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

ΟΜΑΔΑ: ΜΑ7

ΜΕΛΗ
ΚΟΝΤΟΓΙΑΝΝΗΣ ΑΝΔΡΕΑΣ (03115187)
ΣΠΥΡΟΥ ΝΙΚΟΛΑΟΣ (03116201)
ΠΕΠΠΑΣ ΠΑΝΑΓΙΩΤΗΣ (03115146)

1. Εισαγωγή

Θέμα της παρακάτω μελέτης αποτελεί η εισαγωγή στις Deep Learning τεχνικές και η εξοικείωση με το TensorFlow 2.1.0 για την επιλύση του Image Classification προβλήματος του CIFAR-100 Dataset. Πιο αναλυτικά, στην μελέτη μας θα χρησιμοποιήσουμε ένα υποσύνολο αυτού του dataset, και μάλιστα θα χρησιμοποιήσουμε 20 από τις 100 διαφορετικές κατηγορίες (labels) του πρωτοτύπου. Οι κατηγορίες που θα χρησιμοποιηθούν είναι οι εξής:

['apple', 'bicycle', 'camel', 'chair', 'cockroach', 'crab', 'kangaroo', 'lamp', 'lawn_mower', 'plate', 'porcupine', 'raccoon', 'rose', 'skunk', 'streetcar', 'sunflower', 'tiger', 'trout', 'turtle', 'woman']

Επιπλέον, το dataset μας περιέχει συνολικά 12,000 δείγματα, από τα οποία παίρνουμε τα 8500 για training, τα 1500 για validation και τα υπόλοιπα 2000 για testing. Σημειώνουμε πως καθόλη την διάρκεια της μελέτης μας τα παραπάνω τρία σύνολα δεδομένων μένουν αμετάβλητα, δηλαδή δεν ξαναδιαχωρίζουμε το αρχικό μας dataset.

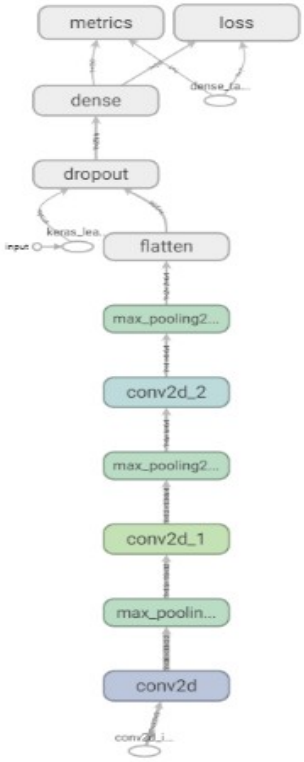
Ζητούμενο της εργασίας μας αποτελεί η ανάπτυξη αποτελεσματικών αρχιτεκτονικών βαθιών νευρωνικών δικτύων για την βέλτιστη επίδοση τους στην ταξινομήση των εικόνων, λαμβάνοντας υπόψιν το test accuracy score, το χρόνο εκτέλεσης της εκπαίδευσης, το σύνολο των παραμέτρων του κάθε μοντέλου καθώς και πολλές άλλες παραμέτρους βελτιστοποιήσης που θα συζητηθούν εκτενώς παρακάτω. Ωστόσο, για να θεωρήσουμε ότι γίνεται βελτιστοποιήση ως προς ένα από τα παραπάνω, χρειάζεται να υπάρχει πάντοτε και ένα ικανοποιητικό test accuracy score.

Θα ξεκινήσουμε με την δημιουργία ενός απλού CNN και ενός state-of-the-art δικτύου και καλούμαστε να τα βελτιστοποιήσουμε και τα δύο και να βγάλουμε αντίστοιχα συμπεράσματα. Σημειώνουμε πως η ανάπτυξη της μελέτης μας πραγματοποιήθηκε στο Google Colab με την χρήση της default GPU και της RAM, που παρέχει αυτό δωρεάν. Για όλες τις αρχιτεκτονικές θα χρησιμοποιήσαμε data prefetching, προκειμένου να πετύχουμε γρήγορη εκπαίδευση.

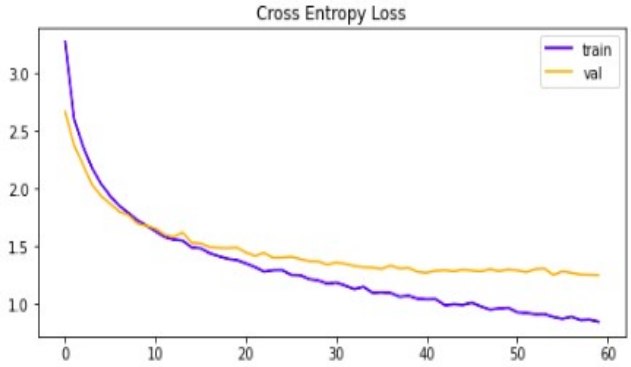
2. Αρχικές μη βελτιστοποιημένες αρχιτεκτονικές

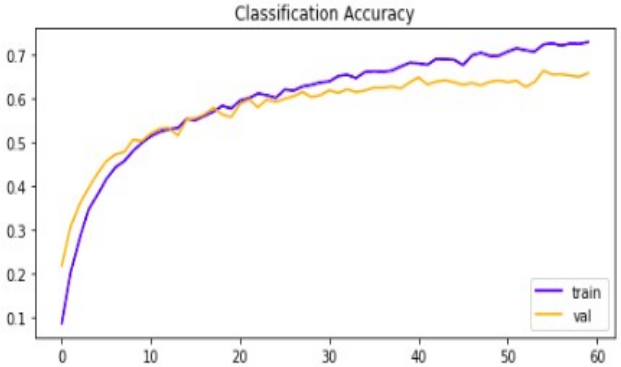
2.1 From Scratch (CNN3)

