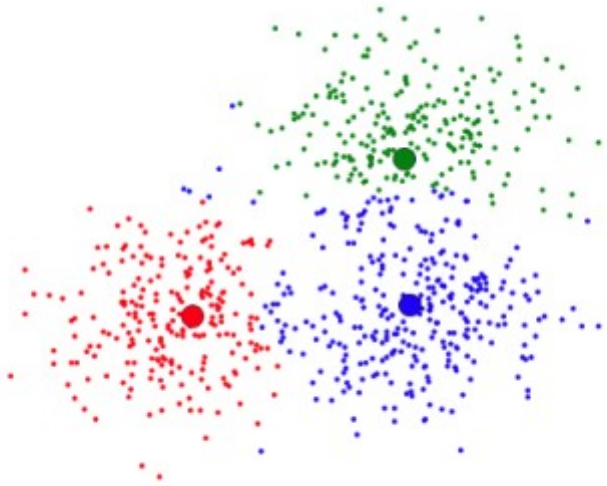
Αναγνώριση Προτύπων

Εργασία εργαστηρίου



Μασλάρης Ιωάννης-Αριστείδης
57348

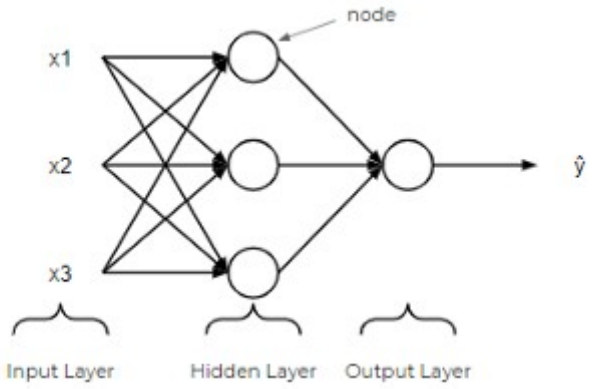
Φεβρουάριος 2021

Πίνακας περιέχωμένων

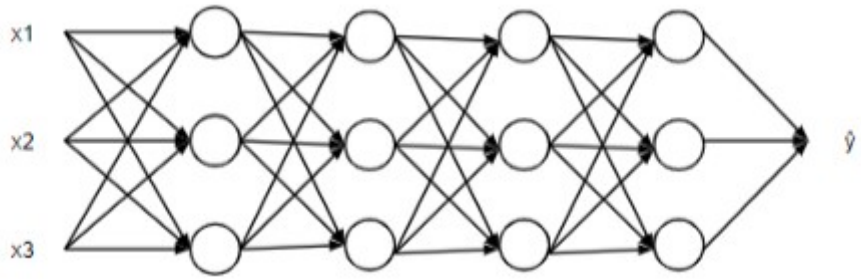
Εισαγωγή

Τα τεχνητά νευρωνικά δίκτυα είναι υπολογιστικά συστήματα δομής διασυνδεμένων κόμβων, των νευρώνων, η λειτουργία των οποίων είναι παρεμφερής με αυτήν των νευρώνων του ανθρώπινου εγκεφάλου. Με τη χρήση αλγορίθμων μπορούν να αποκτήσουν ιδιότητες όπως αναγνώριση προτύπων σε δεδομένα και ταξινόμηση εικόνων. Το πρώτο νευρωνικό δίκτυο πρώτηναν οι Warren McCulloch και Walter Pitts το 1943 οι οποίοι περιέγραψαν πως θα μπορούσαν να λειτουργούν οι νευρώνες μοντελοποιώντας τις ιδέες τους φτιάχνοντας ένα ηλεκτρικό κύκλωμα. Σήμερα με την εξέλιξη των υπολογιστικών συστημάτων και το πλήθος δεδομένων που υπάρχει τα τεχνητά νευρωνικά δίκτυα όχι μόνο χρησιμοποιούνται ευρέως αλλά συνεχώς εξελίσσονται προσφέροντας ακόμα περισσότερες ευκαιρίες.

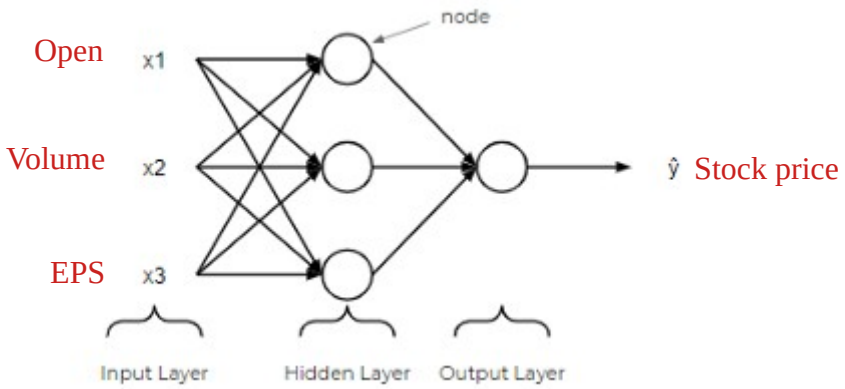
Οι βασικοί μηχανισμοί λειτουργίας των τεχνητών νευρωνικών δικτύων είναι πολύ απλοί όπως θα δούμε και παρακάτω. Σε κάθε νευρωνικό δίκτυο υπάρχει ένα επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα και ένα επίπεδο εξόδου.



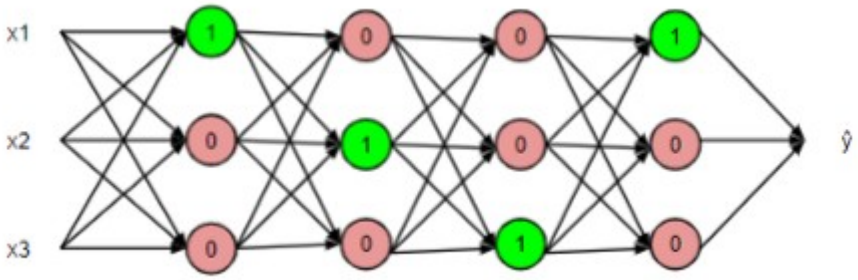
Το επίπεδο εισόδου αποτελείται από τις ανεξάρτητες μεταβλητές του προβλήματος τις οποίες συμβολίζουμε με $X_1, X_2, \ldots, X_n$. Αυτά είναι τα χαρακτηριστικά των δεδομένων στα οποία θα εκπαιδευτεί το δίκτυο. Το κρυφό επίπεδο αποτελείται από ένα ή περισσότερους κόμβους, τους νευρώνες. Αυτοί επεξεργάζονται την πληροφορία που δέχονται και αναλόγως πυροδοτούνται ή παραμένουν στάσιμοι. Τέλος το επίπεδο εξόδου αποτελείται και αυτό από ένα πλήθος από νευρώνες το οποίο εξαρτάται από το εκάστοτε πρόβλημα.



Στο σχήμα που ακολουθεί βλέπουμε ένα απλό παράδειγμα ενός πιθανού νευρωνικού δικτύου. Το δίκτυο προσπαθεί να κάνει πρόβλεψη της τιμής μία μετοχής βασιζόμενο σε τρεις παραμέτρους, opening price (open), volume και EPS. Δίνοντας αυτές τις πληροφορίες στο δίκτυο θα μας επιστρέφει μία έξοδο, την τιμή της μετοχής την επόμενη μέρα ή σε ένα βάθος χρόνου που ορίζουμε.



Κάθε νευρώνας αποτελείται από δύο εξισώσεις, μία γραμμική συνάρτηση και μία συνάρτηση ενεργοποίησης. Οι είσοδοι κάθε νευρώνα οδηγούνται στην γραμμική συνάρτηση η οποία μας επιστρέφει μία τιμή, έστω Φ . Στην συνέχεια η τιμή Φ οδηγείται στην συνάρτηση ενεργοποίησης η οποία εν συνεχεία πυροδοτεί ή όχι ένα σήμα εξόδου στην έξοδο του νευρώνα προς το επόμενο επίπεδο. Λειτουργεί δηλαδή σαν διακόπτης ο οποίος ενεργοποιείται ή όχι ανάλογα με την τιμή που παίρνει από την γραμμική συνάρτηση. Έτσι κάθε νευρώνας καθορίζει ποίοι νευρώνες του επόμενου επιπέδου ενεργοποιούνται και αυτό είναι και η ουσία των τεχνητών νευρωνικών δικτύων.



Τα νευρωνικά δίκτυα έχουν εξελιχθεί σε τέτοιο βαθμό ώστε πλέον υπάρχουν αρκετοί διαφορετικοί τύποι αυτών. Στην εργασία θα μελετήσουμε δύο τύπους τεχνητών νευρωνικών δικτύων, την αρχιτεκτονική ANN(Artificial Neural Network) και την αρχιτεκτονική CNN(Convolution Neural Network). Τα δίκτυα ANN αποτελούνται από ακολουθίες επιπέδων διασυνδεομένων νευρώνων, όπως ακριβώς είδαμε και στις προηγούμενες εικόνες και αποτελούν τον πιο βασικό τύπου νευρωνικού δικτύου. Τα

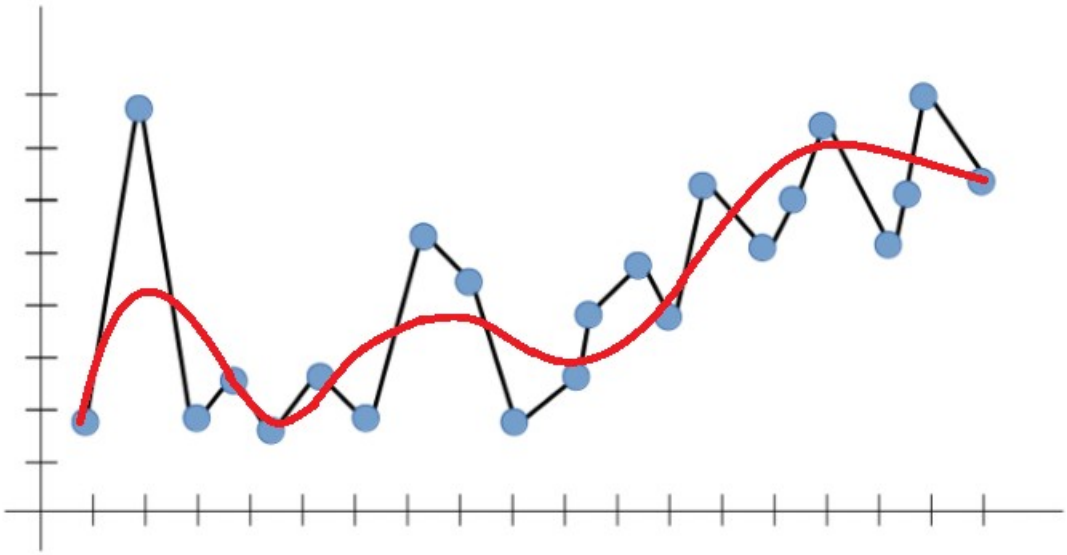
δίκτυα CNN χρησιμοποιούν την πράξη της συνέλιξης στις λειτουργίες τους. Ακολουθούν την δομή επιπέδων νευρώνων ενώ εισάγουν και κάποια νέα επίπεδα τα οποία θα δούμε λεπτομερώς παρακάτω.

Τα νευρωνικά δίκτυα αποτελούν μία πολύ ισχυρή υπολογιστική πρακτική έχοντας οδηγήσει στην υλοποίηση μερικών από τις πιο σύγχρονες και καινοτόμες τεχνολογίες όπως, αναγνώριση προτύπων σε εικόνα και βίντεο, συστήματα προτάσεων (recommendation systems) και αυτόματη οδήγηση.

Overfitting και Underfitting

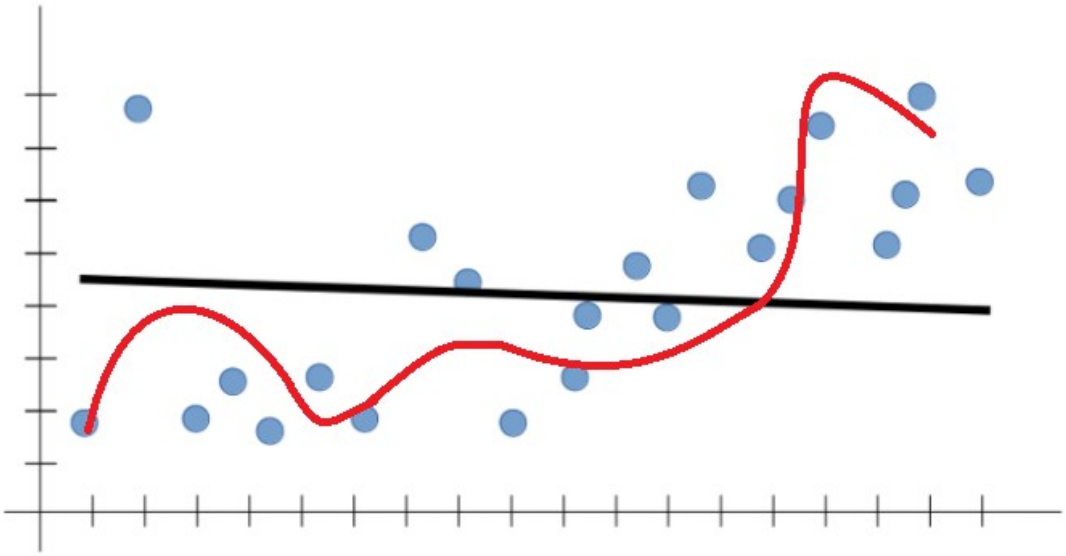
Αποτελούν δύο φαινόμενα τα οποία χαρακτηρίζουν το δίκτυο και μας βοηθούν να καταλάβουμε τόσο την απόδοση που θα έχει όσο και το πόσο καλά έχει εκπαιδευτεί.

- Overfitting: Είναι η περίπτωση κατά την οποία το δίκτυο δεν μπορεί να γενικεύσει, δηλαδή δεν μπορεί να αποδώσει ικανοποιητικά σε νέα δεδομένα. Καθώς εκπαιδεύουμε ένα δίκτυο, με κάθε νέα εποχή το σφάλμα του μειώνεται αφού δέχεται όλο και περισσότερα δεδομένα ή και τα ίδια επανειλημμένα. Εκτελώντας την διαδικασία για πολύ μεγάλο πλήθος εποχών καταλήγουμε σε μία ελάχιστη τιμή για το σφάλμα. Ωστόσο αυτό σημαίνει ότι το δίκτυο 'μαθαίνει' με λεπτομέρεια σχεδόν όλα τα χαρακτηριστικά του training set. Έτσι όμως δεν μπορεί να αναγνωρίσει δευτερεύοντα χαρακτηριστικά που εμφανίζονται σε δεδομένα στα οποία δεν έχει εκπαιδευτεί. Αυτά όμως τα δευτερεύοντα χαρακτηριστικά είναι τα πλέον χρήσιμα ώστε το δίκτυο να αποδίδει ικανοποιητικά σε ένα μεγάλο πλήθος νέων δεδομένων. Επιθυμούμε δηλαδή το δίκτυο να γενικεύει, όσο είναι δυνατόν, αυτά που "έμαθε" κατά την φάση της εκπαίδευσης. Στην φωτογραφία που ακολουθεί βλέπουμε ένα παράδειγμα overfitting.



Με μπλε φαίνονται τα σημεία του training set, με μαύρη γραμμή η συνάρτηση που μαθαίνει ένα overfitted δίκτυο και με κόκκινη γραμμή η επιθυμητή συνάρτηση. Όπως βλέπουμε και από την εικόνα το φαινόμενο οφείλεται στο γεγονός ότι το δίκτυο μοντελοποιεί όλα (ή σχεδόν όλα) τα χαρακτηριστικά του training set. Έτσι ένα σύνολο δεδομένων το οποίο μοιάζει με το training set ίσως να μην μπορεί να το κατηγοριοποιήσει σωστά. Η έλλειψη γενικότητας λοιπόν οφείλεται σε δύο κύριους παράγοντες, στην πολυπλοκότητα του δικτύου και στην απλότητα του training set. Οπότε δύο προφανείς λύσεις στο πρόβλημα θα ήταν πρώτον να απλοποιήσουμε το δίκτυο, αφαιρώντας ένα πλήθος από νευρώνες και δεύτερον να αυξήσουμε το πλήθος των δεδομένων στο training set με ποιο γενικές περιπτώσεις δεδομένων.

- Underfitting: Είναι η περίπτωση κατά την οποία το δίκτυο δεν μπορεί ούτε να μοντελοποιήσει το training set ούτε να γενικεύσει σε νέα δεδομένα. Όπως είπαμε και προηγουμένως θέλουμε το δίκτυο να "μάθει" από το training set αλλά με ένα μέτρο. Μία λύση θα ήταν σταματήσουμε την διαδικασία την εκπαίδευσης νωρίτερα. Ωστόσο αυτό θα μπορούσε να οδηγήσει το μοντέλο να μην προλάβει να "μάθει" αρκετά χαρακτηριστικά, ακόμη και τα πιο κύρια, από το training set. Επομένως χαρακτηρίζουμε ένα δίκτυο underfitted όταν αυτό δεν αποδίδει ικανοποιητικά ακόμη και στο training set. Στην φωτογραφία που ακολουθεί βλέπουμε ένα παράδειγμα underfitting.



Με μπλε φαίνονται τα σημεία του training set, με μαύρη γραμμή η συνάρτηση που μαθαίνει ένα underfitted δίκτυο και με κόκκινη γραμμή η επιθυμητή συνάρτηση. Το φαινόμενο του underfitting οφείλεται στην έλλειψη ικανότητας από το δίκτυο να μοντελοποιήσει τα κύρια χαρακτηριστικά του training set και η αρχιτεκτονική του συμβάλει σε αυτό. Ένα απλό δίκτυο δεν είναι ικανό να μάθει πολύπλοκα χαρακτηριστικά του training set, όσο μεγάλο και αν είναι αυτό. Απλό δίκτυο θα μπορούσε να είναι ένα ANN με λίγα επίπεδα νευρώνων, γενικά με μικρό πλήθος νευρώνων. Όμως ένα πιο σύνθετο ANN μπορεί να θεωρηθεί πιο απλό μπροστά σε ένα απλό CNN δίκτυο, στην περίπτωση όπου έχουμε να κάνουμε με εικόνες. Αυτό οφείλεται στις λειτουργίες της CNN αρχιτεκτονικής, τις οποίες θα μελετήσουμε παρακάτω. Άρα η έννοια του απλού δικτύου είναι πάντα σχετική με το πρόβλημα που λύνουμε.

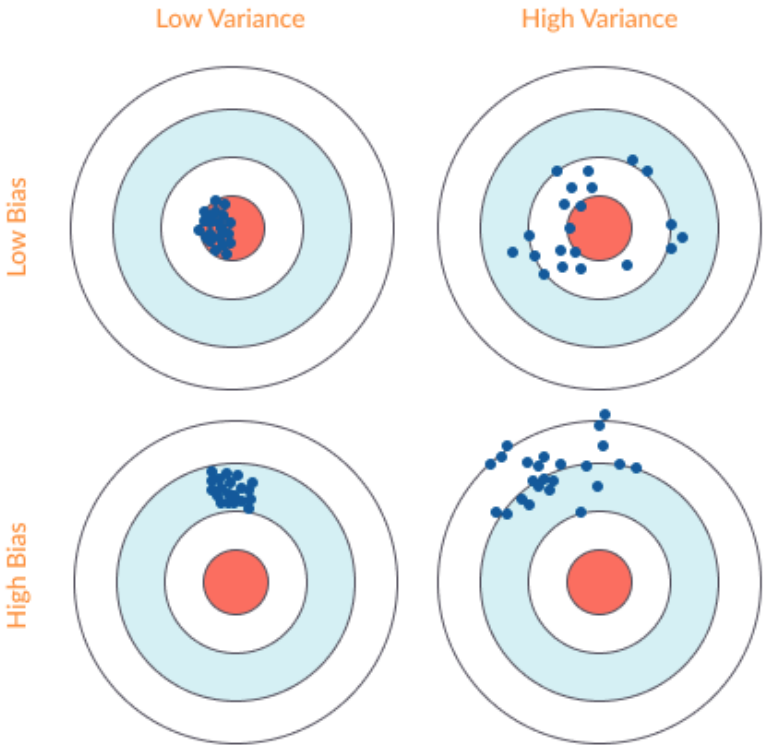
Και στις δύο περιπτώσεις το δίκτυο θεωρείται ανεπαρκές για να λύσει το πρόβλημα μας. Το πως μπορούμε να διακρίνουμε τα φαινόμενα σε δίκτυα αλλά και πως μπορούμε να τα λύσουμε θα το δούμε λεπτομερώς παρακάτω.

