Νευρωνικά Δίκτυα - Βαθιά Μάθηση
1ο Παραδοτέο Εργασίας
Τουλκερίδης Νικόλαος
ΗΜΜΥ 10718

Εισαγωγή:

Θεματική της εργασίας είναι η υλοποίηση ενός νευρωνικού δικτύου εμπρόσθιας τροφοδότησης (feedforward NN) που θα εκπαιδεύεται με τον αλγόριθμο Back Propagation.
Για την εργασία χρησιμοποιήθηκε η βάση δεδομένων CIFAR-10, η οποία αποτελείται από 60.000 32x32 έγχρωμες εικόνες. Οι εικόνες είναι χωρισμένες σε 10 κλάσεις και η κάθε κλάση διαθέτει 6.000 εικόνες. Το dataset διακρίνεται σε 50.000 δεδομένα εκπαίδευσης (training set), τα οποία χωρίζονται σε **40.000 training images** και **10.000 validation** images, και σε **10.000 δεδομένα ελέγχου (test images)**.
Για την ανάπτυξη του νευρωνικού δικτύου εμπρόσθιας τροφοδότησης (feedforward Neural Network) επιλέχθηκε συνδυασμός συνελικτικού (CNN) και πλήρως συνδεδεμένου δικτύου (MLP). Για την εκπαίδευσή του έγιναν δοκιμές και συγκρίσεις στις παραμέτρους εκπαίδευσης και στα layers του νευρωνικού, οι οποίες περιγράφονται στη συνέχεια.

Η γλώσσα προγραμματισμού **python** και η βιβλιοθήκη **keras** επιλέχθηκαν για την υλοποίηση του νευρωνικού δικτύου . Άλλες βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι εξής:

- ➔ matplotlib, για την παρουσίαση διαγραμμάτων
- ➔ numpy, για τον εύκολο χειρισμό δεδομένων
- ➔ time, για τον υπολογισμό των χρόνων εκτέλεσης
- ➔ sklearn, για τον υπολογισμό μετρικών

1. Περιγραφή Αλγόριθμου

Ο αλγόριθμος Back Propagation (BK) είναι η μέθοδος που χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου και αποτελεί δημοφιλή μέθοδο εκπαίδευσης ενός NN γενικότερα. Η εκπαίδευση λαμβάνει χώρα σε δύο φάσεις:

- ➔ Στη φάση που εξελίσσεται προς τα εμπρός (forward pass), στην οποία τα συναπτικά βάρη του δικτύου είναι σταθερά και τα δεδομένα εισόδου διαδίδονται διαμέσου του δικτύου μέχρι να υπολογιστεί η έξοδος.
- ➔ Στη φάση που εξελίσσεται προς τα πίσω (back propagation), στην οποία παράγεται ένα σήμα σφάλματος, μέσω της σύγκρισης της εξόδου του δικτύου με μια επιθυμητή απόκριση και διαδίδεται προς τα πίσω, διαμέσου του δικτύου, με αποτέλεσμα να γίνονται διαδοχικές προσαρμογές στα συναπτικά βάρη.

Forward Pass:
Στο πέρασμα προς τα εμπρός του δικτύου, τα συναπτικά βάρη παραμένουν αμετάβλητα και τα λειτουργικά σήματα του δικτύου υπολογίζονται σε βάση νευρώνα προς νευρώνα. Το λειτουργικό σήμα που εμφανίζεται στην έξοδο κάθε νευρώνα υπολογίζεται με βάση τη συνάρτηση ενεργοποίησης του κάθε νευρώνα. Στην προκειμένη, χρησιμοποιήθηκε η ReLU (g(x)=max(0,x)) για τους κρυφούς νευρώνες (hidden layers) και η Softmax για τον νευρώνα εξόδου.

Back Propagation:
Το πέρασμα προς τα πίσω ξεκινά στο επίπεδο εξόδου, υπολογίζοντας το σήμα σφάλματος για κάθε νευρώνα αυτού του επιπέδου και στέλνοντάς το προς τα αριστερά. Στη συνέχεια υπολογίζονται αναδρομικά τα δ (τοπική κλίση) για κάθε νευρώνα και αυτή η διαδικασία επιτρέπει τις μεταβολές στα συναπτικά βάρη του δικτύου. Για έναν νευρώνα εξόδου το δ είναι απλά ίσο με το σήμα σφάλματος του νευρώνα αυτού πολλαπλασιασμένο με την πρώτη παράγωγο της μη γραμμικότητάς του (δηλ. της συνάρτησης ενεργοποίησης), ενώ για τον υπολογισμό των δ προηγούμενων επιπέδων χρησιμοποιείται ο τύπος οπισθοδιάδοσης της τοπικής κλίσης. Η διόρθωση των βαρών υπολογίζεται αναδρομικά με χρήση του κανόνα Δέλτα.

2. Επεξήγηση Κώδικα
Import necessary libraries and kits
Load and Preprocessing the cifar-10 dataset
Further Split the 50k training set to 40k training and 10k validating
Normalize Image Data from [0, 255] to [0, 1]
Define EarlyStopping (for stopping training before overfitting by measuring ‘val_loss’)
Define ImageDataGenerator (for data augmentation)
Define the Model
Compile the Model
Train the model with model.fit() and measure the training time with time()
Extract important metrics about training with history.history[…]
Test the model to the test set
Plot the metrics you history.history gave us
Loop that for 3 trainings (2 standard and 1 augmented)

3. Δοκιμές

Οι μετρήσεις γίνονται με σημείο αναφοράς το εξής απλό μοντέλο:

Input(32,32,3)

Convolutional layer με 16 φίλτρα, (3x3) κέρνελ, βήμα (1,1), padding 'valid' και activation relu:

Εντοπίζει μοτίβα σε μικρές περιοχές της φωτογραφίας, γωνίες, υφές.

- ➔ 16 μοτίβα μαθαίνει
- ➔ 3x3 το μέγεθος κάθε φίλτρου (οι μικρές περιοχές της φωτογραφίας)
- ➔ 'ReLU' κάνει την στρώση να συγκεντρώνεται σε θετικά μοτίβα
- ➔ (1,1) strides, το φίλτρο κινείται ένα pixel τη φορά και τρέχει όλη τη 32x32 εικόνα
- ➔ padding 'valid': δεν προστίθεται κάποιο padding, οπότε η εικόνα εξόδου είναι ελάχιστα μικρότερη (την μίκρυνε το φίλτρο)

MaxPooling layer (2x2):

Μειώνει το μέγεθος της εικόνας (donw-sampling) για να κάνει το μοντέλο ταχύτερο και να συγκεντρωθεί στα σημαντικά features

- ➔ (2x2): εξετάζει μία περιοχή 2x2 και κρατάει την ΜΑΞ τιμή αυτής.

Τα convolution and pooling layers παράγουν ένα 2D Feature Map.

Flatten():

Ισιώνει το Feature Map από 2D σε 1D, για να μπορέσει να γίνει είσοδος στο Fully Connected Layer. (part of the MLP)

Dense layer of 32 Neurons activation function relu:

Συνδυάζει όλα τα εξαγόμενα features για να μάθει τις σύνθετες σχέσεις μεταξύ αυτών.

- ➔ 32 ο αριθμός των Νευρώνων. Κάθε Νευρώνας είναι συνδεδεμένος με κάθε output value του Flatten layer
- ➔ 'ReLU' δημιουργεί τη μη-γραμμικότητα στο μοντέλο, επιτρέποντας το να μάθει πιο σύνθετα μοτίβα

Final Dense layer of 10 Neurons (10 classes), activation function softmax:

Εξάγει μία πιθανότητα για κάθε κλάση από τις 10.

- ➔ 'softmax' σιγουρεύει ότι οι εξαγόμενες τιμές συμβολίζουν πιθανότητες που αθροίζουν στο 1. Η κλάση με την μεγαλύτερη πιθανότητα, εξάγεται ως predicted label.

Με model.compile()

optimizer: Ο βελτιστοποιητής καθορίζει **πώς το μοντέλο θα προσαρμόσει τα weights του** κατά την εκπαίδευση, ώστε να μειωθεί το σφάλμα.

→ **'Adam'**: Χρησιμοποιείται ο Adam optimizer, ένας προηγμένος αλγόριθμος που συνδυάζει τα πλεονεκτήματα του SGD (Stochastic Gradient Descent) και του RMSprop

loss: Η συνάρτηση απώλειας μετράει **πόσο "λάθος" είναι η πρόβλεψη του μοντέλου** σε σχέση με τις πραγματικές τιμές.

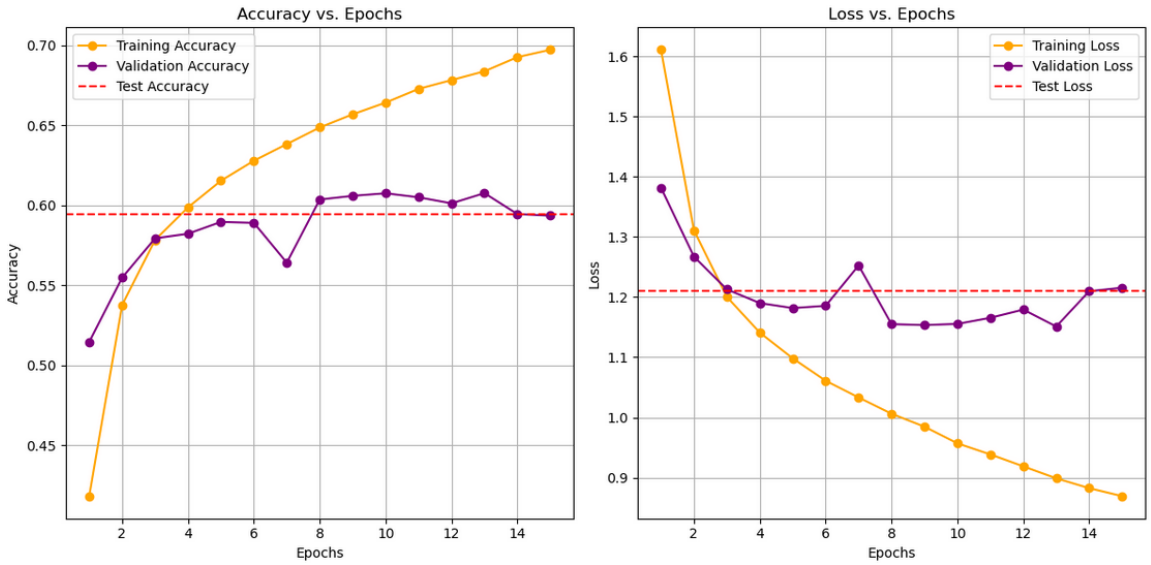
→ **'sparse_categorical_crossentropy'**: παίρνει ετικέτες ως ακέραιους αριθμούς, συγκρίνει την προβλεπόμενη πιθανότητα της σωστής κατηγορίας με την πραγματική, και υπολογίζει πόσο "λάθος" έκανε το δίκτυο. Στόχος είναι να μεγιστοποιηθεί η πιθανότητα της σωστής κατηγορίας.