Για αρχή, φτιάχνουμε ένα from scratch CNN με 3 convolutional layers (όπου το καθένα αποτελείται από Conv2D και MaxPooling), 1 flatten layer, που μας βοηθάει να κάνουμε flatten τις διαστάσεις, και 1 fully-connected με Dropout για την αποφυγή του overfitting. Η συνάρτηση ενεργοποιήσης του output layer είναι η softmax και η loss function είναι η sparse categorical cross entropy, προκειμένου να γίνει ταξινόμηση στις 20 πιθανές κατηγορίες. Σημειώνουμε πως αυτή η μέθοδος είναι όμοια με το logistic regression του κλασικού machine learning. Αναπαριστούμε, παρακάτω, με χρήση του TensorBoard το model graph της αρχιτεκτονικής αυτής.



Εκπαιδεύοντας το μοντέλο αυτό για 60 εποχές παίρνουμε τα παρακάτω learning curves:

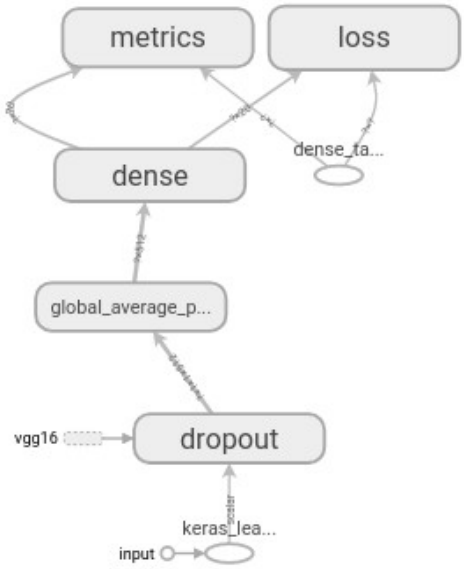




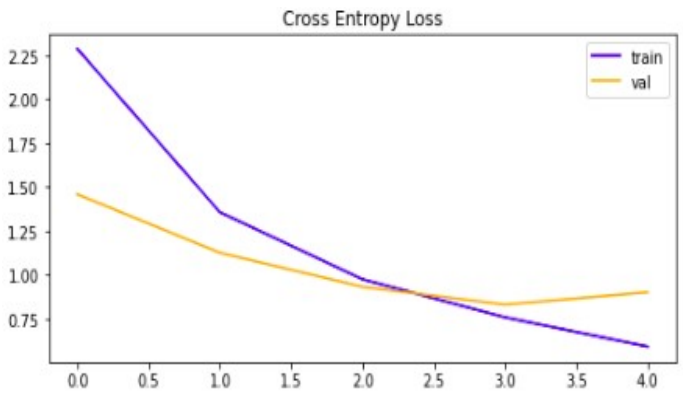
Το accuracy στο test score είναι 63%. Βλέπουμε πως το παραπάνω μοντέλο επιδέχεται μεγάλη βελτίωση, δεδομένου ότι γίνεται underfitting, αφού και το train_loss και το val_loss είναι φθίνουσες συναρτήσεις μέχρι τέλους, ενώ τα αντίστοιχα accuracy είναι αύξουσες. Θα συζητήσουμε στην ενότητα 2.2 εκτενώς για την βελτιστοποιήση του from scratch μοντέλου.

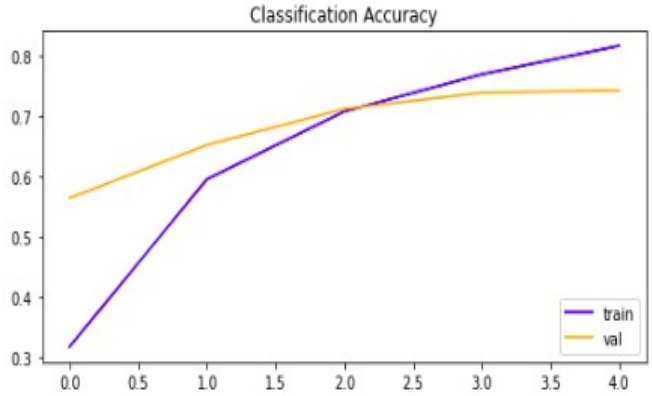
2.2 VGG16 (init)

Αφού είδαμε την χαμηλή επίδοση του παραπάνω μοντέλου, θα χρησιμοποιήσουμε transfer learning, προκειμένου να δούμε αυτή τη φορά καλύτερα αποτελέσματα. Πιο αναλυτικά, θα γίνει χρήση του VGG16 που έχει προεκπαιδευμένα βάρη και θα του προσθέσουμε dropout, GlobalAveragePooling2D και ένα fully-connected. Ορίζουμε την μεταβλητή trainable=True, προκειμένου να εκπαιδεύσουμε όλα τα βάρη, εκπαιδευμένα και μη. Αν δεν είχαμε ορίσει την παραπάνω μεταβλητή True, τότε το προεκπαιδευμένο δίκτυο δεν θα μπορούσε ποτέ να μάθει τα δεδομένα μας και δεν θα είχε νόημα η εκπαίδευση. Η αρχιτεκτονική που χρησιμοποιούμε είναι όπως φαίνεται στο παρακάτω σχήμα:



Με χρήση Adam optimizer και sparse cross entropy loss function παίρνουμε τα παρακάτω learning curves:





Λαμβάνουμε test score: 79%

Βλέπουμε πως με χρήση transfer learning, πετυχαίνουμε πολύ καλύτερη επίδοση, από ότι στο "from scratch", της τάξης του 21% πάνω. Ωστόσο, μπορούμε να πετύχουμε αρκετά καλύτερες επιδόσεις και στους δύο τύπους αρχιτεκτονικών και με αύτο το κομμάτι θα ασχοληθούμε από εδώ και στο εξής στην μελέτη μας.

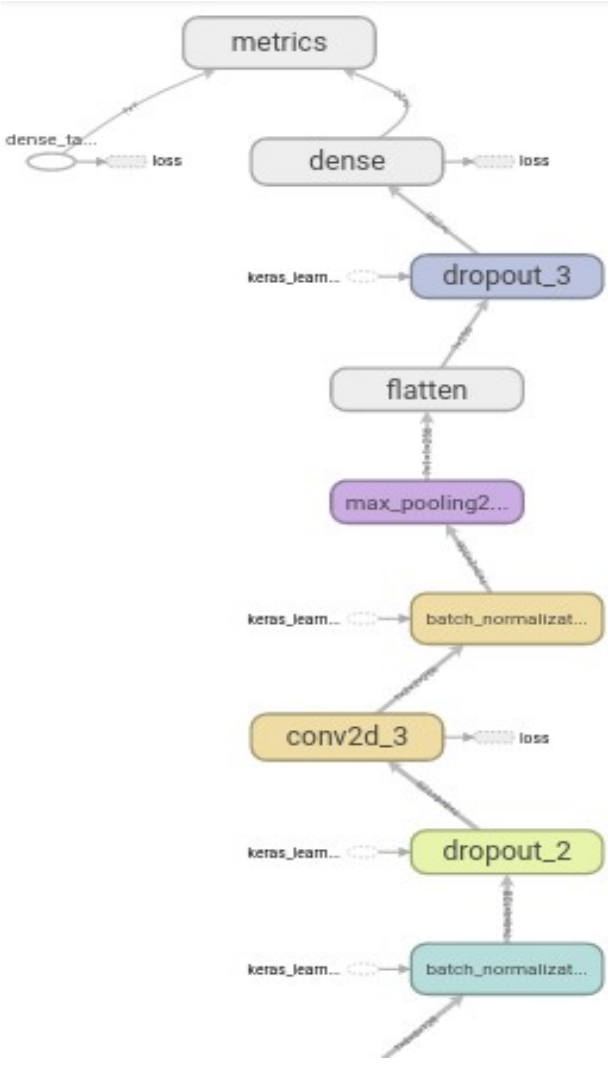
3. Βελτιστοποίηση αρχιτεκτονικών

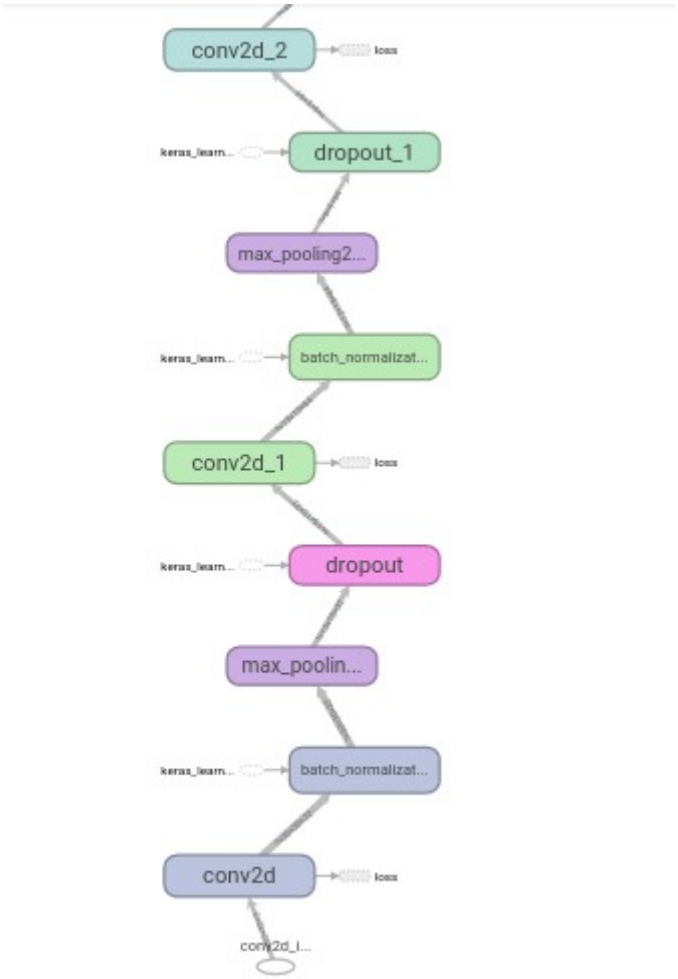
Καλούμαστε, λοιπόν, να βελτιώσουμε τα αποτελέσματα ταξινόμησης στο CIFAR-100 και να βγάλουμε στο τέλος τα αντίστοιχα συμπεράσματα. Έχοντας δοκιμάσει αρκετές διαφορετικές αρχιτεκτονικές, θα παρουσιάσουμε αυτές που έχουν το μεγαλύτερο ερευνητικό ενδιαφέρον και τις καλύτερες αποδόσεις.

