4η Εργασία
Όραση Υπολογιστών
Μπούρχα Ιωάννα 58019

Επιβλέπων καθηγητής: Ιωάννης Πρατικάκης
Επιβλέπων εργαστηρίου: Λάζαρος Τσοχατζίδης

Ακαδημαϊκό έτος: 2022 - 2023

ΠΕΡΙΕΧΟΜΕΝΑ

Πρόλογος

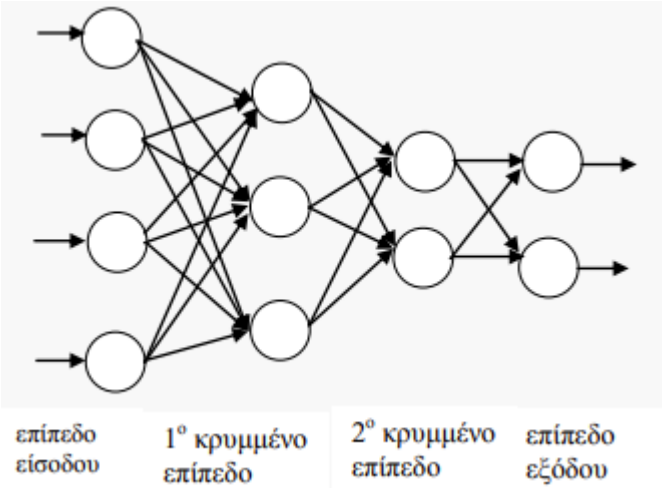
Στην παρούσα εργασία ζητείται η υλοποίηση συνελικτικού νευρωνικού δικτύου για την ταξινόμηση των εικόνων. Οι εικόνες προέρχονται από πλατφόρμα Kaggle και συγκεκριμένα από την ιστοσελίδα: https://vc.ee.duth.gr:6960/index.php/s/wlnkxtlGmqBeATC/download

Θα υλοποιηθούν δύο δίκτυα, ένα μη προεκπαιδευμένο και ένα προεκπαιδευμένο δίκτυο, για τα οποία θα γίνει παρουσίαση της αρχιτεκτονικής τους και ποσοτική εκτίμηση της επίδοσής τους. Ύστερα, θα τροφοδοτηθούν και με το dataset της 3η εργασίας.

Θεωρητικό Υπόβαθρο

Ως νευρωνικό δίκτυο στην επιστήμη των υπολογιστών ονομάζουμε ένα δίκτυο από υπολογιστικούς συνδεδεμένους κόμβους (νευρώνες). Οι νευρώνες αποτελούν τα δομικά στοιχεία, καθώς δέχονται σύνολο αριθμητικών δεδομένων τα οποία επεξεργάζονται και παράγουν αποτελέσματα. Τα δεδομένα εισόδου προέρχονται είτε από το περιβάλλον (χρήστης) είτε από τους νευρώνες τους προηγούμενου επιπέδου (layer). Τα δεδομένα εξόδου προωθούνται στους νευρώνες του επόμενου επιπέδου ή στο περιβάλλον. Στην δεύτερη περίπτωση αποτελούν και τα τελικά αποτελέσματα του δικτύου μας.

Συνεπώς, οι νευρώνες μπορούν να χωριστούν σε νευρώνες εισόδου, υπολογιστικούς νευρώνες (κρυφοί) και νευρώνες εξόδου. Από αυτούς, ιδιαίτερο ενδιαφέρον παρουσιάζουν οι κρυφοί νευρώνες. Οι πράξεις που υλοποιούν είναι ο υπολογισμός του αθροίσματος του πολλαπλασιασμού των δεδομένων εισόδων τους με το αντίστοιχο βάρος (weight). Το βάρος προσδιορίζει πόσο δυνατά είναι συνδεδεμένοι οι δύο νευρώνες και παίρνει τιμές διαφορετικές για κάθε σήμα εισόδου που κυμαίνονται από -1 έως 1. Το αποτέλεσμα του αθροίσματος τροφοδοτείται στην συνάρτηση ενεργοποίησης, η οποία υλοποιείται στο εσωτερικό των νευρώνων. Το αποτέλεσμα αυτής της συνάρτησης τροφοδοτείται στους νευρώνες του επόμενου επιπέδου.



Εικόνα 1: Παράδειγμα Νευρωνικού Δικτύου

Στόχος των νευρωνικών δικτύων είναι η ανάπτυξη εσωτερικής δομής με σκοπό την αναγνώριση προτύπων που μοιάζουν με εκείνα με τα οποία εκπαιδεύτηκαν.

Εκπαίδευση Νευρωνικών Δικτύων

Η εκπαίδευση των νευρωνικών δικτύων αναφέρεται στην αλλαγή των τιμών των βαρών των συνδεδεμένων νευρώνων. Οι αλλαγές αυτές αποσκοπούν στην αύξηση της επιτυχούς αναγνώρισης προτύπων όμοια με εκείνα με τα οποία εκπαιδεύτηκε, δηλαδή την αύξηση της απόδοσης του δικτύου και την καλύτερη εκμάθηση.

Η αλλαγή των τιμών δεν γίνεται πάντα με τον ίδιο τρόπο, εξαρτάται σημαντικά από την μέθοδο που χρησιμοποιούμε. Οι μέθοδοι αυτές είναι:

- Μάθηση με επίβλεψη – Supervised Learning:

Οι τιμές των βαρών τροποποιούνται σε σχέση με το σφάλμα του αποτελέσματος εξόδου του δικτύου με το επιθυμητό. Το εξωτερικό περιβάλλον παρέχει τα επιθυμητά αποτελέσματα για κάθε διάνυσμα εισόδου. Με άλλα λόγια το εξωτερικό περιβάλλον λειτουργεί ως «δάσκαλος».

Ειδική περίπτωση αποτελεί η μέθοδος της ενισχυτικής μάθησης (reinforcement learning) όπου το εξωτερικό περιβάλλον παρέχει μόνο την πληροφορία ότι το αποτέλεσμα του δικτύου είναι σωστό ή όχι, δεν παρέχεται η σωστή απάντηση. Εδώ, το εξωτερικό περιβάλλον λειτουργεί ως «κριτής».

