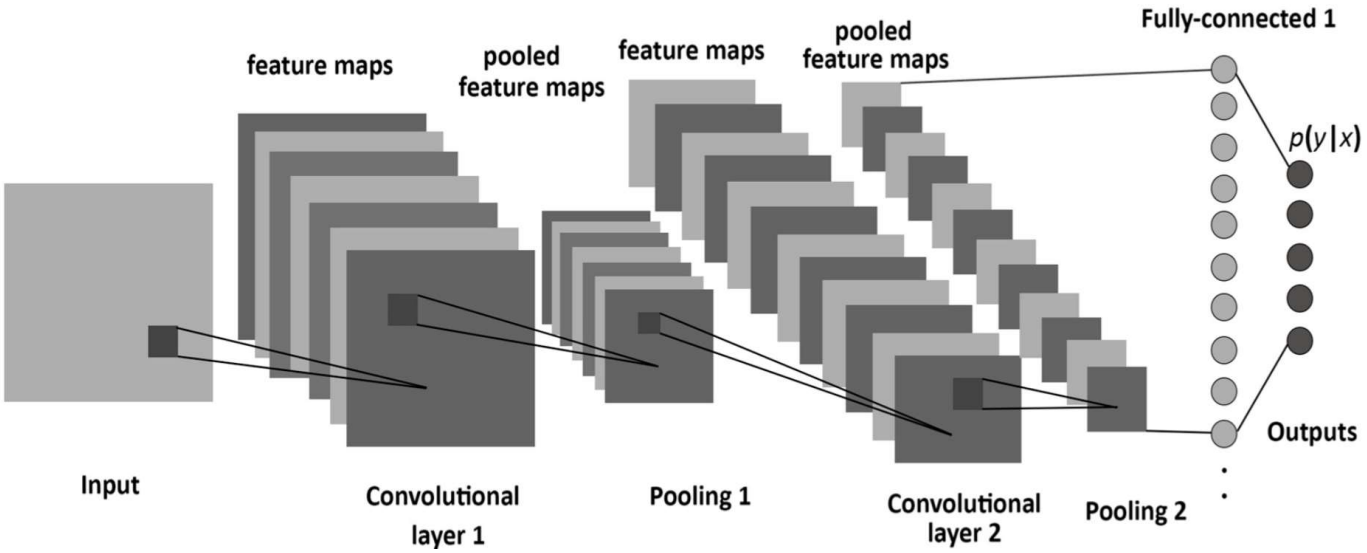
Αναγνώριση Προτύπων 2021-2022

Τελική Εργασία , Παπαδόπουλος Αριστείδης

A.M.:57576



Εικόνα 1 Τυπική δομή Συνελικτικού Νευρωνικού Δικτύου

Τα Dataset

Στην εργασία χρησιμοποιούνται τρία σετ δεδομένων:

- *Fashion_MNIST:*

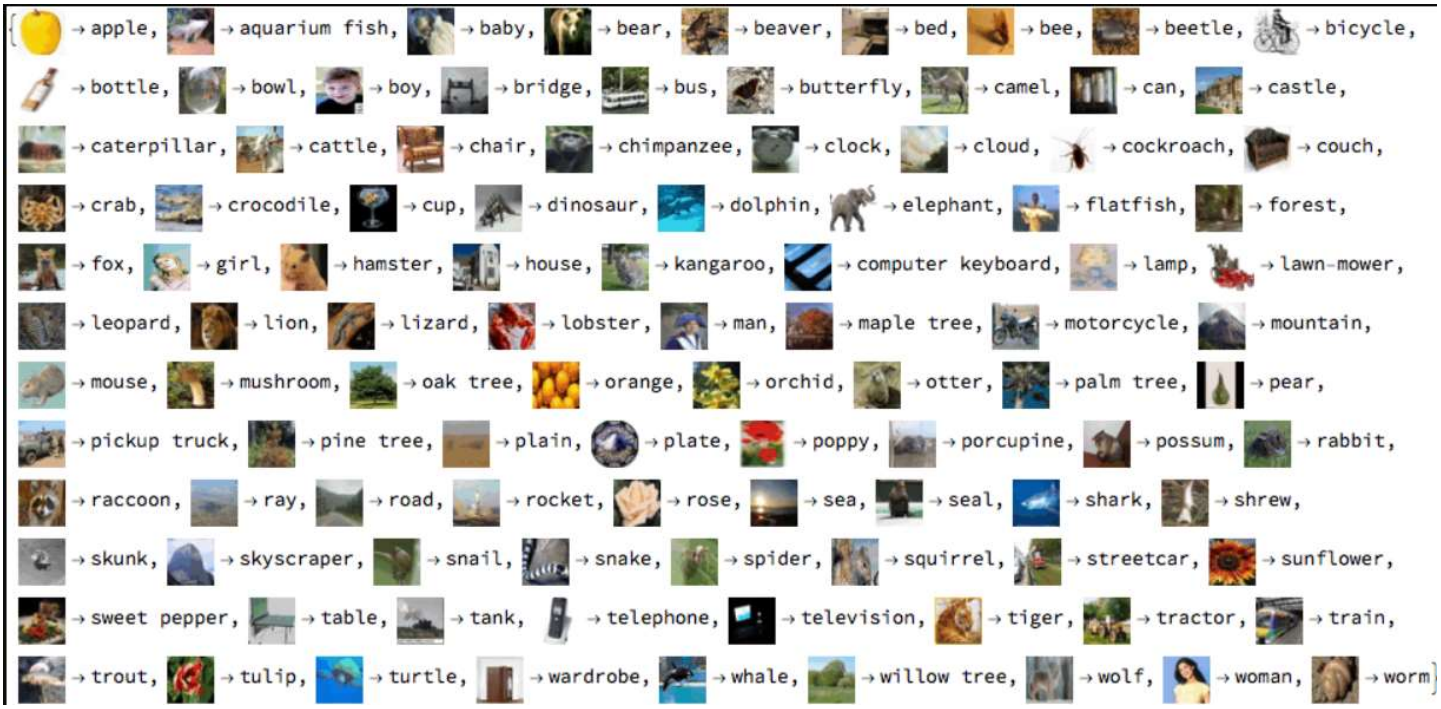
Το συγκεκριμένο dataset περιλαμβάνει 70.000 εικόνες διαστάσεων 28x28 σε grayscale μορφή (μόνο με αποχρώσεις του γκρι) από 10 διαφορετικά είδη ρούχων-υπόδησης (T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle Boot), οι οποίες χωρίζονται σε 60.000 εικόνες εκπαίδευσης και σε 10.000 εικόνες testing. Ένα ενδεικτικό απόσπασμα των εικόνων φαίνεται παρακάτω:

- *Cifar-10:*

Το dataset Cifar-10 περιέχει 60.000 εικόνες, διαστάσεων 32x32 σε RGB μορφή, και πάλι από 10 διαφορετικές ομάδες (αεροπλάνο, αυτοκίνητο, πουλί, γάτα, ελάφι, σκυλί, βάτραχος, άλογο, πλοίο, φορτηγό). Χωρίζονται σε 50.000 εικόνες εκπαίδευσης και σε 10.000 testing, με τις εικόνες να είναι της μορφής:

- *Cifar-100:*

Το dataset Cifar-100 περιέχει 60.000 εικόνες, διαστάσεων 32x32 σε RGB μορφή, και πάλι από 100 διαφορετικές ομάδες οι οποίες ενώνονται σε 20 υπερ-ομάδες. Χωρίζονται σε 50.000 εικόνες εκπαίδευσης (500 για κάθε ομάδα) και σε 10.000 testing (100 για κάθε ομάδα), με τις εικόνες να είναι της μορφής:



Προεπεξεργασία δεδομένων

Η προεπεξεργασία των δεδομένων είναι μια διαδεδομένη τεχνική με σκοπό τον μετασχηματισμό των εικόνων, ώστε να επιτευχθεί όσο το δυνατόν γίνεται καλύτερο τελικό αποτέλεσμα. Οι παρακάτω διεργασίες, υλοποιούνται στα δεδομένα και των τριών dataset γι' αυτό και αναφέρονται σε αυτό το σημείο.

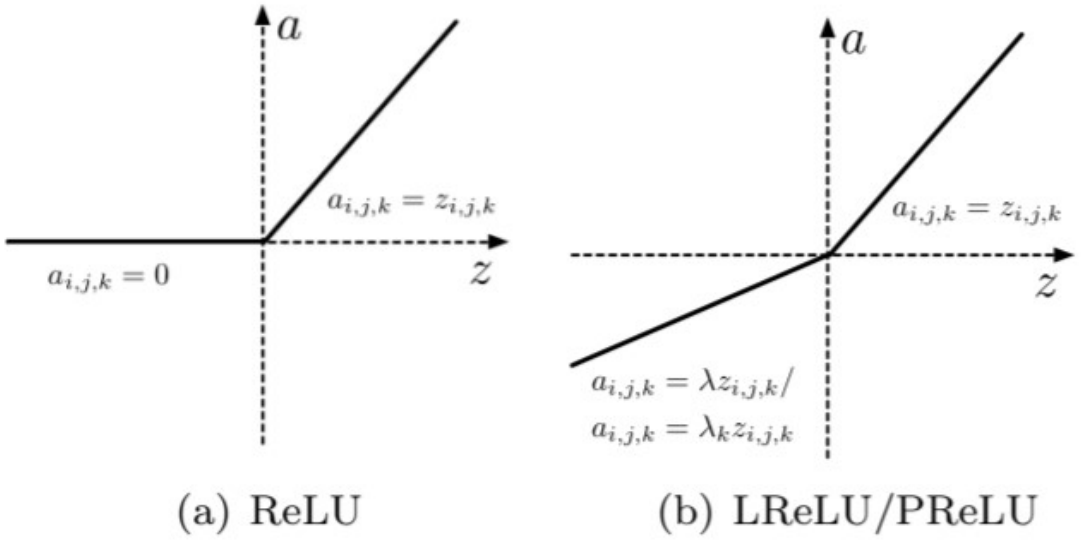
- **Μετασχηματισμός διαστάσεων των εικόνων:**
  Στις περιπτώσεις όπου υλοποιείται Multilayer Perceptron Δίκτυο, μετασχηματίζονται οι εικόνες με τέτοιο τρόπο, ώστε να εισαχθούν με τον κατάλληλο τρόπο στους νευρώνες εισόδου του δικτύου. Επιπλέον, ακόμα και κατά την είσοδο εικόνων σε CNN, τα δεδομένα μετασχηματίζονται με τον απαιτούμενο τρόπο.
- **Διαχωρισμός δεδομένων σε για training,validation και testing:**
  Εξαρχής τα δεδομένα χωρίζονται σε training και testing δεδομένα, όπως αναφέρθηκε προηγουμένως. Επιπλέον όμως, για να γίνει και validation, ορίζεται κατά την εκπαίδευση ο όρος *validation_split=0.1*, όπου επιλέγεται από τα τελευταία δείγματα του training set ένα μέρος αυτών, για να δημιουργηθεί το validation set.
- **Κανονικοποίηση:**
  Αντί να χρησιμοποιούνται οι κανονικές τιμές των δειγμάτων, δηλαδή από την στιγμή που έχουμε εικόνες, τιμές από 0-255,οι τιμές κανονικοποιούνται σε εύρος τιμών 0-1. Με αυτόν τον τρόπο επιταχύνεται η εκπαίδευση του δικτύου και μειώνονται οι πιθανότητες να παγιδευτεί σε τοπικά ελάχιστα.
- **One-Hot κωδικοποίηση:**
  Τα διανύσματα που περιέχουν τις κλάσεις των δειγμάτων, κωδικοποιούνται με one-hot κωδικοποίηση, με σκοπό την χρήση categorical cross-entropy για την ταξινόμηση των δειγμάτων. Δηλαδή, η $1^{η}$ κλάση αντιστοιχίζεται σε 00..01, η $2^{η}$ σε 00..10 κ.ο.κ.
- **Image Augmentation:**
  Προκειμένου το δίκτυο να αποκτήσει «γενικότητα», τα δεδομένα training μετασχηματίζονται είτε με περιστροφή, με zoom, με shearing, με μετατοπίσεις ανά άξονα κ.τ.λ. . Έτσι, παράγονται περισσότερα δεδομένα τα οποία χρησιμοποιούν για να προσδώσουν στο δίκτυο ευελιξία στις δυνατότητες αναγνώρισης που έχει. Ωστόσο, επειδή οι εικόνες που χρησιμοποιούνται σε κάθε περίπτωση είναι διαμορφωμένες από πανεπιστήμια κ.τ.λ. που δεν έχουν ατέλειες, δεν χρησιμοποιείται Image Augmentation.