116,010 trainable parameters

15 εποχές εκπαίδευσης: 259.18' '

Οι μετρήσεις γίνονται για 15 εποχές εκπαίδευσης και οι χρόνοι εκπαίδευσης θα δίνονται στις φωτογραφίες

```
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.3356 - loss: 1.8114 - val_accuracy: 0.5145 - val_loss: 1.3817
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 11s 9ms/step - accuracy: 0.5254 - loss: 1.3417 - val_accuracy: 0.5547 - val_loss: 1.2673
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.5745 - loss: 1.2078 - val_accuracy: 0.5793 - val_loss: 1.2128
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.5989 - loss: 1.1423 - val_accuracy: 0.5822 - val_loss: 1.1900
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6211 - loss: 1.0791 - val_accuracy: 0.5896 - val_loss: 1.1817
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6320 - loss: 1.0534 - val_accuracy: 0.5890 - val_loss: 1.1855
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6430 - loss: 1.0223 - val_accuracy: 0.5641 - val_loss: 1.2526
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6487 - loss: 1.0039 - val_accuracy: 0.6036 - val_loss: 1.1550
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6642 - loss: 0.9634 - val_accuracy: 0.6059 - val_loss: 1.1535
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.6690 - loss: 0.9455 - val_accuracy: 0.6075 - val_loss: 1.1554
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy: 0.6781 - loss: 0.9290 - val_accuracy: 0.6050 - val_loss: 1.1656
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6863 - loss: 0.8990 - val_accuracy: 0.6012 - val_loss: 1.1792
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.6895 - loss: 0.8847 - val_accuracy: 0.6076 - val_loss: 1.1511
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 15ms/step - accuracy: 0.6996 - loss: 0.8629 - val_accuracy: 0.5945 - val_loss: 1.2097
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6989 - loss: 0.8585 - val_accuracy: 0.5935 - val_loss: 1.2156
Training stopped at epoch 15
Total training time: 259.18 seconds
```

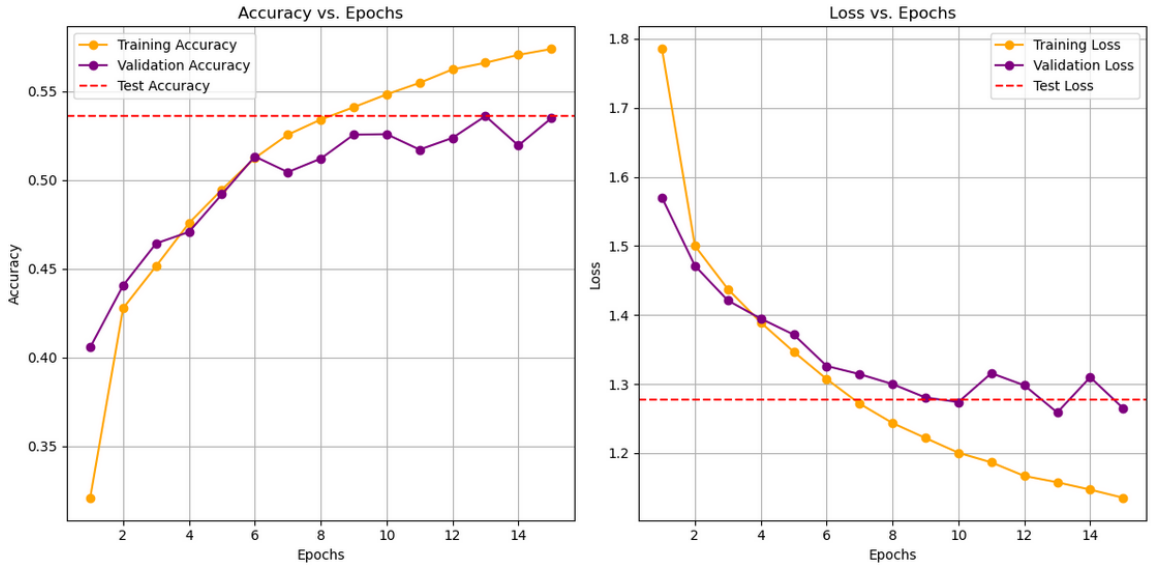


1η: Αύξηση του πλήθος των filters από 16 σε 32
231.658 παράμετροι

Τα **φίλτρα** είναι "μαθηματικά εργαλεία" που ανιχνεύουν features από τα δεδομένα, όπως άκρα, χρώματα ή μοτίβα σε εικόνες. Περνάει πάνω από την εικόνα (convolution) και "μαθαίνει" να εντοπίζει συγκεκριμένα χαρακτηριστικά. Στο πειραμά μας, αυξάνεται το υπολογιστικό κόστος (διπλάσιες παράμετροι) και η πολυπλοκότητα του μοντέλου. Θα προσπαθήσει το layer αυτό να αναγνωρίσει 16 μοτίβα παραπάνω από το αρχικό, αυξάνοντας έτσι τον κίνδυνο για απομνημόνευση των training images. Για 15 εποχές, το Conv2D με 16 φίλτρα **έτρεξε καλύτερα και στον ίδιο περίπου χρόνο**. Γρηγορότερη σύγκλιση (από την 6η εποχή κιόλας ξεκινάει το overfitting) και υψηλότερο validation-test accuracy. Δοκίμασα την προπόνηση του 2ου μοντέλου για 30 εποχές, αλλά τα

φαινόμενα είναι τα ίδια. Φαίνεται ότι το δίκτυο ‘προτιμάει’ τα 16 φίλτρα.

```
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 29s 22ms/step - accuracy: 0.2490 - loss: 1.9936 - val_accuracy: 0.4056 - val_loss: 1.5699
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 28s 22ms/step - accuracy: 0.4203 - loss: 1.5166 - val_accuracy: 0.4405 - val_loss: 1.4713
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 28s 22ms/step - accuracy: 0.4444 - loss: 1.4411 - val_accuracy: 0.4642 - val_loss: 1.4210
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 28s 22ms/step - accuracy: 0.4700 - loss: 1.3954 - val_accuracy: 0.4707 - val_loss: 1.3947
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 32s 15ms/step - accuracy: 0.4919 - loss: 1.3498 - val_accuracy: 0.4919 - val_loss: 1.3716
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5060 - loss: 1.3160 - val_accuracy: 0.5132 - val_loss: 1.3262
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5243 - loss: 1.2730 - val_accuracy: 0.5044 - val_loss: 1.3144
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5345 - loss: 1.2377 - val_accuracy: 0.5120 - val_loss: 1.3000
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5428 - loss: 1.2126 - val_accuracy: 0.5255 - val_loss: 1.2806
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5445 - loss: 1.2054 - val_accuracy: 0.5258 - val_loss: 1.2740
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 15s 12ms/step - accuracy: 0.5557 - loss: 1.1865 - val_accuracy: 0.5171 - val_loss: 1.3158
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5682 - loss: 1.1567 - val_accuracy: 0.5237 - val_loss: 1.2982
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5657 - loss: 1.1568 - val_accuracy: 0.5360 - val_loss: 1.2589
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5731 - loss: 1.1428 - val_accuracy: 0.5194 - val_loss: 1.3099
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 15s 12ms/step - accuracy: 0.5823 - loss: 1.1183 - val_accuracy: 0.5348 - val_loss: 1.2651
Training stopped at epoch 15
Total training time: 289.51 seconds
```

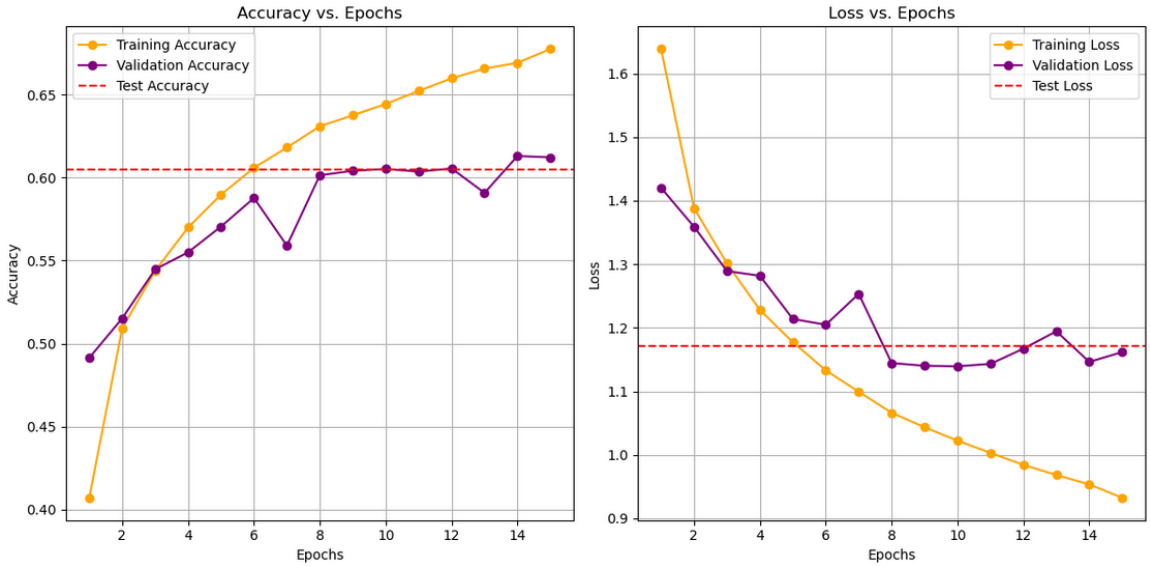


2η: Αύξηση του kernel size από (3x3) σε (7x7)
89.258 παράμετροι

Το **kernel size** σε ένα Conv2D είναι το μέγεθος του "παραθύρου" (ή φίλτρου) που "σαρώνει" την εικόνα. Το kernel εκτελεί υπολογισμούς πάνω σε αυτές τις περιοχές, για να εξάγει features από την εικόνα, όπως άκρα, γωνίες ή υφές.

Στο πείραμά μας, οι παράμετροι μειώνονται λόγω του μικρότερου εξαγόμενου Feature Map, άρα και το υπολογιστικό κόστος. Μεγαλύτερο kernel size ισοδυναμεί με προσπάθεια του μοντέλου για αναγνώριση πιο γενικών μοτίβων και απόρριψη λεπτομερειών. Σε μεγάλου μεγέθους εικόνες αυτό ίσως να βοηθούσε αλλά στην περίπτωση του cifar-10 (32x32x3), μπορεί να χάσει σημαντικές λεπτομέρειες. Ο χρόνος εκπαίδευσης 15 εποχών είναι 1.5 φορές μεγαλύτερος.

```
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 28s 22ms/step - accuracy: 0.3365 - loss: 1.8131 - val_accuracy: 0.4914 - val_loss: 1.4201
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.5000 - loss: 1.4131 - val_accuracy: 0.5152 - val_loss: 1.3589
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 24s 19ms/step - accuracy: 0.5441 - loss: 1.3008 - val_accuracy: 0.5447 - val_loss: 1.2892
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.5682 - loss: 1.2343 - val_accuracy: 0.5550 - val_loss: 1.2818
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.5952 - loss: 1.1583 - val_accuracy: 0.5705 - val_loss: 1.2138
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.6094 - loss: 1.1249 - val_accuracy: 0.5877 - val_loss: 1.2047
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.6242 - loss: 1.0863 - val_accuracy: 0.5590 - val_loss: 1.2533
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.6301 - loss: 1.0647 - val_accuracy: 0.6014 - val_loss: 1.1445
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.6394 - loss: 1.0346 - val_accuracy: 0.6041 - val_loss: 1.1402
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 24s 19ms/step - accuracy: 0.6436 - loss: 1.0155 - val_accuracy: 0.6052 - val_loss: 1.1394
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 28s 23ms/step - accuracy: 0.6559 - loss: 0.9922 - val_accuracy: 0.6036 - val_loss: 1.1432
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 29s 23ms/step - accuracy: 0.6658 - loss: 0.9718 - val_accuracy: 0.6055 - val_loss: 1.1671
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 29s 23ms/step - accuracy: 0.6731 - loss: 0.9472 - val_accuracy: 0.5908 - val_loss: 1.1944
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.6725 - loss: 0.9374 - val_accuracy: 0.6130 - val_loss: 1.1463
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6854 - loss: 0.9158 - val_accuracy: 0.6122 - val_loss: 1.1617
Training stopped at epoch 15
Total training time: 378.76 seconds
```

