

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή εξεύρεσης κοινών χαρακτηριστικών και ομαδοποίηση φωτογραφικού υλικού χρήστη

Νικόλαος Ζούκας Α.Μ.: Π2014156

Επιβλέπων Καθηγητής:

κος. Μιχαήλ Στεφανιδάκης Αναπληρωτής Καθηγητής

Εξεταστική επιτροπή:

κα. Κάτια – Λήδα Κερμανίδου Αναπληρώτρια Καθηγήτρια

κος. Φοίβος Μυλωνάς Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με τη δημιουργία εφαρμογής για την ομαδοποίηση εικόνων με τη βοήθεια αλγορίθμων τεχνητής μάθησης και του μοντέλου νευρωνικού δικτύου VGG16 . Η πτυχιακή εργασία χρησιμοποιεί τη μέθοδο Transfer Learning για να συντομεύσει τον χρόνο εκπαίδευσης του μοντέλου ταξινόμησης εικόνων και να περιορίσει τη χρήση των απαιτούμενων πόρων. Η διαδικασία εκπαίδευσης και ταξινόμησης συντονίζεται μέσω εφαρμογής με διεπαφή web, η οποία αναπτύχθηκε με τη βοήθεια των λογισμικών Python, html και Flask. Στην πτυχιακή εργασία παρουσιάζεται η εγκατάσταση και η χρήση της εφαρμογής που αναπτύχθηκε, καθώς και τα συμπεράσματα και οι προοπτικές εξέλιξης της εργασίας.

SUMMARY

This dissertation deals with the creation of an application for the classification and grouping of images with the help of machine learning algorithms and the neural network model VGG16. The Transfer Learning method is used to shorten training times and limit the computational resources required. The process of training and classification is being coordinated through a web-based user interface, developed with Python, html and Flask software. The dissertation presents the deployment and usage of the application developed, as well as the conclusions and the future prospects of the selected method.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κο. Μιχαήλ Στεφανιδάκη για την πολύτιμη βοήθεια και καθοδήγηση του, αφού χωρίς την επίβλεψη του η πτυχιακή εργασία δεν θα ήταν δυνατή. Επίσης θα ήθελα να ευχαριστήσω τους γονείς και τον αδερφό μου για την υπομονή και τη στήριξη που μου έχουν δείξει κατά τα χρόνια των σπουδών μου. Ιδιαίτερα, τον θείο και την οικογένεια του στην Κέρκυρα που μου έδωσαν τη δυνατότητα να μείνω μαζί τους.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ.....	9
1.1 Σκοπός της πτυχιακής εργασίας.....	9
ΚΕΦΑΛΑΙΟ 2: ΒΑΣΙΚΑ ΛΟΓΙΣΜΙΚΑ.....	10
2.1 Python.....	10
2.1.1 Λίγα λόγια για την Python.....	10
2.1.2 Ιστορικά γεγονότα της Python.....	10
2.2 HTML.....	11
2.2.1 Λίγα λόγια για την HTML.....	11
2.2.2 Ιστορικά γεγονότα της HTML.....	11
2.3 FLASK.....	12
2.3.1 Λίγα λόγια για το Flask.....	12
2.3.2 Ιστορικά γεγονότα του Flask.....	12
ΚΕΦΑΛΑΙΟ 3: ΒΙΒΛΙΟΘΗΚΕΣ PYTHON.....	13
3.1 OS.....	13
3.2 SHUTIL.....	13
3.3 Glob.....	13
3.4 SQLITE3.....	13
3.5 KERAS.....	13
3.6 TENSORFLOW.....	14
3.7 NumPy.....	14
3.8 h5py.....	14
3.9 PIL.....	14
3.10 Requests.....	14
3.11 io.....	14
ΚΕΦΑΛΑΙΟ 4: ΕΞΕΥΡΕΣΗ ΚΟΙΝΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΚΑΙ ΟΜΑΔΟΠΟΙΗΣΗ ΕΙΚΟΝΩΝ.....	15
4.1 Εξεύρεση κοινών χαρακτηριστικών.....	15
4.2 Ομαδοποίηση εικόνων.....	15
4.2.1 Λίγα λόγια για την ομαδοποίηση εικόνων.....	15
4.2.2 Μέθοδοι ομαδοποίησης εικόνων.....	15
4.3 Artificial Neural Networks (ANNs/NNs).....	16

4.3.1 Τι είναι τα Artificial Neural Networks (ANNs/NNs).....	16
4.3.2 Training Artificial Neural Networks (ANNs/NNs).....	16
4.3.3 Ιστορία των Artificial Neural Networks (ANNs/NNs).....	17
4.4 Deep Learning.....	17
4.4.1 Λίγα λόγια για το Deep Learning.....	17
4.4.2 Ιστορικά γεγονότα του Deep Learning.....	17
4.5 Transfer Learning.....	18
4.5.1 Λίγα λόγια για το Transfer Learning.....	18
4.5.2 Μαθηματικός ορισμός του Transfer Learning.....	19
4.5.3 Ιστορικά γεγονότα του Transfer Learning.....	19
4.6 Deep Transfer Learning.....	19
4.7 Στρατηγικές Deep Transfer Learning.....	19
4.7.1 Έτοιμα προ εκπαιδευμένα μοντέλα για την εξαγωγή χαρακτηριστικών.....	20
4.7.2 Προσαρμογή προ εκπαιδευμένων μοντέλων.....	21
4.8 Είδη Deep Transfer Learning.....	21
4.8.1 Domain Adaptation.....	21
4.8.2 Domain Confusion.....	22
4.8.3 Multitask Learning.....	22
4.8.4 One-shot Learning.....	23
4.8.5 Zero-shot Learning.....	23
4.9 Μοντέλο VGG16.....	23
4.9.1 Λίγα λόγια για το μοντέλο VGG16.....	23
4.9.2 Αρχιτεκτονική του μοντέλου VGG16.....	24
ΚΕΦΑΛΑΙΟ 5: ΚΩΔΙΚΑΣ ΤΟΥ TRAINING ΚΑΙ SORTING.....	25
5.1 Training.....	25
5.2 Διαδικασία εκπαίδευσης του μοντέλου “Our Model”.....	30
5.3 Sorting.....	31
ΚΕΦΑΛΑΙΟ 6: USER INTERFACE.....	35
6.1 Grouping Images (UI).....	35
6.2 Images for Sorting.....	36
6.2.1 Images for Sorting (UI).....	36
6.2.2 Κώδικας του Image for Sorting.....	37
6.3 Models (UI).....	38

6.4 Train a new Model.....	39
6.4.1 Options (UI).....	39
6.4.2 Κώδικας Options.....	40
6.4.3 Name Categories (UI).....	41
6.4.4 Upload εικόνων για κάθε κατηγορία (UI).....	42
6.4.5 Κώδικας Upload Εικόνων στις κατηγορίες.....	43
6.4.6 Upload Completed (UI).....	45
6.5 Model's Profile.....	46
6.5.1 Model's Profile (UI).....	46
6.5.2 Train/Retrain results (UI).....	47
6.5.3 Deleting Model (UI).....	48
6.5.4 Κώδικας Deleting Model.....	49
6.6 Sorting.....	50
6.6.1 Sorting (UI).....	50
6.6.2 Categories (UI).....	51
6.6.3 Move (UI).....	52
6.6.4 Κώδικας Move.....	53
ΚΕΦΑΛΑΙΟ 7: ΕΓΚΑΤΑΣΤΑΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ.....	55
7.1 Anaconda Navigator.....	55
7.2 Jupyter Notebook.....	55
7.3 Δημιουργία Εικονικού Περιβάλλοντος (Virtual Environment).....	56
7.4 Εγκατάσταση Πακέτων.....	57
ΚΕΦΑΛΑΙΟ 8: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ.....	59
8.1 Συμπεράσματα.....	59
8.2 Προοπτικές.....	60
ΒΙΒΛΙΟΓΡΑΦΙΑ.....	61

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Σκοπός της πτυχιακής εργασίας

Ο σκοπός της πτυχιακής εργασίας είναι η δημιουργία ενός web προγράμματος, το οποίο ομαδοποιεί εικόνες χρησιμοποιώντας αλγορίθμους μηχανικής μάθησης. Στο πρόγραμμα αυτό ο χρήστης δημιουργεί τις κατηγορίες, και σε κάθε μία κατηγορία βάζει λίγες εικόνες που αντιπροσωπεύουν την κατηγορία. Το πρόγραμμα κατά την διάρκεια της εκπαίδευσης βρίσκει τα κοινά χαρακτηριστικά των εικόνων σε κάθε κατηγορία. Έτσι, δημιουργείται το μοντέλο και ο χρήστης μπορεί να το χρησιμοποιήσει για την ομαδοποίηση των υπόλοιπων εικόνων του. Για να πετύχει αυτό, χρησιμοποιούμε την μέθοδο Transfer Learning. Με την μέθοδο αυτή παίρνουμε τις γνώσεις από ένα ήδη εκπαιδευμένο μοντέλο και τις μετατρέπουμε ώστε να ομαδοποιεί τις εικόνες όπως θέλουμε. Το μοντέλο που χρησιμοποιούμε είναι το VGG16 το οποίο είναι εκπαιδευμένο σε πολλές κατηγορίες. Στο πρόγραμμα ο χρήστης πέρα από την δημιουργία του μοντέλου που αποθηκεύεται, μπορεί να το επανεκπαιδεύσει ή να το διαγράψει. Ο κώδικας αυτός τρέχει πίσω από html για να είναι πιο εύκολη η χρήση του κώδικα από τον χρήστη.

Το πρόγραμμα δημιουργήθηκε με Python για τον κώδικα, html για την εμφάνισή του και Flask για τη σύνδεσή τους. Ο κώδικας γράφτηκε στο Jupyter Notebook. Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι OS, Shutil, Glob, Sqlite3, Keras, Tensorflow, Numpy, h5py, PIL, Requests, io. Όλα αυτά εγκαταστάθηκαν σε ψηφιακό περιβάλλον από το Anaconda Navigator.

Το κείμενο που ακολουθεί διαρθρώνεται ως εξής:

- Στο Κεφάλαιο 2 περιγράφονται τα βασικά λογισμικά που χρησιμοποιήθηκαν στην πτυχιακή εργασία, Python, html και Flask.
- Στο Κεφάλαιο 3 παρατίθενται οι βιβλιοθήκες που χρησιμοποιήθηκαν για το πρόγραμμα.
- Στο Κεφάλαιο 4 παρουσιάζεται το θεωρητικό υπόβαθρο της εργασίας και η μέθοδος που χρησιμοποιήθηκε (Transfer Learning), μαζί με το μοντέλο (VGG16).
- Στο Κεφάλαιο 5 γίνεται περιγραφή του βασικού κώδικα, δηλαδή πως γίνεται η εκπαίδευση του μοντέλου και η ομαδοποίηση των εικόνων.
- Στο Κεφάλαιο 6 γίνεται περιγραφή της εμφάνισης του προγράμματος (UI) και τον υπόλοιπο κώδικα της εφαρμογής.
- Στο Κεφάλαιο 7 δίνονται λεπτομέρειες για την εγκατάσταση των προγραμμάτων.
- Στο Κεφάλαιο 8 βρίσκονται τα συμπεράσματα και οι προοπτικές του προγράμματος.

ΚΕΦΑΛΑΙΟ 2: ΒΑΣΙΚΑ ΛΟΓΙΣΜΙΚΑ

2.1 Python



2.1.1 Λίγα λόγια για την Python

Η Python είναι διερμηνευόμενη (interpreted), γενικού σκοπού (general-purpose) και υψηλού επιπέδου, γλώσσα προγραμματισμού. Είναι μία από τις προστακτικές γλώσσες προγραμματισμού (imperative programming) και υποστηρίζει το διαδικαστικό (procedural programming) και το αντικειμενοστραφές προγραμματισμό (object-oriented programming). Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (garbage collection). Οι βασικοί στόχοι της είναι η αναγνωσιμότητα του κώδικα της και η ευκολία χρήσης της. Η σύνταξη της δίνει την δυνατότητα στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα σε σχέση με άλλες γλώσσες όπως την C++ ή την Java. Ένας λόγος που έχει την δυνατότητα στην γραφή κώδικα σε λιγότερες γραμμές είναι η διαθεσιμότητα πολλών βιβλιοθηκών. Επειδή είναι διερμηνευόμενη είναι πιο αργή από τις μεταγλωττιζόμενες (compiled) γλώσσες όπως η C και η C++. [1]

2.1.2 Ιστορικά γεγονότα της Python

Δημιουργήθηκε από τον Ολλανδό Γκίντο βαν Ρόσσουμ (Guido van Rossum) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989. Θεωρείται διάδοχος της γλώσσας προγραμματισμού ABC, η οποία υπήρξε η βασική πηγή έμπνευσης για τον Γκίντο βαν Ρόσσουμ. Στης 16 Οκτωβρίου του 2000 εμφανίστηκε η πρώτη μεγάλη αναβάθμιση (Python 2.0) με πολλά νέα σημαντικά χαρακτηριστικά. Στης 3 Δεκεμβρίου του 2008 κυκλοφόρησε η Python 3.0 η οποία ήταν μια σημαντική αναθεώρηση της γλώσσας, με αποτέλεσμα να μην είναι εντελώς συμβατή με τις προηγούμενες εκδόσεις. Το 2015 ήταν σχεδιασμένο να τελειώσει η υποστήριξη (end-of-life) της Python 2.7, αλλά επειδή η μετατροπή του κώδικα σε Python 3 ήταν δύσκολη μεταφέρθηκε για το 2020. Τέλος λόγω προβλημάτων ασφάλειας, που είχαν οι εκδόσεις της Python αυξήθηκε η ταχύτητα δημιουργίας της Python 3.9.2 και 3.8.8. [2]

2.2 HTML



2.2.1 Λίγα λόγια για την HTML

Η HTML (Hyper Text Markup Language) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα. Έχει την δυνατότητα δημιουργίας δομημένων εγγράφων για να καθορίζονται δομικά στοιχεία για το κείμενο. Επίσης ενσωματώνονται γλώσσες όπως JavaScript για να επηρεάσουν την ιστοσελίδα. [3]

2.2.2 Ιστορικά γεγονότα της HTML

Η HTML δημιουργήθηκε από τον φυσικό Τιμ Μπέρνερς Λι στα τέλη του 1990. Η πρώτη δημόσια διαθέσιμη περιγραφή της HTML εμφανίστηκε στα τέλη του 1991. Περιέγραφε τα 20 στοιχεία για τον σχεδιασμό της HTML, από τα οποία δεκατρία υπάρχουν ακόμα και σήμερα στην HTML 4.0. Η HTML 4.0 δημοσιεύτηκε από το W3C (World Wide Web Consortium) τον Δεκέμβριο του 1997. Αυτή η έκδοση ενσωμάτωνε πολλά στοιχεία και ιδιότητες που προηγουμένως λίγοι browser χρησιμοποιούσαν, προσδοκούσε στην κατάργηση των δυνατοτήτων οπτικής διαμόρφωσης του Netscape προτείνοντας τα CSS. Τον Ιανουάριο του 2008 δημοσιεύτηκε η HTML5 ως ένα πρόχειρο εργασίας από το W3C, αυτόνομη. [3]

2.3 FLASK



2.3.1 Λίγα λόγια για το Flask

Το Flask είναι μία δομή ιστού (web framework) γραμμένο στη γλώσσα Python, παρέχει εργαλεία, βιβλιοθήκες και τεχνολογίες που επιτρέπουν στην δημιουργία εφαρμογών ιστού (web applications). Το Flask είναι απλό αλλά επεκτάσιμο επειδή είναι μικρό-πλαίσιο (micro-framework). Βασίζεται στο εργαλείο Werkzeug WSGI, με πρότυπη μηχανή (template engine) την Jinja2. [4]

2.3.2 Ιστορικά γεγονότα του Flask

Το Flask δημιουργήθηκε από τον Armin Ronacher, ο οποίος ήταν μέλος μίας διεθνούς ομάδας ενθουσιωδών χρηστών της Python που ιδρύθηκε 2004. Η πρώτη έκδοση του Flask δημοσιεύτηκε στις 1 Απριλίου του 2010, ως πρωταπριλιάτικο αστείο. Λόγο της διασημότητας που απέκτησε έγινε σοβαρή εφαρμογή.

