']))[0]

                img = getattr(img_object, 'img')
```

```

        res = results_pic(img)

        return render(request, 'results.html', {'img': img, 'res': res})

    else:

        print(form)

    form = QueryForm()

    return render(request, 'results.html', {'form': form})

def image_upload_view(request):

    """Process images uploaded by users"""

    form = ArtistForm()

    return render(request, 'uploadpic.html', {'form': form})

```

Code snippet: views.py

Πρότυπα: Τα πρότυπα ή αλλιώς Templates πραγματεύονται την παραγωγή του αρχείου HTML το οποίο θα αντικρίσει ο χρήστης όταν περιηγείται στο διαδίκτυο, δηλαδή την ιστοσελίδα που θα βλέπει ο τελικός χρήστης.

Το τελικό στάδιο απαιτεί να παρέχεται στον χρήστη μια απάντηση HTML (HyperText Markup Language) ώστε να μπορεί να ερμηνευθεί από το πρόγραμμα περιήγησης του χρήστη αφού η σχετική όψη συγκεντρώσει και επεξεργαστεί τα δεδομένα που απαιτούνται για την ικανοποίηση του αιτήματος του. Η HTML είναι μια γλώσσα σήμανσης που περιέχει τα μέρη που συνθέτουν μια ιστοσελίδα. Στο διαδίκτυο, είναι πλέον με διαφορά η πιο ευρέως χρησιμοποιούμενη γλώσσα σήμανσης. Τα σύμβολα “λιγότερο από <” και “μεγαλύτερο από >” χρησιμοποιούνται για να οριοθετήσουν τα πολυάριθμα διακριτικά στοιχεία της ιστοσελίδας, τα οποία χαρακτηρίζονται από ένα σύστημα ετικετών. Αυτά τα σύμβολα δεν εμφανίζονται ως έχουν, αλλά αντίθετα διαβάζονται από το πρόγραμμα περιήγησης προκειμένου το περιεχόμενό τους να αποτυπώνεται κατάλληλα στην οθόνη του τελικού χρήστη. Ένα παράδειγμα σε HTML της αρχικής και πιο απλοϊκής μορφής της πρώτης εικόνας που βλέπει ο χρήστης, βγάζει σαν αποτέλεσμα στο χρήστη την λέξη “HOME PAGE” και ένα link το οποίο προωθεί τον χρήστη στην εφαρμογή uploadpic:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        body {background-color: #c7000f;}
        h1 {
            font-family: "Times New Roman";
            font-size: 200%
        }
        h2 {
            font-size:150%
        }
    </style>

```

```

    <meta charset="UTF-8">
    <title>Home Page</title>
</head>
<body>
<h1 style="text-align:center;">
    HOME PAGE
</h1>
<h2 style="text-align:center;">
<a href="http://127.0.0.1:8000/uploadpic">This is a link</a>
</h2>
</body>
</html>

```

Code snippet: home.html

Το Django παρέχει γλώσσα μορφοποίησης των προτύπων, η οποία επιτρέπει επίσης στον δημιουργό να χρησιμοποιήσει δεδομένα τα οποία είναι αποθηκευμένα χωρίς την χρήση της JavaScript, καθώς και την δυνατότητα εφαρμογής κάποιων τεχνικών από γλώσσες προγραμματισμού μέσα στο αρχείο HTML. Παραδείγματα τέτοιων λειτουργιών είναι ο έλεγχος (if, if else) και βρόγχοι επανάληψης (for). Στο παράδειγμα παρακάτω θέλουμε να εμφανίσουμε τους 5 καλλιτέχνες που είναι πιο κοντά στην ζωγραφιά που ανέβασε ο χρήστης, αυτές οι πληροφορίες βρίσκονται αποθηκευμένες και γίνεται η ανάκτηση τους μέσα στην HTML:

```

{% if res %}
    <h3 style="text-align:center;">The closest matches</h3>
    <table class="center">
        <thead>
            <tr>
                <th>artists</th>
            </tr>
        </thead>
        <tbody align="center">
            {% for name in res %}
                <tr>
                    <td style="text-align: center; vertical-align:
middle;">{{name}}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endif %}

```

Code snippet: uploadpic.html

Αυτή η ικανότητα της Django να δημιουργηθούν δυναμικές ιστοσελίδες μέσω των προτύπων ονομάζεται Django Template Language (DTL) και ενώ είναι πολύ εύχρηστη λόγω της απλότητας της έχει μερικούς περιορισμούς και για πιο περίπλοκες και συγκεκριμένες διαδικασίες θα χρειαστεί και πάλι κώδικας JavaScript είτε εμφωλευμένος μέσα στην HTML είτε σε ξεχωριστό αρχείο.