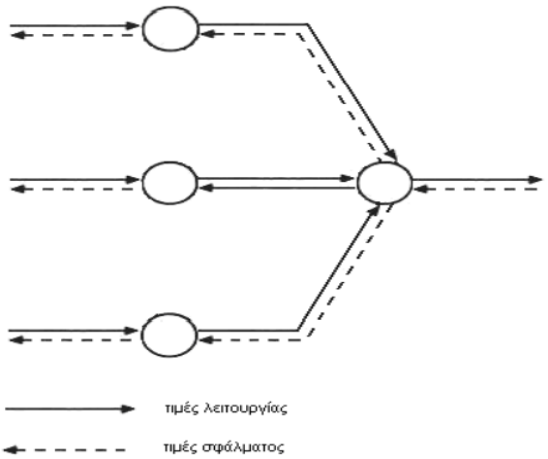
- Μάθηση χωρίς επίβλεψη – Unsupervised Learning:

Οι τιμές των βαρών τροποποιούνται με βάση τα δεδομένα εισόδου. Το εξωτερικό περιβάλλον δεν παρέχει ούτε τα επιθυμητά αποτελέσματα ούτε το εάν αυτά του δικτύου είναι σωστά ή όχι.

Αλγόριθμος Υπολογισμού Βαρών

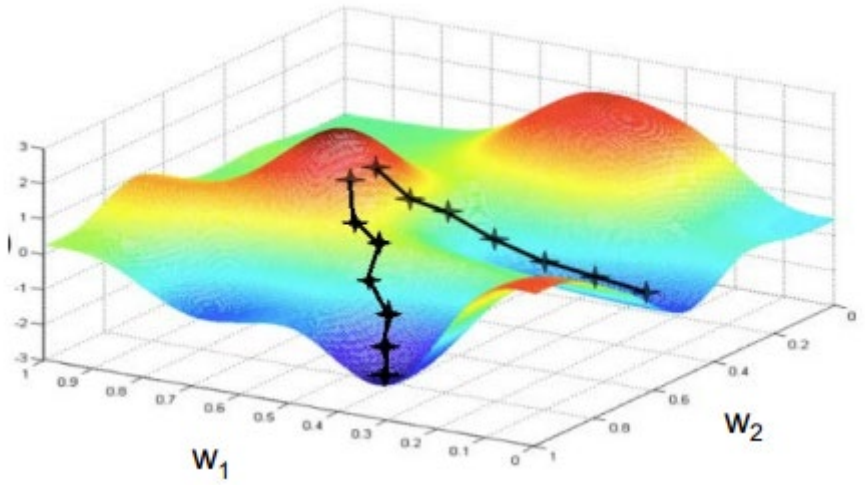
Ο αλγόριθμος που θα αξιοποιηθεί στην παρούσα εργασία για τον τελικό υπολογισμό των τιμών των βαρών είναι ο αλγόριθμος Backpropagation (αλγόριθμος οπισθοδιάδοσης). Εφαρμόζεται σε δίκτυα πολλών επιπέδων (layers) με συνεχείς συναρτήσει ενεργοποίησης που διαδίδονται προς τα μπροστά. Η συνέχεια της συνάρτησης ενεργοποίησης είναι σημαντικό στοιχείο είναι απαραίτητο στοιχείο προκειμένου να είναι δυνατή η παραγώγισή της.

Ο αλγόριθμος αποτελείται από δύο «περάσματα» όλων των επιπέδων του δικτύου: το ευθύ και το ανάδρομο. Στο ευθύ πέρασμα (forward pass) οι τιμές των βαρών παραμένουν ως έχουν και υπολογίζεται το τελικό αποτέλεσμα. Στο τελικό επίπεδο (layer) υπολογίζεται το σφάλμα του αποτελέσματος του δικτύου με το επιθυμητό για κάθε νευρώνα. Στο ανάδρομο πέρασμα (reverse pass) ξεκινώντας από το τελικό επίπεδο με δεδομένο τις τιμές σφάλματος υπολογίζεται αναδρομικά η τιμή της τοπικής κλήσης της εισόδου του νευρώνα για κάθε νευρώνα του δικτύου. Έτσι, τροποποιούνται οι τιμές των βαρών.



Εικόνα 2: Αναπαράσταση ευθύ και ανάδρομου περάσματος

Στόχος, προφανώς, είναι η ελαχιστοποίηση του σφάλματος. Για την ελαχιστοποίηση της συνάρτησης σφάλματος σε σχέση με τα βάρη του δικτύου αξιοποιείται η μέθοδος gradient descent, σύμφωνα με την οποία η ελαχιστοποίηση της συνάρτησης γίνεται με τις κατευθύνσεις αναζήτησης στην παράγωγό της. Αξίζει να σημειωθεί ότι δεν δίνεται πάντα η βέλτιστη λύση, καθώς υπάρχει η περίπτωση να παγιδευτεί σε τοπικά ελάχιστα.



Εικόνα 3: Κατευθύνσεις ελαχιστοποίησης στην συνάρτηση σφάλματος

Η σύγκλιση του αλγορίθμου εξαρτάται από:

- τα αρχικά βάρη,
- την τιμή του σφάλματος,
- τον κανόνα αναπροσαρμογής,
- το σύνολο εκμάθησης,
- την αρχιτεκτονική.

Η ανανέωση των τιμών των βαρών καθορίζεται από τον optimizer. Από τους διαθέσιμους που αξιοποιούν την μέθοδο gradient descent, στην παρούσα εργασία θα χρησιμοποιηθεί ο Adam, όπως ακριβώς και στα εργαστήρια του μαθήματος.

Οι αλλαγές των τιμών των βαρών, δηλαδή το πόσο θα κατέβουμε στην συνάρτηση κόστους, καθορίζεται από τον παράγοντα learning rate (ρυθμός εκμάθησης). Ο παράγοντας αυτός πολλαπλασιάζεται με το gradient το αποτέλεσμα είναι η αλλαγή στην τιμή των βαρών. Εάν το learning rate είναι πολύ μεγάλο ενδέχεται να μην καταλήξουμε στην βέλτιστη λύση, ενώ αν είναι

πολύ μικρό υπάρχει η πιθανότητα να εγκλωβιστούμε σε τοπικό ακρότατο. Ύστερα από συζήτηση και δοκιμές επιλέχτηκε η τιμή 0,0001.