ΚΕΦΑΛΑΙΟ 3: ΒΙΒΛΙΟΘΗΚΕΣ ΡΥΤΗΘΝ

3.1 OS

Το OS δίνει την δυνατότητα στο χρήστη να επεξεργαστεί ένα αρχείο, το path του, να διαβάσει όλες τις γραμμές του και να δημιουργήσει ένα προσωρινό αρχείο. [5]

3.2 SHUTIL

Το Shutil προσφέρει λειτουργίες υψηλού επιπέδου σε αρχεία και συλλογές αρχείων. Συγκεκριμένα, παρέχει λειτουργίες που υποστηρίζουν την αντιγραφή και διαγραφή αρχείων. [6]

3.3 Glob

Το Glob έχει την δυνατότητα να βρει όλα τα ονόματα διαδρομών (path names) που ταιριάζουν με ένα καθορισμένο μοτίβο, σύμφωνα με τους κανόνες της Unix. [7]

3.4 SQLITE3

Το SQLite3 είναι μία βιβλιοθήκη που παρέχει μια “ελαφριά” βάση δεδομένων που αποθηκεύεται στον δίσκο, χωρίς να χρειάζεται ξεχωριστό διακομιστή, επιτρέποντας έτσι την πρόσβαση της βάσης δεδομένων με την χρήση μίας μη τυπικής παραλλαγής της γλώσσας SQL. [8]

3.5 KERAS

Το Keras είναι μία βιβλιοθήκη με ανοιχτό πηγαίο κώδικα που παρέχει διεπαφές σε Python και χρησιμεύει σε τεχνητά νευρωνικά δίκτυα. Μέχρι την έκδοση 2.3, το Keras υποστήριζε TensorFlow, Microsoft Cognitive Toolkit, Theano και PlaidML. Από την έκδοση 2.4, το Keras υποστηρίζει μόνο TensorFlow. Επικεντρώνεται στο να είναι φιλικό στους χρήστες και επεκτάσιμο. Η πρώτη δημοσίευση ήταν στις 27 Μαρτίου 2015 από τον Francois Chollet. [9]

3.6 TENSORFLOW

Το TensorFlow είναι μία βιβλιοθήκη ανοιχτού πηγαίου κώδικα για μηχανική μάθηση. Μπορεί να χρησιμοποιηθεί σε αρκετών ειδών διεργασίες, αλλά δίνει ιδιαίτερη έμφαση στην εκπαίδευση βαθέν νευρωνικών δικτύων (deep neural networks). Δημιουργήθηκε από την ομάδα Google Brain για εσωτερική χρήση της Google και δημοσιεύτηκε το 2015. [10]

3.7 NumPy

Το NumPy (**N**umerical **P**ython) είναι μία βιβλιοθήκη Python, ανοιχτού πηγαίου κώδικα, που χρησιμοποιείται σε εργασίες με πίνακες. Επιπλέον, έχει λειτουργίες για γραμμική άλγεβρα και μετασχηματισμούς Fourier. Η βιβλιοθήκη δημιουργήθηκε το 2005 από τον Travis Oliphant. [11]

3.8 h5py

Σκοπός του h5py είναι η δημιουργία διεπαφών ανάμεσα στην Python και του συστήματος αρχείων HDF5, παρέχοντας την δυνατότητα αποθήκευσης και μετατροπής μεγάλων ποσοτήτων αριθμητικών δεδομένων από την NumPy. [12]

3.9 PIL

Το PIL (Python Imaging Library) δημιουργήθηκε το 1995 και είναι μία βιβλιοθήκη ανοιχτού πηγαίου κώδικα που προσφέρει υποστήριξη στην επεξεργασία πολλών διαφορετικών ειδών εικόνων. Η ανάπτυξη του αρχικού PIL σταμάτησε το 2011, αλλά δημιουργήθηκε το Pillow με αντιγραμμένο PIL κώδικα (forked) για να υποστηρίξει τις εκδόσεις Python 3.x. [13]

3.10 Requests

Η βιβλιοθήκη Requests επιτρέπει την αποστολή HTTP αιτημάτων με την χρήση Python, κάνοντάς τη διαδικασία πιο απλή και φιλική στον άνθρωπο. Η πρώτη έκδοσή της κυκλοφόρησε στις 14 Φεβρουαρίου του 2011 από τον Kenneth Reitz. [14] [15]

3.11 io

Η βιβλιοθήκη io βοηθάει την Python στην διαχείριση διάφορων τύπων I/O. Οι τρεις βασικοί είναι text I/O, binary I/O και raw I/O. Αυτές είναι γενικές κατηγορίες και διαφορετικοί τρόποι αποθήκευσης που μπορούν να χρησιμοποιηθούν για κάθε ένα από αυτά. [16]

ΚΕΦΑΛΑΙΟ 4: ΕΞΕΥΡΕΣΗ ΚΟΙΝΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΚΑΙ ΟΜΑΔΟΠΟΙΗΣΗ ΕΙΚΟΝΩΝ

4.1 Εξεύρεση κοινών χαρακτηριστικών

Για να μπορέσει ο υπολογιστής να καταλάβει σε ποια ομάδα ανήκει μία εικόνα, θα πρέπει πρώτα να βρει τα κοινά χαρακτηριστικά της σε σχέση με τις υπόλοιπες. Γι' αυτό, όταν εκπαιδεύεται ένα μοντέλο, ο υπολογιστής αναλύει τις εικόνες κάθε κατηγορίας, ώστε να βρει τα κοινά τους χαρακτηριστικά, τα οποία θα χαρακτηρίζουν την κατηγορία.

4.2 Ομαδοποίηση εικόνων

4.2.1 Λίγα λόγια για την ομαδοποίηση εικόνων

Κατά το στάδιο ομαδοποίησης εικόνας (Image classification), ο υπολογιστής αναλύει μία εικόνα και την κατηγοριοποιεί σε μία ομάδα. Οι ομάδες στην ουσία είναι μια ταμπέλα για το περιεχόμενο της εικόνας. Παλαιότερα, η ομαδοποίηση εικόνας βασιζόταν σε ακατέργαστα pixel δεδομένα. Αυτό σήμαινε ότι ο υπολογιστής θα έπρεπε να “σπάσει” την εικόνα σε ξεχωριστά pixels. Το πρόβλημα με αυτό είναι πως δύο φωτογραφίες του ίδιου αντικειμένου μπορεί να φαίνονται πολύ διαφορετικές. Δηλαδή, μπορούν να έχουν διαφορετικό υπόβαθρο, γωνίες, πόζες και άλλα πολλά. Αυτό είχε ως αποτέλεσμα τη δυσκολία του υπολογιστή να βρει τη σωστή κατηγορία στην οποία ανήκουν οι εικόνες. [17]

4.2.2 Μέθοδοι ομαδοποίησης εικόνων

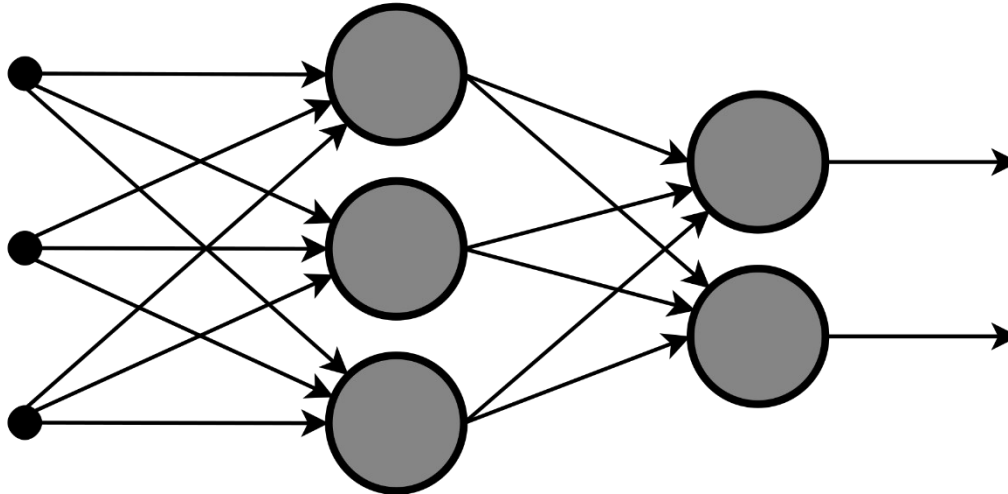
Υπάρχουν τρεις μέθοδοι ομαδοποίησης εικόνων. Η πρώτη και βασικότερη μέθοδος είναι η ταξινόμηση χωρίς επίβλεψη (Unsupervised Classification), η οποία πρώτα ομαδοποιεί τα pixels σε στοίβες (clusters), βασιζόμενη στις ιδιότητες τους και ταξινομώντας τα σε στοίβα (cluster) με μία κατηγορία.

Η δεύτερη μέθοδος είναι η ταξινόμηση με επίβλεψη (Supervised Classification), στην οποία διαλέγονται αντιπροσωπευτικά δείγματα για κάθε κατηγορία και μετά το λογισμικό χρησιμοποιεί την γνώση από την εκπαίδευση και την εφαρμόζει σε ολόκληρη την εικόνα.

Τέλος υπάρχει η ανάλυση εικόνας βάση αντικειμένου (Object-Based Image Analysis). Σε αντίθεση με τους άλλους δύο τρόπους που η ταξινόμηση τους βασίζεται στα pixels, αυτός ο τρόπος ομαδοποιεί τα pixels σε αντιπροσωπευτικά διανύσματα με διάφορα σχήματα και μεγέθη. [18]

4.3 Artificial Neural Networks (ANNs/NNs)

4.3.1 Τι είναι τα Artificial Neural Networks (ANNs/NNs)



Εικόνα 1: Artificial Neural Networks [19]

Τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks) είναι υπολογιστικά συστήματα, εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα, όπως των εγκεφάλων των ζώων. Ένα Τεχνητό Νευρωνικό Δίκτυο είναι βασισμένο στη συλλογή των συνδεδεμένων μονάδων ή κόμβων που ονομάζονται τεχνητοί νευρώνες. Ένας τεχνητός νευρώνας δέχεται ένα σήμα, το επεξεργάζεται και μετά έχει τη δυνατότητα να το στείλει στους συνδεδεμένους νευρώνες. Το σήμα κατά τη σύνδεση είναι ένας πραγματικός αριθμός, ενώ στην έξοδο μια μη-γραμμική συνάρτηση υπολογίζει το άθροισμα των εισόδων του κάθε νευρώνα. Οι συνδέσεις ονομάζονται άκρες (edges). Συνήθως οι νευρώνες και οι άκρες τους έχουν κάποιο βάρος (weight), το οποίο προσαρμόζεται με την πάροδο της μάθησης. Το βάρος αυξάνει ή μειώνει την δύναμη ενός σήματος σε μία σύνδεση. Οι νευρώνες μπορεί να έχουν κάποιο όριο όπου το σήμα θα σταλεί μόνο αν ξεπερνάει αυτό το όριο. Συνήθως οι νευρώνες έχουν επίπεδα. Κάθε επίπεδο μπορεί να εκτελέσει διαφορετική μετατροπή στην είσοδο του. Τα σήματα ξεκινάνε από το πρώτο επίπεδο (επίπεδο εισόδου) και καταλήγουν στο τελευταίο επίπεδο (επίπεδο εξόδου). [19]

4.3.2 Training Artificial Neural Networks (ANNs/NNs)

Τα νευρωνικά δίκτυα εκπαιδεύονται με την επεξεργασία παραδειγμάτων. Καθένα από αυτά περιέχει είσοδο και αποτέλεσμα. Με αυτόν τον τρόπο, σχηματίζονται και αποθηκεύονται τα πιθανά βάρη. Η εκπαίδευση ενός νευρωνικού δικτύου από ένα παράδειγμα συνήθως είναι αποτέλεσμα της διαφοράς μεταξύ της επεξεργασμένης εξόδου του δικτύου (συνχνά μία πρόβλεψη) και της εξόδου που είχε ως στόχο. Εάν υπάρχει διαφορά μεταξύ τους,

αυτό σημαίνει ότι υπάρχει σφάλμα και το δίκτυο προσαρμόζει τα βάρη με την χρήση της τιμής σφάλματος. Αν η προσαρμογή είναι επιτυχής, αυτό σημαίνει ότι η έξοδος που παρήγαγε ήταν παρόμοια με την έξοδο που είχε ως στόχο. [19]

4.3.3 Ιστορία των Artificial Neural Networks (ANNs/NNs)

Η πρώτη δημιουργία ενός υπολογιστικού μοντέλου για νευρωνικά δίκτυα έγινε το 1943 από τον Warren McCulloch και Walter Pitts [19]. Το πρώτο νευρωνικό δίκτυο με πολλά επίπεδα δημοσιεύτηκε το 1965 από τον Ivakhnenko και Lapa. Ο Ciresan και οι συνάδελφοι του δημιούργησαν το πρώτο αναγνωριστικό μοτίβου το οποίο επιτυγχάνει ανθρώπινη απόδοση. [19]

4.4 Deep Learning

4.4.1 Λίγα λόγια για το Deep Learning

Το Deep Learning είναι ένα είδος μηχανικής εκπαίδευσης. Αποτελεί ένα υποσύνολο της τεχνητής νοημοσύνης (AI) που επιτρέπει στις μηχανές να μαθαίνουν από τα δεδομένα. Με τη χρήση πολλών επιπέδων εξάγει υψηλού επιπέδου πληροφορίες. Κάθε επίπεδο μαθαίνει να μετατρέπει τα δεδομένα εισόδου του σε μία ελαφρώς πιο αφηρημένη και σύνθετη αναπαράσταση. Έτσι, τα pixels μίας φωτογραφίας καθώς περνάνε από τα επίπεδα, σχηματίζουν πληροφορίες για την φωτογραφία. Το Deep Learning περιλαμβάνει τη χρήση των υπολογιστών ως νευρωνικά δίκτυα (Neural Network), με το πιο γνωστό να είναι το Convolutional Neural Networks (CNN). [17] [20]

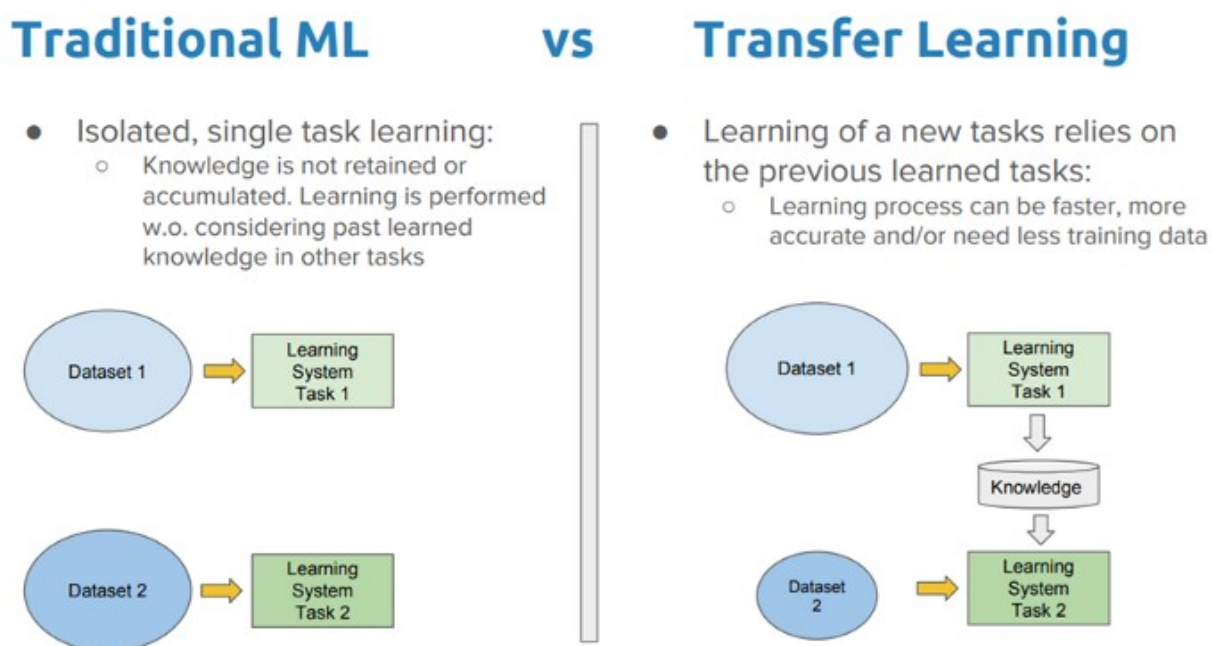
4.4.2 Ιστορικά γεγονότα του Deep Learning

Η πρώτη δουλειά πάνω στην εκπαίδευση αλγόριθμου με τα στοιχεία του Deep Learning ήταν από τον Alexey Ivakhnenko και Lapa το 1967. Από τότε υπήρξαν πολλών ειδών δημοσιεύσεις για τη βελτίωσή της. Μία σημαντική δημοσίευση ήταν το 2009 με την ανάπτυξη των γραφικών επεξεργασιών από την Nvidia που με βάση τον Andrew Ng μπορούσε να αυξήσει την ταχύτητα του Deep Learning κατά εκατό φορές. [20]

4.5 Transfer Learning

4.5.1 Λίγα λόγια για το Transfer Learning

Με τη μέθοδο του Transfer Learning υπάρχει ήδη ένα εκπαιδευμένο μοντέλο, το οποίο είναι εκπαιδευμένο και αναγνωρίζει ένα σύνολο αντικειμένων για την επίλυση ενός προβλήματος και υπάρχει και ένα δεύτερο, παρόμοιο πρόβλημα, το οποίο χρειάζεται λύση με παρόμοια αντικείμενα. Αντί να φτιαχτεί ένα μοντέλο από την αρχή, χρησιμοποιούνται οι γνώσεις του πρώτου μοντέλου που έχουμε ήδη για την εκπαίδευση και επίλυση του δεύτερου προβλήματος. Με αυτή τη μέθοδο γλυτώνουμε χρόνο, γιατί δεν χρειάζεται να εκπαιδεύσουμε ένα νέο μοντέλο από την αρχή. [21] [22]



Εικόνα 2: Διαφορές παραδοσιακής εκπαίδευσης με Transfer Learning [21]

4.5.2 Μαθηματικός ορισμός του Transfer Learning

Ο τομέας D αποτελείται από ένα χαρακτηριστικό χώρο (feature space) X και μία οριακή κατανομή πιθανότητας (marginal probability distribution) $P(X)$, όπου

$$X = \{x_1, \dots, x_n\} \in X.$$

Δίνοντας έναν συγκεκριμένο τομέα, $D = \{x, P(x)\}$, μία εργασία αποτελείται από δύο συστατικά: ένα διάστημα ετικέτας (label space) Y και μία αντικειμενική προγνωστική συνάρτηση (objective predictive function) $f: X \rightarrow Y$. Η συνάρτηση f χρησιμοποιείται για την πρόβλεψη της αντίστοιχης ετικέτας $f(x)$ μίας νέας τιμής x . Αυτή η εργασία, συμβολίζεται από $T = \{y, f(x)\}$, μαθαίνεται από την εκπαίδευση των δεδομένων που αποτελούνται από $\{x_i, y_i\}$, όπου $x_i \in X$ και $y_i \in Y$. Δίνοντας έναν τομέα προέλευσης (source domain) D_S και εκπαιδευτική εργασία (learning task) T_S , ένας στοχευμένος τομέας (target domain) D_T και εκπαιδευτική εργασία (learning task) T_T , όπου $D_S \neq D_T$, ή $T_S \neq T_T$, το Transfer Learning στοχεύει να βοηθήσει στην ανάπτυξη της εκπαίδευσης, της στοχευμένης προβλεπόμενης συνάρτησης (target predictive function) $f_T(\cdot)$ σε T_T χρησιμοποιώντας την γνώση από D_S και T_S . [22]

4.5.3 Ιστορικά γεγονότα του Transfer Learning

Η πρώτη δημοσίευση του Transfer Learning έγινε το 1976 από τον Stevo Bozinovski και Ante Fulgosi [22]. Το 1993, η Lorien Pratt διατύπωσε τον αλγόριθμο μεταφοράς βάσει διακρίσεων (discriminability-based transfer). Το 1998, οι Pratt και Sebastian Thrun δημοσίευσαν για την εκμάθηση πολλαπλών εργασιών (multi-task learning). [22]

