ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΜΑΘΗΣΗΣ

Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα

Άσκηση 3: Βαθιά Μάθηση

Ομάδα (team_seed = 55):

Διολέτη Ίλια 03115055
Κυριάκου Αθηνά 03117405
Σουμπασάκου Αρτεμησία 03115061

Στόχος της άσκησης αυτή είναι να βελτιστοποιηθεί η απόδοση μοντέλων Βαθιάς Μάθησης στο σύνολο δεδομένων CIFAR-100 χρησιμοποιώντας το TensorFlow 2.0. Για να γίνει αυτό, προχωρήσαμε στον ορισμό μοντέλων, είτε εκ του μηδενός ("from scratch") είτε με μεταφορά μάθησης (transfer learning).

Στην εργασία αυτή έχει γίνει η παρουσίαση, η βελτιστοποίηση και η σύγκριση πέντε συνολικά μοντέλων, δύο from scratch και τριών transfer learning.

Έχει γίνει βελτιστοποίηση κάθε μοντέλου ως προς τον εαυτό του, εξετάζοντας την μεταβολή διαφόρων παραμέτρων και πώς οι μεταβολές αυτές επηρεάζουν τις μετρικές ενδιαφέροντός μας (accuracy, loss, training time). Οι τομείς ενδιαφέροντος πάνω στους οποίους έγινε η μελέτη ήταν: υπερεκπαίδευση, χρόνος εκπαίδευσης, optimizers, αριθμός κατηγοριών, learning rate, batch size, μνήμη, κλάσεις, επίπεδα εκπαίδευσης.

Αφού γίνει η βελτιστοποίηση του κάθε μοντέλου, συγκρίνονται οι δύο κατηγορίες μοντέλων μεταξύ τους, σύμφωνα με τις τελικές τιμές των μετρικών βελτιστοποίησης.

Η δομή της αναφοράς είναι η εξής:

1. **Βελτιστοποίηση Μοντέλων From Scratch**
    1. **Μοντέλο 1**
    2. **Μοντέλο 2**
2. **Βελτιστοποίηση Μοντέλων Transfer Learning**
    1. **VGG16**
    2. **DenseNet 169**
    3. **Mobile Net**
3. **Σύγκριση Μοντέλων From Scratch και Μοντέλων Transfer Learning**

4. References

Βελτιστοποίηση Μοντέλων From Scratch

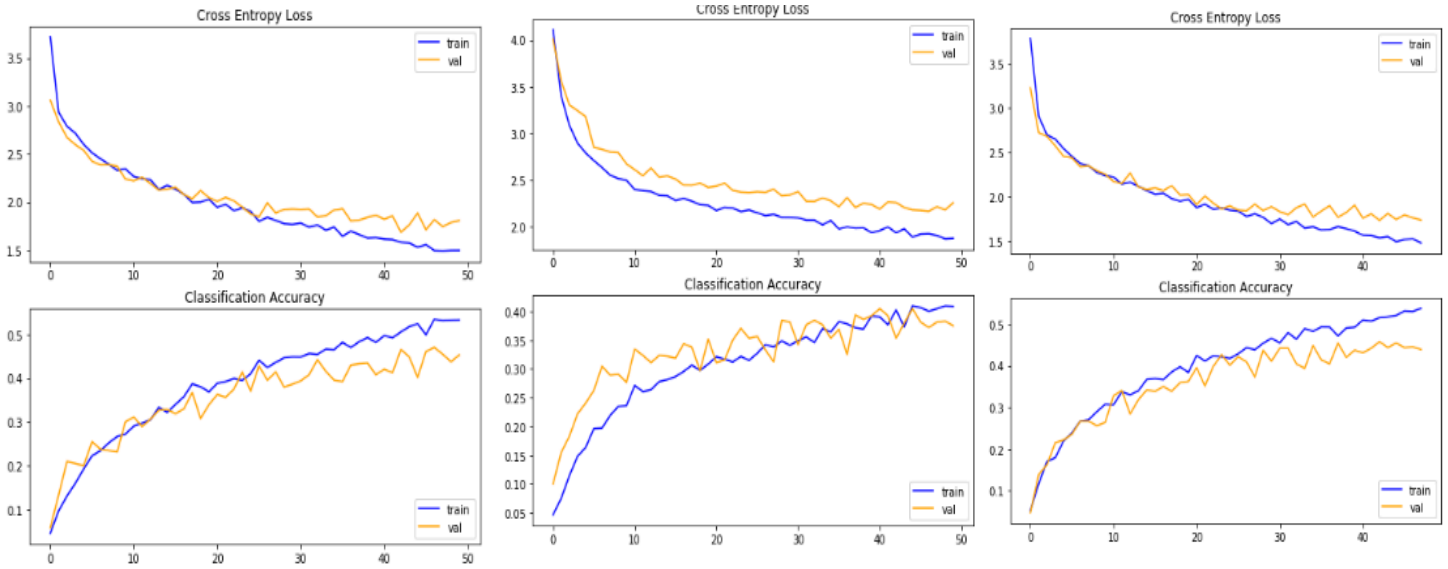
Μοντέλο 1:

Πρόκειται για ένα μοντέλο from scratch,το οποίο αποτελείται από μια συνελικτική βάση με 3 επίπεδα Conv2D ακολουθούμενα από MaxPooling2D επίπεδα. Δέχεται ως είσοδο tensors με σχήμα (32,32,3), δηλαδή αυτό των εικόνων στο CIFAR100. Στη συνέχεια προσθέτουμε 2 dense layers, με διαφορετικές συναρτήσεις ενεργοποίησης για να γίνει το classification. Προηγείται ένα flatten layer ώστε τα τρισδιάστατα δεδομένα που παρήχθησαν από τα προηγούμενα στρώματα να γίνουν μονοδιάστατα, που είναι η μορφή εισόδου που δέχονται τα dense layers. Το τελικό dense στρώμα έχει 100 εξόδους , όσες κατηγορίες δηλαδή έχει το CIFAR100

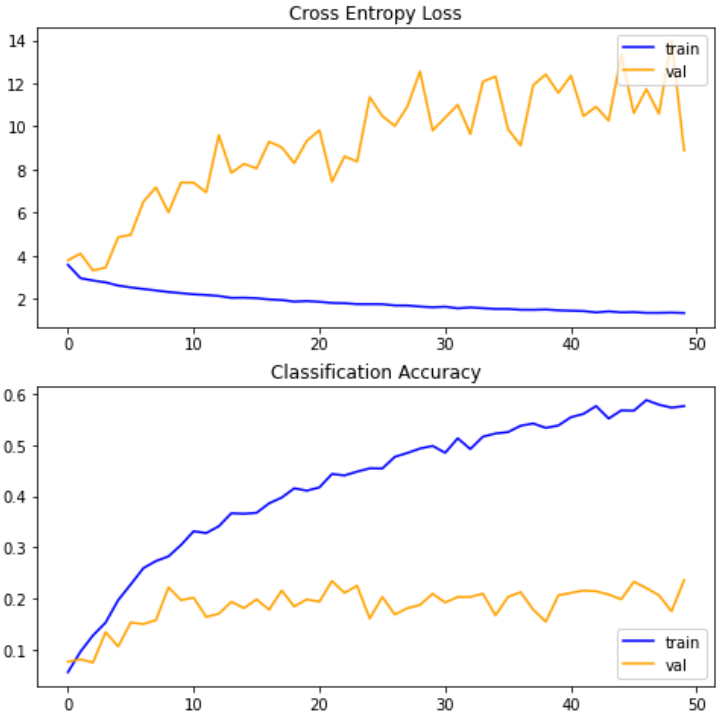
ΥΠΕΡΕΚΠΑΙΔΕΥΣΗ

Για τη μείωση της υπερεκπαίδευσης δοκιμάσαμε την προσθήκη ***dropout*** layers και ***early stopping*** . Βλέπουμε πως δεν συμβαίνει κάποια σημαντική αλλαγή σε αυτό το πολύ απλό μοντέλο. Εικόνα (από αριστερά στα δεξιά): αρχικά, με dropout, early stopping.

Το early stopping δεν σταματάει την εκπαίδευση μέχρι τις 50 εποχές.



Η τεχνική ***Data augmentation*** με horizontal flip και χωρίς ταυτόχρονη χρήση άλλων τεχνικών που περιγράφονται παραπάνω, έδωσε loss: 1.80, accuracy: 0.45.



ΧΡΟΝΟΣ ΕΚΠΑΙΔΕΥΣΗΣ

Οι αρχικές μετρικές που έχουν παρουσιαστεί και παραπάνω προκύπτουν όταν για την εκπαίδευση του μοντέλου γίνεται data prefetching. Αυτή η τεχνική επιτρέπει όσο η εκπαίδευση του μοντέλου βρίσκεται στο βήμα κ, το input pipeline διαβάζει τα δεδομένα για το επόμενο βήμα εκπαίδευσης, κ+1. Τα αποτελέσματα στον παρακάτω πίνακα αφορούν train για 50 εποχές, με adam optimizer (default learning rate).

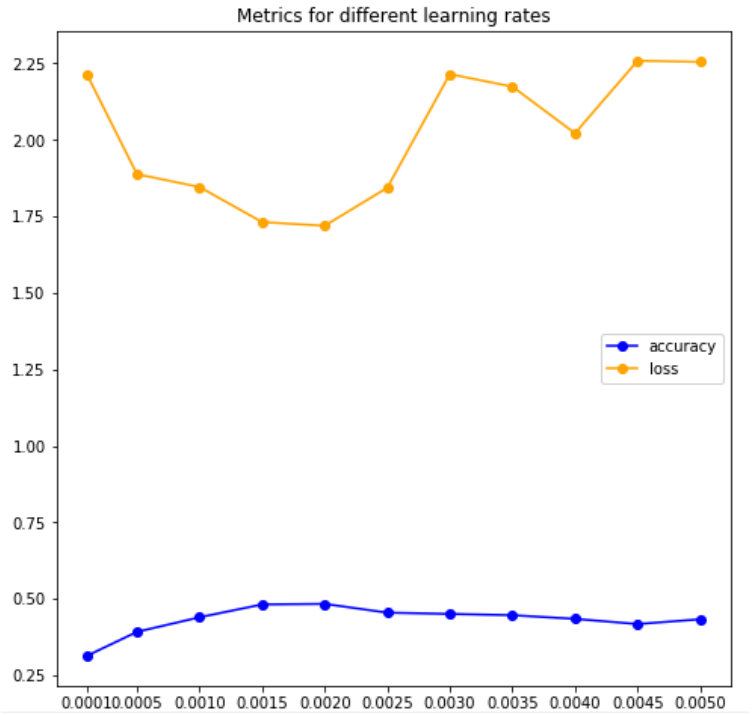
Τεχνική	**Χρόνος εκτέλεσης (seconds)**
Naive (batch - repeat -> shuffle)	μη επαρκής μνήμη για την εκτέλεση
Prefetching	17.2
Data reading parallelization	N/A (1 dataset loaded από 1 source)
Map transformation parallelization	N/A (δε χρησιμοποιούνται map operations)
Caching (shuffle - repeat -> batch)	38.2
Reducing Memory Footprint (shuffle - repeat -> batch)	21.0
Reducing Memory Footprint (repeat - shuffle -> batch)	22.4

ΕΠΙΔΡΑΣΗ ΕΠΙΛΟΓΗΣ OPTIMIZER

Ο έλεγχος έγινε με default παραμέτρους για κάθε optimizer (πχ. learning rate=0.001) και μας ενδιαφέρει περισσότερο ποιοτικά η επίδραση στις μετρικές που αναφέρονται.

	Adam	**Adadelta**	**Adagrad**	**Adamax**	**Nadam**	**RMSprop**	**SGD**
Accuracy	0.45	0.05	0.13	0.39	0.45	0.45	0.22
loss	1.80	4.37	2.86	1.94	1.80	1.85	2.58

Για τον Adam optimizer έγιναν δοκιμές για να διαπιστωθεί πώς επηρεάζονται η απόδοση και loss από διαφορετικά learning rates.



Παρατηρούμε πως για learning rates 0.0015-0.002 έχουμε ταυτόχρονα από τις χαμηλότερες τιμές loss και μεγαλύτερες του accuracy. Αυτά τα learning rates είναι τα βέλτιστα για τον Adam optimizer στο συγκεκριμένο μοντέλο.

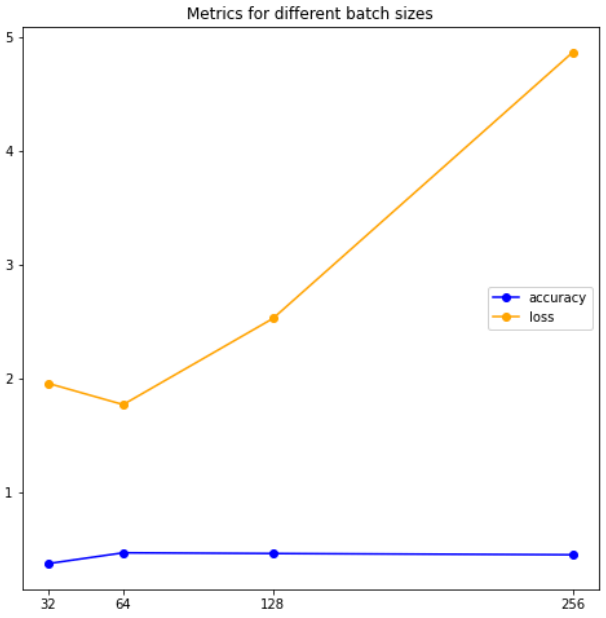
ΕΠΙΔΡΑΣΗ ΑΡΙΘΜΟΥ ΚΑΤΗΓΟΡΙΩΝ

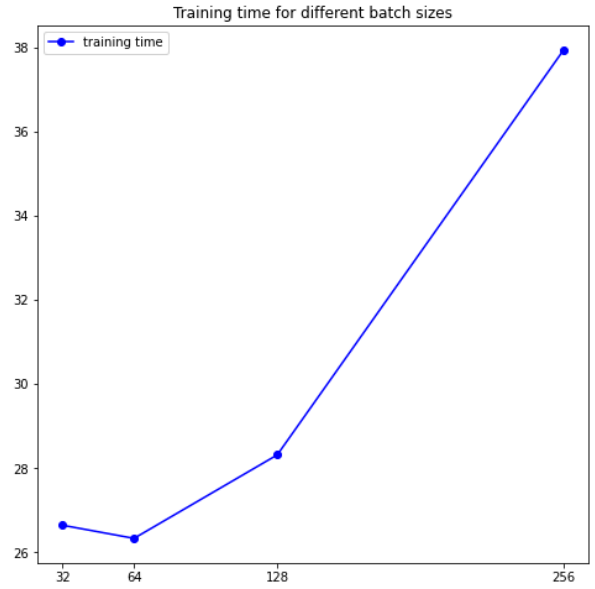
Ως τώρα εργαστήκαμε με τον προεπιλεγμένο αριθμό κατηγοριών, που είναι 20. Αυξάνοντας αυτό τον αριθμό αυξάνεται ο χρόνος εκπαίδευσης, πράγμα

αναμενόμενο εφόσον το training περιλαμβάνει τη δημιουργία του dataset με prefetch,shuffle,repeat, και εργαζόμαστε με μεγαλύτερο όγκο δεδομένων.

Αριθμός κατηγοριών	**20**	**40**	**60**	**80**
Accuracy	0.46	0.47	0.45	0.46
loss	1.77	1.77	1.76	1.80
trainning time (sec)	26.7	35.0	51.4	61.3

ΕΠΙΔΡΑΣΗ BATCH SIZE





Φαίνεται πως για batch size 64, ο χρόνος εκπαίδευσης είναι ο μικρότερος που παρατηρήθηκε, το accuracy είναι μέγιστο και το loss είναι ελάχιστο. Επομένως στο συγκεκριμένο dataset και μοντέλο batch μεγέθους 64 είναι το βέλτιστο.

ΣΥΜΠΕΡΙΦΟΡΑ ΑΝΑ ΚΛΑΣΗ

	precision	recall	f1-score	support
2	0.42	0.37	0.39	100
6	0.53	0.62	0.57	100
18	0.53	0.52	0.52	100
24	0.62	0.75	0.68	100
26	0.34	0.37	0.35	100
29	0.49	0.38	0.43	100
35	0.47	0.45	0.46	100
42	0.45	0.40	0.43	100
45	0.31	0.31	0.31	100
51	0.51	0.42	0.46	100
55	0.22	0.22	0.22	100
58	0.70	0.63	0.66	100
62	0.70	0.77	0.73	100
65	0.32	0.45	0.37	100
72	0.37	0.30	0.33	100
80	0.24	0.29	0.26	100
82	0.79	0.73	0.76	100
84	0.32	0.34	0.33	100
88	0.37	0.43	0.40	100
97	0.60	0.38	0.47	100

[' baby', ' bee', ' caterpillar', ' cockroach', ' crab', ' dinosaur', ' girl', ' leopard', ' lobster', ' mushroom', ' otter', ' pickup_truck', ' poppy', ' rabbit', ' seal', ' squirrel', ' sunflower', ' table', ' tiger', ' wolf']

Μοντέλο 2:

Πρόκειται για ένα μοντέλο from scratch, με 3x 2 διαδοχικά επίπεδα Conv2D με συνάρτηση ενεργοποίησης Exponential Linear Unit, ακολουθούμενα από ένα maxPooling επίπεδο. Αυτά διαδέχονται 2 dense επίπεδα, αφού φυσικά προηγηθεί ένα στρώμα flatten.