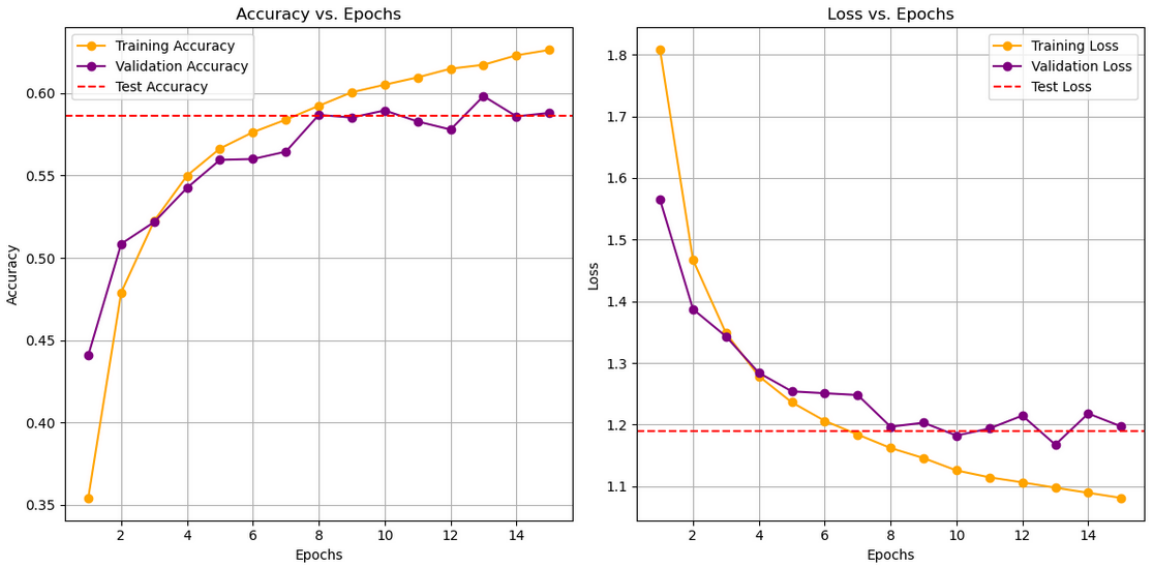


3η: Αύξηση του βήματος strides από (1,1) σε (2,2)
25.898 παράμετροι

Το **strides** είναι το "βήμα" με το οποίο το φίλτρο μετακινείται πάνω στα pixel της εικόνας.

Στο πείραμά μας, μειώνεται σίγουρα το υπολογιστικό κόστος λόγω των υποτετραπλάσιων παραμέτρων. Παρατηρούμε αργότερη σύγκλιση και όχι τόσο μεγάλη βελτίωση στην απόδοση του μοντέλου. Ο χρόνος των 15 εποχών είναι αισθητά μικρότερος.

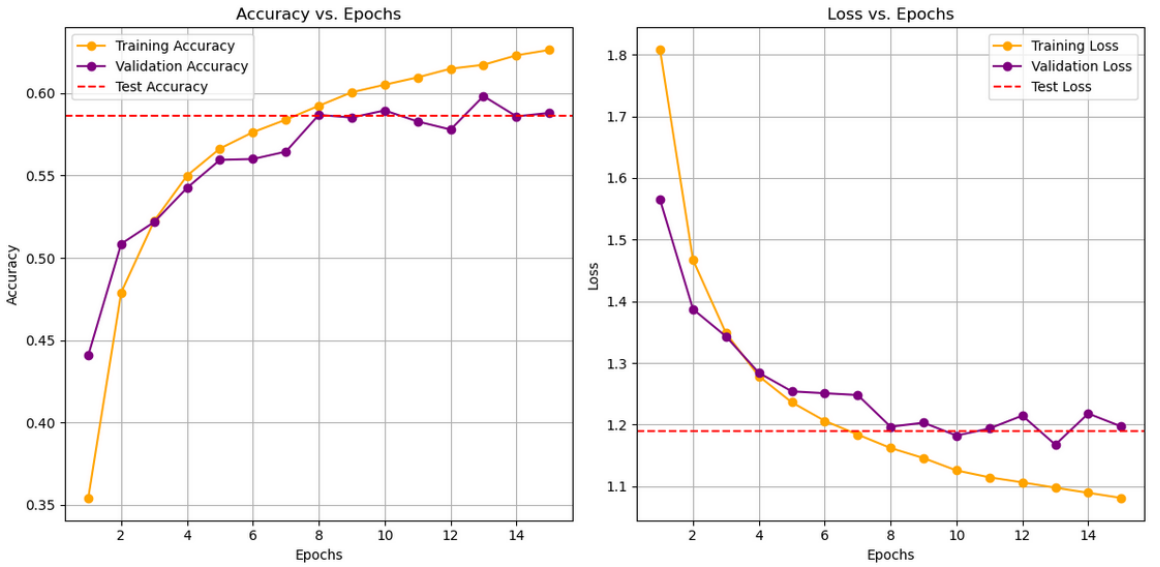
```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.2788 - loss: 1.9975 - val_accuracy: 0.4408 - val_loss: 1.5645
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.4676 - loss: 1.5040 - val_accuracy: 0.5085 - val_loss: 1.3872
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.5145 - loss: 1.3695 - val_accuracy: 0.5217 - val_loss: 1.3432
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5481 - loss: 1.2900 - val_accuracy: 0.5425 - val_loss: 1.2841
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.5669 - loss: 1.2344 - val_accuracy: 0.5595 - val_loss: 1.2540
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.5790 - loss: 1.2064 - val_accuracy: 0.5600 - val_loss: 1.2510
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.5811 - loss: 1.1900 - val_accuracy: 0.5645 - val_loss: 1.2481
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.5928 - loss: 1.1692 - val_accuracy: 0.5868 - val_loss: 1.1967
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6009 - loss: 1.1409 - val_accuracy: 0.5852 - val_loss: 1.2032
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6078 - loss: 1.1236 - val_accuracy: 0.5895 - val_loss: 1.1824
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6103 - loss: 1.1163 - val_accuracy: 0.5828 - val_loss: 1.1939
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.6195 - loss: 1.1005 - val_accuracy: 0.5779 - val_loss: 1.2148
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6164 - loss: 1.0953 - val_accuracy: 0.5982 - val_loss: 1.1676
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6278 - loss: 1.0783 - val_accuracy: 0.5859 - val_loss: 1.2181
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.6266 - loss: 1.0827 - val_accuracy: 0.5880 - val_loss: 1.1971
Training stopped at epoch 15
Total training time: 70.30 seconds
```



4η: Αλλαγή του padding από ‘valid’ σε ‘same’
131.882 παράμετροι

Το **padding** είναι ο τρόπος με τον οποίο χειρίζεται το μοντέλο τις άκρες της εικόνας όταν εφαρμόζει τα φίλτρα. Στο **Valid padding** δεν προστίθεται padding (δηλαδή δεν επεκτείνεται η εικόνα). Το φίλτρο δεν "βγαίνει έξω" από τα όρια της εικόνας και η έξοδος (feature map) έχει **μικρότερες διαστάσεις** σε σχέση με την αρχική εικόνα. Στο **Same padding** προστίθεται padding (συνήθως μηδενικά) γύρω από την εικόνα. Το φίλτρο μπορεί να καλύπτει και τα άκρα της εικόνας. Η έξοδος (feature map) έχει τις **ίδιες διαστάσεις** με την αρχική εικόνα (εξού και "same"). Υπολογιστικά πιο ακριβό. Τελούνται και περισσότεροι υπολογισμοί διότι το μοντέλο επεξεργάζεται και τις άκρες μιας εικόνας. Με valid padding χάνεται η πληροφορία στις άκρες, αλλά το δίκτυο καθίσταται πιο ακριβές. Στο πείραμά μας παρατηρείται μόνο πιο αργή σύγκλιση.

```
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 9ms/step - accuracy: 0.2589 - loss: 2.0065 - val_accuracy: 0.4882 - val_loss: 1.4262
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 9ms/step - accuracy: 0.5063 - loss: 1.3662 - val_accuracy: 0.5268 - val_loss: 1.3365
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.5626 - loss: 1.2414 - val_accuracy: 0.5775 - val_loss: 1.2101
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.5923 - loss: 1.1563 - val_accuracy: 0.5816 - val_loss: 1.1889
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.6120 - loss: 1.0990 - val_accuracy: 0.5947 - val_loss: 1.1550
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.6384 - loss: 1.0356 - val_accuracy: 0.6073 - val_loss: 1.1269
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 11s 8ms/step - accuracy: 0.6510 - loss: 0.9921 - val_accuracy: 0.6024 - val_loss: 1.1457
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.6528 - loss: 0.9814 - val_accuracy: 0.6064 - val_loss: 1.1224
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 11s 9ms/step - accuracy: 0.6729 - loss: 0.9445 - val_accuracy: 0.6109 - val_loss: 1.1367
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.6755 - loss: 0.9163 - val_accuracy: 0.6146 - val_loss: 1.1517
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.6829 - loss: 0.8994 - val_accuracy: 0.6157 - val_loss: 1.1336
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.6929 - loss: 0.8773 - val_accuracy: 0.6113 - val_loss: 1.1541
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.7024 - loss: 0.8539 - val_accuracy: 0.5870 - val_loss: 1.1980
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.7083 - loss: 0.8368 - val_accuracy: 0.6188 - val_loss: 1.1398
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.7110 - loss: 0.8270 - val_accuracy: 0.6085 - val_loss: 1.1646
Training stopped at epoch 15
Total training time: 156.88 seconds
```

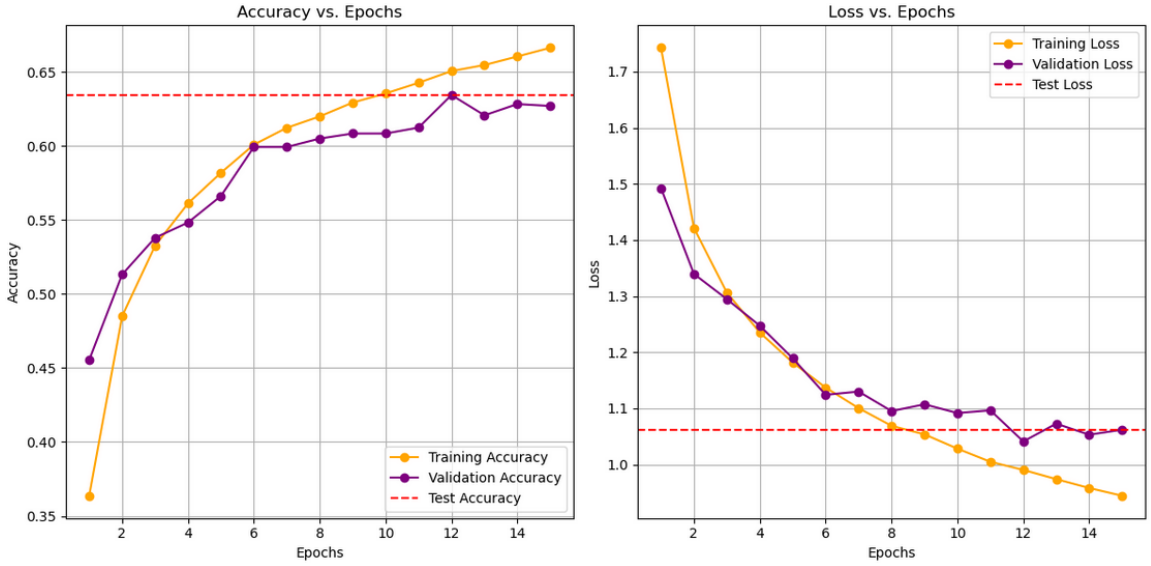


5η: Εισαγωγή 2ου Convolutional Layer με 32 φίλτρα: 47.370 παράμετροι

Με την εισαγωγή 2ου layer ενισχύεται η ικανότητα του δικτύου να κατανοεί σύνθετα μοτίβα, αλλά αυξάνεται ο κίνδυνος υπερπροσαρμογής αν δεν γίνει σωστή διαχείριση. Για περισσότερες εποχές, θα επιτυγχάναμε καλύτερα νούμερα στο performance του νέου μοντέλου.