Βελτίωση αποτελεσμάτων

Προκειμένου να έχουμε το καλύτερο δυνατό αποτέλεσμα, ο αλγόριθμός μας είναι επαναληπτικός. Σε αυτό το σημείο εξηγούνται ορισμένοι βασικοί σχετικοί όροι.

Εποχή – Epoch:
Θεωρούμε ότι έχουμε ολοκληρώσει μία εποχή όταν όλα τα δεδομένα του συνόλου εκπαίδευσης (training set) έχουν περάσει από το δίκτυό μας μία φορά τόσο για το ευθύ όσο και για το ανάδρομο πέρασμα.

Ομάδα ή παρτίδα - Batch:
Σύνολο δεδομένων, υποσύνολο του συνόλου εκπαίδευσης, το οποίο εισάγεται στο δίκτυο. Η ανανέωση των τιμών των βαρών γίνεται στο τέλος κάθε batch.

Η επιλογή των κατάλληλων τιμών για αυτές τις παραμέτρους γίνεται μέσω δοκιμής του δικτύου.

Συνελικτικά Νευρωνικά Δίκτυα – Convolutional Neural Networks (CNN)

Τα δίκτυα αυτά, κατηγορία των νευρωνικών δικτύων, έχουν αποδειχθεί ιδιαιτέρως αποτελεσματικά σε προβλήματα ταξινόμησης εικόνων, και για αυτό πρόκειται να αξιοποιηθούν για την υλοποίηση της παρούσας εργασίας.

Ένα συνελικτικό δίκτυο (ΣΝΔ) αποτελείται από συνελικτικά επίπεδα, τα οποία είναι ένα σύνολο νευρώνων που εκτελεί συνέλιξη των δεδομένων εισόδου τους με προκαθορισμένα φίλτρα. Όπως και στα μη συνελικτικά δίκτυα, κάθε επίπεδο έχει διακριτές εισόδους και εξόδους. Τα συνελικτικά φίλτρα σαρώνουν τα δεδομένα εισόδου κάθε επιπέδου και δημιουργούν έναν πίνακα δύο διαστάσεων ο οποίος περιέχει τα εσωτερικά γινόμενα των δεδομένων που σαρώνονται κάθε φορά. Ο πίνακας αυτός αποτελεί τον χάρτη χαρακτηριστικών εισόδου. Αυτός τροφοδοτείται στην συνάρτηση ενεργοποίησης και εξάγεται ένας μη γραμμικός μετασχηματισμός του.

Οι διαστάσεις των φίλτρων που χρησιμοποιούνται είναι: το ύψος, το πλάτος και το βάθος τους. Συνήθως, το βάθος ταυτίζεται με εκείνο των δεδομένων εισόδου. Κατά κανόνα, το συνελικτικό φίλτρο είναι μικρότερων διαστάσεων από την είσοδο. Προφανώς, οι διαστάσεις αυτές δύναται να διαφοροποιούνται από επίπεδο σε επίπεδο.

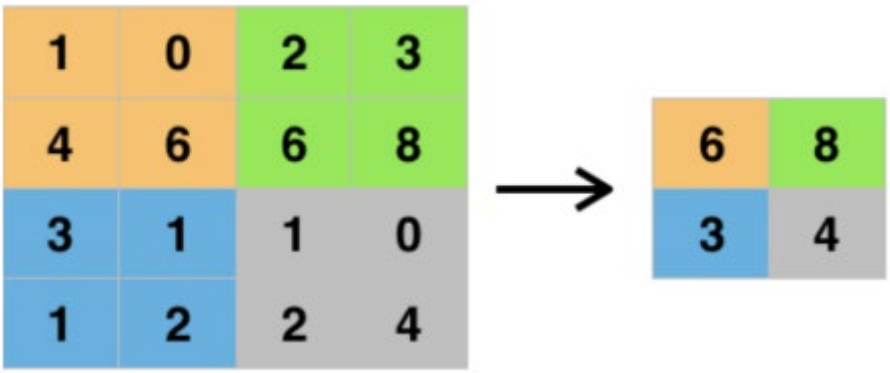
Τα επίπεδα που εκτελούν μονάχα συνέλιξη ονομάζονται συνελικτικά (convolutional layers). Αυτά συνήθως τοποθετούνται στην αρχή του ΣΝΔ προκειμένου να αφαιρεθεί η πολυπλοκότητα της εισόδου.

Πέρα από την συνέλιξη, ένα επίπεδο μπορεί να εκτελέσει και άλλες διεργασίες όπως η χωρική υποδειγματοληψία (pooling layers). Η διαδικασία αυτή εφαρμόζεται μεταξύ δύο διαδοχικών συνελικτικών επιπέδων αποσκοπώντας στην περεταίρω μείωση των διαστάσεων της εισόδου. Απόρροια αυτών, είναι η μείωση της πιθανότητας υπερεκαπίδευσης του δικτύου μας (overfitting). Ωστόσο, δημιουργεί προβλήματα στην γενίκευση (underfitting) και για αυτό η χρήση τους γίνεται με ιδιαίτερη προσοχή.

Για την επίτευξη της υποδειγματοληψίας χρησιμοποιούνται κυρίως οι παρακάτω συναρτήσεις:

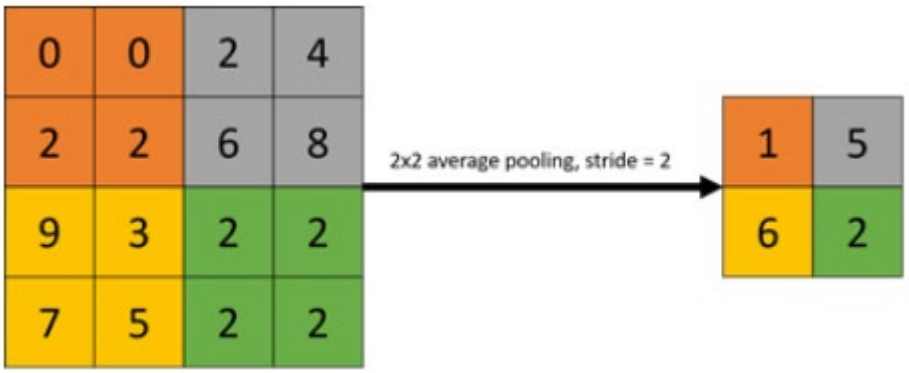
- Εξαγωγή τοπικού μέγιστου – Max Pooling

Από την περιοχή στην οποία εφαρμόζεται το φίλτρο κρατώ την μέγιστη τιμή.



