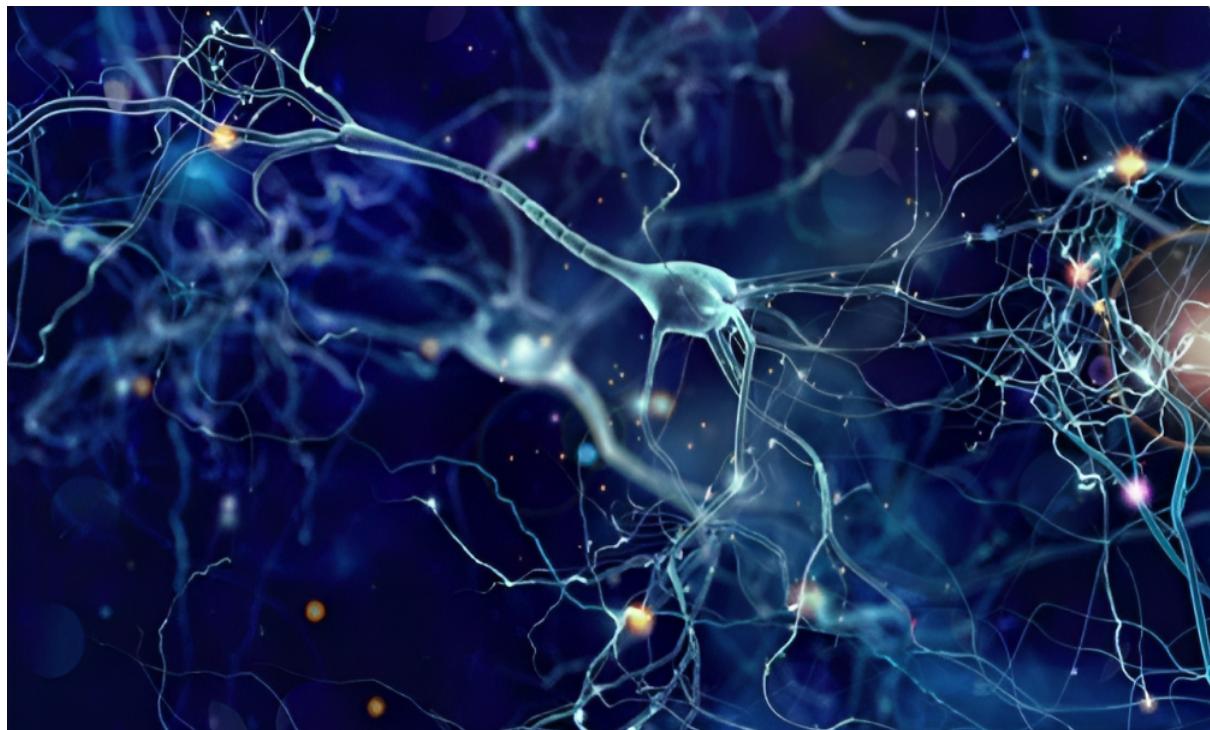


Νευρωνικά Δίκτυα - Βαθιά Μάθηση

Εργασία 1



Ονοματεπώνυμο: Μαχμουτάι Έλενα
Τμήμα: Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
ΑΕΜ: 10012

1. Υλοποίηση της Εργασίας

Σκοπός της εργασίας ήταν να πραγματοποιηθεί μια εκτενής ανάλυση διαφόρων συνδυασμών hidden layer ενός νευρωνικού δικτύου προκειμένου να βρεθεί το δίκτυο που θα έχει τα πιο ακριβή αποτελέσματα με την λιγότερη απώλεια.

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε η γλώσσα Python και συγκεκριμένα το Tensorflow framework και το keras. Η καταγραφή και μελέτη των γραφικών παραστάσεων έγινε με την βοήθεια του εργαλείου οπτικοποίησης TensorBoard ενώ το dataset που χρησιμοποιήθηκε ήταν το CIFAR-10 και ο διαχωρισμός του έγινε στα υποσύνολα 70% (35,000 εικόνες) για το training και 30% για το evaluation (15,000 εικόνες).

2. Πρώτο Πείραμα

Για το πρώτο πείραμα που βρίσκεται στο αρχείο με όνομα *nn_first_experiment.py* μας ενδιέφερε να μελετήσουμε την διαφορά ανάμεσα στον αριθμό των νευρώνων που είχαν τα 2 Dense hidden layer στην απόδοση και στην απώλεια για να μπορέσουμε να αποφασίσουμε ποιός συνδυασμός είναι ο βέλτιστος αλλά και να τα συγκρίνουμε μεταξύ τους. Αποφασίσαμε ότι το input layer του κάθε δικτύου έχει σχήμα (32,32,3) και είναι σε flatten μορφή (1D Array), ενώ τα output layers θεωρήθηκαν τυχαία με 10 νευρώνες και Activation function την συνάρτηση Softmax. Η ενεργοποίηση Softmax προτιμήθηκε λόγω της ικανότητάς της να μετατρέπει τις πρωτογενείς βαθμολογίες εξόδου σε κατανομή πιθανότητας σε πολλαπλές κλάσεις. Αυτός ο μετασχηματισμός διευκολύνει την ερμηνεία των πιθανοτήτων κλάσεων, ζωτικής σημασίας για εργασίες όπου οι είσοδοι εκχωρούνται σε διαφορετικές κλάσεις. Επιπλέον, η softmax κανονικοποιεί τις βαθμολογίες εξόδου, διασφαλίζοντας ότι το άθροισμα των πιθανοτήτων σε όλες τις κατηγορίες είναι ίσο με 1, απλοποιώντας τις συγκρίσεις και τη λήψη αποφάσεων.

Στα hidden layer χρησιμοποιήθηκε RELU Activation Function καθώς προσφέρει μια μη γραμμική αλλα πολύ απλή ενεργοποίηση που χρειαζόμασταν στο συγκεκριμένο πείραμα. Επιπλέον, είναι υπολογιστικά αποδοτική, αφού αφήνει θετικές τιμές να περνούν και θέτει τις αρνητικές τιμές στο μηδέν, πράγμα που δεν δημιουργεί επιβάρυνση και άρα το δίκτυο μας εκπαιδεύεται πιο γρήγορα σε σχέση με άλλες συναρτήσεις και μειώνεται η πολυπλοκότητα του.

Όπως παρατηρούμε στον κώδικα, έχουμε ορίσει τα *layer_sizes* (αριθμοί των νευρώνων) των οποίων το συνδυασμό θέλουμε να μελετήσουμε σε 10, 64, 128 και 256. Για κάθε δίκτυο υπάρχουν 10 Epochs καθώς παρατηρήθηκε ότι για τα συγκεκριμένα δίκτυα έδινε το πιο επιθυμητό αποτέλεσμα. Όταν ο αριθμός των epochs ξεπερνούσε το 10 παρατηρήσαμε μια σταδιακή φθορά του και μείωση της ακριβειας και αύξηση την απώλειας, δηλώνοντας σημάδια υπερεκπαίδευσης του.

Τα γραφήματα που εξάγαμε από το TensorBoard είναι τα παρακάτω:

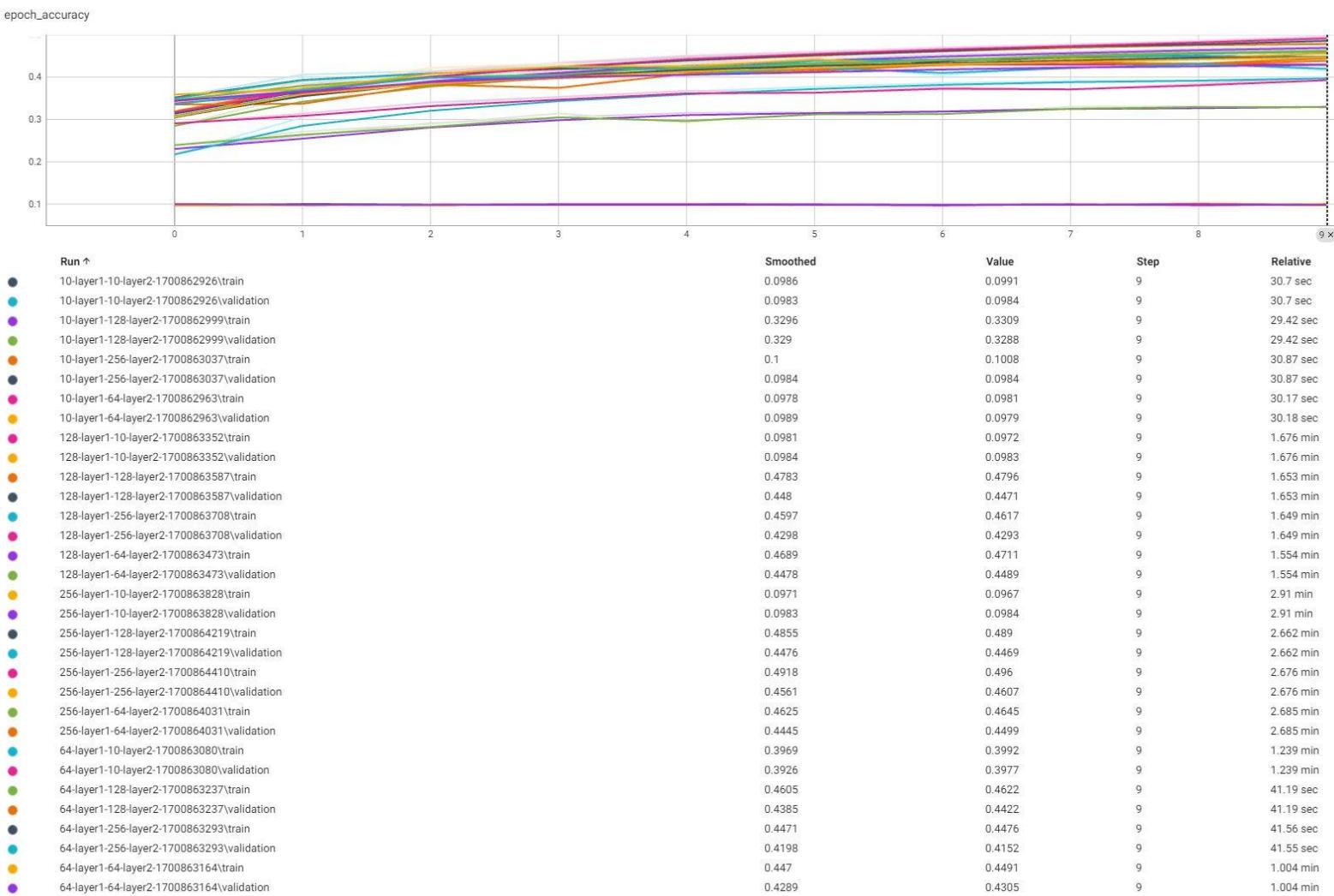


Figure 2.1 Training and Validation Accuracy evolution over epochs

Από το παραπάνω διάγραμμα παρατηρούμε ότι δεν υπάρχουν μεγάλες διαφορές στην ακρίβεια του validation και του training και άρα δεν υπάρχουν σημαντικά σημάδια overfitting ή underfitting του δικτύου. Επίσης, όπως θα περιμέναμε, με την αύξηση του αριθμού των νευρώνων, αυξάνεται και ο χρόνος που κάνει να εκτελεστεί το πρόγραμμα. Αυτό που παρατηρείται κυρίως όμως είναι ότι ο χρόνος αυτός επηρεάζεται πολύ περισσότερο από τον αριθμό των νευρώνων του πρώτου hidden layer από ότι του δευτέρου.

Γενικότερα, παρατηρούμε ότι από το training Dataset μεγαλύτερη ακρίβεια εμφάνισε το νευρωνικό δίκτυο '256-layer1-256-layer2-1700864410\train' που έχει 256 νευρώνες στο πρώτο εσωτερικό στρώμα και 256 νευρώνες στο δεύτερο. Από τα συγκεκριμένα διαγράμματα φαίνεται ότι τα δίκτυα που έχουν σε ένα από τα δύο layer 10 νευρώνες έχουν σημαντικά μικρότερη απόδοση από αυτά που έχουν 256. Άρα θα μπορούσε να συμπεράνει κανείς ότι στο συγκεκριμένο νευρωνικό δίκτυο η αύξηση των νευρώνων έχει σημαντικό ρόλο στην απόδοση του δικτύου.

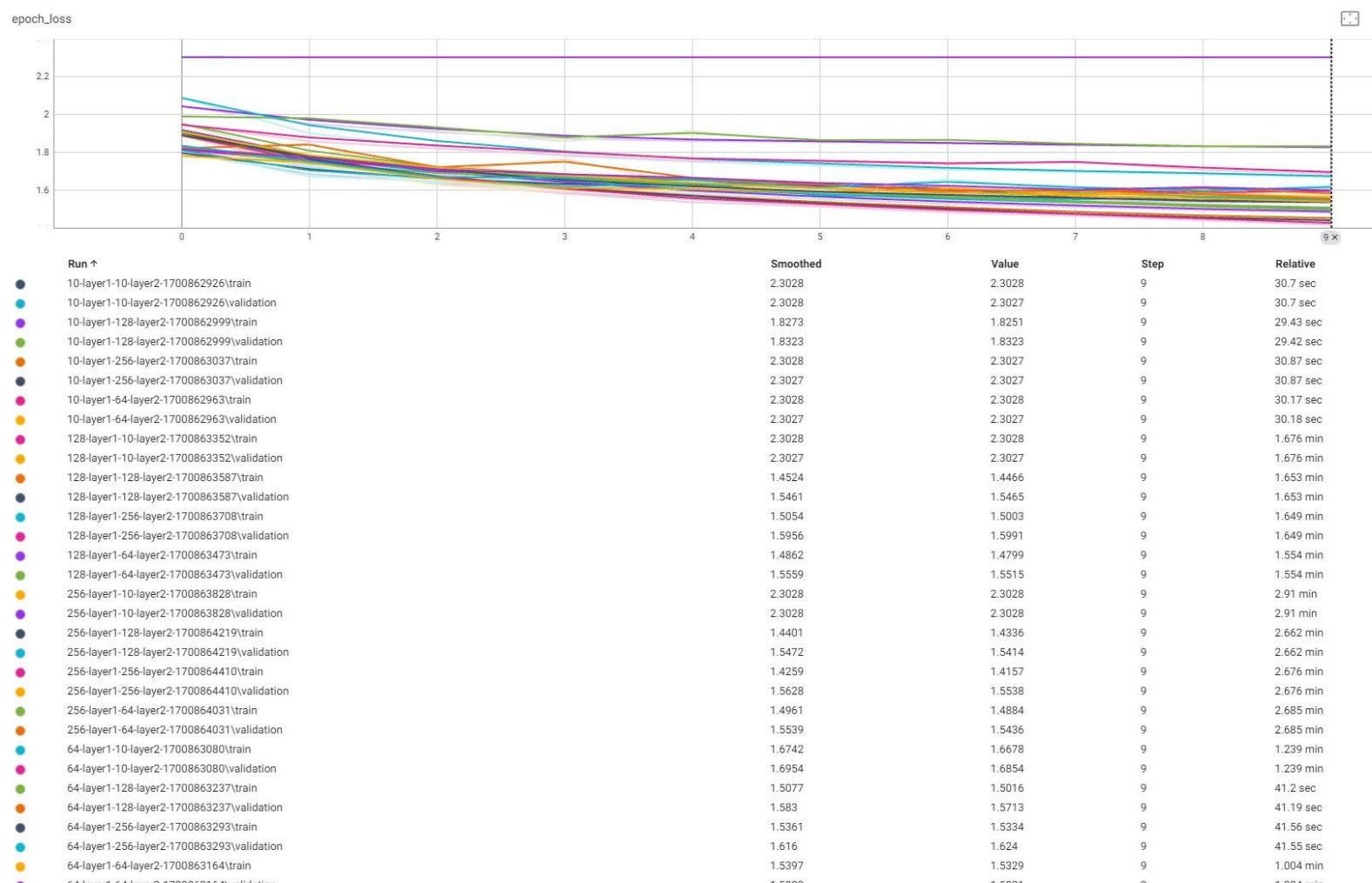


Figure 2.2 Training and Validation Loss reduction across epochs

Στο παραπάνω διάγραμμα παρατηρούμε ότι λιγότερη απώλεια παρατηρείται στο δίκτυο '256-layer1-256-layer2-1700864410\train' αφού το δίκτυο έχει μεγαλύτερη δυνατότητα να μαθαίνει τα μοτίβα λόγω της αυξημένης πολυπλοκότητας.

Αντίστοιχα δίκτυο με την περισσότερη απώλεια είναι το '256-layer1-10-layer2-1700863828\train'. Κάτι τέτοιο θα μπορούσε να οφείλεται στο γεγονός ότι με μόνο 10 νευρώνες στο δεύτερο εσωτερικό στρώμα δεν έχουμε την επαρκή πολυπλοκότητα να δημιουργήσουμε τα μοτίβα που θέλουμε και μπορεί κάποιες πληροφορίες να μην λαμβάνονται υπόψη.