Είδη στρωμάτων που χρησιμοποιούνται

Στις υλοποιήσεις που ακολουθούν, χρησιμοποιήθηκαν τα εξής στρώματα:

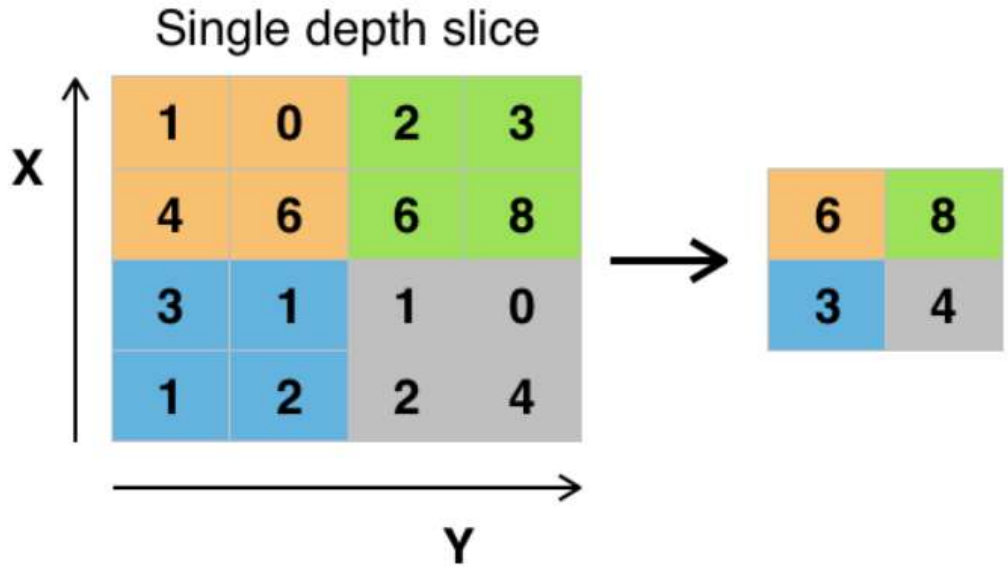
- **Conv2D:**

Πρόκειται για το layer που υλοποιεί την συνέλιξη μεταξύ της εισόδου και του πυρήνα, ώστε να ανιχνεύονται χαρακτηριστικά που θα βοηθήσουν στην κατηγοριοποίηση των εικόνων. Μια παράμετρος του στρώματος αυτού, είναι το μέγεθος του πυρήνα, δηλαδή του φίλτρου που χρησιμοποιείται. Όσο πιο μικρό το μέγεθος του, ανιχνεύονται πιο τοπικά χαρακτηριστικά, ενώ όσο μεγαλύτερο είναι, εντοπίζονται πιο καθολικά και αντιπροσωπευτικά χαρακτηριστικά. Επιπλέον, το padding αποτελεί μια παράμετρο, όπου καθορίζει την συμπεριφορά του φίλτρου στα όρια της εικόνας. Πάντα έχει την τιμή *same*, δηλαδή το αποτέλεσμα της συνέλιξης να έχει ίδιες διαστάσεις με την είσοδο, προσθέτοντας μηδενικά στις ακριανές περιπτώσεις. Εν συνεχεία, υπάρχει η παράμετρος του stride, όπου καθορίζεται πόσα pixel θα παραλείπονται μεταξύ δύο διαδοχικών συνελίξεων. Τέλος, ορίζεται η συνάρτηση ενεργοποίησης, όπου χρησιμοποιείται η *relu* και η *leaky_relu* ενώ το πλήθος των καναλιών/μονάδων του επιπέδου όπου όσο μεγαλύτερο είναι, τόσο καλύτερα/περισσότερα χαρακτηριστικά μαθαίνονται από το δίκτυο.



(a) ReLU (b) LReLU/PReLU

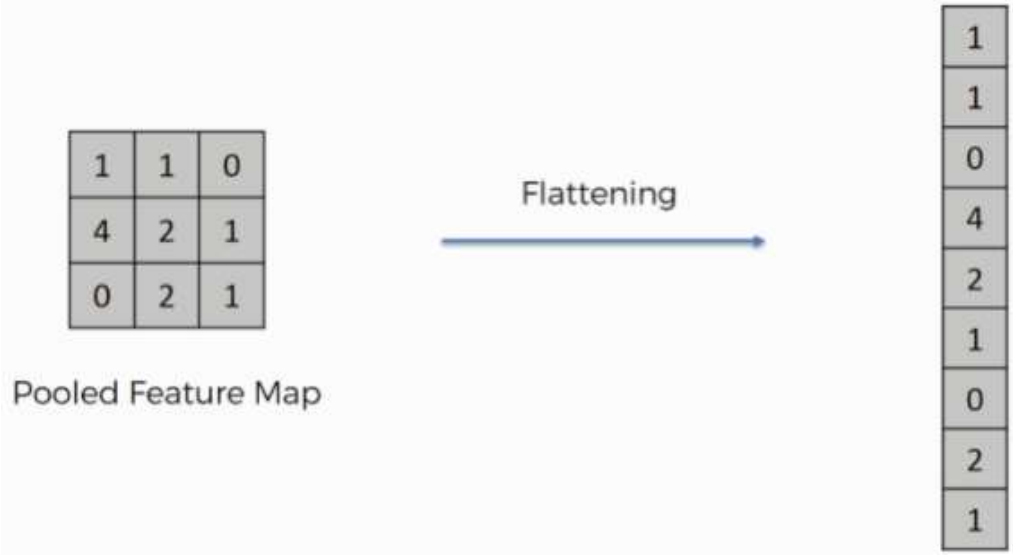
- **MaxPool:**



Παράδειγμα max-pooling, με μέγεθος 2x2 και stride 2

Πρόκειται για στρώμα μη-γραμμικής υπό-δειγματοληψίας. Ουσιαστικά, το αποτέλεσμα της συνέλιξης, χωρίζεται σε υπο-τετράγωνα (το μέγεθος των οποίων αποτελεί παράμετρο αυτού του στρώματος), από τα οποία επιλέγεται η μέγιστη τιμή του κάθε τετραγώνου. Έτσι, μειώνεται η πολυπλοκότητα της εκπαίδευσης, έχοντας ως μειονέκτημα την πιθανή απώλεια σημαντικής πληροφορίας. Για παράδειγμα, αν το τετράγωνο οριστεί να είναι 2x2 επιλέγεται η μέγιστη τιμή εκ των 4, ενώ αν οριστεί 3x3 επιλέγεται μία τιμή εκ των 9. Επιπλέον, υπάρχει και εδώ η δυνατότητα του stride, η οποία λειτουργεί με παρόμοιο τρόπο με το Convolutional Layer.

- **Dense:**
  Αποτελεί το πιο «κλασσικό» στρώμα, όπου οι νευρώνες του συνδέονται με κάθε νευρώνα του προηγούμενου επιπέδου. Σχεδόν πάντα αποτελεί το τελευταίο επίπεδο των νευρωνικών δικτύων, δηλαδή την έξοδο του, όπου γίνεται η ταξινόμηση. Επίσης, αποτελείται από τις παραμέτρους του πλήθους, πόσοι νευρώνες βρίσκονται σε αυτό το δίκτυο και από την συνάρτηση ενεργοποίησης, που είναι η *relu* ή η *leaky_relu*. Το Dense επίπεδο της εξόδου, ενεργοποιείται με την συνάρτηση softmax, μέσω της οποίας η ταξινόμηση εκφράζεται με μια πιθανότητα να ανήκει στην εκάστοτε κλάση το δείγμα.
- **Flatten:**
  Μετατρέπει την είσοδο που δέχεται σε μονοδιάστατο διάνυσμα, προκειμένου να εισαχθεί σαν είσοδος στο dense/fully connected επίπεδο που ακολουθεί.



Παράδειγμα flattening

- **Dropout:**
  Με την χρήση αυτού του στρώματος, μειώνεται η επίδραση του φαινομένου overfitting, καθώς απενεργοποιούνται με τυχαίο τρόπο συνδέσεις μεταξύ των νευρώνων, προκειμένου το δίκτυο να μην βασίζεται σε κάποιο συγκεκριμένο χαρακτηριστικό. Δέχεται ως όρισμα μια τιμή που παίρνει τιμές από 0 έως 1, καθώς εκφράζει την πιθανότητα απενεργοποίησης κάποιου νευρώνα.

- **BatchNormalization :**
  Το στρώμα αυτό χρησιμοποιείται με σκοπό την κανονικοποίηση των εξόδων των παραπάνω στρωμάτων. Μέσω της κανονικοποίησης, επιτυγχάνεται ταχύτητα στην εκπαίδευση του δικτύου και πιο σταθερά αποτελέσματα. Κατά κανόνα, χρησιμοποιούνται μετά τα στρώματα Dense,Conv2D και πριν τα Dropout layers.