4.6 Deep Transfer Learning

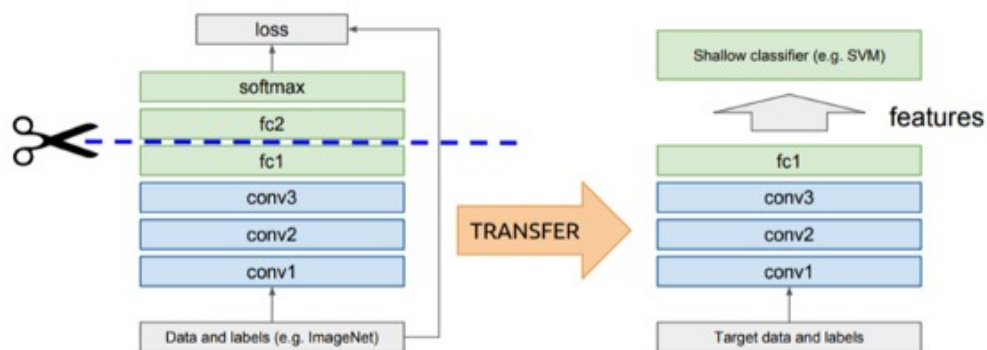
Με την χρήση του Deep Learning στο Transfer Learning, ο χρήστης μπορεί να μάθει παραστάσεις από δεδομένα χωρίς ετικέτα, να κάνει εκπαίδευση σε παρόμοια αντικείμενα στα οποία είναι εύκολο να παράγει ετικέτες και μπορεί να μεταφέρει μαθημένες παραστάσεις από μία σχετική εργασία. [21]

4.7 Στρατηγικές Deep Transfer Learning

4.7.1 Έτοιμα προ εκπαιδευμένα μοντέλα για την εξαγωγή χαρακτηριστικών

Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that $D_S = D_T$



Εικόνα 3: Off-the-shelf Pre-trained Models as Feature Extractors [21]

Μία από τις δημοφιλέστερες στρατηγικές του Deep Transfer Learning είναι η αφαίρεση του τελευταίου επιπέδου ενός εκπαιδευμένου μοντέλου, στο οποίο γίνεται η ομαδοποίηση. Αφαιρώντας την, ο χρήστης μπορεί να χρησιμοποιήσει τις γνώσεις των άλλων επιπέδων και να ορίσει τις εξόδους που θέλει. [21]

4.7.2 Προσαρμογή προ εκπαιδευμένων μοντέλων

Fine-tuning: supervised domain adaptation

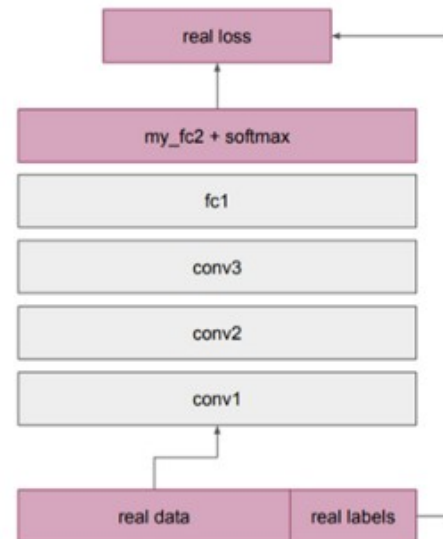
Train deep net on “nearby” task for which it is easy to get labels using standard backprop

- E.g. ImageNet classification
- Pseudo classes from augmented data
- Slow feature learning, ego-motion

Cut off top layer(s) of network and replace with supervised objective for target domain

Fine-tune network using backprop with labels for target domain until validation loss starts to increase

Aligns D_S with D_T



Εικόνα 4: Fine-tuning [21]

Με αυτή τη μέθοδο, πέρα από την αφαίρεση του τελευταίου επιπέδου, εκπαιδεύονται ξανά μερικά από τα άλλα επίπεδα. Για να γίνει αυτό, θα χρειαστεί να παγώσουν κάποιες μεταβλητές βάρους (fix weights) συγκεκριμένων επιπέδων για να μπορέσει ο χρήστης να ξανά-εκπαιδεύσει τα υπόλοιπα επίπεδα. [21]

4.8 Είδη Deep Transfer Learning

4.8.1 Domain Adaptation

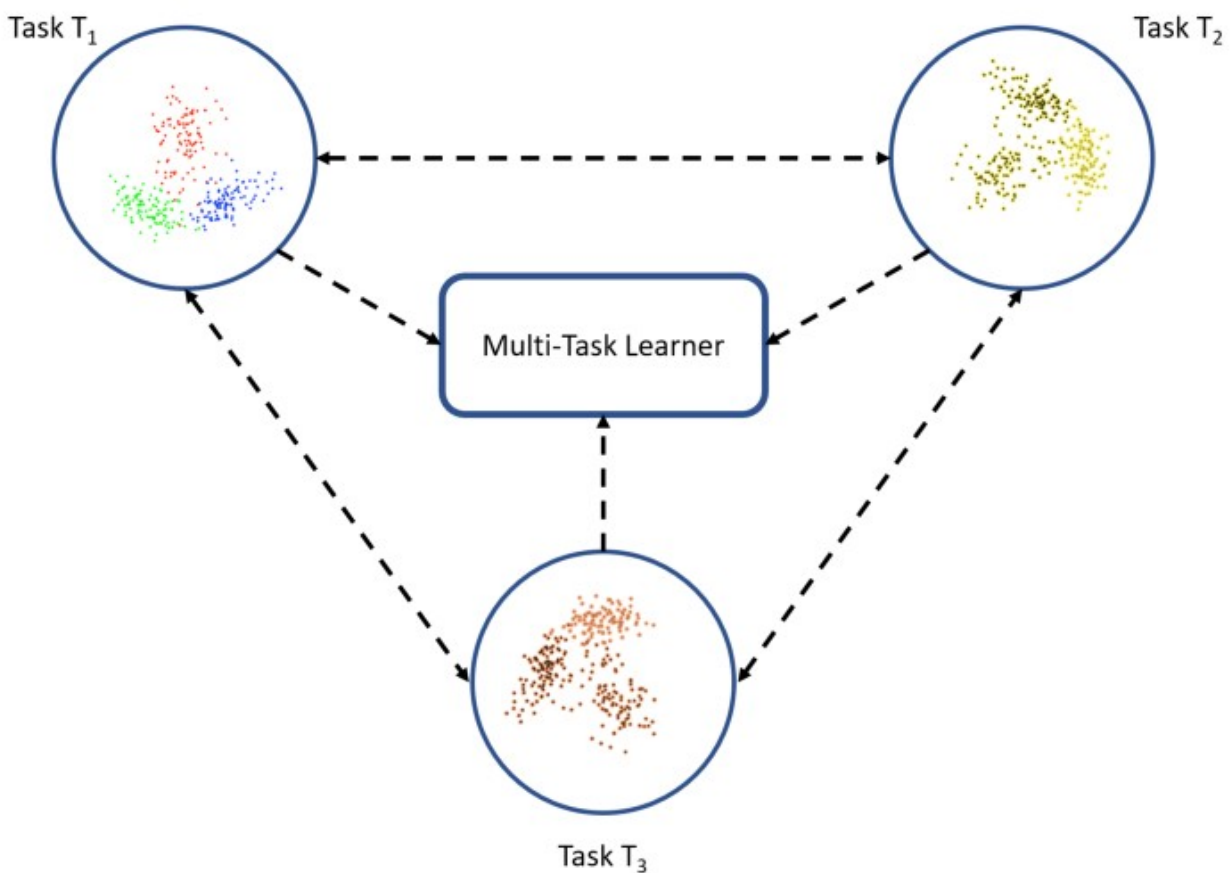
Κατά το Domain Adaptation, η οριακή πιθανότητα μεταξύ της πηγής και του στόχου του τομέα (Domain) είναι διαφορετική ($P(x_S) \neq P(x_t)$). Μία μετατόπιση στην κατανομή δεδομένων του τομέα πηγής και τροποποίηση στόχου κρίνεται απαραίτητη για να εκτελεστεί το Transfer Learning. [21]

4.8.2 Domain Confusion

Σε αυτό το είδος Deep Transfer Learning προστίθεται ένας ακόμα στόχος στο αρχικό μοντέλο για να ενθαρρυνθεί η ομοιότητα με σύγχυση (Confusion) του ίδιου του τομέα. [21]

4.8.3 Multitask Learning

Σε αυτήν την περίπτωση, πολλές εργασίες εκπαιδεύονται ταυτόχρονα, χωρίς διάκριση μεταξύ της πηγής και των στόχων. Με αυτόν τον τρόπο λαμβάνουμε πληροφορίες για πολλές εργασίες ταυτόχρονα. [21]



Εικόνα 5: Multitask Learning [21]

4.8.4 One-shot Learning

Με αυτόν τον τρόπο γίνεται προσπάθεια να επιτευχθεί ο στόχος με ένα ή λίγα εκπαιδευτικά παραδείγματα. Αυτό χρησιμεύει καλύτερα όταν δεν υπάρχουν πολλά δεδομένα με ετικέτες (labeled) για κάθε κατηγορία. [21]

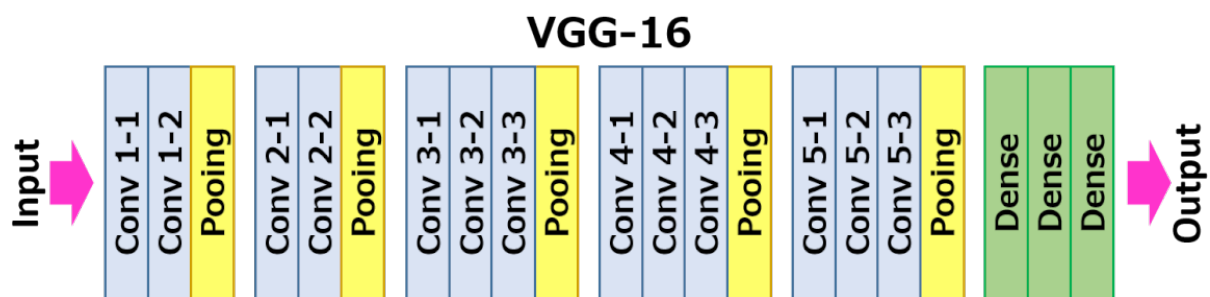
4.8.5 Zero-shot Learning

Το Zero-shot Learning βασίζεται σε παραδείγματα χωρίς ετικέτες (no labeled) για την εκπαίδευση μίας εργασίας. Κάνοντας έξυπνες αλλαγές την ώρα της εκπαίδευσης, εξαγονται επιπλέον πληροφορίες από τα δεδομένα. [21]

4.9 Μοντέλο VGG16

4.9.1 Λίγα λόγια για το μοντέλο VGG16

Στην εφαρμογή της πτυχιακής εργασίας το προ εκπαιδευμένο μοντέλο που χρησιμοποιείται είναι το VGG16. Το VGG16 είναι ένα συνελκτικό νευρωνικό δίκτυο που προτάθηκε από τους K. Simonyan και A. Zisserman, από το πανεπιστήμιο της Οξφόρδης στο άρθρο «Very Deep Convolutional Network for Large-Scale Image Recognition». Το μοντέλο έχει πετύχει 92,7% ακρίβεια στο ImageNet. Το ImageNet είναι ένα σύνολο δεδομένων με πάνω από 14 εκατομμύρια εικόνες που ανήκουν σε 1000 κατηγορίες. Το VGG16 εκπαιδεύτηκε για εβδομάδες και χρησιμοποιούσε NVIDIA Titan Black GPUs. [23]



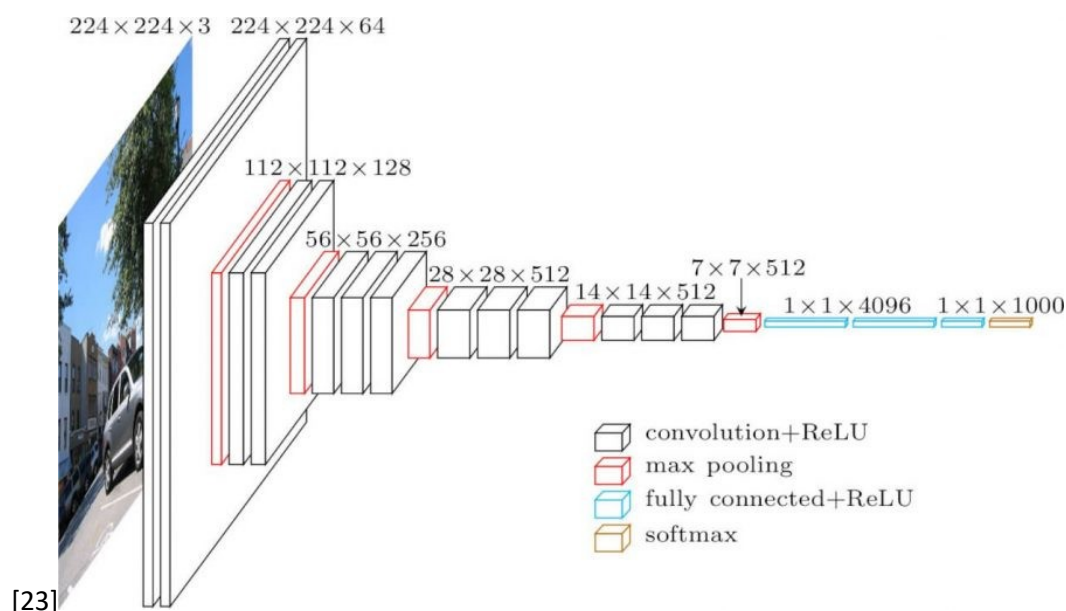
Εικόνα 6: VGG16 [23]

4.9.2 Αρχιτεκτονική του μοντέλου VGG16

Η είσοδος στο conv1 επίπεδο είναι σταθερή στο μέγεθος της εικόνας σε 224×224 RGB (Red Green Blue). Η εικόνα περνάει από μία στοίβα συνελκτικών (conv.) επιπέδων, όπου τα φίλτρα χρησιμοποιήθηκαν με πολύ μικρό πεδίο υποδοχής (3×3), το οποίο είναι το μικρότερο μέγεθος για να μπορεί να συλλάβει την έννοια του αριστερά / δεξιά, πάνω / κάτω, κέντρο.

Υπάρχουν φίλτρα συνέλιξης 1×1 , τα οποία μπορούν να θεωρηθούν ως γραμμικός μετασχηματισμός των καναλιών εισόδου. Το βήμα της συνέλιξης είναι σταθερό σε 1 pixel, δηλαδή το χωρικό περιθώριο (padding) της συνέλιξης. Η είσοδος στο επίπεδο είναι τέτοια ώστε η χωρική ανάλυση να διατηρείται μετά από συνελιγμό, δηλαδή το περιθώριο (padding) είναι 1-pixel για κάθε 3×3 επίπεδο συνέλιξης. Η χωρική συγκέντρωση (pooling) πραγματοποιείται από πέντε επίπεδα μέγιστης συγκέντρωσης (max-pooling), τα οποία ακολουθούν ορισμένα από τα συνελκτικά (conv.) επίπεδα. Η μέγιστη συγκέντρωση (max-pooling) πραγματοποιείται σε παράθυρο 2×2 pixel.

Τρία πλήρως συνδεδεμένα επίπεδα ακολουθούν μία στοίβα από συνελκτικά επίπεδα. Τα δύο πρώτα έχουν 4096 κανάλια το καθένα, το τρίτο εκτελεί 1000 κατευθύνσεων ILSVRC ταξινόμηση και έτσι περιέχει 1000 κανάλια (ένα για κάθε ομάδα). Το τελευταίο επίπεδο είναι το soft-max επίπεδο. Η διαμόρφωση των πλήρων συνδεδεμένων επιπέδων είναι η ίδια σε όλα τα δίκτυα. Όλα τα κρυφά στρώματα είναι εξοπλισμένα με τη μη γραμμική διόρθωση (ReLU).



Εικόνα 7: Αρχιτεκτονική VGG16 [23]

ΚΕΦΑΛΑΙΟ 5: ΚΩΔΙΚΑΣ ΤΟΥ TRAINING ΚΑΙ SORTING

5.1 Training

```
@app.route("/train")

def train():
    import tensorflow
    import keras
    vgg16_model = keras.applications.vgg16.VGG16()
    vgg16_model.summary()
    type(vgg16_model)
    keras.engine.training.Model
    import keras.models
    import numpy as np
    from keras.models import Sequential
    from keras.layers import Activation
    from keras.layers.core import Lambda, Dense, Flatten
    from keras.optimizers import Adam
    global categ, epoch, mname, allcname
    import os
    from tensorflow.keras.layers import GlobalAveragePooling2D
    from tensorflow.keras.preprocessing import image
    model = Sequential()
```

Σε αυτό το σημείο του κώδικα ξεκινάει η εκπαίδευση ή η επανεκπαίδευση του μοντέλου. Στην συνέχεια ακολουθείται από την εισαγωγή των βιβλιοθηκών και του αρχικού μοντέλου VGG16.

```
for layer in vgg16_model.layers:

    model.add(layer)

for layer in model.layers:

    layer.trainable = False

model.add(Dense(categ, activation='softmax'))
```

Έπειτα, αφαιρούμε τα τελευταία επίπεδα του μοντέλου και προσθέτουμε τα νέα επίπεδα που μετά θα εκπαιδεύσουμε. Στην τελευταία σειρά του κώδικα προσθέτουμε το επίπεδο soft-max με το “categ” να είναι ο αριθμός των κατηγοριών που διάλεξε ο χρήστης.

```

train_generator=train_datagen.flow_from_directory
    ('C:\\Users\\Zouki\\Flask Folder Try 4\\train',
    target_size=(224,224),
    color_mode='rgb',
    batch_size=1,
    class_mode='categorical',
    shuffle=True)

```

Σε αυτό το σημείο δίνουμε την διεύθυνση που βρίσκονται οι φωτογραφίες για την εκπαίδευση του μοντέλου. Το “target_size” το κάνουμε (224, 224), ώστε κάθε εικόνα να μετατρέπεται σε μέγεθος 224 x 224. Το “color_mode” το ορίζουμε “rgb” για να μετατρέπει τις εικόνες έγχρωμες. Το “batch_size” είναι το μέγεθος των παρτίδων δεδομένων. Στο “class_mode” έχουμε “categorical”, γιατί οι ομάδες που είναι για πρόβλεψη μπορεί να είναι όσες θέλει ο χρήστης. Το “shuffle” είναι στο “true”, ώστε να επιλέγονται με τυχαία σειρά οι εικόνες. [24]

```

checkpoint_path =
"C:\\Users\\Zouki\\Flask Folder Try 4\\usermodel\\"+ mname +"\\cp.ckpt"
checkpoint_dir=os.path.dirname(checkpoint_path)
cp_callback= tensorflow.keras.callbacks.ModelCheckpoint
(filepath=checkpoint_path, save_weights_only=True, verbose=1)

