ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Νευρωνικά Δίκτυα και Ευφυή Υπολογιστικά Συστήματα

Άσκηση 3: Βαθιά Μάθηση

Ομάδα 01

Κωνσταντίνος Αλεξάνδρου **03116708**

Μαρία Παναγιώτου **03116703**

Ματθαίος Φικάρδος **03116705**

2021

Εισαγωγή

Στόχος της παρούσας εργασίας είναι η δημιουργία και βελτιστοποίηση μοντέλων Βαθιάς Μηχανικής Μάθησης στο σύνολο δεδομένων CIFAR-100. Στην ομάδα μας αντιστοιχεί ένα μοναδικό υποσύνολο κατηγοριών, το οποίο προκύπτει με την προσαρμογή του seed στον αριθμό της ομάδας μας.

Η ανάπτυξή των μοντέλων έγινε με Jupyter Notebooks στο περιβάλλον "**Google Colab**" και όχι τοπικά, αφού ήταν αναγκαία η επιτάχυνση της διαδικασίας με GPU, κάτι το οποίο μας προσφέρει η συγκεκριμένη πλατφόρμα.

Στην άσκηση έγινε μελέτη μοντέλων "from scratch" και μοντέλων με "μεταφορά μάθησης" από έτοιμα μοντέλα που υπάρχουν στις βιβλιοθήκες του TensorFlow. Η υλοποίηση της άσκησης έγινε υποχρεωτικά με την έκδοση TF2.

Δεδομένα και Υλοποίηση

Το σύνολο δεδομένων CIFAR-100 αποτελείται από 100 κλάσεις, η κάθε μία εκ των οποίων περιέχει 600 εικόνες. Οι εικόνες χωρίζονται σε 500 εικόνες εκπαίδευσης και 100 εικόνες δοκιμές. Η κάθε εικόνα είναι διαστάσεων 32x32x3. Για το validation set μας χρησιμοποιήθηκε το 15% του training set.

Στην υλοποίηση των μοντέλων έγινε πειραματισμός με μεγάλο αριθμό τιμών για τις παραμέτρους που χρησιμοποιήθηκαν για να βρεθεί η καλύτερη αρχιτεκτονική και διάταξη της. Έγινε χρήση data prefetching προκειμένου να επιτευχθεί μείωση του χρόνου εκπαίδευσης.

Κατά την εκπαίδευση, χρειάστηκε να περιοριστεί το φαινόμενο της υπερεκπαίδευσης. Για τον περιορισμό του χρησιμοποιήθηκαν τεχνικές:

Early Stopping: Μια μέθοδος που τερματίζει την εκπαίδευση αν δεν υπάρχει βελτίωση ως προς τη μετρική απόδοσης που παρακολουθούμε.

Dropout: Ένα είδος ομαλοποίησης (regularization) που επιβάλλει στα βάρη του δικτύου να παίρνουν μόνο μικρές τιμές. Εάν εφαρμόσουμε dropout σε ένα επίπεδο του δικτύου, τότε ένα ποσοστό των βαρών του γίνεται τυχαία μηδενικό κατά την εκπαίδευση.

Data Augmentation: 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. Στην εργασία αυτή χρησιμοποιήθηκαν rotation, width shift, height shift, horizontal flip και vertical flip.

Στα μοντέλα που υλοποιήθηκαν για τους σκοπούς της εργασίας, έγινε πειραματισμός με παραμέτρους όπως optimizers, learning rate, διαστάσεις εικόνας, αριθμός κλάσεων και τα συνολικά επίπεδα του δικτύου. Έγιναν δοκιμές με διάφορες τιμές για τις παραμέτρους που χρησιμοποιήθηκαν.

Μοντέλα “from scratch”

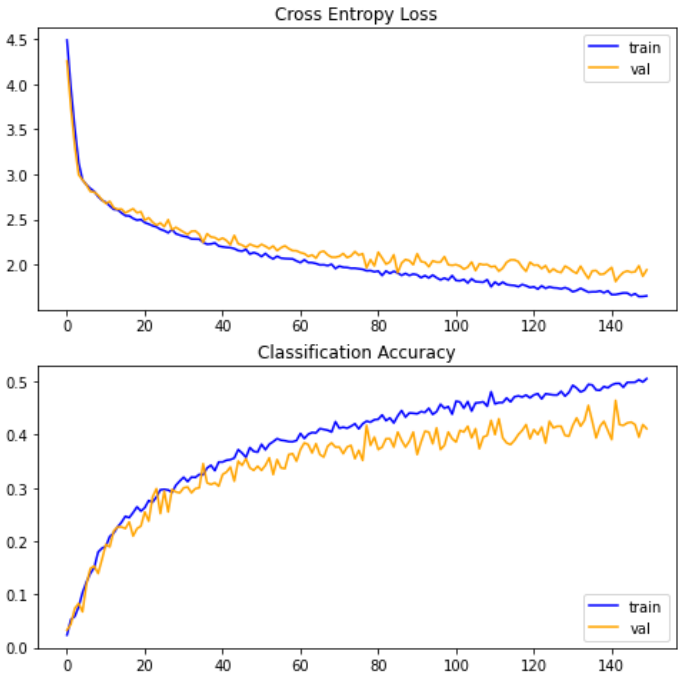
Για την υλοποίηση ενός μοντέλου μάθησης “from scratch” έγινε μελέτη από το διαδίκτυο και το υλικό που μας είχε δοθεί. Ακολούθησε πειραματισμός με τον αριθμό των επιπέδων, των φίλτρων και των παραμέτρων. Έγινε καταγραφή των αποτελεσμάτων για κάθε δοκιμή.