Ο φάκελος template είναι η εμφάνιση της ιστοσελίδας και περιέχει τα αρχεία html. Η λειτουργικότητα του κάθε αρχείου μπορεί να γίνει κατανοητή από τα ονόματα. Σε κάθε ένα από

τα αρχεία, το **base.html** εισάγεται στη αρχή και το **footer.html** στο τέλος. Το **home.html** αποτελεί την πρώτη όψη που αντικρίζει ο χρήστης με κάποιες πληροφορίες για την παρούσα πτυχιακή, το **uploadpic.html** δίνει την δυνατότητα στον χρήστη να ανεβάσει το έργο του και αυτόματα τον προωθεί στα αποτελέσματα και τέλος στο **results.html** μπορεί να επιλέξει οποιαδήποτε από τις ζωγραφιές που έχει ανεβάσει και να ξαναδεί τα αποτελέσματα του.

Ξεκινώντας με την λογική του Django, οι λειτουργίες κάθε εφαρμογής μπορούν να δημιουργηθούν ξεχωριστά. Αυτή η λειτουργικότητα συμφωνεί και με τον στόχο του django, καθώς εφαρμογές (apps) που δημιουργήθηκαν σε ένα project μπορούν να μεταφερθούν σε ένα άλλο, αν ταιριάζουν στις ανάγκες του. Για να χρησιμοποιηθεί αυτή η λειτουργικότητα, υπάρχει ο φάκελος της κύριας εφαρμογής **ArtIdentifier**, η οποία συνδέει τις υπόλοιπες εφαρμογές και στην συγκεκριμένη περίπτωση δείχνει σε μία, την **UploadPic**.

Στον φάκελο csv έχει μέσα το **metadata.csv** αρχείο το οποίο περιέχει τις πληροφορίες που συνοδεύουν τις φωτογραφίες.

Ο φάκελος migrate περιέχει το αρχείο **0001_initial.py**, τα migrations είναι ο τρόπος του Django να μεταδίδει τις αλλαγές που συμβαίνουν στα μοντέλα της βάσης δεδομένων.

```
class Migration(migrations.Migration):

    initial = True

    dependencies = []

    operations = [

        migrations.CreateModel( name='Artist', fields=[

            ('id', models.BigAutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),

            ('name', models.CharField(default='', max_length=200,
unique=True)),

            ('img', models.ImageField(default='', upload_to='images')),],),]
```

Code snippet: 0001_initial.py

Στον φάκελο static υπάρχει το **artidentifier.css**, το οποίο χαρακτηρίζει την αισθητική που θα υπάρχει στην ιστοσελίδα .

```
body {background-color: #fff; margin: 0; padding: 0;}

h1 {font: italic small-caps bold 12px/30px Apple Chancery, cursive;

    font-family: 'Monotype Corsiva', 'Apple Chancery', 'ITC Zapf Chancery', 'URW
Chancery L', cursive;
```

```

    font-size: 300%;
    color: white;}
h2 {font-size:150%}
/* Add a black background color to the top navigation */
.topnav {background-color: #333;overflow: hidden;}
.topbar {
    width: 100%;
    height: 100%;
    padding:0;
    margin: 0;
    overflow: auto;
    background-image: linear-gradient(to right, #9a0202 , #014931);
}
/* Style the links inside the navigation bar */
.topnav a {
    float: left;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}
/* Change the color of links on hover */
.topnav a:hover {
    background-color: #ddd;
    color: black;
}
/* Add a color to the active/current link */
.topnav a.active {

```

```

background-color: #04AA6D;
color: white;
}

.container {
padding: 20px 40px 40px;
display: flex;
}

.container__image {
margin: 20px 30px 20px 20px;
object-fit: cover;
float: left;
width: 45%;
}

.container__text {
margin: 20px 30px 20px 20px;
width: 45%;
float:right;
}

.sideline {
width: 100%;
text-align: center;
border-bottom: 1px solid #4CAF50;
line-height: 0.1em;
margin: 10px 0 20px;
}

.sideline span {
background:#fff;
color:#4CAF50;
padding:0 10px;