Ρύθμιση Υπερπαραμέτρων

- **Εποχές & Callbacks:**
  Υποδεικνύει πόσες φορές θα περάσουν όλα τα δείγματα από το δίκτυο. Μία εποχή ισοδυναμεί με το πέρασμα όλων των δειγμάτων μία φορά. Πολύ μεγάλη τιμή έχει τον κίνδυνο του overfitting, δηλαδή της πολύ καλής εκπαίδευσης του δικτύου μόνο στα δείγματα εκπαίδευσης και την απουσία ικανότητας γενίκευσης. Αντίθετα, μικρή τιμή ελλοχεύει τον κίνδυνο της μη ορθής εκπαίδευσης του δικτύου και της αδυναμίας σωστής κατηγοριοποίησης. Τίθεται ίση με 50 ή 100, ωστόσο με την δυνατότητα των callbacks, δεν είναι απαραίτητο να γίνουν όλες οι επαναλήψεις. Ειδικότερα, με τα callbacks είναι δυνατή η αποθήκευση των καλύτερων βαρών και στην συνέχεια, αν δεν βελτιωθεί ένα κριτήριο για ένα προκαθορισμένο όριο εποχών, η εκπαίδευση σταματάει και χρησιμοποιούνται τα αποθηκευμένα βάρη. Το όριο αυτό έχει την τιμή 10 ή 15, ενώ τέλος, η μεταβλητή που χρησιμοποιείται ως κριτήριο είναι το validation loss.
- **Batch Size:**
  Η τιμή της ορίζει μετά από ποιον αριθμό εικόνων θα γίνει η ανανέωση των βαρών. Μικρή τιμή σημαίνει πως τα βάρη ανανεώνονται συνεχώς, με αποτέλεσμα την αδυναμία σύγκλισης της εκπαίδευσης, ενώ μεγάλη τιμή θα οδηγήσει σε γρήγορη σύγκλιση των βαρών, όπου θα είναι αδύνατη η εκπαίδευση χαρακτηριστικών υψηλών διαστάσεων. Έχει τιμή 100,250 ή 500 , ενώ μερικές φορές δεν ορίζεται κάποια τιμή, με αποτέλεσμα να υπολογίζεται αυτόματα εκείνη την ώρα.
- **Επιλογή Optimizer & Learning Rate:**
  Ο optimizer ουσιαστικά είναι ο αλγόριθμος ανανέωσης των βαρών κάθε στρώματος. Ο Adam έχει ευρεία χρήση και θεωρείται από τους καλύτερους, οπότε χρησιμοποιείται αυτός. Επιπλέον, χρησιμοποιείται και ο Nadam , που προσφέρει και αυτός ικανοποιητικά αποτελέσματα και λειτουργεί με παρόμοιο τρόπο με τον Adam. Ο ρυθμός μάθησης παίζει ρόλο στην ταχύτητα σύγκλισης και στην δυνατότητα διαφυγής από local optima. Μεγάλη τιμή έχει ως αποτέλεσμα την πιθανή απομάκρυνση από local optima, αλλά και την ταχεία σύγκλιση του αλγορίθμου. Αντίθετα, μικρή τιμή οδηγεί σε πιο αργή σύγκλιση αλλά και πιθανή παγίδευση της διαδικασίας σε local optima. Ως default τιμή τίθεται η 0.001, ωστόσο δοκιμάζεται και η τιμή 0.01. Επιπλέον, υπάρχει η δυνατότητα σταδιακής ελάττωσης του ρυθμού μάθησης, όσο προχωράει η εκπαίδευση. Αυτή η επιλογή, δεν είναι διαθέσιμη για

τον Nadam optimizer, ενώ δεν παρατηρήθηκε αύξηση της απόδοσης όταν χρησιμοποιήθηκε μεταβλητό learning rate.

- **Πλήθος hidden layer και αριθμός νευρώνων στο καθένα:**
  Πρέπει να επιλεχθούν κατάλληλα, διότι υψηλός αριθμός οδηγεί σε overfitting και στην δημιουργία ενός υπερβολικά σύνθετου δικτύου, ενώ χαμηλός αριθμός οδηγεί σε αδυναμία αποτελεσματικής αναγνώρισης και/ή underfitting. Επιλέγονται έπειτα από trial & error στα δίκτυα ιδίας υλοποίησης, καθώς καθορίζονται οι τιμές τους στα άλλα ζητούμενα. Συνήθης τιμές είναι οι 16,32,64,128,256 κ.τ.λ.
- **Συναρτήσεις Ενεργοποίησης:**
  Όπως έχει ήδη αναφερθεί, χρησιμοποιούνται οι relu ή leaky_relu. Όμως υπάρχουν οι sigmoid,tanh κ.α.
- **Δυνατότητα Shuffle:**
  Με την ενεργοποίηση του shuffle, τα δείγματα εισέρχονται με τυχαίο τρόπο στο δίκτυο κάθε εποχή, αντί να εισέρχονται κάθε φορά με την ίδια σειρά. Η τυχαιότητα αυτή, έχει παρατηρηθεί ότι αυξάνει την απόδοση του δικτύου σε μερικές περιπτώσεις. Όπου παρατηρήθηκε βελτίωση της ταξινόμησης, η επιλογή αυτή έχει ενεργοποιηθεί.

Δίκτυα που υλοποιήθηκαν

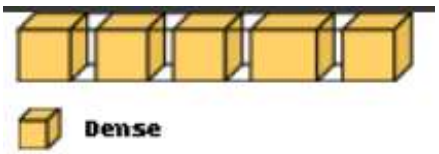
Γενικά:

- Στις προσωπικές υλοποιήσεις επιχειρήθηκε ένας συμβιβασμός μεταξύ training parameters και accuracy.
- Χρησιμοποιήθηκαν σχεδόν σε απόλυτο βαθμό φίλτρα διαστάσεων 3x3, καθώς φίλτρα με ζυγά μεγέθη διαιρούν συμμετρικά τα πίξελ του προηγούμενου επιπέδου γύρω από την έξοδο, ενώ 1x1 εμφανίζουν υπερβολικά μεγάλη τοπικότητα στα χαρακτηριστικά που εντοπίζουν.
- Ακολουθήθηκε η λογική ότι όσο βαθαίνει το δίκτυο, αυξάνεται ο αριθμός των νευρώνων στα εκάστοτε στρώματα, ενώ μετά από Conv2d layer ακολουθεί σχεδόν πάντα ένα max-pooling και ένα dropout layer.
- Το loss ως μετρική αναφέρεται στην τιμή της loss function, που ορίζεται να είναι η crossentropy.

- Fashion_MNIST Dataset (*αρχείο fashion_mnist_57576.ipynb)*:
  1. Multi-Layer Perceptron Network(FC NN):
     Ακολουθώντας την υπόδειξη της εκφώνησης σχετικά με αυτό το δίκτυο, υλοποιείται ένα Multilayered Perceptron δίκτυο, με 3 κρυφά επίπεδα, 64, 128 και 256 νευρώνων(units). Επιπλέον, το επίπεδο εξόδου περιλαμβάνει 10 units, καθώς οι κλάσεις στο συγκεκριμένο πρόβλημα είναι 10, ενώ το επίπεδο εισόδου έχει

input_shape ίσο με 28*28*1 = 784, αριθμός που προκύπτει από τις διαστάσεις της εικόνας. Στην εικόνα που ακολουθεί αποτυπώνεται ο αριθμός των παραμέτρων του δικτύου (μέσω του *model.summary*) και η δομή του δικτύου, οπτικοποιημένη με το εργαλείο *VisualKeras*:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                25120

 dense_1 (Dense)             (None, 64)                2112

 dense_2 (Dense)             (None, 128)               8320

 dense_3 (Dense)             (None, 256)               33024

 dense_4 (Dense)             (None, 10)                2570

=================================================================
Total params: 71,146
Trainable params: 71,146
Non-trainable params: 0
```

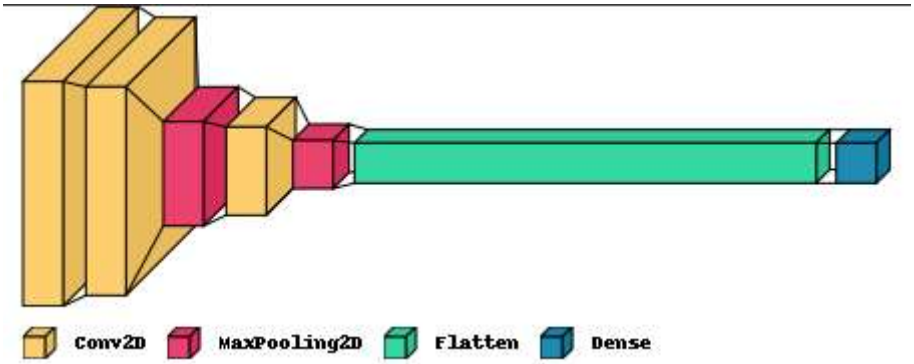


Με τις trainable parameters, να αναφέρονται στις μονάδες που μπορούν να αλλάξουν τα βάρη τους μέσω backpropagation (ή και άλλων αλγορίθμων) , ενώ οι non-trainable parameters, αντιστοιχούν στις μονάδες των που δεν μπορούν να αλλάξουν τα βάρη τους, όπως για παράδειγμα αυτές των Dropout & BatchNormalization στρωμάτων.

2. <u>Convolutional Network (CNN):</u>

   Ομοίως με πριν, το ζητούμενο δίκτυο που αναφέρεται στη εκφώνηση περιέχει 2 κρυφά επίπεδα με 32 και 64 φίλτρα, το καθένα ακολουθούμενο από ένα max-pooling επίπεδο. Το επίπεδο εισόδου περιέχει 32 φίλτρα(αυθαίρετη τιμή) και ως input_shape τίθενται οι διαστάσεις των εικόνων, δηλαδή 28x28x1(1, διότι οι εικόνες είναι σε grayscale). Το επίπεδο εξόδου, έχει 10 μονάδες, όσες και οι κλάσεις των εικόνων. Στην επόμενη εικόνα φαίνεται η έξοδος της εντολής *model.summary* :

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 conv2d_1 (Conv2D)           (None, 26, 26, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten (Flatten)           (None, 2304)              0

 dense_5 (Dense)             (None, 10)                23050

=================================================================
Total params: 51,114
Trainable params: 51,114
Non-trainable params: 0
```

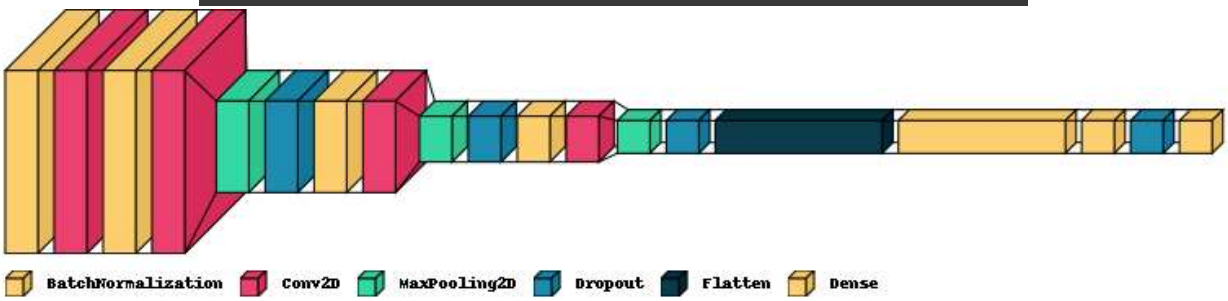


Το στρώμα flatten, ουσιαστικά χρησιμοποιείται για να μετατρέψει το feature map που προκύπτει σε διάνυσμα, ώστε να μπορέσει εν συνεχεία να γίνει η ταξινόμηση στο τελευταίο στρώμα, το στρώμα εξόδου.