Ας εξετάσουμε το μοντέλο που μας δόθηκε αρχικά. Στο συγκεκριμένο συνελικτικό δίκτυο (Convolutional Neural Network – CNN) είχαμε 20 κλάσεις, 150 epochs εκπαίδευσης. Σε κάθε epoch είχαμε 40 steps και 5 validation steps.

Η αρχιτεκτονική του δικτύου ήταν:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 100)               6500
=================================================================
Total params: 128,420
Trainable params: 128,420
Non-trainable params: 0
```

Τα αποτελέσματα που μας δίνει το μοντέλο είναι ένα accuracy ίσο με **42%** και τιμή **loss = 1.96** .



Προχωρήσαμε στην κατασκευή του δικού μας μοντέλου με την ακόλουθη αρχιτεκτονική:

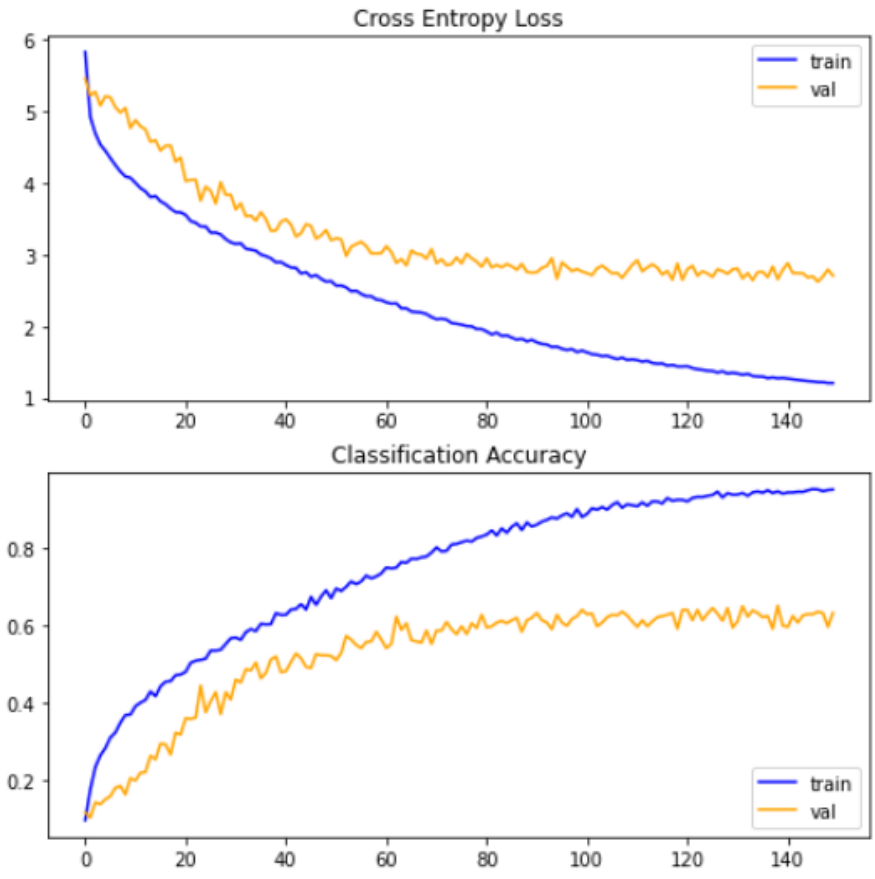
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 30, 30, 32)        128
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
batch_normalization_2 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 7, 7, 128)         147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 128)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 3, 3, 128)         512
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 3, 3, 256)         295168
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 256)         590080
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 1, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_4 (Dense)              (None, 100)               25700
=================================================================
Total params: 1,264,900
Trainable params: 1,264,324
Non-trainable params: 576
```

Σχεδιαστικές αποφάσεις:

Μετά από δοκιμές με τους optimizers, "Adam", "SGD" και "Nadam" συμπεράναμε ότι ο καλύτερος optimizer είναι ο "Adam" με learning rate = 0.00015(3% ψηλότερη απόδοση από το 0.0001). Συγκεκριμένα με τον "Adam" πετύχαμε μια αύξηση 3% σε σχέση με τον δεύτερο καλύτερο optimizer που ήταν ο "SGD" με 58%. Το Batch Size ρυθμίστηκε στα 128.
Συγκρίναμε τις συναρτήσεις Exponential Linear Unit (ELU) και Rectified Linear Units (RELU) και παρατηρήσαμε ότι με την χρήση της ELU είχαμε αύξηση της επίδοσης του μοντέλου κατά 2%.
Συγκριτικά με το αρχικό μοντέλο που μας δόθηκε, η προσθήκη επιπλέον επιπέδων συνέλιξης βελτίωσε την απόδοση, αλλά η προσθήκη περισσοτέρων επιπέδων από αυτά που βάλαμε, επιδρά αρνητικά στην απόδοση του μοντέλου.
Μετά από κάθε 2 επίπεδα συνέλιξης προστέθηκαν επίπεδα Dropout για την αποφυγή του overfitting στα οποία καταλήξαμε μετά από δοκιμές ότι οι ιδανικές τιμές τους βρίσκονται στο εύρος 0.25-0.4.
Η συνάρτηση ενεργοποίησης του output layer είναι η "softmax" και η Loss Function είναι η Sparse Categorical Cross Entropy. Σε κάθε layer γίνεται χρήση ως weight decay η l1 norm με λ=0.001, τιμή που βελτιώνει την απόδοση του μοντέλου κατά 1% σε σχέση με την λ=0.0001 .