```

```

}

.center {
    margin-left: auto;
    margin-right: auto;
}

table, th, tr, td {
    border: 1px solid black;
    border-color: #04AA6D;
    border-collapse: collapse;
}

th {
    border: 1px solid white;
    background-color: #04AA6D;
    color: white;
}

input[type=submit] {
    background-color: white; /* Green */
    border: 2px solid black;
    border-color: #4CAF50;
    color: #4CAF50;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    border-radius: 12%;
}

.frm-lbl {
    background-color: #333;
    border: none;

```

```

    color: white;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
}

div.ftr {
    display: table;
    width: 100%;
    table-layout: fixed;    /* For cells of equal size */
}

div.ftr span {
    display: table-cell;
    text-align: center;
}

footer {
    background-image: linear-gradient(to right, #9a0202 , #014931);
    box-shadow: 0 50vh 0 50vh #06BA7D;
    color: white;
    position: fixed;
    padding: 10px 10px 0px 10px;
    bottom: 0;
}

.icon {
    background-image:
url(https://static.vecteezy.com/system/resources/previews/004/261/121/original/illustration-of-brush-paint-icon-free-free-vector.jpg);
    height: 10px;
    width: 10px;
    background-position: center;
    background-repeat: no-repeat;
}

```

```
background-size: contain;
}
```

Code snippet: artidentifier.css

Στον φάκελο process_image, το test_an_image.py είναι το αρχείο που περιέχει τις βασικές λειτουργίες της εφαρμογής. Ακολουθείται παρόμοια διαδικασία όπως και στο train.py, διαβάζοντας τα δεδομένα και το csv, κάνοντας encode τους καλλιτέχνες, αρχικοποιώντας το μοντέλο και τα transforms. Με την χρήση του PIL, η ζωγραφιά που έχει δώσει ο χρήστης “περνάει” μέσα από το μοντέλο και παρέχει ως απάντηση, μετά την αντιστροφή του encoder, τους 5 πρώτους καλλιτέχνες σύμφωνα με το μοντέλο.

```
import torch
import timm
import torch
import os
import numpy as np
import pandas as pd
from torch import nn
from PIL import Image
from sklearn.preprocessing import LabelEncoder
from torchvision import transforms

def results_pic(pic):
    meta_path = os.path.join("uploadpic", "csv", "metadata.csv")
    metadata = pd.read_csv(meta_path)
    enc = LabelEncoder()
    metadata['artist_cat'] = enc.fit_transform(metadata['artist'].astype(str))
    model = timm.create_model(model_name='resnet50', pretrained=True)
    last_fc = nn.Linear(in_features=2048, out_features=2319)
    model.fc = last_fc

    if torch.cuda.is_available():
```

```

        model.load_state_dict(torch.load(r'model.pth'))
    else:
        model.load_state_dict(torch.load(r'model.pth',map_location=torch.device('cpu'))))

    model.eval()

    transform = transforms.Compose([

        transforms.Resize(256),
        transforms.ToTensor(),
        transforms.Normalize(
            mean=[0.485, 0.456, 0.406],
            std=[0.229, 0.224, 0.225] )

    ])

    img = Image.open(pic).convert('RGB') # Load image as PIL.Image
    img_t = transform(img)
    batch_t = torch.unsqueeze(img_t, 0)
    out = model(batch_t)
    out = torch.softmax(out, dim=1).detach().numpy()
    preds = np.argmax(out, -1)[0][-5:]
    finalpreds = enc.inverse_transform(preds).tolist()[::-1]
    return finalpreds

```

Code snippet: test_an_image.py

Μέσα στο **ArtIdentifier**, το **settings.py** περιέχει τις επιθυμητές ρυθμίσεις για διάφορες λειτουργίες της εφαρμογής, από την ασφάλεια, το database, της εφαρμογής και τέλος για το κατέβασμα και την σωστή αποθήκευση των αρχείων που θα ανεβάζουν οι χρήστες.

```

# Base url to serve media files
MEDIA_URL = '/media/'

# Path where media is stored
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