Bias και Variance

Το bias και το variance αποτελούν δύο ευρέως χρησιμοποιούμενες μετρικές για την αξιολόγηση τεχνητών νευρωνικό δικτύων. Συνήθως δεν χρησιμοποιούνται αυτούσιες αλλά μέσω του σφάλματος (ή του loss) που χαρακτηρίζει το εκάστοτε δίκτυο. Είναι ιδιαίτερα χρήσιμες για να καταλάβουμε εάν εμφανίζεται το φαινόμενα Underfitting ή Overfitting κατά την εκπαίδευση του δικτύου.

- Bias: Αποτελεί μέτρο του κατά πόσο οι προβλέψεις του δικτύου διαφέρουν από τις πραγματικές τιμές. Η σύνδεση του bias με σφάλμα και κατ' επέκταση με το loss γίνεται ως εξής. Όσο μεγαλύτερη η τιμή του bias τόσο μεγαλύτερη θα είναι και η τιμή του loss στο training set.
- Variance: Αποτελεί μέτρο του κατά πόσο διαφέρουν μεταξύ τους επαναλαμβανόμενες προβλέψεις. Η σύνδεση του variance με το σφάλμα και κατ' επέκταση με το loss γίνεται ως εξής. Όσο μεγαλύτερη η τιμή του variance, τόσο μεγαλύτερη θα είναι και η τιμή του loss στο validation set.

Παρακάτω βλέπουμε γραφικά την έννοια των bias και variance. Θεωρώντας το κέντρο του στόχου την επιθυμητή πρόβλεψη βλέπουμε πως συνδυασμοί υψηλού και χαμηλού bias και variance επηρεάζουν την ακρίβεια της πρόβλεψης

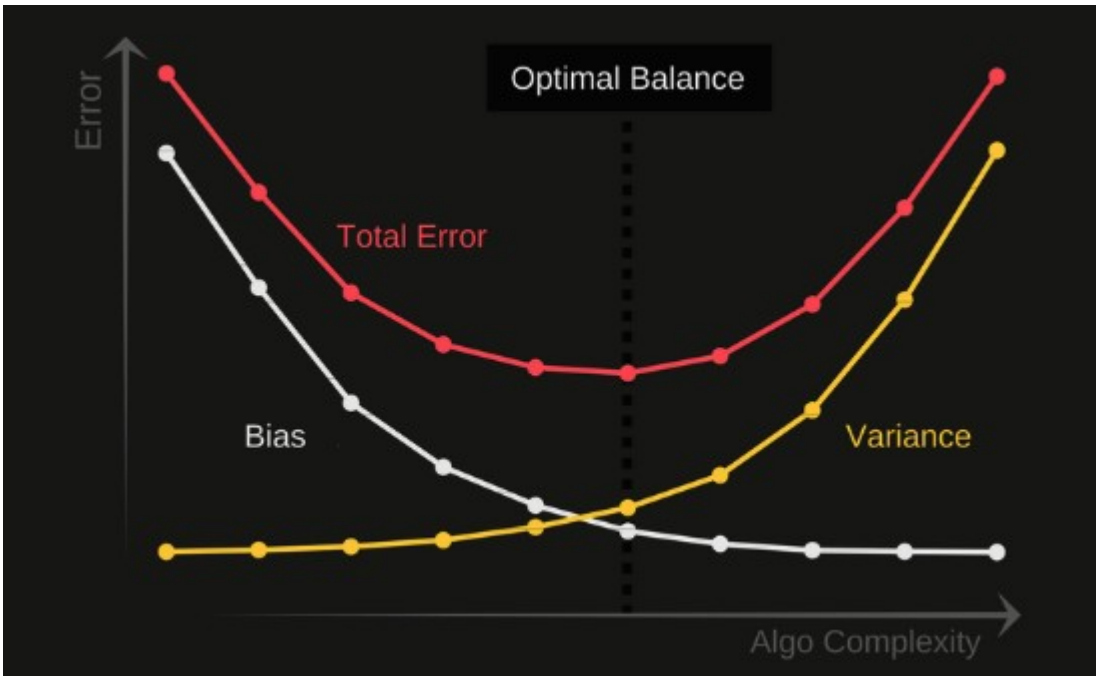


Στο πάνω δεξί μέρος της εικόνας βλέπουμε την απόδοση ενός overfitted δικτύου. Ένα τέτοιο δίκτυο έχει μικρή τιμή bias και μεγάλη τιμή variance. Αντίστοιχα στο κάτω αριστερό μέρος της εικόνας βλέπουμε την απόδοση ενός underfitted δικτύου. Σε αυτή την περίπτωση το

δίκτυο χαρακτηρίζεται από υψηλή τιμή bias και χαμηλή τιμή variance. Οι μετρικές bias και variance αποδεικνύονται πολύ χρήσιμες για την αναγνώριση των φαινομένων overfitting και underfitting σε δίκτυα. Το bias ταυτίζεται με το trainings loss ενώ το variance ταυτίζεται με την διαφορά μεταξύ training loss και validation loss. Το overfitting είναι ένα πρόβλημα υψηλού bias, άρα για να το λύσουμε στοχεύουμε στην μείωσή του. Αντίστοιχα το underfitting είναι ένα πρόβλημα υψηλού variance, άρα το λύνουμε μειώνοντάς το.

Bias – Variance trade off

Όπως είδαμε και στις προηγούμενες 2 παραγράφους, οι τεχνικές που χρησιμοποιούμε για να λύσουμε το underfitting μας "φέρνουν " πιο κοντά στο overfitting και αντίστροφα. Οπότε σε καμία περίπτωση η λύση ενός από τα δύο φαινόμενα δεν απαιτεί ακραία εφαρμογή των εκάστοτε μεθόδων. Πάντα για την βελτιστοποίηση του δικτύου θα πρέπει να βρίσκουμε μία ισορροπία μεταξύ μείωσης των bias και variance ώστε το δίκτυο να "μαθαίνει" τόσα όσα είναι αρκετά για την γενικευμένη ικανοποιητική του απόδοση.



Απλό τεχνητό νευρωνικό δίκτυο (ANN)

Λειτουργία

Η λειτουργία του απλού νευρωνικού δικτύου μπορεί να χωριστεί σε δύο επιμέρους φάσης, την φάση Forward propagation και την φάση Back propagation. Ας ξεκινήσουμε αναλύοντας την πρώτη φάση.

Κατά την φάση του Forward propagation το δίκτυο τροφοδοτείται με τις τιμές των ανεξάρτητων μεταβλητών ενός δείγματος εκπαίδευσης. Αυτό έχει σαν αποτέλεσμα την προώθηση των "σημάτων" από την είσοδο του δικτύου μέχρι και την έξοδό του, περνώντας από τους νευρώνες των κρυφών επιπέδων οι οποίοι συνδέονται με τα τρέχοντα βάρη. Οι ανεξάρτητες μεταβλητές πολλαπλασιάζονται με τα βάρη των ενώσεων των νευρώνων του επόμενου επιπέδου. Στο επόμενο επίπεδο υπολογίζεται η τιμή της γραμμικής συνάρτησης καθώς και η τιμή της συνάρτησης ενεργοποίησης. Ανάλογα με την τελευταία ο κάθε νευρώνας προωθεί τις τιμές 1 ή 0 στους νευρώνες του επόμενου επιπέδου. Η έξοδος του δικτύου συγκρίνεται με την πραγματική τιμή της εξαρτημένης μεταβλητής του δείγματος εκπαίδευσης και υπολογίζεται η παράγωγος της συνάρτησης κόστους ως προς την τιμή εξόδου του δικτύου. Στη συνέχεια αυτή η παράγωγος πρέπει να υπολογιστεί ως προς τα βάρη σε όλα τα επίπεδα κατά την φάση backward propagation.

Κατά την φάση Back propagation υπολογίζεται η παράγωγος της συνάρτησης κόστους ως προς τα βάρη χρησιμοποιώντας τον κανόνα της αλυσίδας. Απώτερος σκοπός της φάσης αυτής είναι η εύρεση των βέλτιστων τιμών παραμέτρων του δικτύου. Επαναληπτικά με κάθε εποχή γίνεται ανανέωση των παραμέτρων του δικτύου. Μία συνάρτηση βελτιστοποίησης εφαρμόζεται κατά την φάση αυτή η οποία υλοποιεί τον σκοπό που αναφέρθηκε προηγουμένως. Μερικές από αυτές τις συναρτήσεις είναι οι Gradient descent, Adam optimizer και RMS prop. Ο κανόνας της αλυσίδας όπως είπαμε παίζει πολύ σημαντικό ρόλο κατά την εκτέλεση της φάσης back propagation.

α) Κατασκευή δικτύου

Το δίκτυο που χρησιμοποιούμε για το πρώτο ερώτημα αποτελείται από δύο κρυφά επίπεδα 256 και 128 νευρώνων αντίστοιχα καθώς και τα επίπεδα εισόδου και εξόδου. Ας ξεκινήσουμε με τα επίπεδα εισόδου και εξόδου τα οποία επηρεάζονται άμεσα από το πρόβλημα που λύνουμε και τα δεδομένα που χρησιμοποιούμε. Τα δεδομένα μας είναι έγχρωμες φωτογραφίες μεγέθους 100 X 100 εικονοστοιχεία οι οποίες αντιπροσωπεύουν 10 ξεχωριστές κλάσεις. Όπως αναφέραμε και προηγουμένως τη είσοδος του απλού τεχνητού νευρωνικού δικτύου (ANN) είναι μονοδιάστατη, δηλαδή αποτελείται από ένα διάνυσμα νευρώνων. Οι έγχρωμες φωτογραφίες όμως είναι δομή τριών διαστάσεων, δύο χωρικές διαστάσεις που περιέχουν πληροφορία για τις δομές που απεικονίζει η εικόνα καθώς και μία τρίτη που περιέχει πληροφορία για τα χρωματικά της κανάλια. Αρχικά πρέπει να σκεφτούμε ότι για να τροφοδοτήσουμε το δίκτυο με μία φωτογραφία θα πρέπει αυτή να μετατραπεί σε μία δομή πίνακα αριθμών. Άρα κάθε φωτογραφία στην ουσία για εμάς είναι ένας πίνακας τριών διαστάσεων μεγέθους 100 X 100 X 3. Άρα εφόσον έχουμε κάθε φωτογραφία σε μορφή πίνακα δεν είναι δύσκολο να τη μετατρέψουμε σε μορφή διανύσματος. Δηλαδή όλη η πληροφορία του τρισδιάστατου πίνακα βρίσκεται πλέον σε ένα διάνυσμα μεγέθους $100 \times 100 \times 3 = 30000$ θέσεις. Οπότε τελικά θα πρέπει να ορίσουμε το επίπεδου εισόδου του δικτύου μας να έχει μέγεθος 30000 θέσεις. Αντίστοιχα υπολογίζουμε και το μέγεθος του επιπέδου εξόδου του δικτύου ανατρέχοντας στα δεδομένα μας και το πρόβλημα που λύνουμε. Το δίκτυο θέλουμε στην έξοδο να μας υποδεικνύει την κλάση της φωτογραφίας που του δίνουμε σαν είσοδο. Εφόσον εμείς έχουμε 10 κλάσεις θέλουμε στην έξοδο του δικτύου να έχουμε 10 απολήξεις ώστε κάθε φορά να ενεργοποιείται μία, αυτή που αντιστοιχεί στην προβλεπόμενη κλάση. Τελικά, ορίζουμε το δίκτυο όπως φαίνεται και στην παρακάτω εικόνα.

```
model = models.Sequential()

model.add(tf.keras.Input(shape=(30000)))
model.add(layers.Dense(256))
model.add(layers.Dense(128))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

Ορίζουμε το επίπεδο εισόδου να έχει μέγεθος ίσο με 30000 νευρώνες. Στην συνέχεια ακολουθούν τα δύο κρυφά επίπεδα. Αυτά είναι τύπου Dense, δηλαδή κάθε νευρώνας συνδέεται με κάθε νευρώνα του προηγούμενου επιπέδου. Ορίζουμε το πλήθος νευρώνων 256 και 128 αντίστοιχα όπως ζητείται από την εκφώνηση. Η συνάρτηση ενεργοποίησης αφήνεται η default που ορίζεται από την βιβλιοθήκη Keras, αυτή είναι

η tanh. Τέλος ορίζουμε το επίπεδο εξόδου να είναι τύπου Dense με πλήθος νευρώνων 10 όπως εξηγήσαμε και προηγουμένως. Απαραίτητη ρύθμιση για το επίπεδο εξόδου είναι η συνάρτηση ενεργοποίησης του να είναι η Softmax. Μόνο με τη χρήση αυτής είναι δυνατόν στο τελευταίο επίπεδο να γίνει διάκριση της τιμής εξόδου μεταξύ των πιθανών κλάσεων. Στην συνέχεια παίρνουμε τις εξής πληροφορίες για το δίκτυο.

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_54 (Dense)             (None, 256)               7680256
_________________________________________________________________
dense_55 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_56 (Dense)             (None, 10)                1290
=================================================================
Total params: 7,714,442
Trainable params: 7,714,442
Non-trainable params: 0
_________________________________________________________________
```

Αρχικά ας εξηγήσουμε σε τι ανταποκρίνεται η λέξη Param ή Params. Με αυτόν τον όρο εννοείται το πλήθος των βαρών που εντοπίζονται σε κάθε επίπεδο του δικτύου. Βλέπουμε ότι στο πρώτο κρυφό επίπεδο μας δίνει 7680256 Params. Αυτό ερμηνεύεται ως εξής, κάθε νευρώνας του πρώτου κρυφού επιπέδου συνδέεται με όλους του προηγούμενο, δηλαδή με τους νευρώνες του επιπέδου εισόδου. Άρα συνολικά έχουμε $30000 \times 256 = 7680000$ συνδέσεις, άρα και βάρη. Επίσης κάθε νευρώνας του κρυφού επιπέδου συνδέεται με έναν νευρώνα bias, άρα συνολικά 256 συνδέσεις (άρα και βάρη) με νευρώνες bias. Οπότε συνολικά επιβεβαιώνουμε ότι έχουμε 7680256 βάρη. Με την ίδια λογική βλέπουμε ότι μεταξύ πρώτου και δεύτερου κρυφού επιπέδου έχουμε $256 \times 128 = 32768$ συνδέσεις ενώ έχουμε επιπλέον 128 συνδέσεις με νευρώνες bias. Οπότε για το δεύτερο κρυφό επίπεδο έχουμε συνολικά $32768 + 128 = 32896$ βάρη. Τέλος για το επίπεδο εξόδου με την ίδια λογική επιβεβαιώνονται 1290 βάρη. Φυσικά αν προσθέσουμε το πλήθος των βαρών σε κάθε επίπεδο προκύπτει το συνολικό τους πλήθος $Total\ params = 7.714.442$ Στην συνέχεια βλέπουμε το πλήθος των Learnable Parameters. Όπως αναφέραμε και στην εισαγωγή ένα δίκτυο στην ουσία “μαθαίνει”¨μέσω των βαρών που υπάρχουν στις συνδέσεις μεταξύ νευρώνων διαφορετικών επιπέδων. Άρα με τον όρο Learnable Parameters εννοούμε το πλήθος των βαρών σε ένα δίκτυο. Στην περίπτωση του δικού μας δικτύου βλέπουμε ότι όλα τα βάρη του ανανεώνονται κατά την εκπαίδευση. Τέλος βλέπουμε το πλήθος των Non-trainable parameters. Σε αντιστοιχία με τον προηγούμενο ορισμό

οι Non-trainable parameters είναι απλά τα βάρη τα οποία δεν ανανεώνονται κατά την διάρκεια της εκπαίδευσης. Υπάρχουν δύο κύρια είδη Non-trainable parameters:

- Τα βάρη τα οποία εμείς έχουμε επιλέξει να μένουν σταθερά και αυτά δεν θα ανανεωθούν κατά την εκπαίδευση.
- Τα βάρη τα οποία ανανεώνονται των BatchNormalization επιπέδων τα οποία ανανεώνονται σύμφωνα με τις μετρικές mean και variance και όχι σύμφωνα με τον αλγόριθμο backpropagation.

Εφόσον δεν έχουμε κανέναν από τους δύο τύπους Non-trainable parameters τότε είναι αναμενόμενο να παίρνουμε $Non-trainable\ params=0$.

b) Εκπαίδευση δικτύου

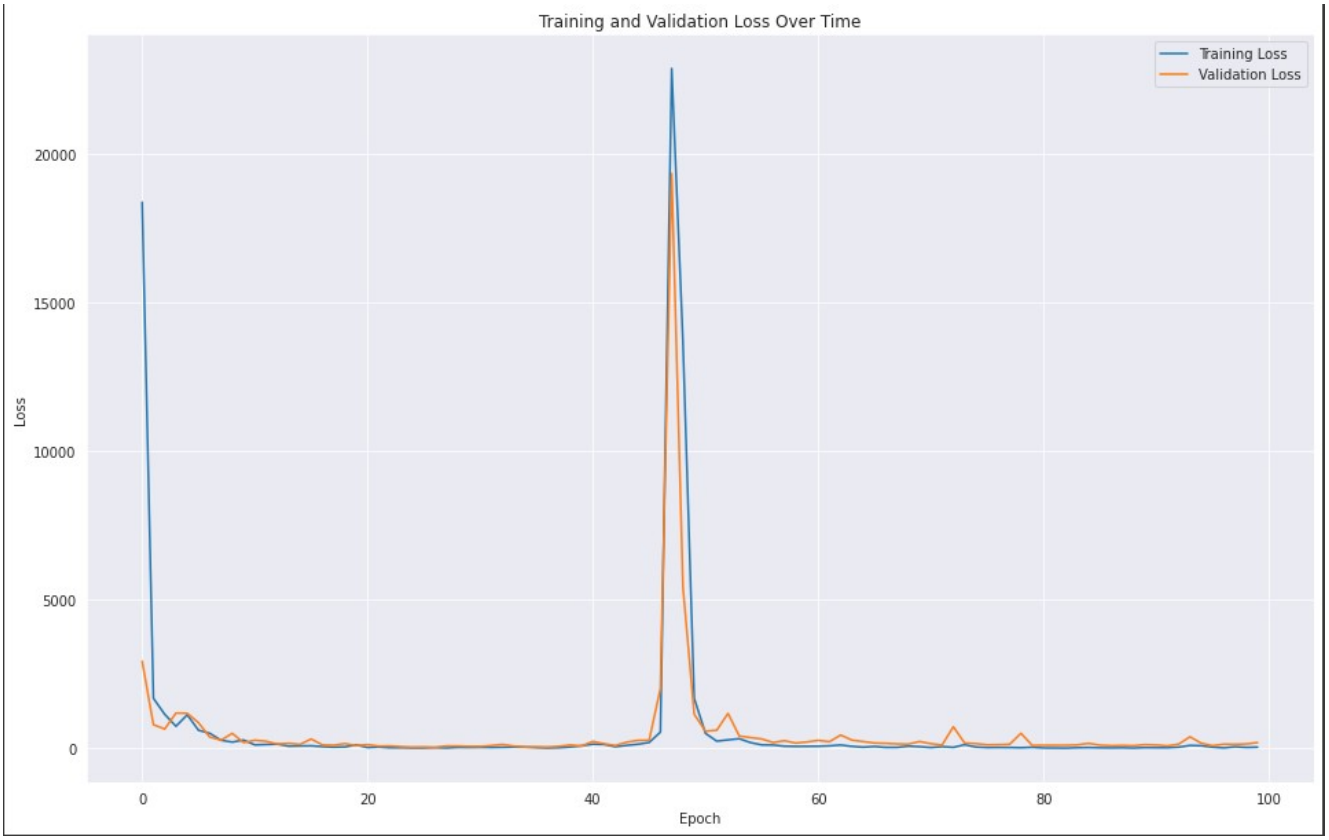
Έχοντας πλέον ορίσει το δίκτυο περνάμε στην φάση της εκπαίδευσης. Αρχικά εκτελούμε την εντολή compile για την κατασκευή του δικτύου. Επιλέγουμε συνάρτηση κόστους την categorical crossentropy, optimizer τον adam και μετρική την ακρίβεια. Τέλος με την εντολή fit() εκπαιδεύουμε το δίκτυο. Περνάμε σαν ορίσματα φυσικά το training set και το validation set και ορίζουμε το πλήθος των εποχών ίσο με 100. Παίρνουμε ενημέρωση για κάθε εποχή αναφορικά με την τιμή της συνάρτησης κόστους και της ακρίβειας στα training και validation sets. Ακολουθεί δείγμα αυτού.

```
Epoch 1/100
49/49 [==============================] - 1s 8ms/step - loss: 27189.2678 - accuracy: 0.1473 - val_loss: 2937.3733 - val_accuracy: 0.3499
Epoch 2/100
49/49 [==============================] - 0s 6ms/step - loss: 2177.4995 - accuracy: 0.3685 - val_loss: 802.2663 - val_accuracy: 0.4632
Epoch 3/100
49/49 [==============================] - 0s 6ms/step - loss: 1249.2112 - accuracy: 0.4577 - val_loss: 655.5023 - val_accuracy: 0.5070
Epoch 4/100
49/49 [==============================] - 0s 6ms/step - loss: 657.6878 - accuracy: 0.5308 - val_loss: 1193.3932 - val_accuracy: 0.4195
Epoch 5/100
49/49 [==============================] - 0s 6ms/step - loss: 1131.7989 - accuracy: 0.4648 - val_loss: 1185.9327 - val_accuracy: 0.4533
```