3.1 From Scratch CNN

3.1.1 CNN4

Θα χρησιμοποιήσουμε ένα δίκτυο που θα αποτελείται από 4 Convolutional layers, που το καθένα περιέχει Conv2D, BatchNormalization, MaxPooling, Dropout, και 1 fully-connected output layer. Σε κάθε layer γίνεται χρήση ως weight decay η l1 norm με λ=0.001, ενώ ως optimizer του συστήματος παίρνουμε τον Adam (lr=0.0001) και σαν Loss function την sparse cross-entropy, συνδυασμένη με softmax στο output layer, όπως είδαμε και παραπάνω. Η παραπάνω αρχιτεκτονική αναπαριστάται παρακάτω μέσω TensorBoard:

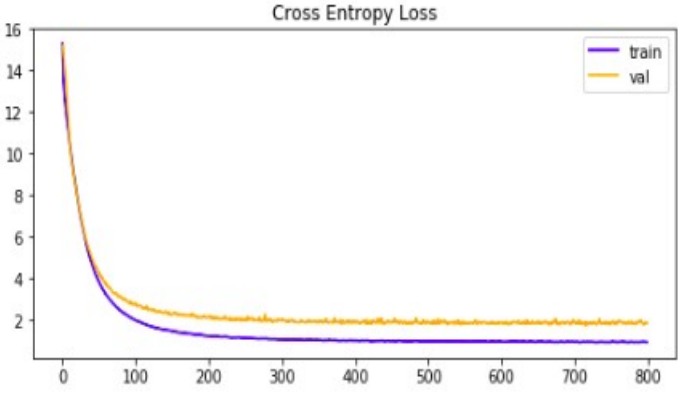


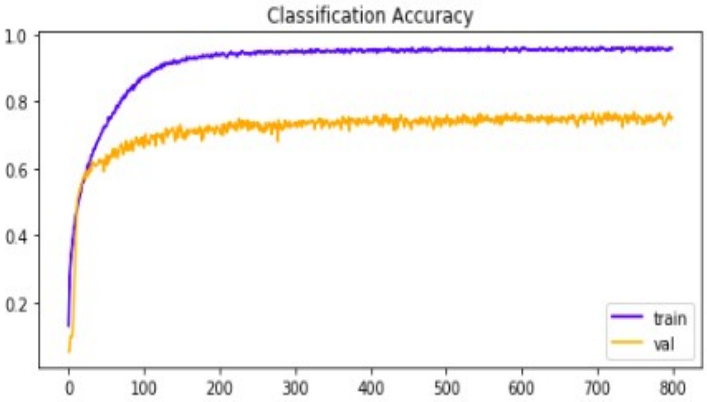


Το αρχικό μας "from scratch" μοντέλο, ήταν ήδη κάπως βελτιστοποιημένο σε σχέση με της εκφώνησης, δεδομένου ότι περιείχε 1 fully-connected layer (αντί για 2) και dropout πριν την έξοδο, με αποτέλεσμα να συρρίκνωνε αρκετά το overfitting.

Με σκοπό να αυξήσουμε την επίδοση, πειραματιστήκαμε με αρκετούς τρόπους. Αρχικά προσθέσαμε ενα Conv2D ακόμα, έχοντας μόνο το τελικό dropout, και το test score ήταν γύρω στο 67%. Στην συνέχεια προσθέσαμε ένα ακόμα fully-connected και παρατηρήσαμε overfitting. Βάλαμε παντού χαμηλά ποσοστά dropout και batch normalization και είδαμε πως για το lr=0.001 έκανε ταλαντώσεις το loss, και για αυτό το λόγο το μειώσαμε κατά μία τάξη μεγέθους. Το μοντέλο είδαμε πως συνεχίζει να κάνει overfitting, με αποτέλεσμα να βγάλουμε το fully-connected που προσθέσαμε. Τρέξαμε το μοντέλο και είδαμε test score ίσο περίπου με 69%. Επιπλέον, βάλαμε κι άλλο Conv2D, αλλά δεν είδαμε κάποια σημαντική διαφορά σε σχέση με το να έχουμε συνολικά τέσσερα, και το αφαιρέσαμε για λόγους μνήμης και χρονικής διάρκειας εκπαίδευσης. Θελήσαμε να βελτίωσουμε το μοντέλο αυτό με αλλαγές ως προς το learning rate, το validation checkpoint, τον optimizer, το dropout, το batch_size και το weight decay.