- Εξαγωγή τοπικού μέσου όρου – Average pooling:

Από την περιοχή στην οποία εφαρμόζεται το φίλτρο κρατώ το μέσο όρο.



Ακόμη, υπάρχει και το επίπεδο απόρριψης (dropout layer). Το επίπεδο αυτό απορρίπτει με μία πιθανότητα (dropout rate) ορισμένους νευρώνες κατά την διάρκεια της εκπαίδευσης. η πιθανότητα αυτή κυμαίνεται από 0,1 έως 0,5. Οι νευρώνες αυτοί διαφέρουν από εποχή σε εποχή. Απόρροια αυτής της απόρριψης είναι να αφαιρούνται, προσωρινά, τόσο από το μπροστινό όσο και από το ανάδρομο πέρασμα, με αποτέλεσμα να μην ενημερώνονται τα βάρη τους. Η χρήση αυτού του επιπέδου αποτελεί μέτρο για την αντιμετώπιση του προβλήματος υπερεξιδίκευσης (overfitting). Για χρήση πολλών τέτοιων επιπέδων σε διάφορα σημεία του δικτύου συνήθως υιοθετείται η μείωση της πιθανότητας από τα πρώτα προς τα τελευταία επίπεδα.

Τέλος, προκειμένου να δημιουργηθεί το μοντέλο απόφασης, τοποθετείται ένα πλήρως συνδεδεμένο επίπεδο (fully connected ή Dense), κάθε νευρώνας του οποίου συνδέεται με κάθε νευρώνα του προηγούμενου επιπέδου. Για το τελευταίο επίπεδο η συνάρτηση ενεργοποίησης

δεν είναι η ReLU, αλλά η softmax. Η έξοδος ενός fully connected επιπέδου αναπαριστά χαρακτηριστικά υψηλών στρωμάτων, τα οποία χρησιμοποιούνται προκειμένου να κατηγοροποιηθεί η εικόνα εισόδου. Η συνάρτηση ενεργοποίησης αυτού του επιπέδου, μετατρέπει τα αποτελέσματα αυτά σε πιθανότητες, των οποίων το συνολικό άθροισμα είναι 1, και η εικόνα κατηγοροποιείται στην κλάση με την μέγιστη πιθανότητα.

Κατά κανόνα, ανάμεσα στα προηγούμενα επίπεδα και τα πλήρως συνδεδεμένα υπάρχει ένα επίπεδο επίπεδο (flatten layer) το οποίο μετατρέπει τα τρισδιάστατα δεδομένα εξόδου των πρώτων σε μονοδιάστατα δεδομένα εισόδου των δεύτερων.

Επιπλέον, προκειμένου να μην κολλήσει το μοντέλο σε κάποια κακή τοπική βέλτιστη κατάσταση, αξιοποιείται ένα επίπεδο κανονικοποίησης (batch normalization). Σύμφωνα και με το αγγλικό όνομα, η κανονικοποίηση γίνεται σε κάθε παρτίδα. Η κανονικοποίηση γίνεται με την αφαίρεση από τα δεδομένα εισόδου τον μέσο όρο του και έπειτα διαιρώντας με την τοπική απόκλιση για κάθε παρτίδα. Συνεπώς, το επίπεδο αυτό εξυπηρετεί στην σταθεροποίηση της εκπαίδευσης και στην αύξηση της απόδοσης.

Εισαγωγή δεδομένων

Αρχικά, εισάγω τις βιβλιοθήκες που θα χρειαστώ:

```
import numpy as np
import cv2 as cv
import os
import tensorflow as tf
```

Έπειτα, εισάγω τις εικόνες προς ταξινόμηση και προσδιορίζω εκείνες που θα αξιοποιηθούν στην εκπαίδευση και στην αξιολόγηση. Οι δοθείσες εικόνες απεικονίζουν σήματα οδήγησης και είναι ταξινομημένες σε φακέλους ανάλογα με το περιεχόμενό τους.

```
import zipfile
!rm /content/download
!rm -r /content/imagedb/
!rm -r /content/imagedb_test
!wget https://vc.ee.duth.gr:6960/index.php/s/wlnkxtlGmqBeATC/download
local_zip = '/content/download'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

```
# Declaring training and testing directories
train_dir = '/content/imagedb'                # train_folders
test_dir  = '/content/imagedb_test'           # testing_folders
```

Στην συνέχεια προσδιορίζω τον αριθμό των συνολικών κλάσεων στις οποίες είναι ταξινομημένα τα δεδομένα μου.

```
## Auto find the nodes of the last layer ##
```

```
classes = []
for dirname, _, filenames in os.walk('/content/imagedb_test'):
  # print(dirname)
  a,folder = dirname.split("/content/imagedb_test")
  folder = folder[1:]
  # print(folder)
  classes.append(folder)

classes = classes[1:]
print(classes)

out_nodes = len(classes)
print("\nThe nodes of the last layer and the number of clas-
ses is: " + str(out_nodes))
```

Στην παρούσα εργασία εφαρμόζεται η τεχνική εκπαίδευσης και πιστοποίησης (cross validation). Το σύνολο δεδομένων εκπαίδευσης χωρίζεται σε δύο υποσύνολα: στο σύνολο εκπαίδευσης (training set) και στο σύνολο επικύρωσης (validation set). Αρχικά, στο δίκτυο τροφοδοτείται το training set και εκπαιδεύεται, δηλαδή μαθαίνει τα χαρακτηριστικά των δεδομένων. Στο τέλος κάθε εποχής εκπαίδευσης, αξιοποιεί τα χαρακτηριστικά αυτά και κατηγοροποιεί τις εικόνες του validation set. Στο τέλος κάθε εποχής παρουσιάζεται το σφάλμα τόσο για το στάδιο της εκπαίδευσης (training) όσο και για εκείνο της επικύρωσης (validation). Τονίζεται ότι το δεύτερο σφάλμα θα έχει μεγαλύτερη τιμή του πρώτου. Η επίδοση του δικτύου στο validation set υποδεικνύει την ικανότητα γενίκευσής του, δηλαδή πόσο καλά ανταποκρίνεται σε δεδομένα με τα οποία δεν έχει εκπαιδευτεί αλλά ανήκουν στα δεδομένα του προβλήματος. Με άλλα λόγια αντιμετωπίζουμε το πρόβλημα της υπερεξειδίκευσης (overfitting). Στόχος μας είναι η επίδοση του δικτύου στο σύνολο αξιολόγησης (testing) να είναι παρόμοια με εκείνη της επικύρωσης (validation).

Όπως έγινε φανερό από την θεωρητική ανάλυση, απαιτείται ο προσδιορισμός ορισμένων παραμέτρων. Οι παράμετρες αυτοί καθώς και οι αντίστοιχες τιμές τους παρουσιάζονται στο ακόλουθο κομμάτι κώδικα.