3. <u>Προσωπική Υλοποίηση:</u>
   Επιλέχθηκε να υλοποιηθεί ένα συνελικτικό νευρωνικό δίκτυο, έχοντας κατά νου την αύξηση της απόδοσης με όσο το δυνατόν λιγότερες παραμέτρους. Όσο πιο πολλές παράμετροι, τόσο πιο πολλοί πόροι χρησιμοποιούνται από το δίκτυο. Έτσι, έπειτα από διάφορες δοκιμές, κατέληξα στο εξής δίκτυο:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 batch_normalization (BatchN  (100, 28, 28, 1)         112
 ormalization)

 conv2d_3 (Conv2D)           (100, 28, 28, 32)         320

 batch_normalization_1 (Batc  (100, 28, 28, 32)        112
 hNormalization)

 conv2d_4 (Conv2D)           (100, 28, 28, 64)         18496

 max_pooling2d_2 (MaxPooling  (100, 14, 14, 64)        0
 2D)

 dropout (Dropout)           (100, 14, 14, 64)         0

 batch_normalization_2 (Batc  (100, 14, 14, 64)        56
 hNormalization)

 conv2d_5 (Conv2D)           (100, 14, 14, 64)         16448

 max_pooling2d_3 (MaxPooling  (100, 7, 7, 64)          0
 2D)

 dropout_1 (Dropout)         (100, 7, 7, 64)           0

 batch_normalization_3 (Batc  (100, 7, 7, 64)          28
 hNormalization)

 conv2d_6 (Conv2D)           (100, 7, 7, 64)           36928

 max_pooling2d_4 (MaxPooling  (100, 4, 4, 64)          0
 2D)

 dropout_2 (Dropout)         (100, 4, 4, 64)           0

 flatten_1 (Flatten)         (100, 1024)               0

 batch_normalization_4 (Batc  (100, 1024)              4096
 hNormalization)
```

```
dense_6 (Dense)              (100, 32)                 32800

dropout_3 (Dropout)          (100, 32)                 0

dense_7 (Dense)              (100, 10)                 330

=================================================================
Total params: 109,726
Trainable params: 107,524
Non-trainable params: 2,202
_________________________________________________________________
```



Το οποίο ουσιαστικά αποτελεί μια εξέλιξη του προηγούμενου δικτύου.

4. <u>Σύγκριση Αποτελεσμάτων:</u>
   Αξίζει να σημειωθεί πως τα αποτελέσματα είναι ενδεικτικά, καθώς σημαντικό ρόλο παίζει η αρχικοποίηση των βαρών και οι τιμές που λαμβάνουν οι τυχαίες παράμετροι του δικτύου, όπου αλλάζουν κάθε φορά που εκτελείται από την αρχή η εκπαίδευση. Έτσι, τα παρακάτω αποτελέσματα αποτελούν μια γενική «εικόνα» της λειτουργίας των εκάστοτε δικτύων.
   Αρχικά, εκπαιδεύοντας τα παραπάνω δίκτυα και χρησιμοποιώντας το test set, μέσω της συνάρτησης *model.evaluate* προκύπτουν τα εξής αποτελέσματα:

```
Test loss: 0.3622902035713196
Test accuracy: 0.8731999993324280
```

Εικόνα 4.1 Αποτελέσματα τρέχοντας το fully connected δίκτυο