Εκπαιδεύοντας το μοντέλο αυτό, αναπαριστούμε τα παρακάτω learning curves:





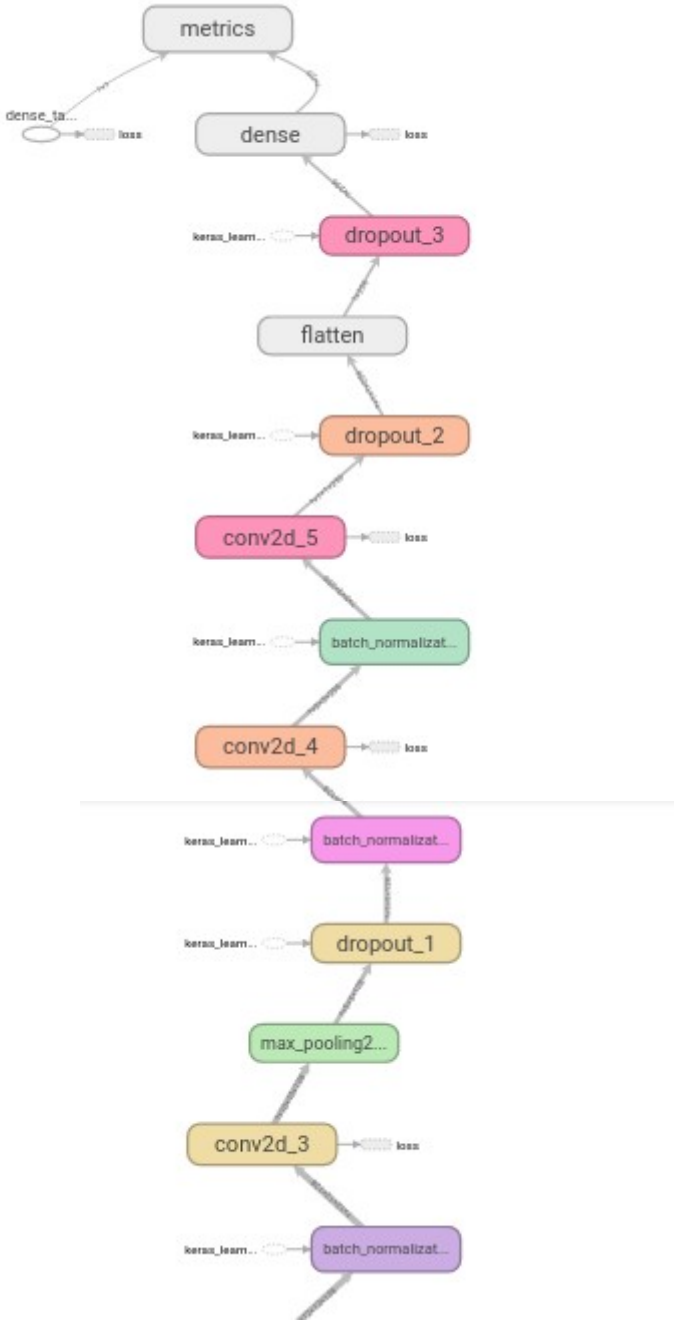
Λαμβάνουμε test score: 75%

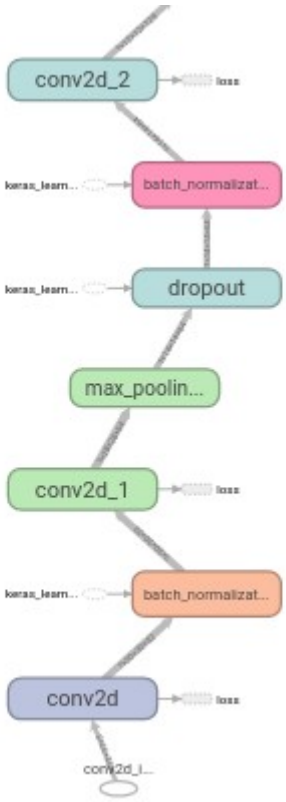
Αξίζει στο σημείο αυτό να περιγράψουμε την διαδικασία βελτιστοποιήσης που ακολουθήσαμε. Αρχικά, αυξήσαμε αρκετά το batch_size και είδαμε πως το μοντέλο μας αργούσε να φθάσει σε σύγκλιση, καθώς γίνονταν λιγότερα updates ανά εποχή, με αποτέλεσμα να το ορίσουμε 150. Αλλάξαμε τον optimizer σε SGD, και είδαμε πως με αυτό τον τρόπο δεν έφθανε σε σύγκλιση γρήγορα. Αυξήσαμε το learning rate και είδαμε πως το test score δεν βελτιώθηκε. Οπότε, χρησιμοποιήσαμε Adam. Βάζοντας weight decay και σχετικά μεγάλο dropout (0.2-0.35) και με learning rate ίσο με 0.0001, αφού για μεγαλύτερη τάξη μεγέθους το μοντέλο έκανε ταλαντώσεις τόσο στο train_loss όσο και στο val_loss, στοχεύσαμε το train loss να μειώνεται με όσο το δυνατόν παρόμοιο ρυθμό όπως το val loss ή να φθήνουν συνεχώς και τα δυο, με αποτελέσμα να αυξήσουμε τις εποχές σε 400 (6.5 λεπτά, μικρός χρόνος training) και να πετύχουμε σκορ πάνω απο 70%, και συγκεκριμένα 73%. Επειδή παρατηρήσαμε πως τα learning curves ήταν φθήνουσες συναρτήσεις και για το train loss και για το val loss, δηλαδή είχαμε underfitting, αυξήσαμε τις εποχές, γνωρίζοντας πως θα ανεβάσουμε και άλλο την επιδοση. Και πράγματι, τρέχοντας στις διπλάσιες επαναλήψεις (800, 12.5 λεπτά εκπαίδευσης), φτάσαμε το test score σε 74% και το validation score έως και 77.5%. Σημειώνουμε πως οι πολλές εποχές δεν παραμονέυουν τον κίνδυνο του overfitting, αφού με την χρήση του checkpoint κρατάμε το καλύτερο βάσει του validation loss. Ακόμα, παίρνοντας υπόψιν στο checkpoint (αντί για Early Stopping) την μεταβλητή val_accuracy αντί του val_loss, αφού είχαμε δει και το 77% στο validation (σκορ που δεν αντιστοιχούσε στο μικρότερο val_loss), καταλήγουμε σε test score ίσο με 75%. Αξίζει να σημειωθεί πως η παραπάνω εκπαίδευση δεν αποτελεί περίπτωση overfitting, αφού το val_loss και το train_loss "παγώνουν" στην ίδια εποχή, δηλαδή αρχίζουν και ταλαντώνονται αμφότερα, μετά την οποία φτάνουν τελικά σε κάποιο ελάχιστο. Τέλος, επισημαίνουμε πως καταφέραμε να φέρουμε το "from scratch" σε απόδοση πολύ κοντινή με την απόδοση του μη βελτιστοποιημένου transfer learning (79%).