Τα καλύτερα μας αποτελέσματα προέκυψαν ρυθμίζοντας τις τιμές παραμέτρων όπως αναφέρθηκαν πιο πάνω και με χρήση 20 κλάσεων και 150 epochs εκπαίδευσης, όπου σε κάθε epoch είχαμε 40 steps και 5 validation steps.

Τα αποτελέσματα που μας δίνει το μοντέλο είναι ένα accuracy ίσο με **61%** και τιμή **loss = 2.73** .



Μοντέλα μεταφοράς μάθησης

Στη συνέχεια θα εξετάσουμε την απόδοση μοντέλων μέσω της τεχνικής της μεταφοράς μάθησης από μοντέλο που έχουν εκπαιδευτεί στο τεράστιο σύνολο του ImageNet. Τα μοντέλα που θα εξετάσουμε είναι τα **VGG16**, **VGG19** και **DenseNet**. Στα μοντέλα αυτά μελετήσαμε εκτός από τις προαναφερόμενες, άλλες δύο τεχνικές για να βελτιστοποιήσουμε τα αποτελέσματα μας.

Fine Tuning: Επιλέγουμε να εκπαιδεύσουμε, όλα, κανένα ή ένα μέρος από τα επίπεδα της αρχιτεκτονικής που έχει το κάθε μοντέλο με σκοπό να εντοπίσουμε την καλύτερη έκδοση κάθε αρχιτεκτονικής. (Στους πίνακες που ακολουθούν False σημαίνει δεν χρησιμοποιήθηκε η τεχνική, True σημαίνει έγινε training στο top layer και μετά σε όλο το μοντέλο.)

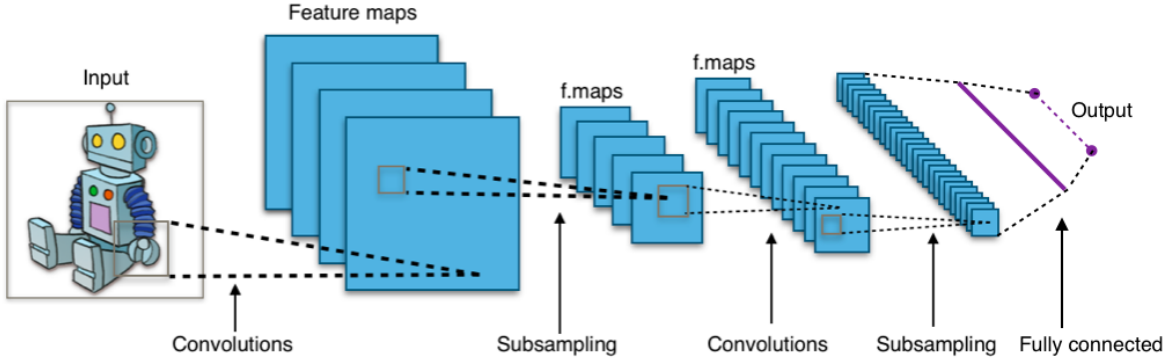
Resize: Τροποποιούμε τις διαστάσεις των εικόνων του συνόλου δεδομένων ούτως ώστε να δούμε αν βελτιώνεται η απόδοση των μοντέλων.

Στα πλαίσια του πειραματισμού μας με τα μοντέλα εκτελέστηκαν πολλές διαφορετικές δοκιμές, οι οποίες δεν θα παρουσιαστούν όλες στην αναφορά που ακολουθεί, καθώς θα παρουσιάσουμε μόνο κάποιες σημαντικές, είναι όμως διαθέσιμες για προβολή εδώ.

Σημαντικές παρατηρήσεις σχετικά με το Google Colab

Κατά την διάρκεια των δοκιμών μας αντιμετωπίσαμε περιορισμούς από την πλατφόρμα Google Colab. Η υπηρεσία έκανε crash και αδυνατούσε να μας παρέχει αποτελέσματα για μέγεθος εικόνων πάνω από 160x160 στις 20 κλάσεις, και 64x64 στις 80 κλάσεις αντίστοιχα.

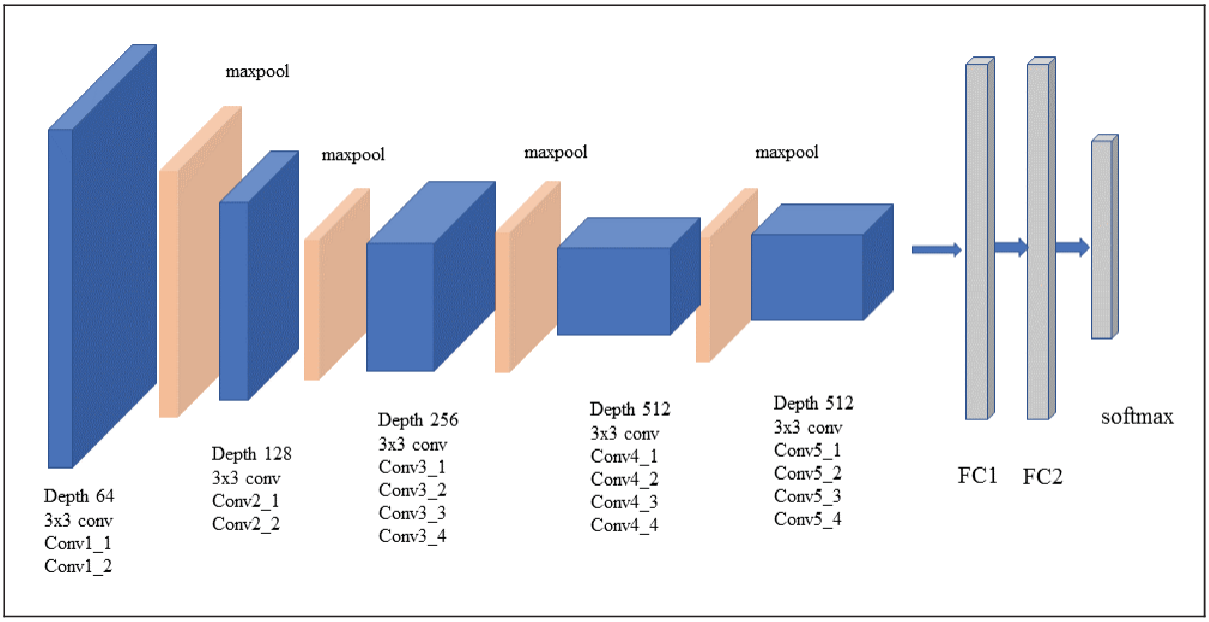
Αντιμετωπίσαμε επιπλέον περιορισμούς στη χρήση των GPUs της Google αφού μετά από κάποιες επαναλήψεις, η υπηρεσία δε μας επέτρεπε τη χρήση GPU Runtime μέχρι να περάσουν κάποιες ώρες.