```

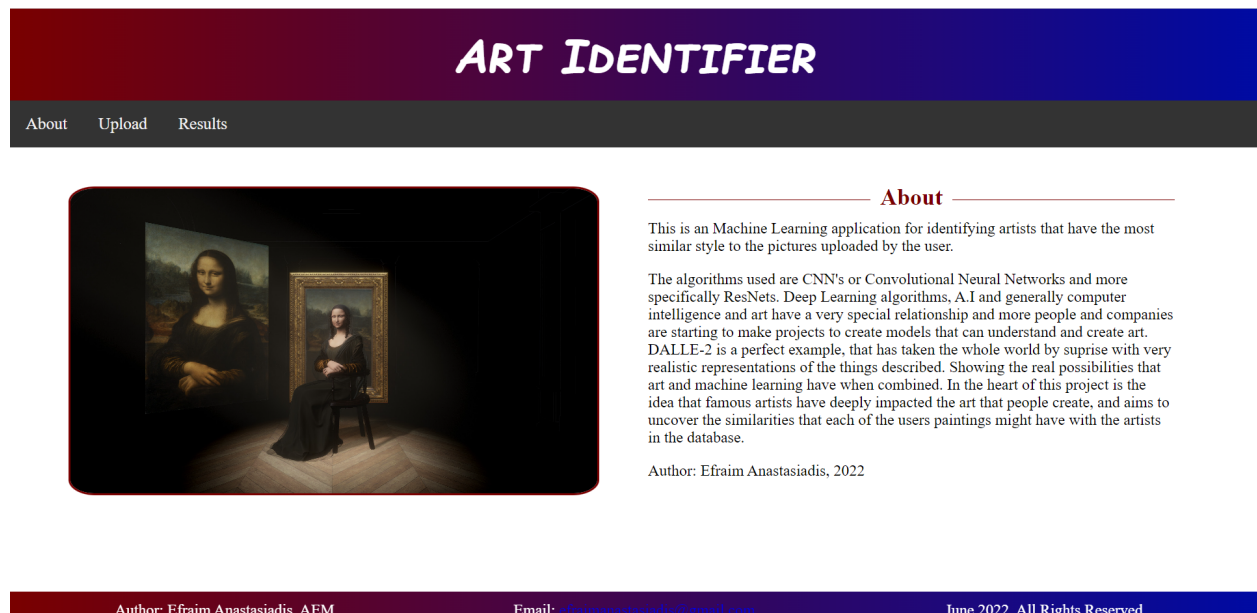
Code snippet: settings.py

Στο **urls.py**, προωθούνται 2 paths. Το ένα παίρνει τα urls απο το urls της εφαρμογής uploadpic και το άλλο το έτοιμο admin site που υπάρχει στο django.

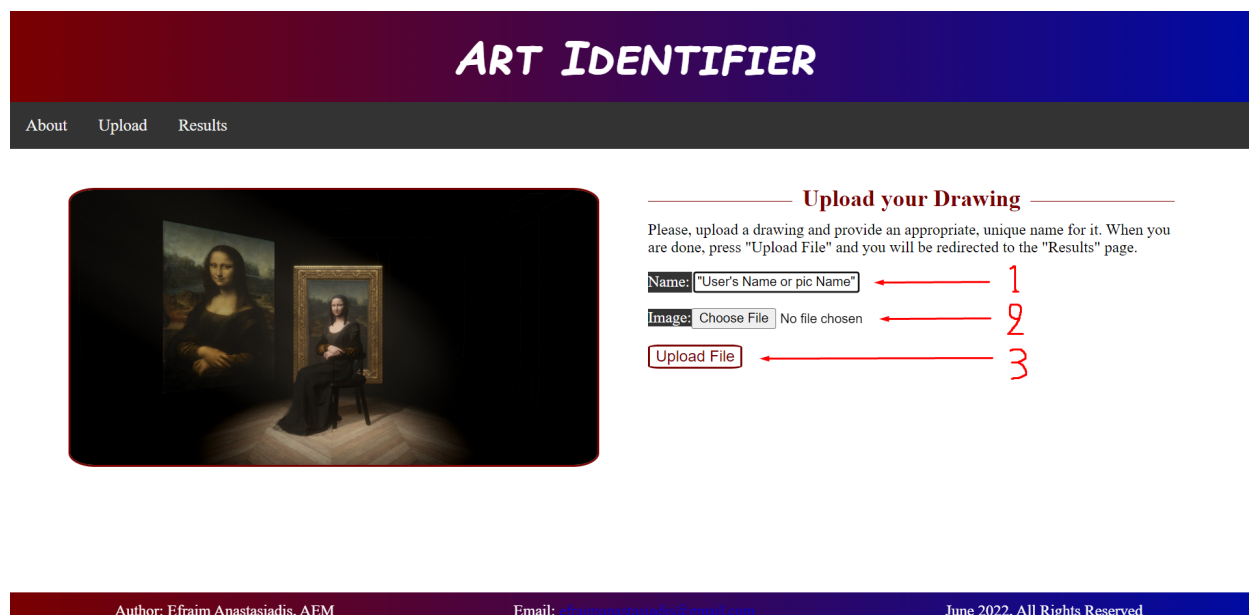
```
urlpatterns = [  
    path('', include('uploadpic.urls')),  
    path('admin/', admin.site.urls),]
```