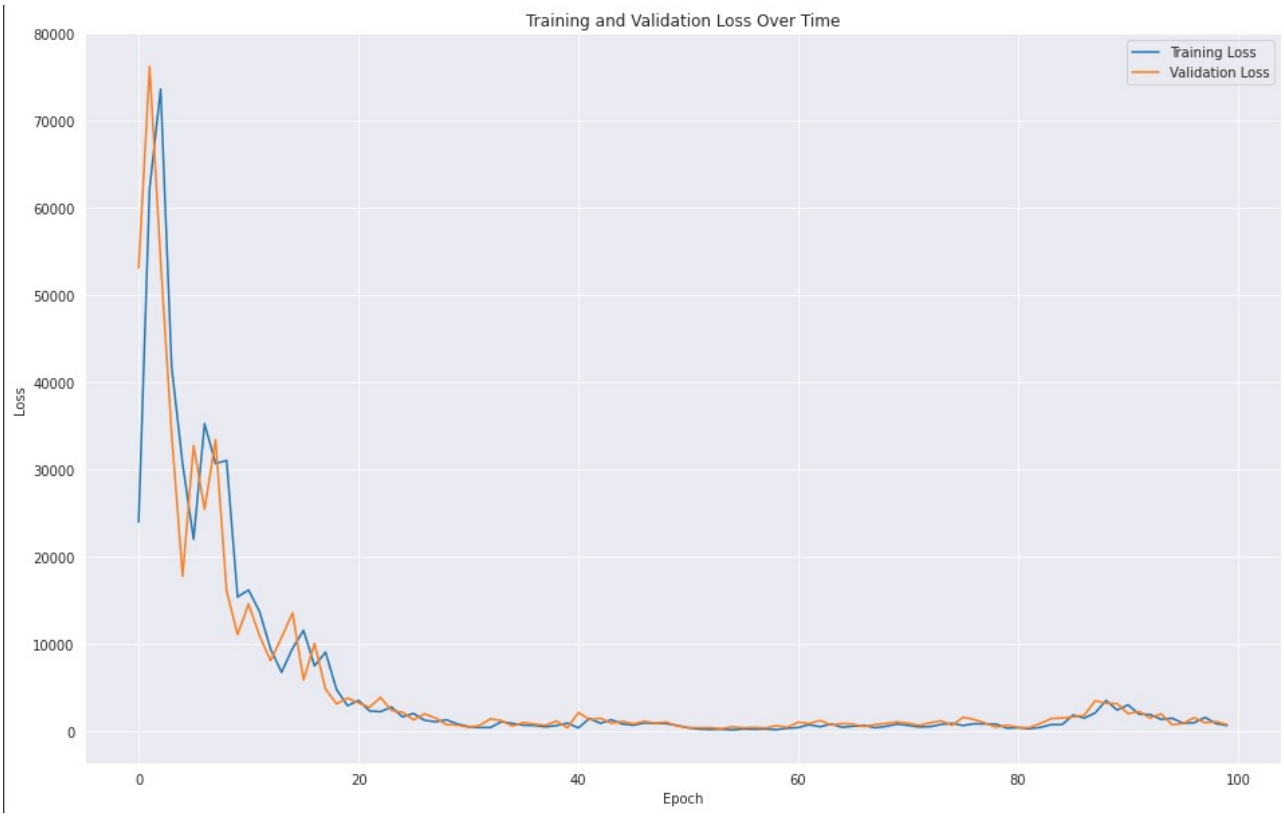
Η συγκεκριμένη αρχιτεκτονική πετυχαίνει μία ακρίβεια 64.8% .

```
16/16 [==============================] - 0s 4ms/step - loss: 1.2994 - acc: 0.6481
===Testing loss and accuracy===
Test loss:  1.2994415760040283
Test accuracy:  0.6481113433837891
```

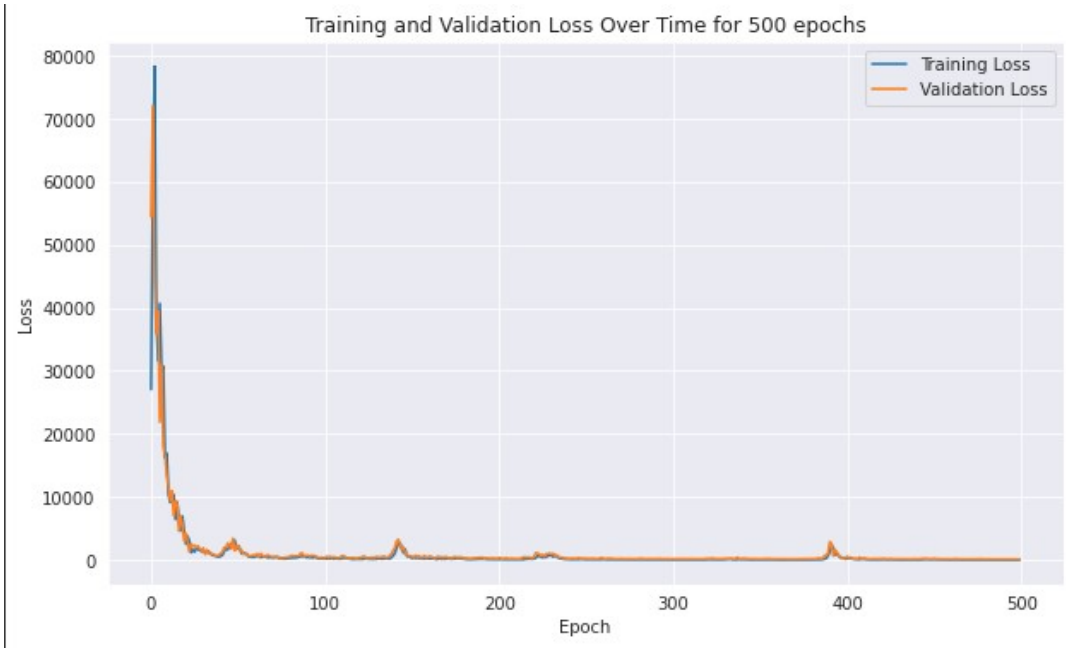
Στην συνέχεια χρησιμοποιώντας αυτά τα στοιχεία φτιάχνουμε το γράφημα train – validation loss.

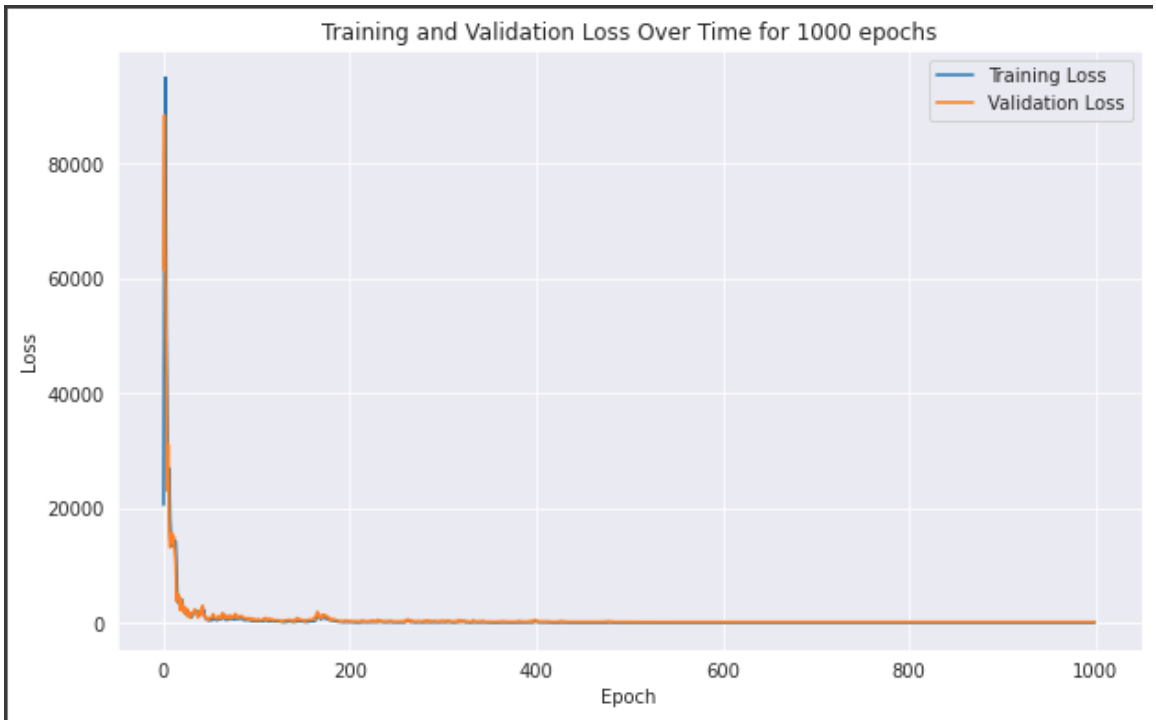


Όπως εξηγήσαμε και στην εισαγωγή το training loss είναι μία καλή μετρική για να συμπεράνουμε αν το δίκτυο είναι underfited. Βλέπουμε ότι το training loss δεν πέφτει αρκετά χαμηλά. Χρησιμοποιώντας ως συνάρτηση κόστους την categorical crossentropy θα θέλαμε το training loss να είναι τουλάχιστον κοντά στο 1. Μετά από 100 εποχές βλέπουμε ότι το training loss κυμαίνεται μεταξύ 25 και 50, πολύ μεγαλύτερο από το επιθυμητό όριο. Επίσης βλέπουμε από το train-val loss graph ότι ανά διαστήματα προκύπτουν spikes, μικρά και μεγάλα. Κύρια αιτία για αυτά είναι το batch size που επιλέγουμε κατά το compilation του δικτύου. Σε αυτή την φάση έχουμε αφήσει την default τιμή του, δηλαδή 32 δείγματα ανά epoch. Δηλαδή από το σύνολο των δεδομένων εκπαίδευσης σε κάποια εποχή μπορεί τυχαία να επιλέγονται δείγματα δεδομένων που δεν συμβάλουν στην βελτίωση των βαρών. Ακόμη, λόγω του underfitting, τα δεδομένα που επιλέγονται σε κάποια εποχή μπορεί να έχουν χαρακτηριστικά τα οποία το δίκτυο δεν έχει μοντελοποιήσει ακόμα, ή ακόμη χαρακτηριστικά τα οποία δεν μπορεί καθόλου να μοντελοποιήσει λόγο της απλής αρχιτεκτονικής του. Ας δοκιμάσουμε να ορίσουμε το batch size ίσο με 512 δείγματα δεδομένων.

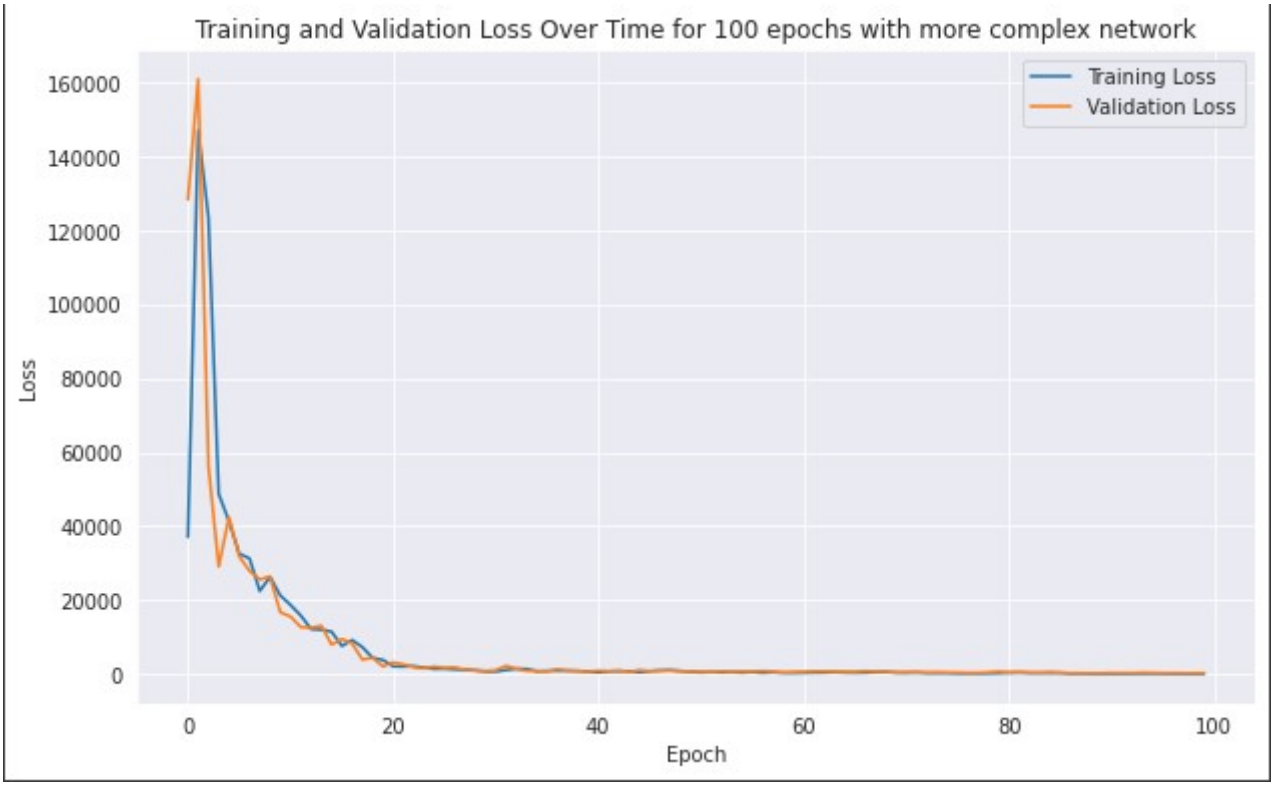


Τα μεγάλα spikes εξαφανίζονται και οι μετρικές training και validation loss αποκτούν ένα trend μείωσης, ωστόσο το δίκτυο παραμένει underfitted. Όπως αναφέραμε και στην εισαγωγή οι δύο πιο συνηθισμένες τεχνικές που δίνουν λύση στο συγκεκριμένο πρόβλημα είναι η αύξηση της πολυπλοκότητας του δικτύου και η αύξηση των εποχών εκπαίδευσης. Ας δοκιμάσουμε αρχικά να αυξήσουμε το πλήθος των εποχών εκπαίδευσης σε 500 και 1000, κρατώντας όλα τα υπόλοιπα ίδια όπως και πριν με batch size = 512.

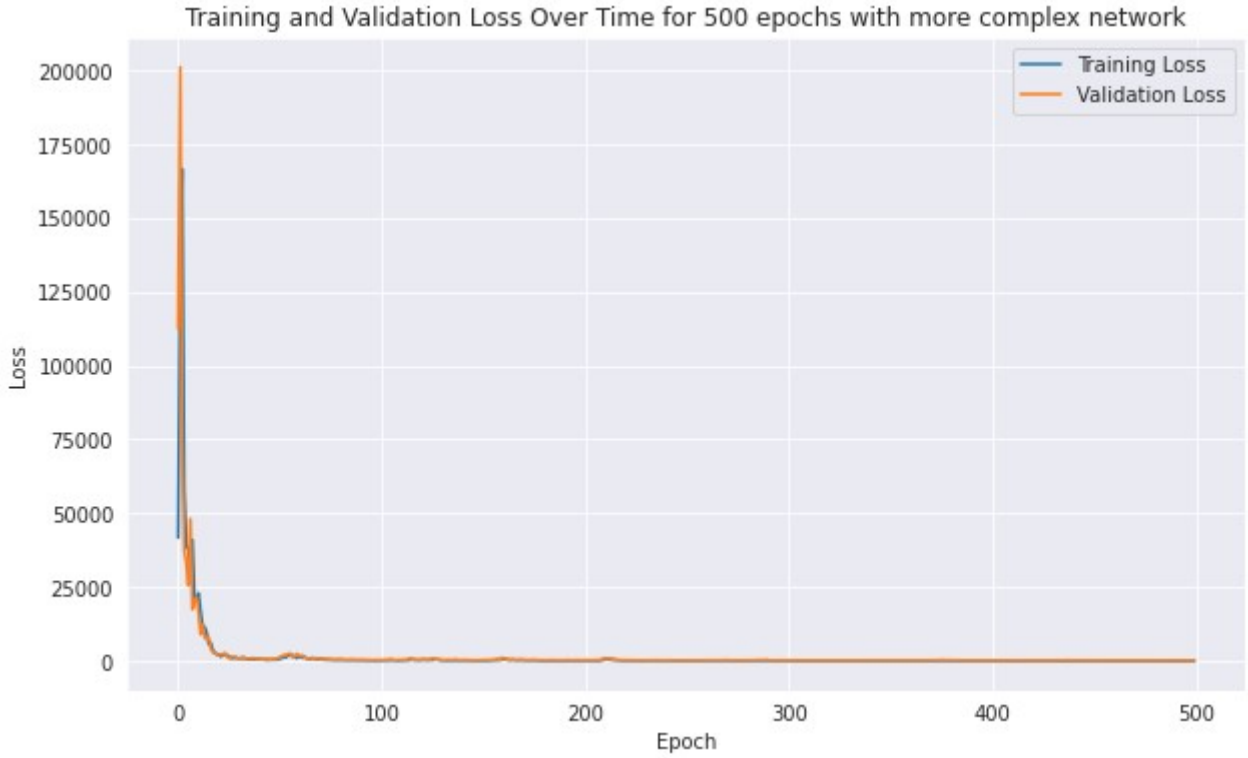




Παρατηρούμε ότι τελικά ,μετά τις 500 εποχές, έχουμε τιμή training loss κάτω από το 1. Ωστόσο η τιμή του validation loss δεν μειώνεται και παραμένει σταθερά κοντά στο 100. Ας δοκιμάσουμε τώρα μια πιο πολύπλοκη αρχιτεκτονική δικτύου. Θα προσθέσουμε ένα ακόμη κρυφό επίπεδο 512 νευρώνων ενώ κατά την εκπαίδευση κρατάμε το πλήθος των εποχών στις 100 και το Batch size = 512.



Βλέπουμε ότι η απόδοση του δικτύου είναι πιο σταθερή από αυτήν του πιο απλού δικτύου με το ίδιο πλήθος εποχών. Όμως το training loss δεν πέφτει κάτω από 90. Αυξάνουμε τις εποχές σε 500.