"Typical CNN Architecture". Πηγή Wikipedia

VGG19

Η αρχιτεκτονική του μοντέλου φαίνεται στην παρακάτω εικόνα:



Δικαιώματα *Sai Charan Arishanapally*

Το μοντέλο αποτελείται από συνελικτικά επίπεδα, επίπεδα max pooling και η συνάρτηση ενεργοποίησης την ReLU. Στο output layer η συνάρτηση ενεργοποίησης είναι η softmax.

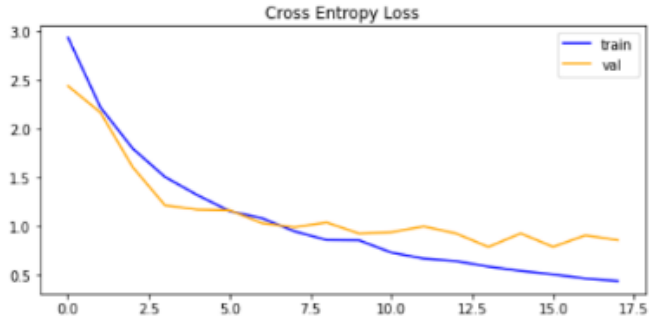
Το train set μας εκπαιδεύτηκε σε 50 epochs με 40 βήματα ανά εποχή και 10 βήματα ανά validation. Ο optimizer ήταν σταθερός στον "Adam". Οι διάφορες παράμετροι που χρησιμοποιήθηκαν και οι τιμές τους φαίνονται στον παρακάτω πίνακα, που παρουσιάζει μόνο μερικές από τις δοκιμές:

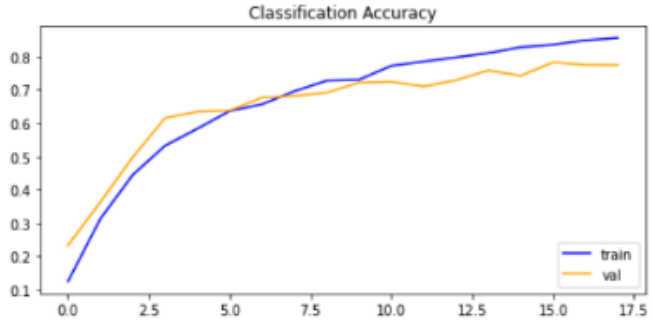
Parameters								Αποτελέσματα		
EarlyStop	FineTune	Dropout	Data Augmentation	Resize	Classes	Batch Size	LR	Time/ epoch	Accuracy	Loss
Yes (15)	False	0.5	False	False	20	32	1.50E-04	20	55%	1.52
Yes (14)	False	0.5	False	False	20	32	1.00E-05	20	63%	1.26
False	True	False	False	False	20	32	1.50E-04	-	73%	1.82
False	True	0.5	False	False	20	32	1.50E-05	-	80%	1.16
False	False	0.5	False	False	40	32	1.50E-05	20	71%	1.92
False	False	False	True	False	20	32	1.50E-04	10	73%	0.99
False	True	0.5	False	160x160	20	64	1.5E-04	-	87%	0.48
Yes (18)	False	0.5	True	160x160	20	128	1.5E-04	-	90%	0.45
Yes (20)	False	0.5	False	64x64	80	128	1.5E-04	-	80%	0.85

Παρατηρούμε ότι το μοντέλο μας αποδίδει αρκετά καλά και έχει μεγάλη δυνατότητα βελτίωσης στην απόδοση με τις σωστές παραμέτρους. Συγκεκριμένα όπως μπορούμε να

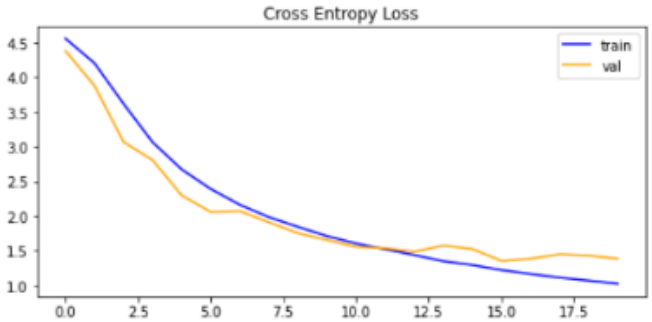
δούμε το μοντέλο αποδίδει καλύτερα με μεγάλο learning rate και batch size στα 128. Σημαντική βελτίωση προσφέρει το Resize που έγινε στα δεδομένα, ενώ και το Data Augmentation βοηθάει σε κάποιο βαθμό. Για την αποφυγή overfitting χρησιμοποιήσαμε ένα Dropout Layer ρυθμισμένο στο 50% και τη μέθοδο Early Stop. Παρατηρούμε επίσης ότι το μοντέλο μας αποδίδει καλύτερα εάν γίνει εκπαίδευση σε όλα τα επίπεδα και όχι μόνο στο top layer.