```
nodes = 32
epochs = 600
do = 0.5  # drop out rate
bs = 20   # batch_size      # κρατάω σταθερό
ks = 3    # kernel size
image_dimension = 256
```

Για την παράμετρο image_dimension επιλέχτηκε κάθε φορά η default τιμή της.

Επαύξηση του συνόλου εκπαίδευσης

Τα ΣΝΔ πολλές φορές αποτυγχάνουν να κατηγοριοποιήσουν ορθά μετασχηματισμούς των δεδομένων εισόδου. Εάν ένα ΣΝΔ εκπαιδευτεί να ταξινομεί ένα αντικείμενο κάτω από μια συγκεκριμένη γωνία, κατά πάσα πιθανότητα θα αποτύχει να το ταξινομήσει όταν του δοθεί, έστω και ελαφρώς, περιστραμμένο, παρόλο που πρόκειται για το ίδιο αντικείμενο. Η ίδια υπόθεση ισχύει και για οριζόντια αναστροφή του αντικειμένου, λοξό αντικείμενο ή ακόμα και ελαφρώς μετατοπισμένο σε μια διαφορετική θέση εντός της περιοχής εισόδου.

Για την αντιμετώπιση αυτού του προβλήματος παράγω επιπλέον τεχνητά δεδομένα εκπαίδευσης μέσω εφαρμογής μετασχηματισμών. Οι επιλεχθείς μετασχηματισμοί προέκυψαν αναλογιζόμενη τις διαφορετικές συνθήκες απεικόνισης των σημάτων οδήγησης.

```
# DATA AGMENTATION
from keras.preprocessing.image import ImageDataGenerator

train_data_gen = ImageDataGenerator(rescale = 1./255,
                                    rotation_range = 10,
                                    brightness_range = [0.5 , 1],
                                    zoom_range = [0.15 , 1],
                                    validation_split = 0.2,
                                    horizontal_flip=False,
                                    vertical_flip=False,
                                    fill_mode="nearest")

# --------------------
# Flow training images in batches of bs using train_data_gen generator
# --------------------
train_generator = train_data_gen.flow_from_directory(train_dir,
                                                     batch_size=bs,
                                                     class_mode='categorical',
                                                     # color_mode='grayscale',
                                                     target_size=(image_dimension, im-
age_dimension),
                                                     shuffle=True,
                                                     subset='training', seed=1)
# --------------------
# Flow validation images in batches of bs using train_data_gen generator
# --------------------
validation_generator = train_data_gen.flow_from_directory(train_dir,
                                                       batch_size=100,
                                                       class_mode='categorical',
                                                       # color_mode='grayscale',
                                                       target_size=(image_dimen-
sion, image_dimension),
                                                       subset='validation', seed=1)
```

Η εντολή `validation_split = 0.2` καταχωρεί ως δεδομένα εκπαίδευσης (training set) το 80% των εικόνων που βρίσκονται στην τοποθεσία train_dir, ενώ τα υπόλοιπα 20% ως δεδομένα πιστοποίησης (validation set).

Η παράμετρος target_size ορίζει τις διαστάσεις των εικόνων στις οποίες θα εφαρμοστούν οι μετασχηματισμοί και θα δοθούν τελικά ως είσοδοι στο δίκτυο. Όσο μεγαλύτερες οι διαστάσεις του δικτύου τόσο πιο βαθύ, δηλαδή τόσο μεγαλύτερο το πλήθος των επιπέδων που έχει, θα πρέπει να είναι το δίκτυο.

Η παράμετρος batch_size προσδιορίζει τις ομάδες των εικόνων με τις οποίες εκπαιδεύεται το δίκτυο σε κάθε εποχή. Παρατηρήθηκε ότι όσο πιο μεγάλη τιμή έχει η παράμετρος αυτή, τόσο πιο βαθύ, πρέπει να είναι και το δίκτυό. Ύστερα από συζήτηση στα πλαίσια του τελευταίου εργαστηρίου προτάθηκε και υιοθετήθηκε η τιμή 20.

Επιλογή Αρχιτεκτονικής ΣΝΔ

Η πρώτη απόπειρα της σχεδίασης ενός ΣΝΔ δικής μου αρχιτεκτονικής φαίνεται παρακάτω. Η επιλογή αυτή είναι βασισμένη σε εκείνες που διδάχτηκαν στα εργαστήρια του μαθήματος. Η επιλογή του αρχικού πλήθους κόμβων έγινε τυχαία.

```
model = tf.keras.Sequential([
    # nodes = 32
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu',in-
put_shape=(image_dimension, image_dimension, 3)),
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(rate=do),
    tf.keras.layers.Dense(out_nodes,activation='softmax')
])
```

Στην συνέχεια, δοκίμασα να βάλω και μία ακόμα ομάδα των πρώτων επιπέδων αλλά με διπλάσιο πλήθος κόμβων. Η αύξηση στην απόδοση του δικτύου τόσο στο validation όσο και στο testing ήταν αξιοσημείωτη.

```
model = tf.keras.Sequential([
    # nodes = 32
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu',in-
put_shape=(image_dimension, image_dimension, 3)),
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    # nodes = 64
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    tf.keras.layers.Flatten(),
```

```
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(rate=do),
    tf.keras.layers.Dense(out_nodes,activation='softmax')
])
```

Έπειτα, πρόσθεσα ένα επιπλέον συνελικτικό επίπεδο σε κάθε ομάδα και παρατήρησα αύξηση στην απόδοση του δικτύου.