Code snippet: urls.py

Η πρώτη σελίδα που αντικρίζει ο χρήστης είναι το Home Page.

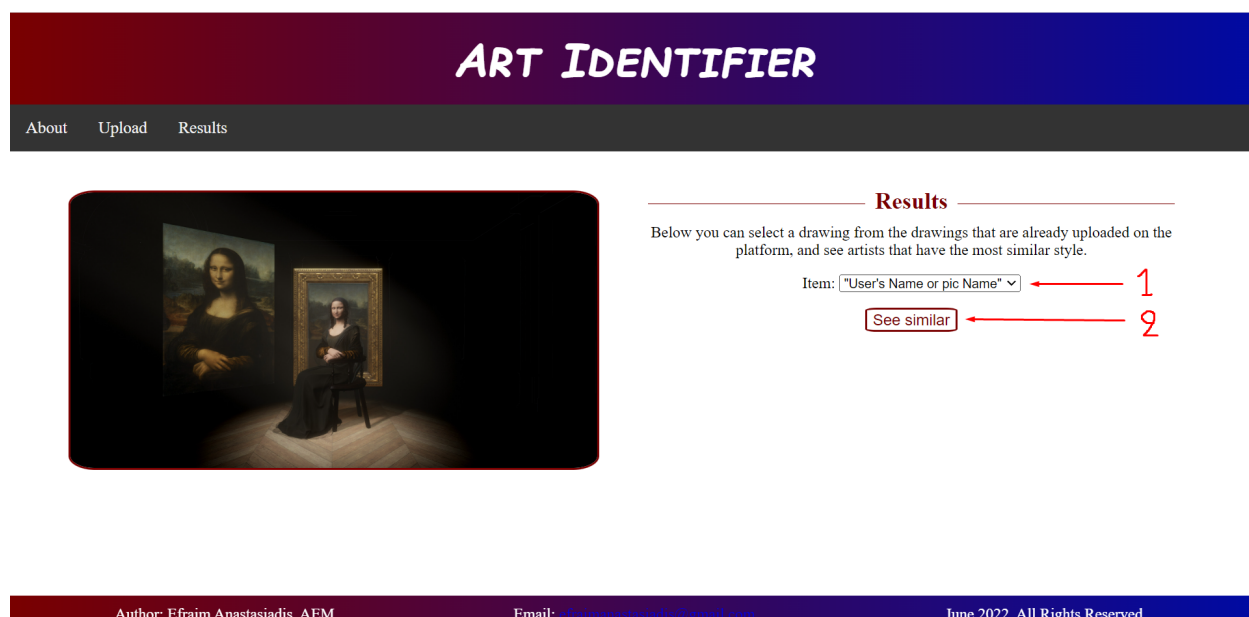


Στη συνέχεια όταν πατάει στο Upload στο **1.** ο χρήστης βάζει είτε το όνομα του είτε το όνομα της ζωγραφιάς του. Στο **2.** επιλέγει μια ζωγραφιά απο τα local αρχεία του και πατώντας στο **3.** πηγαίνει στη σελίδα αποτελεσμάτων.