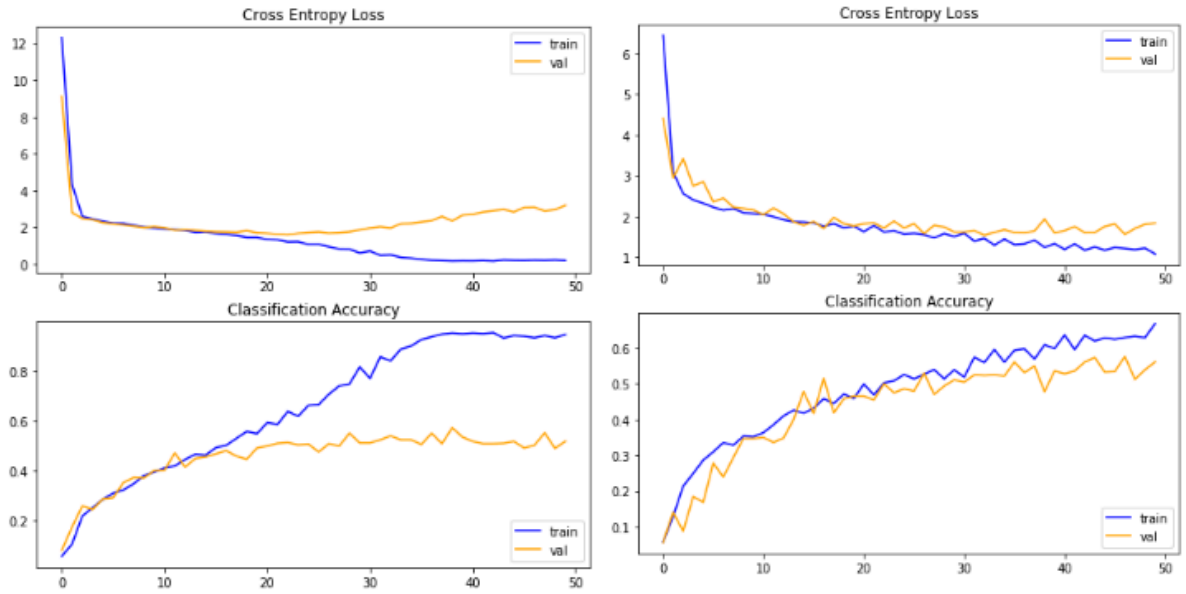
ΥΠΕΡΕΚΠΑΙΔΕΥΣΗ

Για τη μείωση της υπερεκπαίδευσης αρχικά δοκιμάσαμε την προσθήκη ***dropout*** layers.

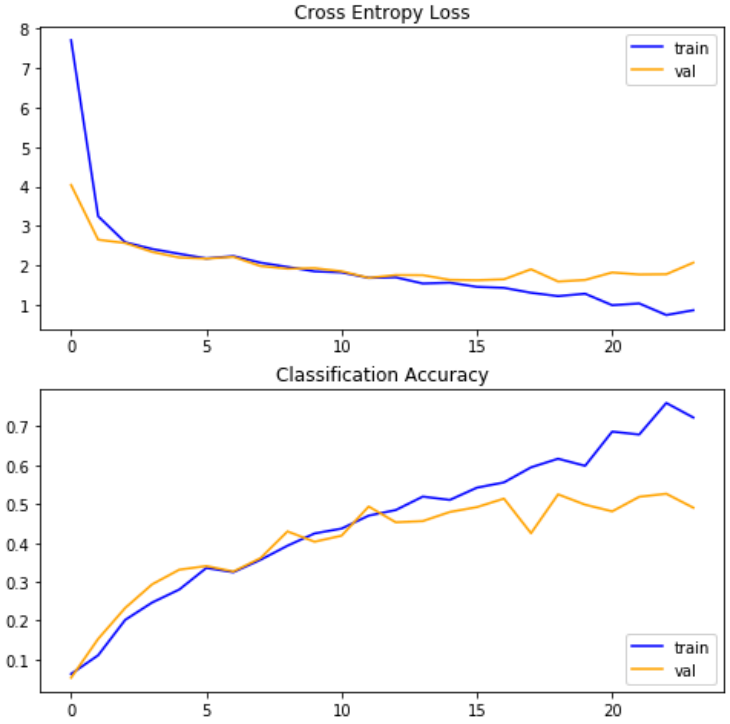
Αρχικά χωρίς dropout (αριστερά) βλέπουμε πως μετά από 20 περίπου εποχές το validation loss σταματάει να μειώνεται, ενώ το validation accuracy μετά από 20 περίπου εποχές φτάνει σε ένα πλατώ.

Με χρήση dropout επιπέδων (δεξιά) το validation loss είναι πολύ χαμηλότερο (1,77 vs 3,21). Επιτεύχθηκε μεγαλύτερη και αυξανόμενη validation accuracy και το

μοντέλο οδηγήθηκε σε σύγκλιση γρηγορότερα. Η ακρίβεια στο test set δεν διαφέρει πολύ από αυτή που πετύχαμε χωρίς dropout (0.56 vs 0.52), πιθανόν λόγω περιορισμένου αριθμού δειγμάτων.

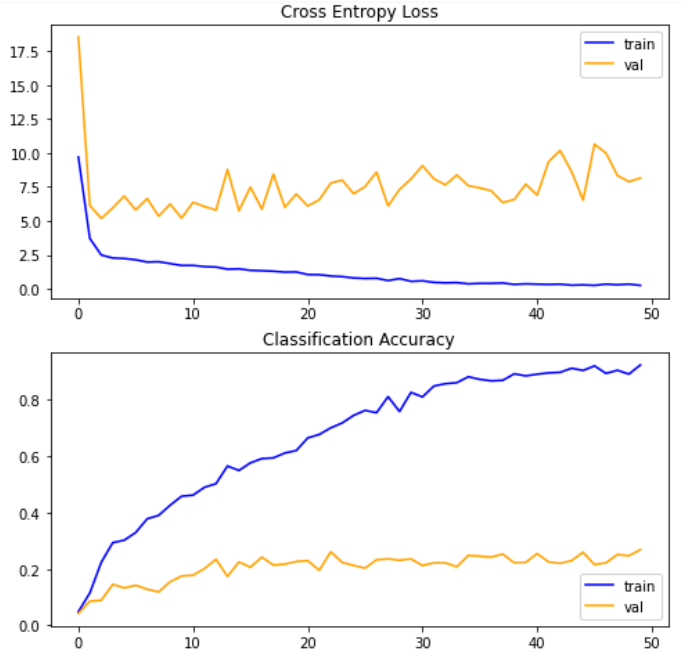


Με τη μέθοδο **early stopping :** loss: 1.91, accuracy: 0.51. Σταματάει η εκπαίδευση μετά από ~30 εποχές.



Μετά από ***data augmentation***, χωρίς μεθόδους dropout/early stopping και με horizontal flip, οι τιμές των μετρικών ήταν: loss: 2.85, accuracy: 0.50.

Και οι αντίστοιχες γραφικές παραστάσεις φαίνονται παρακάτω:



ΧΡΟΝΟΣ ΕΚΠΑΙΔΕΥΣΗΣ

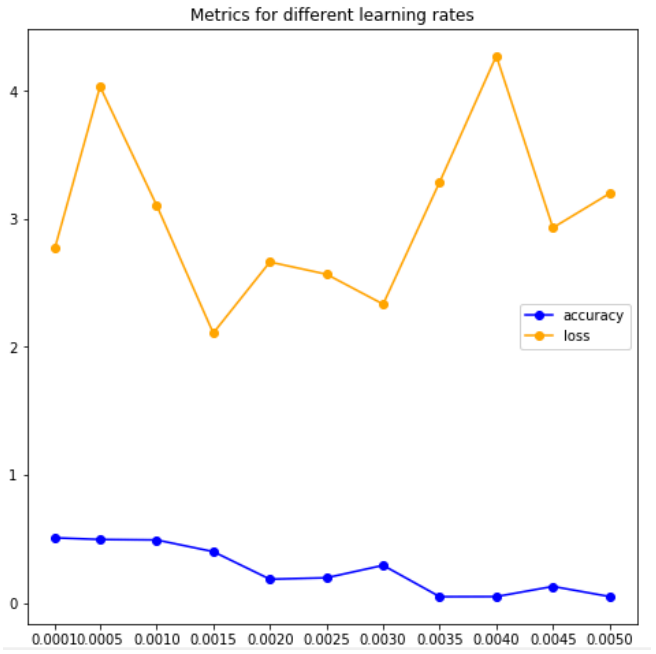
Τεχνική	**Χρόνος εκτέλεσης (seconds)**
Naive (batch - repeat -> shuffle)	μη επαρκής μνήμη για την εκτέλεση
Prefetching	59.8
Data reading parallelization	N/A (1 dataset loaded από 1 source)
Map transformation parallelization	N/A (δε χρησιμοποιούνται map operations)
Caching (shuffle - repeat -> batch)	69.7
Reducing Memory Footprint (shuffle - repeat -> batch)	61.5
Reducing Memory Footprint (repeat - shuffle -> batch)	62.2

ΕΠΙΔΡΑΣΗ ΕΠΙΛΟΓΗΣ OPTIMIZER

Ο έλεγχος έγινε με default παραμέτρους για κάθε optimizer (πχ. learning rate=0.001) και μας ενδιαφέρει περισσότερο ποιοτικά η επίδραση στις μετρικές που αναφέρονται.

	Adam	Adadelta	Adagrad	Adamax	Nadam	RMSprop	SGD
Accuracy	0.52	0.17	0.24	0.53	0.10	0.08	0.40
loss	2.94	2.85	2.49	2.69	3.13	2.90	1.97

Για τον Adam optimizer έγιναν δοκιμές για να διαπιστωθεί πώς επηρεάζονται η απόδοση και loss από διαφορετικά learning rates.

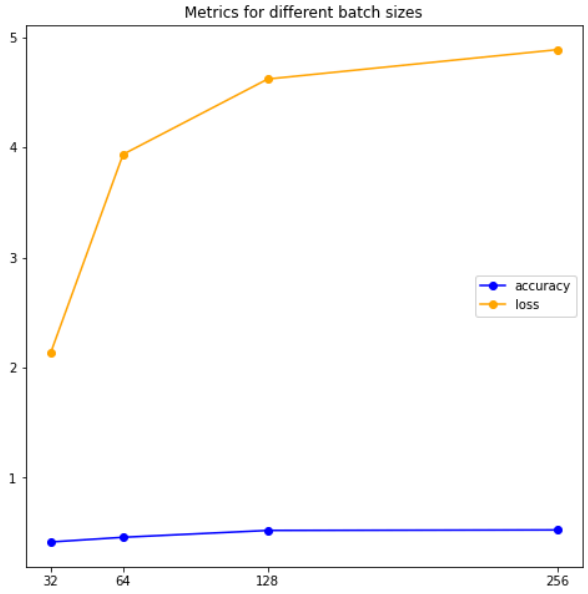


Είναι εμφανές πως δεν μπορούμε να πούμε ότι είναι καλύτερο ένα μεγαλύτερο ή μικρότερο learning rate, αλλά πρέπει να αναφερόμαστε σε συγκεκριμένες τιμές. Τιμές ανάμεσα στο 0.001-0.0015 φαίνεται να δίνουν καλύτερες τιμές στις μετρικές που εξετάζουμε.

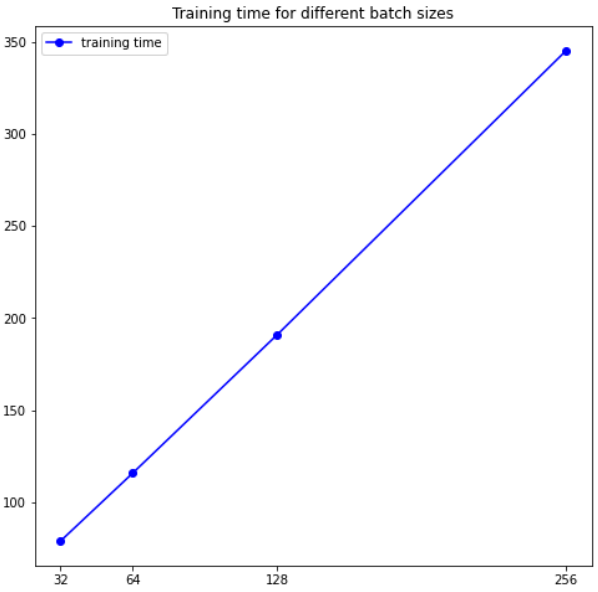
ΕΠΙΔΡΑΣΗ ΑΡΙΘΜΟΥ ΚΑΤΗΓΟΡΙΩΝ

Αριθμός κατηγοριών	**20 (default)**	**40**	**60**	**80**
Accuracy	0.51	0.46	0.33	0.44
loss	3.45	3.32	2.66	2.44
training time (sec)	193.4	200.4	215.7	232.2

ΕΠΙΔΡΑΣΗ BATCH SIZE



Παρατηρούμε πως το loss αυξάνεται μαζί με το batch size , ενώ το accuracy επηρεάζεται λιγότερο. Ο χρόνος εκπαίδευσης φαίνεται να αυξάνεται γραμμικά με το μέγεθος του batch.



ΣΥΜΠΕΡΙΦΟΡΑ ΑΝΑ ΚΛΑΣΗ

	precision	recall	f1-score	support
2	0.32	0.50	0.39	100
6	0.53	0.56	0.55	100
18	0.45	0.61	0.51	100
24	0.73	0.75	0.74	100
26	0.47	0.37	0.41	100
29	0.53	0.46	0.49	100
35	0.44	0.34	0.38	100
42	0.45	0.40	0.42	100
45	0.42	0.39	0.40	100
51	0.48	0.42	0.45	100
55	0.24	0.28	0.26	100
58	0.82	0.56	0.67	100
62	0.61	0.84	0.71	100
65	0.33	0.27	0.30	100
72	0.34	0.36	0.35	100
80	0.25	0.24	0.24	100
82	0.75	0.76	0.76	100
84	0.46	0.33	0.39	100
88	0.48	0.51	0.49	100
97	0.56	0.55	0.56	100

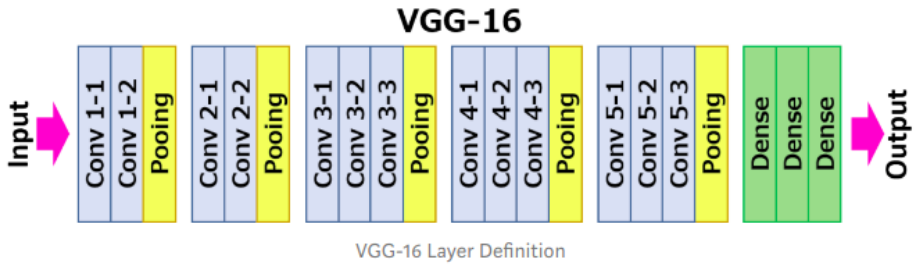
[' baby', ' bee', ' caterpillar', ' cockroach', ' crab', ' dinosaur', ' girl', ' leopard', ' lobster', ' mushroom', ' otter', ' pickup_truck', ' poppy', ' rabbit', ' seal', ' squirrel', ' sunflower', ' table', ' tiger', ' wolf']

Sunflower: best f1 score / squirrel: worst f1 score

Βελτιστοποίηση Μοντέλων Transfer Learning

VGG16:

Πρόκειται για VGG transfer learning model με 16 layers. Συγκεκριμένα, αποτελείται από 13 convolutional layers, 5 Max Pooling layers και 3 Dense layers εκ των οποίων μόνο τα 16 είναι layers βαρών. Τα layers ομαδοποιούνται σε blocks όπως φαίνεται στην παρακάτω εικόνα.



VGG-16 Layer Definition

Η υλοποίηση που έχει δοθεί χρησιμοποιεί το προεκπαιδευμένο μοντέλο VGG16 του [tf.keras.applications] με τις ακόλουθες τροποποιήσεις:

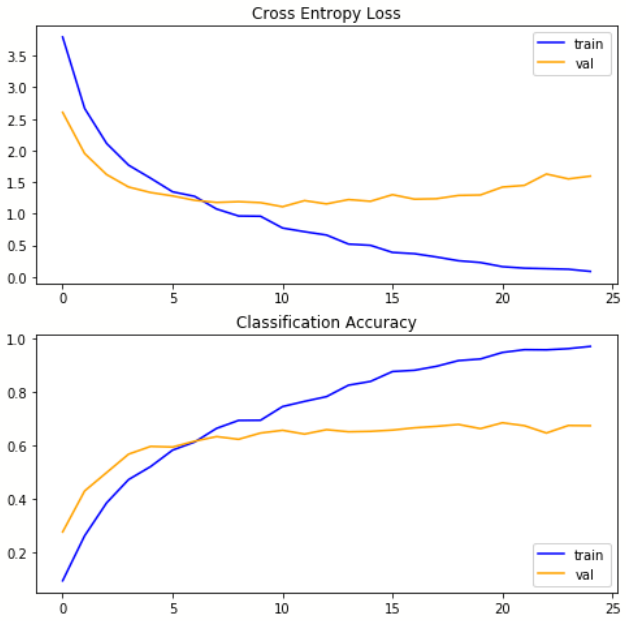
- δεν περιλαμβάνει τα 3 πλήρως διασυνδεδεμένα layers (dense) στην κορυφή του δικτύου (include_top=False)
- έχουν γίνει unfreeze τα convolutional layers με αποτέλεσμα τα βάρη να ενημερώνονται κατά την εκπαίδευση του δικτύου (VGG16_MODEL.trainable = True)
- για να αποφευχθεί το overfitting έχουν προστεθεί ένα Dropout layer και ένα Global Average Pooling (GAP) layer
- έχει προστεθεί ένα layer για CIFAR100 classification