```

Στην συνέχεια, δημιουργούμε την διεύθυνση που θα αποθηκεύσουμε το μοντέλο. Το μοντέλο θα αποθηκευτεί στον φάκελο με το όνομα που ο χρήστης έδωσε στο μοντέλο. Το “filepath” είναι το μέρος που θα αποθηκεύει το μοντέλο. Έχουμε “True” στο “save_weights_only” για να αποθηκεύσει τα “weights”. Με το “verbose” στο 1 ο κώδικας εμφανίζει περισσότερες πληροφορίες για την εκπαίδευση ανά επανάληψη. [25]

```
step_size_train=train_generator.n//train_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=step_size_train,
                    epochs=epoch,
                    callbacks=[cp_callback])
```

Σε αυτό το σημείο γίνεται η εκπαίδευση του μοντέλου. Στο “generator” έχουμε το “train_generator” που αποθηκεύσαμε την διεύθυνση των εικόνων και τις ρυθμίσεις τους. Στο “steps_per_epoch” έχουμε το πόσες από τις φωτογραφίες που έχουμε θα χρησιμοποιηθούν για την εκπαίδευση, που είναι όλες. Στο “epoch” έχουμε την μεταβλητή “epoch” που είναι ο αριθμός των επαναλήψεων που ο χρήστης έχει ορίσει. Το “callbacks” είναι για την αποθήκευση και φόρτωση του μοντέλου σε κάθε επανάληψη.

```
macc=model.history.history['accuracy']
lastmacc = "%.2f" % (macc[-1]*100)
```

Αποθηκεύουμε στο “macc” τα ποσοστά επιτυχίας που είχε κάθε επανάληψη και στο “lastmacc” το ποσοστό επιτυχίας της τελευταίας επανάληψης.

```

conn = sqlite3.connect('model.db')

c = conn.cursor()

i=0

if (trorre == 0):

    c.execute("INSERT INTO models VALUES (?, ?, ?, ?)",

              (mname, categ, epoch, lastmacc))

    conn.commit()

    while (i < categ):

        c.execute("SELECT model from categories")

        c.execute("INSERT INTO categories VALUES (?, ?, ?)",

                  (mname, i+1, allcname[i]))

        conn.commit()

        i=i+1

    allcname.clear()

```

Αφού τελειώνει η εκπαίδευση του μοντέλου και έχουμε όλες τις πληροφορίες που θέλουμε να αποθηκεύσουμε, ελέγχουμε αν η μεταβλητή “trorre” (**train or retrain**) είναι 0. Αν είναι 0, σημαίνει ότι έγινε εκπαίδευση στο μοντέλο. Συνδεόμαστε στη βάση δεδομένων “model.db” και αποθηκεύουμε στον πίνακα “models” το όνομα του μοντέλου από την μεταβλητή “mname”, τον αριθμό των κατηγοριών από την μεταβλητή “categ”, τον αριθμό των επαναλήψεων από τη μεταβλητή “epoch”, και το ποσοστό επιτυχίας της τελευταίας επανάληψης από τη μεταβλητή “lastmacc”.

Στην συνέχεια αποθηκεύουμε στον πίνακα “categories” το όνομα του μοντέλου, τον αριθμό της κατηγορίας και το όνομα της, το οποίο επαναλαμβάνεται για κάθε κατηγορία για να έχουμε όλα τα ονόματα των κατηγοριών που ο χρήστης έβαλε. Τα ονόματα των κατηγοριών τα παίρνουμε από τον global πίνακα “allcname”.

```

if (trorre == 1):
    global data

    c.execute("DELETE FROM models WHERE model = ?", (mname,))

    epoch = epoch + data[0][2]

    c.execute("INSERT INTO models VALUES (?, ?, ?, ?)", (mname, categ, epoch, lastmacc))

    conn.commit()

```

Αν η μεταβλητή “trorre” είναι 1, σημαίνει ότι έγινε επανεκπαίδευση του μοντέλου. Διαγράφουμε τις πληροφορίες που είχαμε για το μοντέλο, προσθέτουμε τις επαναλήψεις που είχε κάνει σε άλλες εκπαιδεύσεις με τις επαναλήψεις που έκανε σε αυτήν την επανεκπαίδευση και ξανά βάζουμε τις πληροφορίες στον πίνακα “models”. Αφού είναι επανεκπαίδευση δεν χρειάζεται να αποθηκεύσουμε κάτι στον πίνακα “categories”.

```

return render_template("train.html", macc=macc)

```

Στο τέλος στέλνουμε στο “train.html” το “macc” για να εμφανίσουμε το ποσοστό επιτυχίας της εκπαίδευσης σε κάθε επανάληψη.

5.2 Διαδικασία εκπαίδευσης του μοντέλου “Our Model”

Στο πρόγραμμα υπάρχει διαθέσιμο μοντέλο, το οποίο έχει εκπαιδευτεί με την χρήση του Transfer Learning και του VGG16 μοντέλου. Ομαδοποιεί τις εικόνες σε τέσσερις κατηγορίες: φαγητά, ανθρώπους, αστικά μέρη και δασικά μέρη. Αφού οι κατηγορίες είναι τέσσερις και οι κατηγορίες είναι γενικές, χρειάστηκαν πολλές εικόνες και επαναλήψεις εκπαίδευσης για να έχουμε καλό ποσοστό επιτυχίας. Γι’ αυτό χρησιμοποιήθηκαν 305 εικόνες για κάθε κατηγορία, σύνολο 1.220 φωτογραφίες για την εκπαίδευση.

Οι εικόνες βρέθηκαν από την ιστοσελίδα “kaggle” στην οποία υπάρχουν πολλά σύνολα δεδομένων (<https://www.kaggle.com/>). Επειδή κάθε επανάληψη έκανε περίπου στα 20 λεπτά χωρίστηκαν οι επαναλήψεις ανά 10. Δηλαδή, έγιναν οι πρώτες 10 επαναλήψεις και ανάλογα με το ποσοστό επιτυχίας κάναμε ξανά εκπαίδευση μέχρι να αποκτήσει ένα καλό ποσοστό επιτυχίας ή μέχρι να σταματήσει να υπάρχει σημαντική αύξηση στο ποσοστό επιτυχίας. Για την εκπαίδευση αυτού του μοντέλου χρειάστηκαν τέσσερις φορές των 10 επαναλήψεων με την κάθε 10αδα να χρειάζεται περίπου τρεις ώρες. Στην παρακάτω φωτογραφία έχουμε την τελευταία 10αδα επαναλήψεων.

Το ποσοστό επιτυχίας (accuracy) καθορίζεται από το πόσες από τις φωτογραφίες που έχουμε δώσει τις έχει ομαδοποιήσει σωστά. Δηλαδή με την τελευταία επανάληψη να έχει 0,8787 (87,87%), σημαίνει ότι από τις 1.220 εικόνες έχει ομαδοποιήσει σωστά τις 1.072 εικόνες. Σε αυτό το σημείο φαίνεται ότι το μοντέλο δεν μπορεί να ανεβάσει παραπάνω το ποσοστό επιτυχίας του με τις εικόνες που έχει, γιατί σε αυτές τις επαναλήψεις το ποσοστό επιτυχίας του δεν ανέβηκε σημαντικά.

```

Epoch 1/10
1220/1220 [=====] - ETA: 0s - loss: 1.2856 - accuracy: 0.8689
Epoch 00001: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1202s 985ms/step - loss: 1.2856 - accuracy: 0.8689
Epoch 2/10
1220/1220 [=====] - ETA: 0s - loss: 1.2764 - accuracy: 0.8721
Epoch 00002: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1162s 953ms/step - loss: 1.2764 - accuracy: 0.8721
Epoch 3/10
1220/1220 [=====] - ETA: 0s - loss: 1.2672 - accuracy: 0.8746
Epoch 00003: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1107s 908ms/step - loss: 1.2672 - accuracy: 0.8746
Epoch 4/10
1220/1220 [=====] - ETA: 0s - loss: 1.2581 - accuracy: 0.8779
Epoch 00004: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1048s 859ms/step - loss: 1.2581 - accuracy: 0.8779
Epoch 5/10
1220/1220 [=====] - ETA: 0s - loss: 1.2491 - accuracy: 0.8787
Epoch 00005: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1038s 851ms/step - loss: 1.2491 - accuracy: 0.8787
Epoch 6/10
1220/1220 [=====] - ETA: 0s - loss: 1.2402 - accuracy: 0.8770
Epoch 00006: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1119s 917ms/step - loss: 1.2402 - accuracy: 0.8770
Epoch 7/10
1220/1220 [=====] - ETA: 0s - loss: 1.2314 - accuracy: 0.8811
Epoch 00007: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1052s 862ms/step - loss: 1.2314 - accuracy: 0.8811
Epoch 8/10
1220/1220 [=====] - ETA: 0s - loss: 1.2226 - accuracy: 0.8754
Epoch 00008: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1215s 996ms/step - loss: 1.2226 - accuracy: 0.8754
Epoch 9/10
1220/1220 [=====] - ETA: 0s - loss: 1.2139 - accuracy: 0.8795
Epoch 00009: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1117s 916ms/step - loss: 1.2139 - accuracy: 0.8795
Epoch 10/10
1220/1220 [=====] - ETA: 0s - loss: 1.2054 - accuracy: 0.8787
Epoch 00010: saving model to C:\Users\Zouki\anaconda3\envs\venv\train5\cp.ckpt
1220/1220 [=====] - 1191s 977ms/step - loss: 1.2054 - accuracy: 0.8787

```

5.3 Sorting

```

@app.route('/code2/<filename>')
def code2(filename):
    import tensorflow
    import keras
    vgg16_model = keras.applications.vgg16.VGG16()
    type(vgg16_model)
    keras.engine.training.Model
    import keras.models
    import numpy as np
    from keras.models import Sequential
    from keras.layers import Activation
    from keras.layers.core import Lambda, Dense, Flatten
    from keras.optimizers import Adam
    import os
    from tensorflow.keras.layers import GlobalAveragePooling2D
    from tensorflow.keras.preprocessing import image

```

Σε αυτό το σημείο του κώδικα ξεκινάει η ομαδοποίηση (sorting) των εικόνων. Στην συνέχεια ακολουθούν οι εισαγωγές των βιβλιοθηκών και του αρχικού μοντέλου VGG16.

```
conn = sqlite3.connect('model.db')
c = conn.cursor()
c.execute("SELECT model from models")
conn.commit()
c.execute("SELECT * FROM models WHERE model = ?", (filename,))
global categ,mname
mname = filename
categ = c.fetchall()
c.execute("SELECT * FROM models WHERE model = ?", (filename,))
data = c.fetchall()
categ = data [0][1]
print(categ)
print(c.fetchall())
conn.close()
```

Στην συνέχεια, με την μεταβλητή “filename” που έχουμε το όνομα του μοντέλου, παίρνουμε τις υπόλοιπες πληροφορίες για το μοντέλο, “categ” για τον αριθμό των κατηγοριών, “mname” για το όνομα του μοντέλου. Μετά ακολουθούν παρόμοιες σειρές κώδικα με τον κώδικα της εκπαίδευσης, με ένα από αυτά να είναι η φόρτωση του μοντέλου για τη χρήση του.

```
def make_square(img):
    cols,rows = img.size

    if rows>cols:
        pad = (rows-cols)/2
        img = img.crop((pad,0,cols,cols))
    else:
        pad = (cols-rows)/2
        img = img.crop((0,pad,rows,rows))

    return img
```

Εδώ, τετραγωνίζουμε την εικόνα, παίρνοντας τις διαστάσεις. Έπειτα, ελέγχουμε ποια διάσταση είναι μεγαλύτερη και την κόβουμε, ώστε οι διαστάσεις “cols” και “rows” να είναι ίσες. Δηλαδή το ύψος και το πλάτος τις εικόνας, τα κάνουμε ίσα για να γίνει τετράγωνη η εικόνα.

```

i=0;
for url in images:

    x = []
    ImageFile.LOAD_TRUNCATED_IMAGES = False
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    img.load()
    img = img.resize((IMAGE_WIDTH, IMAGE_HEIGHT), Image.ANTIALIAS)

    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    pred = model.predict(x)

    result.append(np.argmax(pred))
    display("_____")
    display(img)
    print(np.argmax(pred, axis=1))
    print(result)
    i+=1

```

Εδώ έχουμε επανάληψη για κάθε εικόνα που θέλουμε να ομαδοποιήσουμε. Αρχικά, μετατρέπουμε την εικόνα σε 224 x 224 διαστάσεις (IMAGE_WIDTH και IMAGE_HEIGHT είναι 224). Το "Image.ANTIALIAS" είναι ένα υψηλής ποιότητας φίλτρο δειγματοληψίας. Έπειτα, εξάγουμε τα χαρακτηριστικά της εικόνας και γίνεται η πρόβλεψη σε ποια κατηγορία ανήκει η εικόνα. Όσο γίνεται η επανάληψη, αυξάνουμε κατά ένα την μεταβλητή "i" για να ξέρουμε σε πόσες εικόνες κάνουμε ταξινόμηση.

```

delcounter=1
reset=1
while reset>0:
    dir=("C:\\Users\\Zouki\\Flask Folder Try 4\\folder"+str(delcounter))
    if (os.path.isdir(dir)):
        shutil.rmtree("C:\\Users\\Zouki\\Flask Folder Try 4\\
folder"+str(delcounter))
        delcounter+=1
    else:
        reset=0

```

Σε αυτό το σημείο, αν υπάρχουν ήδη οι φάκελοι "folder1", "folder2", ..., "foldern", όπου "n" είναι οι συνολικές κατηγορίες που είχε η προηγούμενη ομαδοποίηση που έγινε, τότε τους διαγράφουμε, ώστε να δημιουργήσουμε νέους φακέλους, στους οποίους θα αποθηκεύσουμε τις εικόνες που ομαδοποιούμε τώρα.

```

icateg=int(1)
while icateg<=categ:
    j=0
    os.mkdir("C:\\Users\\Zouki\\Flask Folder Try 4\\folder"+str(icateg))
    fol_dir = "C:/Users/Zouki/Flask Folder Try 4/folder"+str(icateg)+"/"
    print (fol_dir)
    while j<i:
        print(result[j]+1,icateg)
        if ((result[j]+1)==(icateg)):
            os.rename(src_dir+image_names[j], fol_dir+image_names[j])
        j+=1
    icateg+=1
return render_template("code2.html")

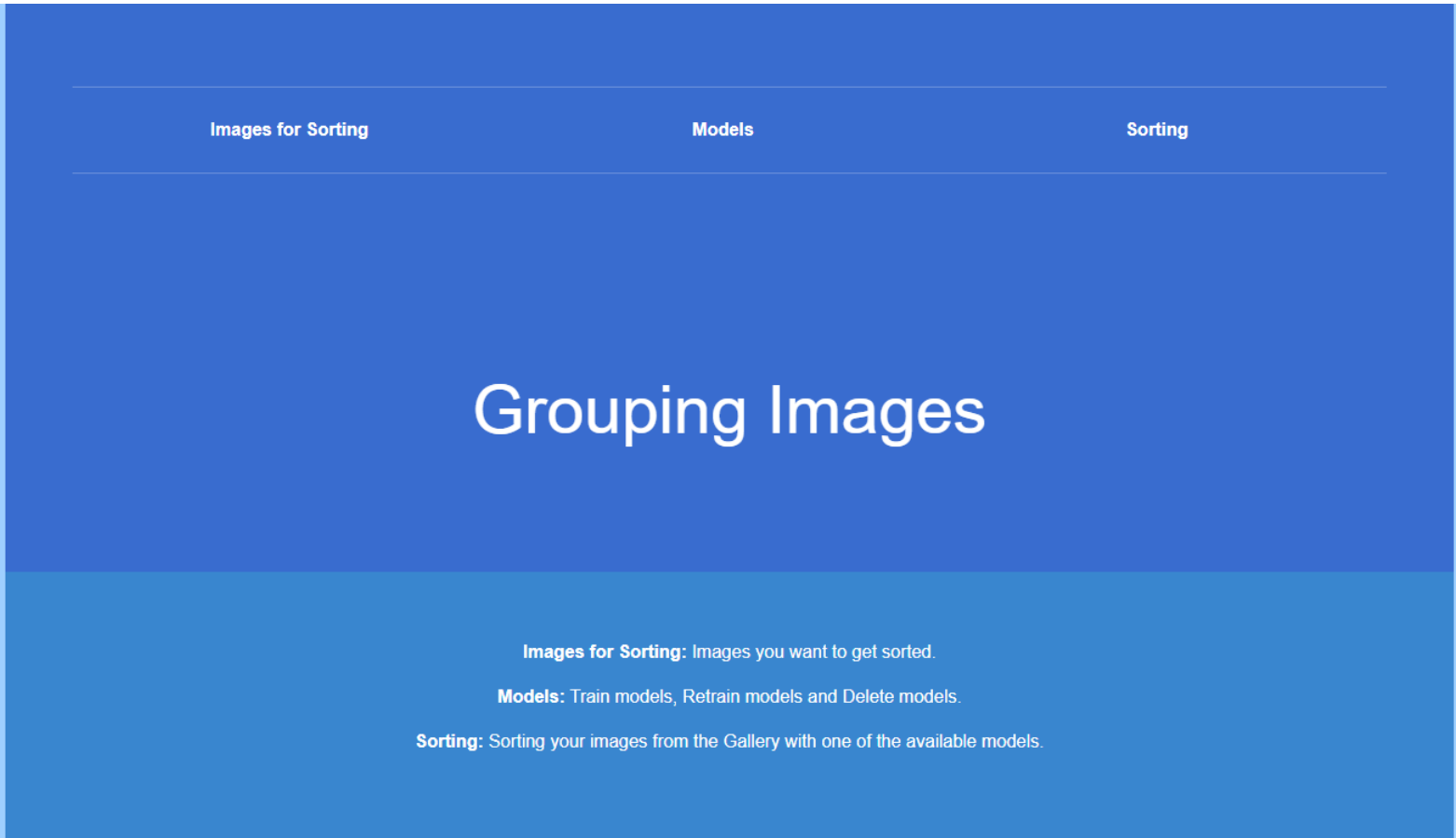
```

Στην συνέχεια φτιάχνουμε τους φακέλους “folder1”, “folder2”, ..., “foldern”, όπου “n” είναι οι συνολικές κατηγορίες που έχει αυτή η ομαδοποίηση. Έτσι, το “folder1” θα αντιστοιχεί για την πρώτη κατηγορία μέχρι το “foldern” που θα αντιστοιχεί για την τελευταία κατηγορία. Η επανάληψη γίνεται για κάθε κατηγορία και σε κάθε επανάληψη, πέρα από την δημιουργία του φακέλου, γίνεται έλεγχος για κάθε εικόνα, αν δηλαδή ανήκει στην κατηγορία του φακέλου που δημιουργήθηκε στην συγκεκριμένη επανάληψη. Αν ανήκει, τότε την μεταφέρουμε σε εκείνον τον φάκελο.