Παρατηρούμε εμφανή βελτίωση του μοντέλου μας μετά τη συγκεκριμένη αλλαγή, αν και διπλασιασμό του χρόνου εκπαίδευσης 15 εποχών. Επιπλέον, οι παράμετροι είναι αρκετά λιγότεροι, μειώνοντας έτσι το υπολογιστικό κόστος.

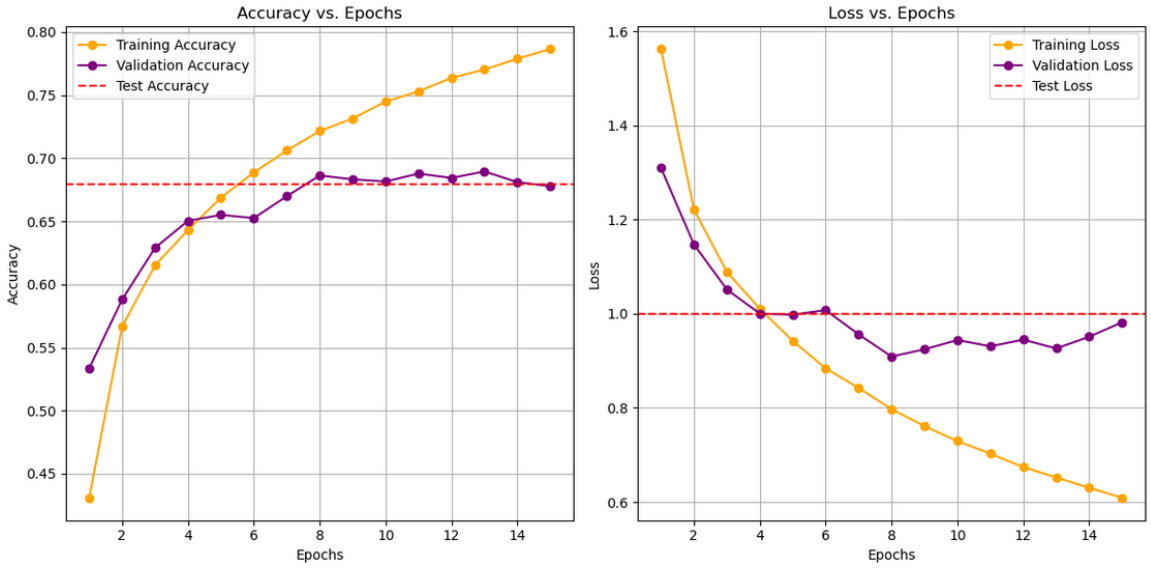
```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 46s 35ms/step - accuracy: 0.2767 - loss: 1.9623 - val_accuracy: 0.4554 - val_loss: 1.4925
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 42s 33ms/step - accuracy: 0.4738 - loss: 1.4550 - val_accuracy: 0.5131 - val_loss: 1.3395
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 40s 32ms/step - accuracy: 0.5243 - loss: 1.3306 - val_accuracy: 0.5379 - val_loss: 1.2947
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 39s 31ms/step - accuracy: 0.5594 - loss: 1.2439 - val_accuracy: 0.5482 - val_loss: 1.2472
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 39s 31ms/step - accuracy: 0.5773 - loss: 1.1939 - val_accuracy: 0.5661 - val_loss: 1.1892
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 40s 32ms/step - accuracy: 0.5987 - loss: 1.1400 - val_accuracy: 0.5992 - val_loss: 1.1241
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 41s 33ms/step - accuracy: 0.6149 - loss: 1.0963 - val_accuracy: 0.5992 - val_loss: 1.1299
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 42s 33ms/step - accuracy: 0.6233 - loss: 1.0611 - val_accuracy: 0.6049 - val_loss: 1.0951
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 40s 32ms/step - accuracy: 0.6284 - loss: 1.0521 - val_accuracy: 0.6083 - val_loss: 1.1072
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 41s 33ms/step - accuracy: 0.6393 - loss: 1.0169 - val_accuracy: 0.6083 - val_loss: 1.0916
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 42s 33ms/step - accuracy: 0.6452 - loss: 0.9995 - val_accuracy: 0.6123 - val_loss: 1.0965
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy: 0.6549 - loss: 0.9760 - val_accuracy: 0.6343 - val_loss: 1.0410
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy: 0.6594 - loss: 0.9625 - val_accuracy: 0.6206 - val_loss: 1.0727
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy: 0.6597 - loss: 0.9575 - val_accuracy: 0.6282 - val_loss: 1.0530
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.6684 - loss: 0.9350 - val_accuracy: 0.6269 - val_loss: 1.0619
Training stopped at epoch 15
Total training time: 539.51 seconds
```



6: Εισαγωγή 2ου Convolutional Layer με 64 φίλτρα: 93.482 παράμετροι

Αν το 2ο Convolutional Layer περιέχει 64 φίλτρα τότε μπορεί να διπλασιάζονται οι παράμετροι εκπαίδευσης, αλλά βελτιώνεται περαιτέρω το μοντέλο μας. Ο χρόνος εκπαίδευσης αυξάνεται κατά 1 λεπτό. Προτιμήθηκαν τα 64 φίλτρα.

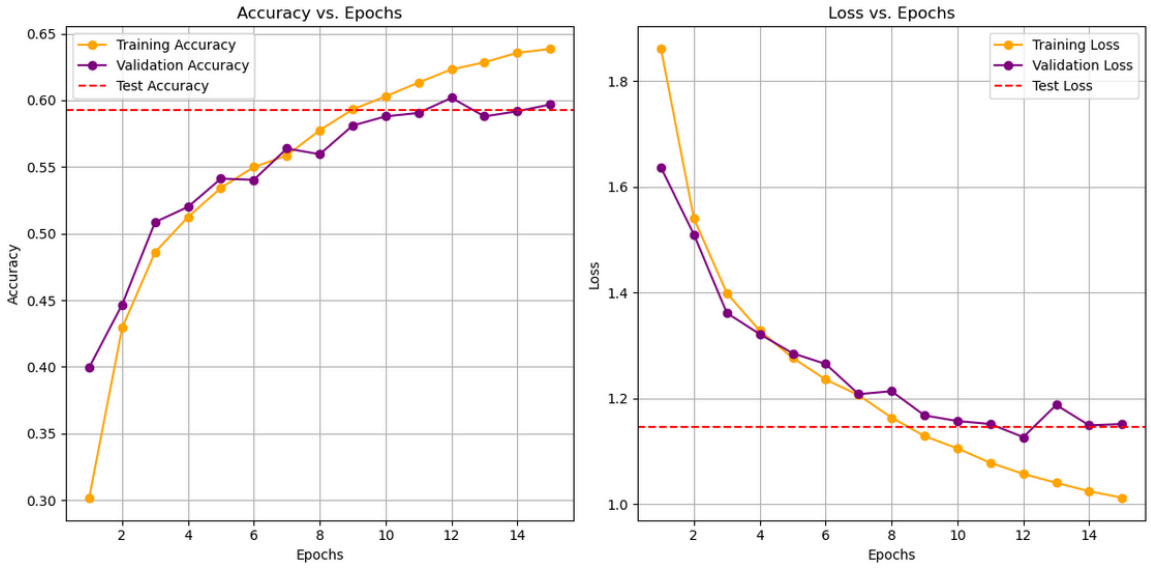
```
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 26s 20ms/step - accuracy: 0.3357 - loss: 1.7964 - val_accuracy: 0.5335 - val_loss: 1.3096
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 24s 19ms/step - accuracy: 0.5566 - loss: 1.2544 - val_accuracy: 0.5880 - val_loss: 1.1472
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 24s 19ms/step - accuracy: 0.6088 - loss: 1.1086 - val_accuracy: 0.6290 - val_loss: 1.0514
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 24s 19ms/step - accuracy: 0.6428 - loss: 1.0082 - val_accuracy: 0.6504 - val_loss: 1.0002
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 41s 19ms/step - accuracy: 0.6689 - loss: 0.9410 - val_accuracy: 0.6551 - val_loss: 0.9979
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 48s 38ms/step - accuracy: 0.6927 - loss: 0.8831 - val_accuracy: 0.6526 - val_loss: 1.0076
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 46s 37ms/step - accuracy: 0.7067 - loss: 0.8396 - val_accuracy: 0.6699 - val_loss: 0.9561
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 44s 35ms/step - accuracy: 0.7245 - loss: 0.7906 - val_accuracy: 0.6863 - val_loss: 0.9090
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 44s 35ms/step - accuracy: 0.7309 - loss: 0.7611 - val_accuracy: 0.6833 - val_loss: 0.9244
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 45s 36ms/step - accuracy: 0.7498 - loss: 0.7096 - val_accuracy: 0.6816 - val_loss: 0.9441
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 45s 36ms/step - accuracy: 0.7604 - loss: 0.6837 - val_accuracy: 0.6878 - val_loss: 0.9309
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 48s 38ms/step - accuracy: 0.7704 - loss: 0.6564 - val_accuracy: 0.6844 - val_loss: 0.9448
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 46s 37ms/step - accuracy: 0.7746 - loss: 0.6379 - val_accuracy: 0.6895 - val_loss: 0.9262
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 44s 36ms/step - accuracy: 0.7842 - loss: 0.6133 - val_accuracy: 0.6811 - val_loss: 0.9511
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 46s 37ms/step - accuracy: 0.7940 - loss: 0.5868 - val_accuracy: 0.6778 - val_loss: 0.9818
Training stopped at epoch 15
Total training time: 594.93 seconds
```



7η: Αύξηση MaxPooling από (2,2) σε (3,3)
103.658 παράμετροι

Το **(3,3)** pooling οδηγεί σε πιο επιθετική μείωση της ανάλυσης, που μπορεί να επιταχύνει την εκπαίδευση κατά 1 λεπτό, αλλά ενδέχεται να μειώσει την ακρίβεια λόγω απώλειας λεπτομερειών. Συγκλίνει πιο αργά. Προτιμήθηκε το (2,2).