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 1, 1, 512)         14714688
_________________________________________________________________
dropout (Dropout)            (None, 1, 1, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 100)               51300
=================================================================
Total params: 14,765,988
Trainable params: 14,765,988
Non-trainable params: 0
```

Το compilation χρησιμοποιεί τον Adam optimizer με learning rate 0.00005 ενώ έχει πραγματοποιηθεί prefetching των δεδομένων με batch size 128 και autotune.

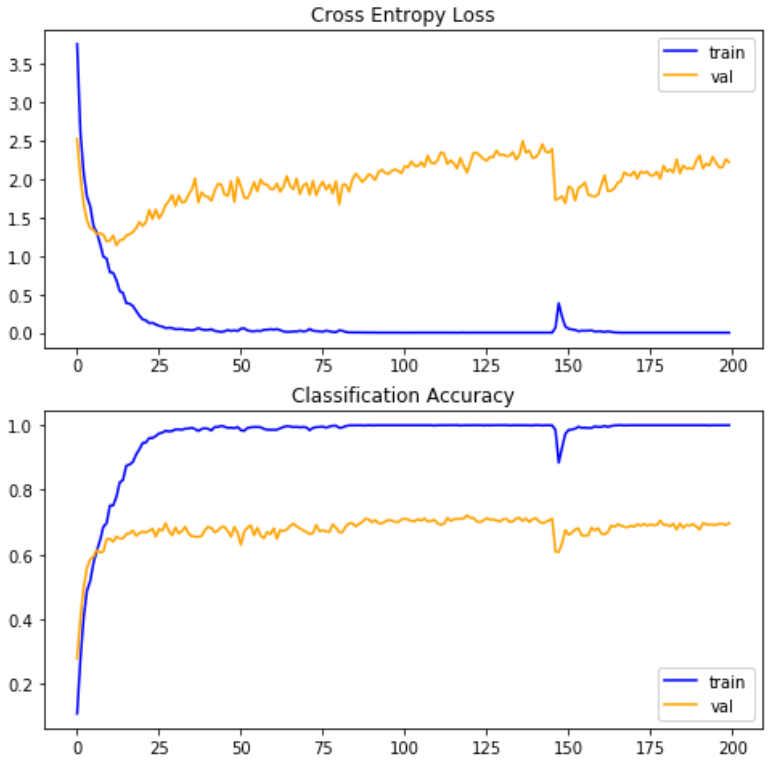
Με εκπαίδευση του μοντέλου σε 25 εποχές με 40 βήματα εκπαίδευσης και 10 βήματα validation ανά εποχή:

Παράμετρος βελτιστοποίησης (25 epochs)	**Τιμή**
Total parameters	14,765,988
Trainable parameters	14,765,988
Non-trainable parameters	0
Συνολικός χρόνος εκπαίδευσης	70.329
Loss	1.53
Accuracy	0.67



Για πιο εποπτικό σχολιασμό των αποτελεσμάτων, αυξάνονται οι εποχές εκπαίδευσης σε 200 στο υπάρχον μοντέλο.

Παράμετρος βελτιστοποίησης (200 epochs)	**Τιμή**
Loss	2.17
Accuracy	0.69



Παρατηρούμε ότι πέρα από τις 80 περίπου εποχές, η μετρική του accuracy δε βελτιώνεται περαιτέρω. Βελτίωση μικρότερη από 1% εμφανίζεται μετά τις 40 εποχές. Συνεπώς, με αυτό το threshold, δε χρειάζεται να αυξήσουμε περαιτέρω το πλήθος των εποχών.

Υπερεκπαίδευση (για 25 epochs, χρήση prefetching)

1. Μελέτη προστιθέμενων layers

Αρχικά ελέγχθηκαν τα επιπλέον layers που έχουν προστεθεί στο μοντέλο για την αποφυγή της υπερκπαίδευσης. Παρατηρείται πως και τα δύο προστιθέμενα layers συμβάλλουν στην αποφυγή της υπερεκπαίδευσης, με καθοριστικότερο το GPA.

Αφαίρεση layer	**Loss**	**Accuracy**
Dropout	1.73	0.65
GPA	1.50	0.05
Dropout και GPA	1.74	0.05

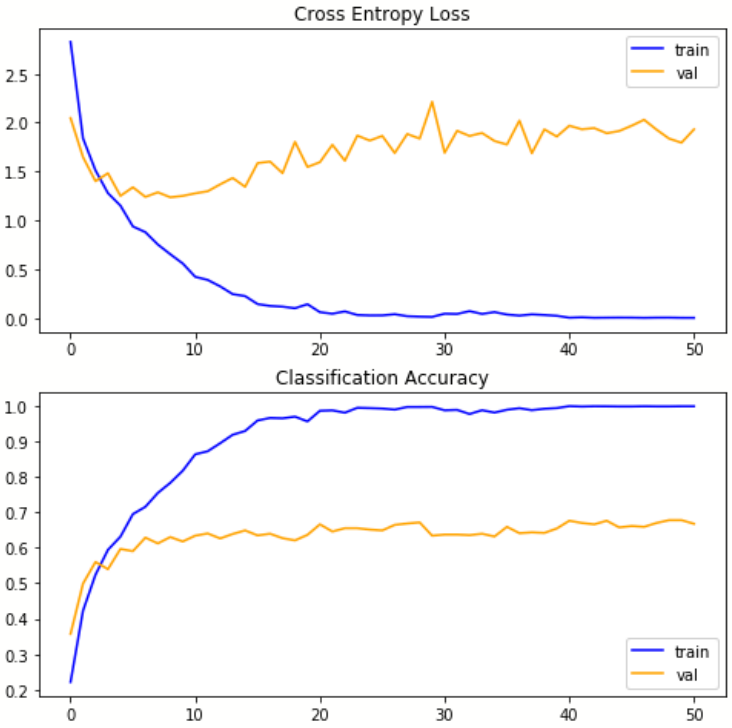
2. Χρήση Dropout layer

Παρατηρούμε ότι όσο αυξάνεται το rate του Dropout μέχρι περίπου την τιμή 0.8 το loss μειώνεται και το accuracy αυξάνεται. Ωστόσο, για τιμές rate [0.85,1), παρατηρείται ραγδαία μείωση του accuracy και αύξηση του loss, άρα και απώλεια της απαιτούμενης πληροφορίας. Σημειώνεται ότι όσο το rate αυξάνεται, δε μεταβάλλεται το πλήθος των παραμέτρων, αφού η συγκεκριμένη παράμετρος δεν εξαρτάται από το πλήθος των δεδομένων αλλά από την αρχιτεκτονική του μοντέλου. Ωστόσο, με την αύξηση του rate μειώνεται ο χρόνος εκτέλεσης, κάτι που είναι λογικό εφόσον μειώνεται το πλήθος των δεδομένων προς επεξεργασία.

	Loss	**Accuracy**	**Χρόνος εκτέλεσης (seconds)**
Χωρίς dropout	1.73	0.65	71.761
dropout rate 0.5	1.53	0.67	70.329
dropout rate 0.8	1.48	0.67	70.305
dropout rate 0.85	1.36	0.65	70.228
dropout rate 0.9	4.53	0.05	67.541

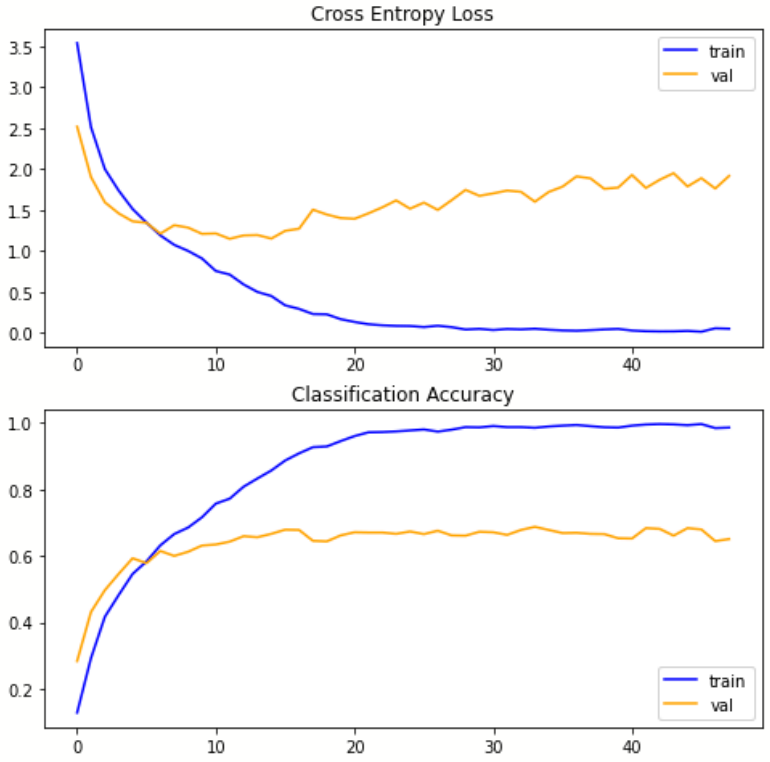
3. Χρήση Early Stopping

Παράμετροι: monitor='accuracy', mode='max', patience=30, min_delta=0.110 και δε χρησιμοποιείται το Dropout layer. Το πλήθος των εποχών τέθηκε σε 200, δεδομένου ότι με χρήση του Early Stopping, η εκπαίδευση θα διακοπεί όταν για περισσότερες από 30 εποχές, η τιμή του accuracy δεν αυξηθεί περισσότερο από 0.110. Η εκπαίδευση σταματάει στην εποχή 51, στην οποία όπως παρατηρείται από τις γραφικές το loss και το accuracy έχουν σταθεροποιηθεί. Για αυτό το πλήθος εποχών: loss 1.86 και accuracy 0.69.



Η τιμή του accuracy είναι βελτιστοποιημένη σε σχέση με το Dropout δεδομένου ότι έχουν αυξηθεί οι εποχές εκπαίδευσης, ενώ του loss χειρότερη. Σημειώνεται ότι η βελτίωση του accuracy στηρίζεται αποκλειστικά στον διπλασιασμό των εποχών εκπαίδευσης και όχι στην επεξεργασία του συνόλου των δεδομένων, καθώς για 25 εποχές χωρίς το Dropout layer το accuracy είχε τιμή 0.65. Συνεπώς, το σύνολο των δεδομένων οδηγεί σε υπερεκπαίδευση του μοντέλου.

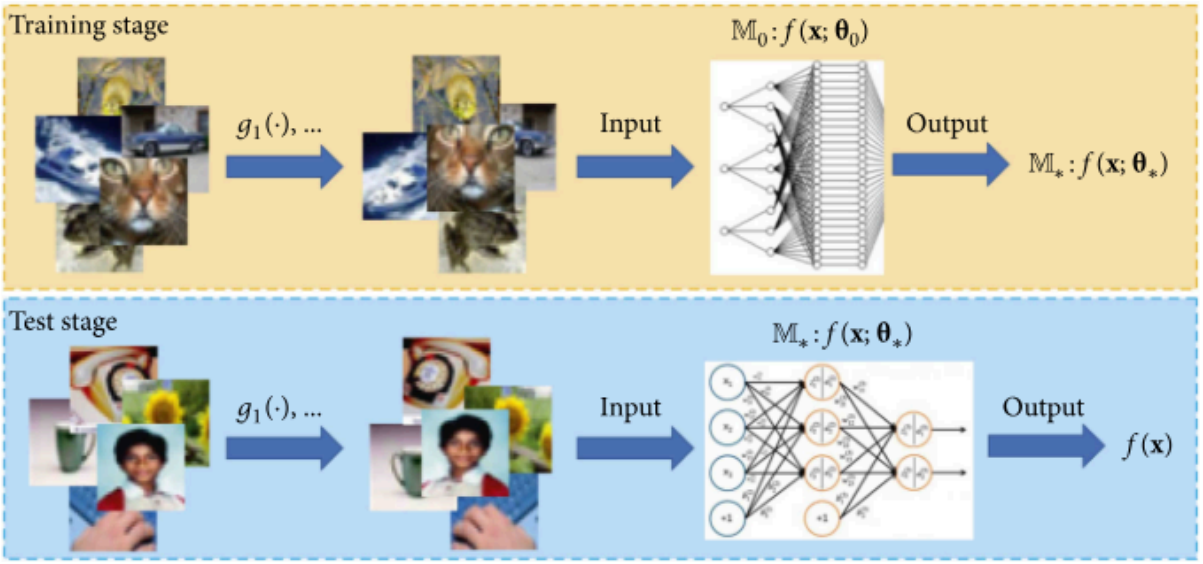
Χρησιμοποιώντας και το Dropout layer με rate=0.5, για τις ίδιες παραμέτρους του Early Stopping, η εκπαίδευση σταματάει στην εποχή 48 με τιμές loss 1.94 και accuracy 0.65, χειρότερα αποτελέσματα σε σχέση με πριν. Οι γραφικές που προκύπτουν είναι:



Παρατηρούμε ότι χωρίς το Dropout layer, το Early Stopping συγκλίνει στη μέγιστη τιμή του accuracy σε μικρότερο αριθμό εποχών.

4. Χρήση Data Augmentation

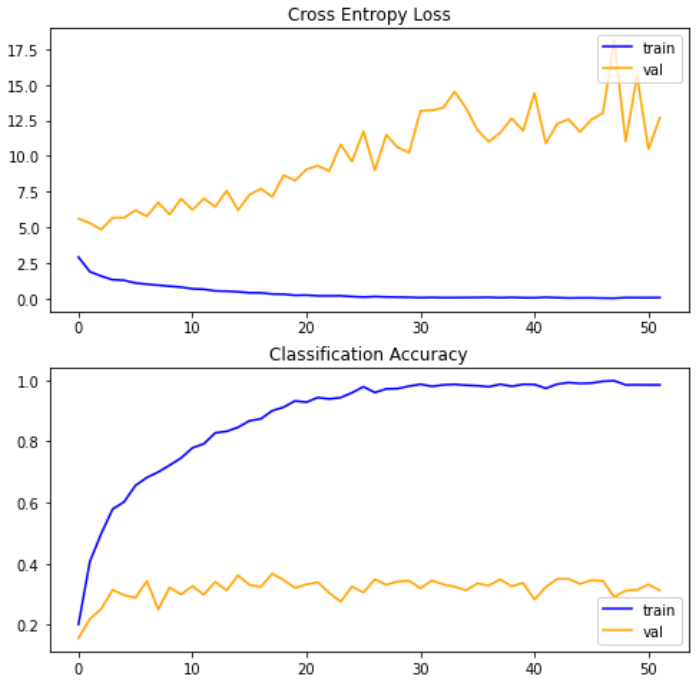