```
Test loss: 0.2415214180946350
Test accuracy: 0.9165999889373779
```

Εικόνα 4.2 Αποτελέσματα τρέχοντας το convolutional δίκτυο

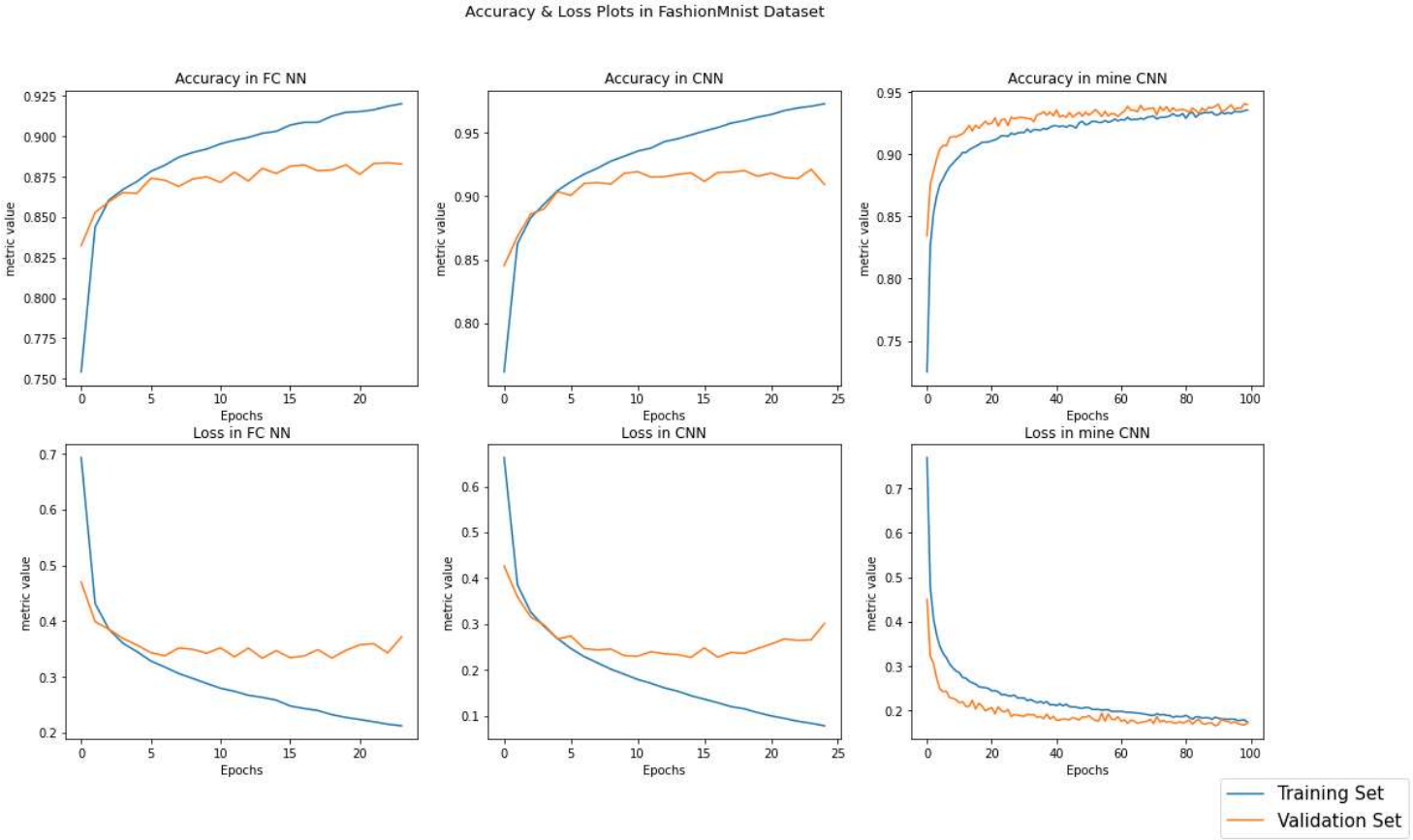
```
Test loss: 0.1892558783292770
Test accuracy: 0.9359999895095825
```

Εικόνα 4.3 Αποτελέσματα τρέχοντας την προσωπική μου υλοποίηση

Η τιμή test loss αντιστοιχεί στην τιμή που παίρνει η cost function, ενώ η τιμή test accuracy αντιστοιχεί στο ποσοστό σωστών ταξινομήσεων του δικτύου.

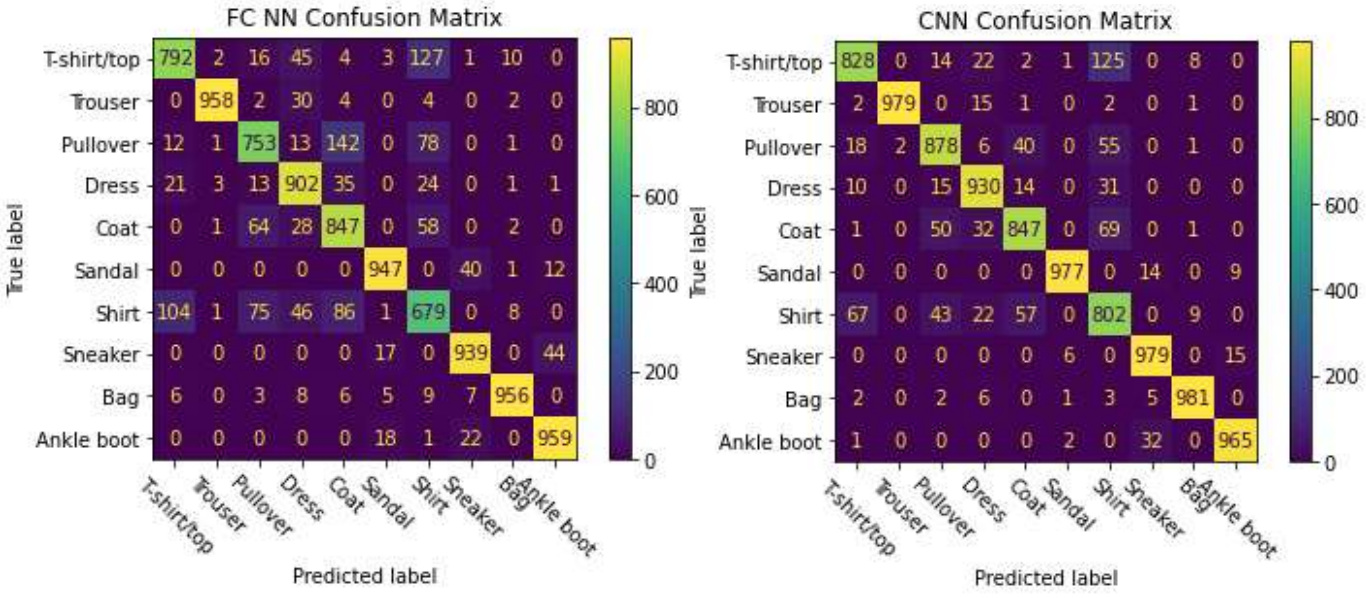
Ακολούθως, παρατηρώντας την εξέλιξη των training,validation accuracy & loss, γίνεται αντιληπτή η ροή της εκπαίδευσης και το αν αυτή γίνεται καλά ή όχι, όπου παρατηρείται πως όσο πιο σύνθετο γίνεται το μοντέλο, τόσο περισσότερες εποχές

απαιτούνται για να επιτευχθούν αποδεκτά αποτελέσματα (δεδομένου ότι αυξάνεται και το μέγεθος του δικτύου). Επίσης φαίνεται πως υπάρχει μια διακύμανση στις μετρικές που αναφέρονται στο validation set, ενώ αντίθετα οι μετρικές του training set έχουν μια ομαλότερη εξέλιξη:

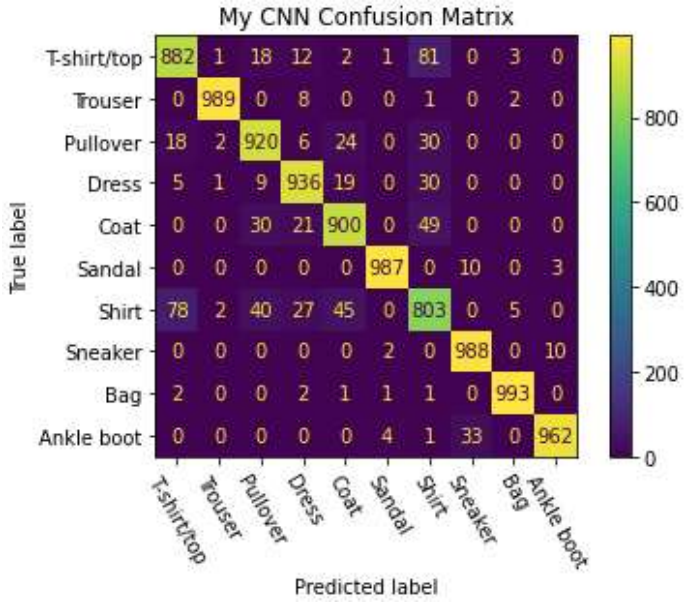


Εικόνα 4.4 Γραφήματα μετρικών loss,accuracy για κάθε περίπτωση

Εν συνεχεία, με την χρήση confusion matrix, μπορεί να γίνει πιο εύκολα αντιληπτό, το πως γίνεται η ταξινόμηση των δειγμάτων του test set, καθώς γίνεται απεικόνιση των ταξινομήσεων που γίνονται σε κάθε κλάση:



Εικόνα 4.5α,β Confusion matrices για Multilayer Perceptron Network(αριστερά) & Convolutional NN (δεξιά)

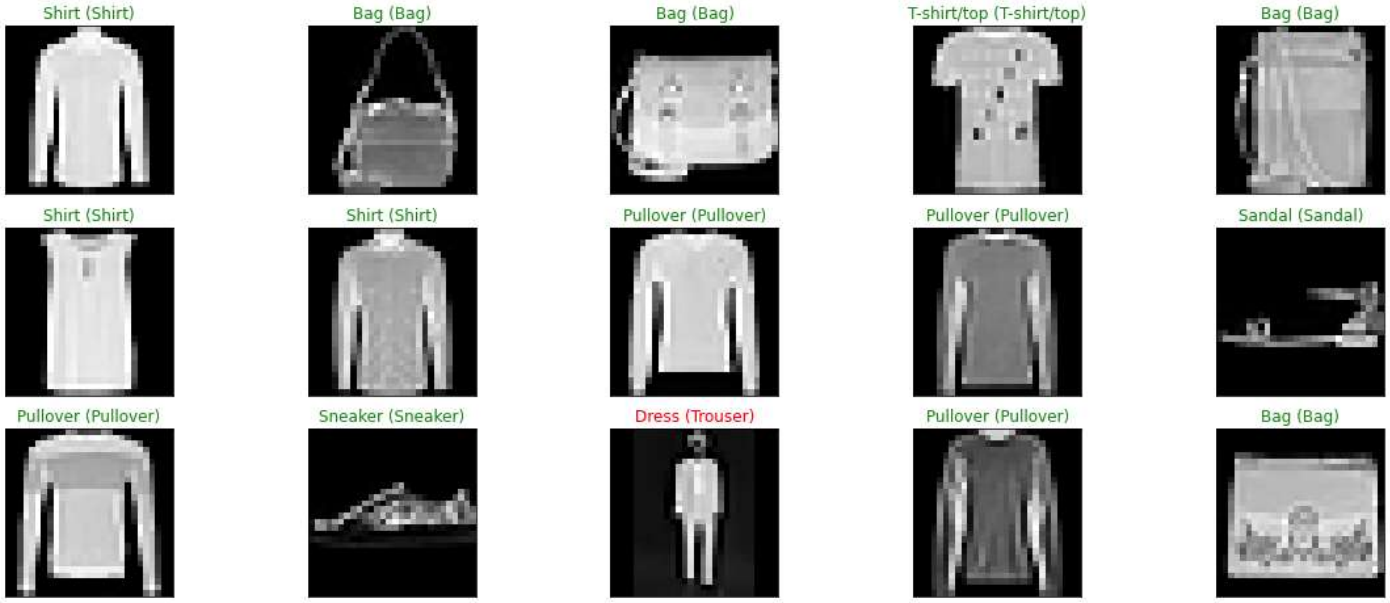


Εικόνα 4.5γ συνέχεια Confusion Matrix για την προσωπική μου υλοποίηση

Με βάση τα παραπάνω, παρατηρώ πως ιδιαίτερο πρόβλημα δημιουργούν οι κλάσεις T-shirt/Top και Shirt, καθώς όπως είναι λογικό μοιάζουν πολύ μεταξύ τους. Φαίνεται πως τα μανίκια ως χαρακτηριστικό είναι αυτό που δυσκολεύει το δίκτυο, καθώς οι λάθος ταξινομήσεις των δειγμάτων Shirt γίνονται ως επί το πλείστων σε κλάσεις που τα δείγματά τους περιέχουν επίσης μανίκια.

Τέλος, ενδεικτικά παρουσιάζονται 15 προβλέψεις από κάθε μοντέλο για να γίνει κατανοητή η απόδοση κάθε δικτύου (παρόλο που οι 15 εικόνες επιλέγονται τυχαία, μπορεί να εξαχθεί ένα συμπέρασμα):

Εικόνα 4.6 Αποτελέσματα προβλέψεων χρησιμοποιώντας το FC NN



Εικόνα 4.7 Αποτελέσματα προβλέψεων χρησιμοποιώντας το CNN

Εικόνα 4.8 Αποτελέσματα προβλέψεων χρησιμοποιώντας το δικό μου CNN

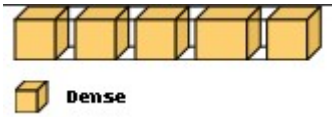
Εν κατακλείδι, παρατηρείται πως όσο αυξάνεται η πολυπλοκότητα της δομής του δικτύου, αυξάνεται η ευστοχία του μοντέλου, ενώ αντίστοιχα μειώνεται η απώλεια (loss) αυτού. Επιπλέον, επειδή το συγκεκριμένο dataset είναι σχετικά απλό, οι ευστοχίες των μοντέλων είναι κοντά μεταξύ τους, κάτι που δεν ισχύει για τα επόμενα dataset.

- Cifar10 Dataset (*αρχείο cifar10 _57576.ipynb)*:
  1. Multi-Layer Perceptron Network (FC NN):
     Ομοίως με το προηγούμενο dataset, δημιουργείται ένα Fully Connected NN, με την διαφορά ότι υπάρχει ένα ακόμα κρυφό επίπεδο, μεγέθους 256. Επιπλέον, αλλάζουν και οι διαστάσεις εισόδου, καθώς πλέον οι εικόνες είναι 32x32 RGB (δηλαδή 3 κανάλια, κόκκινο, πράσινο, μπλε), άρα οι διαστάσεις εισόδου είναι 32*32*3 = 3072. Εφόσον οι κλάσεις είναι και πάλι 10, το επίπεδο εξόδου τύπου

dense έχει 10 νευρώνες, μία για κάθε κλάση (όπως και προηγουμένως). Έτσι, το *model.summary* έχει την εξής έξοδο, με την διπλανή εικόνα να είναι η οπτικοποίηση του δικτύου:

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 32)	98336
dense_6 (Dense)	(None, 64)	2112
dense_7 (Dense)	(None, 128)	8320
dense_8 (Dense)	(None, 256)	33024
dense_9 (Dense)	(None, 10)	2570

Total params: 144,362
Trainable params: 144,362
Non-trainable params: 0



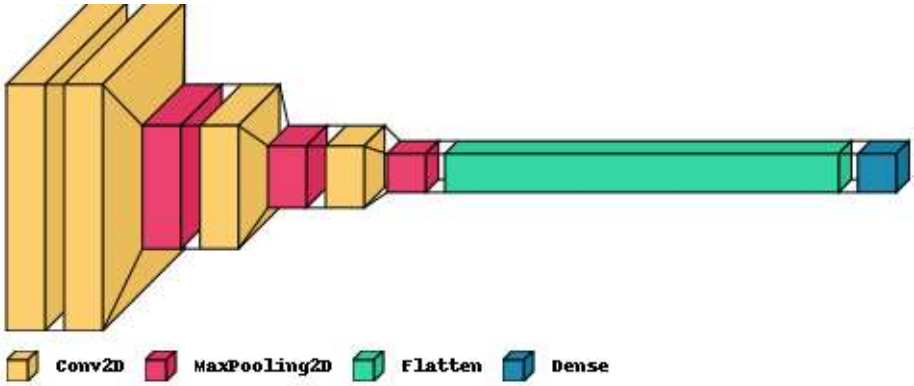
Πλέον, με την αύξηση των επιπέδων, αυξήθηκαν και οι trainable παραμέτροι του δικτύου, όπως είναι αναμενόμενο.

2. Convolutional Network (CNN):

Όπως και στο προηγούμενο αντίστοιχο δίκτυο, πλέον υπάρχουν 3 hidden layers με 64,128,256 μονάδες, ακολουθούμενα από max-pooling στρώματα, ενώ το επίπεδο εξόδου έχει 10 μονάδες για τον ίδιο λόγο με πριν και τέλος το επίπεδο εισόδου έχει input shape τις διαστάσεις των εικόνων (32x32x3) με αυθαίρετη τιμή 32 μονάδων:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense_5 (Dense)	(None, 10)	20490

Total params: 122,986
Trainable params: 122,986
Non-trainable params: 0

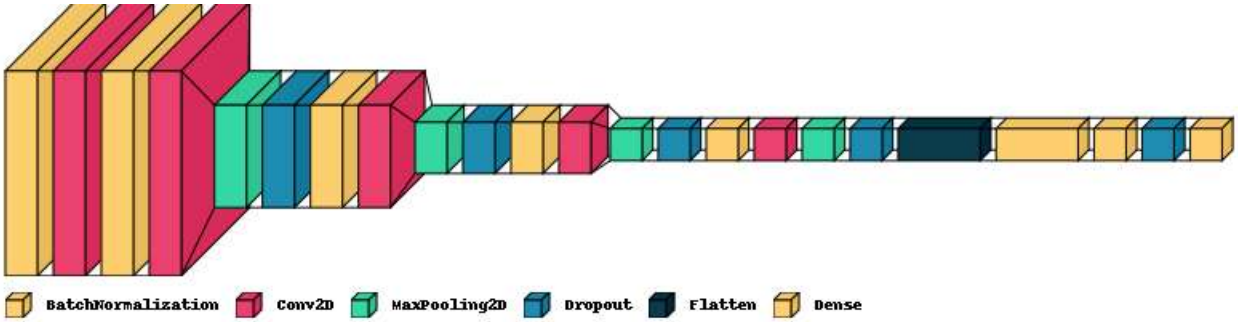


3. <u>Προσωπική Υλοποίηση:</u>
   Επειδή το συγκεκριμένο dataset είναι σύνθετο, και για να επιτευχθούν όσο το δυνατόν καλύτερα αποτελέσματα, επιλέχθηκε ένα μοντέλο με 4 hidden layers με 32,64,64,128 μονάδες το καθένα αντίστοιχα, αυξάνοντας με αυτόν τον τρόπο τις συνολικές παραμέτρους , όπως φαίνεται παρακάτω:

Layer (type)	Output Shape	Param #
batch_normalization (BatchNormalization)	(100, 32, 32, 3)	128
conv2d_4 (Conv2D)	(100, 32, 32, 32)	896
batch_normalization_1 (BatchNormalization)	(100, 32, 32, 32)	128
conv2d_5 (Conv2D)	(100, 32, 32, 32)	9248
max_pooling2d_3 (MaxPooling2D)	(100, 16, 16, 32)	0
dropout (Dropout)	(100, 16, 16, 32)	0
batch_normalization_2 (BatchNormalization)	(100, 16, 16, 32)	64
conv2d_6 (Conv2D)	(100, 16, 16, 64)	18496
max_pooling2d_4 (MaxPooling2D)	(100, 8, 8, 64)	0
dropout_1 (Dropout)	(100, 8, 8, 64)	0
batch_normalization_3 (BatchNormalization)	(100, 8, 8, 64)	32
conv2d_7 (Conv2D)	(100, 8, 8, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(100, 4, 4, 64)	0
dropout_2 (Dropout)	(100, 4, 4, 64)	0
batch_normalization_4 (BatchNormalization)	(100, 4, 4, 64)	16
conv2d_8 (Conv2D)	(100, 4, 4, 128)	73856
max_pooling2d_6 (MaxPooling2D)	(100, 2, 2, 128)	0
dropout_3 (Dropout)	(100, 2, 2, 128)	0
flatten_1 (Flatten)	(100, 512)	0