3.1.2 CNN6

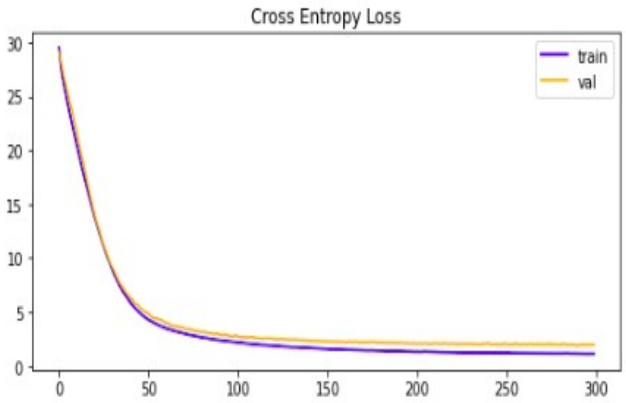
Ομολογουμένως, την προηγούμενη αρχιτεκτονική την φθάσαμε σε ένα αρκέτα καλό σημείο βελτιστοποίησης. Ωστόσο, θα προσπαθήσουμε να ανεβάσουμε την επίδοση μας με την χρήση ενός ακόμα πιο περίπλοκου μοντέλου, που περιέχει 6 Conv2D layers. Πιο αναλυτικά, θα χρησιμοποιήσουμε ένα δίκτυο που θα αποτελείται από 3 διπλά Convolutional layers (το καθένα περιέχει Conv2D, BatchNormalization, MaxPooling, Dropout) και 1 fully-connected output layer. Χρησιμοποιούμε ακόμα σε κάθε layer ως weight decay την l1 norm με λ=0.001, ενώ ως optimizer του συστήματος παίρνουμε τον Adam (lr=0.0001) και σαν Loss function την cross-entropy, συνδυασμένη με softmax στο output layer, όπως έγινε και στο CNN4 παραπάνω. Έτσι, ουσιαστικά, επεκτείνουμε το CNN4 με την προσθήκη διπλών convolutional layers, ενώ χρησιμοποιούμε ίδιες υπερπαραμέτρους, με εξαίρεση το πλήθος των εποχών που τώρα είναι 300.

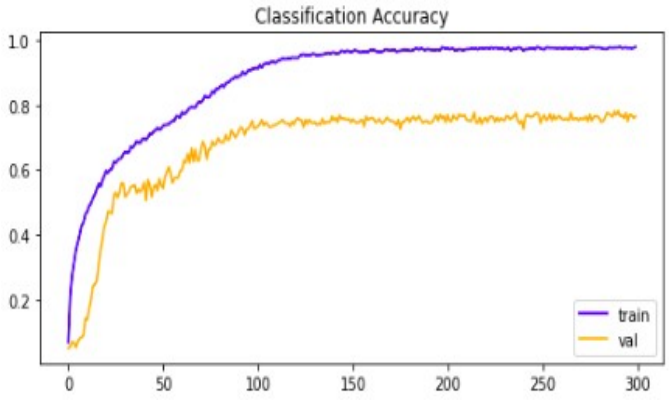
Η αρχιτεκτονική του μοντέλου φαίνεται με την βοήθεια του TensorBoard στο παρακάτω σχήμα:





Εκπαιδεύοντας το μοντέλο, τα learning curves που προέκυψαν ήταν τα παρακάτω:





Λαμβάνουμε test score: 74%

Παρατηρούμε πως το validation score έφτασε πάνω από 78% σε κάποιες εποχές. Αυτό σημαίνει πως περιμέναμε και το test score να είναι τόσο υψηλό. Ωστόσο, το test score βγήκε 74%, λόγω τυχαιότητας. Αν είχε επιλεχτεί κάποιο άλλο model στο training με val_accuracy παρόμοιο της μέγιστης ενδεχομένως να είχαμε κοντά στο 77% test score. Βλέπουμε πως και σε αυτήν την περίπτωση είμαστε σε ίδια επίπεδα με το VGG16 που είδαμε παραπάνω. Παρατηρούμε ακόμα πως ούτε εδώ έχουμε περίπτωση overfitting, και πως ο χρόνος εκπαίδευσης ήταν πάλι στα 12.5 λεπτά, γεγονός που συνέβαλε και ο διπλασιασμός του batch size.