Για να γίνει ο έλεγχος κάθε εικόνας, έχουμε μία δεύτερη επανάληψη που ελέγχει αν η εικόνα ανήκει στην κατηγορία. Επειδή η μεταβλητή με τις προβλέψεις (result) ξεκινάει από το 0, ενώ η κατηγορίες ξεκινάνε από το 1 (δηλαδή η εικόνα με αριθμό πρόβλεψης 0 ανήκει στην κατηγορία 1), προσθέτουμε κατά 1 τη μεταβλητή του πίνακα “result” στον έλεγχο.

ΚΕΦΑΛΑΙΟ 6: USER INTERFACE

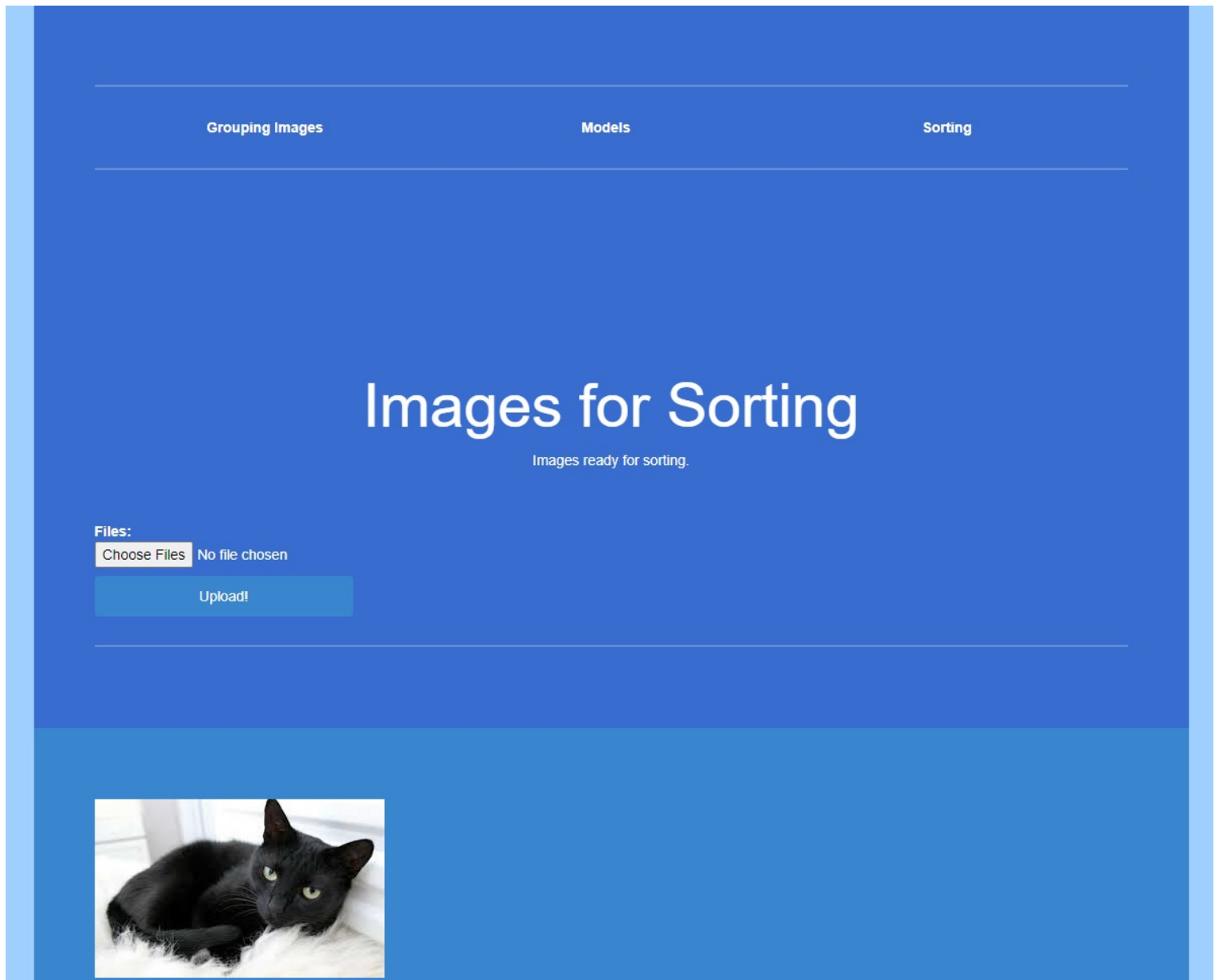
6.1 Grouping Images (UI)



Η αρχική σελίδα περιέχει 3 βασικά κουμπιά που μαζί με το κουμπί της αρχικής “Grouping Images” υπάρχουν στο πάνω μέρος κάθε σελίδας. Το πρώτο κουμπί είναι το “Images for Sorting”. Το κουμπί αυτό είναι το μέρος όπου ο χρήστης μπορεί να κάνει “upload” τις εικόνες που θέλει να ομαδοποιήσει. Το δεύτερο κουμπί είναι το “Models”. Εκεί βρίσκονται διαθέσιμα όλα τα μοντέλα για επεξεργασία (επανεκπαίδευση και διαγραφή μοντέλου) ή μπορεί να γίνει η εκπαίδευση για ένα νέο μοντέλο. Το τελευταίο κουμπί της αρχικής σελίδας είναι το “Sorting”. Είναι το μέρος όπου ο χρήστης μπορεί να διαλέξει ένα από τα διαθέσιμα μοντέλα για την ομαδοποίηση των εικόνων που βρίσκονται στο “Images for Sorting”.

6.2 Images for Sorting

6.2.1 Images for Sorting (UI)



Σε αυτήν την σελίδα πέρα από τα βασικά κουμπιά υπάρχει και το “Upload!”, ώστε να μπορέσει ο χρήστης να ανεβάσει τις εικόνες που θέλει να ομαδοποιήσει αργότερα. Σε περίπτωση που δεν υπάρχουν φωτογραφίες, εμφανίζεται το μήνυμα “There is no image available.”

6.2.2 Κώδικας του Image for Sorting

```
@app.route("/upload", methods=["POST"])

def upload():
    target = os.path.join(APP_ROOT, 'images/')
    print(target)
    if not os.path.isdir(target):
        os.mkdir(target)
    else:
        print("Couldn't create upload directory: {}".format(target))
    print(request.files.getlist("file"))
    for upload in request.files.getlist("file"):
        print(upload)
        print("{} is the file name".format(upload.filename))
        filename = upload.filename
        destination = "/".join([target, filename])
        print("Accept incoming file:", filename)
        print("Save it to:", destination)
        upload.save(destination)
    return render_template("complete.html", image_name=filename)
@app.route('/upload/<filename>')
def send_image(filename):
    return send_from_directory("images", filename)
```

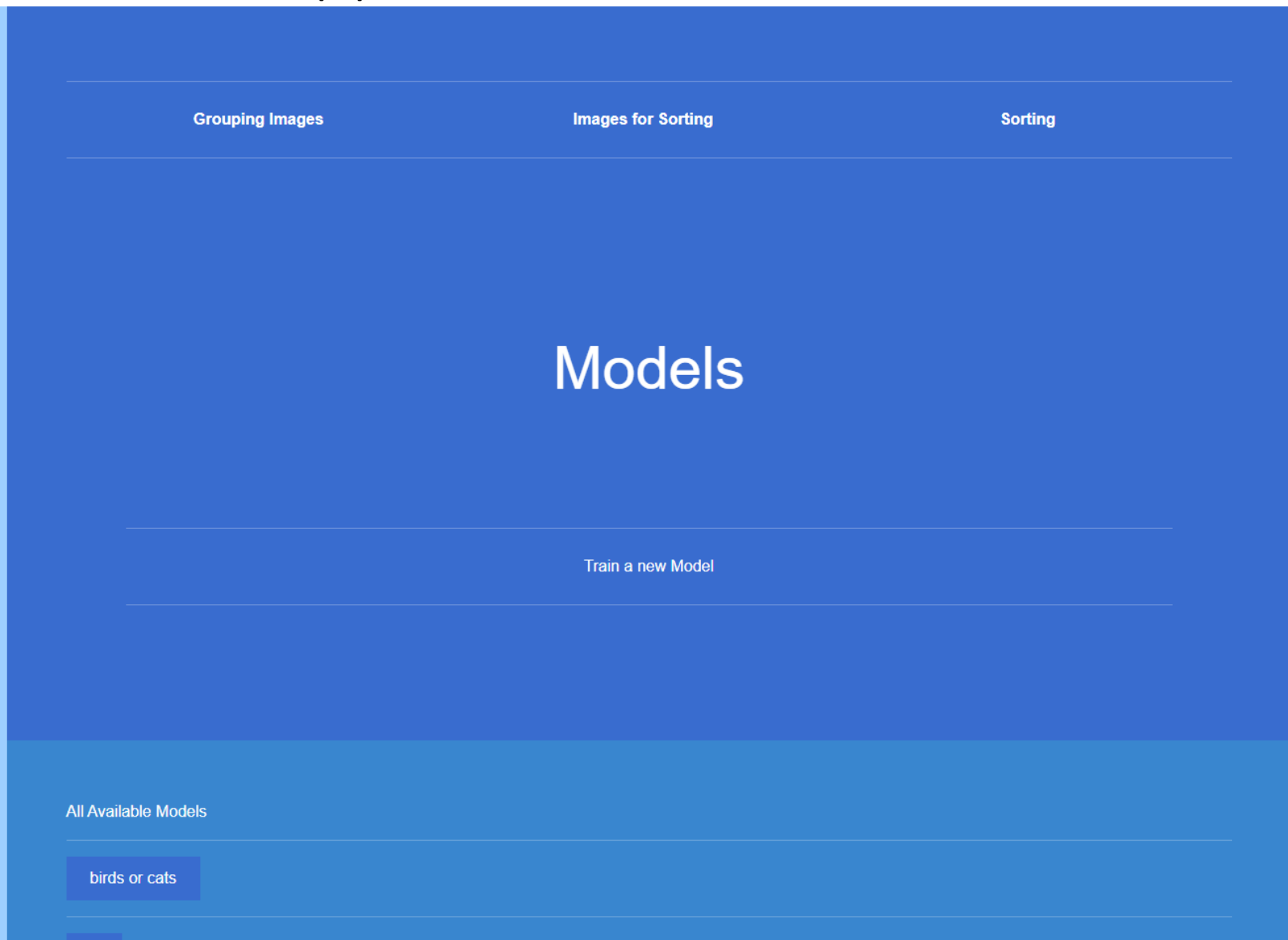
Σε αυτό το κομμάτι κώδικα, οι εικόνες αποθηκεύονται στον φάκελο “images” και δημιουργούνται διευθύνσεις (link) για κάθε εικόνα.

```
@app.route('/gallery')
def get_gallery():
    image_names = os.listdir('./images')
    image_url = ["http://127.0.0.1:4555/upload/" + item for item in
image_names]
    print(image_url)
    i=""
    print(image_names)
    if len(image_names)==0:
        i="There is no image available."
        print(i)
    return render_template("gallery.html", image_names=image_names,i=i)
```

Για την εμφάνιση των εικόνων στην σελίδα “Image for Sorting”, εισάγουμε στη μεταβλητή “image_names” τα ονόματα των εικόνων που βρίσκονται στον φάκελο “images” και τις διευθύνσεις τους στη μεταβλητή “image_url”. Στην συνέχεια βάζουμε στη μεταβλητή “i” ένα κενό κείμενο και αν το “image_names” είναι μηδέν, δηλαδή δεν υπάρχουν φωτογραφίες τότε βάζουμε στο “i” το μήνυμα “There is no image available.”. Έτσι, στέλνουμε στο html

πρότυπο (template) τις εικόνες για να τις εμφανίσει και το “i” που πάντα εμφανίζεται, απλά όταν υπάρχουν φωτογραφίες το “i” είναι κενό οπότε δεν εμφανίζεται κάτι.

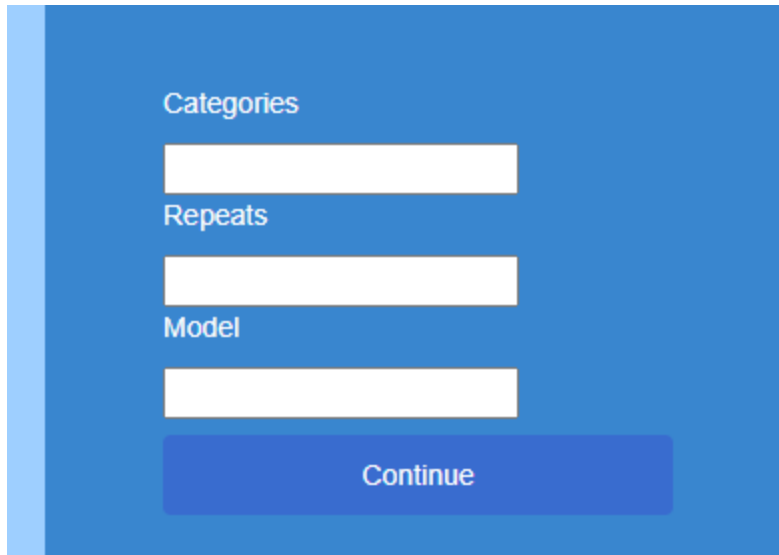
6.3 Models (UI)



Όπως σε όλες τις σελίδες υπάρχουν τα βασικά κουμπιά στο πάνω μέρος, έτσι και εδώ, κάτω από τον τίτλο υπάρχει το κουμπί “Train a new Model”, το οποίο είναι για τη δημιουργία νέου μοντέλου. Κάτω από το μήνυμα “All Available Models” βρίσκονται όλα τα μοντέλα που ο χρήστης εκπαιδευσε.

6.4 Train a new Model

6.4.1 Options (UI)



The image shows a user interface for training a new model. It has a blue background. On the left, there are three white input fields stacked vertically. The first field is labeled 'Categories', the second 'Repeats', and the third 'Model'. Below these fields is a blue button with the text 'Continue' in white.

Πατώντας στο κουμπί “Train a new Model”, πηγαίνουμε στα “options”, όπου ο χρήστης εισάγει στοιχεία για το μοντέλο που θέλει να εκπαιδεύσει. Τα στοιχεία αυτά είναι: το όνομα του μοντέλου, τον αριθμό των κατηγοριών και τον αριθμό των επαναλήψεων της εκπαίδευσης του μοντέλου.

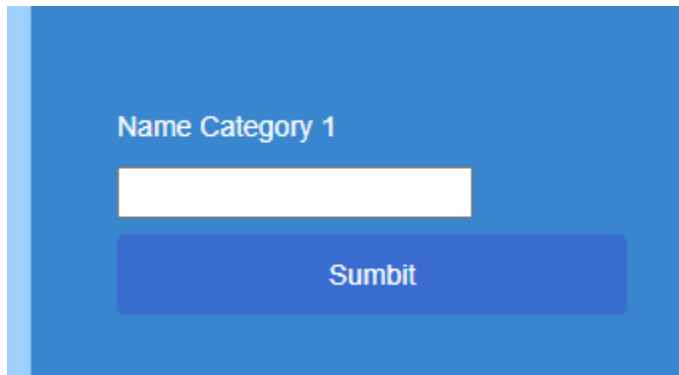
6.4.2 Κώδικας Options

```
@app.route("/options", methods=['POST'])
def options():
    global curcateg, categ, epoch, mname, trorre, allcname
    allcname=['hi']
    allcname.clear()
    trorre=0
    curcateg=int(1)
    categ = int(request.form['categ'])
    epoch = int(request.form['epoch'])
    mname = str(request.form['mname'])
    print(curcateg, categ, epoch, mname)
    if os.path.isdir("C:\\Users\\Zouki\\Flask Folder Try 4\\
usermodel\\"+str(mname)):
        return render_template ("modelexist.html")
    else:
        return render_template ("namecategories.html", curcateg=int(1))
```

Στην αρχή του κώδικα, σημειώνουμε τις “global” μεταβλητές για να διατηρούνται οι τιμές τους όταν αλλάζουμε σελίδα. Βάζουμε στη μεταβλητή “allcname” ένα τυχαίο μήνυμα, όπως “hi” και μετά αδειάζουμε τη μεταβλητή. Ο λόγος που το κάνουμε αυτό είναι ώστε σε περίπτωση που προσπαθήσουμε να αδειάσουμε τη μεταβλητή, ενώ αυτή είναι ήδη άδεια, να βγάλει “error”.

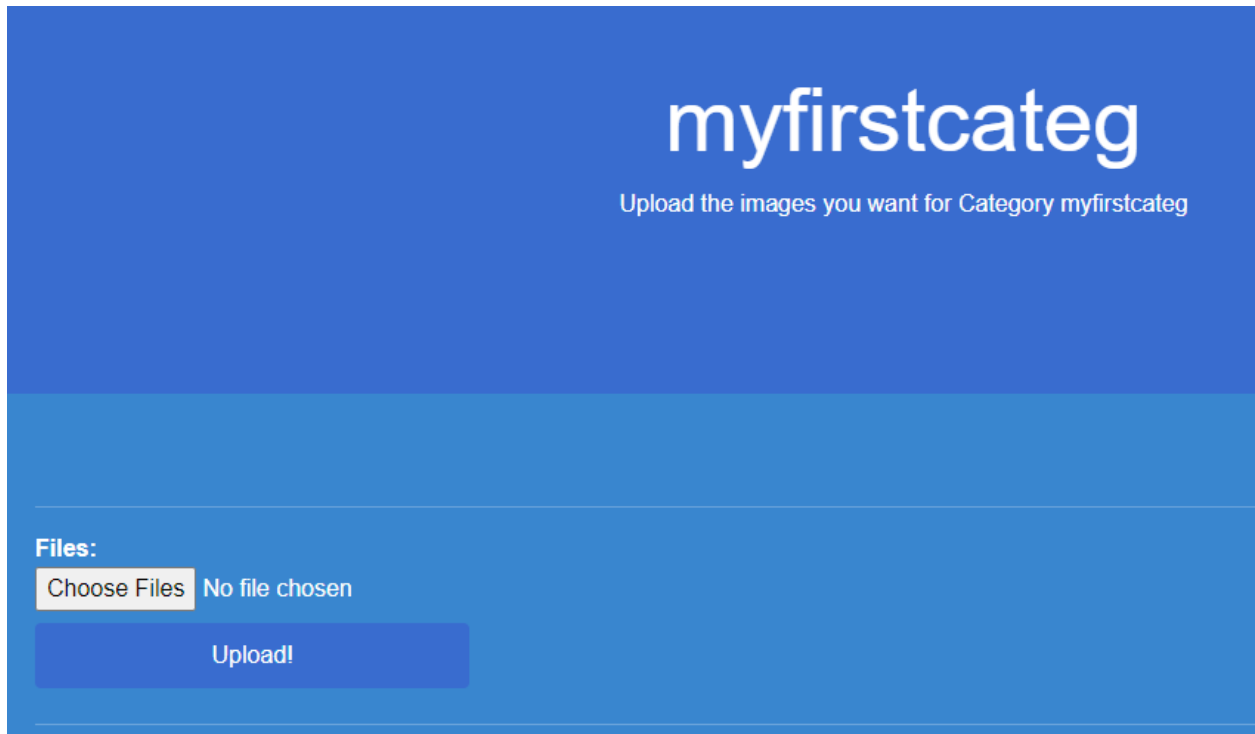
Στην συνέχεια κάνουμε τη μεταβλητή “trorre” (**train or retrain**), μηδέν για να ξέρει ο κώδικας αργότερα στην εκπαίδευση ότι κάνουμε εκπαίδευση. Αποθηκεύουμε τα νέα στοιχεία του χρήστη στις “global” μεταβλητές και ελέγχουμε αν υπάρχει ήδη το μοντέλο, σε αυτήν την περίπτωση εμφανίζεται το μήνυμα “Model Already Exists”. Σε αντίθετη περίπτωση, ο χρήστης συνεχίζει στην εισαγωγή ονομάτων των κατηγοριών.