```
batch_normalization_5 (Batc  (100, 512)              2048
hNormalization)

dense_6 (Dense)              (100, 16)               8208

dropout_4 (Dropout)          (100, 16)               0

dense_7 (Dense)              (100, 10)               170

=================================================================
Total params: 150,218
Trainable params: 149,010
Non-trainable params: 1,208
_________________________________________________________________
```

Ενώ η αδρή εικόνα του δικτύου είναι:



4. <u>Σύγκριση Αποτελεσμάτων:</u>

   Η επίδοση των δικτύων αποτυπώνεται στις επόμενες εικόνες, όσο αναφορά την τιμή της συνάρτησης σφάλματος(categorical crossentropy) και της ευστοχίας του δικτύου:

```
Test loss: 1.4195231199264526
Test accuracy: 0.5001000165939331
```

Εικόνα 4.1 Αποτελέσματα τρέχοντας το fully connected δίκτυο

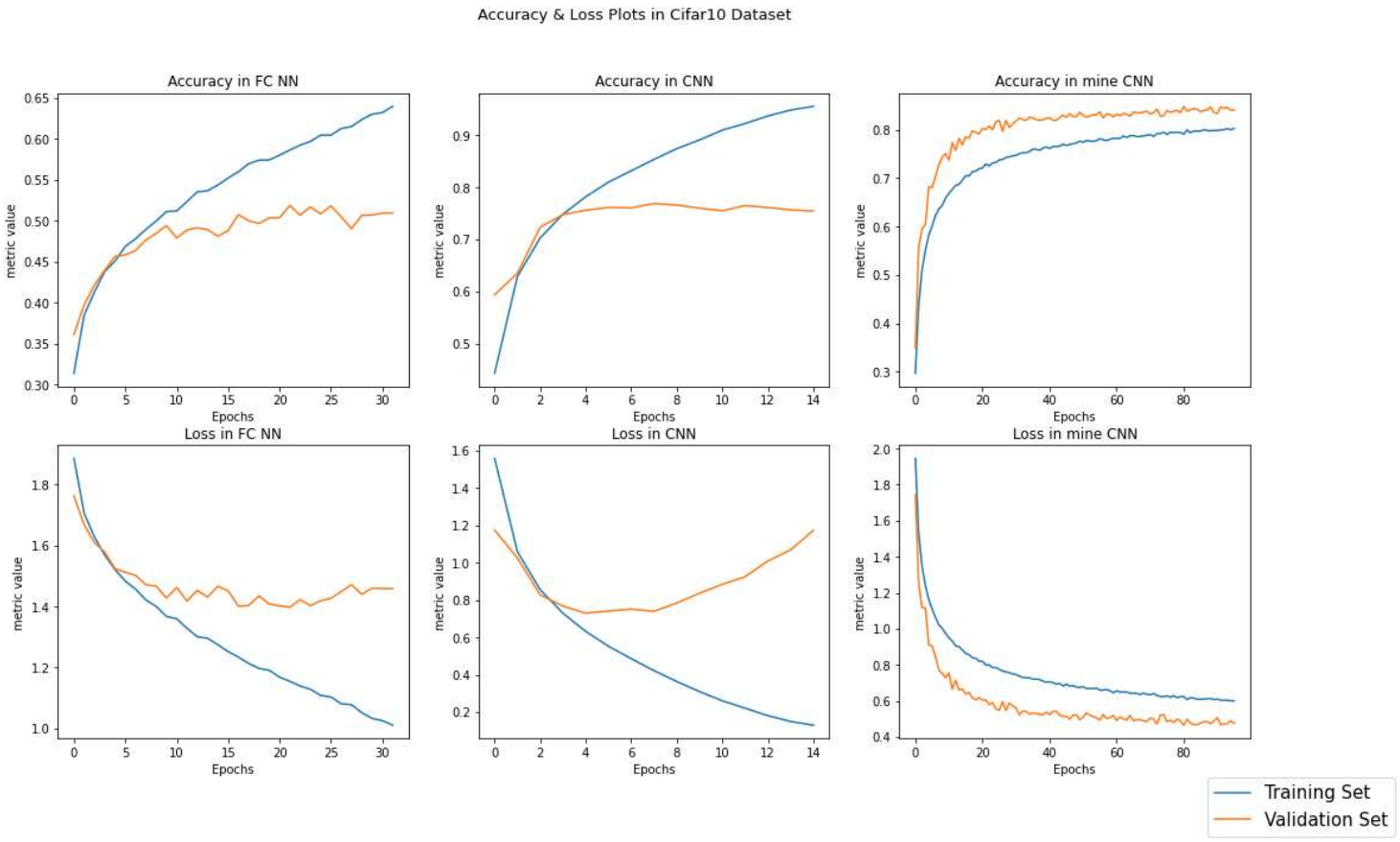
```
Test loss: 0.7626890540122986
Test accuracy: 0.7437999844551086
```

Εικόνα 4.2 Αποτελέσματα τρέχοντας το convolutional δίκτυο