Ο κώδικας που χρησιμοποιήθηκε στηρίζεται στα άρθρα 1 και 2. Υλοποιήθηκε full stage data augmentation, όπως περιγράφεται στο άρθρο 2, δεδομένου ότι παρουσιάσει καλύτερα αποτελέσματα. Στόχους του data augmentation είναι να αυξηθεί το πλήθος των δεδομένων εκπαίδευσης του μοντέλου. Μετασχηματισμοί κάθε εικόνας προστίθενται στα δεδομένα εκπαίδευσης αντί για όμοια αντίγραφα της, ώστε το μοντέλο να εκπαιδεύεται σε περισσότερες διαφορετικές εισόδους και να αυξάνεται έτσι η ικανότητα γενίκευσής του.



Συνδυαστικά	**Παράμετροι (train + validation)**	**Epochs**	**Loss**	**Accuracy**
early stopping και dropout (rate 0.5)	horizontal flip	58	2.58	0.58
early stopping χωρίς dropout	horizontal flip	52	2.08	0.60
early stopping και dropout (rate 0.5)	horizontal flip και vertical flip	72	2.25	0.57
early stopping χωρίς dropout	horizontal flip και vertical flip	56	2.21	0.57

Από τα παραπάνω αποτελέσματα καταλήγουμε στο συμπέρασμα ότι μεγαλύτερο accuracy σε μικρότερο αριθμό εποχών εκπαίδευσης και με μικρότερο loss έχουμε όταν το υλοποιημένο data augmentation συνδυάζεται με Early Stopping, χωρίς Dropout και για τη δημιουργία εικόνων χρησιμοποιούνται οι μετασχηματισμοί samplewise_center=True, zca_whitening=True, horizontal_flip=True για τα train και validation datasets. Η χρήση του vertical flip συνδυαστικά με το horizontal flip δεν οδηγεί σε καλύτερα αποτελέσματα.

Γραφικές για Early Stopping, χωρίς Dropout:



5. Συμπεράσματα

Βάσει των παραπάνω, το μοντέλο συγκλίνει στη μέγιστη τιμή του accuracy σε μικρότερο αριθμό εποχών εκπαίδευσης όταν χρησιμοποιείται η τεχνική του Dropout, άρα με το σύνολο των δεδομένων το μοντέλο υπερ εκπαιδεύεται. Επίσης παρατηρήθηκε ότι συνδυασμός των τεχνικών Early Stopping με το Dropout και Data Augmentation με το Dropout οδηγεί σε χειρότερα αποτελέσματα.

Χρόνος εκπαίδευσης

Τα αποτελέσματα που προέκυψαν για τις διάφορες μεθόδους ελαχιστοποίησης του χρόνου εκπαίδευσης του μοντέλου συνοψίζονται στον παρακάτω πίνακα. Όλες οι δοκιμές πραγματοποιήθηκαν στο δοθέν μοντέλο (VGG16 με Dropout με rate 0.5, GPA και Dense layer) για 25 εποχές και batch size = 128.

Τεχνική	Χρόνος εκτέλεσης (seconds)
Naive (batch - repeat -> shuffle)	μη επαρκής μνήμη για την εκτέλεση
Prefetching	45.612
Data reading parallelization	N/A (1 dataset loaded από 1 source)
Map transformation parallelization	N/A (δε χρησιμοποιούνται map operations)
Caching (shuffle - repeat -> batch)	51.258
Reducing Memory Footprint (shuffle - repeat -> batch)	46.545
Reducing Memory Footprint (repeat - shuffle -> batch)	53.015

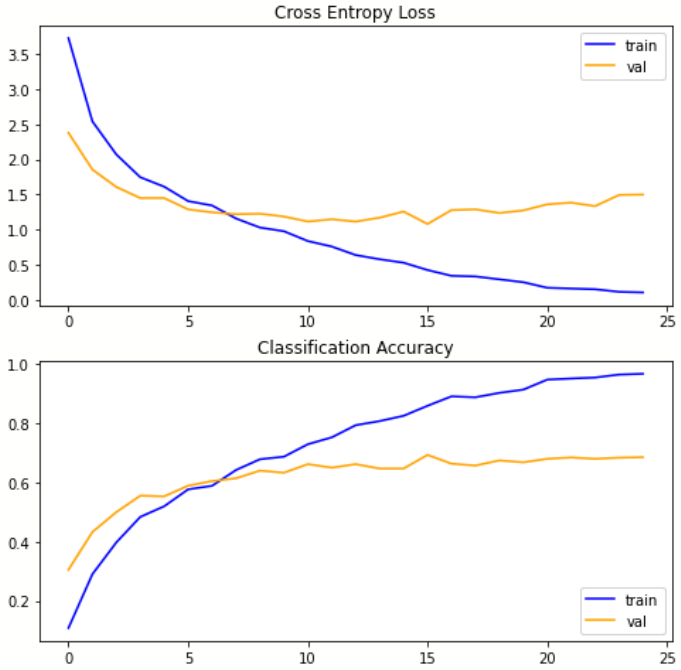
Παρατηρούμε ότι η καλύτερη τεχνική για την ελαχιστοποίηση του χρόνου εκτέλεσης είναι αυτή του prefetching. Επίσης, σημαντική διαφορά στον χρόνο έχει η σειρά με την οποία πραγματοποιούνται οι λειτουργίες shuffle, repeat και batch. Όπως εξηγείται σε αυτόν τον σχολιασμό.

Εκπαίδευση βαρών

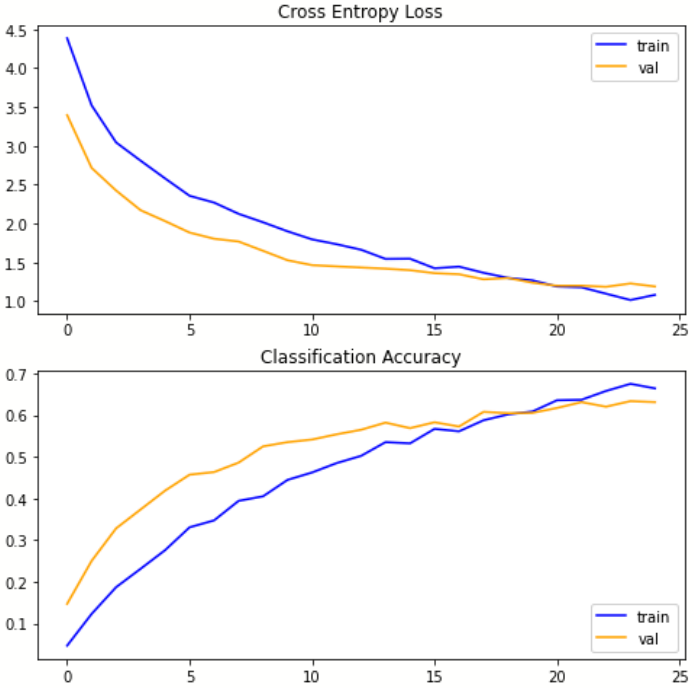
Τα πειράματα πραγματοποιήθηκαν στο δοθέν μοντέλο με Dropout rate=0.5 για 25 εποχές και με χρήση prefetching. Λαμβάνεται υπόψη ότι το μοντέλο VGG16 έχει 16 layers βαρών.

Trainable flag	Loss	Accuracy	Χρόνος εκτέλεσης (seconds)	Πλήθος trainable parameters	Γραφικές
true	1.51	0.67	90.695	14,765,988	1
false για τα top 5 output layers	1.23	0.62	81.408	7,686,564	2
false για τα top 10 output layers	2.01	0.41	70.052	1,196,708	3
false	2.86	0.20	36.164	51,300	4

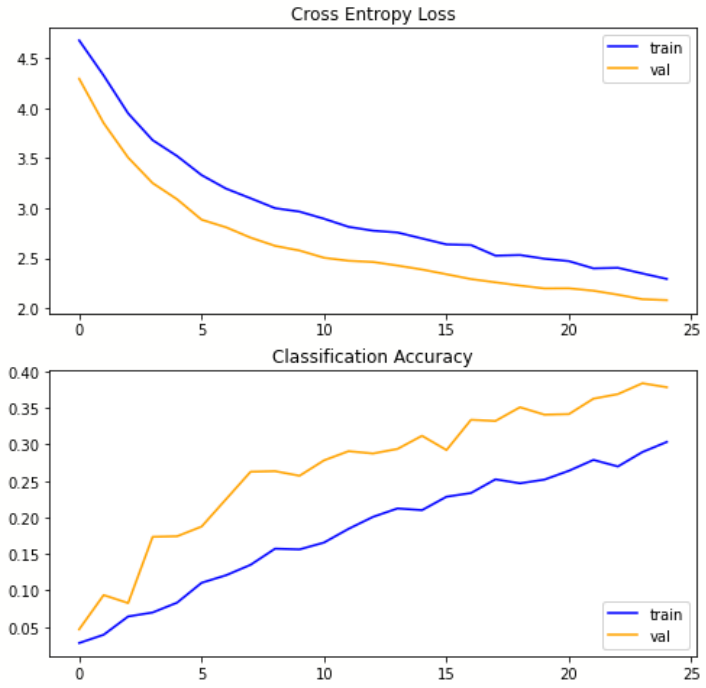
Γραφικές 1



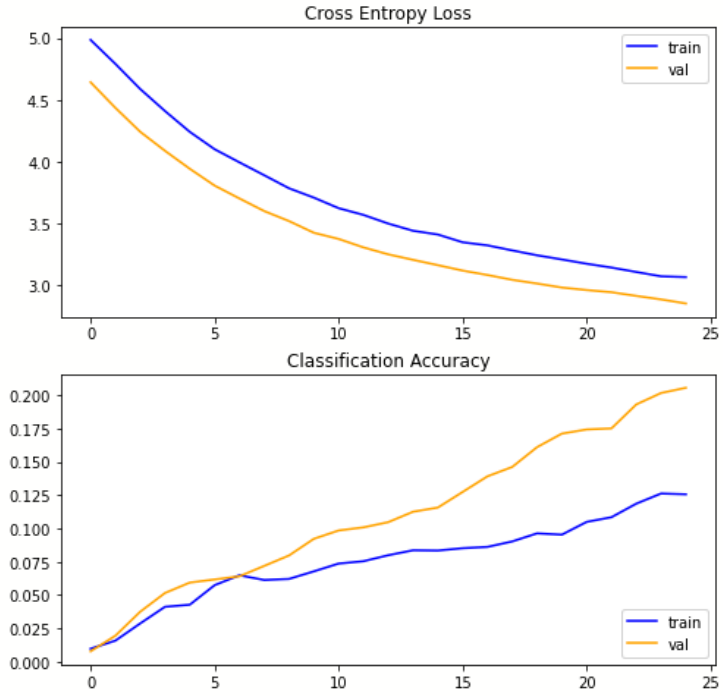
Γραφικές 2



Γραφικές 3



Γραφικές 4



Από τα παραπάνω αποτελέσματα παρατηρούμε ότι όσο αυξάνεται το πλήθος των layers του μοντέλου για τα οποία δεν ανανεώνονται οι τιμές των βαρών τους, δηλαδή η σημαία trainable γίνεται false, τόσο μειώνεται το πλήθος των παραμέτρων προς εκπαίδευση και άρα μειώνεται και ο χρόνος εκπαίδευσης, όπως είναι λογικό.

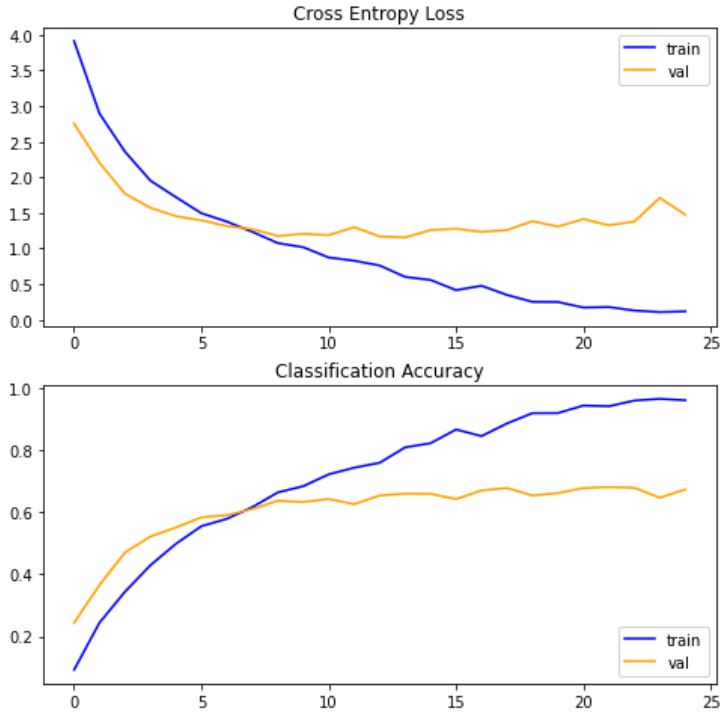
Επίσης, από τις γραφικές παραστάσεις είναι εμφανές ότι το accuracy συγκλίνει στη μέγιστη τιμή του και το loss στην ελάχιστη όσο το πλήθος των layer με σημαία trainable = true αυξάνεται, δεδομένου ότι το μοντέλο μαθαίνει και προσαρμόζεται κατάλληλα σε κάθε επόμενη εποχή.

Επίδραση αριθμού κατηγοριών