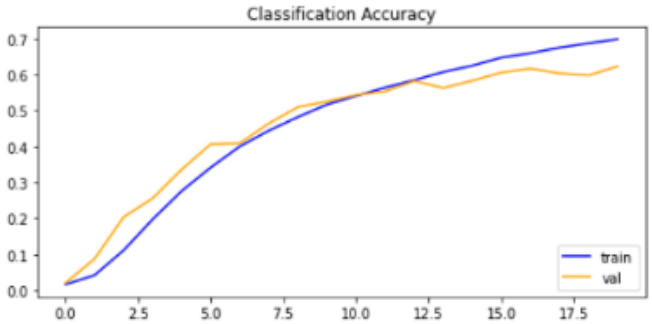
Ακολουθούν αποτελέσματα του μοντέλου σε 20 και 80 κλάσεις αντίστοιχα.






```
Test set evaluation metrics
30/30 [==============================] - 1s 12ms/step - loss: 0.4506 - accuracy: 0.9000
loss: 0.45
accuracy: 0.90
```

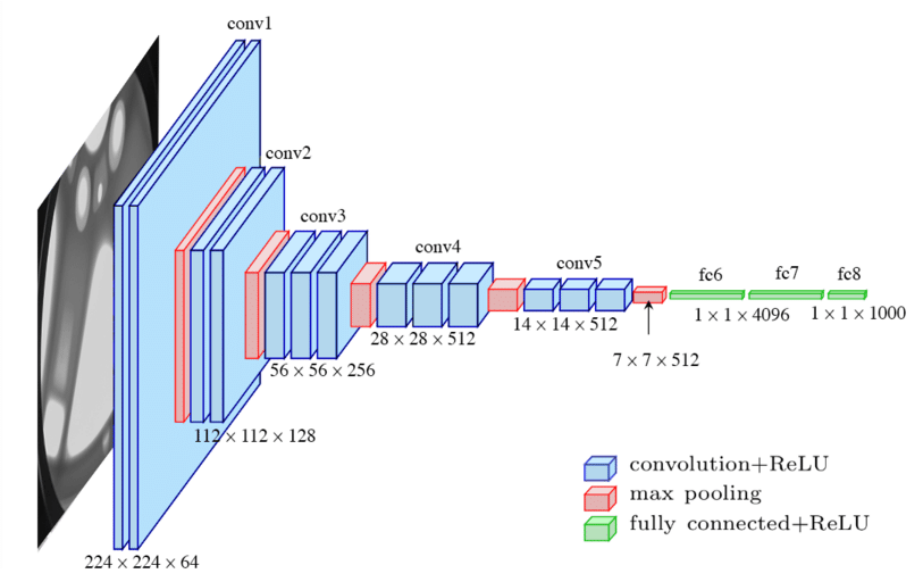





```
Test set evaluation metrics
30/30 [==============================] - 1s 5ms/step - loss: 0.8554 - accuracy: 0.8000
loss: 0.86
accuracy: 0.80
```

VGG16

Η αρχιτεκτονική του μοντέλου φαίνεται στην παρακάτω εικόνα:



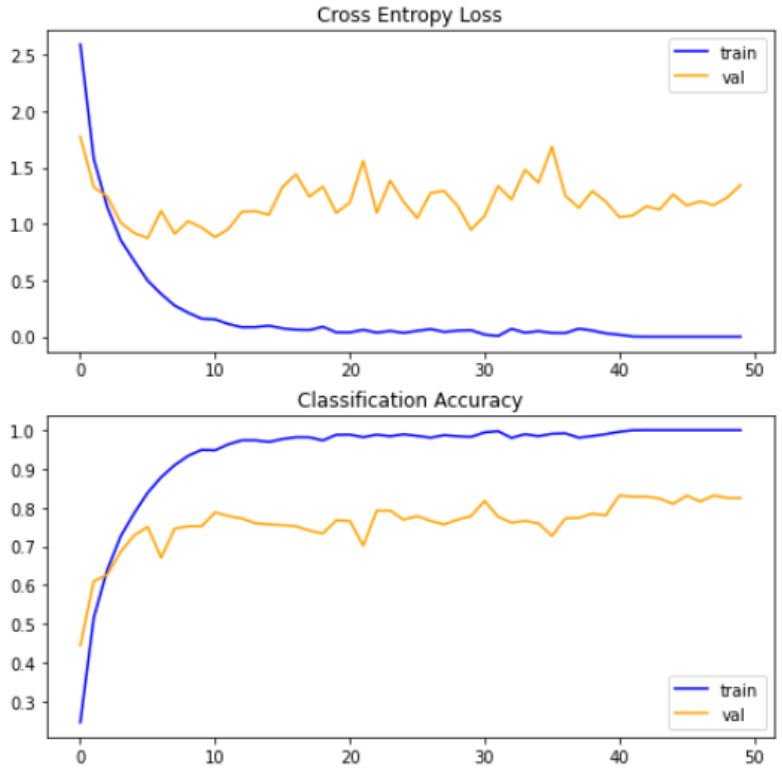
Πηγή researchgate.net

Το μοντέλο αποτελείται από συνελικτικά επίπεδα, επίπεδα max pooling και η συνάρτηση ενεργοποίησης την ReLU. Στο output layer η συνάρτηση ενεργοποίησης είναι η softmax.