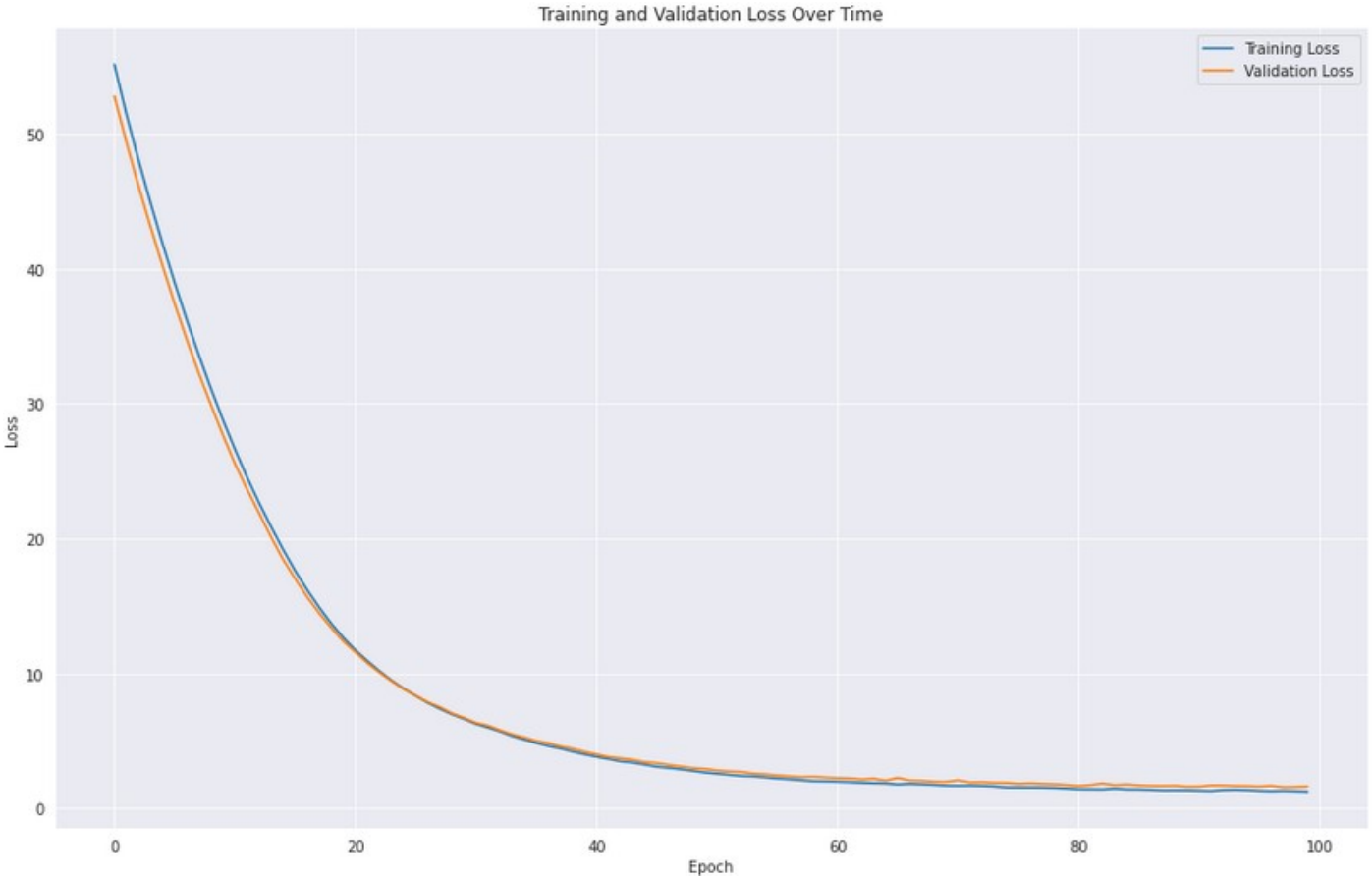



```
Epoch 495/500
4/4 [==============================] - 0s 57ms/step - loss: 1.9548 - accuracy: 0.9686 - val_loss: 61.9182 - val_accuracy: 0.7694
Epoch 496/500
4/4 [==============================] - 0s 28ms/step - loss: 0.6234 - accuracy: 0.9840 - val_loss: 75.3618 - val_accuracy: 0.7316
Epoch 497/500
4/4 [==============================] - 0s 27ms/step - loss: 2.8924 - accuracy: 0.9682 - val_loss: 62.3943 - val_accuracy: 0.7594
Epoch 498/500
4/4 [==============================] - 0s 27ms/step - loss: 1.7655 - accuracy: 0.9569 - val_loss: 68.6229 - val_accuracy: 0.7674
Epoch 499/500
4/4 [==============================] - 0s 27ms/step - loss: 1.2482 - accuracy: 0.9741 - val_loss: 65.0230 - val_accuracy: 0.7773
Epoch 500/500
4/4 [==============================] - 0s 26ms/step - loss: 0.7634 - accuracy: 0.9779 - val_loss: 65.5899 - val_accuracy: 0.7614
Time taken: 53.6 seconds
```

Παρατηρούμε ότι πετυχαίνουμε τιμή training loss μικρότερη του 1, πράγμα για το οποίο χρειάστηκε πλήθος εποχών μεγαλύτερο του 500 για να το πετύχουμε με το αρχικό δίκτυο.

Άρα συνολικά παραπατούμε ότι η αύξηση των εποχών και η αύξηση της πολυπλοκότητας του δικτύου συμβάλουν στην μείωση του underfitting. Όμως παρατηρούμε επίσης το εξής. Με την εφαρμογή των προηγούμενων μεθόδων πετύχαμε μείωση του training loss, παρόλα αυτά το validation loss δεν μειώθηκε ανάλογα. Άρα στην προσπάθειά μας να λύσουμε το underfitting προκαλέσαμε overfitting. Άρα το επόμενο βήμα θα είναι να εντοπίσουμε την αρχιτεκτονική με το καλύτερο bias-variance trade off, πράγμα που θα γίνει στο τελευταίο ερώτημα της άσκησης. Μία κίνηση για να λύσουμε το πρόβλημα του overfitting είναι το data augmentation. Κατά την μέθοδο αυτή αυξάνουμε το μέγεθος των δεδομένων εφαρμόζοντας ένα πλήθος μετασχηματισμών στο σύνολο των φωτογραφιών που ήδη

έχουμε. Αποθηκεύουμε τις φωτογραφίες που προκύπτουν από την διαδικασία της επαύξησης. Είναι βολικό να φτιάξουμε ένα νέο dataset το οποίο αποτελείται από τις αρχικές φωτογραφίες ενώ και από τις φωτογραφίες που προκύπτουν από την επαύξηση. Η διαδικασία αυτή γίνεται αυτόματα με script. Αφού έχουμε ετοιμάσει το νέο επαυξημένο σύνολο δεδομένων προχωράμε στην εκπαίδευση του δικτύου. Χρησιμοποιούμε την απλή αρχιτεκτονική του ερωτήματος a.

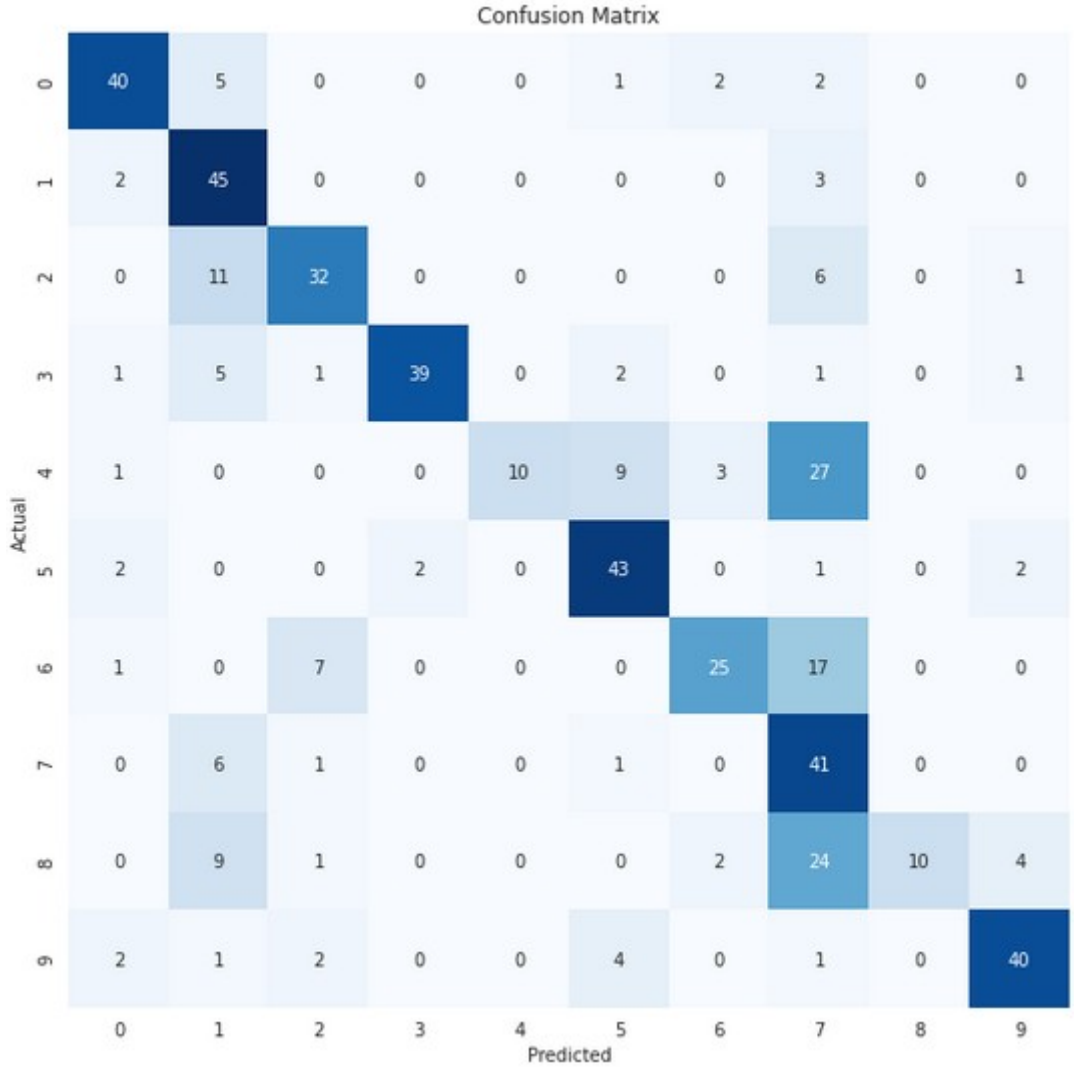


Βλέπουμε ότι η συμπεριφορά το δικτύου είναι πιο ομαλή, ενώ επίσης οι τιμές τις συνάρτησης κόστους για τα train και test set σχεδόν ταυτίζονται για όλη την διάρκεια της εκπαίδευσης. Αυτό σημαίνει ότι έχει μειωθεί ή ένταση του overfitting. Επίσης σημαντική βελτίωση εντοπίζουμε και όσο αφορά την ακρίβεια η οποία είναι 74%.

```
16/16 [==============================] - 0s 4ms/step - loss: 1.6083 - acc: 0.7416
===Testing loss and accuracy===
Test loss:  1.6082500219345093
Test accuracy:  0.7415506839752197
```

Ας θυμηθούμε ότι η ίδια αρχιτεκτονική εκπαιδευμένη στο αρχικό dataset είχε ακρίβεια 64.8%. Άρα βλέπουμε την σημαντική βελτίωση που φέρνει η επαύξηση δεδομένων.

Τελικά από τα γραφήματα φαίνεται ότι το βέλτιστο πλήθος εποχών εκπαίδευσής του αρχικού δικτύου είναι 50. Για το συγκεκριμένο πλήθος εποχών φτιάχνουμε ένα confusion matrix καθώς εξάγουμε και τις ζητούμενες μετρικές.




```
Classification Report:
----------------------
              precision    recall  f1-score

           0       0.82      0.80      0.81
           1       0.55      0.90      0.68
           2       0.73      0.64      0.68
           3       0.95      0.78      0.86
           4       1.00      0.20      0.33
           5       0.72      0.86      0.78
           6       0.78      0.50      0.61
           7       0.33      0.84      0.48
           8       1.00      0.20      0.33
           9       0.83      0.80      0.82
```

c) Κανονικοποίηση και Relu

Η κανονικοποίηση των δεδομένων αποτελεί πολύ σημαντικό βήμα για την βέλτιστη εκπαίδευση του νευρωνικού δικτύου. Όπως θα διαπιστώσουμε σε αυτήν την ενότητα, η κανονικοποίηση είναι σε θέση από μόνη της να φέρει σημαντική βελτίωση στην απόδοση του δικτύου. Όπως εξηγήσαμε και στην ενότητα 1.a οι εικόνες τροφοδοτούνται στο δίκτυο σε μορφή διανύσματος μεγέθους 30000 στοιχείων. Κάθε ένα στοιχείο είναι ένας ακέραιος μεταξύ 0 και 255, τιμή που αντιπροσωπεύει την ένταση του εκάστοτε χρωματικού καναλιού για κάθε εικονοστοιχείο. Με την εφαρμογή κανονικοποίησης φέρνουμε τις τιμές των εικονοστοιχείων στο εύρος [-1, 1]. Αυτή η μείωση του εύρους των τιμών εισόδου στο δίκτυο διευκολύνει σε μεγάλο βαθμό την διαδικασία της εκπαίδευσης.

Επίσης στα επίπεδα Dense ορίζουμε ως συνάρτηση ενεργοποίησης την Relu. Η συγκεκριμένη συνάρτηση ορίζεται ως εξής $h=max(0,a)$ $όπου$ $a=Wx+b$. Ένα από τα πλεονεκτήματα της relu είναι ότι ελαχιστοποιεί την πιθανότητα εμφάνισης του φαινομένου vanishing gradient. Το συγκεκριμένο φαινόμενο εμφανίζεται κυρίως σε μεγάλα (βαθιά) δίκτυα στα οποία λόγο του μεγάλου πλήθους επιπέδων η κλήση της συνάρτησης κόστους πλησιάζει το μηδέν. Έτσι καθίσταται δύσκολη η εκπαίδευση του δικτύου γιατί τα βάρη του δεν θα ανανεώνονται ή θα ανανεώνονται με πολύ αργό ρυθμό. Η relu επιταχύνει την ταχύτητα σύγκλησης του αλγορίθμου εκπαίδευσης. Τα αποτελέσματα επιβεβαιώνουν την αποδοτικότητα των δύο αυτών μεθόδων.

```
16/16 [==============================] - 0s 4ms/step - loss: 0.9530 - acc: 0.6899
===Testing loss and accuracy===
Test loss:  0.9529592990875244
Test accuracy:  0.6898608207702637
```

Εφαρμόζοντας αυτές τις δύο μεθόδους στην αρχική απλή αρχιτεκτονική παίρνουμε ακρίβεια 68.9%

d) Εφαρμογή batch normalization

Παρά τα πολλά οφέλη που παρουσιάζουν τα νευρωνικά δίκτυα και κατ΄επέκταση και τα ANN, μπορεί να παρουσιάζεται μεγάλη καθυστέρηση κατά την εκπαίδευσή τους ενώ είναι ευάλωτα σε overfitting. Το batch normalization είναι μία μέθοδος που έρχεται να δώσει λύσεις σε αυτά τα προβλήματα. Το normalization σαν έννοια αναφέρεται σε τεχνική που εφαρμόζεται σε επίπεδο preprocessing, ώστε να βρίσκονται όλα τα δεδομένα στο ίδιο εύρος.

Το batch normalization είναι μία μέθοδος normalization η οποία γίνεται ενδιάμεσα στα επίπεδα του νευρωνικού δικτύου αντί στα δεδομένα πριν εισέλθουν σε αυτό. Η λειτουργία του βασίζεται στο ότι κανονικοποιεί τις εξόδους των νευρώνων του προηγούμενου επιπέδου, ουσιαστικά εφαρμόζεται ένας μετασχηματισμός ώστε η έξοδος των νευρώνων του προηγούμενου επιπέδου να διατηρούν μέση τιμή κοντά στο 0 και τυπική απόκλιση κοντά στο 1. Χρησιμοποιώντας batch normalization επίπεδα, η φάση του training γίνεται αποδοτικότερα και από άποψη χρόνου και από άποψη ποιότητας. Επίσης μπορεί να χρησιμοποιηθεί ως μέθοδος regularzation ώστε να αποφεύγετε το Overfitting. Συγκεκριμένα για την υλοποίησή μας χρησιμοποιούμε το επίπεδο BatchNormalization το οποίο προσφέρεται από την βιβλιοθήκη Keras. Προσθέτουμε το BatchNormalization επίπεδο στο αρχικό δίκτυο ως εξής.

```
model_d = models.Sequential()

model_d.add(tf.keras.Input(shape=(30000)))
model_d.add(layers.BatchNormalization())
model_d.add(layers.Dense(256,activation='relu'))
model_d.add(layers.BatchNormalization())
model_d.add(layers.Dense(128,activation='relu'))
model_d.add(layers.Dense(10, activation='softmax'))

model_d.summary()
```

```
16/16 [==============================] - 0s 5ms/step - loss: 0.7557 - acc: 0.7594
===Testing loss and accuracy===
Test loss:  0.755653440952301
Test accuracy:  0.7594433426856995
```

Μετά από εκπαίδευση 50 εποχών το δίκτυο αποδίδει με μία ακρίβεια 76%, μία πολύ σημαντική βελτίωση σε σχέση με την αρχιτεκτονική του πρώτου ερωτήματος. Η εκπαίδευση διαρκεί 10.8 δευτερόλεπτα, έχουμε δηλαδή μία αύξηση στον χρόνο εκπαίδευσης κατά 1.5 δευτερόλεπτα από την αρχιτεκτονική του ερωτήματος c. Η αύξηση αυτή είναι αναμενόμενη, καθώς με την προσθήκη επιπλέον επιπέδων αυξάνεται και ο υπολογιστικός φόρτος του δικτύου. Η μέθοδος batch normalization δεν μειώνει τον υπολογιστικό φόρτο του δικτύου. Όμως θα μπορούσαμε να επωφεληθούμε σε επίπεδο χρόνου από την εφαρμογή της συγκεκριμένης μεθόδου αν αυξάναμε το learning rate. Με χρήση batch normalization μπορούμε με ασφάλεια να αυξήσουμε το learning rate διατηρώντας την ποιότητα της εκπαίδευσης, μειώνοντας όμως έτσι τον χρόνο εκπαίδευσης. Αυτό αποδεικνύεται ιδιαίτερα χρήσιμο κατά την εκπαίδευση βαθέων νευρωνικών δικτύων (Deep Neural Networks).