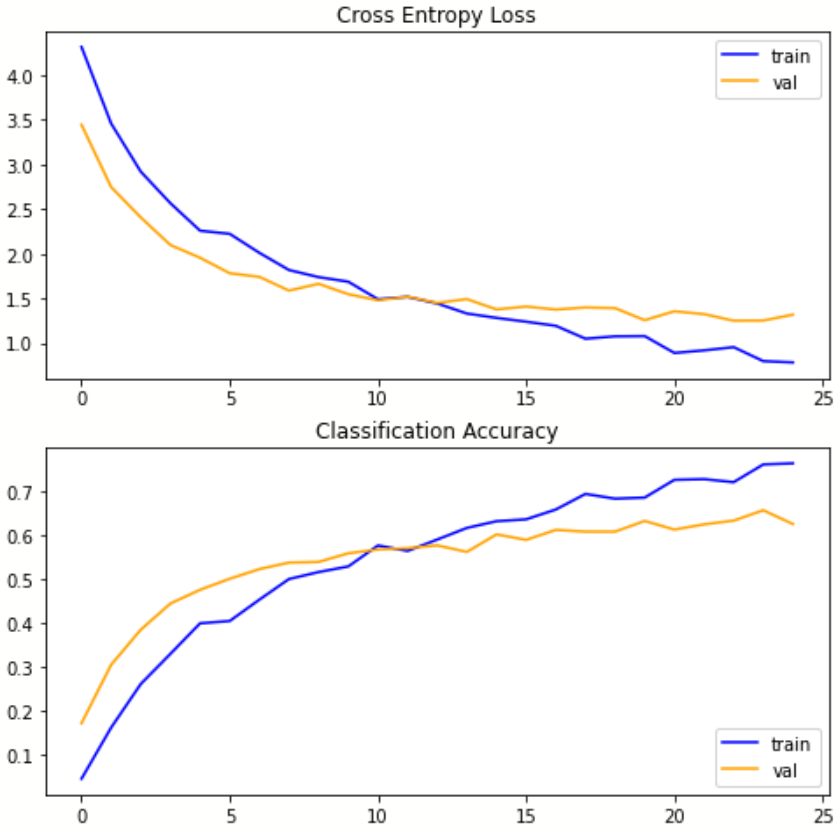
Τα πειράματα πραγματοποιήθηκαν στο δοθέν μοντέλο με Dropout rate=0.5 για 25 εποχές και με χρήση prefetching και flag trainable=true για όλα τα layers.

Αριθμός κατηγοριών	Loss	Accuracy	Χρόνος εκτέλεσης (seconds)	Γραφικές
20	1.50	0.67	90.272	1
40	1.28	0.64	95.410	2
60	1.39	0.61	99.928	3
80	1.60	0.56	104.648	4

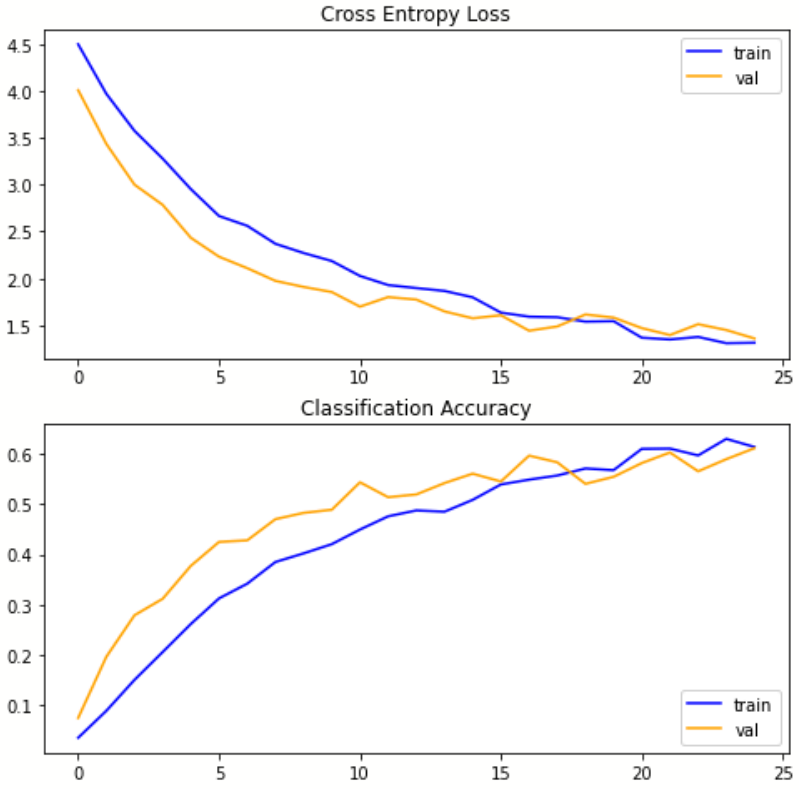
Γραφικές 1



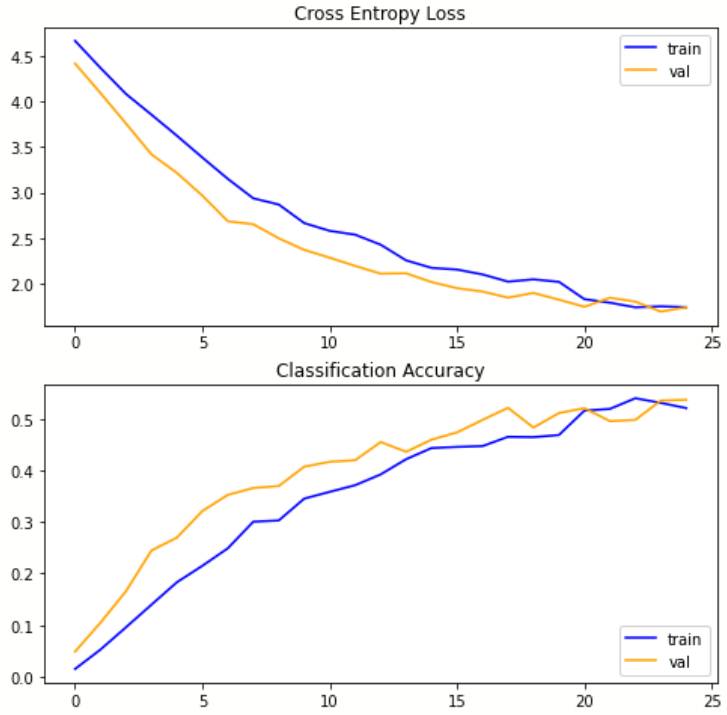
Γραφικές 2



Γραφικές 3



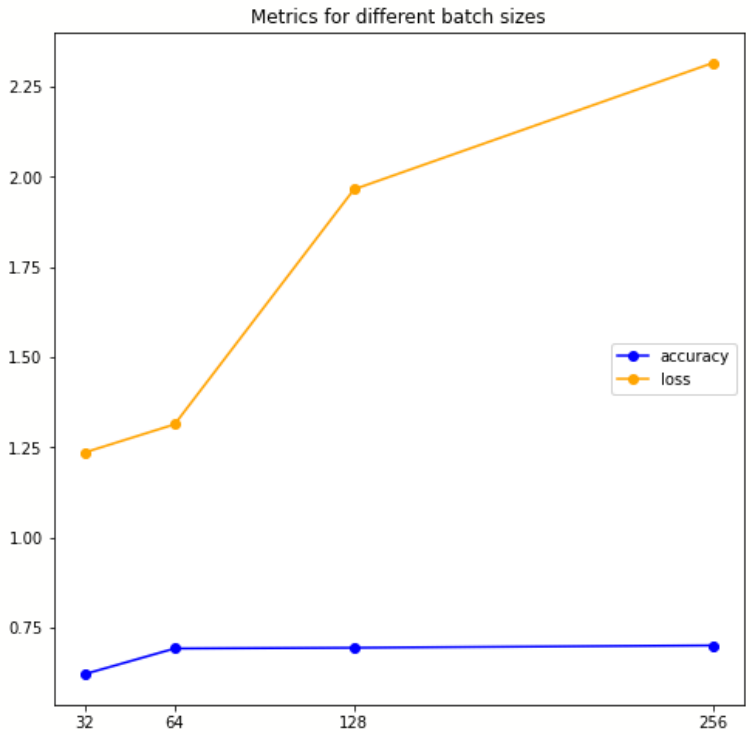
Γραφικές 4



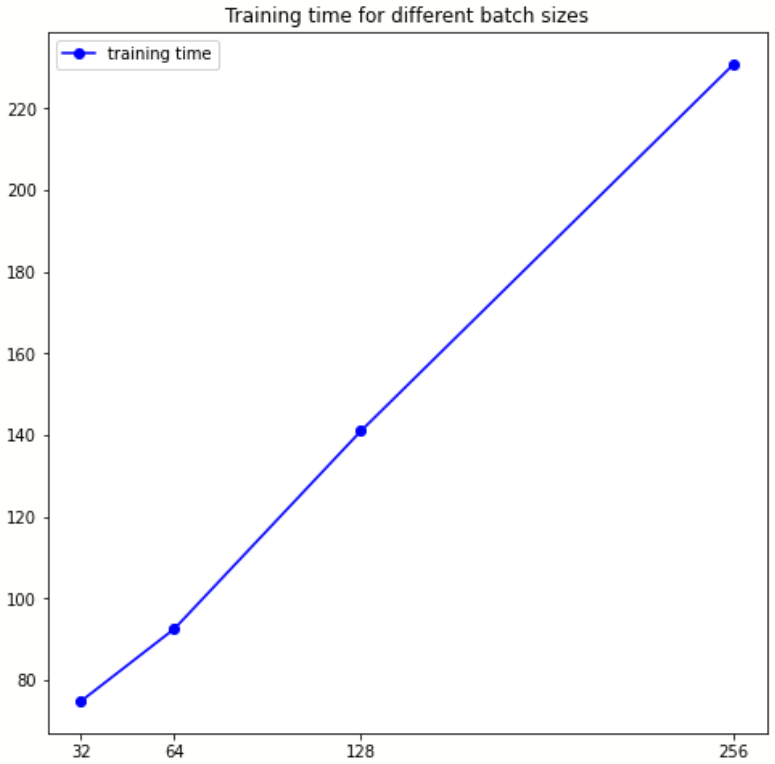
Από τα παραπάνω αποτελέσματα προκύπτει ότι η σύγκλιση στη μέγιστη τιμή της μετρικής του accuracy και στην ελάχιστη του loss είναι ταχύτερη όσο το πλήθος των κατηγοριών είναι μικρότερο. Αυτό εξηγείται από το γεγονός ότι ίδιο πλήθος δεδομένων κάθε φορά χρειάζεται να ταξινομηθούν σε διαφορετικό αριθμό κατηγοριών. Άρα, όσο περισσότερες είναι οι κατηγορίες, σε τόσο λιγότερα πρότυπα εισόδου εκπαιδεύεται το μοντέλο για τη διάκρισή τους. Χάνει δηλαδή τη διακριτική του ικανότητα αφού περιορίζονται τα πρότυπα που ανήκουν σε κάθε κατηγορία (θεωρώντας μία περίπου ομοιόμορφη συνολική κατανομή των δεδομένων του train set σε κατηγορίες). Επίσης, όσο αυξάνεται το πλήθος κατηγοριών αυξάνεται και ο χρόνος εκπαίδευσης του μοντέλου, αφού είναι δυσκολότερη η ταξινόμηση.

Επίδραση του μεγέθους δέσμης (batch size)

Η μελέτη της επίδρασης του batch size στην απόδοση, πραγματοποιήθηκε στο δοθέν μοντέλο με Dropout rate=0.5 για 25 εποχές και με χρήση prefetching και flag trainable=true για όλα τα layers. Για την επίδραση στο accuracy / loss:

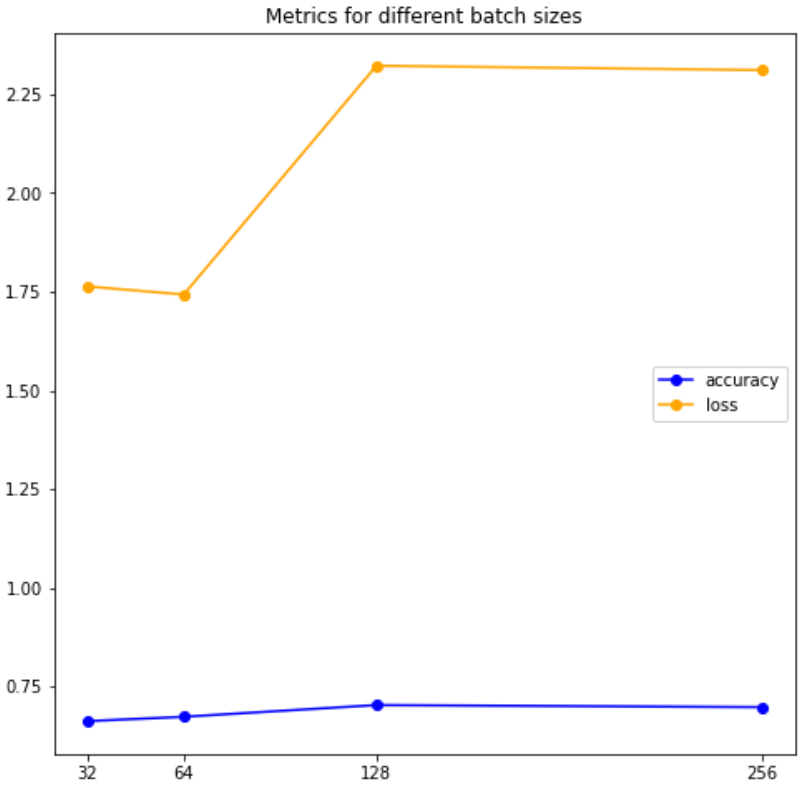


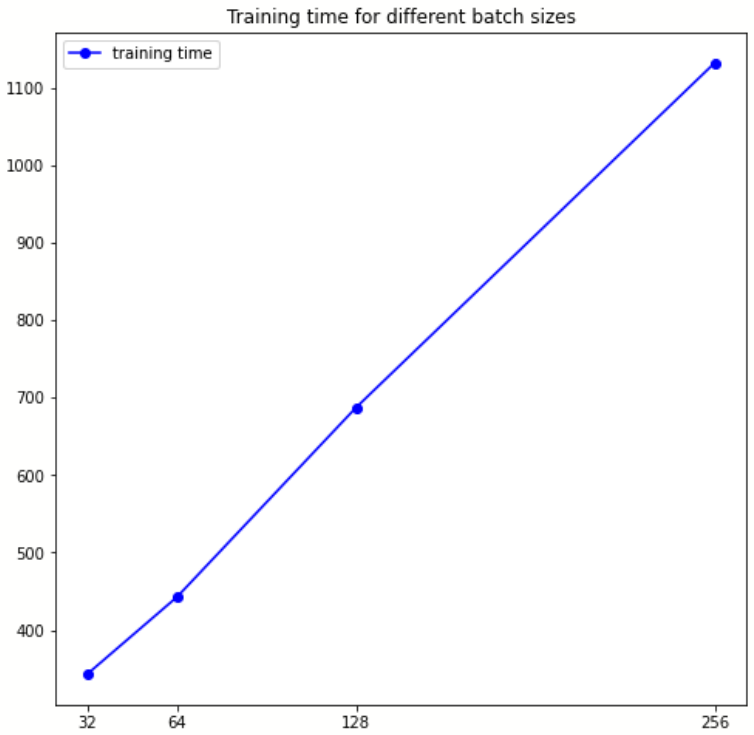
Και στον χρόνο εκτέλεσης:



Από τις παραπάνω γραφικές είναι εμφανές ότι για το συγκεκριμένο μοντέλο δεν παρατηρείται βελτίωση του accuracy για batch size μεγαλύτερο του 64. Ωστόσο, όσο το batch size αυξάνεται, αυξάνεται το loss και ο συνολικός χρόνος εκπαίδευσης για 25 εποχές, αφού μεγαλύτερο πλήθος δεδομένων επεξεργάζονται ανά εποχή.

Για 125 εποχές, οι γραφικές που προκύπτουν είναι οι παρακάτω, όπου παρατηρούμε σταθεροποίηση του accuracy και του loss για batch size μεγαλύτερο από 128. Ο χρόνος εκτέλεσης και πάλι αυξάνεται όσο το batch size αυξάνεται.



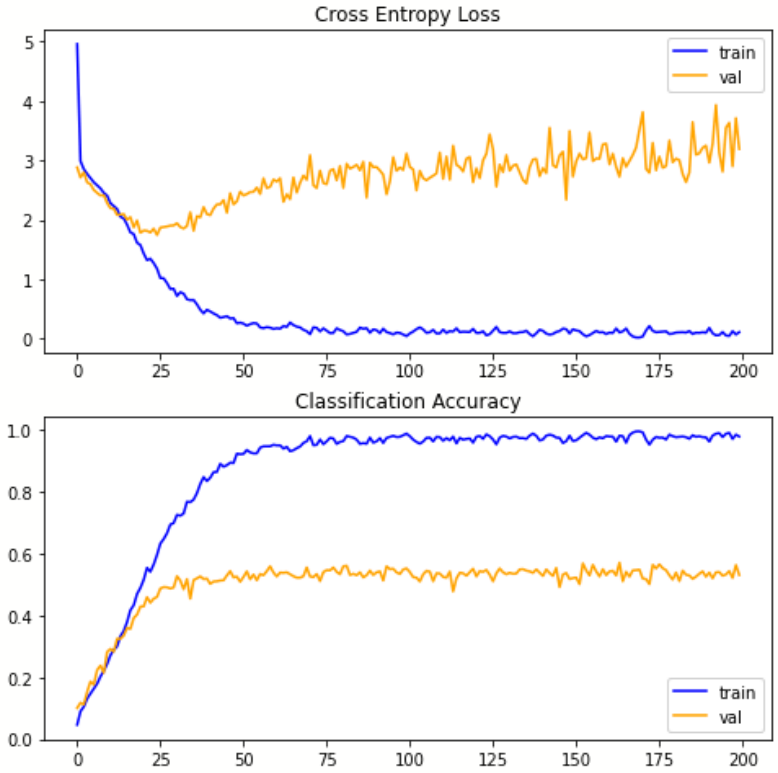


Επίδραση επιλογής optimizer

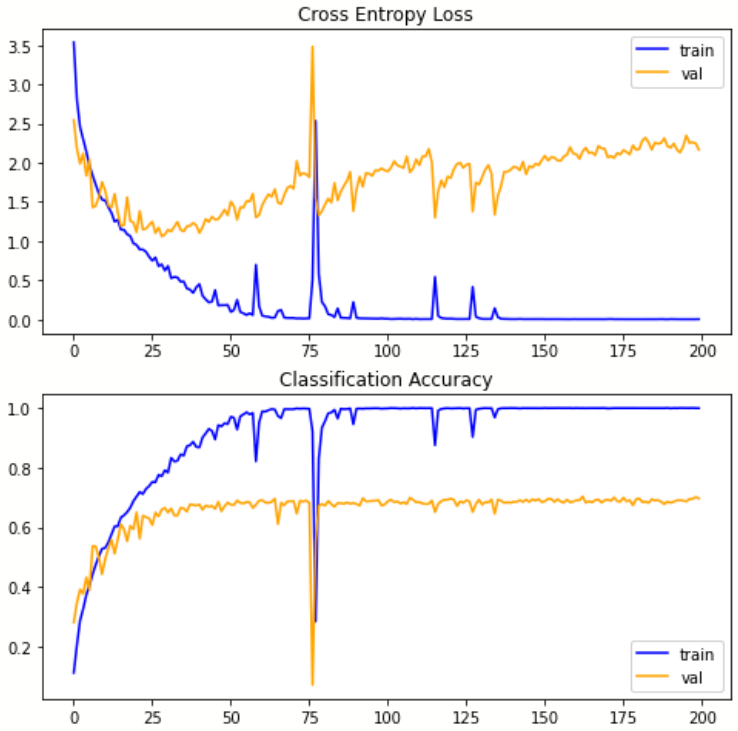
Τα πειράματα πραγματοποιήθηκαν στο δοθέν μοντέλο με Dropout rate=0.5 για 200 εποχές και με χρήση prefetching και flag trainable=true για όλα τα layers.

Optimizer	Learning Rate	Loss	Accuracy	Χρόνος εκτέλεσης (seconds)	Γραφικές
Adam	dflt	3.18	0.54	681.781	1
SGD	dflt	2.26	0.70	655.753	2
RMSprop	adaptive	3.02	0.05	715.164	3
Adagrad	adaptive	1.95	0.67	671.676	4

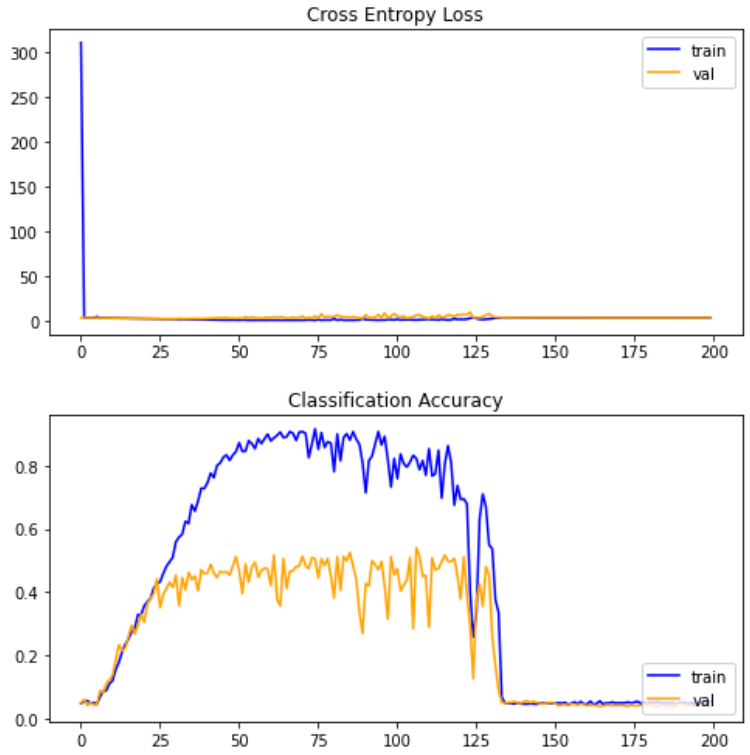
Γραφικές 1



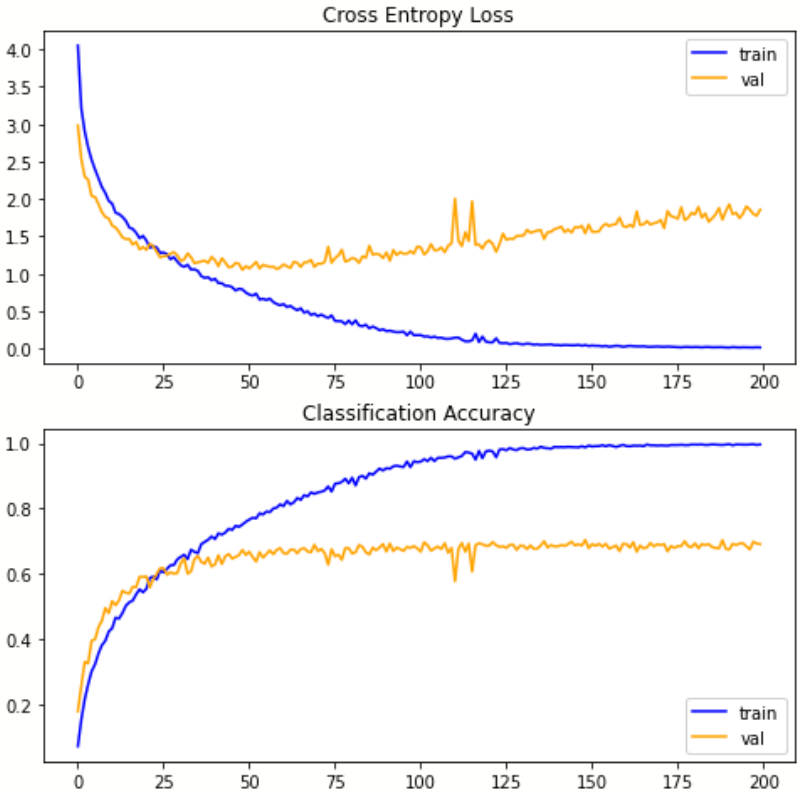
Γραφικές 2



Γραφικές 3



Γραφικές 4

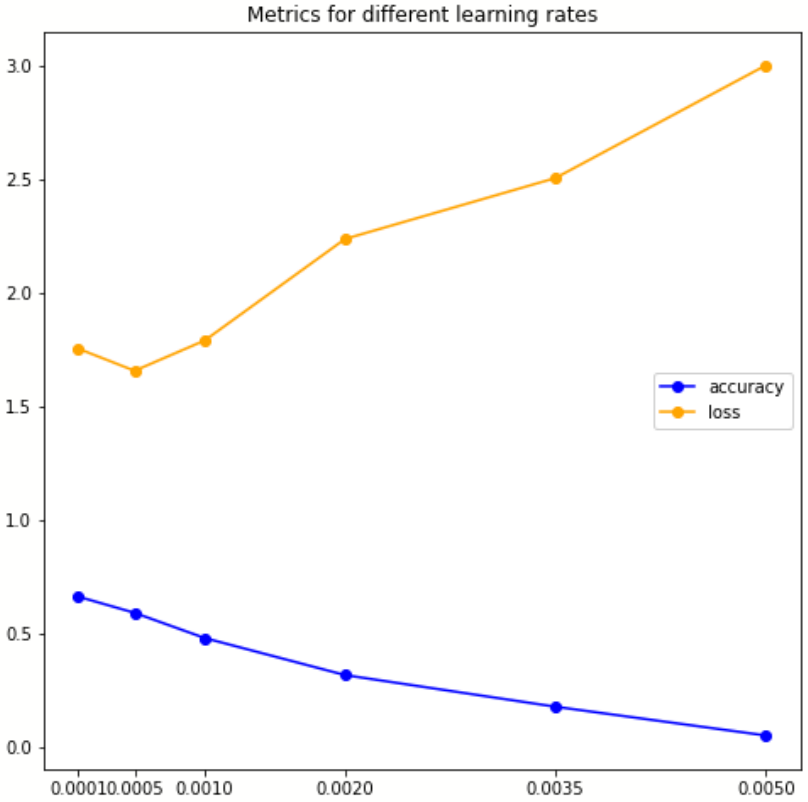


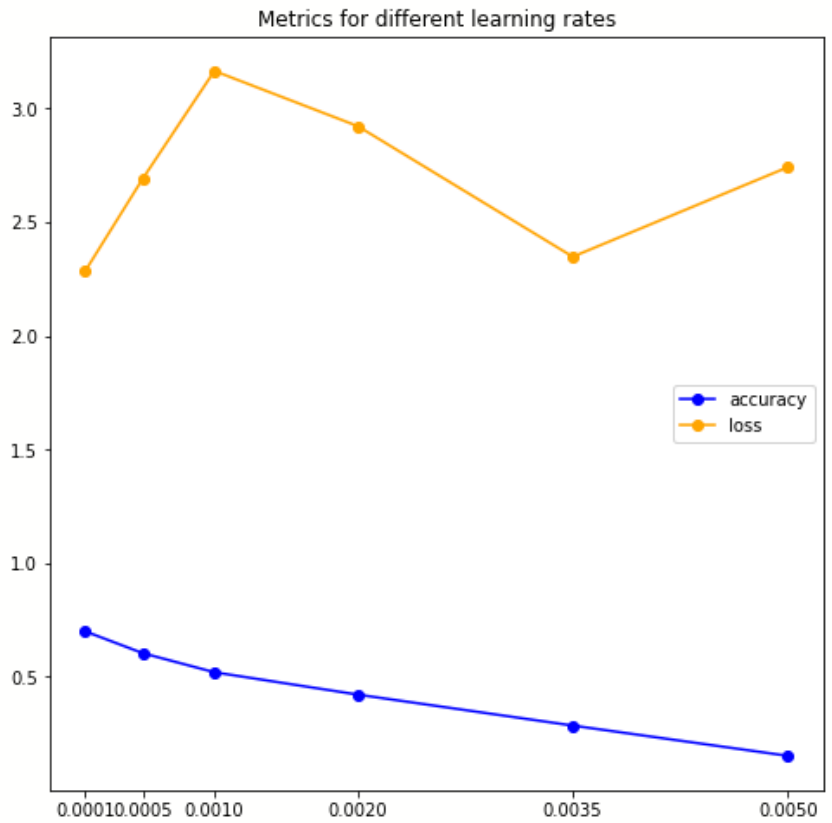
Παρατηρούμε ότι συνολικά τη χειρότερη απόδοση έχει για το συγκεκριμένο μοντέλο ο RMSprop optimizer. Σε βάθος 200 εποχών, καλύτερα αποτελέσματα έχει ο SGD optimizer, αφού συγκλίνει σε μεγαλύτερο accuracy και μικρότερο loss σε μικρότερο χρόνο εκτέλεσης. Ωστόσο, καλύτερα αποτελέσματα με λιγότερες αυξομειώσεις για μικρότερο αριθμό εποχών παρουσιάζει ο Adagrad, όπως φαίνεται από τις γραφικές παραστάσεις.

Επίδραση μεταβολής του learning rate

Ταχύτερη σύγκλιση σε βέλτιστες τιμές σε λιγότερες από 50 εποχές έχει ο Adam, στον οποίο μελετάται η επίδραση της μεταβολής του learning rate για 25 εποχές. Παρατηρούμε ότι όσο το learning rate αυξάνεται, τόσο το accuracy παραδόξως μειώνεται.

Για 125 εποχές ώστε με κάποιες μεταβολές να έχει επιτευχθεί σύγκλιση στο μέγιστο accuracy. Ωστόσο παρατηρείται το ίδιο αποτέλεσμα.





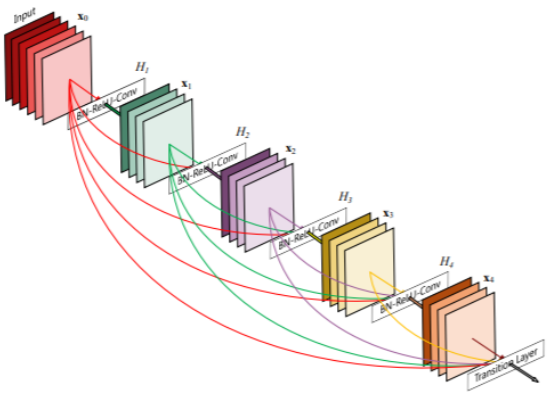
Επίδοση ανά κλάση

Στο δοθέν μοντέλο με Dropout rate=0.5 για 125 εποχές και με χρήση prefetching και flag trainable=true για όλα τα layers, η απόδοση ανά κλάση είναι:

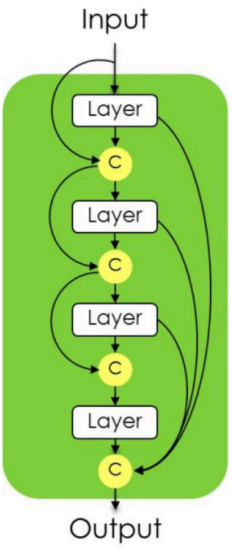
	precision	recall	f1-score	support
2	0.67	0.56	0.61	100
6	0.72	0.78	0.75	100
18	0.76	0.76	0.76	100