Το train set μας εκπαιδεύτηκε σε 50 epochs με 40 βήματα ανά εποχή και 10 βήματα ανά validation. Ο optimizer ήταν σταθερός στον "Adam". Οι διάφορες παράμετροι που χρησιμοποιήθηκαν και οι τιμές τους φαίνονται στον παρακάτω πίνακα, που παρουσιάζει μόνο μερικές από τις δοκιμές:

Parameters								Αποτελέσματα		
EarlyStop	FineTune	Dropout	Data Augmentation	Resize	Classes	Batch Size	LR	Time/ epoch	Accuracy	Loss
Yes (15)	False	0.5	False	False	20	64	1.50E-05	13	67%	1.13
False	False	0.5	False	False	20	64	1.50E-05	3s	70%	1.74
False	False	0.5	False	False	20	128	1.50E-05	3s	70%	1.79
False	False	0.5	False	False	40	32	1.00E-05	10s	70%	1.78
False	False	0.5	False	False	80	64	1.50E-05	13	62%	2.47
False	False	False	True	False	20	32	1.50E-04	8s	63%	1.49
False	False	False	True	False	20	64	1.00E-05	6s	53%	2.34
False	True	False	False	160x160	20	128	1.00E-04	-	77%	1.71
False	True	0.5	True	160x160	20	64	1.50E-04	-	83%	1.15
False	true	0.5	False	64x64	80	64	1.50E-04	-	67%	2.13

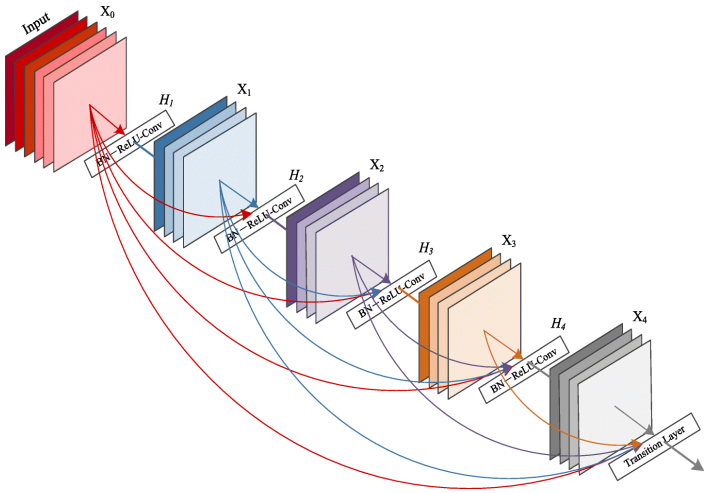
Παρατηρούμε ότι το μοντέλο μας έχει ικανοποιητική απόδοση. Συγκεκριμένα όπως μπορούμε να δούμε το μοντέλο αποδίδει καλύτερα με μεγάλο learning rate (0.0001). Σημαντική βελτίωση προσφέρει το Resize που έγινε στα δεδομένα. Το Data Augmentation βελτιώνει σε κάποιο βαθμό την απόδοση του μοντέλου, όπως φαίνεται από τις δοκιμές στον πίνακα. Για την αποφυγή overfitting χρησιμοποιήσαμε ένα Dropout Layer ρυθμισμένο στο 50%, που όπως βλέπουμε στις δύο καλύτερες δοκιμές, βελτιώνει την απόδοση. Όπως φαίνεται στο σχήμα, το overfitting δεν μπόρεσε να εξαλειφθεί εντελώς. Κάτι τέτοιο θα μπορούσε να αποφευχθεί με Early Stop. Παρατηρούμε επίσης ότι το μοντέλο μας αποδίδει καλύτερα εάν γίνει εκπαίδευση με Fine Tuning εκπαιδεύοντας το top layer ενώ το υπόλοιπο μοντέλο είναι frozen.




```
Test set evaluation metrics
30/30 [==============================] - 2s 17ms/step - loss: 1.1576 - accuracy: 0.8333
loss: 1.16
accuracy: 0.83
```

DenseNet

Η αρχιτεκτονική του μοντέλου φαίνεται στην παρακάτω εικόνα:



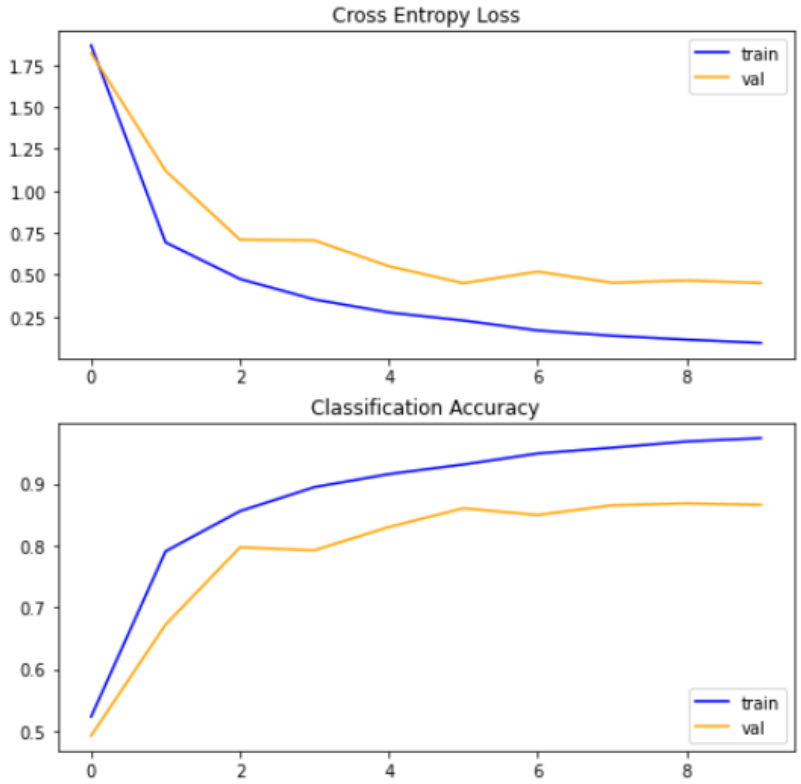
Πηγή Ai-Pool

Το train set μας εκπαιδεύτηκε σε 50 epochs με 40 βήματα ανά εποχή και 10 βήματα ανά validation. Ο optimizer ήταν σταθερός στον "Adam". Οι διάφορες παράμετροι που χρησιμοποιήθηκαν και οι τιμές τους φαίνονται στον παρακάτω πίνακα, που παρουσιάζει μόνο μερικές από τις δοκιμές:

Parameters								Αποτελέσματα		
EarlyStop	FineTune	Dropout	Data Augmentation	Resize	Classes	Batch Size	LR	Time/ epoch	Accuracy	Loss
False	False	0.5	False	False	20	128	1.00E-05	4s	55%	1.66
False	False	0.5	False	False	20	32	1.00E-04	9s	70%	1.72
False	yes	False	False	False	20	32	1.50E-04	-	77%	1.54
False	False	0.5	False	False	20	32	1.50E-04	10s	70%	1.55
False	False	False	True	False	20	64	1.00E-05	8s	73%	1.01
False	True	0.5	False	160x160	20	64	1.5E-04	-	83%	0.63
Yes (10)	False	0.5	False	64x64	80	128	1.5E-04	-	73%	0.76
Yes (10)	False	0.5	False	160x160	20	128	1.5E-04	-	90%	0.33
False	True	0.5	True	False	20	64	1.50E-04	-	64%	1.64
False	False	0.5	False	64x64	80	128	1.5E-04	-	73%	1.24

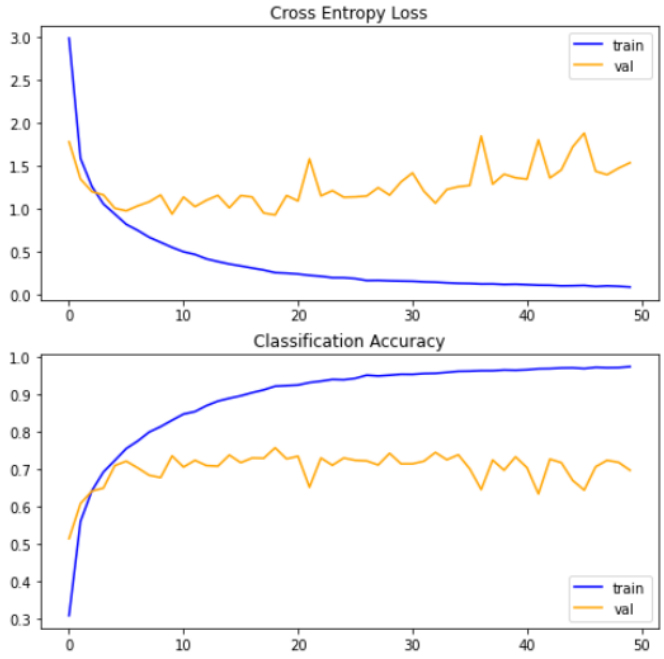
Παρατηρούμε ότι το μοντέλο μας αποδίδει αρκετά καλά και έχει μεγάλη δυνατότητα βελτίωσης στην απόδοση με τις σωστές παραμέτρους. Συγκεκριμένα όπως μπορούμε να δούμε το μοντέλο αποδίδει καλύτερα με μεγάλο learning rate και batch size στα 128. Σημαντική βελτίωση προσφέρει το Resize που έγινε στα δεδομένα. Το Data Augmentation δεν βελτιώνει την απόδοση του μοντέλου. Για την αποφυγή overfitting χρησιμοποιήσαμε ένα Dropout Layer ρυθμισμένο στο 50% και τη μέθοδο Early Stop. Παρατηρούμε επίσης ότι το μοντέλο μας αποδίδει καλύτερα εάν γίνει εκπαίδευση σε όλα τα layers και όχι μόνο στο top layer, οπότε είναι προτιμότερο να μη γίνει Fine Tuning.

Η απόδοση για 20 κλάσεις είναι αρκετά ψηλή, ενώ όπως είναι αναμενόμενο, πέφτει για μεγαλύτερο αριθμό κλάσεων, αλλά παραμένει ικανοποιητική.