evaluation_accuracy_vs_iterations

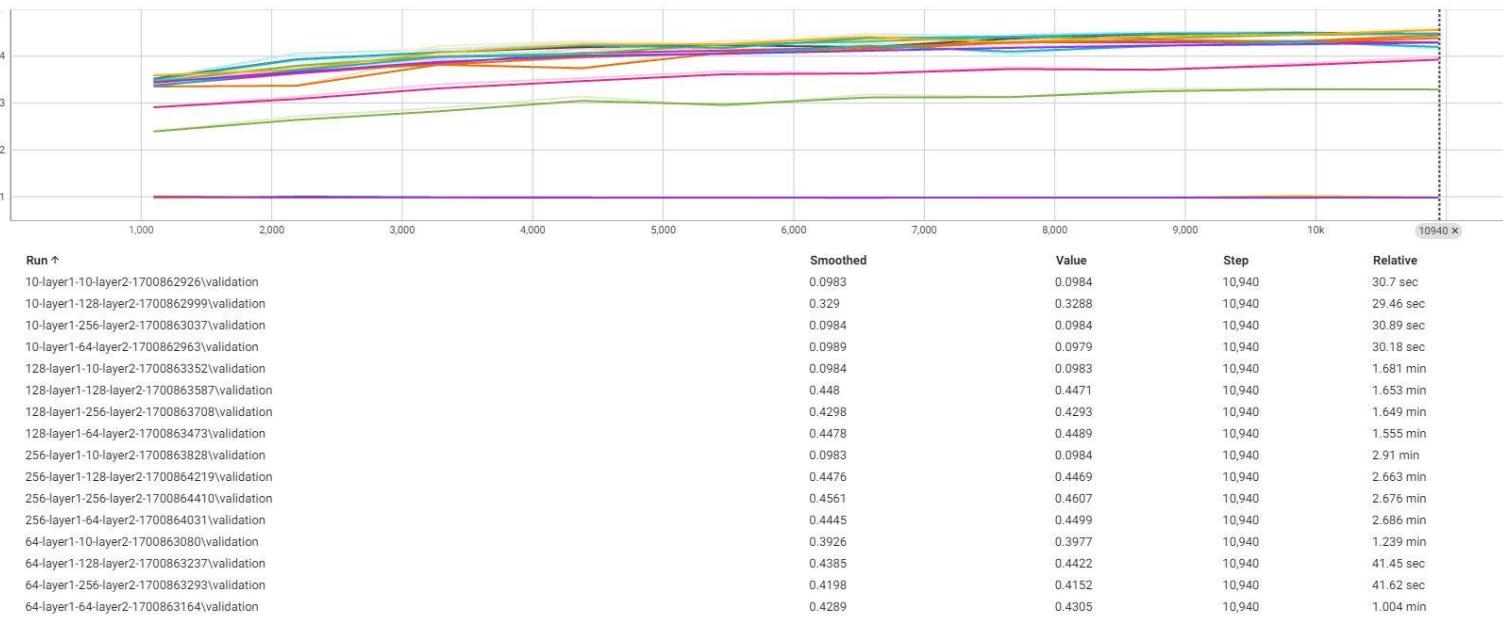


Figure 2.3 Model Accuracy during evaluation across iterations

evaluation_loss_vs_iterations

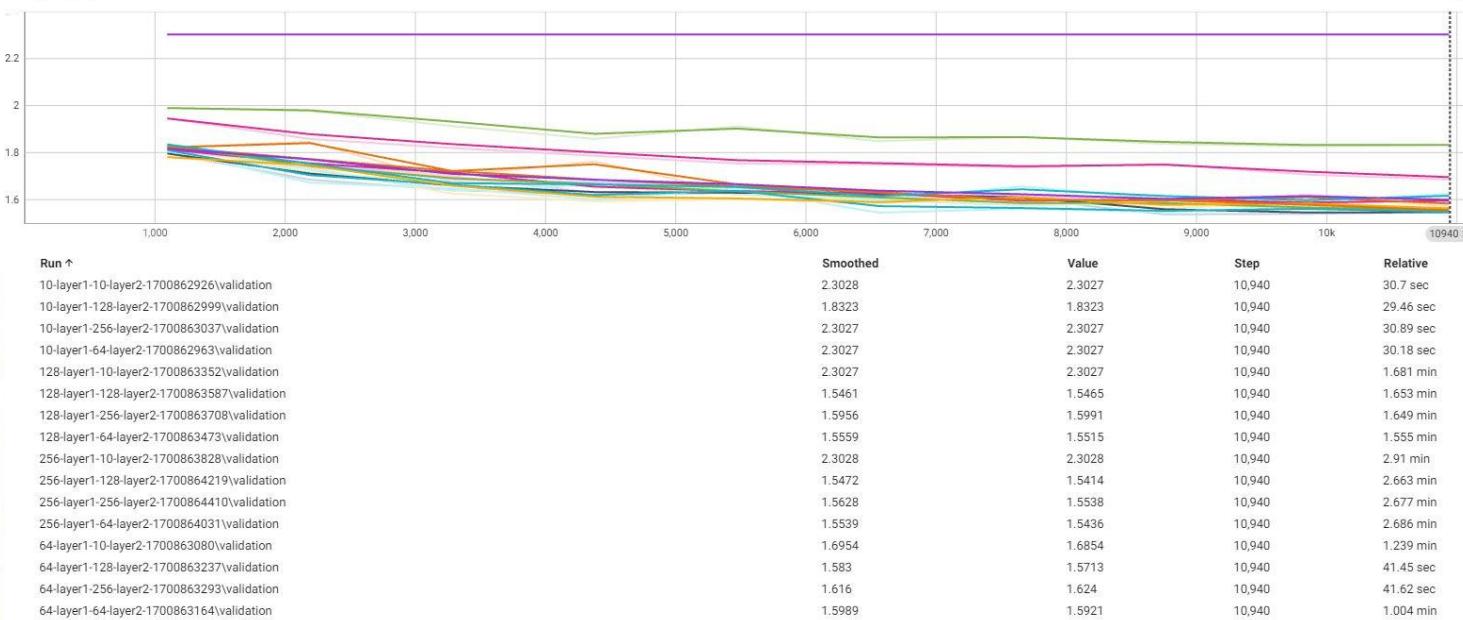


Figure 2.4 Model loss during evaluation across iterations

Στα παραπάνω διάγραμμα παρατηρούμε ότι τα δίκτυα που έχουν πιο πολύπλοκη δομή χρειάζονται περισσότερο χρόνο να σταθεροποιηθούν. Επιπλέον παρατηρούμε ότι όσο περνάει ο χρόνος αυξάνεται η απόδοση και μειώνονται οι απώλειες. Τα διαγράμματα δεν έχουν πολλές διακυμάνσεις και οροπέδια που σημαίνει ότι δεν έχουμε μεγάλη αστάθεια στα δίκτυα.

3. Δεύτερο Πείραμα

Το δεύτερο πείραμα βρίσκεται στο αρχείο με όνομα *nn_second_experiment.py*. Πρόκειται για μια CNN μελέτη για ταξινόμηση των εικόνων του CIFAR-10 Dataset. Έχουμε 2 Convolutional layers (2D layers) στα οποία χρησιμοποιήσαμε RELU Activation function. Επιπλέον πριν το output layer χρησιμοποιούμε την συνάρτηση *layers.Flatten* προκειμένου να μεταβούμε από τα conv layers(2D) σε dense layers (1D), κάνοντας με αυτόν τον τρόπο ένα reshape της εξόδου σε flat διάνυσμα.

Όπως παρατηρούμε στον κώδικα, τα *layer_sizes* (αριθμοί των νευρώνων) έχουν οριστεί σε 10, 64, 128 και 256 προκειμένου να μπορούμε να τα συγκρίνουμε και με το παραπάνω πείραμα. Για τον ίδιο λόγο όπως και νωρίτερα, ο αριθμός των epochs έχει οριστεί σε 10. Επιπλέον και πάλι στα hidden layer χρησιμοποιήθηκε RELU Activation Function ενώ στο output layer έχουμε 10 νευρώνες και activation function τη Softmax. Το input layer του κάθε δικτύου έχει σχήμα (32,32,3) και είναι σε flatten μορφή (1D Array). Είναι φυσικό ότι συγκριτικά με το προηγούμενο πείραμα περιμένουμε να έχουμε αρκετά πιο αυξημένες αποδόσεις και πολύ λιγότερη απώλεια αφού διαθέτει εσωτερικά φίλτρα στους νευρώνες προκειμένου να μπορεί να αναγνωρίζει μοτίβα στις εικόνες.

Τα γραφήματα που εξάγαμε από το TensorBoard είναι τα παρακάτω:

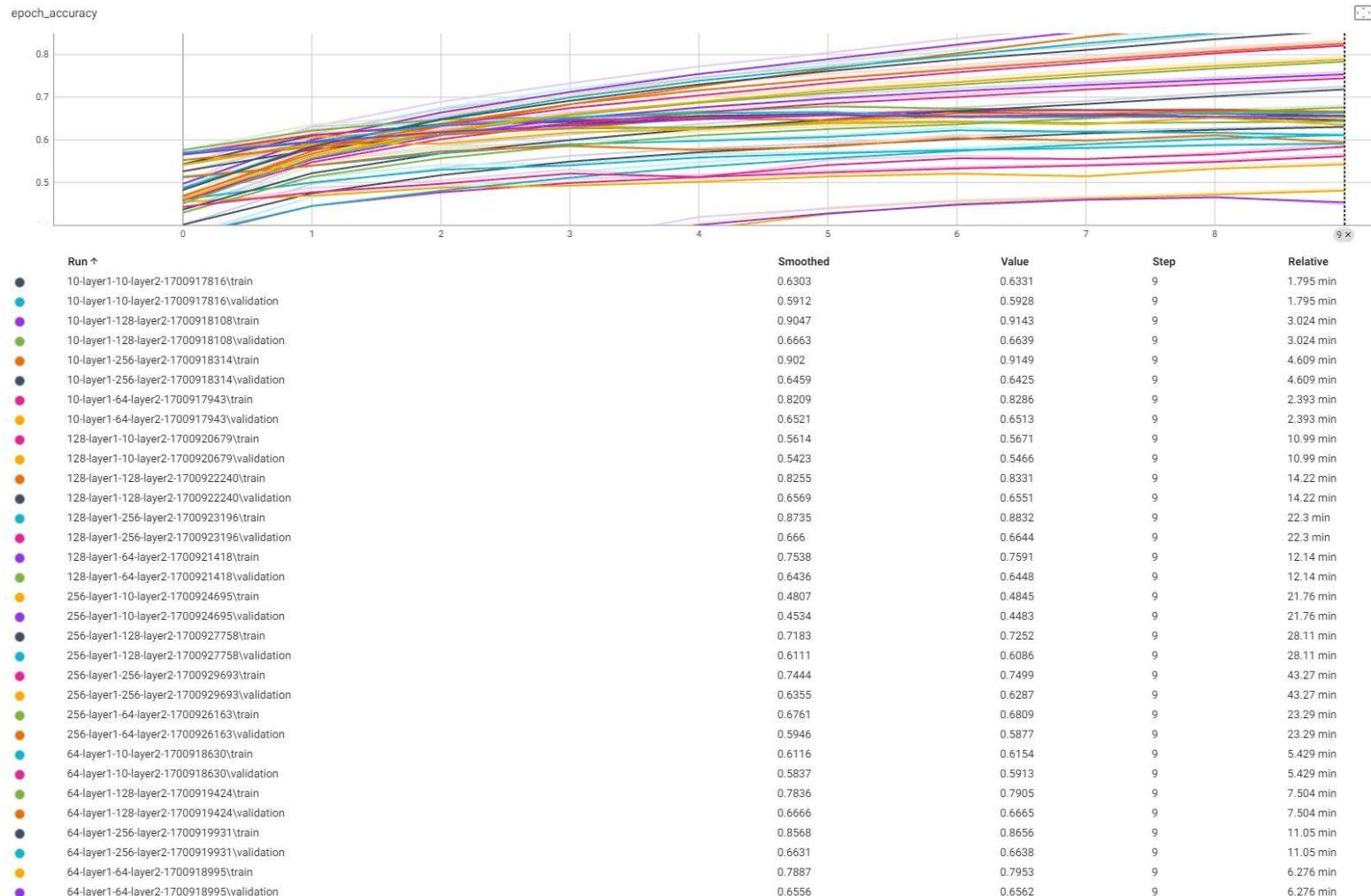


Figure 3.1 Training and Validation Accuracy evolution over epochs

Στο παραπάνω διάγραμμα παρατηρούμε ότι υπάρχουν μεγάλες διαφορές μεταξύ training και validation data και τα training data έχουν πολύ πιο αυξημένη επίδοση, γεγονός που μας δείχνει ότι το δίκτυο έχει υποστεί overfitting. Και ενώ τα αποτελέσματα στην ακρίβεια του δικτύου ήταν πολύ υψηλά αποφασίστηκε να γίνει βελτίωση του προκειμένου να συνεχιστεί η ανάλυση του, καθώς γνωρίζουμε ότι ένα overfit μοντέλο έχει εξαιρετικά καλή απόδοση στα training data, αλλά αποτυγχάνει σε νέα δεδομένα, αφού ουσιαστικά απομνημονεύει το σετ εκπαίδευσης, χωρίς την ικανότητα να κάνει ακριβείς προβλέψεις σε άγνωστες περιπτώσεις.

Το βελτιωμένο δίκτυο βρίσκεται στο αρχείο `nn_second_experiment_fixed_overfitting.py`. Όπως βλέπουμε έχει προστεθεί η γραμμή `model.add(layers.Dropout(0.5))` έπειτα από κάθε convolutional layer. Το dropout είναι μια τεχνική που χρησιμοποιείται συνήθως για την αποφυγή υπερβολικής τοποθέτησης, αφού θέτει τυχαία ένα κλάσμα των μονάδων εισόδου στο μηδέν κατά τη διάρκεια της προπόνησης, κάτι που βοηθά στην αποφυγή της υπερβολικής εξάρτησης του μοντέλου από συγκεκριμένους νευρώνες και ενθαρρύνει την εκμάθηση πιο ισχυρών χαρακτηριστικών.

Τα νέα γραφήματα που εξάγαμε από το TensorBoard είναι τα παρακάτω:

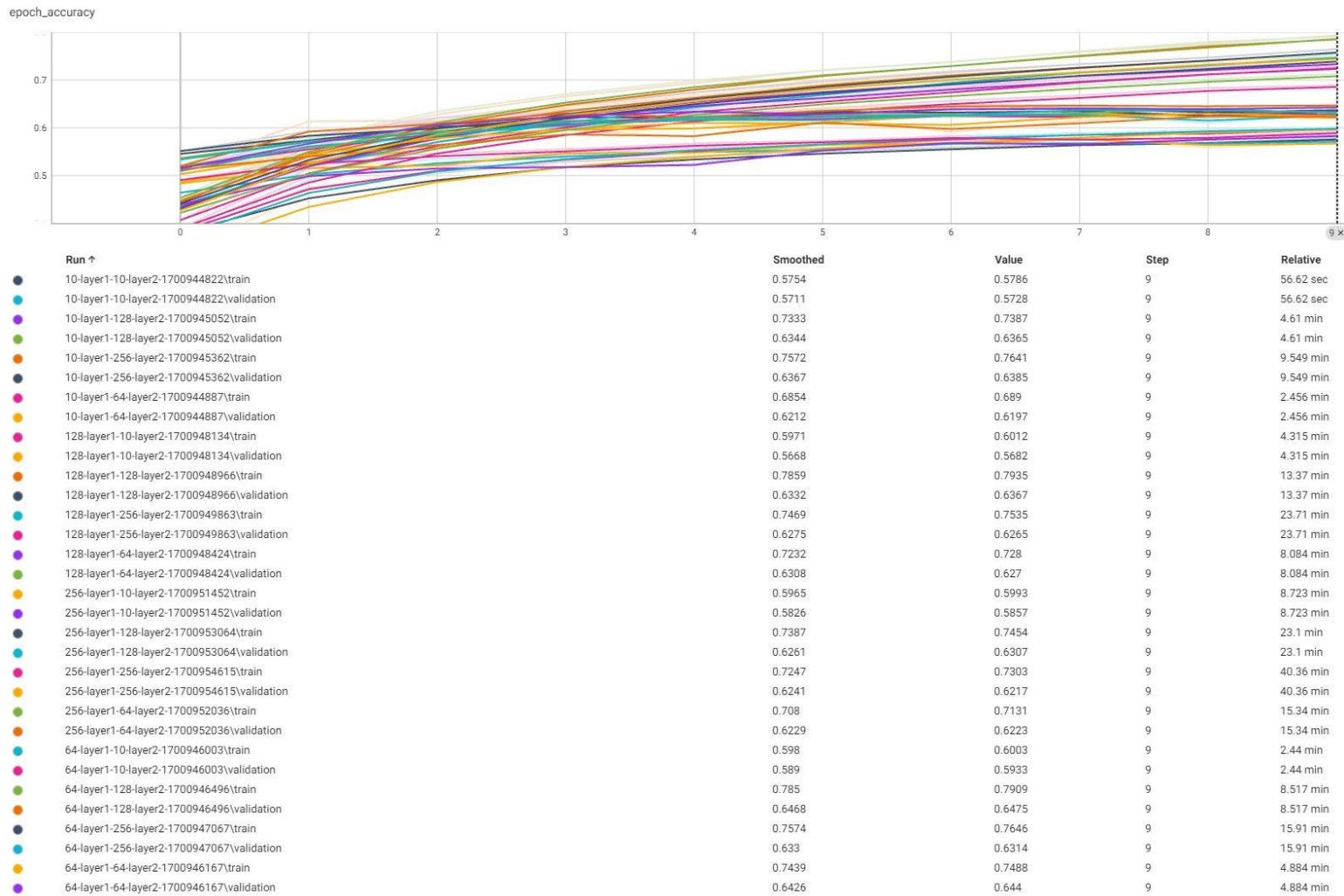


Figure 3.2 Training and Validation Accuracy evolution over epochs

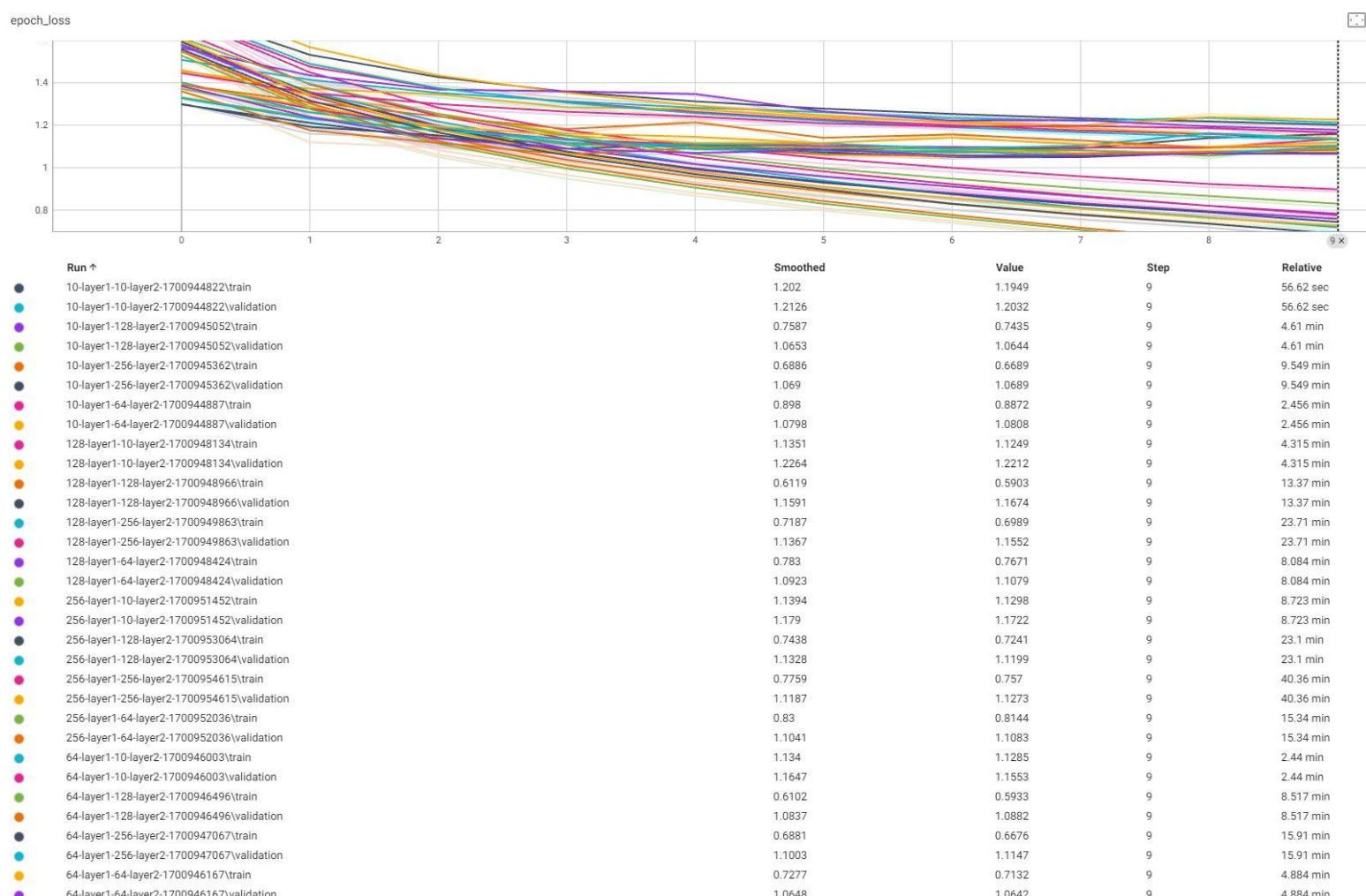


Figure 3.3 Training and Validation Loss reduction across epochs

Με βάση τα παραπάνω διαγράμματα παρατηρούμε ότι το overfitting έχει βελτιωθεί σημαντικά, αφού οι διαφορές είναι πολύ μικρότερες μεταξύ training και validation dataset.