24	0.93	0.89	0.91	100
26	0.71	0.67	0.69	100
29	0.75	0.71	0.73	100
35	0.67	0.58	0.62	100
42	0.63	0.64	0.64	100
45	0.59	0.63	0.61	100
51	0.77	0.64	0.70	100
55	0.43	0.53	0.47	100
58	0.95	0.92	0.93	100
62	0.86	0.90	0.88	100
65	0.61	0.64	0.62	100
72	0.55	0.59	0.57	100
80	0.51	0.53	0.52	100
82	0.91	0.91	0.91	100
84	0.79	0.76	0.78	100
88	0.68	0.65	0.66	100
97	0.66	0.73	0.70	100
accuracy			0.70	2000
macro avg	0.71	0.70	0.70	2000
weighted avg	0.71	0.70	0.70	2000

DenseNet:

Στην αρχιτεκτονική αυτή το κάθε layer συνδέεται με τα υπόλοιπα με έναν feed-forward τρόπο. Ενώ τα παραδοσιακά convolutional networks με L layers έχουν L συνδέσεις—ένα μεταξύ του κάθε layer και του ακόλουθού του—το δικό μας δίκτυο έχει L(L+1)/2 απευθείας συνδέσεις. Για κάθε layer, τα feature-maps όλων των προηγούμενων layers χρησιμοποιούνται ως είσοδοι, και τα δικά του feature-maps χρησιμοποιούνται ως είσοδοι σε αυτά που ακολουθούν.

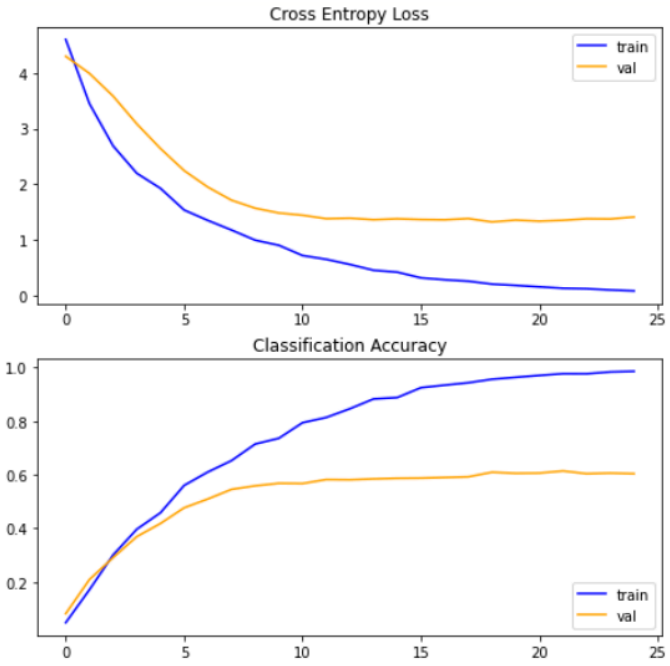


Παράδειγμα για 4 conv layers φαίνεται στην παρακάτω εικόνα. Το δικό μας μοντέλο έχει 169.



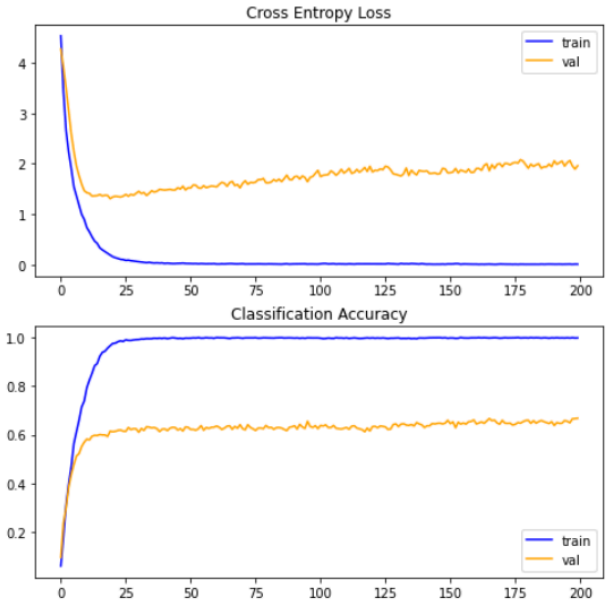
Το compilation χρησιμοποιεί τον Adam optimizer με learning rate 0.00005 ενώ έχει πραγματοποιηθεί prefetching των δεδομένων με batch size 128 και autotune. Με εκπαίδευση του μοντέλου σε 25 εποχές με 40 βήματα εκπαίδευσης και 10 βήματα validation ανά εποχή:

Παράμετρος βελτιστοποίησης (25 epochs)	**Τιμή**
Total parameters	12,809,380
Trainable parameters	12,650,980
Non-trainable parameters	158,400
Loss	1.36
Accuracy	0.62



Για πιο εποπτικό σχολιασμό των αποτελεσμάτων, αυξάνονται οι εποχές εκπαίδευσης σε 200 στο υπάρχον μοντέλο.

Παράμετρος βελτιστοποίησης (200 epochs)	**Τιμή**
Loss	2.12
Accuracy	0.64



Παρατηρούμε ότι πέρα από τις 25 περίπου εποχές, η μετρική του accuracy δε βελτιώνεται περαιτέρω. Συνεπώς δε χρειάζεται να αυξήσουμε περαιτέρω το πλήθος των εποχών.

Επίδραση υπερεκπαίδευσης

1. Μελέτη προστιθέμενων layers

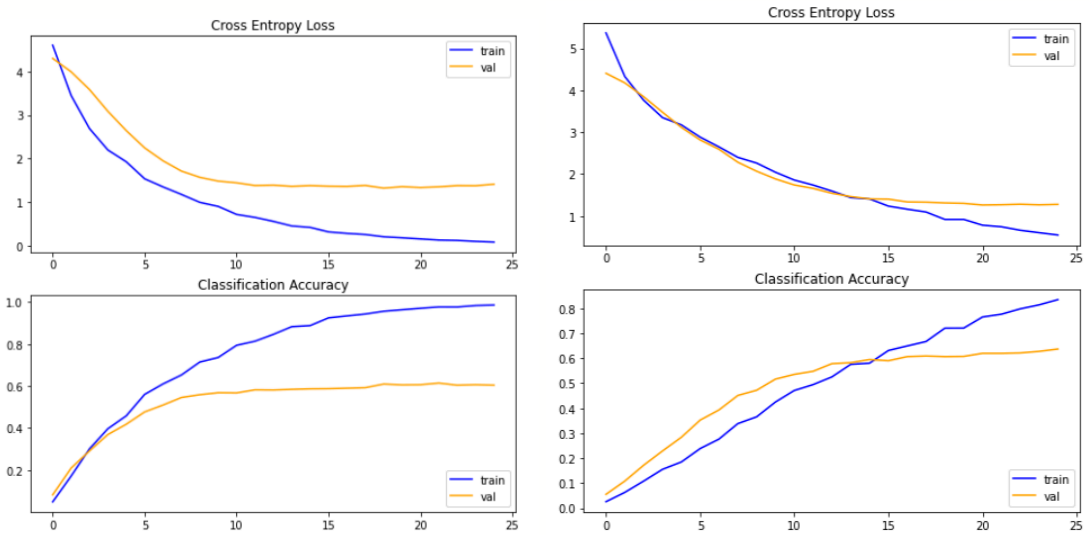
Αρχικά, ελέγχθηκαν τα επιπλέον layers που έχουν προστεθεί στο μοντέλο για την αποφυγή της υπερκπαίδευσης. Παρατηρείται πως και τα δύο προστιθέμενα layers συμβάλλουν στην αποφυγή της υπερεκπαίδευσης, με καθοριστικότερη τη συμβολή του GPA. Τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα.

Αφαίρεση layer	**Loss**	**Accuracy**
Dropout	1.41	0.61
GPA	1.32	0.05
Dropout και GPA	1.41	0.05

2. Χρήση Dropout

Η τεχνική της προσθήκης του dropout layer εξαρτάται από τις διάφορες τιμές του rate του. Στον παρακάτω πίνακα παρουσιάζεται το πώς οι τιμές αυτές επιδρούν στις μετρικές.

	Loss	**Accuracy**
Χωρίς Dropout layer	1.41	0.61
Dropout layer με rate 0.2	1.43	0.6
Dropout layer με rate 0.5	1.36	0.62
Dropout layer με rate 0.8	1.31	0.62



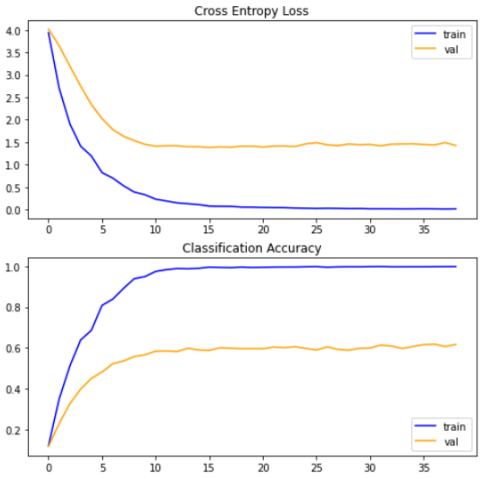
Παρατηρείται μια ενδιαφέρουσα τροποποίηση των γραφικών παραστάσεων. Ενώ οι τελικές τιμές των μετρικών μας δεν υφίστανται μεγάλες αλλαγές στις δύο περιπτώσεις, σημειώνεται μια μεγαλύτερη σύγκλιση μεταξύ accuracy και val_accuracy, όπως και μεταξύ loss και val_loss, στις δεξιά γραφικές παραστάσεις. Αυτό εξηγείται εύκολα μέσα από την γενικότερη λογική της εξάλειψης της υπερεκπαίδευσης, μιας που στις αριστερά γραφικές (rate 0.5), οι τιμές των μετρικών αυξάνονται πολύ για το train set, ενώ το test set δεν ακολουθεί στον ίδιο βαθμό, με ασφαλές συμπέρασμα την υπερεκπαίδευση του μοντέλου στο train set. Στην δεξιά περίπτωση (rate 0.8), παρατηρείται μια πιο συγκρατημένη συμπεριφορά απέναντι στο train set, την οποία ακολουθεί με μεγαλύτερη συνέπεια και το test set.

3. Χρήση Early Stopping

Ομοίως με τα προηγούμενα μοντέλα μεταφοράς μάθησης, θα χρησιμοποιηθεί η τεχνική του Early Stopping ως μια επιπλέον λύση του προβλήματος της υπερεκπαίδευσης. Οι παράμετροι είναι οι εξής: monitor='accuracy', mode='max', patience=30, min_delta=0.110. Το πλήθος των εποχών τέθηκε σε 200, δεδομένου ότι με χρήση του Early Stopping, η εκπαίδευση θα διακοπεί όταν για περισσότερες από 30 εποχές, η τιμή του accuracy δεν αυξηθεί περισσότερο από 0.110. Σημειώνεται, επίσης, ότι έχει αφαιρεθεί το dropout layer, ώστε να γίνει η σύγκριση του ποια από τις δύο μεθόδους είναι πιο αποτελεσματική στο δεδομένο πρόβλημα.