6.4.3 Name Categories (UI)

A screenshot of a web interface with a blue background. At the top, the text "Name Category 1" is displayed. Below it is a white rectangular input field. At the bottom is a blue rectangular button with the word "Submit" in white text.

Εδώ, ο χρήστης ονομάζει κάθε κατηγορία. Στο τέλος του “Name Category” υπάρχει ένας αριθμός που δείχνει ποια κατηγορία ο χρήστης ονομάζει. Παράδειγμα αποτελεί η συγκεκριμένη φωτογραφία που γράφει “Name Category 1”. Αυτό σημαίνει ότι ονομάζει την πρώτη κατηγορία. Όταν ονομαστεί και η τελευταία κατηγορία, τότε χρειάζεται να γίνει “upload” ένα δείγμα εικόνων σε κάθε κατηγορία.

6.4.4 Upload εικόνων για κάθε κατηγορία (UI)



myfirstcateg

Upload the images you want for Category myfirstcateg

Files:

Choose Files No file chosen

Upload!

Με αυτόν τον τρόπο, ο χρήστης τοποθετεί εικόνες σε κάθε κατηγορία μέσω του κουμπιού “upload”, όπως γίνεται και με τις εικόνες που είναι για ομαδοποίηση. Εδώ, η σελίδα έχει ως τίτλο το όνομα της κατηγορίας που έδωσε ο χρήστης, ώστε να γνωρίζει πώς οι εικόνες που θα κάνει “upload” σε ποια κατηγορία θα αποθηκευτούν. Στη συγκεκριμένη εικόνα, ο χρήστης έχει ονομάσει την κατηγορία “myfirstcateg” στην οποία θα κάνει “upload” εικόνες. Οι εικόνες που θα γίνουν “upload” σε κάθε κατηγορία είναι ένα δείγμα για να εκπαιδευτεί το μοντέλο ώστε να ομαδοποιεί τις εικόνες σωστά.

6.4.5 Κώδικας Upload Εικόνων στις κατηγορίες

```
@app.route("/upload1", methods=["POST"])
def upload1():
    import shutil
    global curcateg, categ, mname, allcname, trorre
```

Αυτό είναι το σημείο που ξεκινάει ο κώδικας για το ανέβασμα (upload) και αποθήκευση των εικόνων για την εκπαίδευση ή την επανεκπαίδευση. Στη συνέχεια, εισάγονται οι βιβλιοθήκες και οι global μεταβλητές.

```
if os.path.isdir("C:\\Users\\Zouki\\Flask Folder Try 4\\train\\
images"+str(curcateg)):

    shutil.rmtree("C:\\Users\\Zouki\\Flask Folder Try 4\\train\\
images"+str(curcateg))

dir=("C:\\Users\\Zouki\\Flask Folder Try 4\\train\\images"+str(curcateg))

os.mkdir("C:\\Users\\Zouki\\Flask Folder Try 4\\train\\
images"+str(curcateg))

target = os.path.join("C:/Users/Zouki/Flask Folder Try 4/train/images"+
str(curcateg) + '/')
print(target)
if not os.path.isdir(target):
    os.mkdir(target)
else:
    print("Couldn't create upload directory: {}".format(target))
print(request.files.getlist("file"))
for upload1 in request.files.getlist("file"):
    print(upload1)
    print("{} is the file name".format(upload1.filename))
    filename = upload1.filename
    destination = "/".join([target, filename])
    print ("Accept incoming file:", filename)
    print ("Save it to:", destination)
    upload1.save(destination)
```

Οι εικόνες θα αποθηκευτούν στους φακέλους "images1", "images2", ..., "imagesn", με "n" ο συνολικός αριθμός κατηγοριών. Οι εικόνες από τον φάκελο "images1" αντιστοιχούν στην πρώτη κατηγορία μέχρι τον φάκελο "imagesn" που είναι για την "n" κατηγορία. Αλλά πριν φτιαχτούν οι φάκελοι, ελέγχουμε αν υπάρχουν ήδη από προηγούμενη εκπαίδευση ή επανεκπαίδευση. Στην περίπτωση που υπάρχουν, τους διαγράφουμε και φτιάχνουμε νέους φακέλους όπου μετά θα αποθηκεύσουμε τις εικόνες. Στη συνέχεια, ακολουθεί η αποθήκευση των εικόνων όπως γίνεται στο "Image for Sorting".

```

curcateg = curcateg + 1
if curcateg<=categ:
    i=curcateg-1
    if (trorre==1):
        conn = sqlite3.connect('model.db')
        c = conn.cursor()
        c.execute("SELECT model from categories")
        print (mname,curcateg)
        c.execute("SELECT categoryname FROM categories WHERE model = ? and
number = ?", (mname,curcateg,))
        conn.commit()
        cdata=c.fetchall()
        print(cdata)
        print( c.fetchall())
        conn.close()

        return render_template("upload1.html",foldername=cdata[0][0])
    else:
        return render_template("upload1.html",foldername=allcname[i])
if curcateg>categ:
    curcateg = 1
    return render_template("complete2.html")

```

Αφού γίνει το “upload” των εικόνων μίας κατηγορίας έχουμε τη μεταβλητή “curcateg” για να μετράει τις κατηγορίες που έχουμε κάνει το “upload”. Στο πρώτο “if” ελέγχουμε αν υπάρχει άλλη κατηγορία για να κάνουμε “upload”. Εάν υπάρχει, τότε ελέγχουμε αν η μεταβλητή “trorre” είναι ίση με το ένα, δηλαδή αν είναι για επανεκπαίδευση. Γιατί, αν είναι για επανεκπαίδευση, σημαίνει ότι έχουμε τις πληροφορίες για το μοντέλο και τα ονόματα των κατηγοριών στην βάση δεδομένων, όπου και παίρνουμε για να δείξουμε στον χρήστη το όνομα της κατηγορίας, στην οποία ο χρήστης θα βάλει εικόνες. Αν η μεταβλητή “trorre” δεν είναι ίση με 1, σημαίνει ότι κάνουμε εκπαίδευση, οπότε έχουμε τα ονόματα των κατηγοριών στη global μεταβλητή “allcname”.

6.4.6 Upload Completed (UI)

Grouping Images

Images for Sorting

Models

Sorting

Upload Completed

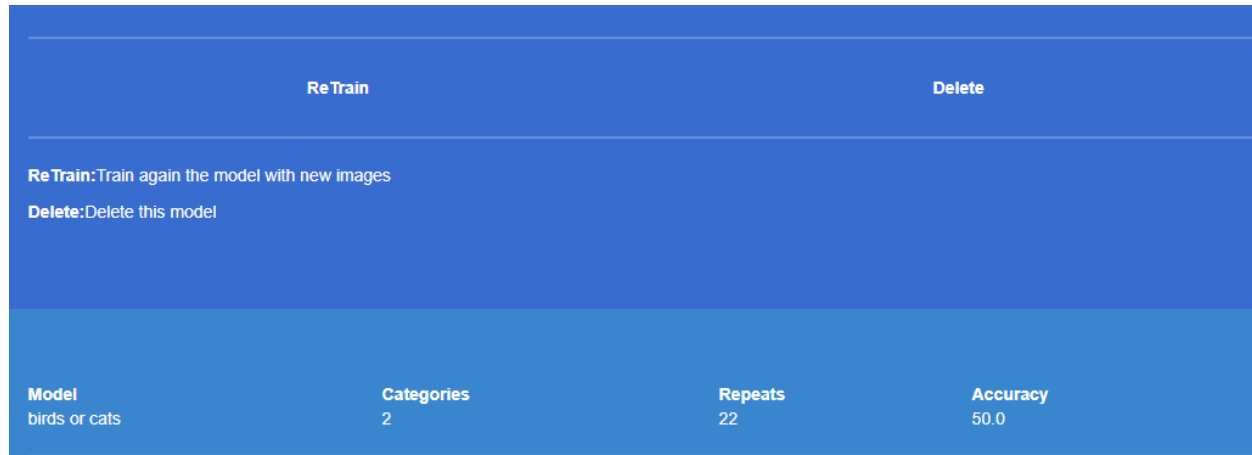
Ready to train the model with the new images.

Train

Με το “upload” των εικόνων της τελευταία κατηγορίας ολοκληρώνονται τα απαιτούμενα “uploads” και ο χρήστης πατώντας το κουμπί “Train” ξεκινάει την εκπαίδευση του νέου μοντέλου ή την επανεκπαίδευση του υπάρχοντος μοντέλου. Όταν πατηθεί το κουμπί “Train” εμφανίζεται ένα “animation” με έναν κύκλο, ως ένδειξη ότι γίνεται η εκπαίδευση, ή η επανεκπαίδευση του μοντέλου. Όταν τελειώσει η εκπαίδευση, εμφανίζεται το μήνυμα “Train Completed” και τα ποσοστά επιτυχίας σε κάθε επανάληψη.

6.5 Model's Profile

6.5.1 Model's Profile (UI)

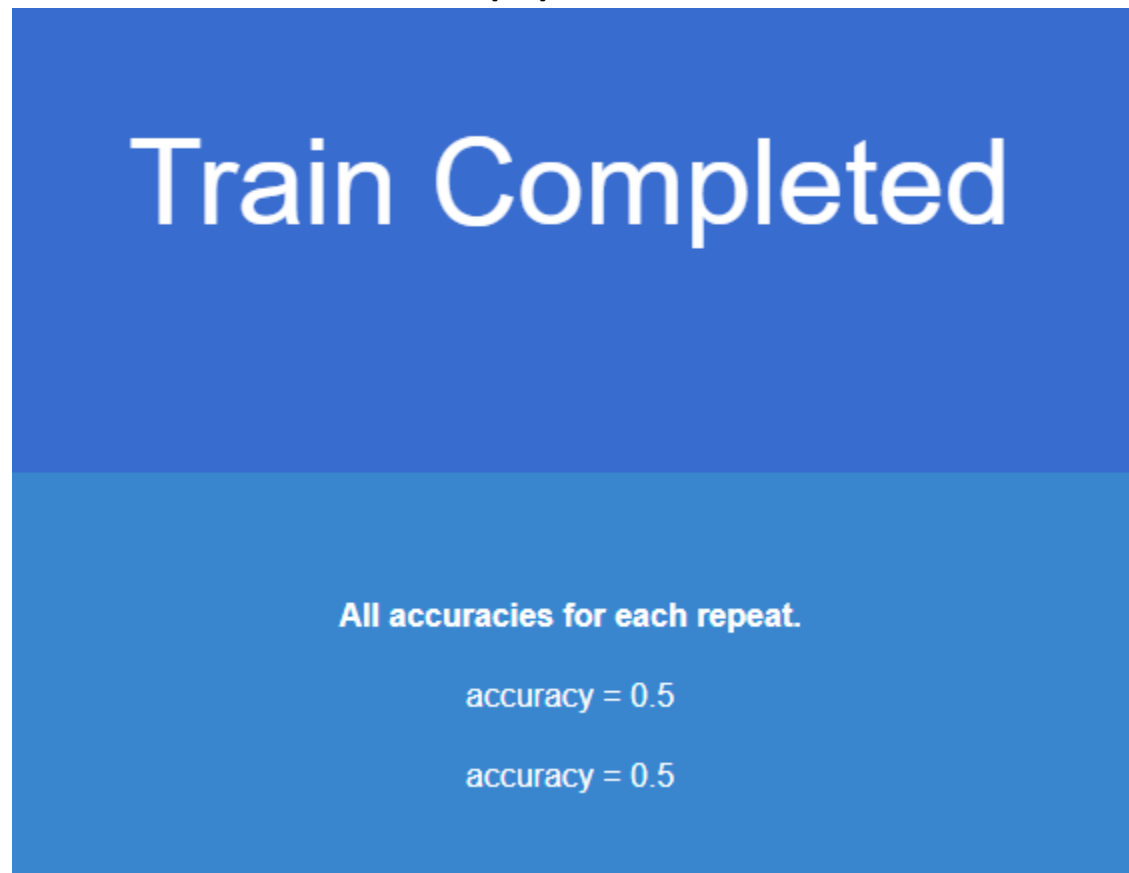


Εάν ο χρήστης επιλέξει ένα από τα διαθέσιμα μοντέλα που βρίσκονται στη σελίδα “Models”, τότε εμφανίζεται το προφίλ του μοντέλου “Model’s Profile” που διάλεξε. Εκεί, βρίσκονται πληροφορίες για το μοντέλο, για το πόσες κατηγορίες έχει, για το πόσες επαναλήψεις έχουν γίνει στην εκπαίδευση του μοντέλου και για το ποσοστό επιτυχίας που είχε στην τελευταία επανάληψη.

Επιπλέον, εδώ ο χρήστης έχει την δυνατότητα να επανεκπαιδεύσει το μοντέλο ή να το διαγράψει. Σε περίπτωση που θελήσει να επανεκπαιδεύσει το μοντέλο, θα του ζητηθεί να διαλέξει πόσες επαναλήψεις επιθυμεί να γίνουν. Στην συνέχεια κάνει “upload” τις εικόνες που θέλει να βάλει σε κάθε κατηγορία για να γίνει η επανεκπαίδευση, όπως γίνεται και στην εκπαίδευση, χωρίς να χρειαστεί να ξανα-ονομάσει τις κατηγορίες.

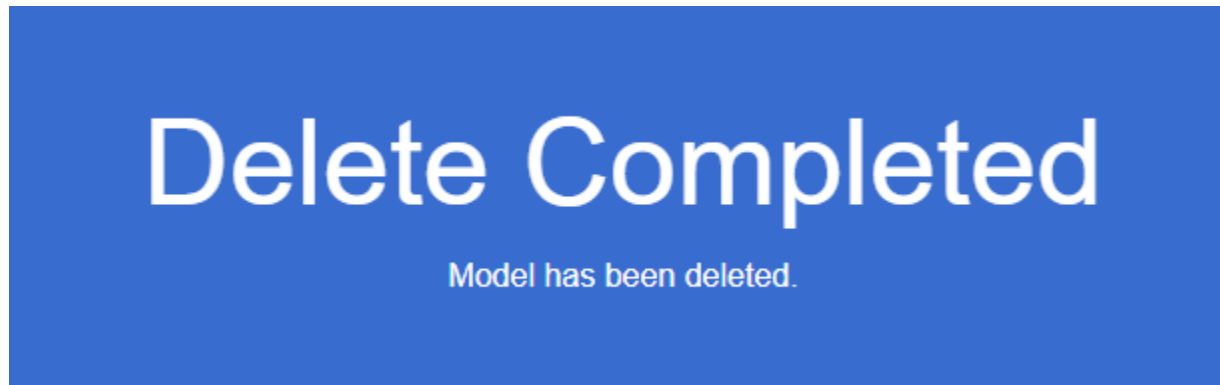
Όπως και στην εκπαίδευση, έτσι και στην επανεκπαίδευση, στο τέλος εμφανίζεται το ποσοστό επιτυχίας κάθε επανάληψης. Προσθέτουμε της επαναλήψεις που έκανε με τις επαναλήψεις που είχε κάνει σε άλλες εκπαιδεύσεις, ώστε να έχουμε τον συνολικό αριθμό επαναλήψεων που έχει πραγματοποιήσει το μοντέλο και αλλάζουμε το ποσοστό επιτυχίας με το τελευταίο ποσοστό επιτυχίας που έκανε σε αυτήν την επανεκπαίδευση.

6.5.2 Train/Retrain results (UI)



Με την ολοκλήρωση της εκπαίδευσης ή της επανεκπαίδευσης εμφανίζεται το μήνυμα “Train Completed” και από κάτω όλα τα ποσοστά επιτυχίας που είχε κάθε επανάληψη της εκπαίδευσης. Για παράδειγμα, στην παραπάνω εικόνα βλέπουμε πως το πρόγραμμα στην πρώτη και δεύτερη επανάληψη της εκπαίδευσης του μοντέλου ομαδοποίησε σωστά τις μισές εικόνες από αυτές που κάναμε “upload” για την εκπαίδευση. Δηλαδή 50% ποσοστό επιτυχίας.

6.5.3 Deleting Model (UI)



Σε περίπτωση που ο χρήστης θελήσει να διαγράψει ένα μοντέλο και έχει επιλέξει το κουμπί “Delete” στο προφίλ του μοντέλου που θέλει να διαγράψει, εμφανίζεται το παραπάνω μήνυμα, όπου σημαίνει ότι το μοντέλο πλέον έχει διαγραφτεί.