```
Test loss: 0.48939210176467896
Test accuracy: 0.8429999947547913
```

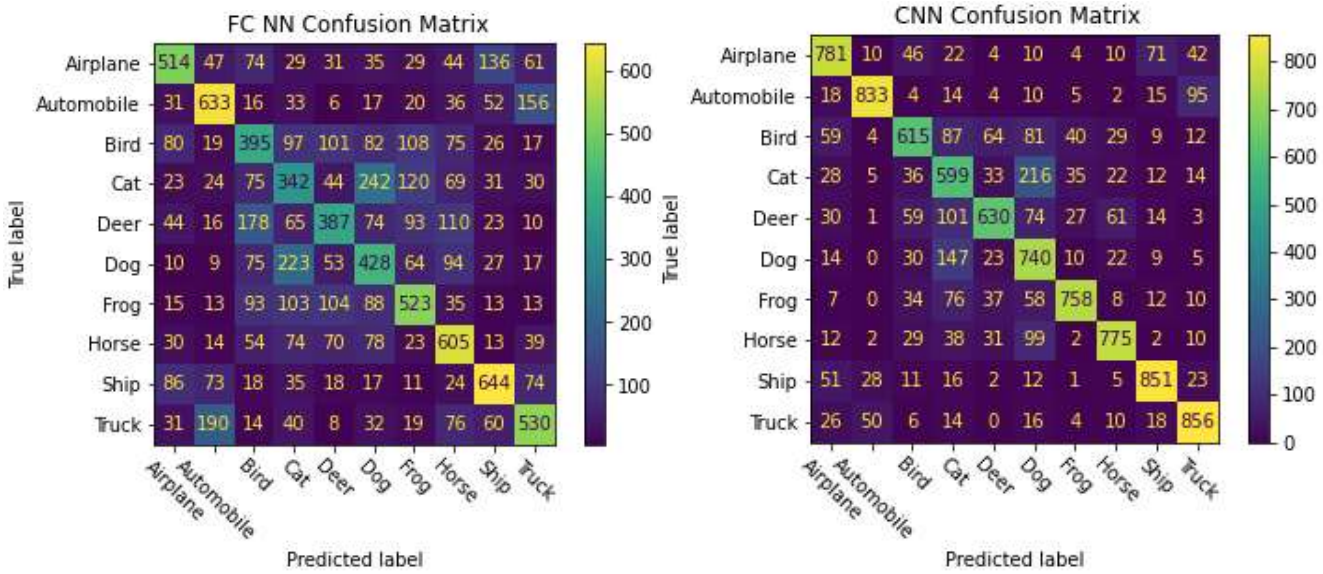
Εικόνα 4.3 Αποτελέσματα τρέχοντας την προσωπική μου υλοποίηση

Δεδομένου ότι χρησιμοποιούνται λίγο παραπάνω παράμετροι από το ζητούμενο CNN δίκτυο, το αποτέλεσμα είναι αρκετά ικανοποιητικό. Εν συνεχεία, παρατηρώντας τα plot των μετρικών του loss & του accuracy, εμφανίζεται η ίδια γενική εικόνα με πριν, μόνο που χρειάζονται παραπάνω εποχές για την επίτευξη των παραπάνω αποτελεσμάτων. Το γεγονός ότι εμφανίζεται διακύμανση στις μετρικές του validation set αποτελεί ένα καλό σημάδι ότι δεν υπάρχει overfitting.

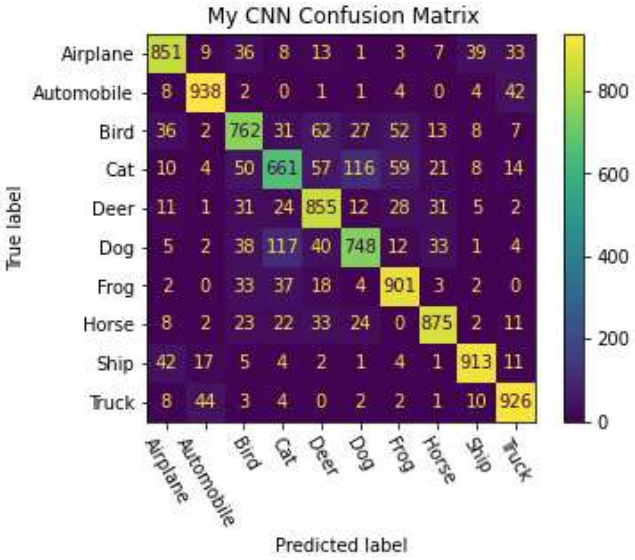


Εικόνα 4.4 Γραφήματα μετρικών loss,accuracy για κάθε περίπτωση

Στις επόμενες εικόνες αποτυπώνονται οι confusion matrices κάθε δικτύου. Σύμφωνα με αυτούς, ιδιαίτερο πρόβλημα στο δίκτυο δημιουργούν οι κλάσεις dog & cat, καθώς και σε λιγότερο βαθμό οι κλάσεις car & truck, όπως και το ζεύγος deer & horse. Μεταξύ αυτών των κλάσεων εμφανίζονται τα περισσότερα λάθος ταξινομημένα δείγματα.

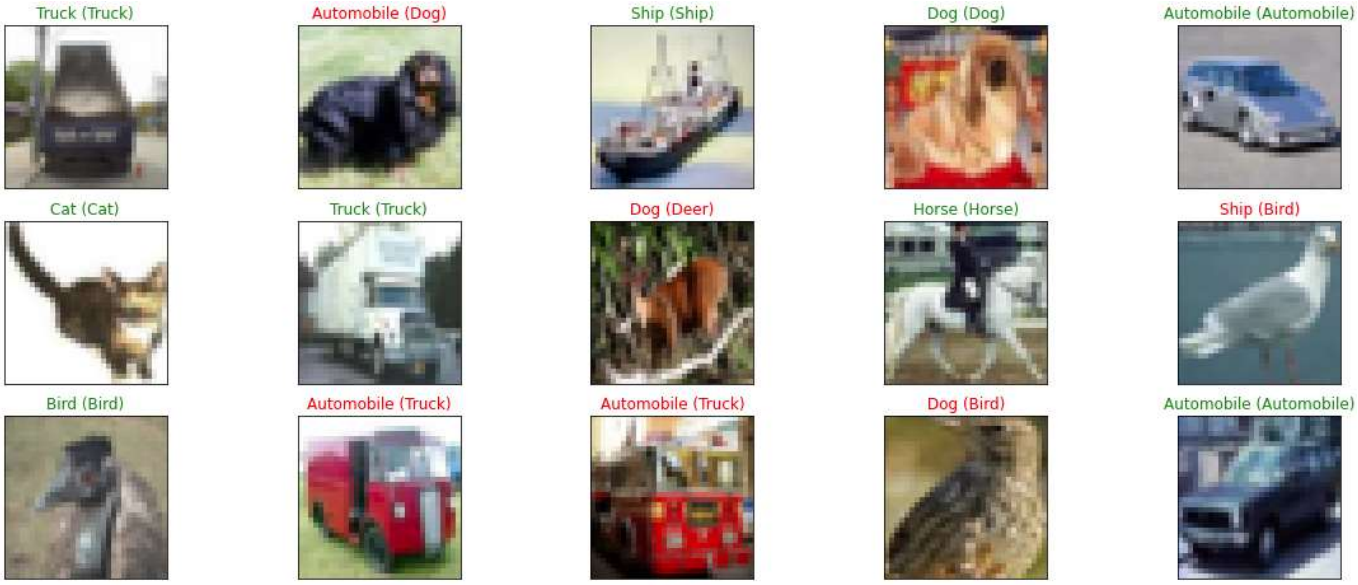


Εικόνα 4.5α,β Confusion matrices για Multilayer Perceptron Network(αριστερά) & Convolutional NN (δεξιά)

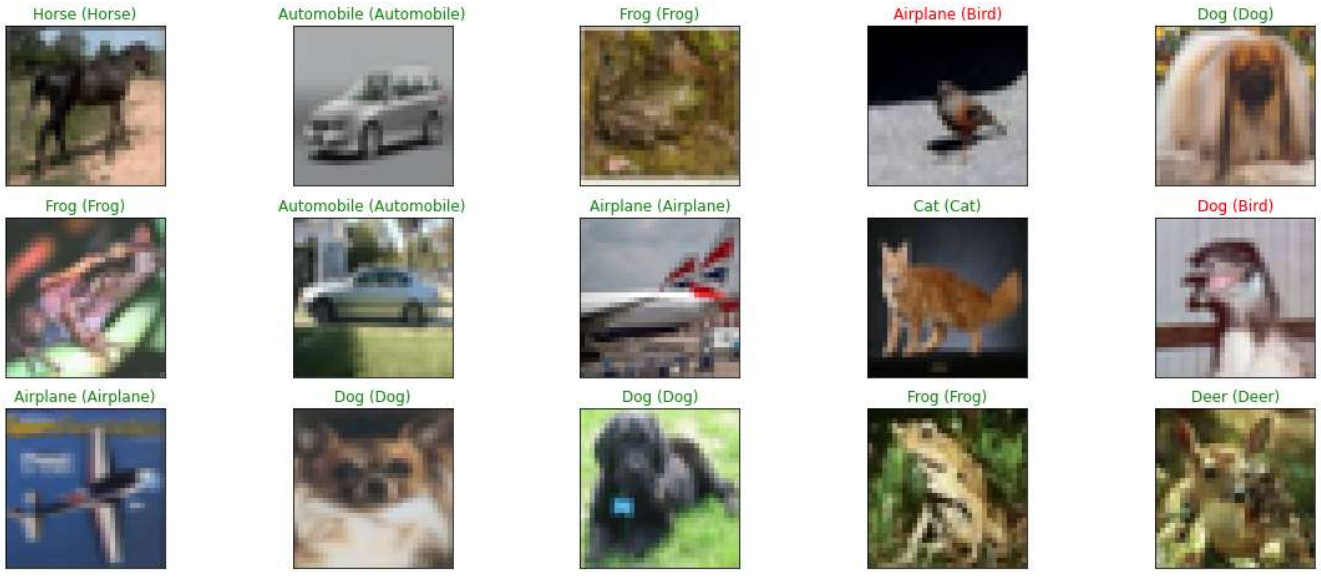


Εικόνα 4.5γ συνέχεια Confusion Matrix για την προσωπική μου υλοποίηση

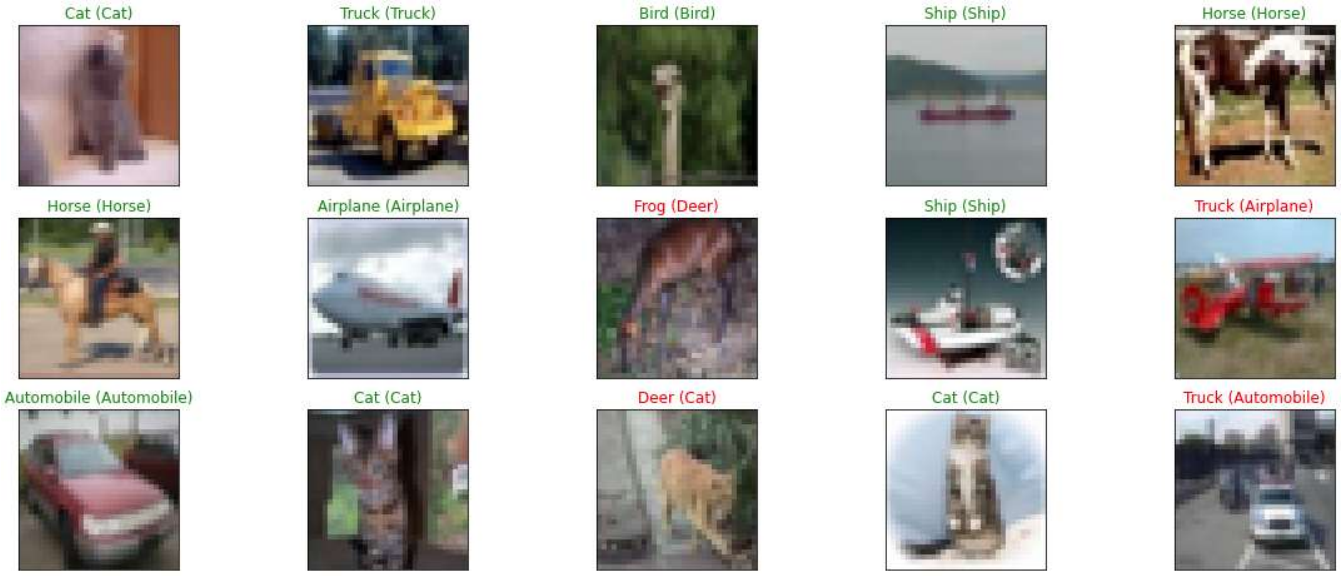
Τέλος, με βάση τα παραπάνω δίκτυα, γίνονται προβλέψεις για την ταξινόμηση των εικόνων του test set με τα εξής αποτελέσματα:



Εικόνα 4.6 Αποτελέσματα προβλέψεων χρησιμοποιώντας το FC NN

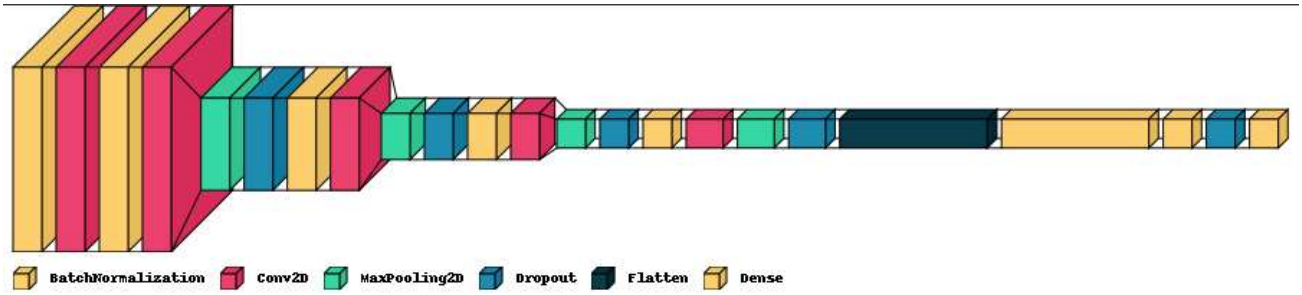


Εικόνα 4.7 Αποτελέσματα προβλέψεων χρησιμοποιώντας το CNN



Εικόνα 4.8 Αποτελέσματα προβλέψεων χρησιμοποιώντας το δικό μου CNN

- Cifar100 Dataset (*αρχείο cifar100 _57576.ipynb)*:



1. <u>Προσωπική Υλοποίηση:</u>
   Βασιζόμενος στις προηγούμενες υλοποιήσεις, τελικά το δίκτυο που χρησιμοποιήθηκε είναι το εξής:

Layer (type)	Output Shape	Param #
batch_normalization (BatchNormalization)	(100, 32, 32, 3)	128
conv2d (Conv2D)	(100, 32, 32, 32)	896
batch_normalization_1 (BatchNormalization)	(100, 32, 32, 32)	128
conv2d_1 (Conv2D)	(100, 32, 32, 64)	18496
max_pooling2d (MaxPooling2D)	(100, 16, 16, 64)	0
dropout (Dropout)	(100, 16, 16, 64)	0
batch_normalization_2 (BatchNormalization)	(100, 16, 16, 64)	64
conv2d_2 (Conv2D)	(100, 16, 16, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(100, 8, 8, 128)	0
dropout_1 (Dropout)	(100, 8, 8, 128)	0
batch_normalization_3 (BatchNormalization)	(100, 8, 8, 128)	32
conv2d_3 (Conv2D)	(100, 8, 8, 128)	147584
max_pooling2d_2 (MaxPooling2D)	(100, 4, 4, 128)	0
dropout_2 (Dropout)	(100, 4, 4, 128)	0
batch_normalization_4 (BatchNormalization)	(100, 4, 4, 128)	16