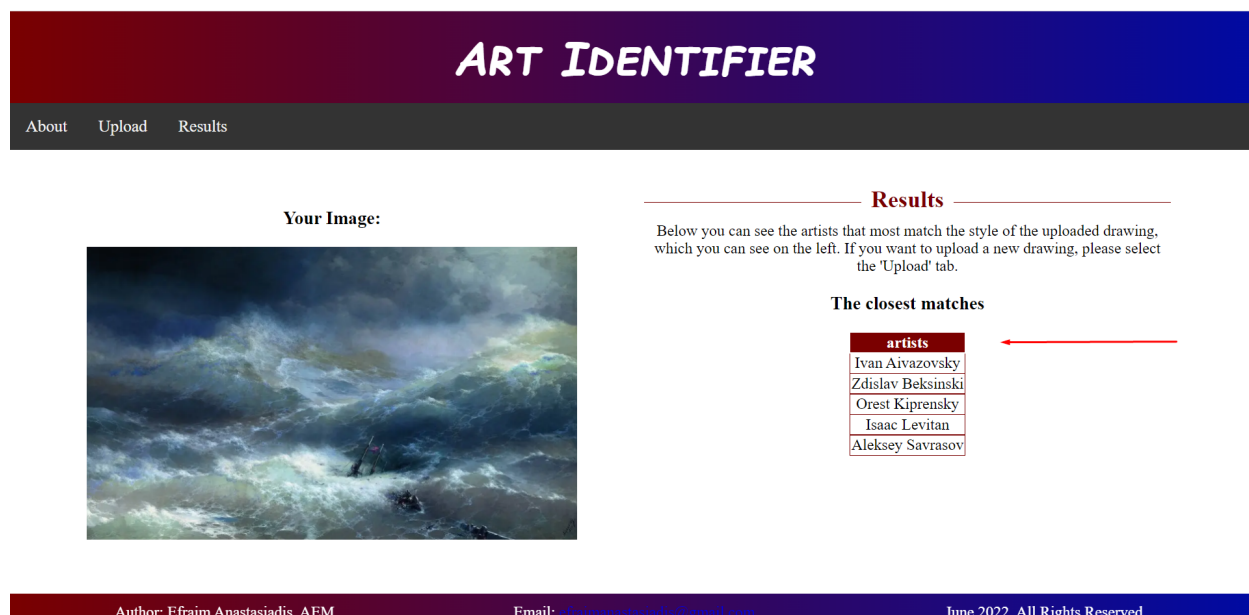
Σχήμα 3

Πηγαίνοντας στο Results, στο 1. μπορεί να επιλέξει το όνομα που πρόσθεσε και να δει τα αποτελέσματα στο 2. πατώντας στο see similar.



Σχήμα 4

Τέλος εμφανίζονται τα αποτελέσματα,



Σχήμα 5

8^ο Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η διαδικασία της εκπαίδευσης ενός προγράμματος για την αναγνώριση καλλιτέχνη έχει πολλούς περιορισμούς κάνοντας την επιλογή της αρχιτεκτονικής του μοντέλου μία αρκετά δύσκολη διαδικασία.

Τα μεγαλύτερα προβλήματα που αντιμετωπίστηκαν προέρχονταν από την περιορισμένη υπολογιστική δύναμη, χρησιμοποιώντας μία κάρτα γραφικών (GPU) RTX2060. Αυτό οδήγησε σε περιορισμό στον αριθμό των πειραμάτων, κάτι το οποίο λύθηκε με την ενσωμάτωση μίας λειτουργίας στον κώδικα που επιτρέπει την τυχαία μείωση του αριθμού των εικόνων, κάτι που οδηγεί με την σειρά του σε μικρότερο χρόνο που χρειάζεται για την κάθε εποχή της επανάλληψης.

Σε όλη την διάρκεια της δημιουργίας αυτής της ιστοσελίδας και προπονώντας το μοντέλο, αυτό που έκανε την μεγαλύτερη θετική εντύπωση είναι η τεράστια ποσότητα δεδομένων και πληροφοριών, με πολλά χρήσιμα εργαλεία και πληθώρα δεδομένων να συνεχίζουν να μεταφέρονται σε ψηφιακή μορφή. Δημιουργώντας ένα περιβάλλον το οποίο προωθεί όλο τον τομέα και κάνει πιο κατανοητή την ραγδαία εξέλιξη του τα τελευταία χρόνια.

Όπως σε κάθε πρόβλημα εκπαίδευσης, η διαδικασία συλλογής δεδομένων με σωστές ταμπέλες είναι από τα πιο σημαντικά μέρη, κάτι το οποίο κάνει απόλυτα κατανοητή την αισθητή επίδραση που είχε το ImageNet. Βλέποντας την νέα τάση ψηφιοποίησης της τέχνης και γενικά της ζωής σε