e) Βέλτιστη αρχιτεκτονική

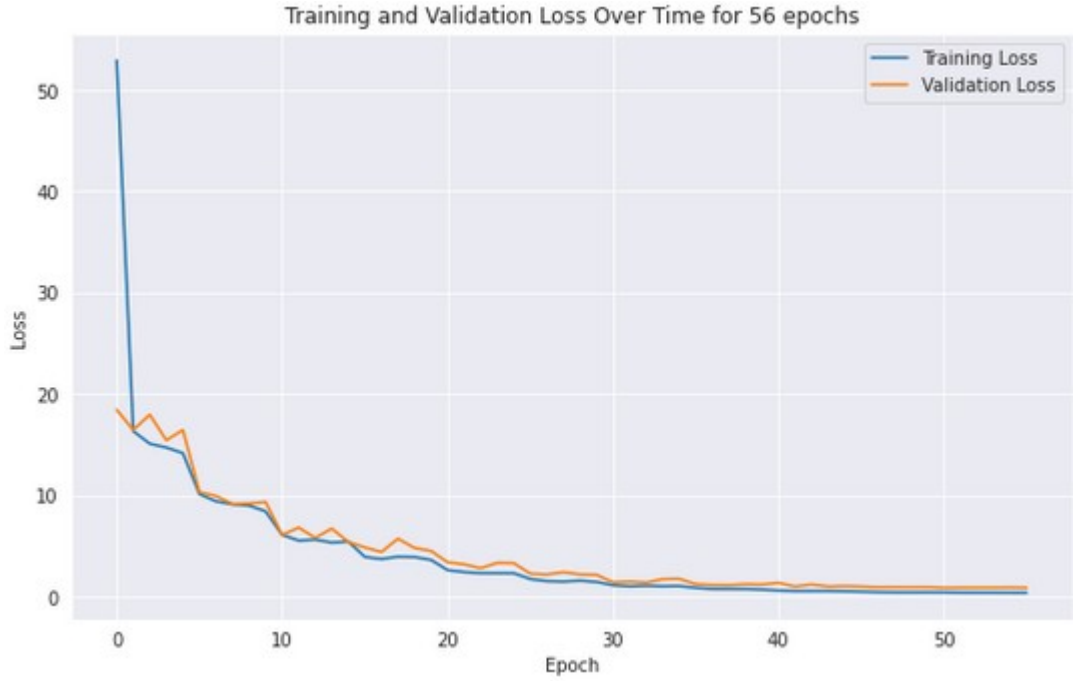
Για να ορίσουμε την βέλτιστη αρχιτεκτονική θα στηριχθούμε στα συμπεράσματα τα οποία εξάγαμε στις προηγούμενες ενότητες. Σίγουρα θέλουμε ένα δίκτυο μεγαλύτερης πολυπλοκότητας από το αρχικό ώστε να αντιμετωπίσουμε το πρόβλημα του underfitting. Σχετικά με το πλήθος των εποχών εκπαίδευσης θα χρησιμοποιήσουμε την μέθοδος early stopping, κατά την οποία η εκπαίδευσης σταματάει αυτόματα εφόσον ισχύουν ορισμένες συνθήκες. Η συνθήκη για τον πρόωρο τερματισμό είναι η μικρή μεταβολή της τιμής validation loss για 5 εποχές. Επίσης θα κάνουμε χρήση του augmented dataset, το οποίο μας έδωσε πολύ θετικά αποτελέσματα στην απλή αρχιτεκτονική. Θα εφαρμόσουμε dimensionality reduction χρησιμοποιώντας τις εικόνες σε ασπρόμαυρη μορφή. Ακόμη, θα επιλέξουμε την συνάρτηση ενεργοποίησης relu στα κρυφά επίπεδα, ενώ επίσης θα κανονικοποιήσουμε τα δεδομένα. Τέλος θα χρησιμοποιήσουμε τις τεχνικές learning rate scheduler και model checkpoint. Κατά τη μέθοδο learning rate scheduler ορίζουμε μία συνάρτηση η οποία ανά ένα συγκεκριμένο πλήθος εποχών μειώνεται η τιμή του ρυθμού εκπαίδευσης. Η μέθοδος model checkpoint λειτουργεί αποθηκεύοντας το καλύτερο δίκτυο που προκύπτει κατά την εκπαίδευση, οπότε ακόμη και αν οι εποχές είναι πολλές θα κρατάμε το καλύτερο δίκτυο που προέκυψε. Το καλύτερο δίκτυο που προκύπτει ορίζουμε να είναι αυτό με την μεγαλύτερη τιμή του validation accuracy. Λαμβάνοντας όλα αυτά υπ όψιν και μετά από δοκιμές προκύπτει η καλύτερη αρχιτεκτονική να είναι η εξής.

```
model = models.Sequential()

model.add(tf.keras.Input(shape=(10000)))
model.add(layers.Dense(1024,activation='relu',kernel_regularizer=tf.keras.regularizers.l1(0.001)))
model.add(layers.BatchNormalization())
model.add(layers.Dense(512,activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(128,activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dropout(0.6))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

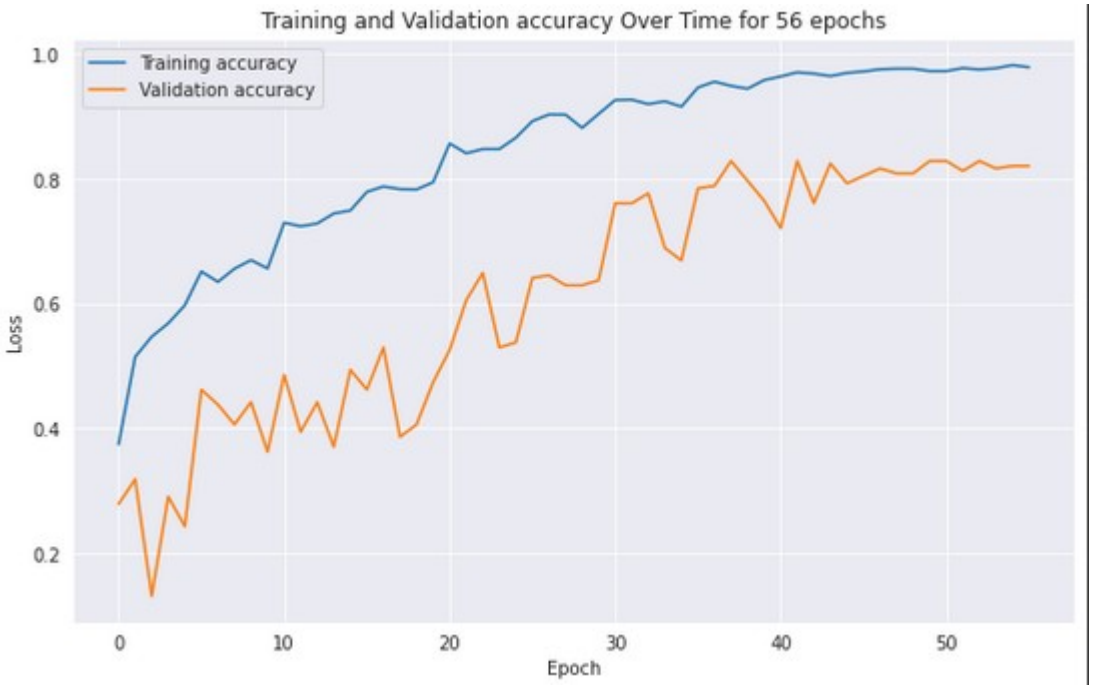
Έχουμε τέσσερα κρυφά Dense επίπεδα για τα οποία ορίζουμε συνάρτηση ενεργοποίησης την relu. Στον πρώτο κρυφό επίπεδο Dense εφαρμόζουμε και regularization. Μεταξύ όλων των κρυφών επιπέδων χρησιμοποιούμε επίσης επίπεδα batch normalization, ενώ προσθέτουμε και ένα επίπεδο dropout μεταξύ του τελευταίου κρυφού επιπέδου και του επιπέδου εξόδου.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 1024)              10241024
_________________________________________________________________
batch_normalization_3 (Batch (None, 1024)              4096
_________________________________________________________________
dense_6 (Dense)              (None, 512)               524800
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dense_7 (Dense)              (None, 128)               65664
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                650
=================================================================
Total params: 10,847,050
Trainable params: 10,843,722
Non-trainable params: 3,328
_________________________________________________________________
```

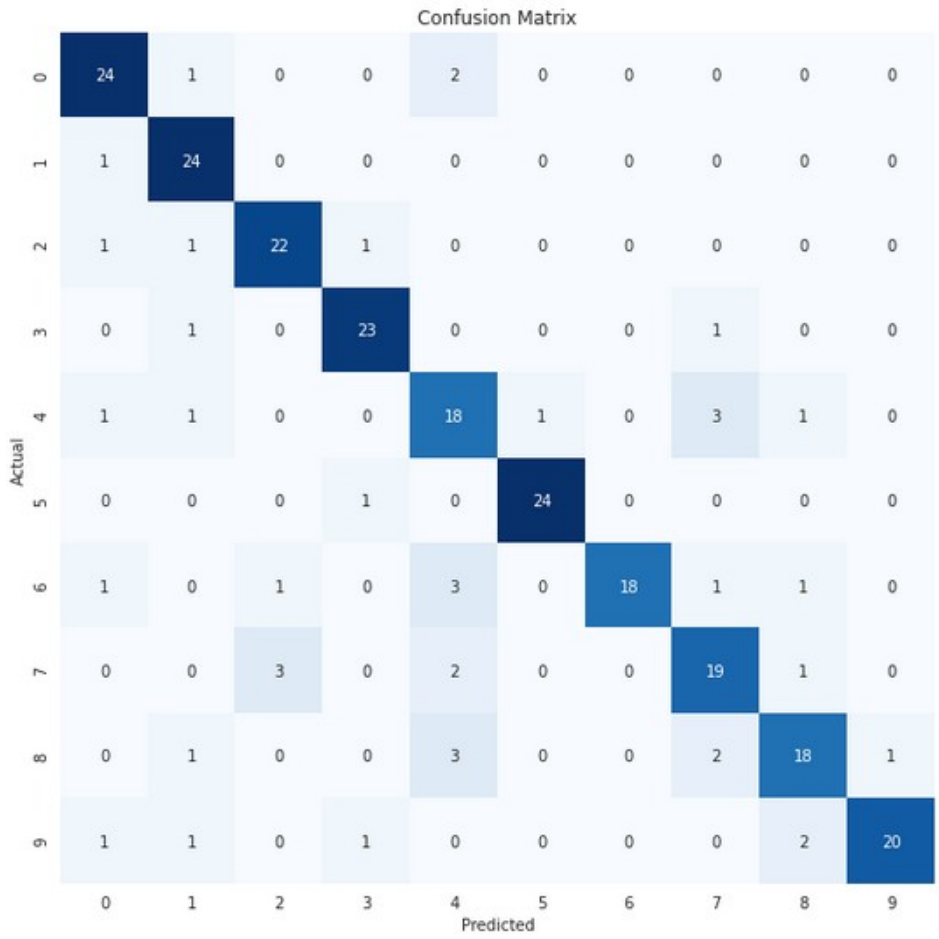
Βλέπουμε ότι έχουμε 3,328 Non-trainable parameters τα οποία οφείλονται στα επίπεδα batch normalization. Το δίκτυο με την βέλτιστη αρχιτεκτονική δίνει ακρίβεια 83%, μία μεγάλη βελτίωση από το αρχικό απλό ANN.



Η μέθοδος early stopping βλέπουμε ότι σταμάτησε την εκπαίδευση στην εποχή 56 ώστε να προλάβει το overfitting.



Επίσης βλέπουμε ότι τα γραφήματα train/test loss και train/test accuracy αρχίζουν μετά τις 50 εποχές να σταθεροποιούνται, οπότε είναι ασφαλές να πούμε ότι το δίκτυο έχει φτάσει στα όρια της ικανότητάς του. Τέλος βλέπουμε για την βέλτιστη αρχιτεκτονική το confusion matrix καθώς και τις μετρικές precision, recall, f1-score.



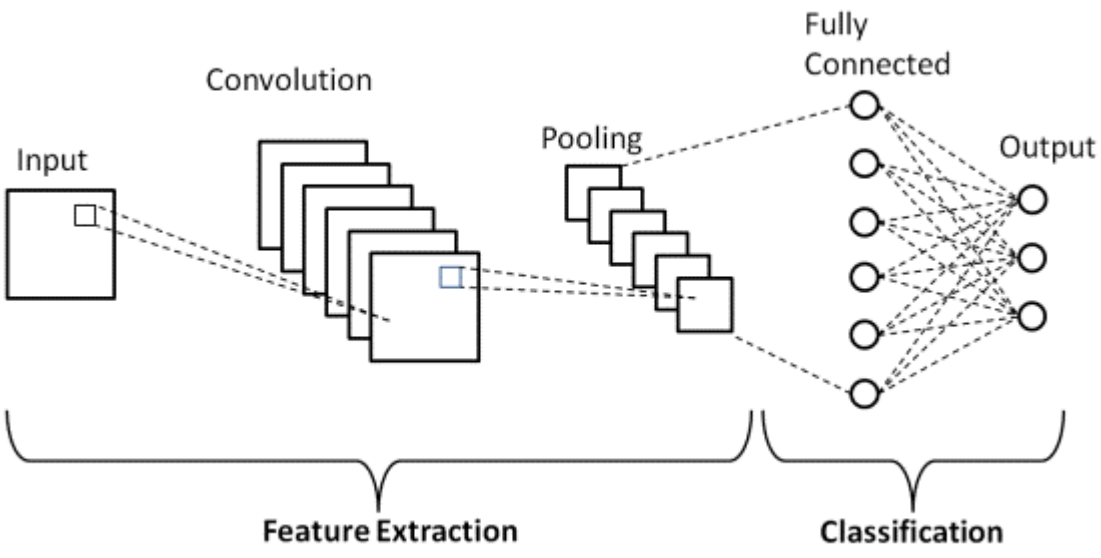
Συνελικτικό Νευρωνικό Δίκτυο (CNN)

Λειτουργία

Ας εξετάσουμε την λειτουργία των συνελικτικών νευρωνικών δικτύων επικεντρώνοντας την προσοχή μας στην επίλυση ενός classification task. Ένα CNN παίρνει σαν είσοδο μία εικόνα, την επεξεργάζεται και την κατηγοριοποιεί σε μία κατηγορία από ένα σύνολο κατηγοριών που έχουμε ορίσει. Όπως αναφέραμε και στην ανάλυση του απλού νευρωνικού δικτύου (ANN) οι υπολογιστές και κατ' επέκταση και τα δίκτυα αντιλαμβάνονται μία φωτογραφία σε μορφή πίνακα αριθμών. Ένα CNN χρησιμοποιεί αυτούς τους αριθμούς για να αναγνωρίσει δομές εντός τις εικόνας.

Το πρώτο επίπεδο νευρώνων προσπαθεί να εξάγει χαρακτηριστικά χαμηλού επιπέδου από την εικόνα, τέτοια είναι γραμμές οποιασδήποτε κατεύθυνσης, περιοχές συγκεκριμένου χρώματος και άλλα. Όσο προχωράμε στα βαθύτερα επίπεδα το δίκτυο καταφέρνει να αναγνωρίζει πιο ολοκληρωμένες δομές οι οποίες όμως ακόμη από μόνες τους δεν αποτυπώνουν ολόκληρη την πληροφορία της αρχικής εικόνας, τέτοιες δομές είναι περιγράμματα προσώπων, ματιών, χεριών, και άλλα, ανάλογα με το περιεχόμενο της εικόνας. Στην έξοδό του το δίκτυο αποτελείται από μία δομή classifier, η οποία είναι υπεύθηνη για την επιλογή της σωστής κλάσης ανάλογα με τις εξόδους που του προωθούν τα προηγούμενα επίπεδα. Συνολικά δηλαδή, ένα CNN χαρακτηρίζεται από μία ακολουθία από φίλτρα τα οποία ενεργοποιούνται με την ύπαρξη συγκεκριμένων δομών στην εκάστοτε εικόνα. Η ενεργοποίηση ή μη αυτών των φίλτρων είναι που δίνει την κατάλληλη πληροφορία στον classifier ώστε να πάρει την σωστή απόφαση αναφορικά με την κλάση της εικόνας. Το εντυπωσιακό και άκρος χρήσιμο των CNN είναι ότι αυτά τα φίλτρα παράγονται κατά την εκπαίδευση και δεν δίνονται εξωτερικά από εμάς.

Ας εξετάσουμε τώρα μερικά από τα βασικά επίπεδα από τα οποία αποτελείται ένα CNN. Διαχωρίζουμε την δομή ενός CNN σε δύο βασικά blocks, το feature extraction block και το classification block.

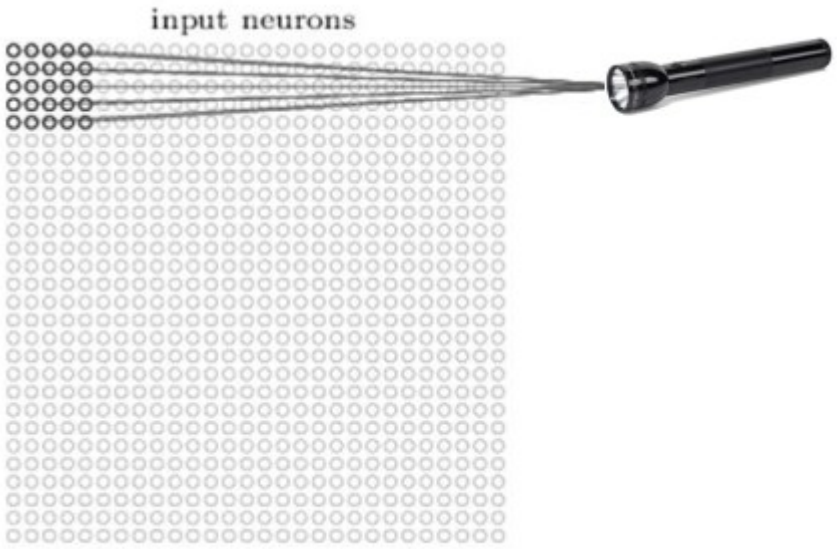


- Feature extraction block

- Convolutional layer

Το συνελλικτικό επίπεδο είναι το βασικό δομικό στοιχείο ενός CNN. Ευθύνεται για το μεγαλύτερο ποσοστό του υπολογιστικού φόρτου του δικτύου. Ο κύριος στόχος του συγκεκριμένου επιπέδου είναι η εξαγωγή χαρακτηριστικών από την εικόνα εισόδου όπως ακμές, χρώματα και γωνίες. Καθώς προχωράμε πιο βαθιά μέσα στο δίκτυο τότε αυτό αρχίζει και αναγνωρίζει πιο πολύπλοκα χαρακτηριστικά όπως συγκεκριμένα σχήματα, ψηφία ή μέρη προσώπου.

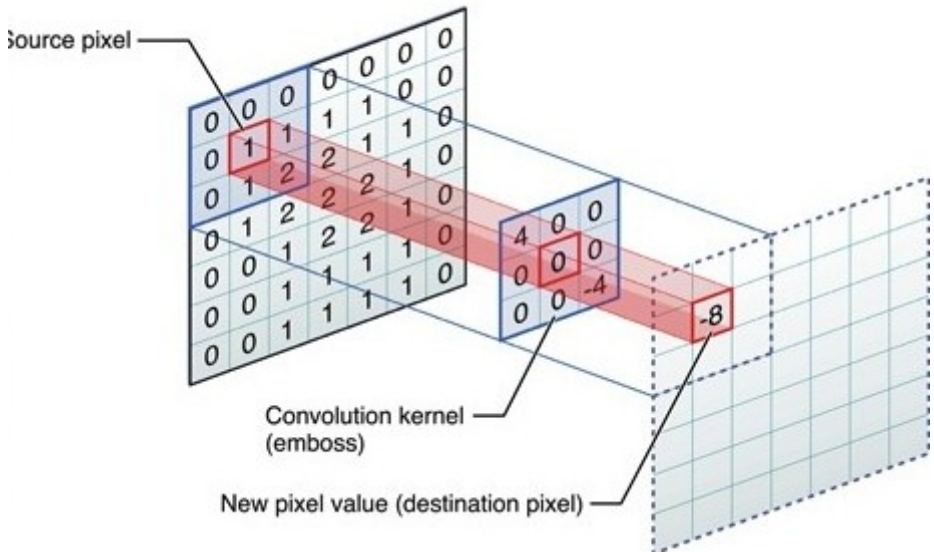
Μπορούμε να φανταστούμε την λειτουργία του με το εξής παράδειγμα. Έστω ότι έχουμε ένα φακό που φωτίζει ένα συγκεκριμένο μέρος της εικόνας, ας πούμε ότι το εύρος το οποίο είναι ικανό να φωτιστεί από τον φακό είναι μία περιοχή 5 x 5 εικονοστοιχεία. Ας φανταστούμε τώρα ότι ο φακός φωτίζει την πάνω αριστερή περιοχή της εικόνας. Στη συνέχεια ο φακός αρχίζει να σαρώνει όλη την εικόνα κατά μήκος κάθε γραμμής με βήμα ένα εικονοστοιχείο.



Ας αντικαταστήσουμε τώρα τον φακό με ένα φίλτρο (kernel) μεγέθους 3 x 3. Το φίλτρο δεν είναι τίποτε άλλο από έναν πίνακα αριθμών. Έστω ότι έχουμε το παρακάτω φίλτρο.

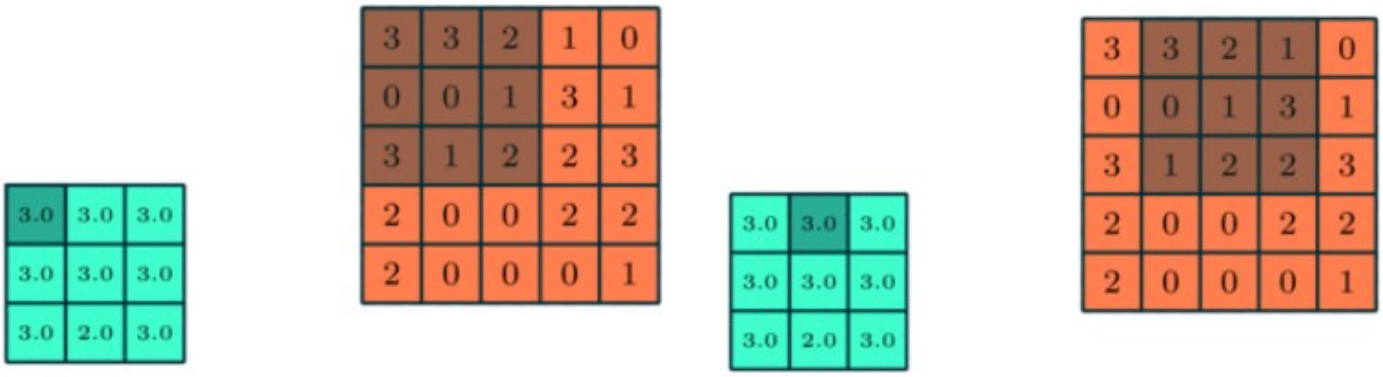
0	3	0
0	3	0
0	3	0

Καθώς λοιπόν το φίλτρο σαρώνει την εικόνα με τον ίδιο τρόπο που έκανε και ο φακός, εκτελεί το εσωτερικό γινόμενο μεταξύ του φίλτρου και του υποπίνακα της εικόνας που καλύπτει. Το αποτέλεσμα για ένα εικονοστοιχείο αποθηκεύεται σε έναν τρίτο πίνακα με όνομα feature map.