```
model = tf.keras.Sequential([
    # nodes = 32
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu',in-
put_shape=(image_dimension, image_dimension, 3)),
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    # nodes = 64
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(rate=do),
    tf.keras.layers.Dense(out_nodes,activation='softmax')
])
```

Η προσθήκη μίας επιπλέον ομάδας διπλάσιων κόμβων από της προηγούμενης σημείωσε εκ νέου αύξηση στην απόδοση του δικτύου.

```
model = tf.keras.Sequential([
    # nodes = 32
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu',in-
put_shape=(image_dimension, image_dimension, 3)),
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    # nodes = 64
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),

    # nodes = 128
    tf.keras.layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(3,3)),
```

```
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(rate=do),
    tf.keras.layers.Dense(out_nodes,activation='softmax')
])
```

Αντίθετα, η προσθήκη επιπλέον ομάδας συνελικτικών επιπέδων ίδιων ή διπλάσιων κόμβων δεν οδήγησε σε αύξηση της απόδοσης. Τέλος, η μεταβολή της πιθανότητας απόρριψης δεν οδήγησε σε αύξηση της απόδοσης οπότε διατηρήθηκε στο 0,5.

Τελικώς, η αρχιτεκτονική του μοντέλου μου αποτελείται από 18 επίπεδα. Συγκεκριμένα, διαθέτει:

- 8 συνελικτικά επίπεδα
- 5 εξαγωγής τοπικού μεγίστου
- 1 flatten
- 2 πλήρως συνδεδεμένα
- 1 batch normalization
- 1 τυχαίας απόρριψης

Επιπρόσθετα στοιχεία εκπαίδευσης

Ένα επιπλέον χαρακτηριστικό που χρησιμοποιούμε για την εκπαίδευση του δικτύου μας είναι η συνάρτηση callbacks. Αυτή καλείται επανειλημμένα κατά την διάρκεια της εκπαίδευσης ώστε να ανανεώσει ή όχι τις τιμές των βαρών. Η ανανέωση εξαρτάται από την μείωση του σφάλματος στο στάδιο της επικύρωσης (validation). Ακόμη, ορίζουμε ότι εάν το σφάλμα της επικύρωσης(val_loss) δεν μεταβληθεί κατά min_delta για ορισμένες διαδοχικές εποχές (patience), να σταματήσει η εκπαίδευση του δικτύου. Η μέθοδος αυτή ονομάζεται early stopping.

```
callbacks = []

save_best_callback = tf.keras.callbacks.ModelCheck-
point(f'hw4.hdf5', save_best_only=True, verbose=1)
callbacks.append(save_best_callback)

early_stop_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                       min_delta=0,
                                                       patience=5,
                                                       verbose=1,
                                                       mode='auto',
                                                       restore_best_weights=True)
callbacks.append(early_stop_callback)
```

Για την εκπαίδευση του δικτύου καλούμαι την συνάρτηση:

```
history = model.fit_generator(
      train_generator,
```

```
      steps_per_epoch=train_generator.samples/train_generator.batch_size ,
      epochs=epochs,
      validation_data=validation_generator,
      verbose=1,
      shuffle=True,
      callbacks=callbacks)
```

Έλεγχος και αποτελέσματα

Εφόσον έχω εκπαιδεύσει το δίκτυο και τα αποτελέσματα στο στάδιο της πιστοποίησης είναι επιθυμητά, προχωρώ στο στάδιο του τελικού ελέγχου και τροφοδοτώ τα αρχεία που βρίσκονται στον προορισμό test_dir.

```
# --------------------
# Flow validation images in batches of 20 using datagen generator
# --------------------
test_data_gen = ImageDataGenerator( rescale = 1./255,
                                    brightness_range = [0.5 , 1],
                                    zoom_range = [0.15 , 1],
                                    horizontal_flip=False,
                                    vertical_flip=False
                                   )

test_generator = test_data_gen.flow_from_directory(test_dir,
                                                   batch_size=100,
                                                   class_mode='categorical',
                                                   # color_mode='grayscale',
                                                   shuffle=False,
                                                   target_size=(image_dimension, im-
age_dimension))