Επιπλέον, αντίθετα με το πρώτο πείραμα, εδώ βλέπουμε ότι το νευρωνικό δίκτυο που είχε την καλύτερη απόδοση και τις λιγότερες απώλειες ήταν το '128-layer1-128-layer2-1700948966\train'. Κάτι τέτοιο θα μπορούσε να οφείλεται στο ότι η διαμόρφωση 256-layer1-256-layer2 έχει μεγαλύτερο αριθμό παραμέτρων και αυξημένη πολυπλοκότητα μοντέλου και μπορεί να δυσκολεύεται να προσαρμοστεί σε ένα σχετικά μικρό σύνολο δεδομένων όπως το CIFAR-10.

Γενικότερα, όμως, βλέπουμε για τους ίδιους αριθμούς νευρώνων αυξημένες αποδόσεις και μειωμένες απώλειες στο δεύτερο πείραμα από αυτές στο πρώτο, γεγονός που φυσικά περιμέναμε αφού το CNN βασίζεται σε μοτίβα που είναι ιδανικά για αναλύσεις εικόνων.

evaluation_accuracy_vs_iterations

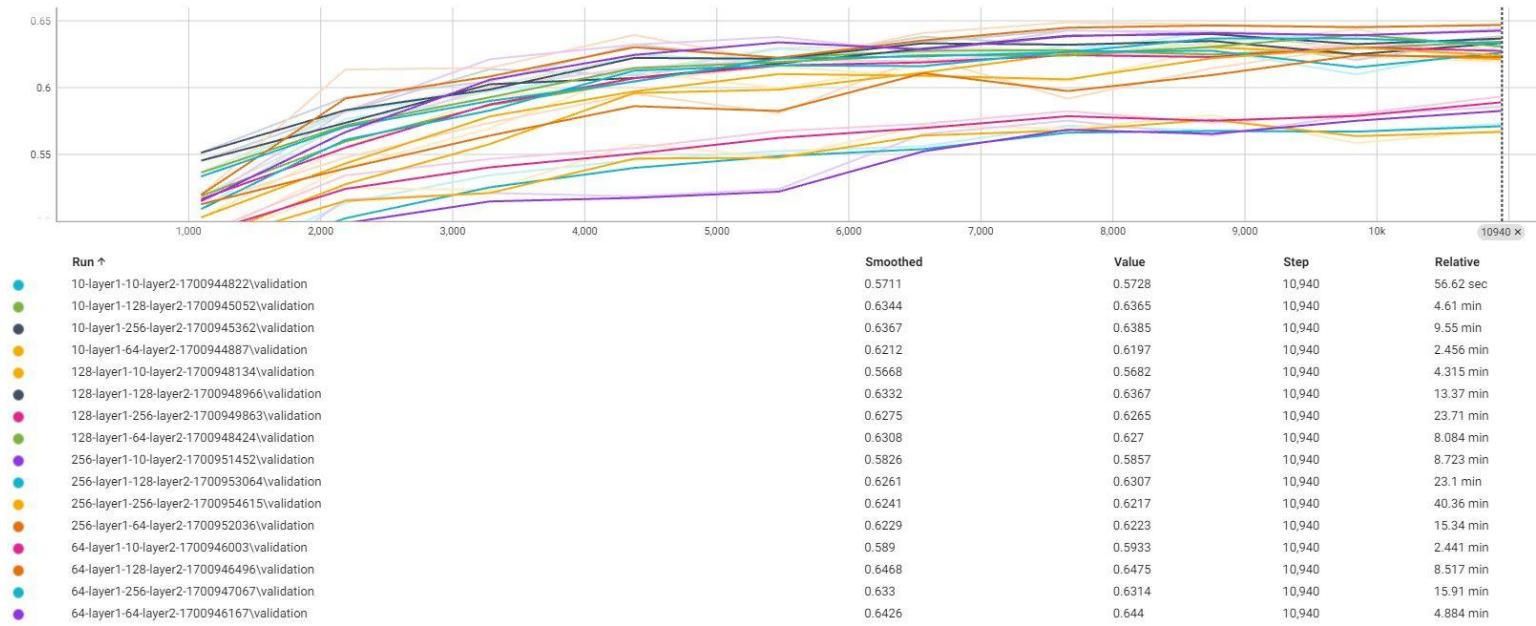


Figure 3.4 Model Accuracy during evaluation across iterations

evaluation_loss_vs_iterations

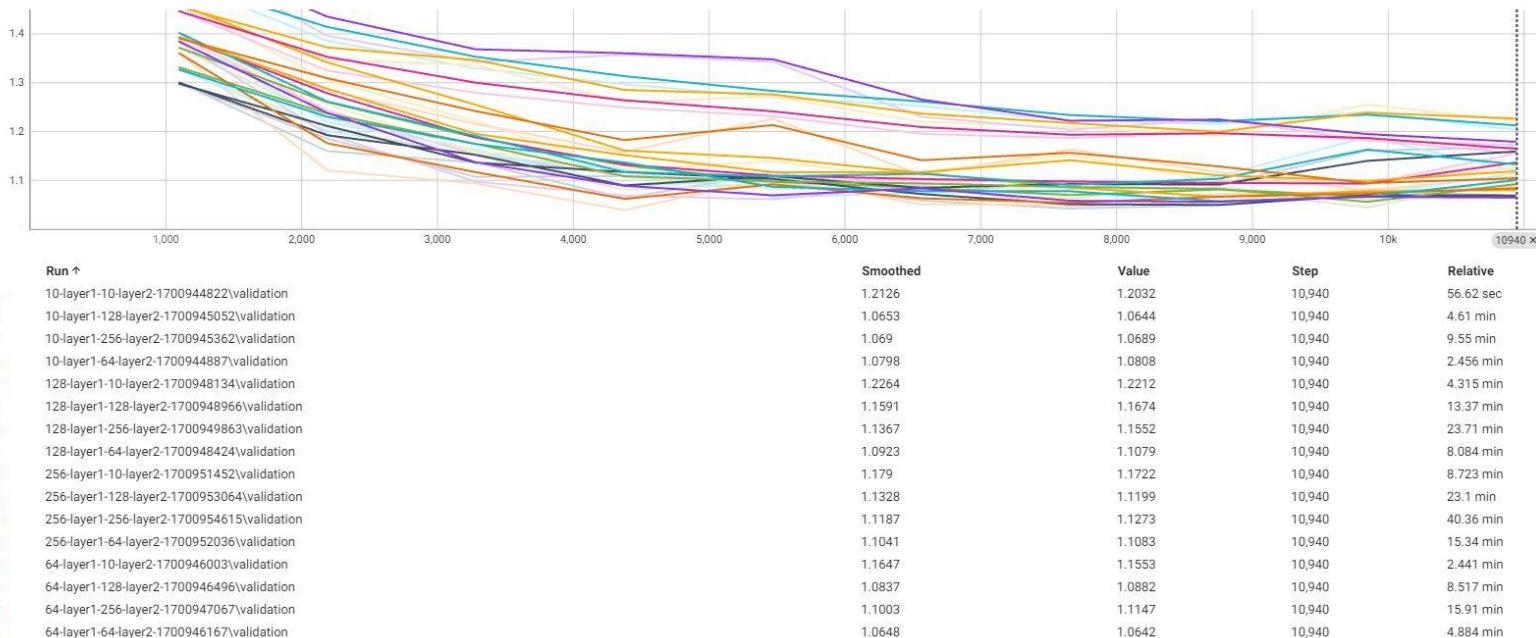


Figure 3.5 Model loss during evaluation across iterations

Στα παραπάνω διάγραμμα παρατηρούμε ότι τα δίκτυα που έχουν μεγάλες διαφορές μεταξύ πρώτου και δεύτερου layer χρειάζονται περισσότερο χρόνο να σταθεροποιηθούν. Επιπλέον, τα δίκτυα αυτά έχουν περισσότερες διακυμάνσεις και οροπέδια και άρα μεγαλύτερη αστάθεια.