- <u>Pooling layer</u>

Αυτό το επίπεδο έχει ως στόχο την μείωση διαστάσεων του feature map. Σαν αποτέλεσμα προκαλεί και μείωση στον υπολογιστικό φόρτο που απαιτείται για την επεξεργασία των δεδομένων από το δίκτυο. Η λειτουργία του βασίζεται στο ότι εφαρμόζεται στο feature map και εξάγει τα πιο έντονα χαρακτηριστικά του. Η παρακάτω εικόνες δείχνουν ξεκάθαρα την λειτουργία του επιπέδου.



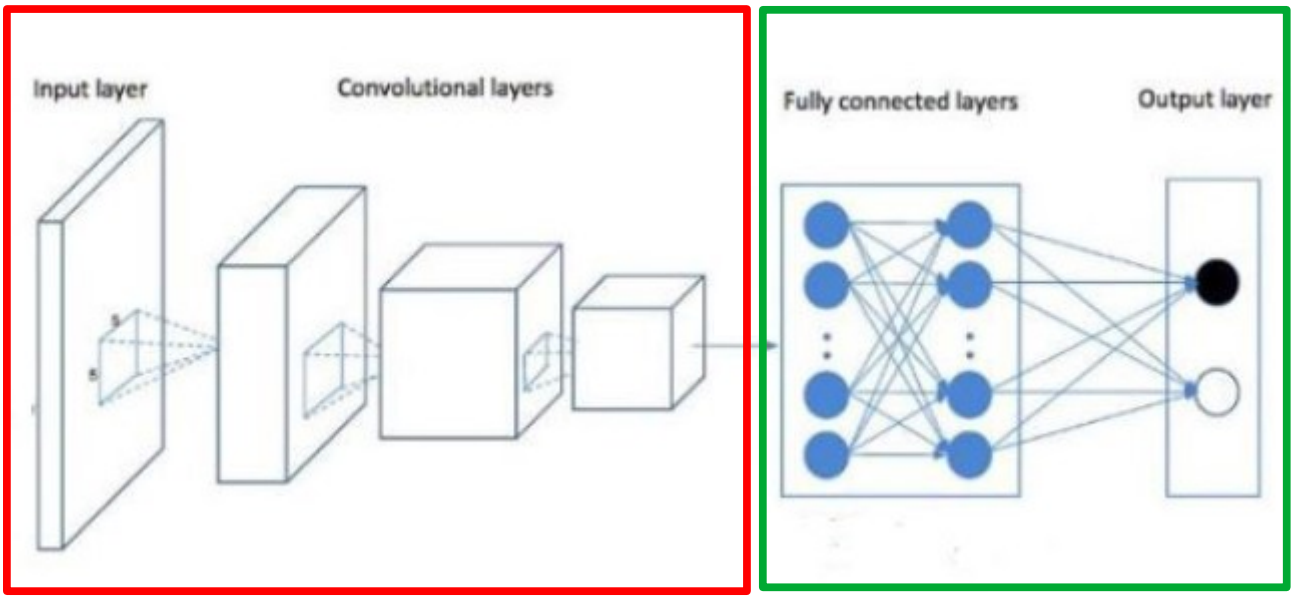
Με πορτοκαλί βλέπουμε το feature map ενώ η σκιασμένη περιοχή στο feature map είναι η περιοχή όπου δρα το επίπεδο pooling. Ο πίνακας με ανοιχτό πράσινο είναι το αποτέλεσμα του επιπέδου. Βλέπουμε ότι σαρώνονται περιοχές του feature

map μεγέθους 3 x 3 από τις οποίες επιλέγεται η μέγιστη τιμή. Βλέπουμε ότι ενώ το feature map είναι μεγέθους 5 x 5, το αποτέλεσμα μετά το επίπεδο pooling είναι ένας πίνακας μεγέθους μόλις 3 x 3.

Υπάρχουν δύο είδη μεθόδων pooling, το average pooling και το max pooling. Η διαφορά τους έγκειται στο ότι κατά τον average pooling παίρνουμε την μέση τιμή από όλες τις τιμές από αυτές στο pooling region ενώ στο max pooling παίρνουμε την μέγιστη τιμή από αυτές που βρίσκονται στο pooling region. Συνοπτικά μπορούμε να πούμε ότι μετά το επίπεδο Pooling παίρνουμε έναν πίνακα με μειωμένο πλήθος διαστάσεων ο οποίος θα μας βοηθήσει στο επόμενο βήμα.

- Classification block

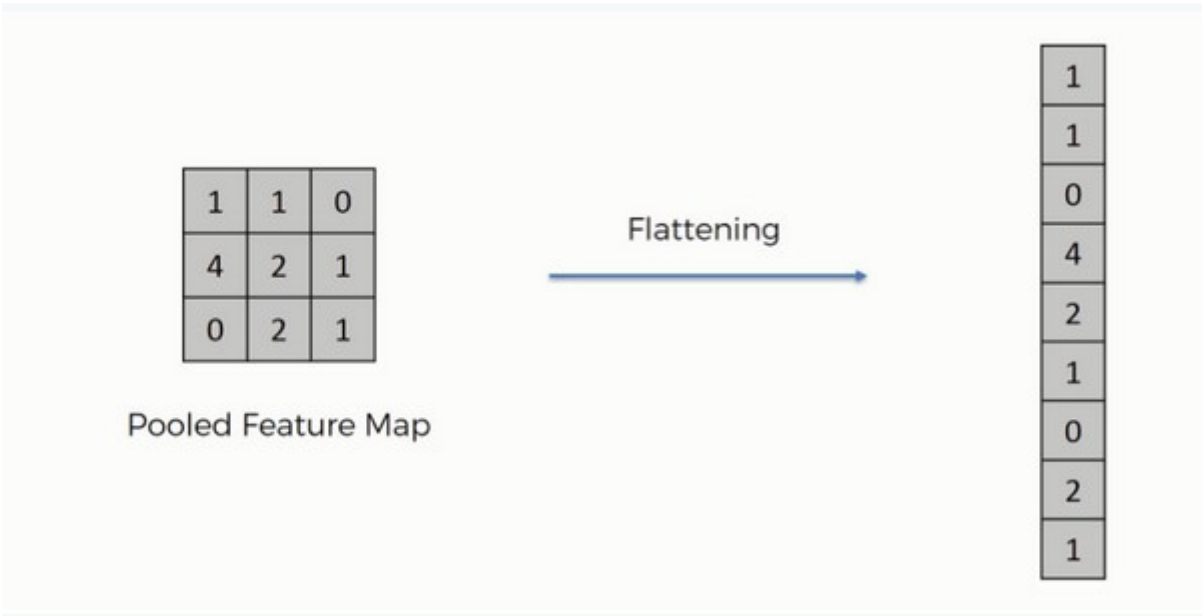
Έως τώρα, το δίκτυό μας δεν εκτελεί κάποια λειτουργία που να έχει σχέση με την κατηγοριοποίηση εικόνων (classification). Αυτό που έχουμε κάνει είναι να εξάγουμε κάποια χαρακτηριστικά από την εικόνα εισόδου και να μειώσουμε τις διαστάσεις της.

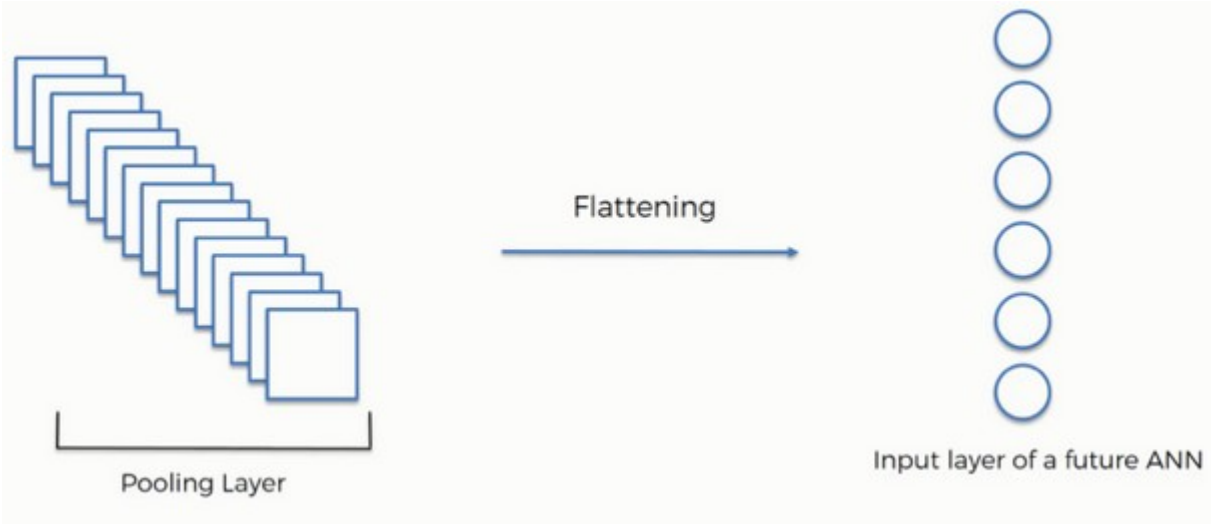


Από εδώ και πέρα ορίζουμε δομές οι οποίες υλοποιούν την διαδικασία του classification. Στην προηγούμενη εικόνα βλέπουμε με κόκκινο περίγραμμα την δομή που είναι υπεύθυνη για την εξαγωγή χαρακτηριστικών από την εικόνα, η οποία όπως εξηγήσαμε αποτελείται από τα convolutional και pooling επίπεδα. Με πράσινο περίγραμμα βλέπουμε την δομή η οποία είναι υπεύθυνη για την διαδικασία του classification. Η συγκεκριμένη δομή αποτελείται από δύο βασικά είδη επιπέδων. Το flatten layer και το dense layer.

- Flatten layer

Το αποτέλεσμα που παίρνουμε από τον feature extraction block είναι ένας ή ένα πλήθος από πίνακες (feature maps). Το classification block όμως στην ουσία αποτελείται από ένα ANN. Όπως λοιπόν είδαμε και στην ερώτηση 1 της εργασίας, ένα ANN δέχεται σαν είσοδο ένα διάνυσμα. Μάλιστα για την χρήση του σε image classification είχαμε μετατρέψει τις εικόνες σε διανύσματα πριν τις τροφοδοτήσουμε στο δίκτυο. Αυτή ακριβώς είναι και η λειτουργία του συγκεκριμένου επιπέδου. Παίρνει τα feature maps των προηγούμενων επιπέδων και τα μετατρέπει σε ένα συνεχόμενο διάνυσμα.





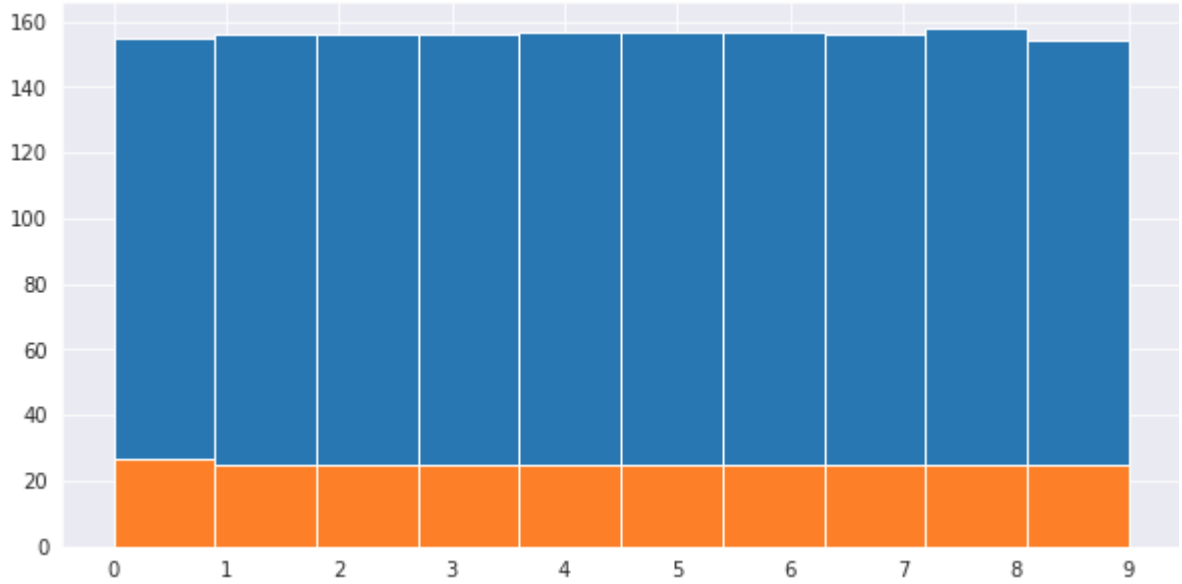
- Dense layer

Στο τέλος ενός CNN βρίσκουμε ένα πλήθος ή ακόμη και ένα μοναδικό dense layer. Αυτό δέχεται σαν είσοδο το διάνυσμα που παράγει το flatten layer και επεξεργάζεται την πληροφορία του όπως ένα απλό ANN, την λειτουργία του οποίου εξετάσαμε στην πρώτη ερώτηση της εργασίας. Τελικά το CNN μας επιστρέφει σαν έξοδο την κλάση στην οποία πιστεύει ότι ανήκει η εικόνα εισόδου.

a) Απλό CNN

Το δίκτυο που χρησιμοποιούμε για αυτό το ερώτημα αποτελείται από δύο συνελλικτικά επίπεδα μεγέθους 64 και 32 νευρώνων αντίστοιχα με μέγεθος παραθύρου 3 x 3 και ένα επίπεδο fully connected μεγέθους 128 νευρώνων.

Πριν προχωρήσουμε στην κατασκευή και εκπαίδευση του δικτύου εισάγουμε τα δύο datasets, original και augmented, και φτιάχνουμε τα dataframes τα οποία θα περιέχουν την διεύθυνση κάθε φωτογραφίας καθώς και την πραγματική της κλάση. Η χρήση των dataframes μας βοηθά στην καλύτερη διαχείριση των train, test και validation datasets σε όλη την πορεία της υλοποίησης.



Ιστόγραμμα train set(μπλε), test set(πορτοκαλί) του original dataset

Για την κατασκευή του δικτύου χρησιμοποιούμε την βιβλιοθήκη Keras, η οποία παρέχει εύκολη πρόσβαση στα βασικά επίπεδα που θα χρησιμοποιήσουμε. Αρχικά ορίζουμε ότι το μοντέλο μας θα είναι Sequential, άρα τα επόμενα επίπεδα θα διαδέχονται το ένα το άλλο. Προσθέτουμε ένα συνελλυκτικό επίπεδο μεγέθους 64 με την παραμετροποίηση που ζητά η εκφώνηση, ενώ επίσης προσθέτουμε padding. Επιλέγουμε ως συνάρτηση ενεργοποίησης την relu. Στην συνέχεια προσθέτουμε ένα max pooling επίπεδο με μέγεθος παραθύρου 2 x 2. Αμέσως μετά προσθέτουμε ένα

ακόμη συνελλικτικό επίπεδο μεγέθους αυτή την φορά 32 φίλτρων, ενώ με την ίδια λογική συμπληρώνουμε με ένα επίπεδο max pooling. Τέλος προσθέτουμε ένα επίπεδου flatten το οποίο ακολουθείται από ένα επίπεδο dense 128 νευρώνων και συνάρτηση ενεργοποίησης την relu. Όμως το δίκτυο δεν είναι ακόμη ολοκληρωμένο καθώς το classification block είναι ελλιπές. Για να ολοκληρώσουμε το δίκτυο προσθέτουμε ένα τελευταίο επίπεδο dense μεγέθους 10 νευρώνων με συνάρτηση ενεργοποίησης την softmax. Αυτό είναι το επίπεδο το οποίο θα μας επιστρέφει μία από τις 10 κλάσεις που έχουμε. Βλέπουμε παρακάτω τον κώδικα ορισμού του δικτύου.

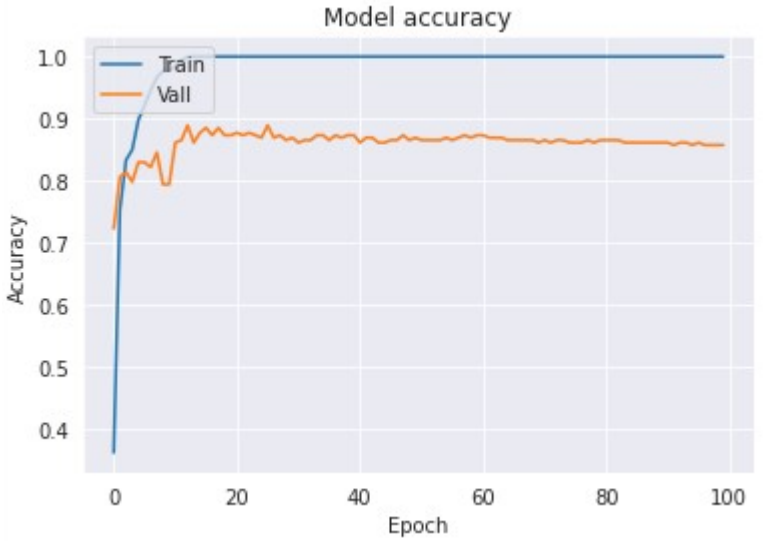
```
model = models.Sequential()

model.add(layers.Conv2D(64, (3,3), input_shape=(100, 100, 3), padding='same', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Conv2D(32, (3,3), input_shape=(100, 100, 3), padding='same', activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

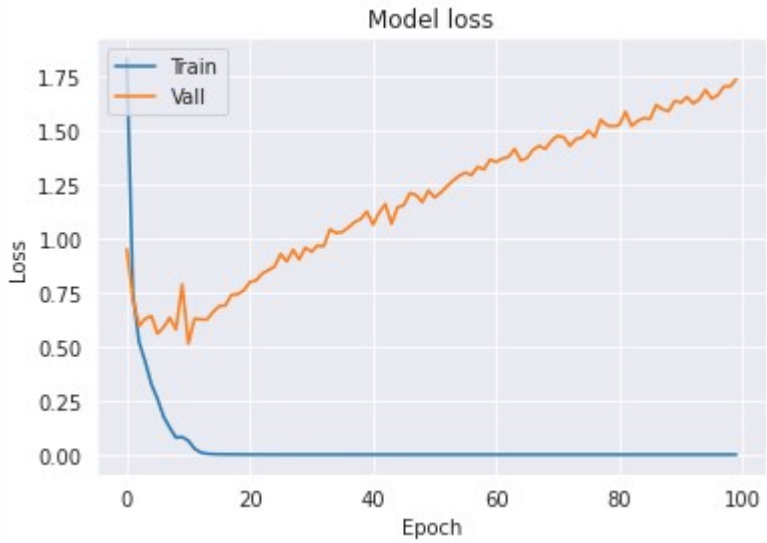
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 64)      1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 32)        18464
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 20000)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2560128
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 2,581,674
Trainable params: 2,581,674
Non-trainable params: 0
_________________________________________________________________
```

Βλέπουμε ότι το δίκτυο αποτελείται συνολικά από 2,581,674 παραμέτρους, εκ των οποίων όλες είναι προς εκπαίδευση.

b) Γραφήματα και overfitting

Προχωράμε στην εκπαίδευσή του δικτύου. Χρησιμοποιούμε την συνάρτηση fit() όπως και για το ANN, όμως αυτή την φορά τροφοδοτούμε τις εικόνες στο δίκτυο με τη βοήθεια του ImageDataGenerator αντικειμένου, με το οποίο δημιουργούμε ένα flow εικόνων χρησιμοποιώντας τις διευθύνσεις των εικόνων από τα dataframes που δημιουργήσαμε στην αρχή. Χρησιμοποιούμε αρχικά το original dataset και παίρνουμε τα εξής αποτελέσματα.