```
Epoch 38/200
40/40 [==============================] - 4s 110ms/step - loss: 0.0116 - acc: 0.9986 - val_loss: 1.4897 - val_acc: 0.6062
Epoch 39/200
40/40 [==============================] - 5s 114ms/step - loss: 0.0146 - acc: 0.9988 - val_loss: 1.4261 - val_acc: 0.6164
Epoch 00039: early stopping
```

Η εκπαίδευση σταματάει στην εποχή 39, στην οποία όπως παρατηρείται το accuracy έχει σταθεροποιηθεί στην τιμή 0.61. Η τιμή του loss είναι ίση με 1.44. Η τιμή του accuracy είναι βελτιστοποιημένη σε σχέση με το Dropout, ενώ του loss χειρότερη.



4. Χρήση Data Augmentation

Ή αλλιώς επαύξηση δεδομένων. Η υπερεκπαίδευση συνήθως συμβαίνει όταν έχουμε λίγα ή/και πολύ όμοια δεδομένα εκπαίδευσης. Ένας τρόπος να διορθωθεί αυτό το πρόβλημα είναι να αυξήσουμε τα δεδομένα (data augmentation). Το data augmentation δημιουργεί νέα δεδομένα εκπαίδευσης με βάση τα υπάρχοντα εφαρμόζοντας τυχαίους μετασχηματισμούς ώστε να προκύπτουν αληθοφανείς εικόνες. Δοκιμάζοντας την τεχνική αυτή, τα αποτελέσματα ήταν:

Συνδυαστικά	Παράμετροι (train + validation)	Epochs	Loss	Accuracy
early stopping και dropout (rate 0.5)	horizontal flip	53	1.99	0.54
early stopping χωρίς dropout	horizontal flip	50	1.94	0.53
early stopping και dropout (rate 0.5)	horizontal flip και vertical flip	78	2.12	0.55
early stopping χωρίς dropout	horizontal flip και vertical flip	60	2.03	0.53

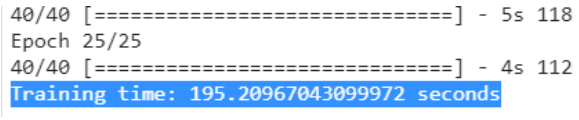
5. Συμπεράσματα

Συγκεντρωτικά τα βέλτιστα αποτελέσματα για τις μετρικές του accuracy και loss για κάθε μέθοδο περιορισμού της υπερεκπαίδευσης παρατίθενται στον παρακάτω πίνακα. Στην τρίτη σειρά έχουν εφαρμοστεί ταυτόχρονα και οι δύο τεχνικές για την εξάλειψη της υπερεκπαίδευσης, δηλαδή η τεχνική Dropout (rate = 0.8, 25 epochs) και η τεχνική Early Stopping (51 epochs όταν εφαρμόζεται μαζί με το dropout).

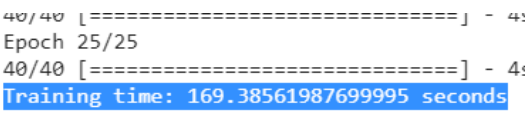
Τεχνική	Loss	Max Accuracy
Dropout (rate = 0.8, 25 epochs)	1.31	0.62
Early Stopping (51 epochs)	1.44	0.61
Both techniques	1.51	0.64
Data Augmentation (best case for accuracy)	2.12	0.55

Χρόνος εκπαίδευσης

Οι αρχικές μετρικές που έχουν παρουσιαστεί και παραπάνω προκύπτουν όταν για την εκπαίδευση του μοντέλου γίνεται data prefetching. Αυτή η τεχνική επιτρέπει όσο η εκπαίδευση του μοντέλου βρίσκεται στο βήμα κ, το input pipeline διαβάζει τα δεδομένα για το επόμενο βήμα εκπαίδευσης, κ+1. Για train για 40 steps, 25 εποχές, με adam optimizer (learning rate 0.00005) χρειάστηκαν:



Στη συνέχεια, αφαιρούμε την τεχνική του data prefetching, ώστε να δούμε πώς αυτό επιδρά στο χρόνο εκπαίδευσης. Ο χρόνος που χρειάστηκε τελικά ήταν ίσος με:



Παρακάτω εμφανίζονται και οι άλλες επιλογές ελαχιστοποίησης του χρόνου εκπαίδευσης και τι αποτελέσματα αυτές επέφεραν.

Τεχνική	**Χρόνος εκτέλεσης (seconds)**
Naive (batch - repeat -> shuffle)	μη επαρκής μνήμη (session crashed)
Prefetching	169.15
Data reading parallelization	N/A (1 dataset loaded από 1 source)
Map transformation parallelization	N/A (δεν χρησιμοποιούνται map operations)
Caching (shuffle - repeat -> batch)	165.53
Reducing Memory Footprint (shuffle - repeat -> batch)	175.5
Reducing Memory Footprint (repeat - shuffle -> batch)	165.95

Εκπαίδευση βαρών (σημαία trainable)

1. Σημαία trainable ίση με false

Επιλέγουμε να παγώσουμε τη συνελικτική βάση και να εκπαιδεύσουμε την κεφαλή ταξινόμησης (classification head). Αυτό αντιστοιχεί στο να χρησιμοποιήσουμε τη συνελικτική βάση για εξαγωγή χαρακτηριστικών (feature

extraction). Θέτουμε την σημαία trainable ίση με False. Στις default συνθήκες ήταν ίση με True. Οι μετρικές προκύπτουν: acc = 0.15, loss = 2.95.

2. Σημαία ανά επίπεδο

Εδώ επιλέγουμε να εκπαιδευτεί μόνο ένα ποσοστό των επιπέδων ευρισκόμενο προς την έξοδο του δικτύου. Οι σημαίες trainable εδώ ορίζονται ανά επίπεδο. Οι μετρικές προκύπτουν: acc = 0.41, loss = 2.19 για 50 layers με σημαία false και acc = 0.41, loss = 2.2 για 100 layers με σημαία false.

3. Συνοπτικά συμπεράσματα

ΣΗΜΑΙΑ	**Loss**	**Max Accuracy**
True	1.36	0.62
False	2.95	0.15
false για τα 50 layers	2.19	0.41
false για τα 100 layers	2.2	0.41

Επίδραση επιλογής optimizer

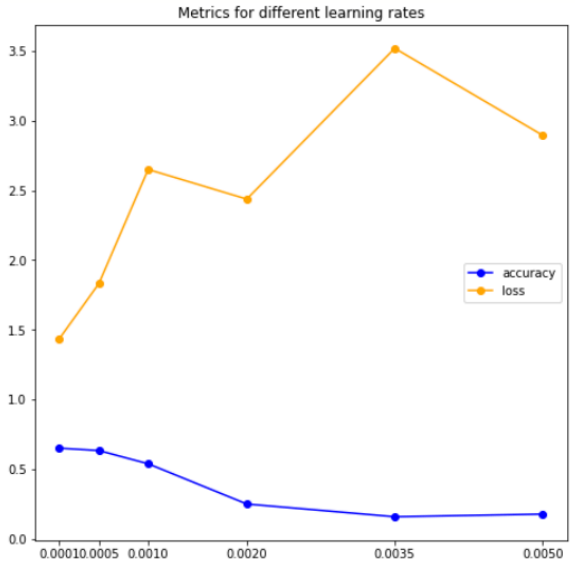
1. Μελέτη διαφόρων optimizers

Ο default optimizer που χρησιμοποιήθηκε ήταν ο Adam, με learning rate 0.00005. Σύμφωνα με τις κατάλληλες βιβλιοθήκες, το default learning rate του Adam είναι 0.001. Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα για τέσσερις διαφορετικούς optimizers, τον Adam (με 0.001 αντί 0.00005), τον SGD (0.01), τον RMSprop (0.001) και τέλος τον Adagrad (0.001).

Metrics	**Adam (0.00005)**	**Adam (0.001)**	**SGD (0.01)**	**RMSprop (0.001)**	**Adagrad (0.001)**
Accuracy	0.62	0.5	0.63	0.49	0.59
Loss	1.36	1.93	1.51	9.83	1.35

2. Βέλτιστο learning rate για δεδομένο optimizer

Στο σημείο αυτό έγινε προσομοίωση για την εύρεση του βέλτιστου learning rate για τον δεδομένο optimizer Adam. Τα αποτελέσματα που προέκυψαν, τα οποία φαίνονται παρακάτω, έδειξαν ότι η βέλτιστη τιμή ήταν ίση με 0.0001.



Επίδραση πλήθους των κατηγοριών

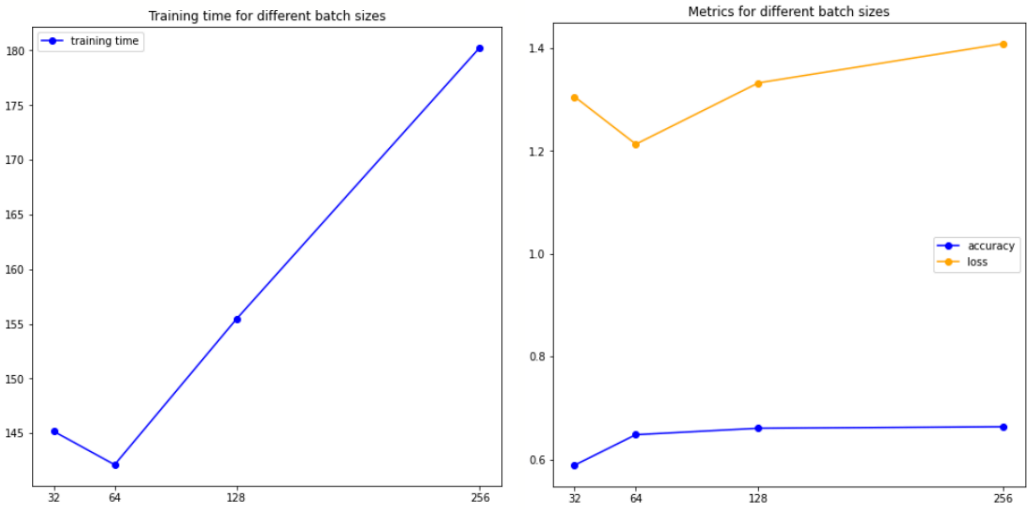
Ο default αριθμός κατηγοριών που εισάγεται είναι 20. Παρακάτω δοκιμάζονται και τέσσερα διαφορετικά νούμερα για τις εισαγόμενες κατηγορίες, καθώς και το πώς αυτά επιδρούν οι μετρικές (προστίθεται και η μετρική του χρόνου εκπαίδευσης).

Metrics	**20**	**40**	**60**	**80**
Accuracy	0.62	0.61	0.57	1.54
Loss	1.36	1.40	1.49	1.73
Time	195.2	220.7	202.6	203.4

Παρατηρούμε χειροτέρευση των μετρικών όσο αυξάνονται οι κατηγορίες, με εξαίρεση την μετρική του χρόνου που φαίνεται να είναι μεγαλύτερος στην τιμή των 60 κατηγοριών και στην συνέχεια να σταθεροποιείται λίγο πάνω από την αρχική του τιμή.

Επίδραση του μεγέθους δέσμης (batch size)

Στη συνέχεια, εξετάστηκαν οι διάφορες τιμές του μεγέθους δέσμης, με στόχο την ελαχιστοποίηση του χρόνου εκπαίδευσης. Χρησιμοποιήθηκαν οι τιμές [32,64,128,256], οι οποίες είχαν σαν αποτέλεσμα την γραφική που ακολουθεί. Σε αυτήν φαίνεται ότι μικρότερο χρόνο εκπαίδευσης έχουμε για μέγεθος δέσμης ίσο με 64.



Συμπεριφορά ανά κλάση

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	0
2	0.17	0.01	0.02	100
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
5	0.00	0.00	0.00	0
6	0.25	0.01	0.02	100
7	0.00	0.00	0.00	0
8	0.00	0.00	0.00	0
10	0.00	0.00	0.00	0
11	0.00	0.00	0.00	0
12	0.00	0.00	0.00	0
13	0.00	0.00	0.00	0
14	0.00	0.00	0.00	0
15	0.00	0.00	0.00	0
17	0.00	0.00	0.00	0
18	0.00	0.00	0.00	100
19	0.00	0.00	0.00	0
21	0.00	0.00	0.00	0
22	0.00	0.00	0.00	0
23	0.00	0.00	0.00	0
24	0.00	0.00	0.00	100
25	0.00	0.00	0.00	0
26	0.00	0.00	0.00	100
27	0.00	0.00	0.00	0
28	0.00	0.00	0.00	0
29	0.00	0.00	0.00	100
30	0.00	0.00	0.00	0
31	0.00	0.00	0.00	0
32	0.00	0.00	0.00	0
33	0.00	0.00	0.00	0
34	0.00	0.00	0.00	0
35	0.25	0.01	0.02	100
37	0.00	0.00	0.00	0
38	0.00	0.00	0.00	0
39	0.00	0.00	0.00	0
40	0.00	0.00	0.00	0
41	0.00	0.00	0.00	0
42	0.10	0.02	0.03	100
43	0.00	0.00	0.00	0
44	0.00	0.00	0.00	0
45	0.00	0.00	0.00	100
47	0.00	0.00	0.00	0
49	0.00	0.00	0.00	0
51	0.00	0.00	0.00	100
53	0.00	0.00	0.00	0
55	0.12	0.11	0.12	100
56	0.00	0.00	0.00	0
57	0.00	0.00	0.00	0
58	0.00	0.00	0.00	100
59	0.00	0.00	0.00	0
60	0.00	0.00	0.00	0
61	0.00	0.00	0.00	0
62	0.00	0.00	0.00	100
63	0.00	0.00	0.00	0
64	0.00	0.00	0.00	0
65	0.00	0.00	0.00	100
66	0.00	0.00	0.00	0
67	0.00	0.00	0.00	0
68	0.00	0.00	0.00	0
69	0.00	0.00	0.00	0
71	0.00	0.00	0.00	0
72	0.07	0.07	0.07	100
73	0.00	0.00	0.00	0
74	0.00	0.00	0.00	0
76	0.00	0.00	0.00	0
77	0.00	0.00	0.00	0
79	0.00	0.00	0.00	0
80	0.09	0.04	0.05	100
81	0.00	0.00	0.00	0
82	0.25	0.02	0.04	100
83	0.00	0.00	0.00	0
84	0.00	0.00	0.00	100
85	0.00	0.00	0.00	0
86	0.00	0.00	0.00	0
88	0.00	0.00	0.00	100
89	0.00	0.00	0.00	0
90	0.00	0.00	0.00	0
91	0.00	0.00	0.00	0
92	0.00	0.00	0.00	0
93	0.00	0.00	0.00	0
94	0.00	0.00	0.00	0
95	0.00	0.00	0.00	0
96	0.00	0.00	0.00	0
97	0.00	0.00	0.00	100
98	0.00	0.00	0.00	0
99	0.00	0.00	0.00	0
accuracy			0.01	2000
macro avg	0.01	0.00	0.00	2000
weighted avg	0.06	0.01	0.02	2000

[' baby', ' bee', ' caterpillar', ' cockroach', ' crab', ' dinosaur', ' girl', ' leopard', ' lobster', ' mushroom', ' otter', ' pickup_truck', ' poppy', ' rabbit', ' seal', ' squirrel', ' sunflower', ' table', ' tiger', ' wolf']

MobileNet:

Η αρχιτεκτονική του Mobilenet είναι όπως φαίνεται παρακάτω, έχοντας 30 layers:

1. convolutional layer με stride 2
2. depthwise layer

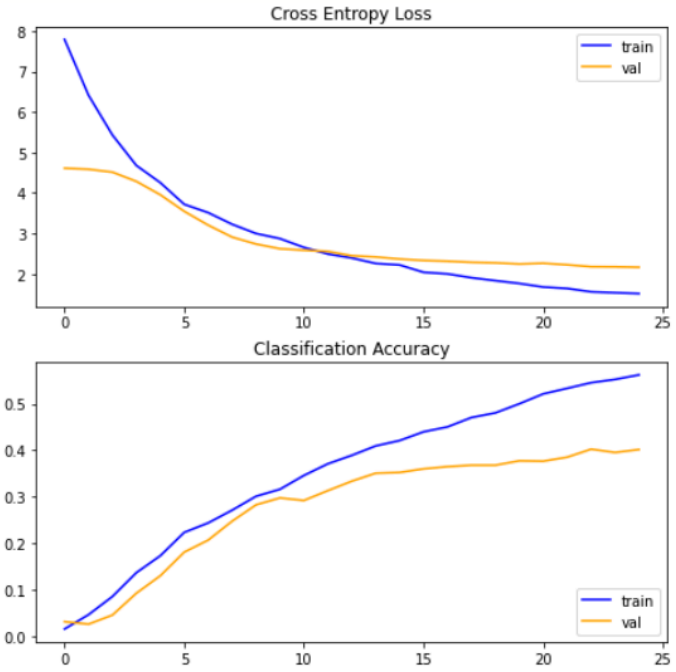
3. pointwise layer που διπλασιάζει τον αριθμό των channels
4. depthwise layer με stride 2
5. pointwise layer που διπλασιάζει τον αριθμό των channels
6. κ.ο.κ.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

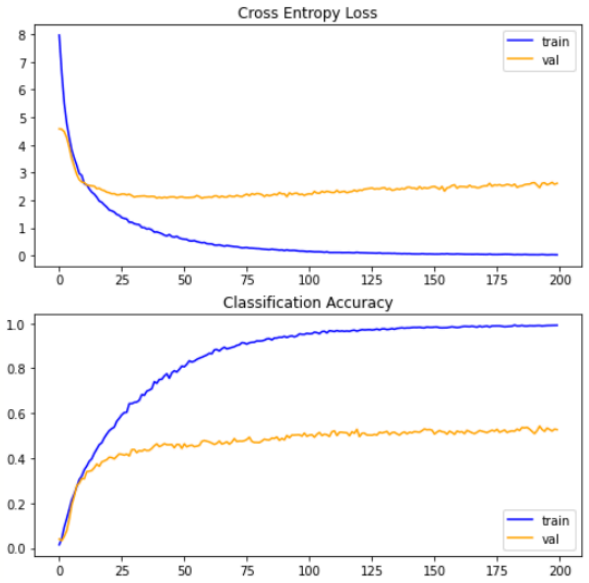
Το compilation χρησιμοποιεί τον Adam optimizer με learning rate 0.00005 ενώ έχει πραγματοποιηθεί prefetching των δεδομένων με batch size 128 και autotune. Με εκπαίδευση του μοντέλου σε 25 εποχές με 40 βήματα εκπαίδευσης και 10 βήματα validation ανά εποχή:

Παράμετρος βελτιστοποίησης (25 epochs)	**Τιμή**
Total parameters	3,331,364
Trainable parameters	3,309,476
Non-trainable parameters	21,888
Loss	2.13
Accuracy	0.41



Για πιο εποπτικό σχολιασμό των αποτελεσμάτων, αυξάνονται οι εποχές εκπαίδευσης σε 200 στο υπάρχον μοντέλο.

Παράμετρος βελτιστοποίησης (200 epochs)	**Τιμή**
Loss	2.51
Accuracy	0.53



Παρατηρούμε ότι πέρα από τις 100 περίπου εποχές, η μετρική του accuracy δε βελτιώνεται περαιτέρω. Συνεπώς, δε χρειάζεται να αυξήσουμε σε παραπάνω από 100 το πλήθος των εποχών.

Επίδραση υπερεκπαίδευσης

1. Μελέτη προστιθέμενων layers

Αρχικά, ελέγχθηκαν τα επιπλέον layers που έχουν προστεθεί στο μοντέλο για την αποφυγή της υπερκπαίδευσης. Παρατηρείται πως και τα δύο προστιθέμενα layers συμβάλλουν στην αποφυγή της υπερεκπαίδευσης, με καθοριστικότερη τη συμβολή του GPA. Τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα.

Αφαίρεση layer	**Loss**	**Accuracy**
Dropout	2.33	0.4
GPA	2.17	0.05
Dropout και GPA	2.39	0.05

2. Χρήση Dropout

Η τεχνική της προσθήκης του dropout layer εξαρτάται από τις διάφορες τιμές του rate του. Στον παρακάτω πίνακα παρουσιάζεται το πώς οι τιμές αυτές επιδρούν στις μετρικές.

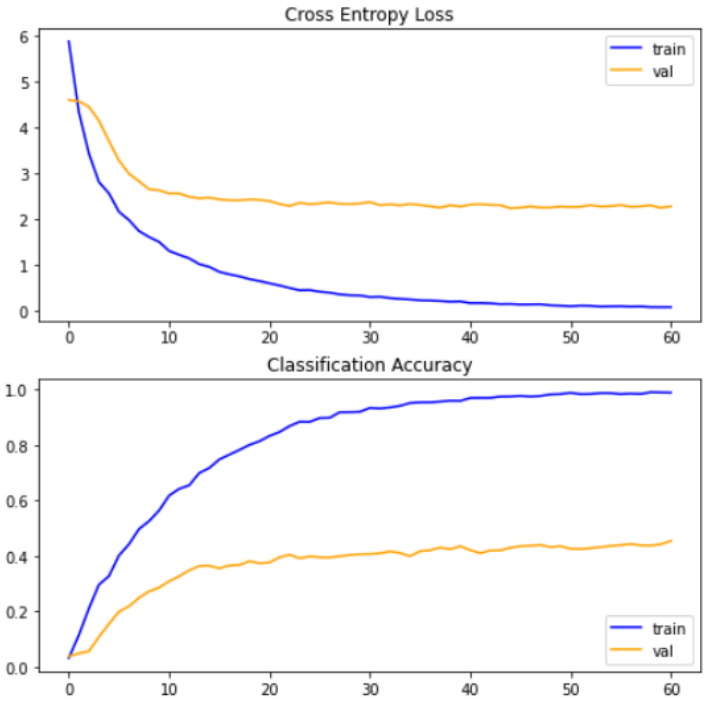
	Loss	**Accuracy**
Χωρίς Dropout layer	2.33	0.4
Dropout layer με rate 0.2	2.35	0.4
Dropout layer με rate 0.5	2.13	0.41
Dropout layer με rate 0.8	2.42	0.35

3. Χρήση Early Stopping

Ομοίως με τα προηγούμενα μοντέλα μεταφοράς μάθησης, θα χρησιμοποιηθεί η τεχνική του Early Stopping ως μια επιπλέον λύση του προβλήματος της υπερεκπαίδευσης. Οι παράμετροι είναι οι εξής: monitor='accuracy', mode='max', patience=30, min_delta=0.110. Το πλήθος των εποχών τέθηκε σε 200, δεδομένου ότι με χρήση του Early Stopping, η εκπαίδευση θα διακοπεί όταν για περισσότερες από 30 εποχές, η τιμή του accuracy δεν αυξηθεί περισσότερο από 0.110. Σημειώνεται, επίσης, ότι έχει αφαιρεθεί το dropout layer, ώστε να γίνει η σύγκριση του ποια από τις δύο μεθόδους είναι πιο αποτελεσματική στο δεδομένο πρόβλημα.