test_loss, test_acc = model.evaluate(test_generator)
```

Τελικώς, η ακρίβεια της εκπαίδευσης είναι 95,2%, της επικύρωσης 93,66% και της αξιολόγησης 93,62% Παρατηρώ ότι εφόσον τα αποτελέσματα για το validation και το testing είναι παραπλήσια, το μοντέλο δεν έχει πέσει σε σφάλμα overfitting. Απαιτήθηκαν 28 εποχές για την εκπαίδευση του δικτύου.

Προεκπαιδευμένο δίκτυο

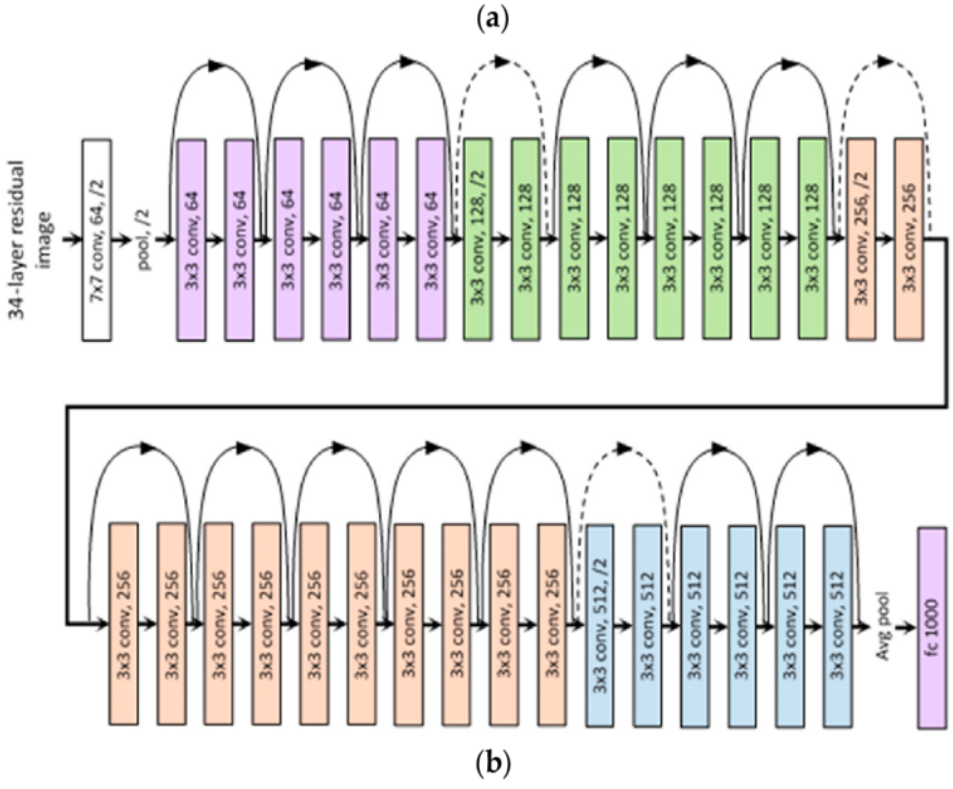
Με τον όρο «προεκπαιδευμένο δίκτυο» αναφερόμαστε σε ένα δίκτυο το οποίο έχει σχεδιαστεί από κάποιον άλλο και είναι εκπαιδευμένο σε ένα μεγάλο σύνολο δεδομένων για την επίλυση παρόμοιων προβλημάτων. Τα δίκτυα αυτά χρησιμοποιούνται ως σημείο εκκίνησης έπειτα του οποίου προσθέτουμε επιπλέον επίπεδα.

Μερικά από τα πιο γνωστά προεκπαιδευμένα δίκτυα για την ταξινόμηση εικόνων είναι:

- VGG-16,
- ResNet50,
- Inceptionv3,
- EfficientNet,

εκ των οποίων στο εργαστήριο διδαχτήκαμε τα δύο πρώτα. Εφόσον δεν είναι δυνατή η χρήση του VGG και δεδομένου ότι το ResNet χρησιμοποιείται εντονότερα στις μέρες μας, αποφάσισα να αξιοποιήσω αυτό το προεκπαιδευμενο δίκτυο και συγκεκριμένα την έκδοση ResNet50.

Η αρχιτεκτονική ResNet (Residual Network) είναι μια βαθιά νευρωνική δικτύωση σε σχήμα στοίβας που επιλύει το πρόβλημα της εξαφάνισης των gradient στις πολύ βαθιές δικτυώσεις. Αποτελείται από μονάδες που ονομάζονται "residual blocks" οι οποίες περιέχουν πολλαπλά επαναλαμβανόμενα επίπεδα (layers) και επαναλαμβανόμενες εντολές normalization. Το αποτέλεσμα αυτών προστίθεται στην είσοδο, επιτρέποντας την πληροφορία να ρέει εύκολα μέσα από το δίκτυο και μειώνοντας τον κίνδυνο της υπερεκπαίδευσης (overfitting). Η αρχιτεκτονική αυτή είναι υψηλά πολυμορφική και μπορεί να περιέχει πολλά επίπεδα. Οι εκδόσεις 50, 101 και 152 επιπέδων είναι οι πιο σύνηθες.



Εικόνα 4: Αρχιτεκτονική ResNet

Αρχικά, εισάγω το προεκπαιδευμένο δίκτυο. Με την παράμετρο include_top=False, δηλώνεται η απουσία ενός πλήρως συνδεδεμένου επιπέδου στο τέλος (fully connected – dence layer).

Επίσης, χρησιμοποιούνται τα βάρη που προέκυψαν ύστερα από την εκπαίδευση του δικτύου στο σύνολο εικόνων ImageNet. Αυτό αποτελείται από εικόνες αντικειμένων όπως μπαλόνια, φράουλες. Περιέχει συνολικά πάνω από 20.000 κατηγορίες και περισσότερες από 14 εκατομμύρια φωτογραφίες. Συνήθως αξιοποιείται για την αναγνώριση αντικειμένων. Αναφέρω ότι για εκπαιδευτικούς σκοπούς δοκιμάστηκε και η επιλογή None για την τιμή της μεταβλητής weights, η οποία έδινε χειρότερα αποτελέσματα, οπότε και απορρίφθηκε.

```
pre_trained = tf.keras.applications.ResNet50(include_top=False,
                                             weights='imagenet',
                                             input_shape=(224, 224, 3),
                                             pooling='None')
```

Στην συνέχεια, δημιουργώ το μοντέλο μου τοποθετώντας ως πρώτο επίπεδο το προεκπαιδευμένο δίκτυο. Στην συνέχεια, προσθέτω και άλλα επίπεδα. Ενδεικτικά, η πρώτη αρχιτεκτονική είναι:

```
# Connecting the pre-trained model with my model
model = models.Sequential()
model.add(pre_trained)

    # nodes = 32
model.add(layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'))
model.add(layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'))
model.add(layers.MaxPool2D(pool_size=(3,3)))


model.add(layers.Flatten())
model.add(layers.Dense(512,activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(rate=do))

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

Η απόδοση του δικτύου είναι περίπου 96% στο testing. Επιχειρώντας να αυξήσω την απόδοση επιχείρησα να προσθέσω και άλλα επίπεδα, αλλά μου εμφάνισε σφάλμα «Negative dimension size caused by subtracting 3 from 1 for '{{node max_pooling2d_9/MaxPool}}». Ύστερα από αρκετή αναζήτηση στο διαδίκτυο είδα ότι λύνεται με την προσθήκη της παραμέτρου padding=’same’ στην συνάρτηση Maxpooling. Έτσι, κατάφερα να προσθέσω ένα επιπλέον συνελικτικό επίπεδο. Επίσης, σκέφτηκα να διπλασιάζω το πλήθος των κόμβων κάθε επιπέδου. Η απόδοση αυξήθηκε σε πολύ μικρό βαθμό.

```
model = models.Sequential()
model.add(pre_trained)

    # nodes = 32
model.add(layers.Conv2D(filters=nodes, kernel_size=(ks,ks), activation='relu'))
model.add(layers.Conv2D(filters=nodes*2, kernel_size=(ks,ks), activation='relu'))
model.add(layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'))
model.add(layers.MaxPool2D(pool_size=(3,3), padding='same'))
```

```
model.add(layers.Flatten())
model.add(layers.Dense(512,activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(rate=do))

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

Έπειτα, σκέφτηκα να απενεργοποιήσω ορισμένα από τα τελευταία επίπεδα του προεκπαιδευμένου δικτύου. Αυτό υλοποιείται με το ακόλουθο κομμάτι κώδικα:

```
# Keeping active the first 100 layers
for layer in pre_trained.layers[100:]:
    layer.trainable = False

# Checking the trainable status of the individual layers
for layer in pre_trained.layers:
    print(layer, layer.trainable)
```

Διατηρώντας σταθερή την αρχιτεκτονική του δικτύου, η απόδοση του μειώθηκε δραματικά. Συνεπώς, μέθοδος αυτή απορρίφθηκε.

Επιπλέον, ο τετραπλασιασμός των κόμβων σε κάθε επίπεδο οδήγησε σε αύξηση της απόδοσης στο 96,8%.

```
model = models.Sequential()
model.add(pre_trained)

    # nodes = 32
model.add(layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'))
model.add(layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'))
model.add(layers.Conv2D(filters=nodes*4, kernel_size=(ks,ks), activation='relu'))
model.add(layers.MaxPool2D(pool_size=(3,3), padding='same'))

model.add(layers.Flatten())
model.add(layers.Dense(512,activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(rate=do))

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

Έπειτα, δοκίμασα να οκταπλασιάσω το πλήθος των κόμβων σε κάθε επίπεδο. Απόρροια αυτού ήταν η μείωση της απόδοσης στο 96,65%. Ακόμη, η προσθήκη επιπλέον επιπέδου μετά εκείνου της εξαγωγής τοπικού μεγίστου περιόρισε τις διαστάσεις της εξόδου οδηγώντας σε σφάλμα. Τέλος, η μεταβολή του ρυθμού απόρριψης (drop out rate) δεν οδήγησε σε καλύτερα αποτελέσματα, οπότε κρατήθηκε και εδώ 0,5.

Από τα παραπάνω επιλέγω ως τελική αρχιτεκτονική αυτή των 32*4=128 κόμβων σε κάθε συνελικτικό επίπεδο η οποία αποτελείται από 58 επίπεδα εκ των οποίων είναι:

- 50 επίπεδα του προεκπαιδευμένου δικτύου
- 4 συνελικτικά επίπεδα

- 1 εξαγωγής τοπικού μεγίστου
- 1 flatten
- 1 batch normalization
- 1 τυχαίας απόρριψης

Η αρχιτεκτονική αυτή δίνει απόδοση 96,8% στην φάση ελέγχου (testing).

Σχόλια

Αξίζει να αναφερθεί ότι το μοντέλο ξεκινάει κάθε φορά την εκπαίδευση με διαφορετικές τιμές για τα βάρη. Επιπλέον, η μικρή τιμή που έχει δοθεί στην παράμετρο patience τόσο στο πρώτο όσο και στο δεύτερο δίκτυο, οδηγεί στην εκπαίδευση για λιγότερο από 100 εποχές. Απόρροια αυτών είναι να εμφανίζεται μία μικρή απόκλιση στις τιμές κάθε φορά που τρέχω το εκάστοτε μοντέλο. Οι τιμές που παρουσιάστηκαν ήταν οι καλύτερες σε ένα συνεχόμενο σετ 5 διαδοχικών εκτελέσεων.

Επιπρόσθετα, παρατηρείται αύξηση της απόδοσης με την χρήση του προεκπαιδευμένου δικτύου. Μία σημαντική διαφορά ανάμεσα στα δύο μοντέλα είναι το βάθος τους. Όσο πιο βαθύ είναι το δίκτυό μου, τόσο περισσότερο μπορεί να πολυπλέξει τα δεδομένα του. Με άλλα λόγια, η αύξηση του βάθους οδηγεί σε αύξηση της συνέλιξης των χαρακτηριστικών (ακμές, σχήματα, χρώματα) που εξάγει το δίκτυο, με αποτέλεσμα αυτά να είναι πιο ακριβή. Ωστόσο, ενδέχεται κίνδυνος το μοντέλο μου να υπερεξειδικευτεί (overfitting). Για αυτό χρησιμοποιείται το επίπεδο τυχαίας απόρριψης, το επίπεδο κανονικοποίησης και η επαύξηση των δεδομένων εισόδου (data augmentation).

Εφαρμογή dataset 3ης εργασίας

Ζητείται να αξιοποιήσουμε τα προαναφερόμενα δίκτυα χρησιμοποιώντας ως δεδομένα εισόδου τα αρχεία για την 3η εργασία. Συνεπώς, στο ήδη υπάρχον πρόγραμμα, αλλάζω μονάχα την διεύθυνση όπου βρίσκονται τα συμπιεσμένα αρχεία και προσαρμόζω κατάλληλα το path για το train_dir και test_dir. Η απόδοση του κάθε δικτύου φαίνεται στους παρακάτω πίνακες.

	Απόδοση
Εκπαίδευση (training)	87,37%
Επικύρωση (validation)	48,84
Αξιολόγηση (testing)	80,77%

Πίνακας 1: Μη προεκπαιδευμένο δίκτυο

	Απόδοση
Εκπαίδευση (training)	95,62%
Επικύρωση (validation)	21,28%
Αξιολόγηση (testing)	21,15%

Πίνακας 2: Προεκπαιδευμένο δίκτυο

Η απόκλιση της απόδοσης στο στάδιο της εκπαίδευσης με εκείνη της επικύρωσης είναι αξιοσημείωτα μεγάλη. Όπως προαναφέρθηκε, η διαφορά αυτή υποδεικνύει ότι το δίκτυο δεν

είναι ικανό να γενικεύσει τα αποτελέσματα του. Συμπεραίνουμε, λοιπόν, ότι το δίκτυο έχει υπερεξειδικευτεί στα δεδομένα της εκπαίδευσης (overfitting). Το αποτέλεσμα αυτό είναι σχετικά αναμενόμενο δεδομένου ότι τώρα έχω σημαντικά μικρότερο dataset.