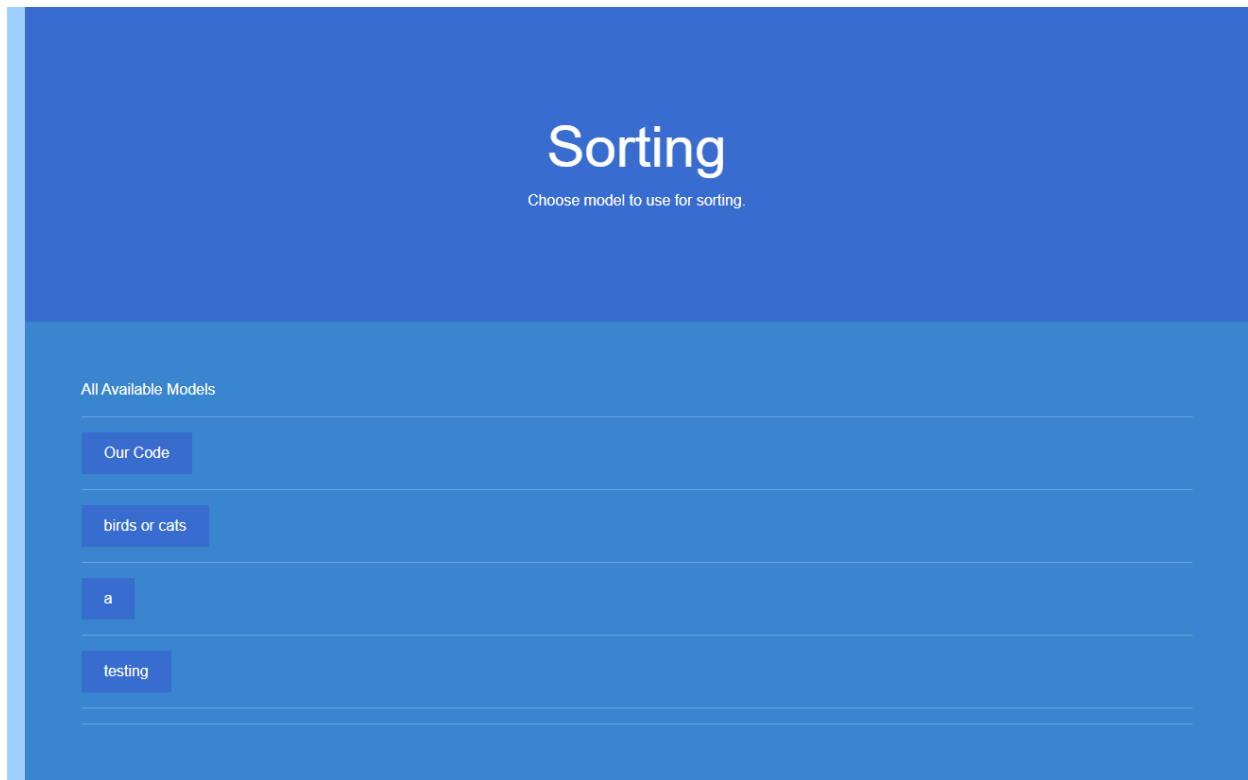
6.5.4 Κώδικας Deleting Model

```
@app.route('/delete')
def delete():
    import shutil
    global data
    conn = sqlite3.connect('model.db')
    c = conn.cursor()
    curmodel=data[0][0]
    print(curmodel)
    c.execute("DELETE FROM models WHERE model = ?", (curmodel,))
    c.execute("DELETE FROM categories WHERE model = ?", (curmodel,))
    conn.commit()
    conn.close()
    shutil.rmtree("C:\\Users\\Zouki\\Flask Folder Try 4\\
usermodel\\"+str(curmodel))
    return render_template('delete.html')
```

Σε αυτό το σημείο του κώδικα γίνεται η διαγραφή του μοντέλου. Αφού περάσουμε τις βιβλιοθήκες και την global μεταβλητή “data” στην οποία έχουμε της πληροφορίες του μοντέλου, δηλαδή το όνομά του, τον αριθμό των κατηγοριών, τον αριθμό των επαναλήψεων και το ποσοστό επιτυχίας που είχε στην τελευταία του επανάληψη. Παίρνουμε το “data [0] [0]”, όπου έχουμε το όνομα του μοντέλου που θέλουμε να διαγράψουμε, κάνουμε έλεγχο για να βρούμε όλες τις πληροφορίες για αυτό το μοντέλο που έχουμε στην βάσει δεδομένων και τα διαγράφουμε. Τέλος, διαγράφουμε τον φάκελο στον οποίο έχουμε το μοντέλο.

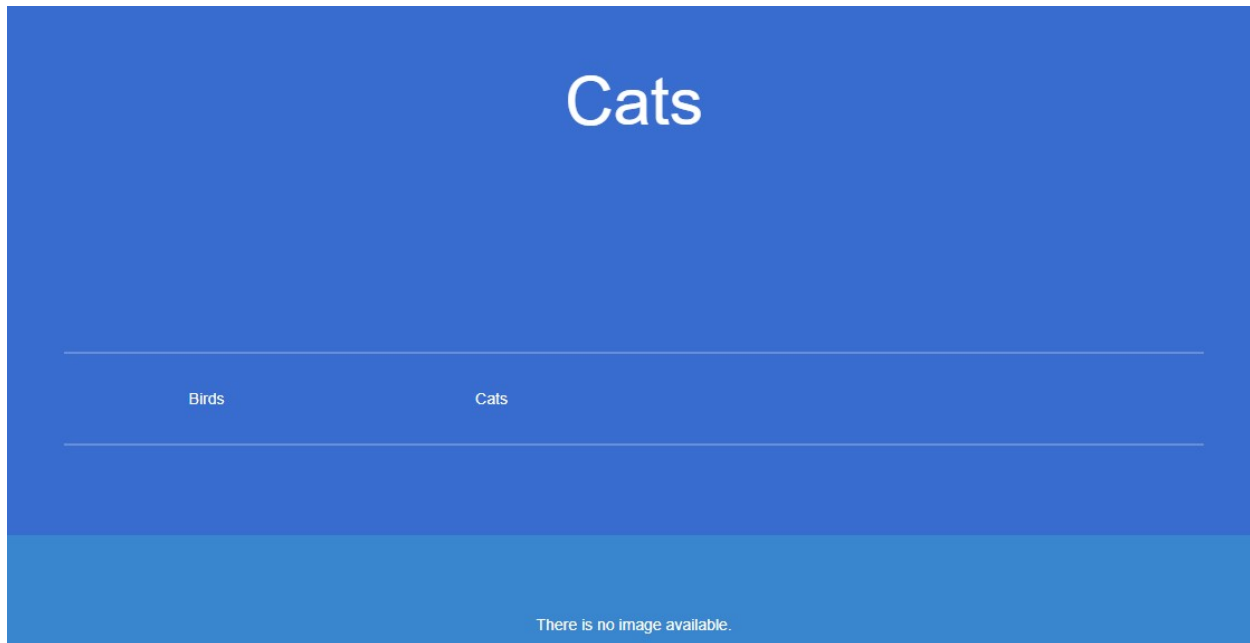
6.6 Sorting

6.6.1 Sorting (UI)



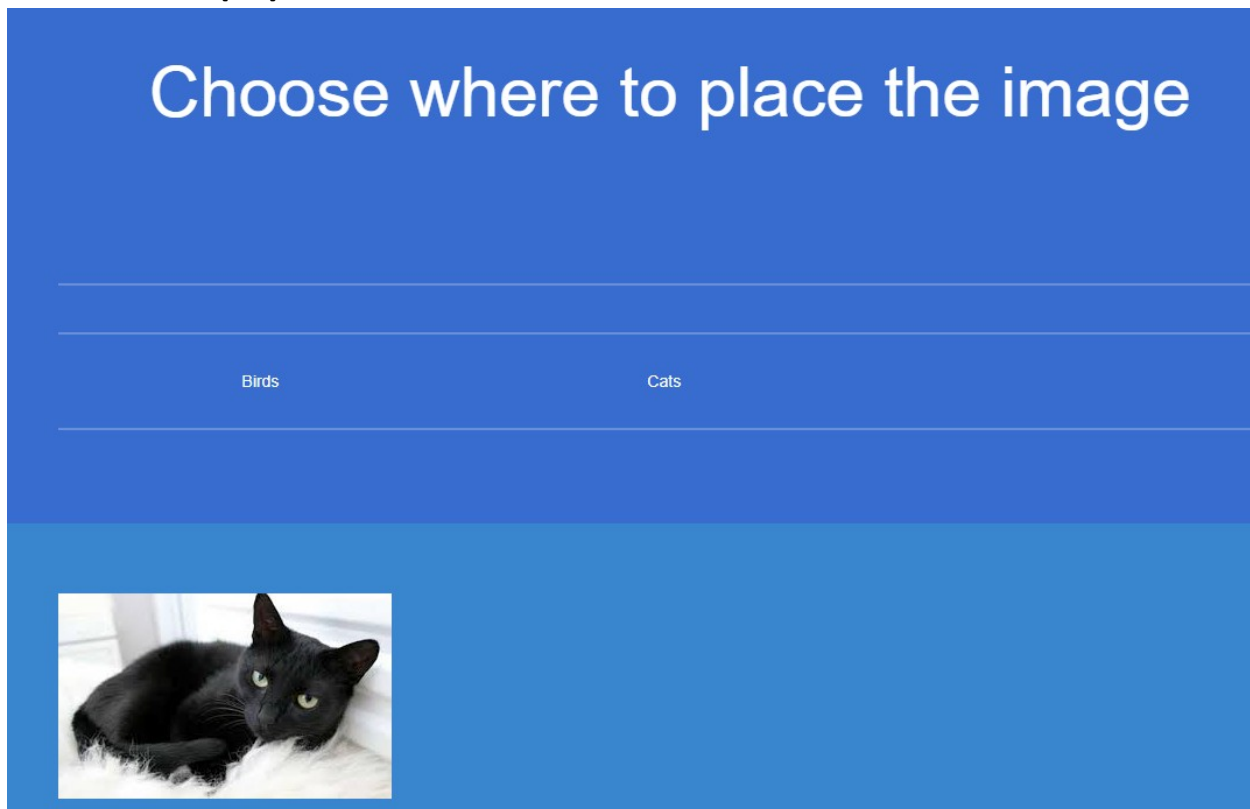
Πατώντας το κουμπί “Sorting”, το οποίο είναι ένα από τα βασικά κουμπιά που βρίσκονται στην κορυφή όλων των σελίδων, ο χρήστης οδηγείται σε αυτήν την σελίδα. Εδώ βρίσκονται όλα τα διαθέσιμα μοντέλα που μπορούν να χρησιμοποιηθούν για ομαδοποίηση. Σε αυτό το μέρος, εκτός από τα μοντέλα που ο χρήστης εκπαιδευσε, υπάρχουν και μοντέλα που δεν είναι εκπαιδευμένα από τον χρήστη και δεν έχει την δυνατότητα να τα επανεκπαιδεύσει ή να τα διαγράψει. Όταν ο χρήστης επιλέξει το μοντέλο που επιθυμεί να χρησιμοποιήσει, εμφανίζεται ένας κύκλος με “animation”, ως ένδειξη ότι γίνεται η ομαδοποίηση των εικόνων που ο χρήστης έκανε “upload” στην σελίδα “Image for Sorting” με το μοντέλο που διάλεξε. Όταν ολοκληρωθεί η ομαδοποίηση εμφανίζεται το μήνυμα “Sort Completed” και ένα κουμπί που γράφει “First Category”, ώστε να δει ο χρήστης τις εικόνες που ομαδοποιήθηκαν.

6.6.2 Categories (UI)



Στην συνέχεια εμφανίζεται το όνομα της κατηγορίας στο οποίο βρισκόμαστε, κουμπιά με τα ονόματα των κατηγοριών και οι εικόνες που έχουν ομαδοποιηθεί σε αυτήν την κατηγορία. Σε περίπτωση που δεν υπάρχουν φωτογραφίες, εμφανίζεται το μήνυμα "There is no image available.". Για παράδειγμα, σε αυτήν την εικόνα βρισκόμαστε στην κατηγορία "Cats" και επειδή δεν υπάρχουν εικόνες για αυτήν την κατηγορία, εμφανίζεται το μήνυμα "There is no image available." Αν υπήρχαν εικόνες, στην θέση του θα ήταν οι εικόνες που ανήκουν σε αυτήν την κατηγορία και επιλέγοντας στο κουμπί "Birds", ο χρήστης θα μπορούσε να μεταβεί σε αυτήν την κατηγορία.

6.6.3 Move (UI)



Αν ο χρήστης δεν συμφωνεί με την ομαδοποίηση του μοντέλου σε μία εικόνα, έχει την δυνατότητα να μετακινήσει την εικόνα σε άλλη κατηγορία. Πατώντας πάνω στην εικόνα που επιθυμεί να μετακινήσει, εμφανίζεται το μήνυμα “Choose where to place the image”, με τις κατηγορίες από κάτω και την εικόνα που διάλεξε ο χρήστης. Επιλέγοντας μία από τις κατηγορίες, η εικόνα μετακινείται στην επιλεγμένη κατηγορία.

6.6.4 Κώδικας Move

```
@app.route('/move/<filename>')
def move_image(filename):
    if (ctype == 0):
        imag_dir = os.path.join(APP_ROOT, 'images/')
        food_dir = os.path.join(APP_ROOT, 'food/')
        people_dir = os.path.join(APP_ROOT, 'people/')
        forest_dir = os.path.join(APP_ROOT, 'forest/')
        city_dir = os.path.join(APP_ROOT, 'city/')
        if (os.path.isfile(food_dir+filename)):
            src_dir=food_dir
        if (os.path.isfile(people_dir+filename)):
            src_dir=people_dir
        if (os.path.isfile(forest_dir+filename)):
            src_dir=forest_dir
        if (os.path.isfile(city_dir+filename)):
            src_dir=city_dir
        os.rename(src_dir+filename, imag_dir+filename)
        return render_template("move.html", image_name=filename)
```

Σε αυτό το σημείο ξεκινάει ο κώδικας για τη μετακίνηση των εικόνων. Αρχικά, ελέγχουμε αν η εικόνα που είναι να μετακινηθεί είναι από μοντέλο που ο χρήστης εκπαιδευσε. Αν η μεταβλητή “ctype” είναι ίση με 0 σημαίνει πως είμαστε στο μοντέλο “Our Code”, το οποίο δεν δημιουργήθηκε από τον χρήστη, οπότε οι κατηγορίες είναι “fixed”. Επιλέγοντας την εικόνα που ο χρήστης θέλει να μετακινήσει αυτόματα, η εικόνα μετακινείται στον φάκελο “image” μέχρι ο χρήστης να διαλέξει την κατηγορία που θέλει να μετακινήσει την εικόνα. Γι’ αυτό ελέγχουμε κάθε φορά που επιλέγουμε μία κατηγορία αν υπάρχει εικόνα στον φάκελο “image” για να μεταφέρουμε από εκεί στην επιλεγμένη κατηγορία.

```

else:
    global categ,mname
    icateg = 0
    imag_dir = os.path.join(APP_ROOT, 'images/')
    while (icateg <= categ):
        icateg = icateg + 1
        first_dir = ("C:\\Users\\Zouki\\Flask Folder Try 4\\folder" +
str(icateg)+ "/" )
        if (os.path.isfile(first_dir + filename)):
            src_dir=first_dir
            os.rename(src_dir+filename, imag_dir+filename)
            conn = sqlite3.connect('model.db')
            c = conn.cursor()
            c.execute("SELECT model from categories")
            print (mname)
            c.execute("SELECT number,categoryname FROM categories WHERE model
= ?", (mname,))
            conn.commit()
            cdata=c.fetchall()
            print("table = ",cdata)
            conn.close()
            return render_template("move2.html",image_name=filename,cdata=cdata)

```

Αν η μεταβλητή “ctype” δεν είναι ίση με το μηδέν, τότε σημαίνει πως το μοντέλο είναι εκπαιδευμένο από τον χρήστη, άρα η κατηγορίες είναι αποθηκευμένες στην βάση δεδομένων. Όπως και προηγουμένως, ελέγχουμε κάθε κατηγορία για να βρούμε σε ποια κατηγορία ανήκει η εικόνα και μετά τη μετακινούμε στον φάκελο “image” μέχρι ο χρήστης διαλέξει την κατηγορία που θέλει να μετακινήσει την εικόνα.

```

@app.route('/folder<foldername>')
def get_folder1(foldername):
    gallery_names = os.listdir('./images')
    if gallery_names:
        src_dir = os.path.join(APP_ROOT, 'images/')
        imag_dir = os.path.join(APP_ROOT, 'folder'+str(foldername)+'/')
        os.rename(src_dir+gallery_names[0], imag_dir+gallery_names[0])

```

Όταν ο χρήστης διαλέξει την κατηγορία, τότε μεταβαίνει στην σελίδα της κατηγορίας. Γι’ αυτό έχουμε έλεγχο στην αρχή του κώδικα της κατηγορίας για το αν υπάρχει κάποια εικόνα στον φάκελο “image”. Επειδή ο χρήστης μπορεί να μετακινήσει μία εικόνα την φορά, αν δεν είναι άδειος ο φάκελος “image”, σημαίνει πως υπάρχει μία εικόνα για μετακινήσει, γι’ αυτό μετακινεί την πρώτη φωτογραφία που υπάρχει στον φάκελο “image”, που είναι και η μοναδική, στην κατηγορία που μεταβαίνει ο χρήστης.