```
conv2d_4 (Conv2D)           (100, 4, 4, 256)          295168

max_pooling2d_3 (MaxPooling  (100, 2, 2, 256)         0
2D)

dropout_3 (Dropout)         (100, 2, 2, 256)          0

flatten (Flatten)           (100, 1024)               0

batch_normalization_5 (Batc  (100, 1024)              4096
hNormalization)

dense (Dense)               (100, 128)                131200

dropout_4 (Dropout)         (100, 128)                0

dense_1 (Dense)             (100, 100)                12900

=================================================================
Total params: 684,564
Trainable params: 682,332
Non-trainable params: 2,232
_________________________________________________________________
```

Ο αριθμός των παραμέτρων με την πρώτη ματιά ίσως να ξενίζει, ωστόσο δεδομένου της συνθετότητας του προβλήματος και αναζητώντας δίκτυα στο ίντερνετ, οι παράμετροι είναι ικανοποιητικές για το ποσοστό ευστοχίας που τελικά επιτυγχάνεται.

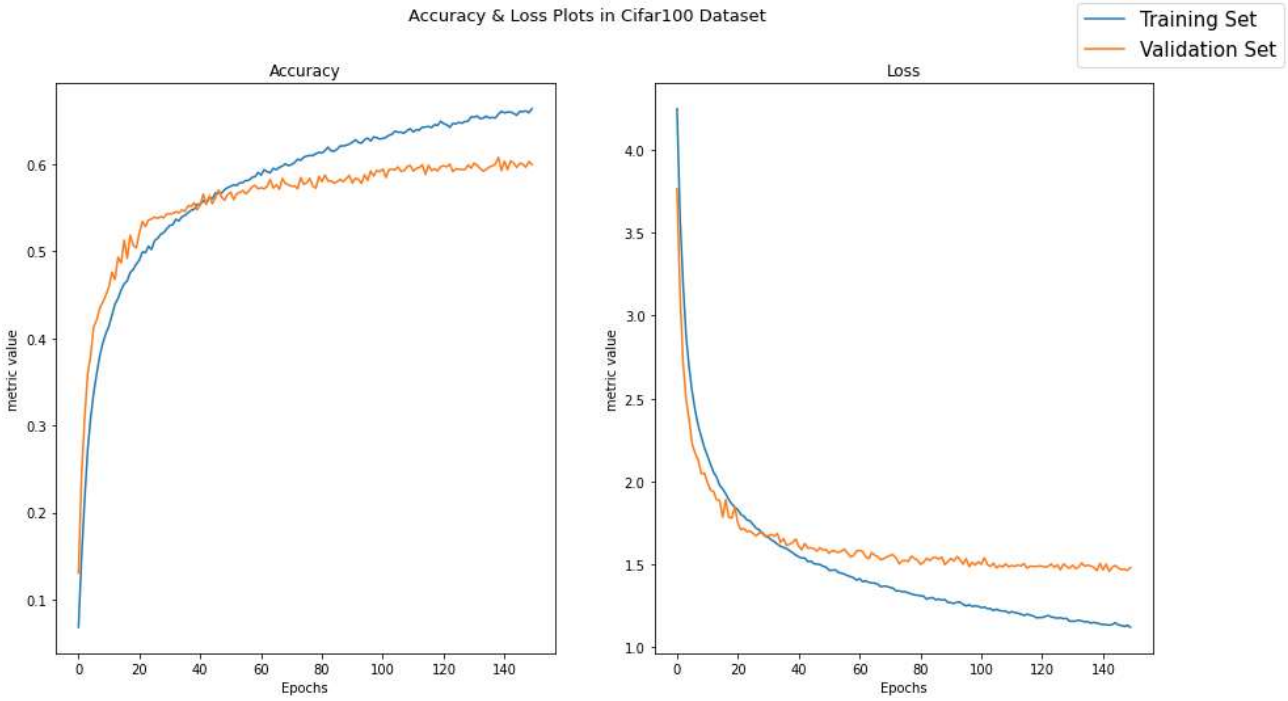
2. <u>Αποτελέσματα:</u>

   Αρχικά, οι μετρικές loss & accuracy για το test set φαίνονται παρακάτω:

```
Test loss: 1.4669256210327148
Test accuracy: 0.6014000177383423
```

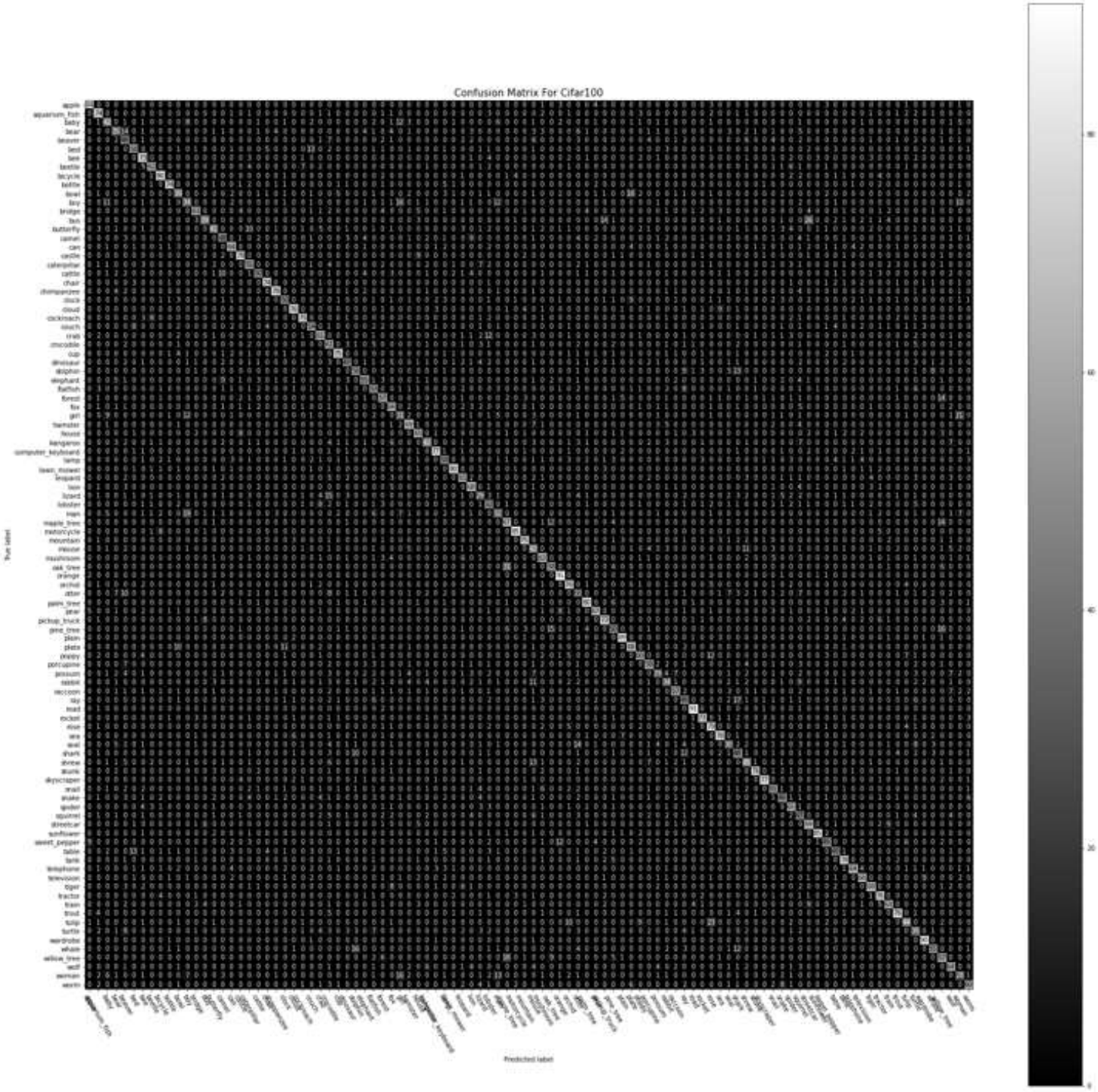
Εικόνα 4.1 Αποτελέσματα ταξινόμησης test set για το Cifar100

Όπου απεικονίζεται η σχετικά καλή απόδοση του δικτύου για το πλήθος των παραμέτρων που χρησιμοποιούνται. Στο επόμενο γράφημα, αποτυπώνονται τα losses & accuracies για τα training & validation set κατά την διάρκεια της εκπαίδευσης:



Εικόνα 4.2 Γραφήματα μετρικών loss,accuracy για κάθε set

Με βάση τα παραπάνω γραφήματα φαίνεται πως υπάρχει μια απόκλιση μεταξύ των μετρικών για το training & το validation set, γεγονός που πιθανόν να οφείλεται στο γεγονός ότι αρχίζει να εμφανίζεται το φαινόμενο του overfitting όσο αναφορά το training set. Έπειτα, ο confusion matrix που προκύπτει είναι ο ακόλουθος:



Εικόνα 4.3 Confusion Matrix για Cifar100

Όπου όσο πιο έντονο είναι το χρώμα (μαύρο δηλαδή) τόσο λιγότερες ταξινομήσεις έγιναν για τον εκάστοτε συνδυασμό κλάσεων, ενώ αντίθετα όσο πιο λευκό τόσο περισσότερες ταξινομήσεις έγιναν για τον χ συνδυασμό. Τέλος, χρησιμοποιώντας το δίκτυο για μερικές προβλέψεις, τα αποτελέσματα είναι τα εξής:

Εικόνα 4.4 Μερικές προβλέψεις χρησιμοποιώντας το δίκτυο που υλοποιήθηκε

Βιβλιογραφία

- https://medium.com/analytics-vidhya/how-to-choose-the-size-of-the-convolution-filter-or-kernel-size-for-cnn-86a55a1e2d15
- https://en.wikipedia.org/wiki/Convolutional_neural_network
- https://keras.io/api/
- https://www.tensorflow.org/api_docs/python/tf
- https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-i-hyper-parameter-8129009f131b
- https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7
- https://github.com/paulgavrikov/visualkeras/
- *Recent Advances in Convolutional Neural Networks by Jiuxiang Gua,_, Zhenhua Wangb,_, Jason Kuenb, Lianyang Mab, Amir Shahroudyb, Bing Shuaib, Ting Liub, Xingxing Wangb, Li Wangb, Gang Wangb, Jianfei Caic, Tsuhan Chenc*
- https://github.com/zalandoresearch/fashion-mnist
- https://www.cs.toronto.edu/~kriz/cifar.html