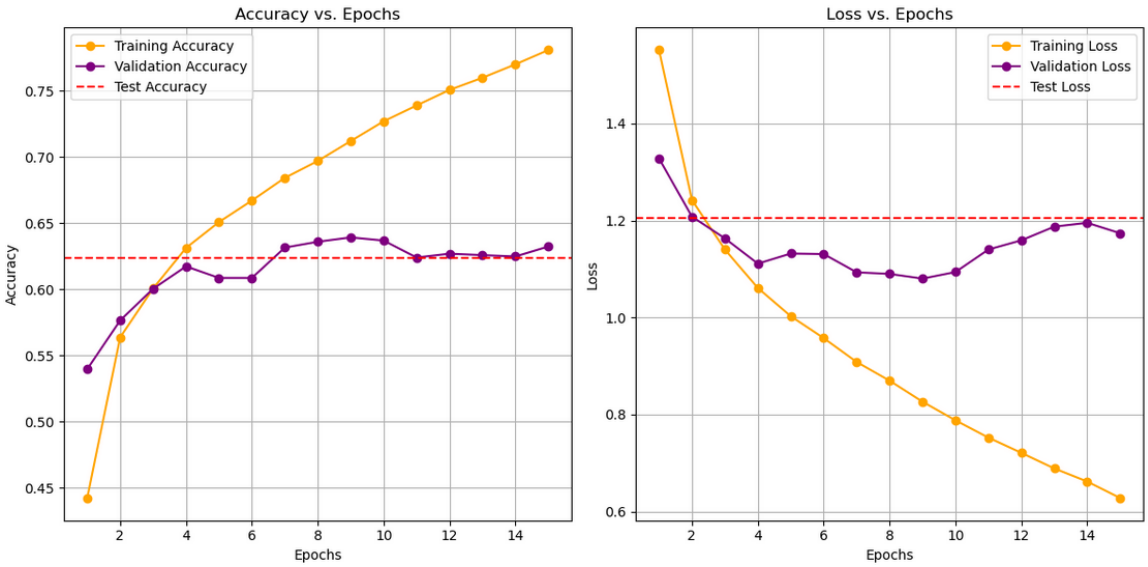
```
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.2315 - loss: 2.0333 - val_accuracy: 0.3996 - val_loss: 1.6364
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.4151 - loss: 1.5897 - val_accuracy: 0.4467 - val_loss: 1.5098
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 13s 10ms/step - accuracy: 0.4749 - loss: 1.4223 - val_accuracy: 0.5086 - val_loss: 1.3616
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.5097 - loss: 1.3314 - val_accuracy: 0.5200 - val_loss: 1.3216
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 14s 11ms/step - accuracy: 0.5360 - loss: 1.2721 - val_accuracy: 0.5412 - val_loss: 1.2855
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 13s 11ms/step - accuracy: 0.5530 - loss: 1.2357 - val_accuracy: 0.5403 - val_loss: 1.2654
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.5532 - loss: 1.2233 - val_accuracy: 0.5638 - val_loss: 1.2083
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.5809 - loss: 1.1577 - val_accuracy: 0.5595 - val_loss: 1.2139
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.5912 - loss: 1.1351 - val_accuracy: 0.5810 - val_loss: 1.1681
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 13s 11ms/step - accuracy: 0.6028 - loss: 1.0984 - val_accuracy: 0.5879 - val_loss: 1.1575
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 13s 10ms/step - accuracy: 0.6174 - loss: 1.0734 - val_accuracy: 0.5905 - val_loss: 1.1519
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.6233 - loss: 1.0533 - val_accuracy: 0.6018 - val_loss: 1.1271
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.6320 - loss: 1.0364 - val_accuracy: 0.5879 - val_loss: 1.1880
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 9ms/step - accuracy: 0.6380 - loss: 1.0190 - val_accuracy: 0.5917 - val_loss: 1.1493
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.6408 - loss: 1.0146 - val_accuracy: 0.5968 - val_loss: 1.1519
Training stopped at epoch 15
Total training time: 192.33 seconds
```



8η: Αύξηση Νευρώνων στο Hidden Layer Dense(64)
462.410 παράμετροι

Η αύξηση από 32 σε 64 νευρώνες ενισχύει την εκφραστικότητα του μοντέλου. Το δίκτυο μπορεί να μάθει πιο σύνθετους συνδυασμούς χαρακτηριστικών, αυξάνοντας τη χωρητικότητά του. Έχει τετραπλάσιες σε αριθμό παραμέτρους αλλά έτσι αυξάνεται ο αριθμός των συνδέσεων, οδηγώντας σε περισσότερη μνήμη και υπολογιστική κατανάλωση. Προφανώς το υπολογιστικό κόστος και ο χρόνος θα είναι αρκετά μεγαλύτεροι. Η ταχύτητα στη σύγκλιση είναι η ίδια (6 εποχές). Προτιμήθηκε η αλλαγή του αρχικού μας.

```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 48s 37ms/step - accuracy: 0.3598 - loss: 1.7587 - val_accuracy: 0.5397 - val_loss: 1.3281
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 32s 26ms/step - accuracy: 0.5551 - loss: 1.2622 - val_accuracy: 0.5765 - val_loss: 1.2081
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 45s 36ms/step - accuracy: 0.6009 - loss: 1.1418 - val_accuracy: 0.6004 - val_loss: 1.1633
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 39s 31ms/step - accuracy: 0.6355 - loss: 1.0536 - val_accuracy: 0.6173 - val_loss: 1.1112
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 46s 35ms/step - accuracy: 0.6585 - loss: 0.9868 - val_accuracy: 0.6086 - val_loss: 1.1320
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 30s 24ms/step - accuracy: 0.6708 - loss: 0.9481 - val_accuracy: 0.6086 - val_loss: 1.1309
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 47s 37ms/step - accuracy: 0.6850 - loss: 0.9053 - val_accuracy: 0.6315 - val_loss: 1.0933
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 42s 34ms/step - accuracy: 0.7044 - loss: 0.8529 - val_accuracy: 0.6359 - val_loss: 1.0899
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 45s 36ms/step - accuracy: 0.7183 - loss: 0.8114 - val_accuracy: 0.6392 - val_loss: 1.0803
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 56s 15ms/step - accuracy: 0.7351 - loss: 0.7680 - val_accuracy: 0.6368 - val_loss: 1.0938
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 26s 21ms/step - accuracy: 0.7487 - loss: 0.7315 - val_accuracy: 0.6240 - val_loss: 1.1403
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 27s 22ms/step - accuracy: 0.7556 - loss: 0.7098 - val_accuracy: 0.6269 - val_loss: 1.1593
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy: 0.7712 - loss: 0.6622 - val_accuracy: 0.6258 - val_loss: 1.1875
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy: 0.7725 - loss: 0.6501 - val_accuracy: 0.6248 - val_loss: 1.1954
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy: 0.7881 - loss: 0.6042 - val_accuracy: 0.6323 - val_loss: 1.1739
Training stopped at epoch 15
Total training time: 548.57 seconds
```



9η: Εισαγωγή BatchNormalization()
231.722 παράμετροι
→ Η Batch Normalization είναι μια τεχνική που επιταχύνει και σταθεροποιεί την εκπαίδευση βαθιών νευρωνικών δικτύων. Χρησιμοποιείται για να εξασφαλίσει ότι οι έξοδοι ενός επιπέδου έχουν μέσο όρο 0 και τυπική απόκλιση 1, και στη συνέχεια αυτές οι έξοδοι κλιμακώνονται και μετατοπίζονται μέσω παραμέτρων που μαθαίνονται κατά την εκπαίδευση

Σταθεροποιεί την Εκπαίδευση:
Εξισορροπεί τις εξόδους μεταξύ επιπέδων, μειώνοντας τον κίνδυνο να "εκραγούν" ή να "εξαφανιστούν" τα βάρη (exploding/vanishing gradients).

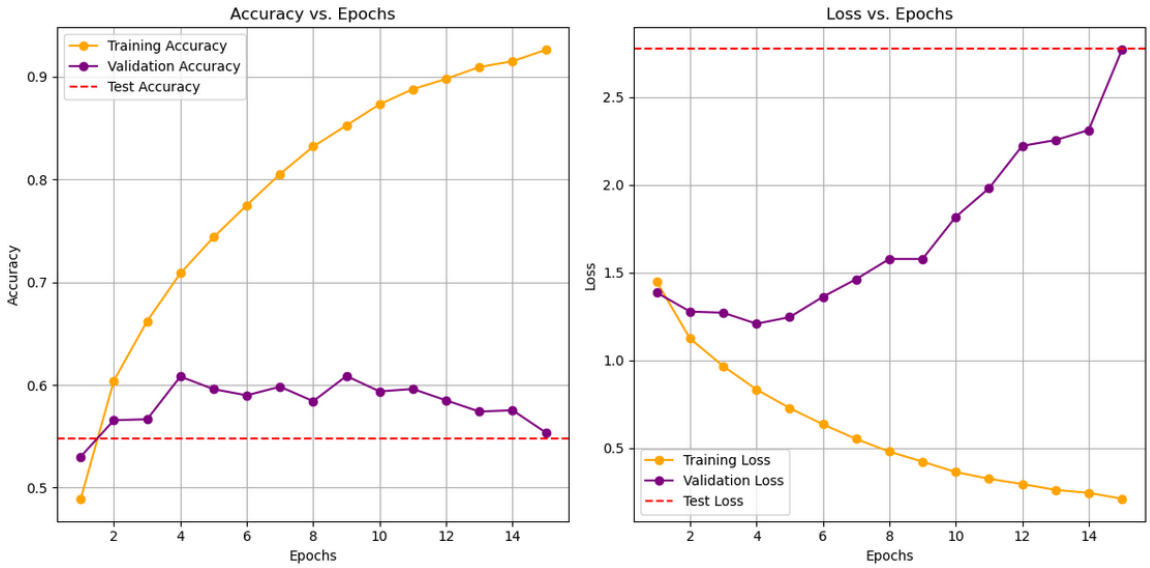
Επιταχύνει τη Σύγκλιση:
Διατηρεί τις ενεργοποιήσεις σε ένα εύρος τιμών που επιταχύνουν την εκπαίδευση. Επιτρέπει τη χρήση υψηλότερων ρυθμών εκμάθησης χωρίς αποσταθεροποίηση του μοντέλου.

Μειώνει την Εξάρτηση από Αρχικές Τιμές:
Μειώνει την ανάγκη για πολύ προσεκτική επιλογή αρχικών τιμών για τα βάρη.

Προσφέρει Κάποια Μορφή Κανονικοποίησης:
Δημιουργεί έναν μικρό θόρυβο κατά την εκπαίδευση λόγω της χρήσης στατιστικών από την παρτίδα, λειτουργώντας σαν μορφή regularization.