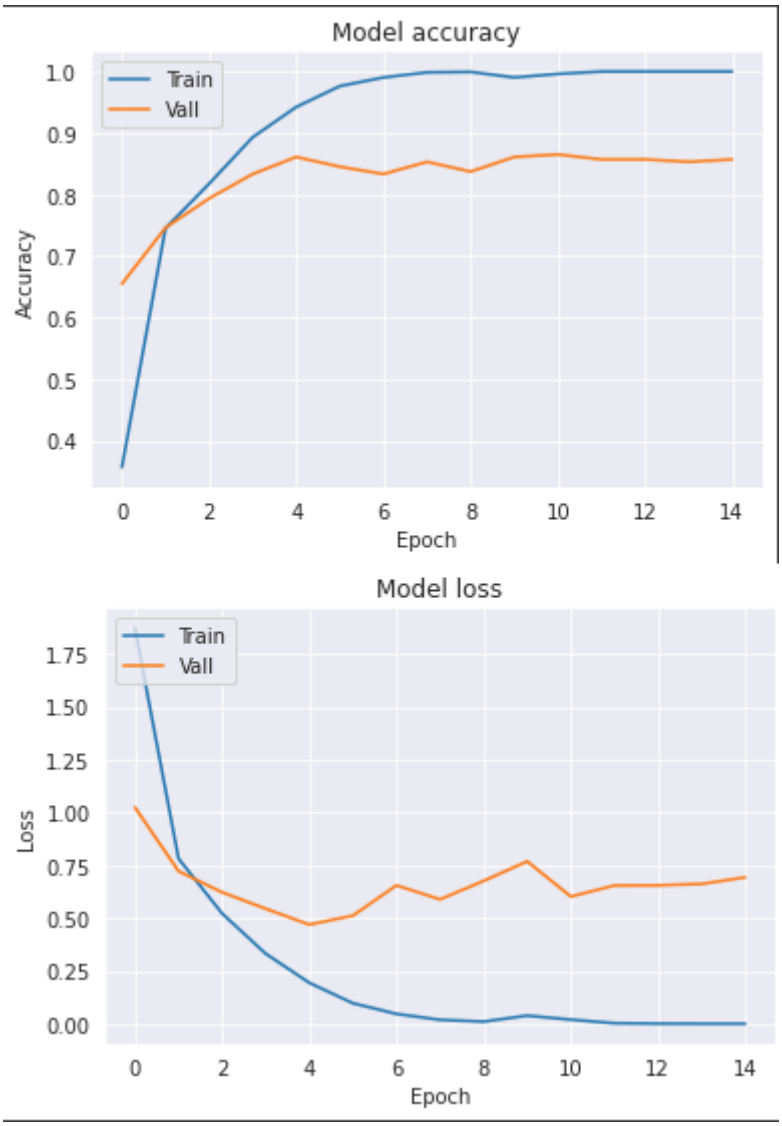


Βλέπουμε ότι πολύ γρήγορα εμφανίζεται το φαινόμενο του overfitting κατά την εκπαίδευση, αφού η τιμή του val_loss αυξάνεται αντί να ακολουθεί μία πορεία κοντά σε αυτήν της τιμής του train_loss. Η ακρίβεια που πετυχαίνει στο test set είναι 86.1% Κοιτώντας το το διάγραμμα με τις τιμές train/validation loss μπορούμε να πούμε ότι 15 εποχές είναι αρκετές για την εκπαίδευση του δικτύου μας.

```
16/16 [==============================] - 1s 34ms/step - loss: 1.6704 - acc: 0.8611
===Testing loss and accuracy===
Test loss:  1.6704331636428833
Test accuracy:  0.8611111044883728
```

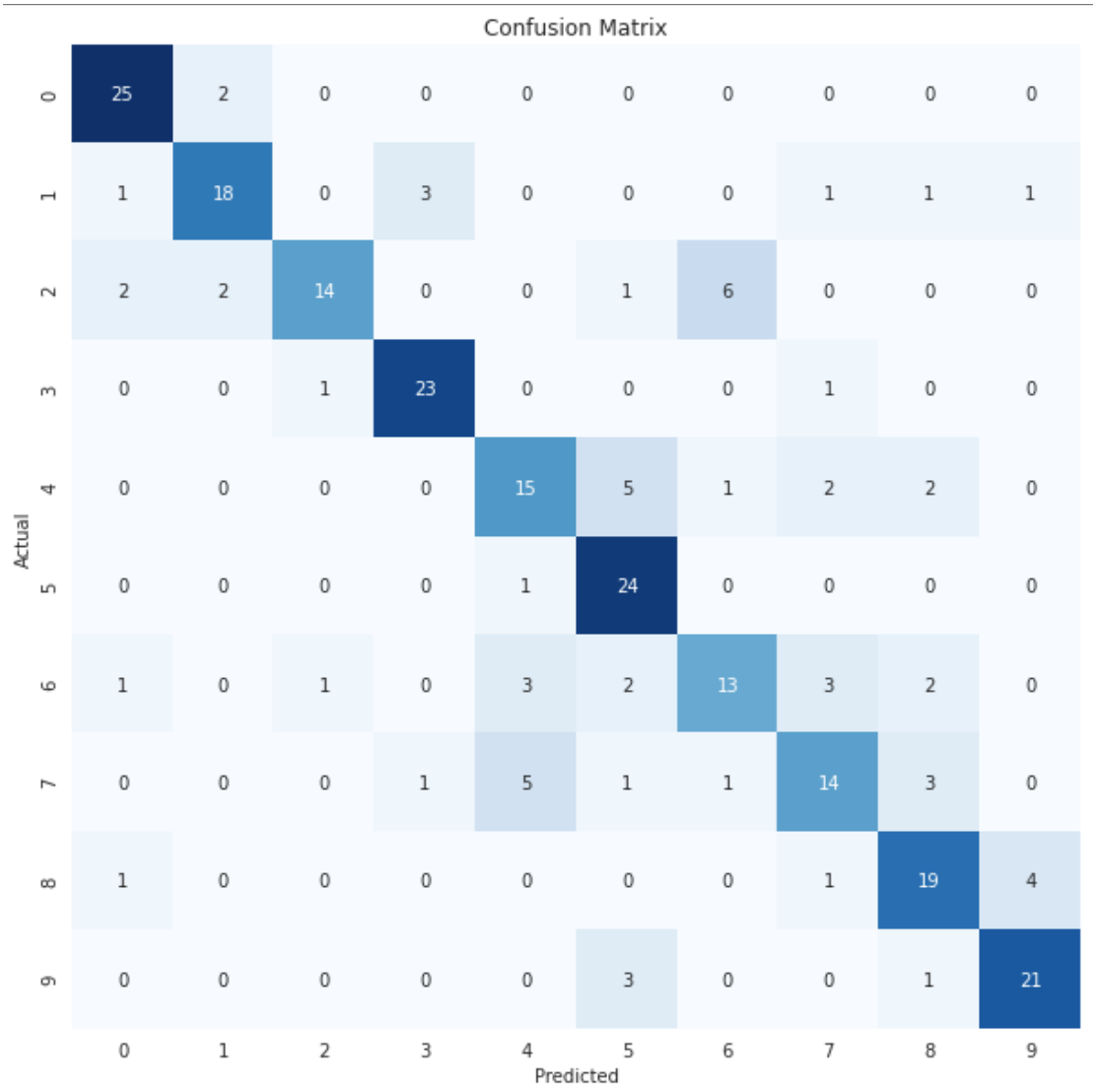
Ας

δοκιμάσουμε να εκπαιδεύσουμε τον δίκτυο μας για 15 αποχές. Τα αποτελέσματα φαίνονται παρακάτω.




```
16/16 [==============================] - 1s 35ms/step - loss: 0.6714 - acc: 0.8889
===Testing loss and accuracy===
Test loss:  0.6714133024215698
Test accuracy:  0.8888888955116272
```

Βλέπουμε ότι για 15 εποχές το δίκτυο αποδίδει καλύτερα με ποσοστό ακριβείας 88.8%. Έχουμε σταματήσει την εκπαίδευση πριν εμφανιστεί το overfitting ή τουλάχιστον πριν γίνει έντονο, γι αυτό το δίκτυο αποδίδει καλύτερα στο test set. Παρακάτω βλέπουμε για το συγκεκριμένο αριθμό εποχών το confusion matrix καθώς και τις ζητούμενες μετρικές. Για την επίλυση του overfitting θα χρησιμοποιήσουμε τις εξής μεθόδους, data augmentation, batch normalization και dropout επίπεδο



	precision	recall	f1-score
0	0.83	0.93	0.88
1	0.82	0.72	0.77
2	0.88	0.56	0.68
3	0.85	0.92	0.88
4	0.62	0.60	0.61
5	0.67	0.96	0.79
6	0.62	0.52	0.57
7	0.64	0.56	0.60
8	0.68	0.76	0.72
9	0.81	0.84	0.82

c) διάφορες δομές CNN

Ας δοκιμάσουμε τώρα διάφορες αρχιτεκτονικές δικτύων. Θα στηριχθούμε στην αρχική αρχιτεκτονική και πάνω σε αυτήν θα κάνουμε διαφορές αλλαγές. Ας δοκιμάσουμε αρχικά να μεταβάλουμε το πλήθος τον νευρώνων του dense επιπέδου, αφήνοντας το υπόλοιπο δίκτυο όπως είναι. Παρακάτω βλέπουμε συνοπτικά όλες τις δοκιμές μαζί με την ακρίβεια που μας δίνουν.

Δοκιμές	Loss	Accuracy
32 νευρώνες στο Dense	1.11	83.3
64 νευρώνες στο Dense	1.67	86%
128 νευρώνες στο Dense	1.09	87.5%
256 νευρώνες στο Dense	1.13	87%

Μπορούμε να παρατηρήσουμε ότι καθώς αυξάνουμε το πλήθος των νευρώνων στο επίπεδο Dense έχουμε καλύτερη απόδοση του δικτύου. Ωστόσο για 256 νευρώνες φαίνεται να έχουμε μία πτώση της απόδοσης. Οι 128 νευρώνες στο Dense επίπεδο φαίνεται ότι είναι αρκετοί για να πετύχουμε μία βελτίωση. Τώρα μπορούμε να δοκιμάσουμε να προσθέσουμε ένα ακόμη επίπεδο Dense. Επιλέγουμε την περίπτωση όπου έχουμε δύο διαδοχικά επίπεδα Dense 256 και 128 νευρώνων αντίστοιχα.

2 x Dense με 256, 128	0.97	88.3%

Βλέπουμε ότι η απόδοση αυτής της αρχιτεκτονικής είναι ελαφρός καλύτερη από την περίπτωση που χρησιμοποιούμε μόνο ένα Dense επίπεδο.

Ας εξετάσουμε τώρα τι γίνεται όταν μεταβάλουμε το πλήθος των φίλτρων στα συνελλικτικά επίπεδα, μεταβάλουμε το πλήθος των φίλτρων μόνο στο πρώτο συνελλικτικό επίπεδο. Παρακάτω βλέπουμε συνοπτικά όλες τις δοκιμές μαζί με την ακρίβεια που μας δίνουν.

Δοκιμές	Loss	Accuracy
16 νευρώνες στο Conv2D	1.89	81.2%
32 νευρώνες στο Conv2D	1.8	82.1%
64 νευρώνες στο Conv2D	1.67	86%
128 νευρώνες στο Conv2D	0.89	87%

Παρατηρούμε το εξής, καθώς το πλήθος των φίλτρων αυξάνεται στο πρώτο συνελλικτικό επίπεδο το δίκτυο αποδίδει καλύτερα. Καθώς αυξάνονται τα φίλτρα που χρησιμοποιούμε, αυξάνονται και τα χαρακτηριστικά τα οποία εξάγονται από την εικόνα, άρα η συμπεριφορά είναι αναμενόμενη. Ας δοκιμάσουμε τώρα να προσθέσουμε ένα τρίτο συνελλικτικό επίπεδο. Εξετάζουμε μία αρχιτεκτονική με τρία συνελλικτικά επίπεδα τα οποία έχουν πλήθος φίλτρων 64, 32, 16 αντίστοιχα, το αποτέλεσμα που παίρνουμε είναι το εξής.

3 x Conv2D με 64,32,16	0.67	90.5%

Η βελτίωση που φέρνει αυτή η αρχιτεκτονική είναι μεγάλη. Προσθέτοντας ένα ακόμη συνελλικτικό επίπεδο, δίνουμε την δυνατότητα στο δίκτυο να εξάγει πιο πολύπλοκα χαρακτηριστικά από την εικόνα.

Τέλος προσπαθούμε να φτιάξουμε μία αρχιτεκτονική η οποία θα προσφέρει την καλύτερη απόδοση. Λαμβάνοντας υπ' όψιν τις παρατηρήσεις που κάναμε προηγουμένως, αλλά και μετά από δοκιμές καταλήγουμε στην εξής αρχιτεκτονική.

```
model_best = models.Sequential()

model_best.add(layers.Conv2D(64, (3,3), input_shape=(100, 100,1), padding='same', activation='relu'))
model_best.add(layers.BatchNormalization(momentum=0.1))
model_best.add(layers.MaxPooling2D(pool_size=(2,2)))
model_best.add(layers.Conv2D(16, (3,3), padding='same', activation='relu'))
model_best.add(layers.BatchNormalization(momentum=0.1))
model_best.add(layers.MaxPooling2D(pool_size=(2,2)))
model_best.add(layers.Conv2D(16, (3,3), padding='same', activation='relu'))
model_best.add(layers.BatchNormalization(momentum=0.1))
model_best.add(layers.MaxPooling2D(pool_size=(2,2)))
model_best.add(layers.Flatten())
model_best.add(layers.Dense(128, activation='relu'))
model_best.add(layers.Dropout(0.2))
model_best.add(layers.Dense(10, activation='softmax'))

model_best.summary()
```

Όπως βλέπουμε, αυτή η αρχιτεκτονική χρησιμοποιεί και επίπεδα τα οποία υλοποιούν batch normalization. Αυτού του είδους τα επίπεδα θα τα αναλύσουμε στην αμέσως επόμενη ενότητα, οπότε είναι ίσος καλύτερα να εξηγήσουμε λεπτομερώς την βέλτιστη αρχιτεκτονική στο τέλος.

d) Χρήση batch normalization

Χρησιμοποιώντας batch normalization επίπεδα, η φάση του training γίνεται αποδοτικότερα και από άποψη χρόνου και από άποψη ποιότητας. Επίσης μπορεί να χρησιμοποιηθεί ως μέθοδος regularzation ώστε να αποφεύγετε το Overfitting. Συγκεκριμένα για την υλοποίησή μας χρησιμοποιούμε το επίπεδο BatchNormalization το οποίο προσφέρεται από την βιβλιοθήκη Keras. Προσθέτουμε το BatchNormalization επίπεδο στο αρχικό δίκτυο ως εξής.

```
model = models.Sequential()

model.add(layers.Conv2D(64, (3,3), input_shape=(100, 100, 3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Conv2D(32, (3,3), input_shape=(100, 100, 3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.summary()
```

Εφαρμόζουμε batch normalization στην έξοδο τον συνελλικτικών επιπέδων, έτσι οι τιμές που αποθηκεύονται στα feature maps είναι κανονικοποιημένες. Βλέπουμε ότι παίρνουμε ακρίβεια 89.6%, μία μεγάλη βελτίωση δηλαδή από το αρχικό CNN.

```
16/16 [==============================] - 1s 35ms/step - loss: 0.5223 - acc: 0.8968
===Testing loss and accuracy===
Test loss:  0.5222882628440857
Test accuracy:  0.89682537317276
```

Επίσης πρέπει να παρατηρήσουμε το εξής. Εμφανίζεται ένας μικρός αριθμός Non-trainable parameters. Αυτή η μείωση οφείλεται στο γεγονός ότι λόγο του batch normalization οι νευρώνες bias απαλείφονται από τα επίπεδα στα οποία εφαρμόζεται η μέθοδος.

```
=================================================================
Total params: 2,582,058
Trainable params: 2,581,866
Non-trainable params: 192
_________________________________________________________________
```

e) Επίπεδα Dense και MaxPooling

Όπως είδαμε και σε προηγούμενη ενότητα, τα fully contented επίπεδα σχηματίζουν τον classifier του δικτύου. Χωρίς αυτά το δίκτυο δεν μπορεί να δουλέψει, ενώ σε επίπεδο κώδικα δεν μπορεί να κάνει compile. Για να λειτουργήσει το δίκτυο είναι απαραίτητο να έχουμε ένα επίπεδο flatten και τουλάχιστον ένα επίπεδο dense. Αυτό που μπορούμε να δοκιμάσουμε είναι να παραλείψουμε το dense επίπεδο 128 νευρώνων και να κρατήσουμε το flatten και το dense 10 νευρώνων. Η ακρίβεια του δικτύου όμως είναι πολύ χαμηλή έτσι.

Τα επίπεδα max pooling δεν είναι απαραίτητο να υπάρχουν, όσο αφορά τη λειτουργικότητα του δικτύου. Αυτό που παρατηρούμε είναι ότι το πλήθος των Total parameters έχει αυξηθεί. Αυτό είναι λογικό αφού τα max pooling επίπεδα συνέβαλαν σε σημαντική μείωση διαστάσεων των feature maps, άρα και των συνδέσεων μεταξύ νευρώνων.

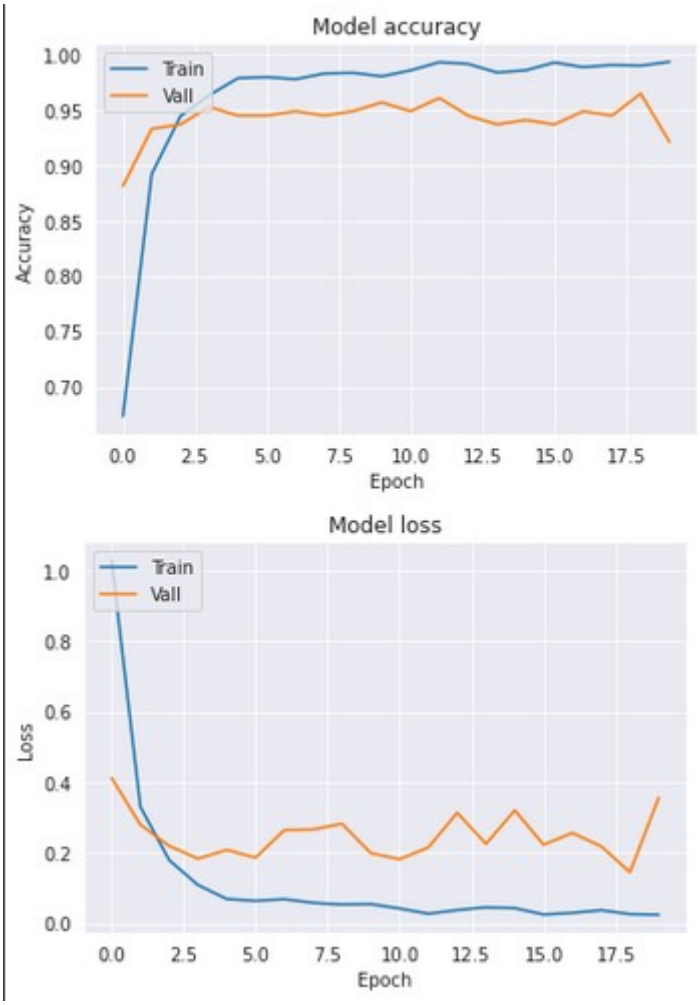
Βέλτιστη αρχιτεκτονική

Η βέλτιστη αρχιτεκτονική αναφέρθηκε και στο ερώτημα 2.c. Όμως περιμέναμε να εξηγήσουμε πρώτα το επίπεδο batch normalization πριν την περιγράψουμε. Η καλύτερη αρχιτεκτονική προκύπτει από τα συμπεράσματα που βγάλαμε στις προηγούμενες ενότητες του ερωτήματος 2, καθώς και μετά από δοκιμές.

```
model_best = models.Sequential()

model_best.add(layers.Conv2D(64, (3,3), input_shape=(100, 100,1), padding='same', activation='relu'))
model_best.add(layers.BatchNormalization(momentum=0.1))
model_best.add(layers.MaxPooling2D(pool_size=(2,2)))
model_best.add(layers.Conv2D(32, (3,3), padding='same', activation='relu'))
model_best.add(layers.BatchNormalization(momentum=0.1))
model_best.add(layers.MaxPooling2D(pool_size=(2,2)))
model_best.add(layers.Conv2D(16, (3,3), padding='same', activation='relu'))
model_best.add(layers.BatchNormalization(momentum=0.1))
model_best.add(layers.MaxPooling2D(pool_size=(2,2)))
model_best.add(layers.Flatten())
model_best.add(layers.Dense(128, activation='relu'))
model_best.add(layers.Dropout(0.2))
model_best.add(layers.Dense(10, activation='softmax'))

model_best.summary()
```

Έχουμε τρία συνελλικτικά επίπεδα μεγέθους 64, 32 και 16 φίλτρων αντίστοιχα. Μεταξύ των συνελλικτικών κρυφών επιπέδων τοποθετούμε batch normalization επίπεδα με momentum = 0.1. Η παράμετρος momentum ορίζει την ένταση με την οποία εφαρμόζεται η μέθοδος batch normalization. Μετά από κάθε επίπεδο batch normalization τοποθετούμε ένα επίπεδο max pooling μεγέθους παραθύρου 2 x 2. Το

classification block αποτελείται από ένα επίπεδο flatten, ένα κρυφό επίπεδο dense μεγέθους 128 νευρώνων, ένα επίπεδο dropout με ποσοστό απόρριψης 0.2 και ένα επίπεδο dense μεγέθους 10 νευρώνων με συνάρτηση ενεργοποίησης την softmax. Το επίπεδο dropout συμβάλει περαιτέρω στην αποφυγή του overfitting. Για την εκπαίδευση του δικτύου χρησιμοποιούμε το επαυξημένο dataset, ενώ εφαρμόζουμε dimentionality reduction διαβάζοντας τις εικόνας σε ασπρόμαυρη μορφή. Το δίκτυο μας δίνει ακρίβεια 94%.




```
16/16 [==============================] - 2s 111ms/step - loss: 0.4789 - acc: 0.9405
===Testing loss and accuracy===
Test loss:  0.47893190383911133
Test accuracy:  0.9404761791229248
```