4. Παραδείγματα ορθών και λανθασμένων κατηγοριοποιήσεων

Οι κώδικές των παραδειγμάτων κατηγοριοποίησης βρίσκονται στα αρχεία `nn_first_experiment_predictions.py`, `nn_my_predictions.py` και `nn_second_experiment_predictions.py`

Για τις δύο περιπτώσεις που μελετήσαμε μερικά παραδείγματα σωστών και λανθασμένων κατηγοριοποιήσεων των μοντέλων που είχαν την υψηλότερη ακρίβεια και την χαμηλότερη απώλεια είναι τα παρακάτω:

4.1 Κατηγοριοποιήσεις πρώτου πειράματος

Μερικά παραδείγματα κατηγοριοποίησης δεδομένων CIFAR-10 του μοντέλου `256-layer1-256-layer2-1700864410\train`, όπου έχουμε Dense hidden layers είναι τα εξής:

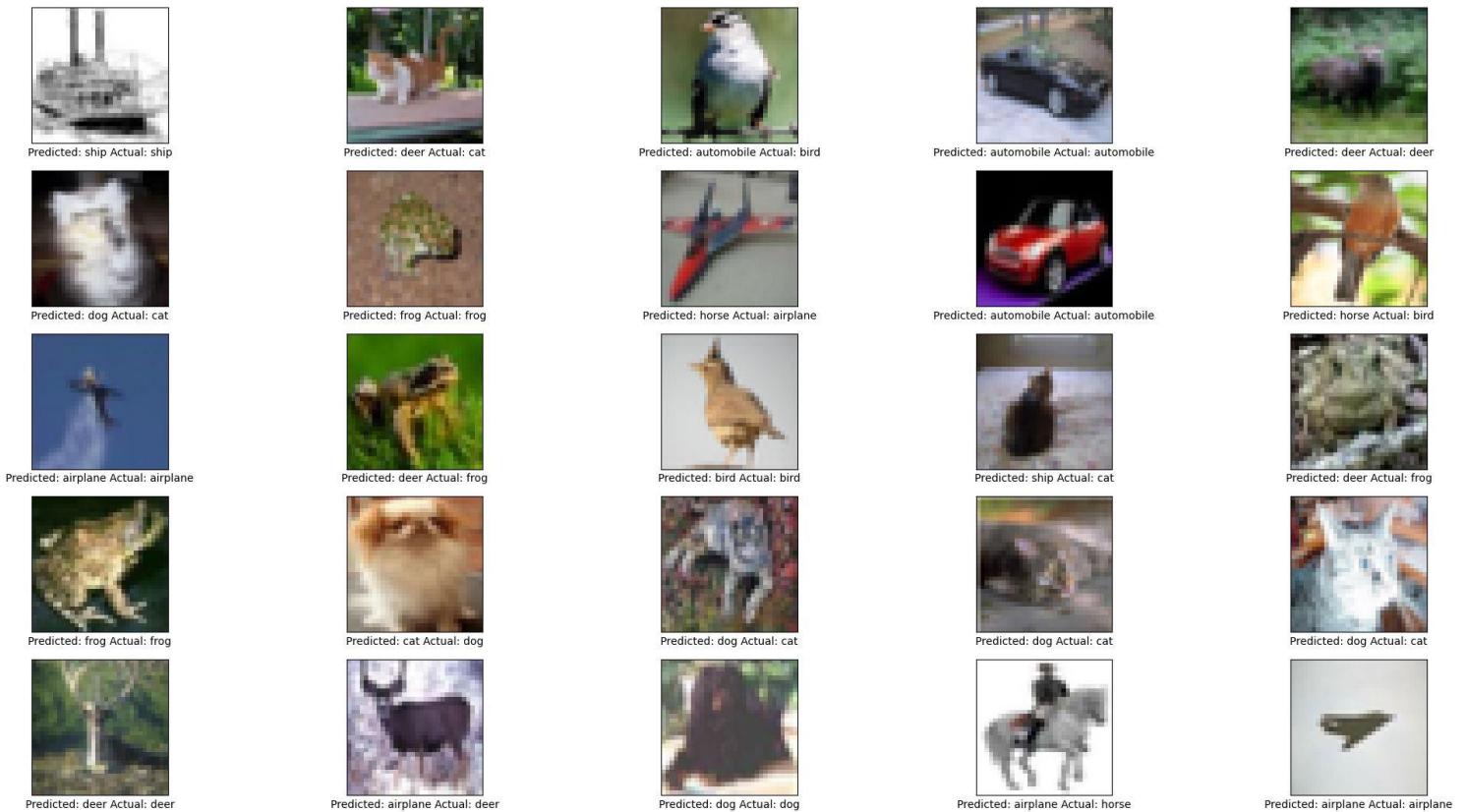


Figure 4.1.1 Predictions of the first network

4.2 Κατηγοριοποιήσεις δευτέρου πειράματος

Μερικά παραδείγματα κατηγοριοποίησης δεδομένων CIFAR-10 του μοντέλου `128-layer1-128-layer2-1700864410\train`, όπου έχουμε Convolutional hidden layers είναι τα εξής:

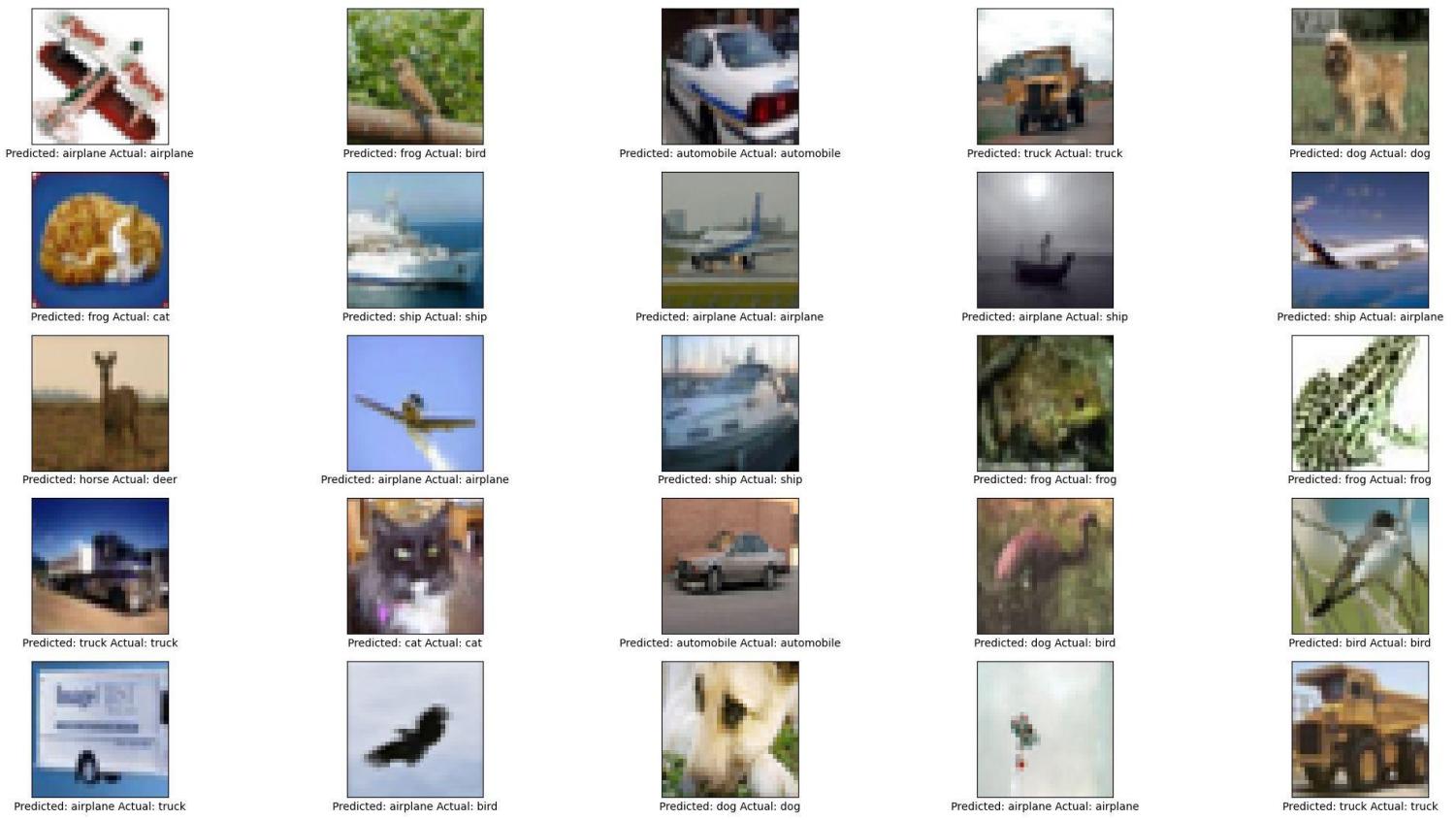


Figure 4.2.1 Predictions of the second network

4.3 Κατηγοριοποιήσεις εικόνων εκτός CIFAR-10

Προκειμένου να ελέγξουμε την αποδοτικότητα των δικτύων μας αποφασίσαμε να βάλουμε ως είσοδο τους δικές μας εικόνες. Θεωρούμε όπου “first network” το 256-layer1-256-layer2-1700864410\train με dense hidden layers νευρωνικό δίκτυο και όπου “second network” το 128-layer1-128-layer2-1700864410\train με Convolutional hidden layers δίκτυο. Τα αποτελέσματα του τεστ μας φαίνονται παρακάτω:

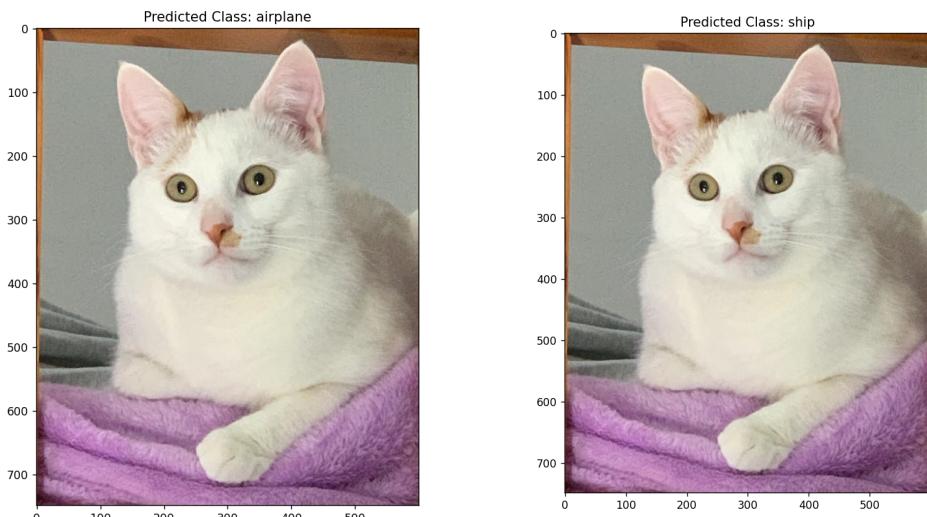


Figure 4.3.1 Predictions of first network (left) and second network (right) of a cat

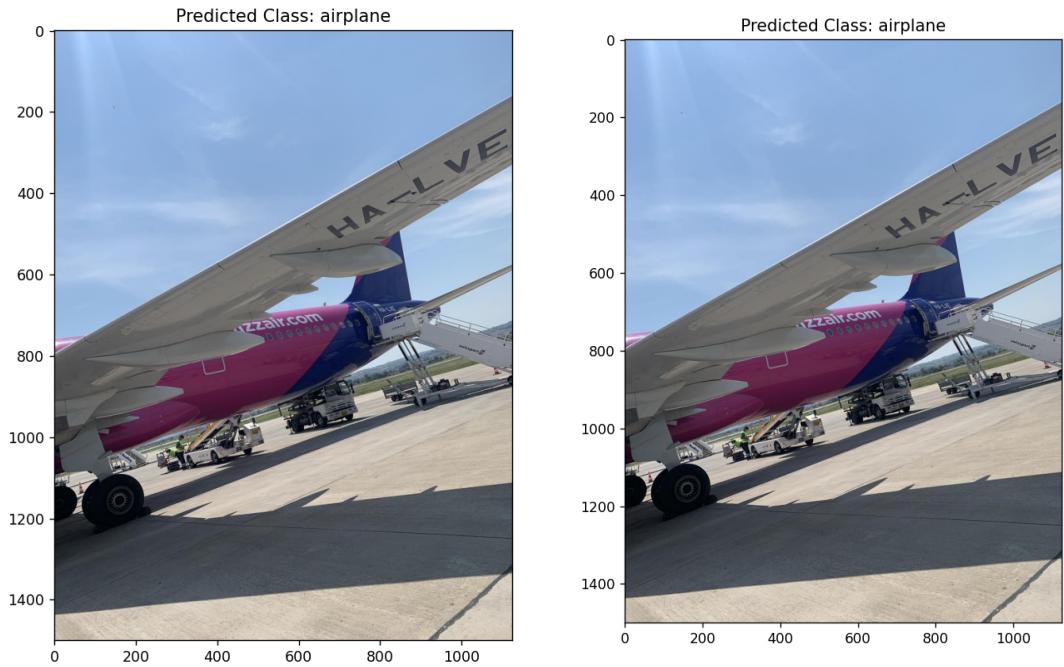


Figure 4.3.2 Predictions of first network (left) and second network (right) of an airplane

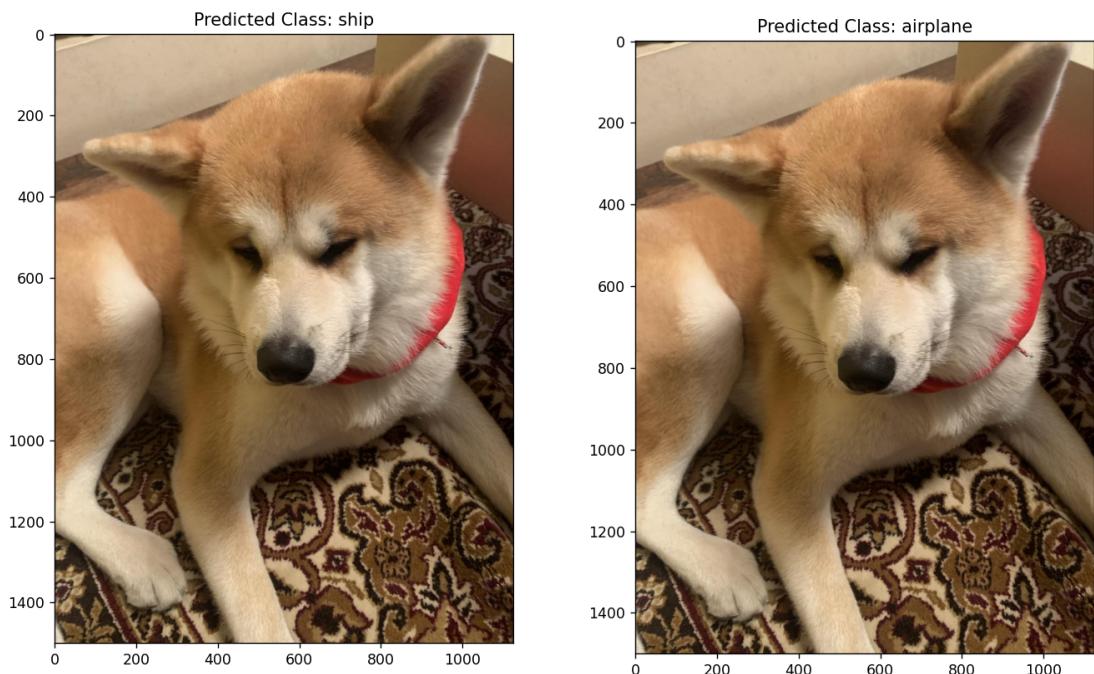


Figure 4.3.3 Predictions of first network (left) and second network (right) of a dog

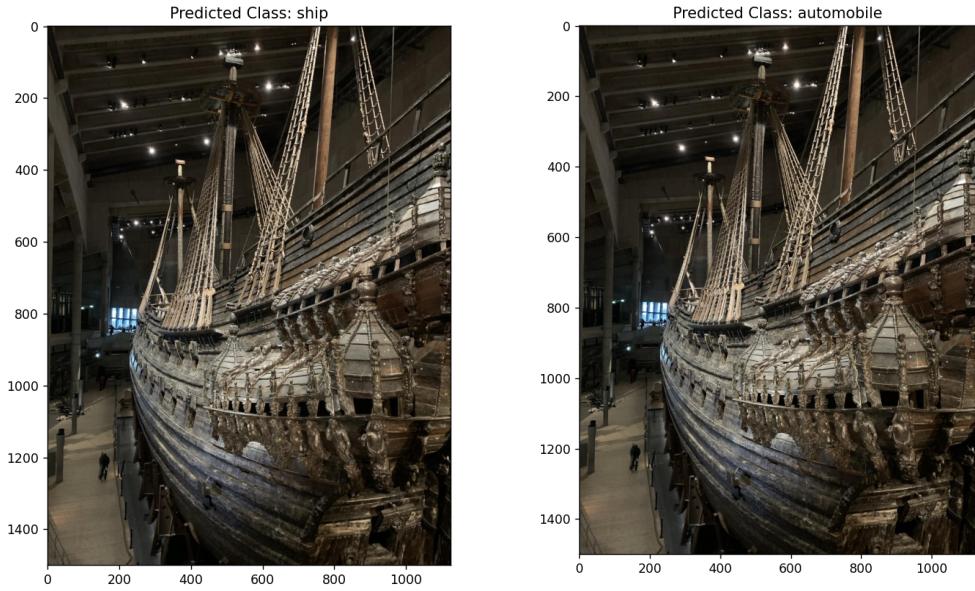


Figure 4.3.4 Predictions of first network (left) and second network (right) of a ship

5. Κατηγοριοποίηση Πλησιέστερου γείτονα

Ο κώδικας για την κατηγοριοποίηση πλησιέστερου γείτονα υλοποιήθηκε στο αρχέιο `knn_and_nearest_center_classifiers.py`.

Τα αποτελέσματα από την κατηγοριοποίηση για $k=1$ για το CIFAR-10 Dataset είναι τα παρακάτω:

Accuracy of KNN Classifier with k = 1 : 35.39%				
Confusion Matrix for k= 1 :				
[[485 16 109 20 73 20 31 11 221 14]				
[107 218 81 55 131 54 74 29 204 47]				
[84 0 384 61 244 57 94 16 55 5]				
[44 5 168 240 152 148 140 41 49 13]				
[58 4 216 51 457 48 85 25 54 2]				
[49 3 162 144 148 290 112 30 51 11]				
[27 2 193 79 239 56 353 11 36 4]				
[64 12 143 62 201 66 86 294 56 16]				
[117 19 50 42 77 22 24 13 619 17]				
[109 57 78 69 112 37 76 56 207 199]]				
The classification Report for k = 1 :				
	precision	recall	f1-score	support
airplane	0.42	0.48	0.45	1000
automobile	0.65	0.22	0.33	1000
bird	0.24	0.38	0.30	1000
cat	0.29	0.24	0.26	1000
deer	0.25	0.46	0.32	1000
dog	0.36	0.29	0.32	1000
frog	0.33	0.35	0.34	1000
horse	0.56	0.29	0.39	1000
ship	0.40	0.62	0.49	1000
truck	0.61	0.20	0.30	1000
accuracy			0.35	10000
macro avg	0.41	0.35	0.35	10000
weighted avg	0.41	0.35	0.35	10000

Figure 5.1 Knn results for $k=1$

Ο ταξινομητής K-Nearest Neighbors (KNN) με $k=1$ πέτυχε συνολική ακρίβεια 35,39% στο σύνολο δεδομένων CIFAR-10. Ο confusion πίνακας παρέχει πληροφορίες για την απόδοση του ταξινομητή σε διαφορετικές κλάσεις. Για παράδειγμα, δείχνει ότι το μοντέλο είχε σχετικά

μεγαλύτερη ακρίβεια στην ταξινόμηση «πλοίων» (62%) και «αεροπλάνων» (48%), αλλά δυσκολεύτηκε με κατηγορίες όπως «αυτοκίνητο» (22%) και «πουλί» (38%).