```
40/40 [==========================
Epoch 61/200
40/40 [==========================
Epoch 00061: early stopping
```

Η εκπαίδευση σταματάει στην εποχή 61, στην οποία όπως παρατηρείται το accuracy έχει σταθεροποιηθεί στην τιμή 0.44. Η τιμή του loss είναι ίση με 2.34. Η τιμή του accuracy είναι βελτιστοποιημένη σε σχέση με το Dropout, ενώ του loss χειρότερη.



4. Χρήση Data Augmentation

Ή αλλιώς επαύξηση δεδομένων. Η υπερεκπαίδευση συνήθως συμβαίνει όταν έχουμε λίγα ή/και πολύ όμοια δεδομένα εκπαίδευσης. Ένας τρόπος να διορθωθεί αυτό το πρόβλημα είναι να αυξήσουμε τα δεδομένα (data augmentation). Το data augmentation δημιουργεί νέα δεδομένα εκπαίδευσης με βάση τα υπάρχοντα εφαρμόζοντας τυχαίους μετασχηματισμούς ώστε να προκύπτουν αληθοφανείς εικόνες. Δοκιμάζοντας την τεχνική αυτή, τα αποτελέσματα ήταν:

Συνδυαστικά	**Παράμετροι (train + validation)**	**Epochs**	**Loss**	**Accuracy**
early stopping και dropout (rate 0.5)	horizontal flip	100	2.52	0.35
early stopping χωρίς dropout	horizontal flip	78	3.04	0.3
early stopping και dropout (rate 0.5)	horizontal flip και vertical flip	89	2.27	0.34
early stopping χωρίς dropout	horizontal flip και vertical flip	95	2.74	0.31

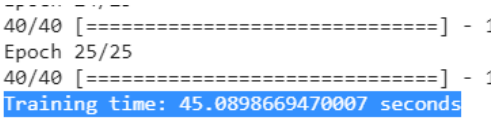
5. Συμπεράσματα

Συγκεντρωτικά τα βέλτιστα αποτελέσματα για τις μετρικές του accuracy και loss για κάθε μέθοδο περιορισμού της υπερεκπαίδευσης παρατίθενται στον παρακάτω πίνακα. Στην τρίτη σειρά έχουν εφαρμοστεί ταυτόχρονα και οι δύο τεχνικές για την εξάλειψη της υπερεκπαίδευσης, δηλαδή η τεχνική Dropout (rate = 0.5, 25 epochs) και η τεχνική Early Stopping (89 epochs όταν εφαρμόζεται μαζί με το dropout).

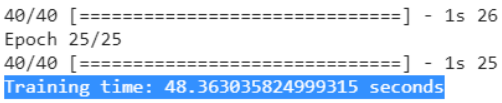
Τεχνική	**Loss**	**Max Accuracy**
Dropout (rate = 0.5, 25 epochs)	2.13	0.41
Early Stopping (51 epochs)	2.34	0.44
Both techniques	2.11	0.51
Data Augmentation (best case for accuracy)	2.52	0.35

Χρόνος εκπαίδευσης

Οι αρχικές μετρικές που έχουν παρουσιαστεί και παραπάνω προκύπτουν όταν για την εκπαίδευση του μοντέλου γίνεται data prefetching. Αυτή η τεχνική επιτρέπει όσο η εκπαίδευση του μοντέλου βρίσκεται στο βήμα κ, το input pipeline διαβάζει τα δεδομένα για το επόμενο βήμα εκπαίδευσης, κ+1. Για train για 40 steps, 25 εποχές, με adam optimizer (learning rate 0.00005) χρειάστηκαν:



Στη συνέχεια, αφαιρούμε την τεχνική του data prefetching, ώστε να δούμε πώς αυτό επιδρά στο χρόνο εκπαίδευσης. Ο χρόνος που χρειάστηκε τελικά ήταν ίσος με:



Παρακάτω εμφανίζονται και οι άλλες επιλογές ελαχιστοποίησης του χρόνου εκπαίδευσης και τι αποτελέσματα αυτές επέφεραν.

Τεχνική	**Χρόνος εκτέλεσης (seconds)**
Naive (batch - repeat -> shuffle)	Μη επαρκής μνήμη (session crashed)
Prefetching	35.35
Data reading parallelization	N/A (1 dataset loaded από 1 source)
Map transformation parallelization	N/A (δεν χρησιμοποιούνται map operations)
Caching (shuffle - repeat -> batch)	42.38
Reducing Memory Footprint (shuffle - repeat -> batch)	38.48
Reducing Memory Footprint (repeat - shuffle -> batch)	42.26

Εκπαίδευση βαρών (σημαία trainable)

1. Σημαία trainable ίση με false

Επιλέγουμε να παγώσουμε τη συνελικτική βάση και να εκπαιδεύσουμε την κεφαλή ταξινόμησης (classification head). Αυτό αντιστοιχεί στο να χρησιμοποιήσουμε τη συνελικτική βάση για εξαγωγή χαρακτηριστικών (feature extraction). Θέτουμε την σημαία trainable ίση με False. Στις default συνθήκες ήταν ίση με True. Οι μετρικές προκύπτουν: acc = 0.06, loss = 4.49.

2. Σημαία ανά επίπεδο

Εδώ επιλέγουμε να εκπαιδευτεί μόνο ένα ποσοστό των επιπέδων ευρισκόμενο προς την έξοδο του δικτύου. Οι σημαίες trainable εδώ ορίζονται ανά επίπεδο. Οι μετρικές προκύπτουν: acc = 0.42, loss = 2.17 για 5 layers με σημαία false και acc = 0.4, loss = 2.21 για 10 layers με σημαία false. Σημειώνεται ότι το MobileNet διαθέτει 30 layers στην αρχιτεκτονική του, με των οποίων την παράμετρο trainable μπορούμε να πειραματιστούμε. Εμείς κάναμε πειραματικά δύο μόνο δοκιμές.

3. Συνοπτικά συμπεράσματα

ΣΗΜΑΙΑ	**Loss**	**Max Accuracy**
True	2.13	0.41
False	4.49	0.06
false για 5 layers	2.17	0.42
false για τα 10 layers	2.21	0.4

Επίδραση επιλογής optimizer

1. Μελέτη διαφόρων optimizers

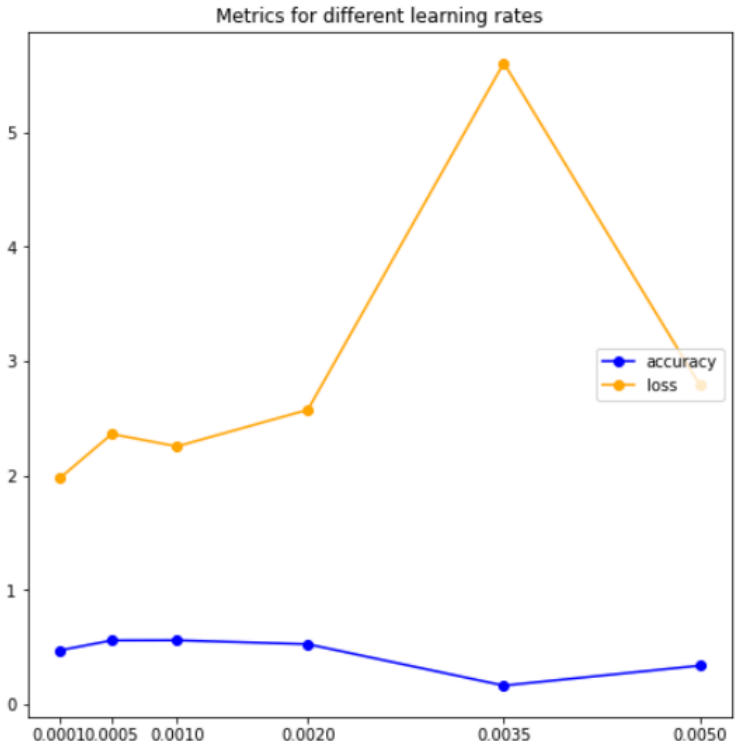
Ο default optimizer που χρησιμοποιήθηκε ήταν ο Adam, με learning rate 0.00005. Σύμφωνα με τις κατάλληλες βιβλιοθήκες, το default learning rate του Adam είναι 0.001. Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα για τέσσερις

διαφορετικούς optimizers, τον Adam (με 0.001 αντί 0.00005), τον SGD (0.01), τον RMSprop (0.001) και τέλος τον Adagrad (0.001).

Metrics	Adam (0.00005)	Adam (0.001)	SGD (0.01)	RMSprop (0.001)	Adagrad (0.001)
Accuracy	0.41	0.57	0.53	0.55	0.4
Loss	2.13	2.34	1.79	3.78	2.23

2. Βέλτιστο learning rate για δεδομένο optimizer

Στο σημείο αυτό έγινε προσομοίωση για την εύρεση του βέλτιστου learning rate για τον δεδομένο optimizer Adam. Τα αποτελέσματα που προέκυψαν, τα οποία φαίνονται παρακάτω, έδειξαν ότι η βέλτιστη τιμή ήταν ίση με 0.005.



Επίδραση πλήθους των κατηγοριών

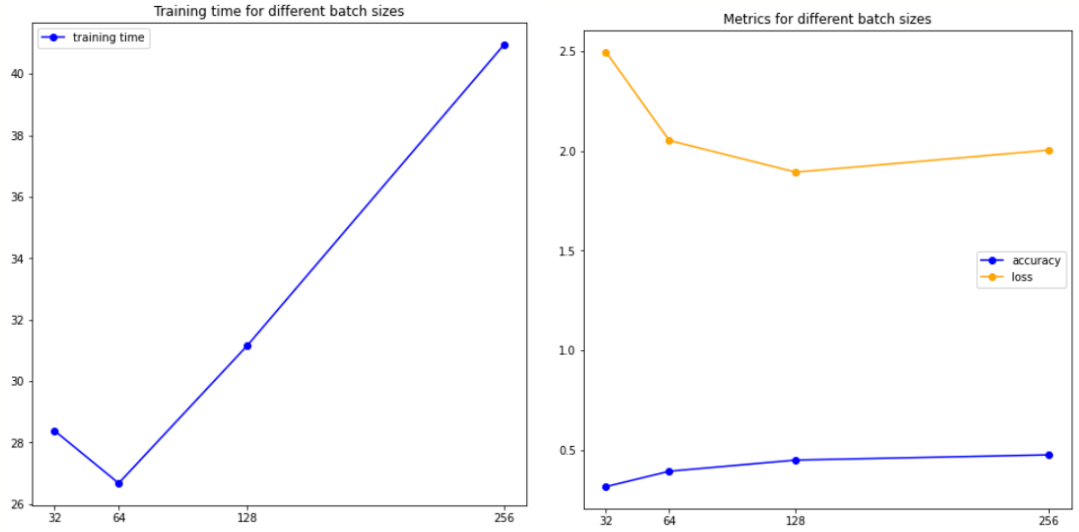
Ο default αριθμός κατηγοριών που εισάγεται είναι 20. Παρακάτω δοκιμάζονται και τέσσερα διαφορετικά νούμερα για τις εισαγόμενες κατηγορίες, καθώς και το πώς αυτά επιδρούν οι μετρικές (προστίθεται και η μετρική του χρόνου εκπαίδευσης).

Metrics	20	40	60	80
Accuracy	0.41	0.38	0.3	0.24
Loss	2.13	2.4	2.9	3.2
Time	45	48.85	53.4	68.65

Παρατηρούμε χειροτέρευση των μετρικών όσο αυξάνονται οι κατηγορίες.

Επίδραση του μεγέθους δέσμης (batch size)

Στη συνέχεια, εξετάστηκαν οι διάφορες τιμές του μεγέθους δέσμης, με στόχο την ελαχιστοποίηση του χρόνου εκπαίδευσης. Χρησιμοποιήθηκαν οι τιμές [32,64,128,256], οι οποίες είχαν σαν αποτέλεσμα την γραφική που ακολουθεί. Σε αυτήν φαίνεται ότι μικρότερο χρόνο εκπαίδευσης έχουμε για μέγεθος δέσμης ίσο με 64.



Συμπεριφορά ανά κλάση

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
2	0.00	0.00	0.00	100
6	0.00	0.00	0.00	100
10	0.00	0.00	0.00	0
12	0.00	0.00	0.00	0
13	0.00	0.00	0.00	0
18	0.00	0.00	0.00	100
24	0.00	0.00	0.00	100
26	0.00	0.00	0.00	100
29	0.00	0.00	0.00	100
33	0.00	0.00	0.00	0
35	0.00	0.00	0.00	100
40	0.00	0.00	0.00	0
41	0.00	0.00	0.00	0
42	0.00	0.00	0.00	100
43	0.00	0.00	0.00	0
44	0.00	0.00	0.00	0
45	0.00	0.00	0.00	100
50	0.00	0.00	0.00	0
51	0.00	0.00	0.00	100
55	0.00	0.00	0.00	100
58	0.00	0.00	0.00	100
59	0.00	0.00	0.00	0
61	0.00	0.00	0.00	0
62	0.17	0.01	0.02	100
65	0.00	0.00	0.00	100
68	0.00	0.00	0.00	0
69	0.00	0.00	0.00	0
72	0.00	0.00	0.00	100
76	0.00	0.00	0.00	0
80	0.00	0.00	0.00	100
82	0.00	0.00	0.00	100
84	0.00	0.00	0.00	100
88	0.00	0.00	0.00	100
93	0.00	0.00	0.00	0
97	0.00	0.00	0.00	100
99	0.00	0.00	0.00	0
accuracy			0.00	2000
macro avg	0.00	0.00	0.00	2000
weighted avg	0.01	0.00	0.00	2000

[' baby', ' bee', ' caterpillar', ' cockroach', ' crab', ' dinosaur', ' girl', ' leopard', ' lobster', ' mushroom', ' otter', ' pickup_truck', ' poppy', ' rabbit', ' seal', ' squirrel', ' sunflower', ' table', ' tiger', ' wolf']

Σύγκριση Μοντέλων From Scratch και Μοντέλων Transfer Learning

Εδώ παρουσιάζονται συνοπτικά και συγκριτικά τα καλύτερα αποτελέσματα, και οι συνθήκες που οδήγησαν σε αυτά, για τις τρεις μετρικές που εξετάστηκαν μέχρι τώρα, δηλαδή το accuracy, το loss, καθώς και το training time. Τα αποτελέσματα αυτά φαίνονται στον παρακάτω πίνακα.

Όσον αφορά το accuracy, παρατηρείται ότι η υψηλότερη τιμή επετεύχθη με το μοντέλο VGG16. Στην τιμή αυτή οδηγηθήκαμε με την αλλαγή του optimizer από *Adam(learning_rate=0.00005)* σε *SGD(learning_rate=0.01)*.

Όσον αφορά το loss, παρατηρείται ότι η χαμηλότερη τιμή επετεύχθη με το μοντέλο VGG16. Στην τιμή αυτή οδηγηθήκαμε με την μέθοδο ορισμού σημαίας trainable ίσης με false για τα top 5 output layers (επιλεκτική εκπαίδευση βαρών).

Όσον αφορά το χρόνο εκπαίδευσης, παρατηρείται ότι η χαμηλότερη τιμή επετεύχθη με το Μοντέλο 1 (from scratch). Στην τιμή αυτή οδηγηθήκαμε με την μέθοδο prefetching.

Μιας που το κύριο χαρακτηριστικό που ζητήθηκε να βελτιστοποιηθεί ήταν το accuracy, καταλήγουμε ότι σύμφωνα με τη μελέτη αυτής της εργασίας ***η επιλογή στην οποία καταλήγουμε είναι αυτή με το καλύτερο accuracy, δηλαδή το VGG16***.

Metric	Best case for this metric
Accuracy	0.7
Loss	1.23
Time	17.2

Φυσικά, δοκιμάσαμε μόνο δύο μοντέλα from scratch, καθώς και μόνο 3 μοντέλα transfer learning. Υπάρχουν ακόμη ποικίλες επιλογές διαμόρφωσης ενός μοντέλου from scratch και κάποιες από τις επιπλέον επιλογές για μοντέλα transfer learning είναι οι εξής: Inception-ResNet V2, Inception V3, NASNet-A, ResNet, ResNet v2, VGG19 και Xception V1.

Οι μεθοδολογίες βελτιστοποίησης μπορούν να εφαρμοστούν στα παραπάνω μοντέλα και ενδεχομένως να οδηγήσουν σε καλύτερα αποτελέσματα. Καθολικά βέλτιστη λύση δεν υπάρχει, μιας που πάντα εξαρτάται από τα εκάστοτε δεδομένα και επομένως η μελέτη πρέπει να γίνεται πάντοτε από την αρχή για το κάθε ξεχωριστό dataset.

References:

https://neurohive.io/en/popular-networks/vgg16/
https://arxiv.org/abs/1409.1556
https://medium.com/datadriveninvestor/cnn-architecture-series-vgg-16-with-implementation-part-i-bca79e7db415
https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/
https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
https://stackoverflow.com/questions/56207651/why-doesnt-the-accuracy-when-training-vgg-16-change-much
https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/
https://stackoverflow.com/questions/49915925/output-differences-when-changing-order-of-batch-shuffle-and-repeat