Ερωτήσεις κατανόησης

a)

Βασικό δομικό στοιχείο των CNN είναι τα συνελλικτικά επίπεδα. Αυτά όπως είπαμε έχουν την ικανότητα εξαγωγής χαρακτηριστικών από τις εικόνες. Επίσης το πλεονέκτημα των CNN έναντι των ANN έχει να κάνει και με τον τρόπο που επεξεργάζονται μία εικόνα. Τα ANN όπως είπαμε δέχονται την εικόνα σε μορφή διανύσματος, η οποία όμως δεν είναι η φυσική μορφή μίας εικόνας. Τα CNN από την άλλη δέχονται κάθε εικόνα με μορφή πίνακα. Άρα τα χαρακτηριστικά που εντοπίζουν έχουν πλήρη συσχέτιση με την μορφή της εικόνας. Επίσης πολύ σημαντικό είναι ότι τα CNN μέσω των επιπέδων pooling μειώνουν σημαντικά τις διαστάσεις των δεδομένων που επεξεργάζονται. Αυτό μειώνει τον υπολογιστικό φόρτο που απαιτείτε κατά την εκπαίδευση, επιταχύνοντας την διαδικασία. Επίσης αυτό συμβάλει στο να επικεντρώνεται το δίκτυο μόνο στα χαρακτηριστικά τα οποία θεωρεί σημαντικά.

b)

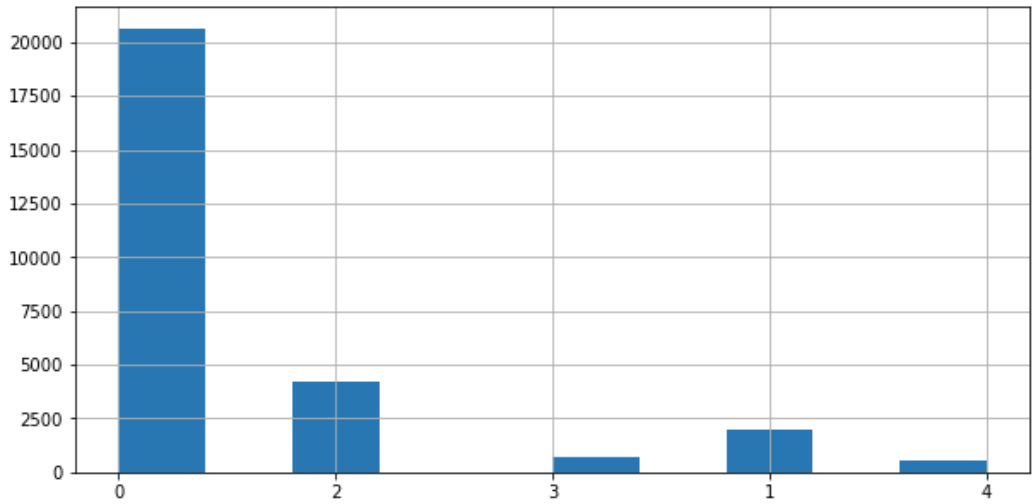
Η πιο σημαντική διαφορά τον CNN έναντι των ANN είναι ότι τα πρώτα χρησιμοποιούν φίλτρα για την αναγνώριση των εικόνων. Τα ANN λειτουργούν με κλασικούς νευρώνες. Κάθε νευρώνας του πρώτου επιπέδου επικεντρώνεται σε ένα εικονοστοιχείο. Αντίθετα το πρώτο επίπεδο ενός CNN σαρώνει την εικόνα κατά περιοχές, εξάγοντας από κάθε μία από αυτές τα σημαντικά χαρακτηριστικά τους. Στην συνέχεια το επίπεδο pooling επικεντρώνεται σε ακόμη πιο σημαντικά χαρακτηριστικά. Καθώς προσθέτουμε επίπεδα σε ένα CNN αυτό φτάνει να αναγνωρίζει ολόκληρες δομές από κάθε εικόνα. Αυτές οι δομές είναι τα χαρακτηριστικά που χρησιμοποιεί για την ταξινόμηση. Ένα ANN δεν αναγνωρίζει δομές της εικόνας ως χαρακτηριστικά, επικεντρόνεται αποκλειστικά στα εικονοστοιχεία της εικόνας. Αυτή η διαφορά μεταξύ των CNN και ANN είναι που κάνει τα πρώτα πολύ πιο αποδοτικά σε εργασίες ταξινόμησης εικόνων. Επίσης μπορούμε να αναφέρουμε ως ουσιαστική διαφορά μεταξύ των δύο δικτύων το εξής. Ένα CNN αποτελείται από δύο επιμέρους bock με το καθένα να επιτελεί μία διαφορετική δουλειά. Ένα block αφιερώνεται στην εξαγωγή των χαρακτηριστικών της εικόνας, ενώ ένα δεύτερο αφοσιώνεται στην διαδικασία της ταξινόμησης. Από ένα ANN δίκτυο λείπει τελείως το πρώτο block και γίνεται απευθείας ταξινόμησης της εικόνας σύμφωνα με τους νευρώνες που ενεργοποιεί κάθε εικονοστοιχείο ξεχωριστά.

Ερωτήσεις bonus

Το dataset με το οποίο δουλεύουμε σε αυτή την ενότητα είναι το Diabetic retinopathy dataset. Θα ξεκινήσουμε αναλύοντας κάποια χαρακτηριστικά του dataset και στην συνέχεια θα περάσουμε στην δοκιμή των ζητούμενων αρχιτεκτονικών.

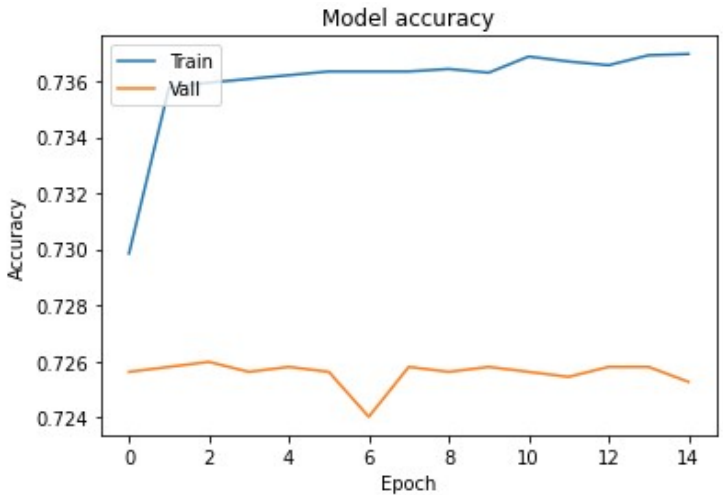
a)

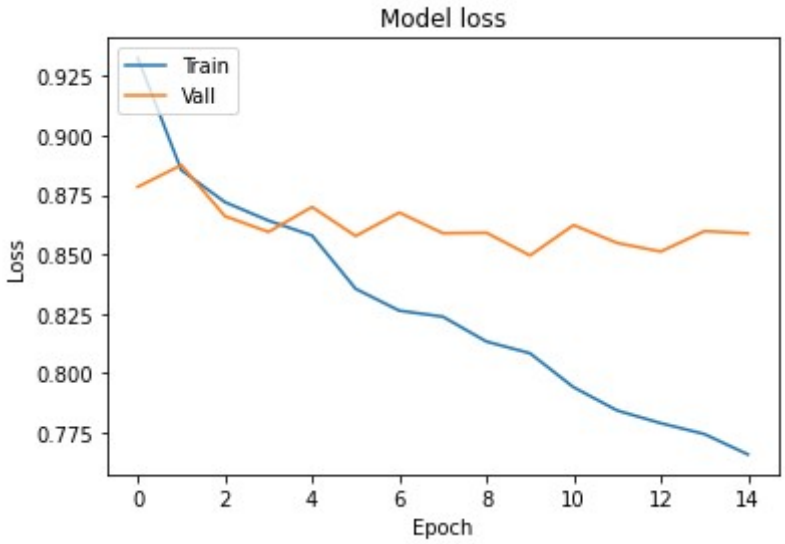
Κατασκευάζουμε 3 dataframes, train, test και validation set. Κάθε ένα περιέχει τις διευθύνσεις των εικόνων καθώς και την κλάση τους.



Βλέπουμε ότι το dataset δεν είναι ισορροπημένο, αφού περιέχει πάρα πολλές εικόνες από την κλάση 0. Οι υπόλοιπες κλάσεις επίσης έχουν δυσανάλογο πλήθος εικόνων..Θα δοκιμάσουμε πρώτα να το χρησιμοποιήσουμε ως έχει ενώ σε δεύτερη φάση θα το εξισορροπήσουμε. Επίσης οι εικόνες έχουν μεγάλο μέγεθος 1024 x 681

Δοκιμάζουμε αρχικά την καλύτερη αρχιτεκτονική από το ερώτημα 2.c. Τροφοδοτούμε το δίκτυο με τη βοήθεια του ImageDatagenerator κάνοντας αποκοπεί τις εικόνες σε μέγεθος 300 x 300. επίσης κανονικοποιούμε τις εικόνες στο διάστημα [0, 1]. Το αποτέλεσμα είναι το εξής.

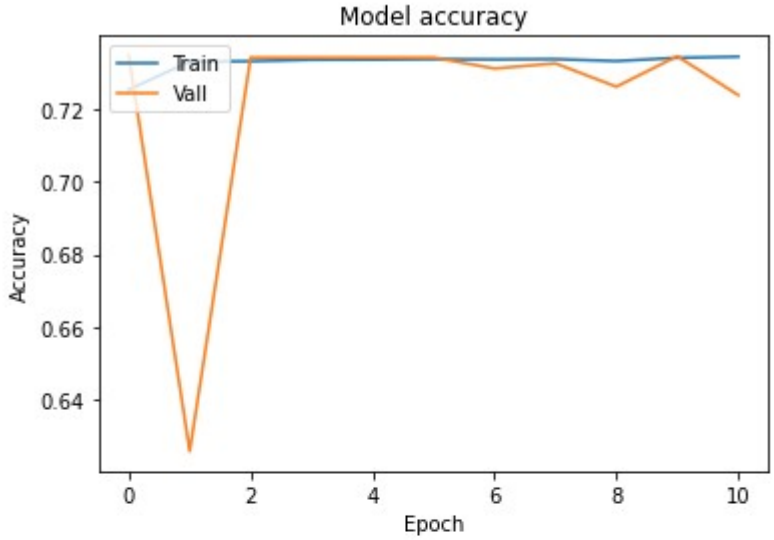


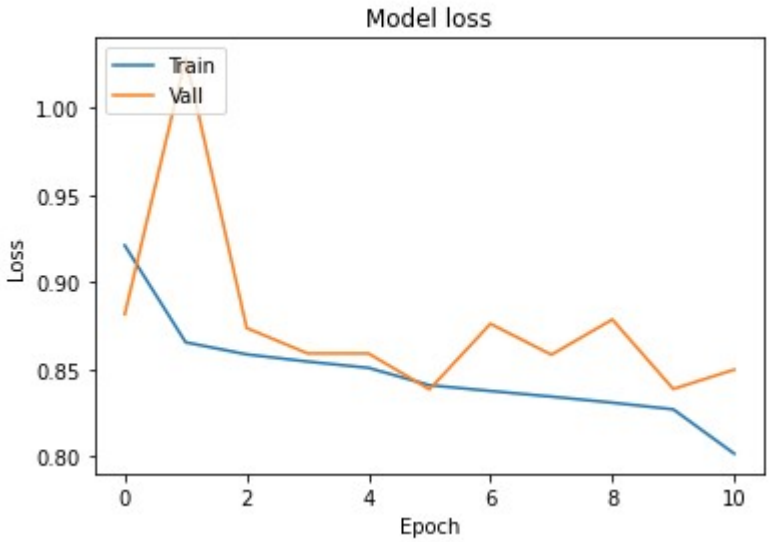



```
320/320 [==============================] - 120s 375ms/step - loss: 0.8294 - acc: 0.7367
===Testing loss and accuracy===
Test loss:  0.8294113874435425
Test accuracy:  0.7366923093795776
```

Το δίκτυο μας δίνει ακρίβεια 73.6%

Δοκιμάζουμε τώρα την αρχιτεκτονική ResNet-50. Δεν χρησιμοποιούμε κάποιο pre-trained μοντέλο, αλλά φτιάχνουμε ένα δίκτυο της αρχιτεκτονικής ResNet-50 το οποίο εκπαιδεύουμε από την αρχή. Εισάγουμε την συγκεκριμένη αρχιτεκτονική έτοιμη από την βιβλιοθήκη Keras και φτιάχνουμε το δίκτυο με τις παραμέτρους που θέλουμε. Το εκπαιδεύουμε κάνοντας χρήση early stopping και learning rate scheduler. Το αποτέλεσμα είναι το εξής.






```
320/320 [==============================] - 161s 503ms/step - loss: 0.8507 - accuracy: 0.7327
===Testing loss and accuracy===
Test loss:  0.8506568074226379
Test accuracy:  0.7327070832252502
```

Το δίκτυο μας δίνει ακρίβεια 73.2%

Τώρα δοκιμάζουμε να χρησιμοποιήσουμε μία αρχιτεκτονική ResNet-50 με transfer learning. Με τον όρο transfer learning εννοούμε την χρήση ενός μοντέλου το οποίο έχει εκπαιδευτεί. Χρησιμοποιούμε τα βάρη του imagenet. Αυτά είναι στην ουσία τα βάρη που απέκτησε ένα δίκτυο αρχιτεκτονικής ResNet-50 το οποίο εκπαιδεύτηκε στο imagenet dataset. Αρχικά ορίζουμε ένα προεκαπιδευμένο μοντέλο ResNet-50. Στην συνέχεια “παγώνουμε” όλα τα επίπεδά του εκτός από αυτά του τελευταίου block. Με τον όρο παγώνουμε εννοούμε ότι καθιστάμε αυτά τ επίπεδα μη εκπαιδεύσημα, θα διατηρήσουν δηλαδή τα βάρη τους και δεν θα μεταβληθούν από την διαδικασία της εκπαίδευσης. Το δίκτυο θα πρέπει να εκτελεί εργασία ταξινόμησης εικόνων μεταξύ 5 κλάσεων. Γι αυτόν τον λόγο προσθέτουμε στο τέλος του μοντέλου αρχιτεκτονικής ResNet-50 ένα classification block. Το block αυτό αποτελείται από ένα επίπεδο flatten και ένα επίπεδο dense 5 νευρώνων. Άρα τα επίπεδα τα οποία θα εκπαιδευτούν είναι αυτά του τελευταίου block της αρχιτεκτονικής ResNet-50 και αυτά του classification block.

```
model_transfer = Sequential()
model_transfer.add(resNet50)
model_transfer.add(layers.Flatten())
model_transfer.add(layers.Dense(5, activation='softmax'))

model_transfer.summary()
```

Το δίκτυο δίνει μία ακρίβεια 73.6%.

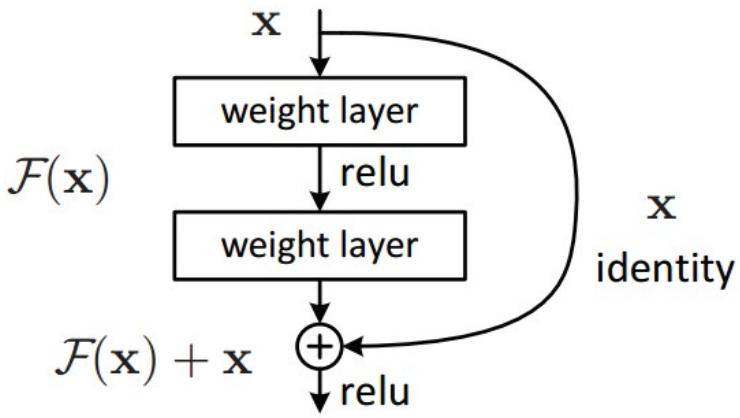
```
320/320 [==============================] - 139s 436ms/step - loss: 0.8258 - accuracy: 0.7370
===Testing loss and accuracy===
Test loss:  0.8257951140403748
Test accuracy:  0.7369769215583801
```

Συνολικά βλέπουμε ότι και οι τρεις αρχιτεκτονικές έχουν την ίδια απόδοση. Όμως παρουσιάζουν διαφορές στο πλήθος των αποχών που χρειάστηκε ώστε να μας δώσουν την εκάστοτε απόδοση. Το CNN χρειάστηκε το μικρότερο πλήθος εποχών, δηλαδή 15 εποχές. Το δίκτυο με την αρχιτεκτονική ResNet-50 χωρίς transfer learning χρειάστηκε 22 εποχές, ενώ το δίκτυο με την αρχιτεκτονική ResNet-50 με χρήση transfer learning χρειάστηκε 42 εποχές. Αυτή η διαβάθμιση είναι λογική. Το δεύτερο δίκτυο είναι πολύ πιο μεγάλος από το πρώτο. Ενώ το τρίτο δίκτυο μπορεί να περιέχει βάρη τα οποία εντέλει δεν είναι χρήσιμα, βάρη τα οποία ανταποκρίνονται σε χαρακτηριστικά που δεν εμφανίζονται στις εικόνες του dataset μας. Οπότε τα επίπεδα που μπορούν να ανανεώσουν τα βάρη τους χρειάζονται αρκετές εποχές ώστε να αντισταθμίσουν τα βάρη αυτά.

Αν δοκιμάσουμε να ισορροπήσουμε το dataset τότε δεν παίρνουμε καλά αποτελέσματα. Και για τις τρεις αρχιτεκτονικές η ακρίβεια που δίνουν είναι κάτω του 40%. Αυτό δείχνει ότι ίσως η κλάση 0 η οποία διαθέτει πολλές περισσότερες φωτογραφίες είναι “δύσκολη” στην εκμάθησής της και άρα χρειάζεται αυτές τις περισσότερες φωτογραφίες.

b)

Η αρχιτεκτονική ResNet χρησιμοποιεί έναν ιδιαίτερο τύπο συνδέσεων, τα shortcut connections. Το βασικό δομικό στοιχείο της αρχιτεκτονικής αυτής είναι το residual block.



Αντίθετα το CNN αποτελείται από ένα πλήθος επιπέδων τα οποία βρίσκονται το ένα πίσω από το άλλο.

c)

Το πρόβλημα που αποφεύγεται κατά την εκπαίδευση βαθέων νευρωνικών δικτύων με την αρχιτεκτονική ResNet είναι το φαινόμενο του vanishing gradient.

Αναφορές

[1] Keras at Github:
https://github.com/keras-team/keras/blob/master/keras/layers/recurrent.py#L2081

[2] How to fight underfitting in a deep neural net:
https://datascience.stackexchange.com/questions/731/how-to-fight-underfitting-in-a-deep-neural-net

[3] Keras model.summary() result- Understanding the # of Parameters:
https://stackoverflow.com/questions/36946671/keras-model-summary-result-understanding-the-of-parameters

[4] Learnable Parameters ("Trainable Params") In A Keras Convolutional Neural Network:
https://deeplizard.com/learn/video/8d-9SnGt5E0

[5] Artificial Neural Networks (Deep Learning):
https://medium.com/@jorgesleonel/artificial-neural-networks-deep-learning-21201054ca29

[6] Understanding of Convolutional Neural Network (CNN) — Deep Learning:
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[7] Everything You Should Know About Dropouts And BatchNormalization In CNN:
https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/

[8] Batch Normalization in Convolutional Neural Networks:
https://www.baeldung.com/cs/batch-normalization-cnn

[9] About Train, Validation and Test Sets in Machine Learning:
https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[10] What is the Difference Between Test and Validation Datasets?:
https://machinelearningmastery.com/difference-test-validation-datasets/

[11] An Overview of Overfitting and its Solutions: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf#:~:text=To%20reduce%20the%20effects%20of,set%3B%203)%20“data-

[12] Why ResNets Are A Major Breakthrough In Image Processing: https://analyticsindiamag.com/why-resnets-are-a-major-breakthrough-in-image-processing/