Η αναφορά ταξινόμησης παρέχει μια πιο λεπτομερή αξιολόγηση, συμπεριλαμβανομένης της ακρίβειας, της ανάκλησης και της βαθμολογίας F1 για κάθε τάξη. Η ακρίβεια μετρά την ακρίβεια των θετικών προβλέψεων, η ανάκληση αξιολογεί την ικανότητα καταγραφής όλων των θετικών περιπτώσεων και η βαθμολογία F1 είναι ο αρμονικός μέσος όρος ακρίβειας και ανάκλησης.

Τα αποτελέσματα από την κατηγοριοποίηση για $k=3$ για το CIFAR-10 Dataset είναι τα παρακάτω:

Accuracy of KNN Classifier with k = 3 : 33.03%									
Confusion Matrix for k= 3 :									
[[573 9 109 19 48 6 22 5 204 5]									
[196 243 125 61 113 29 37 6 168 22]									
[156 7 450 54 197 30 59 7 39 1]									
[102 11 281 225 147 91 89 14 33 7]									
[104 5 291 55 436 21 36 12 40 0]									
[100 5 249 172 144 213 68 11 34 4]									
[49 5 327 89 252 27 229 2 20 0]									
[131 20 217 73 235 38 41 199 44 2]									
[183 30 52 42 51 14 9 3 612 4]									
[193 87 128 76 110 27 43 13 200 123]]									
The classification Report for k = 3 :									
	precision	recall	f1-score	support					
airplane	0.32	0.57	0.41	1000					
automobile	0.58	0.24	0.34	1000					
bird	0.20	0.45	0.28	1000					
cat	0.26	0.23	0.24	1000					
deer	0.25	0.44	0.32	1000					
dog	0.43	0.21	0.28	1000					
frog	0.36	0.23	0.28	1000					
horse	0.73	0.20	0.31	1000					
ship	0.44	0.61	0.51	1000					
truck	0.73	0.12	0.21	1000					
accuracy			0.33	10000					
macro avg	0.43	0.33	0.32	10000					
weighted avg	0.43	0.33	0.32	10000					

Figure 5.2 Knn results for k=3

Ο ταξινομητής K-Nearest Neighbors (KNN) με $k=3$ πέτυχε συνολική ακρίβεια 33,03% στο σύνολο δεδομένων CIFAR-10. Η ανάλυση του confusion πίνακα παρέχει πληροφορίες για την απόδοση του ταξινομητή σε διαφορετικές κλάσεις. Οι αξιοσημείωτες παρατηρήσεις περιλαμβάνουν σχετικά υψηλότερη ακρίβεια στην ταξινόμηση "πλοίων" (61%) και "αυτοκίνητων" (57%), ενώ αντιμετωπίζουμε προκλήσεις με κατηγορίες όπως "γάτα" (23%) και "άλογο" (20%).

6. Κατηγοριοποίηση Πλησιέστερου ταξινομητή κέντρου

Ο κώδικας για την κατηγοριοποίηση πλησιέστερου γείτονα υλοποιήθηκε στο αρχείο `knn_and_nearest_center_classifiers.py`.

Τα αποτελέσματα από τον Nearest Center Classifier είναι τα παρακάτω:

Accuracy of Nearest Center Classifier: 27.74%									
Confusion Matrix for Nearest Center Classifier:									
[[539 54 23 14 19 40 73 18 146 74]									
[149 186 11 18 30 56 254 39 87 170]									
[254 46 107 12 65 78 310 58 29 41]									
[171 31 38 56 29 185 313 75 21 81]									
[107 30 78 14 119 101 394 69 29 59]									
[167 22 45 40 37 286 246 68 40 49]									
[114 34 44 19 23 90 538 82 5 51]									
[148 46 19 20 78 107 192 166 46 178]									
[215 90 7 9 11 92 55 12 370 139]									
[152 118 6 9 21 30 110 35 112 407]]									
The Classification Report for Nearest Center Classifier:									
	precision	recall	f1-score	support					
airplane	0.27	0.54	0.36	1000					
automobile	0.28	0.19	0.22	1000					
bird	0.28	0.11	0.16	1000					
cat	0.27	0.06	0.09	1000					
deer	0.28	0.12	0.17	1000					
dog	0.27	0.29	0.28	1000					
frog	0.22	0.54	0.31	1000					
horse	0.27	0.17	0.20	1000					
ship	0.42	0.37	0.39	1000					
truck	0.33	0.41	0.36	1000					
accuracy			0.28	10000					
macro avg	0.29	0.28	0.25	10000					
weighted avg	0.29	0.28	0.25	10000					

Figure 6.1 Nearest Center Classifier results

Ο Ταξινομητής πλησιέστερου κέντρου πέτυχε συνολική ακρίβεια 27,74% στο σύνολο του CIFAR-10 Dataset. Η εξέταση του confusion πίνακα παρέχει πληροφορίες για την απόδοσή του σε διαφορετικές κλάσεις. Οι αξιοσημείωτες παρατηρήσεις περιλαμβάνουν σχετικά υψηλότερη ακρίβεια στην ταξινόμηση "αεροπλάνο" (54%) και "βάτραχος" (54%), ενώ αντιμετωπίζουμε προκλήσεις με κατηγορίες όπως "γάτα" (6%) και "πουλί" (11%).

Επιπλέον, στο report αναλύονται περαιτέρω οι μετρήσεις αξιολόγησης για κάθε τάξη. Η ακρίβεια, η ανάκλαση και η βαθμολογία F1 αποκαλύπτουν σε ποιά σημεία ο ταξινομητής επιτυγχάνει καλύτερα αποτελέσματα από άλλα.

7. Συμπεράσματα

Στο πλαίσιο αυτής της εργασίας, διερευνήσαμε διάφορες προσεγγίσεις μηχανικής μάθησης για την ταξινόμηση των εικόνων του CIFAR-10 Dataset. Η ανάλυση μας εστιάστηκε σε δύο βασικές κατηγορίες μοντέλων: τα Νευρωνικά Δίκτυα και οι Ταξινομητές Κοντινότερων Γειτόνων.

Με τους πειραματισμούς μας με τα Νευρωνικά Δίκτυα ανακαλύψαμε πόσο σημαντική είναι η επιλογή της σωστής αρχιτεκτονικής με τη χρήση τόσο των Dense layers όσο και των Convolutional. Παρατηρήσαμε ότι τα CNN δίκτυα επέδειξαν εξαιρετική απόδοση στην αναγνώριση εικόνων του dataset, όμως απέτυχαν να κατηγοριοποιήσουν σωστά εικόνες εκτός αυτού. Παρόλο που χρησιμοποιήσαμε την τακτική για να αντιμετωπίσουμε το overfitting με την προσθήκη dropout η γενικοποίηση βελτιώθηκε σημαντικά, αλλά όχι όσο θα θέλαμε.

Επιπλέον, εφαρμόσαμε τόσο τον ταξινομητή k-Nearest Neighbors (KNN) όσο και τον ταξινομητή Nearest Center Classifier. Παρατηρήσαμε ότι ο KNN με $k=1$ πρόσφερε συγκρίσιμα αποτελέσματα με τα Νευρωνικά Δίκτυα, επιτυγχάνοντας 35,39% συνολική ακρίβεια. Η προσθήκη γειτονικών παραδειγμάτων ($k=3$) επηρέασε την απόδοση, αναδεικνύοντας τη σημασία της επιλογής του k . Από την άλλη πλευρά, ο Nearest Center Classifier παρουσίασε πιο μέτρια αποτελέσματα, επιτυγχάνοντας 27,74% συνολική ακρίβεια.

Αντίθετα, το δίκτυο που αποτελείται μόνο από Dense hidden layers ενώ είχε μικρότερη ακρίβεια και περισσότερες απώλειες, κατάφερε να ταξινομήσει με μεγαλύτερη επιτυχία τα δεδομένα εκτός του CIFAR-10 Dataset.

Βάσει των αποτελεσμάτων, προκύπτει ότι η επιλογή του μοντέλου εξαρτάται σημαντικά από τη φύση των δεδομένων και την πολυπλοκότητα του προβλήματος. Οι Νευρωνικοί Ταξινομητές ξεχωρίζουν σε προβλήματα όπως η ταξινόμηση εικόνων, ενώ οι Ταξινομητές Κοντινότερων Γειτόνων είναι αποτελεσματικοί και προσαρμόσιμοι σε διάφορα περιβάλλοντα, ανάλογα με την επιλογή των παραμέτρων τους.

8. Αναφορές

- <https://machinelearningmastery.com/build-multi-layer-perceptron-neural-network-models-keras/>
- https://www.tensorflow.org/tensorboard/get_started
- <https://aws.amazon.com/what-is/overfitting/>
- <https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5>
- https://www.tensorflow.org/tutorials/keras/save_and_load