Στο πείραμα μας, εμφανώς αυξάνεται ο χρόνος εκπαίδευσης. Ήδη από την 4η εποχή συγκλίνει και μετά παρατηρείται το φαινόμενο της υπερπροσαρμογής. Η Batch Normalization μόνη της, αποτυγχάνει να μας βελτιώσει το μοντέλο.

```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 35s 28ms/step - accuracy: 0.4215 - loss: 1.6494 - val_accuracy: 0.5302 - val_loss: 1.3843
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 33s 26ms/step - accuracy: 0.6042 - loss: 1.1261 - val_accuracy: 0.5657 - val_loss: 1.2765
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 35s 28ms/step - accuracy: 0.6713 - loss: 0.9410 - val_accuracy: 0.5666 - val_loss: 1.2700
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 38s 30ms/step - accuracy: 0.7166 - loss: 0.8123 - val_accuracy: 0.6082 - val_loss: 1.2072
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 32s 26ms/step - accuracy: 0.7549 - loss: 0.6886 - val_accuracy: 0.5959 - val_loss: 1.2449
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 32s 25ms/step - accuracy: 0.7922 - loss: 0.5905 - val_accuracy: 0.5899 - val_loss: 1.3607
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 34s 27ms/step - accuracy: 0.8147 - loss: 0.5237 - val_accuracy: 0.5983 - val_loss: 1.4613
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 33s 27ms/step - accuracy: 0.8434 - loss: 0.4490 - val_accuracy: 0.5842 - val_loss: 1.5766
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 33s 27ms/step - accuracy: 0.8658 - loss: 0.3889 - val_accuracy: 0.6087 - val_loss: 1.5765
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 68s 55ms/step - accuracy: 0.8878 - loss: 0.3246 - val_accuracy: 0.5937 - val_loss: 1.8166
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 86s 58ms/step - accuracy: 0.8984 - loss: 0.2951 - val_accuracy: 0.5961 - val_loss: 1.9795
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 69s 55ms/step - accuracy: 0.9086 - loss: 0.2637 - val_accuracy: 0.5850 - val_loss: 2.2210
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 89s 61ms/step - accuracy: 0.9217 - loss: 0.2290 - val_accuracy: 0.5741 - val_loss: 2.2537
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 71s 56ms/step - accuracy: 0.9245 - loss: 0.2188 - val_accuracy: 0.5754 - val_loss: 2.3112
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 32s 26ms/step - accuracy: 0.9326 - loss: 0.1947 - val_accuracy: 0.5537 - val_loss: 2.7680
Training stopped at epoch 15
Total training time: 720.59 seconds
```

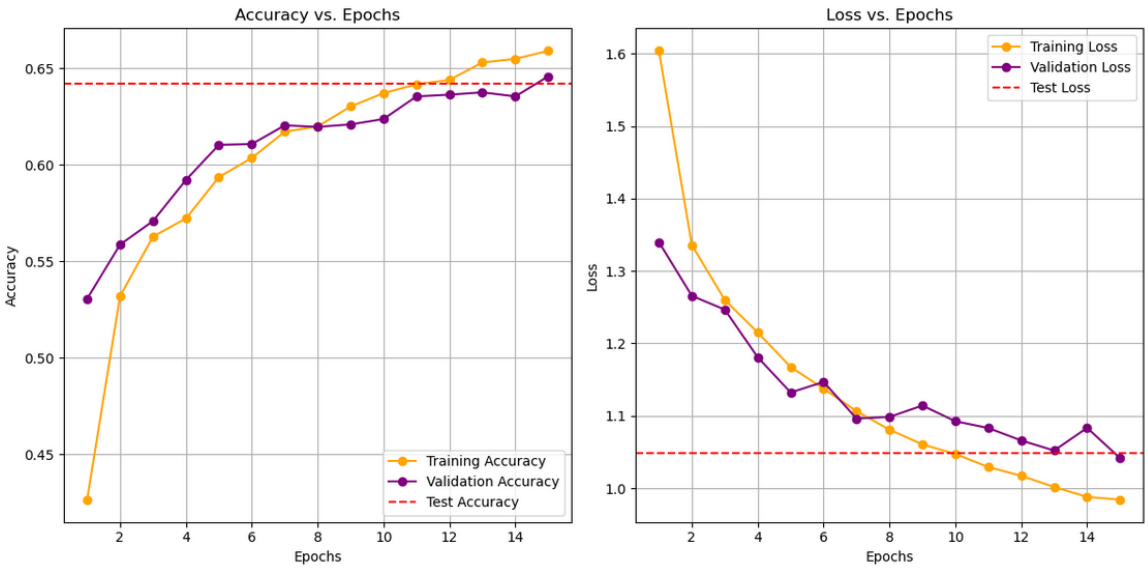


10η: Αφού απέτυχε η Batch Normalization, βάζω dropout(0.5), *to prevent the model from Overfitting*

231.658 παράμετροι

Τυχαία "απενεργοποιεί" το **50% των νευρώνων** κατά την εκπαίδευση, ώστε το δίκτυο να μην εξαρτάται υπερβολικά από συγκεκριμένους νευρώνες. Βοηθά στη μείωση του κινδύνου υπερπροσαρμογής. Παρατηρείται μεγάλη μείωση του χρόνου εκπαίδευσης, ίσου πλέον με τον αρχικό, και βελτίωση της απόδοσης του μοντέλου. Προτιμήθηκε ο συνδυασμός αυτών των δύο τεχνικών.

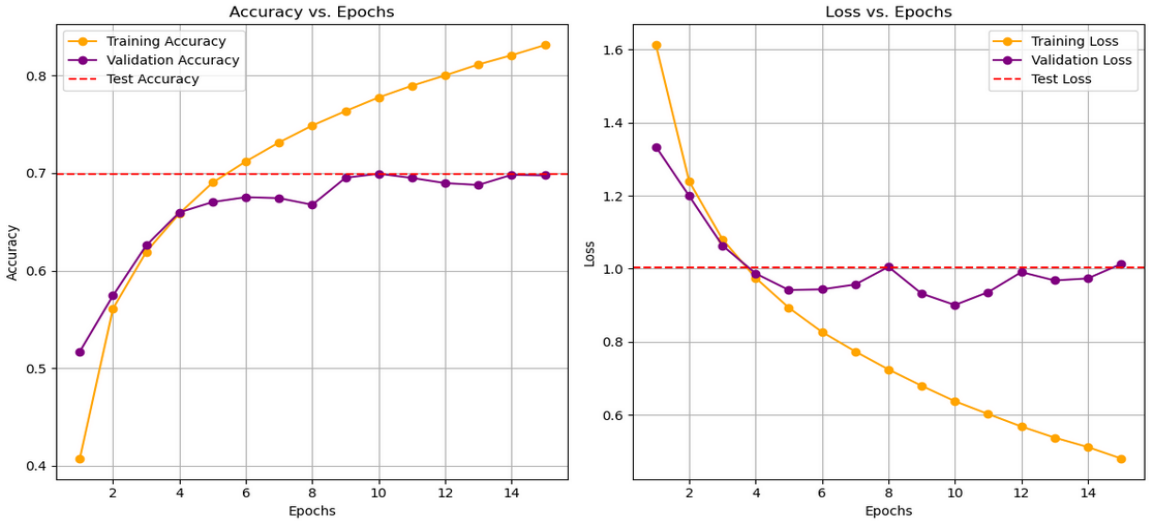
```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 22s 15ms/step - accuracy: 0.3160 - loss: 1.8617 - val_accuracy: 0.5123 - val_loss: 1.3985
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy: 0.5155 - loss: 1.3691 - val_accuracy: 0.5530 - val_loss: 1.2703
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy: 0.5507 - loss: 1.2810 - val_accuracy: 0.5664 - val_loss: 1.2456
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 13ms/step - accuracy: 0.5704 - loss: 1.2313 - val_accuracy: 0.5929 - val_loss: 1.1785
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.5809 - loss: 1.1950 - val_accuracy: 0.5947 - val_loss: 1.1859
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.5887 - loss: 1.1817 - val_accuracy: 0.6157 - val_loss: 1.1286
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 15ms/step - accuracy: 0.5990 - loss: 1.1445 - val_accuracy: 0.6053 - val_loss: 1.1372
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.6095 - loss: 1.1244 - val_accuracy: 0.6192 - val_loss: 1.1007
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6193 - loss: 1.0874 - val_accuracy: 0.6305 - val_loss: 1.0836
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy: 0.6239 - loss: 1.0894 - val_accuracy: 0.6149 - val_loss: 1.1195
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.6316 - loss: 1.0559 - val_accuracy: 0.6182 - val_loss: 1.1050
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy: 0.6323 - loss: 1.0536 - val_accuracy: 0.6412 - val_loss: 1.0458
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6435 - loss: 1.0281 - val_accuracy: 0.6442 - val_loss: 1.0500
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 18s 14ms/step - accuracy: 0.6463 - loss: 1.0170 - val_accuracy: 0.6395 - val_loss: 1.0532
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy: 0.6537 - loss: 0.9989 - val_accuracy: 0.6363 - val_loss: 1.0510
Training stopped at epoch 15
Total training time: 273.46 seconds
```



11η: Προσθήκη 3ου Convolutional Layer με 64 φίλτρα 75.114 παράμετροι

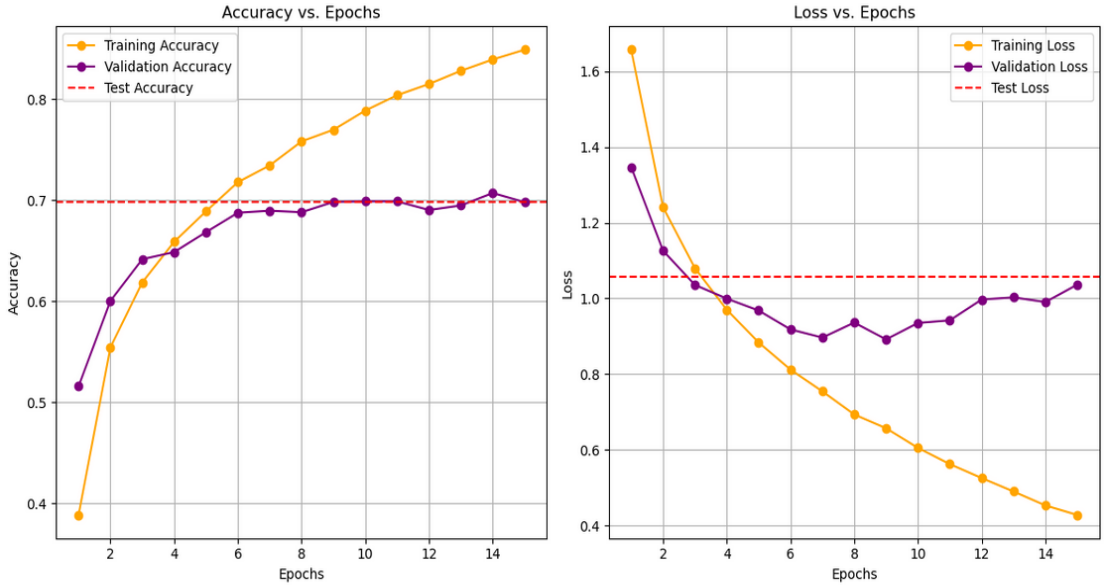
Η ανάλυση έγινε παραπάνω. Προτιμήθηκαν 3 Convolutional Layers, μετά από πειραματισμό και μερικές συμβουλές από ChatGPT. Ο χρόνος εκπαίδευσης μπορεί να είναι πενταπλάσιος, καταφέρνει όμως 0.70 accuracy.

```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 63s 48ms/step - accuracy: 0.3180 - loss: 1.8287 - val_accuracy: 0.5171 - val_loss: 1.3340
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 83s 66ms/step - accuracy: 0.5425 - loss: 1.2815 - val_accuracy: 0.5745 - val_loss: 1.1992
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 88s 70ms/step - accuracy: 0.6080 - loss: 1.1113 - val_accuracy: 0.6259 - val_loss: 1.0636
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 88s 71ms/step - accuracy: 0.6551 - loss: 0.9882 - val_accuracy: 0.6598 - val_loss: 0.9869
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 65ms/step - accuracy: 0.6902 - loss: 0.8895 - val_accuracy: 0.6703 - val_loss: 0.9416
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 92s 74ms/step - accuracy: 0.7167 - loss: 0.8116 - val_accuracy: 0.6753 - val_loss: 0.9434
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 89s 71ms/step - accuracy: 0.7328 - loss: 0.7731 - val_accuracy: 0.6743 - val_loss: 0.9567
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 87s 70ms/step - accuracy: 0.7515 - loss: 0.7141 - val_accuracy: 0.6675 - val_loss: 1.0050
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 136s 65ms/step - accuracy: 0.7643 - loss: 0.6773 - val_accuracy: 0.6952 - val_loss: 0.9317
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 77s 61ms/step - accuracy: 0.7849 - loss: 0.6109 - val_accuracy: 0.6993 - val_loss: 0.9005
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 54s 44ms/step - accuracy: 0.7963 - loss: 0.5840 - val_accuracy: 0.6950 - val_loss: 0.9357
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 44ms/step - accuracy: 0.8061 - loss: 0.5557 - val_accuracy: 0.6897 - val_loss: 0.9905
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 54s 43ms/step - accuracy: 0.8223 - loss: 0.5187 - val_accuracy: 0.6878 - val_loss: 0.9678
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 52s 41ms/step - accuracy: 0.8298 - loss: 0.4875 - val_accuracy: 0.6982 - val_loss: 0.9733
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 54s 43ms/step - accuracy: 0.8378 - loss: 0.4645 - val_accuracy: 0.6976 - val_loss: 1.0127
Training stopped at epoch 15
Total training time: 1182.80 seconds
```