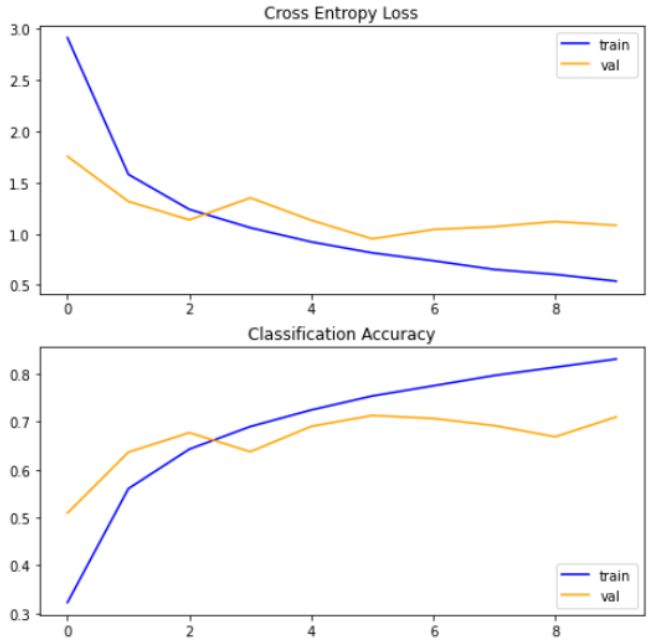

```
Test set evaluation metrics
30/30 [==============================] - 2s 10ms/step - loss: 0.3267 - accuracy: 0.9000
loss: 0.33
accuracy: 0.90
```

Αποτέλεσμα για 20 κλάσεις




```
Test set evaluation metrics
30/30 [==============================] - 1s 10ms/step - loss: 1.2476 - accuracy: 0.7333
loss: 1.25
accuracy: 0.73
```

Αποτέλεσμα για 80 κλάσεις χωρίς early stop




```
Test set evaluation metrics
30/30 [==============================] - 1s 9ms/step - loss: 0.7632 - accuracy: 0.7333
loss: 0.76
accuracy: 0.73
```

Αποτέλεσμα για 80 κλάσεις με early stop

Παρατηρήσεις – Συμπεράσματα

Μεταφορά μάθησης vs εκπαίδευση εκ του μηδενός (“from scratch”)

Χρησιμοποιώντας μοντέλο δικής μας κατασκευής αλλά και μοντέλο state of the art με μεταφορά μάθησης μπορούμε ξεκάθαρα να πούμε πως τα τελευταία αποδίδουν καλύτερα.

Επίδραση της επαύξησης δεδομένων (data augmentation)

Σε γενικές γραμμές ως επί το πλείστο, και με προσεκτική εφαρμογή η επαύξηση δεδομένων είναι σίγουρα βοηθητική όπως διαπιστώσαμε στα 2 από τα 3 μοντέλα μας. Στο μοντέλο DenseNet που δοκιμάσαμε κάποιες φορές δεν είχαμε βελτίωση στην απόδοση μας, αλλά αυτό δε σημαίνει ότι δεν είναι χρήσιμη τεχνική.

Επίδραση του πλήθους των επιπέδων που θα εκπαιδευτούν (fine-tuning) κατά τη μεταφορά μάθησης

Στα μοντέλα μεταφοράς μάθησης που χρησιμοποιήσαμε πειραματιστήκαμε με διάφορες τιμές για τα επίπεδα του μοντέλου αλλά δεν μπορέσαμε να βρούμε κάποιο καλύτερο αποτέλεσμα από το default. Σε μεταγενέστερο στάδιο θα θέλαμε να εξετάσουμε πιο διεξοδικά για όλους τους πιθανούς συνδυασμούς layers για να εντοπίσουμε που βρίσκεται το βέλτιστο στα μοντέλα μας.

Μεταφορά μάθησης vs εκπαίδευση εκ του μηδενός (“from scratch”)

Όσο αυξάνουμε τις κλάσεις δεδομένων του συνόλου μας, αυξάνεται και ο βαθμός δυσκολίας του προβλήματος, γι’ αυτό και η απόδοση του μοντέλου πάντα μειώνεται.

Επίδραση του ρυθμού μάθησης (learning rate)

Ο ρυθμός μάθησης εξαρτάται άμεσα από το μοντέλο που εκπαιδεύουμε και από την επιλογή του optimizer. Στα δικά μας μοντέλα παρατηρήθηκε σταθερά καλύτερη απόδοση με τον Adam για τιμές ρυθμού μάθησης της τάξης του 10-4.

Επίδραση του αλγόριθμου βελτιστοποίησης (optimizer)

Οι optimizers που δοκιμάσαμε είχαν αποτελέσματα αρκετά κοντά σε ποσοστά επιτυχίας, όμως παρατηρήθηκε ότι ο Adam είχε σταθερά καλύτερη απόδοση από τους υπόλοιπους.

Επίδραση του μεγέθους δέσμης (batch size)

Αν και στο πρόβλημα μας δεν φάνηκε τρομερή διαφορά στην απόδοση του μοντέλου, συμπεράναμε ότι τα καλύτερα μεγέθη δέσμης για τα μοντέλα μας ήταν 64 και 128, αναλόγως του μοντέλου. Πρέπει να σημειωθεί ότι όσο μεγαλύτερο είναι το μέγεθος δέσμης, τόσο λιγότερο χρόνο ανά epoch χρειάζεται το μοντέλο για να εκπαιδευθεί.

Επίδραση του μεγέθους των εικόνων (resize input)

Παρατηρήσαμε ότι όταν αλλάζαμε το μέγεθος των εικόνων σε μεγαλύτερες διαστάσεις, (πχ. από 32x32x3 σε 160x160x3) παίρναμε καλύτερα αποτελέσματα στα μοντέλα με μεταφορά μάθησης. Δεν ισχύει κάτι αντίστοιχο στα μοντέλα “from scratch”. Οι μεγαλύτερες διαστάσεις που μπορέσαμε να εξετάσουμε ήταν 160x160x3 (βλ. παρατηρήσεις).