ΚΕΦΑΛΑΙΟ 7: ΕΓΚΑΤΑΣΤΑΣΗ ΠΡΟΓΡΑΜΜΑΤΩΝ

7.1 Anaconda Navigator



Το “Anaconda Navigator” είναι ένα γραφικό περιβάλλον διεπαφής χρήστη (GUI), το οποίο επιτρέπει τους χρήστες να ξεκινήσουν εφαρμογές και να διαχειριστούν πακέτα. Το “Anaconda Navigator” έρχεται με προ-εγκατεστημένες εφαρμογές, με μία από αυτές να είναι το “Jupyter Notebook” στο οποίο γράφτηκε ο κώδικας. Για την εγκατάσταση του Anaconda Navigator κατεβάσαμε το πρόγραμμα από την ιστοσελίδα [anaconda.com](https://www.anaconda.com/products/individual-b) το οποίο και εγκαταστήσαμε. (<https://www.anaconda.com/products/individual-b>) [26]

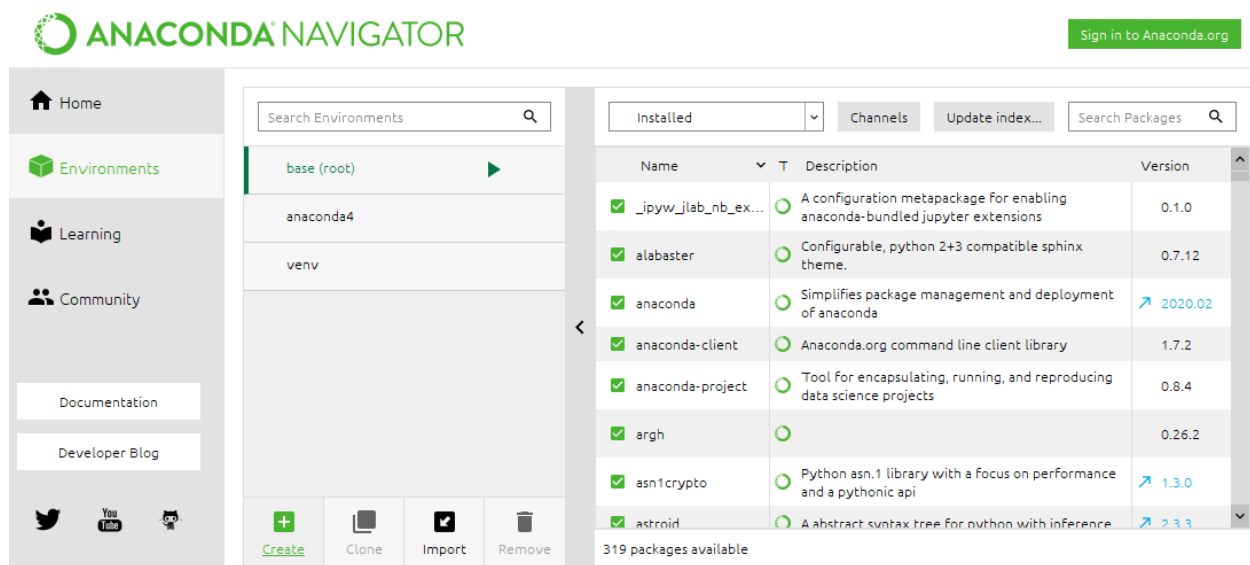
7.2 Jupyter Notebook



Το “Jupyter Notebook” είναι ένα διαδικτυακό δια-δραστικό υπολογιστικό περιβάλλον. Η μορφή των έγγραφων του είναι “JSON”. Το “Jupyter Notebook” περιέχει μία λίστα κελιών εισόδου/εξόδου, τα οποία μπορούν να περιέχουν κώδικα, κείμενο, μαθηματικά, γραφικές παραστάσεις και πολυμέσα. Συνήθως τα αρχεία τελειώνουν με “.ipynb”.

7.3 Δημιουργία Εικονικού Περιβάλλοντος (Virtual Environment)

Αφού εγκαταστήσαμε το “Anaconda Navigator” με το “Jupyter Notebook” εγκαταστημένο, δημιουργούμε ένα εικονικό περιβάλλον. Υπάρχουν δύο βασικοί τρόποι για την δημιουργία ενός εικονικού περιβάλλον. Ο πρώτος είναι με το “Anaconda Navigator”. Επιλέγοντας το κουμπί “Environments”, εμφανίζονται όλα τα διαθέσιμα εικονικά περιβάλλοντα και τα εγκατεστημένα πακέτα του καθενός. Στο κάτω μέρος υπάρχει και το κουμπί “Create” για την δημιουργία νέου εικονικού περιβάλλον.



Στην συνέχεια εμφανίζεται ένα νέο μικρό παραθυράκι για την εισαγωγή του ονόματος του νέου εικονικού περιβάλλον που επιθυμούμε να δημιουργήσουμε και τα πακέτα που θέλουμε να έχει.

Create new environment X

Name:

Location:

Packages: ☒ Python

☐ R

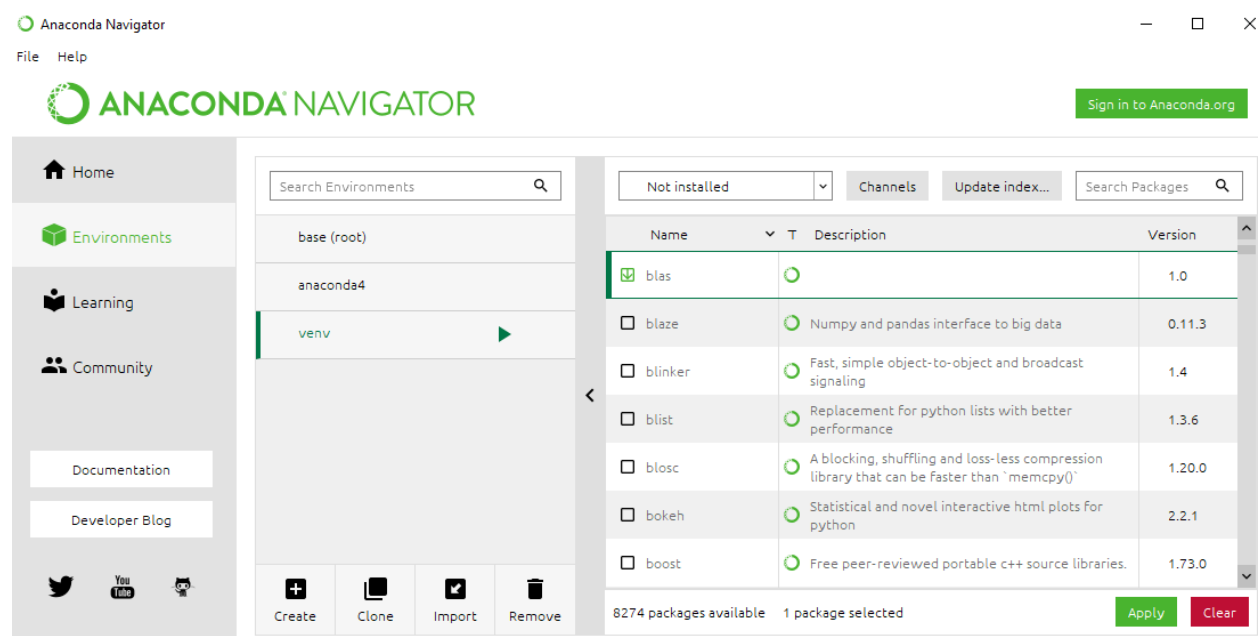
Ο άλλος τρόπος είναι μέσω του “Anaconda Prompt”, γράφοντας την εντολή “conda -n (name) pip python=3.7”. Στην θέση “(name)” γράφουμε το όνομα που θέλουμε να έχει το περιβάλλον και στο τέλος βάζουμε την έκδοση που θέλουμε να έχει η “python”.

```
Anaconda Prompt (anaconda3)

(base) C:\Users\Zouki>conda -n venv pip python=3.7
```

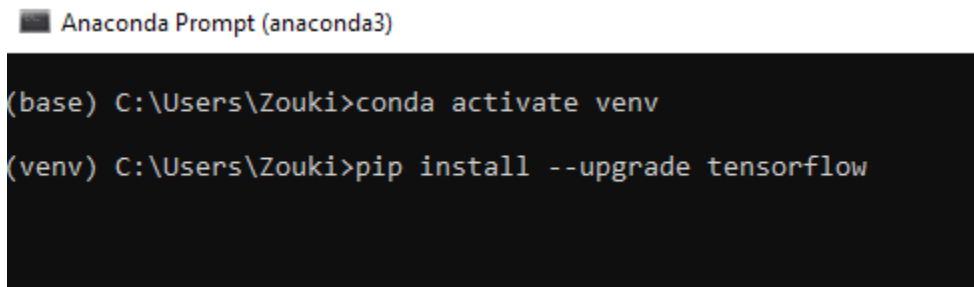
7.4 Εγκατάσταση Πακέτων

Αφού δημιουργήσαμε το εικονικό περιβάλλον, κάνουμε εγκατάσταση των πακέτων που χρειαζόμαστε. Υπάρχουν δύο τρόποι εγκατάστασης πακέτων. Ο πρώτος τρόπος είναι με το “Anaconda Navigator”. Επιλέγουμε το κουμπί “Environments”, όπου βρίσκονται όλα τα διαθέσιμα εικονικά περιβάλλοντα. Επιλέγουμε το εικονικό περιβάλλον που φτιάξαμε και στα δεξιά βρίσκουμε όλα τα εγκαταστημένα πακέτα. Αλλάζοντας το προεπιλεγμένο “Installed” σε “Not installed”, εμφανίζονται όλα τα πακέτα που δεν είναι εγκαταστημένα και μπορούμε να εγκαταστήσουμε. Γράφουμε στο “Search Packages” τα πακέτα που θέλουμε να βρούμε, δηλαδή “Keras”, “Tensorflow” και επιλέγουμε “Apply” για να γίνει η εγκατάσταση τους.



Ο άλλος τρόπος για να το πετύχουμε είναι ανοίγοντας το “Anaconda Prompt”. Εκτελούμε την εντολή “conda activate (name)”, με “(name)” το όνομα του εικονικού περιβάλλοντος που δημιουργήσαμε. Με αυτήν την εντολή ενεργοποιούμε το εικονικό περιβάλλον. Κάνοντάς το αυτό, μπροστά αντί για (base), γράφει σε παρένθεση το όνομα του εικονικού περιβάλλοντος. Έτσι ότι εγκατασταθεί, κάνουμε να γίνει μόνο σε αυτό το περιβάλλον.

Ένα από τα πακέτα που χρειαζόμαστε είναι το “TensorFlow”. Προκειμένου να γίνει η εγκατάστασή του, αρχικά χρειάζεται να έχουμε εγκατεστημένο το “Microsoft Visual C++ Redistributable” για “Visual Studio 2015, 2017, 2019”. Με την εντολή “pip install --upgrade tensorflow” κάνουμε εγκατάσταση το πακέτο “TensorFlow”. Για να εγκαταστήσουμε το πακέτο “Keras” χρησιμοποιούμε την εντολή “pip install keras”.

A screenshot of the Anaconda Prompt terminal window. The title bar at the top reads "Anaconda Prompt (anaconda3)". The terminal shows two commands being executed. The first command is "(base) C:\Users\Zouki>conda activate venv", which changes the environment to 'venv'. The second command is "(venv) C:\Users\Zouki>pip install --upgrade tensorflow", which upgrades the tensorflow package in the 'venv' environment.

```
Anaconda Prompt (anaconda3)

(base) C:\Users\Zouki>conda activate venv

(venv) C:\Users\Zouki>pip install --upgrade tensorflow
```

ΚΕΦΑΛΑΙΟ 8: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

8.1 Συμπεράσματα

Στην πτυχιακή αυτή εργασία δημιουργήθηκε web εφαρμογή για την ομαδοποίηση εικόνων. Ο κώδικας γράφτηκε με την Python και για την εμφάνιση html με Flask να τα συνδέει. Ο χρήστης έχει την δυνατότητα να εκπαιδεύσει ένα νέο μοντέλο με τη βοήθεια της μεθόδου του Transfer Learning, χρησιμοποιώντας το μοντέλο VGG16 ως το αρχικό μοντέλο. Με αυτόν τον τρόπο, ο χρήστης έχει την δυνατότητα να διαλέξει πόσες κατηγορίες θέλει να έχει το μοντέλο και να ορίσει κάθε κατηγορία. Αφού δημιουργήσει το μοντέλο μπορεί να το χρησιμοποιήσει για να ομαδοποιήσει τις εικόνες του.

Με τη μέθοδο Transfer Learning και τη διάθεση καλά εκπαιδευμένων μοντέλων, δεν ήταν αναγκαίο να έχουμε δυνατό υπολογιστή. Μετά από αρκετές εκπαιδεύσεις μοντέλων με την μέθοδο Transfer Learning και το VGG16 ως αρχικό μοντέλο έχουμε τα εξής συμπεράσματα:

- Για την δημιουργία μοντέλου με δύο ή τρεις απλές κατηγορίες (π.χ. γάτες, σκυλιά, καναρίνια), με λίγες επαναλήψεις (10-15) και λίγες φωτογραφίες (10-15 σε κάθε κατηγορία) η διαδικασία φτάνει σε ποσοστό επιτυχίας 85%.
- Όσο περισσότερες οι κατηγορίες, τόσο πιο δύσκολη είναι η εκπαίδευση. Όσο οι κατηγορίες αυξάνονται τόσο μεγαλώνει η πιθανότητα λάθους του μοντέλου. Γι' αυτό απαιτούνται περισσότερες φωτογραφίες για να αποσαφηνιστούν καλύτερα οι διαφορές των κατηγοριών και επειδή έχουμε περισσότερες φωτογραφίες και κατηγορίες χρειάζεται περισσότερος χρόνος για την εκπαίδευση του νέου μοντέλου.
- Όταν οι κατηγορίες έχουν μικρές διαφορές μεταξύ τους, τόσο πιο δύσκολη είναι η εκπαίδευση. Δηλαδή χρειάζονται περισσότερες φωτογραφίες για να ξεχωρίσουν καλύτερα οι διαφορές τους, άρα απαιτείται περισσότερος χρόνος εκπαίδευσης.
- Αυξάνοντας μόνο τις επαναλήψεις, δεν σημαίνει ότι θα αυξηθεί το ποσοστό επιτυχίας.
- Το ιδανικό είναι κάθε κατηγορία που χρησιμοποιούμε για την εκπαίδευση του μοντέλου να έχει τον ίδιο αριθμό εικόνων.
- Κάποιες φορές το μοντέλο μπορεί να παίρνει λάθος πληροφορίες για τις κατηγορίες. Για παράδειγμα, όταν δίνουμε ένα δείγμα εικόνων από φαγητά για να αναγνωριστούν ως μία κατηγορία, επειδή τα φαγητά διαφέρουν και είναι πολύ γενική κατηγορία, η διαδικασία καταλήγει να στηρίζεται μόνο στα πιάτα. Αυτό έχει ως αποτέλεσμα να μην αναγνωρίζεται κάτι ως φαγητό αν είναι πολύ κοντινή η εικόνα και δεν φαίνεται το πιάτο.

8.2 Προοπτικές

Το πρόγραμμα που δημιουργήθηκε θα μπορούσε να αποκτήσει και άλλα χαρακτηριστικά. Κάποια από αυτά είναι:

- Τη δυνατότητα αποθήκευσης της ομαδοποίησης που ο χρήστης έκανε. Δηλαδή να μπορεί ο χρήστης να κρατήσει έναν φάκελο που θα περιέχει τις κατηγορίες του μοντέλου ως φακέλους και τις εικόνες στις κατηγορίες (φακέλους) που ταξινομήθηκαν.
- Θα μπορούσε το πρόγραμμα να τρέχει στο διαδίκτυο και να υπάρχουν λογαριασμοί για κάθε χρήστη με τα μοντέλα που εκπάιδευσε.
- Την μετατροπή του κώδικα για android κινητά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]

Wikipedia, «Βικιπαίδεια», 18 3 2005. [Ηλεκτρονικό]. Available: <https://el.wikipedia.org/wiki/Python>.

[2]

Wikipedia, «Wikipedia», 29 10 2001. [Ηλεκτρονικό]. Available: [https://en.wikipedia.org/wiki/Python_\(programming_language\)#History](https://en.wikipedia.org/wiki/Python_(programming_language)#History).

[3]

Wikipedia, «Βικιπαίδεια», 25 12 2005. [Ηλεκτρονικό]. Available: <https://el.wikipedia.org/wiki/HTML>.

[4]

PythonBasics, «PythonBasics», 25 10 2010. [Ηλεκτρονικό]. Available: <https://pythonbasics.org/what-is-flask-python/#Why-is-Flask-a-good-web-framework-choice>.

[5]

Python, «Python», 2 10 2008. [Ηλεκτρονικό]. Available: <https://docs.python.org/3/library/os.html>.

[6]

Python, «Python», 30 3 2014. [Ηλεκτρονικό]. Available: <https://docs.python.org/3/library/shutil.html>.

[7]

Python, «Python», 30 3 2014. [Ηλεκτρονικό]. Available: <https://docs.python.org/3/library/glob.html>.

[8]

Python, «Python», 5 1 2006. [Ηλεκτρονικό]. Available: <https://docs.python.org/3/library/sqlite3.html>.

[9]

Wikipedia, «Wikipedia», 18 9 2016. [Ηλεκτρονικό]. Available: <https://en.wikipedia.org/wiki/Keras>.

[10]

Wikipedia, «Wikipedia», 9 11 2015. [Ηλεκτρονικό]. Available: <https://en.wikipedia.org/wiki/TensorFlow>.

[11]

w3schools, «w3schools», 17 3 2020. [Ηλεκτρονικό]. Available: https://www.w3schools.com/python/numpy/numpy_intro.asp.

[12]

onestop_data, «onestopdataanalysis», 24 6 2020. [Ηλεκτρονικό]. Available: <https://onestopdataanalysis.com/hdf5-in-python-h5py/>.

[13]

Wikipedia, «Wikipedia,» 4 7 2005. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Python_Imaging_Library.

[14]

w3schools, «w3schools,» 11 6 2019. [Ηλεκτρονικό]. Available: https://www.w3schools.com/python/module_requests.asp.

[15]

Wikipedia, «Wikipedia,» 20 12 2014. [Ηλεκτρονικό]. Available: [https://en.wikipedia.org/w/index.php?title=Requests_\(software\)&dir=prev&action=history](https://en.wikipedia.org/w/index.php?title=Requests_(software)&dir=prev&action=history).

[16]

Python, «Python,» 2 10 2008. [Ηλεκτρονικό]. Available: <https://docs.python.org/3/library/io.html>.

[17]

ThinkAutomation, «ThinkAutomation,» 28 7 2020. [Ηλεκτρονικό]. Available: <https://www.thinkautomation.com/eli5/eli5-what-is-image-classification-in-deep-learning/>.

[18]

GIS Geography, «GIS Geography,» 27 2 2021. [Ηλεκτρονικό]. Available: <https://gisgeography.com/image-classification-techniques-remote-sensing/>.

[19]

Wikipedia, «Wikipedia,» 2 8 2001. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&dir=prev&action=history.

[20]

Wikipedia, «Wikipedia,» 20 7 2011. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Deep_learning.

[21]

D. Sarkar, «towards data science,» 15 11 2018. [Ηλεκτρονικό]. Available: <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>.

[22]

Wikipedia, «Wikipedia,» 1 2 2006. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Transfer_learning.

[23]

M. u. Hassan, «Neurohive,» 20 11 2018. [Ηλεκτρονικό]. Available: <https://neurohive.io/en/popular-networks/vgg16/>.

[24]

J. Vijayabhaskar, «vijayabhaskar96,» 12 3 2018. [Ηλεκτρονικό]. Available: <https://vijayabhaskar96.medium.com/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720>.

[25]

Keras, «Keras,» 8 5 2020. [Ηλεκτρονικό]. Available: https://keras.io/api/callbacks/model_checkpoint/.

[26]

Wikipedia, «Wikipedia,» 8 6 2014. [Ηλεκτρονικό]. Available: [https://en.wikipedia.org/wiki/Anaconda_\(Python_distribution\)](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution))