12η: Προσθήκη 3ου Convolutional Layer με 128 φίλτρα 130.474 παράμετροι

```
Epoch 1/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 32s 25ms/step - accuracy: 0.2806 - loss: 1.9135 - val_accuracy: 0.5158 - val_loss: 1.3470
Epoch 2/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 29s 23ms/step - accuracy: 0.5387 - loss: 1.2766 - val_accuracy: 0.6003 - val_loss: 1.1260
Epoch 3/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 28s 23ms/step - accuracy: 0.6055 - loss: 1.1117 - val_accuracy: 0.6415 - val_loss: 1.0355
Epoch 4/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 30s 24ms/step - accuracy: 0.6520 - loss: 0.9890 - val_accuracy: 0.6486 - val_loss: 0.9994
Epoch 5/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 29s 23ms/step - accuracy: 0.6852 - loss: 0.8939 - val_accuracy: 0.6683 - val_loss: 0.9679
Epoch 6/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 54s 43ms/step - accuracy: 0.7216 - loss: 0.8001 - val_accuracy: 0.6875 - val_loss: 0.9180
Epoch 7/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 86s 69ms/step - accuracy: 0.7397 - loss: 0.7400 - val_accuracy: 0.6895 - val_loss: 0.8964
Epoch 8/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 95s 76ms/step - accuracy: 0.7641 - loss: 0.6810 - val_accuracy: 0.6880 - val_loss: 0.9360
Epoch 9/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 94s 75ms/step - accuracy: 0.7775 - loss: 0.6429 - val_accuracy: 0.6982 - val_loss: 0.8916
Epoch 10/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 87s 69ms/step - accuracy: 0.7948 - loss: 0.5880 - val_accuracy: 0.6987 - val_loss: 0.9351
Epoch 11/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 98s 78ms/step - accuracy: 0.8090 - loss: 0.5511 - val_accuracy: 0.6987 - val_loss: 0.9420
Epoch 12/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 93s 74ms/step - accuracy: 0.8240 - loss: 0.4989 - val_accuracy: 0.6902 - val_loss: 0.9970
Epoch 13/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 88s 70ms/step - accuracy: 0.8363 - loss: 0.4702 - val_accuracy: 0.6947 - val_loss: 1.0026
Epoch 14/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 72s 58ms/step - accuracy: 0.8510 - loss: 0.4166 - val_accuracy: 0.7070 - val_loss: 0.9903
Epoch 15/15
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 103s 74ms/step - accuracy: 0.8590 - loss: 0.3992 - val_accuracy: 0.6979 - val_loss: 1.0364
Training stopped at epoch 15
Total training time: 1018.62 seconds
```



14. Data Augmentation strategy

Για περαιτέρω βελτίωση του μοντέλου, ακολουθήθηκε τεχνική παραγωγής νέων training images από τις υπάρχουσες 40.000. Με τη χρήση μικρών τυχαίων 'πειραγμάτων' της κάθε εικόνας (flip, rotate, width and height shift). Η τελική απόδοση του μοντέλου:

testing_accuracy=79,51%

validation_accuracy=80,14%

testing_loss=0,5947

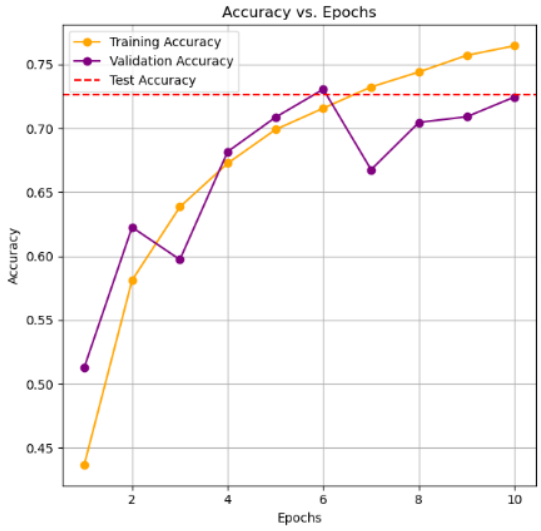
Καταλήγουμε στην εξής 'ιδανική' μορφή του μοντέλου:

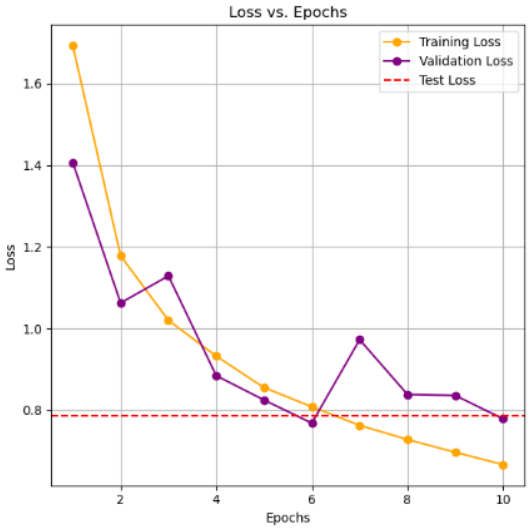
611,626 ΠΑΡΆΜΕΤΡΟΙ

```
# Model Definition
input_shape = (32, 32, 3)
model2 = models.Sequential()

# convolutional layers with Batch Normalization
model2.add(Conv2D(16, (3, 3), activation='relu', input_shape=input_shape, padding='same'))
model2.add(BatchNormalization())
model2.add(MaxPooling2D((2, 2)))

model2.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model2.add(BatchNormalization())
model2.add(MaxPooling2D((2, 2)))

model2.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model2.add(BatchNormalization())
model2.add(MaxPooling2D((2, 2)))

# a dropout layer for regularization
model2.add(Dropout(0.5))

# Flatten and fully connected layers
model2.add(Flatten())
model2.add(Dense(256, activation='relu'))
model2.add(BatchNormalization())
model2.add(Dropout(0.5))

# Output layer for 10 classes
model2.add(Dense(10, activation='softmax'))
```

1η ΕΚΠΑΊΔΕΥΣΗ

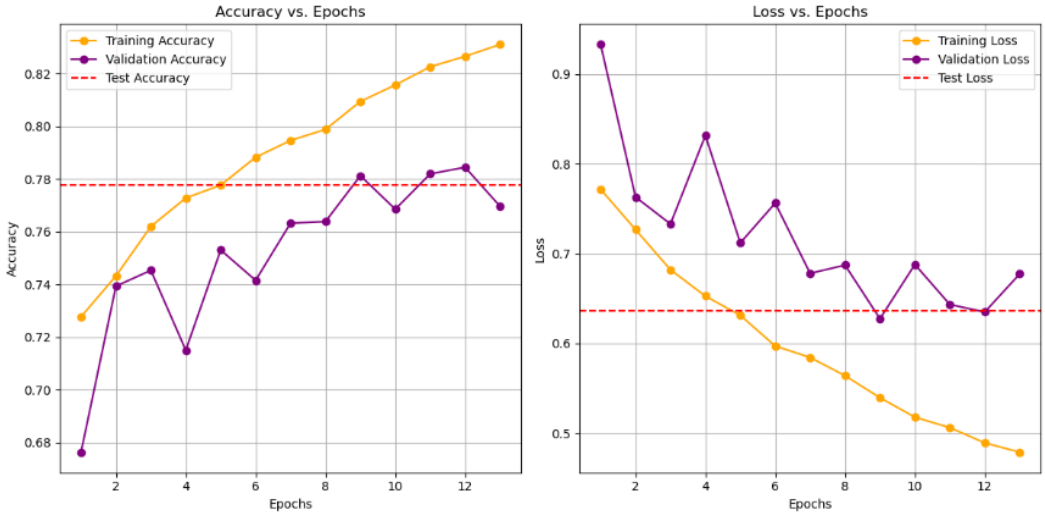
```
625/625 ━━━━━━━━━━━━━━━━━━━━ 79s 120ms/step - accuracy: 0.3591 - loss: 2.0701 - val_accuracy: 0.5130 - val_loss: 1.4063
Epoch 2/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 53s 86ms/step - accuracy: 0.5679 - loss: 1.2155 - val_accuracy: 0.6226 - val_loss: 1.0624
Epoch 3/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 54s 86ms/step - accuracy: 0.6363 - loss: 1.0232 - val_accuracy: 0.5973 - val_loss: 1.1286
Epoch 4/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 84ms/step - accuracy: 0.6723 - loss: 0.9373 - val_accuracy: 0.6818 - val_loss: 0.8843
Epoch 5/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 53s 84ms/step - accuracy: 0.6996 - loss: 0.8565 - val_accuracy: 0.7087 - val_loss: 0.8246
Epoch 6/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.7206 - loss: 0.7965 - val_accuracy: 0.7306 - val_loss: 0.7679
Epoch 7/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 54s 87ms/step - accuracy: 0.7358 - loss: 0.7543 - val_accuracy: 0.6677 - val_loss: 0.9724
Epoch 8/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 54s 86ms/step - accuracy: 0.7482 - loss: 0.7212 - val_accuracy: 0.7047 - val_loss: 0.8382
Epoch 9/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 56s 89ms/step - accuracy: 0.7612 - loss: 0.6844 - val_accuracy: 0.7091 - val_loss: 0.8358
Epoch 10/40
625/625 ━━━━━━━━━━━━━━━━━━━━ 60s 96ms/step - accuracy: 0.7709 - loss: 0.6516 - val_accuracy: 0.7246 - val_loss: 0.7785
Training stopped at epoch 10
Total training time: 568.97 seconds
```