κάθε επίπεδο, ίσως γίνει δυνατή η συνέχεια της εκπαίδευσης είτε με περισσότερους καλλιτέχνες και έργα, είτε με την δημιουργία μιας βάσης δεδομένων εξειδικευμένη σε έργα τέχνης. Το μέλλον σε αυτόν τον τομέα φαίνεται να έχει πάρα πολύ μεγάλες δυνατότητες και με τον ρυθμό που αναπτύσσεται, θα φέρει σίγουρα μεγάλες αναταράξεις σε πάρα πολλούς άλλους τομείς. Όταν αρχίζουν οι υπολογιστές να παράγουν τέχνη και να καταλαβαίνουν τόσο περίπλοκες έννοιες, αξίζει να γίνει η ερώτηση που γινόταν πριν καν δημιουργηθεί ο πρώτος λειτουργικός υπολογιστής. Μεχρι ποιο σημείο μπορεί πραγματικά ένας υπολογιστής να φτάσει τον άνθρωπο και αν ο άνθρωπος είναι το όριο ή απλά ένα ακόμα βήμα που θα ξεπεραστεί.

9ο Βιβλιογραφία

- [1] Taylor, Matthew E., and Peter Stone. "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research* 10.7 (2009).
- [2] Reese, Hope. "Understanding the differences between AI, machine learning, and deep learning." URL: [https://www.techrepublic.com/article/understandingthedifferencesbetweenaimachine learninganddeeplearning](https://www.techrepublic.com/article/understandingthedifferencesbetweenaimachinelearninganddeeplearning) (2017).
- [3] Goodfellow, I., Bengio, Y. and Courville, A., 2017. *Deep learning*. Cambridge, Mass: The MIT Press

- [4] Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." *Journal of Big data* 3.1 (2016): 1-40.
- [5] Mahesh, Batta. "Machine learning algorithms-a review." *International Journal of Science and Research (IJSR)*.*[Internet]* 9 (2020): 381-386.
- [6] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [7] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.
- [9] Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach." (2002).
- [10] Sultana, Farhana, Abu Sufian, and Paramartha Dutta. "Advancements in image classification using convolutional neural network." *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE, 2018.
- [11] Traore, Boukaye Boubacar, Bernard Kamsu-Foguem, and Fana Tangara. "Deep convolution neural network for image recognition." *Ecological Informatics* 48 (2018): 257-268.
- [12] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR. [13] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- [13] Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).
- [14] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).
- [17] Viswanathan, Nitin. "Artist identification with convolutional neural networks." *Stanford193CS231N Report* (2017).
- [18] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).
- [19] Xue Ying 2019 J. Phys.: Conf. Ser. 1168 022022
- [20] He, Tong, et al. "Bag of tricks for image classification with convolutional neural networks." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

10^ο Δικτυογραφία

- [1] <https://dl.acm.org/doi/pdf/10.5555/1577069.1755839>
- [2] <https://deeplearning.lipinyang.org/wp-content/uploads/2016/11/Understanding-the-differences-between-AI-machine-learning-and-deep-learning-TechRepublic.pdf>
- [3] https://books.google.gr/books?id=omivDQAAQBAJ&dq=deep+learning+&lr=&hl=el&source=gb_s_navlinks_s
- [4] <https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-016-0043-6.pdf>
- [5] https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-_A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf
- [6] <https://arxiv.org/pdf/1512.03385.pdf>
- [7] https://www-cs.stanford.edu/groups/vision/pdf/ImageNet_CVPR2009.pdf
- [8] <https://paperswithcode.com/dataset/imagenet>
- [9] https://www.researchgate.net/profile/Mahalakshmi-Neelam/publication/358119068_Neelam_MahaLakshmi_2021_Aspects_of_Artificial_Intelligence_In_KarthikeyanJ_Su-Hie_Ting_and_Yu-Jin_Ng_ed's_Learning_Outcomes_of_Classroom_Research_p250-256_L'_Ordine_Nuovo_Publication_India_978-93-92995/links/61f11e44c5e3103375bf0e88
- [10] <https://arxiv.org/pdf/1905.03288.pdf>
- [11] https://oatao.univ-toulouse.fr/21329/1/Traore_21329.pdf
- [12] <http://proceedings.mlr.press/v97/tan19a/tan19a.pdf>
- [13] <https://arxiv.org/pdf/2010.11929v2.pdf>
- [14] <https://arxiv.org/pdf/1610.02357v3.pdf>
- [15] <https://www.educba.com/what-is-pytorch/>
- [16] <https://www.simplilearn.com/what-is-pytorch-article>
- [18] <https://arxiv.org/pdf/1412.6980.pdf>

- [19] <https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf>
- [20] <https://arxiv.org/abs/1812.01187>