2η ΕΚΠΑΊΔΕΥΣΗ

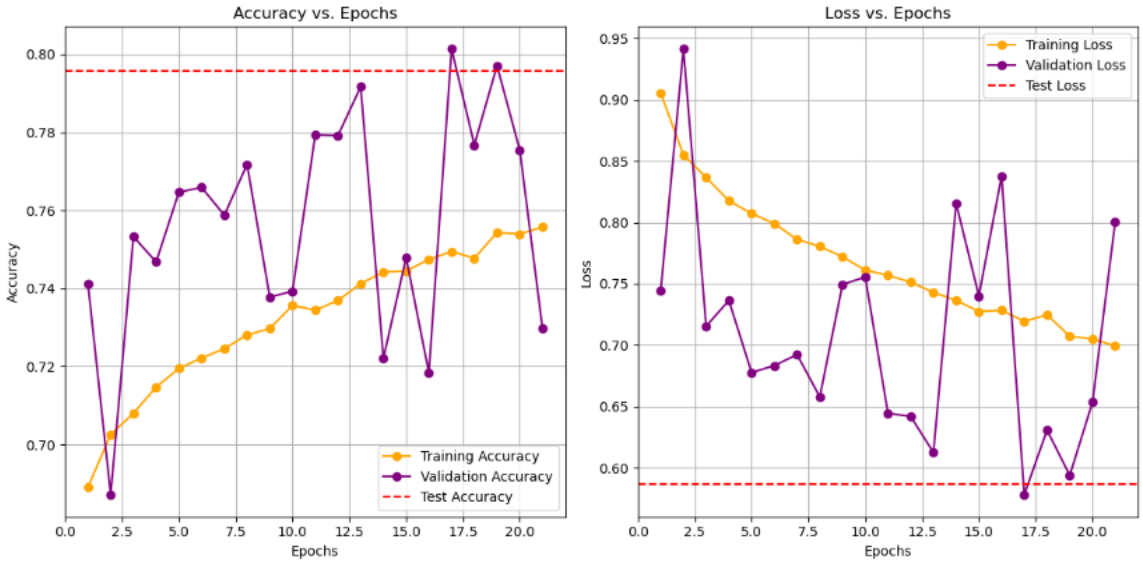
```
625/625 ━━━━━━━━━━━━━━━━━━━━ 58s 93ms/step - accuracy: 0.7350 - loss: 0.7551 - val_accuracy: 0.6764 - val_loss: 0.9332
Epoch 2/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 91ms/step - accuracy: 0.7485 - loss: 0.7121 - val_accuracy: 0.7394 - val_loss: 0.7632
Epoch 3/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 59s 94ms/step - accuracy: 0.7671 - loss: 0.6705 - val_accuracy: 0.7454 - val_loss: 0.7331
Epoch 4/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 92ms/step - accuracy: 0.7760 - loss: 0.6445 - val_accuracy: 0.7151 - val_loss: 0.8317
Epoch 5/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 59s 94ms/step - accuracy: 0.7838 - loss: 0.6162 - val_accuracy: 0.7531 - val_loss: 0.7121
Epoch 6/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 63s 101ms/step - accuracy: 0.7931 - loss: 0.5887 - val_accuracy: 0.7416 - val_loss: 0.7564
Epoch 7/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 58s 93ms/step - accuracy: 0.8014 - loss: 0.5684 - val_accuracy: 0.7633 - val_loss: 0.6781
Epoch 8/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 56s 90ms/step - accuracy: 0.8065 - loss: 0.5462 - val_accuracy: 0.7639 - val_loss: 0.6873
Epoch 9/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 56s 90ms/step - accuracy: 0.8146 - loss: 0.5220 - val_accuracy: 0.7813 - val_loss: 0.6276
Epoch 10/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 56s 90ms/step - accuracy: 0.8223 - loss: 0.4967 - val_accuracy: 0.7686 - val_loss: 0.6880
Epoch 11/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 58s 93ms/step - accuracy: 0.8233 - loss: 0.5038 - val_accuracy: 0.7820 - val_loss: 0.6435
Epoch 12/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 56s 89ms/step - accuracy: 0.8327 - loss: 0.4724 - val_accuracy: 0.7845 - val_loss: 0.6350
Epoch 13/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.8377 - loss: 0.4620 - val_accuracy: 0.7697 - val_loss: 0.6774
Training stopped at epoch 13
Total training time: 746.61 seconds
```



3η ΕΚΠΑΊΔΕΥΣΗ ΜΕ ΤΗ ΧΡΉΣΗ DATA AUGMENTATION

Η εκπαίδευση σταματάει απότομα στην 20η εποχή γιατί crushare το laptop. (έτρεχα παράλληλα άλλες 3 εκπαιδεύσης)

```
625/625 ━━━━━━━━━━━━━━━━━━━━ 59s 92ms/step - accuracy: 0.6812 - loss: 0.9380 - val_accuracy: 0.7411 - val_loss: 0.7443
Epoch 2/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 62s 100ms/step - accuracy: 0.7024 - loss: 0.8557 - val_accuracy: 0.6872 - val_loss: 0.9414
Epoch 3/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 59s 94ms/step - accuracy: 0.7093 - loss: 0.8356 - val_accuracy: 0.7533 - val_loss: 0.7154
Epoch 4/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 59s 94ms/step - accuracy: 0.7136 - loss: 0.8196 - val_accuracy: 0.7468 - val_loss: 0.7364
Epoch 5/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 61s 98ms/step - accuracy: 0.7209 - loss: 0.8022 - val_accuracy: 0.7647 - val_loss: 0.6776
Epoch 6/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 53s 84ms/step - accuracy: 0.7183 - loss: 0.8068 - val_accuracy: 0.7659 - val_loss: 0.6833
Epoch 7/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 83ms/step - accuracy: 0.7252 - loss: 0.7819 - val_accuracy: 0.7588 - val_loss: 0.6922
Epoch 8/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 52s 84ms/step - accuracy: 0.7250 - loss: 0.7910 - val_accuracy: 0.7718 - val_loss: 0.6578
Epoch 9/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 60s 96ms/step - accuracy: 0.7367 - loss: 0.7543 - val_accuracy: 0.7378 - val_loss: 0.7494
Epoch 10/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 83s 97ms/step - accuracy: 0.7396 - loss: 0.7531 - val_accuracy: 0.7393 - val_loss: 0.7555
Epoch 11/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 66s 106ms/step - accuracy: 0.7339 - loss: 0.7618 - val_accuracy: 0.7794 - val_loss: 0.6444
Epoch 12/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 84s 135ms/step - accuracy: 0.7360 - loss: 0.7550 - val_accuracy: 0.7792 - val_loss: 0.6418
Epoch 13/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 85s 136ms/step - accuracy: 0.7420 - loss: 0.7417 - val_accuracy: 0.7917 - val_loss: 0.6124
Epoch 14/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 74s 118ms/step - accuracy: 0.7444 - loss: 0.7354 - val_accuracy: 0.7221 - val_loss: 0.8154
Epoch 15/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 69s 110ms/step - accuracy: 0.7428 - loss: 0.7269 - val_accuracy: 0.7479 - val_loss: 0.7401
Epoch 16/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 60s 95ms/step - accuracy: 0.7472 - loss: 0.7273 - val_accuracy: 0.7184 - val_loss: 0.8378
Epoch 17/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 59s 94ms/step - accuracy: 0.7501 - loss: 0.7243 - val_accuracy: 0.8014 - val_loss: 0.5782
Epoch 18/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 91ms/step - accuracy: 0.7494 - loss: 0.7201 - val_accuracy: 0.7768 - val_loss: 0.6306
Epoch 19/50
625/625 ━━━━━━━━━━━━━━━━━━━━ 57s 92ms/step - accuracy: 0.7549 - loss: 0.7031 - val_accuracy: 0.7971 - val_loss: 0.5938
Epoch 20/50
480/625 ━━━━━━━━━━━━━━━━━━━━ 13s 95ms/step - accuracy: 0.7544 - loss: 0.7027
```



4. Σύγκριση του νευρωνικού με τους κατηγοριοποιητές 1-NN, 3-NN και Nearest Centroid

Συγκρίνοντας το νευρωνικό δίκτυο με τις μεθόδους κατηγοριοποίησης 1-NN, 3-NN και Nearest Centroid, διακρίνουμε διαφορές σε απόδοση, υπολογιστική πολυπλοκότητα, ικανότητα γενίκευσης και χρόνο υλοποίησης. Το νευρωνικό δίκτυο αποδεικνύεται αρκετά αποτελεσματικό για το dataset CIFAR-10, με τελική απόδοση που αγγίζει το 80%, με τη χρήση της βιβλιοθήκης keras. Ωστόσο, απαιτεί κάποιον χρόνο εκπαίδευσης και βελτιστοποίησης των παραμέτρων, καθώς και έρευνα όσον αφορά την δομή του νευρωνικού για προσέγγιση μιας βέλτιστης απόδοσης.

Αντίθετα, η μέθοδος 1-NN, που βασίζεται στον πλησιέστερο γείτονα, δεν χρειάζεται εκπαίδευση, αλλά έχει υψηλό υπολογιστικό κόστος κατά την πρόβλεψη λόγω του μεγάλου αριθμού δειγμάτων και χαρακτηριστικών της CIFAR-10.

Η μέθοδος 3-NN, αν και βελτιώνει την αντοχή στον θόρυβο χρησιμοποιώντας πλειοψηφία μεταξύ τριών γειτόνων, παραμένει υπολογιστικά ακριβή και υπολείπεται σε απόδοση όταν τα δεδομένα είναι πολυδιάστατα και μη γραμμικά διαχωρίσιμα.

Από την άλλη πλευρά, η Nearest Centroid, που κατηγοριοποιεί δείγματα βάσει της απόστασής τους από το κέντρο της κάθε κλάσης, είναι γρήγορη και αποδοτική. Ωστόσο, αποδίδει καλά μόνο αν τα δεδομένα είναι γραμμικά διαχωρίσιμα και ο θόρυβος δεν επηρεάζει τη θέση του κέντρου. Στη CIFAR-10, η απόδοσή της είναι περιορισμένη λόγω της πολυπλοκότητας του dataset.

Συνολικά, το νευρωνικό δίκτυο είναι η πιο κατάλληλη μέθοδος για τη CIFAR-10, προσφέροντας υψηλή ακρίβεια.

5. Παρουσίαση παραδειγμάτων επιτυχίας και αποτυχίας classification.

Εύκολος κώδικας αλλά δεν υπάρχουν τα μέσα για να υλοποιηθεί. Τα έκαψε η εργασία ☺…

Αντιγραφή κώδικα

```python
import numpy as np
import matplotlib.pyplot as plt

# Predict the labels for the test dataset
y_pred = model.predict(X_test)  # Model predictions (probabilities for each clas
y_pred_classes = np.argmax(y_pred, axis=1)  # Convert probabilities to class ind

# Identify correct and incorrect classifications
correct_indices = np.where(y_pred_classes == y_test)[0]
incorrect_indices = np.where(y_pred_classes != y_test)[0]

# Helper function to plot examples
def plot_examples(indices, title, n=5):
    plt.figure(figsize=(15, 5))
    for i, idx in enumerate(indices[:n]):  # Show up to n examples
        plt.subplot(1, n, i + 1)
        plt.imshow(X_test[idx])  # Display image
        plt.title(f"True: {y_test[idx]}, Pred: {y_pred_classes[idx]}")
        plt.axis("off")
    plt.suptitle(title)
    plt.show()

# Plot examples of correct classifications
plot_examples(correct_indices, title="Correct Classifications")

# Plot examples of incorrect classifications
plot_examples(incorrect_indices, title="Incorrect Classifications")
```

3. Explanation

- `model.predict(X_test)`: Generates predictions (class probabilities) for the test set.
- `np.argmax(y_pred, axis=1)`: Converts predicted probabilities to class indices.
- `np.where(y_pred_classes == y_test)`: Finds indices of correctly classified samples.
- `np.where(y_pred_classes != y_test)`: Finds indices of misclassified samples.
- `plot_examples`: Visualizes the examples using `matplotlib`.

---

4. Output

- **Correct Classifications**: Displays examples where the model prediction matches the true label.
- **Incorrect Classifications**: Displays examples where the model prediction is wrong, helping identify common failure cases.

Let me know if you'd like additional adjustments!