3.2 Transfer Learning Optimization

Ταυτόχρονα με την αρχιτεκτονική, στη μεταφορά μάθησης εισάγουμε και τη γνώση που έχει αποκτήσει το μοντέλο, δηλαδή τις τιμές των βαρών του όπως έχουν προκύψει μετά από εκπαίδευση στο ImageNet. Οταν εισάγουμε ένα μοντέλο με μεταφορά μάθησης έχουμε τρεις επιλογές για την εκπαίδευση:

- 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), σημαία trainable=False.
- να συνεχίσουμε να εκπαιδεύουμε όλα τα επίπεδα του δικτύου, σημαία trainable=True.
- να εκπαιδευτεί μόνο ένα ποσοστό των επιπέδων, εβρισκόμενο προς την έξοδο του δικτύου. Οι σημαίες trainable εδώ θα πρέπει να οριστούν ανά επίπεδο.

Ωστόσο, στην μελέτη μας όταν παγώνουμε κάποια layers, το σύστημα δεν αποδίδει το ίδιο καλά με όταν είναι όλα trainable. Οπότε, θα δείξουμε μόνο την βέλτιστη περίπτωση.

Μέσω tf.keras.applications που παρέχει προεκπαιδευμένα μοντέλα από το Keras και συγκεκριμένα τα δίκτυα: DenseNet, Inception-ResNet V2, Inception V3, MobileNet v1, MobileNet v2, NASNet-A, ResNet, ResNet v2, VGG16, VGG19 και Xception V1.

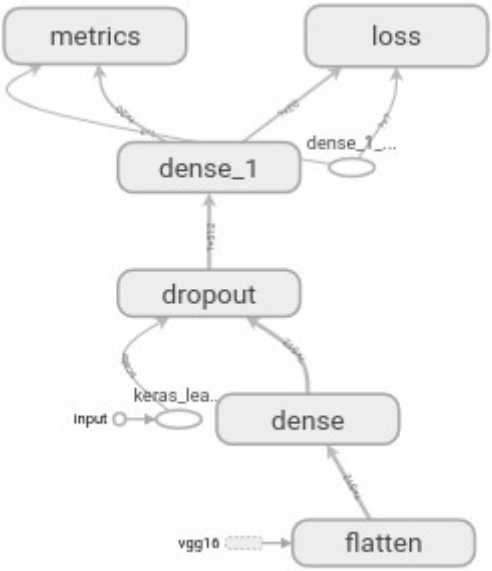
Στην μελέτη μας, μελετήσαμε το VGG16 και το VGG19. Επειδή είδαμε πως το VGG16 έδινε σχετικά καλύτερες επιδόσεις για τις ίδιες παραμέτρους, θα εστιάσουμε κυρίως σε αυτό, αλλά θα παρουσιάσουμε και το VGG19.

3.2.2 VGG16 - Data Augmentation

Συμφώνα την έρευνα, το VGG16 αποδίδει αρκετά καλύτερα στο CIFAR-10, όταν γίνεται χρήση του data augmentation. Μάλιστα, η βελτίωση του επίδοσης με κατάλληλο fine tuning έφτασε σε ποσοστό 8%. Πράγματι, και στην μελέτη μας, δουλεύοντας στο δεδομένο dataset του CIFAR-100, η επίδοση του συστήματος χωρίς data augmentation αλλά με dropout, batch normalization και fine tuning έφτανε κοντά στο αρχικό transfer learning δίκτυο, που δείξαμε παραπάνω, δηλαδή στο 80%, έχοντας, ωστόσο, λιγότερο overfitting, με το συνολικό loss από 1.23 να μειώθηκε στο 0.91. Παρακάτω, παρουσιάζουμε το VGG16 δίκτυο μας που παρουσιάζει το βέλτιστο test accuracy σε σχέση με τα υπόλοιπα που μελετήσαμε.

Η αρχιτεκτονική που θα χρησιμοποιήσουμε είναι το VGG16 δίκτυο, ακολουθούμενο από Flatten, ένα fully-connected(512) και ένα output fully-connected(20). Πριν το output layer χρησιμοποιούμε ένα dropout(0.15), προκειμένου να περιορίσουμε κάπως το αρχικό overfitting. Αξίζει να σημειώθεί πως χρησιμοποιούμε την elu έναντι της relu ως activation function του fully-connected(512), καθώς σημειώσαμε καλύτερη επίδοση με την χρήση αυτής.

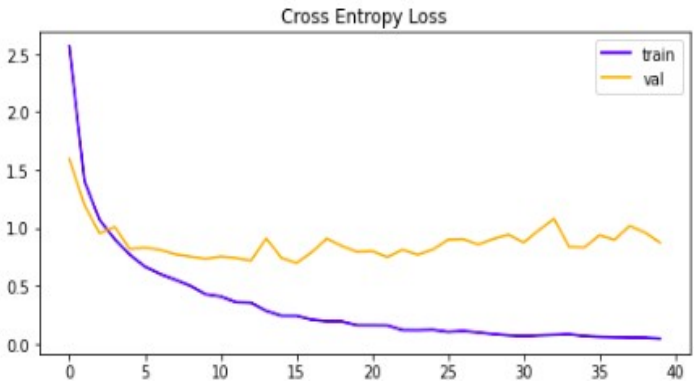
Η αρχιτεκτονική του μοντέλου μας αναπαριστάται στο σχήμα που ακολουθεί με την χρήση του TensorBoard:

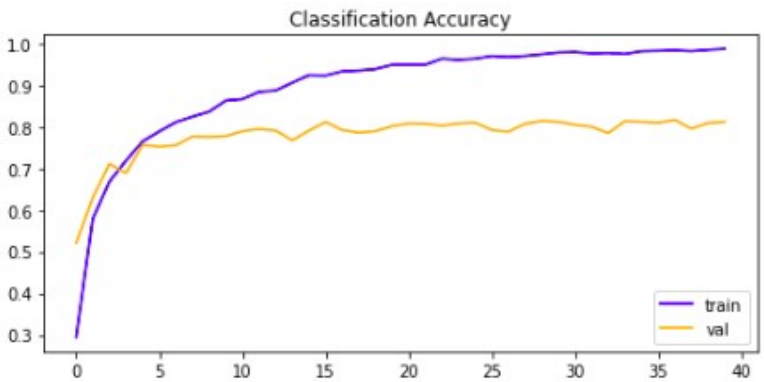


Το data augmentation που χρησιμοποιήσαμε, προκειμένου να αυξήσουμε τα δεδομένα μας και κάθε φορά το μοντέλο να συναντά νέες εικόνες. Αύτες σχηματίζονται σύμφωνα με τους παρακάτω μετασχηματισμούς: rotation, width shift, height shift, horizontal flip. Τα ποσοστά που ακολουθούν στην υλοποίηση μας, προέρχονται από την εξαγωγή αρκετών επιδόσεων διαφορετικών δοκιμών. Το γεγονός πως το μοντέλο βλέπει κάθε φορά νέα εικόνα οδήγει στην μείωση του overfitting, αφού το dataset συνεχώς αλλάζει.

rotation	0.17
width shift	0.11
height shift	0.11
horizontal flip	Yes

Παρακάτω, αναπαριστούμε τα learning curves:





Λαμβάνουμε test score: 83%.

Βλέπουμε πως το παραπάνω μοντέλο καταφέρνει να ανεβάσει αρκετά ψηλά την επίδοση στο test set, φτάνοντας το 83%, ενώ κατεβάζει βέλτιστα σε σχέση με τα παραπάνω μοντέλα και το loss (0.84). Ακομά καταφέρνουμε να μειώσουμε πάρα πολύ και τον χρόνο εκπαίδευσης στα 5 λεπτά, σημειώνοντας βέλτιστη επίδοση σε σχέση με τις υπόλοιπες προσπάθειες βελτιστοποίησης. Σημειώνουμε πως παρόμοια αποτελέσματα δίνει και η χρήση του Adamax optimizer αντί του Adam στο ίδιο πλήθος εποχών.

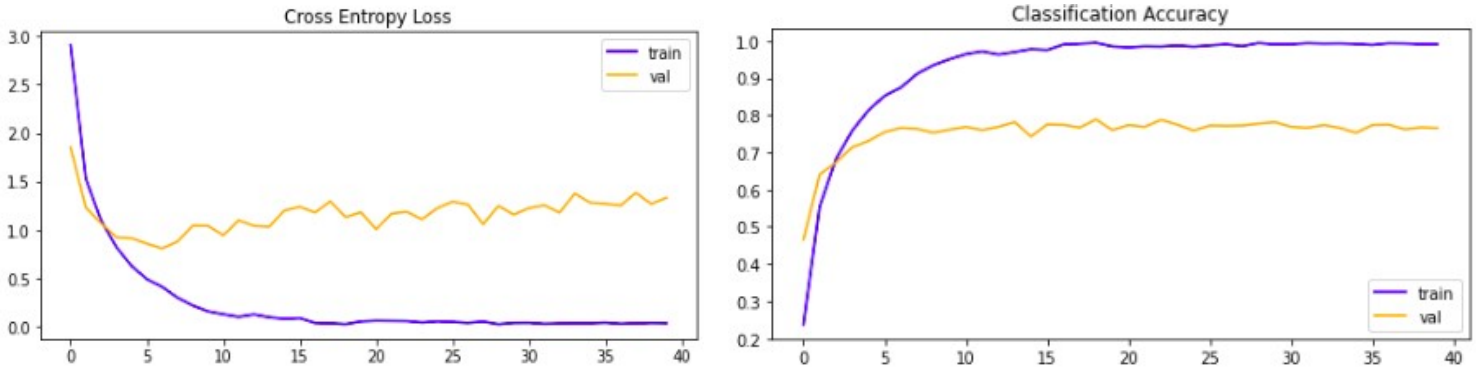
3.2.2 VGG16(s) - Prefetching, Σειριοποίηση, Data Augmentation

Αν και ουσιαστικά δεν χρειάζεται να βελτιστοποιήσουμε ως προς την μνήμη, αφού έχουμε πολύ μικρή κατανάλωση RAM με ένα τόσο σχετικά μικρό σύνολο δεδομένων, για λόγους μελέτης θα υλοποιήσουμε και ένα δίκτυο, ίδιας αρχιτεκτονικής με το VGG16, το οποίο θα χρησιμοποιεί τόσο prefetching, όπως και τα υπόλοιπα που είδαμε έως τώρα, όσο και σειριοποίηση των δεδομένων μαζί με data augmentation. Στόχος μας είναι να εξετάσουμε κατά πόσο αυτή η τεχνική τελικά οφελεί στον συνολικό χρόνο εκπαίδευσης και μνήμης.

Εκπαιδεύοντας το μοντέλο, βλέπουμε learning curves παρόμοια με του VGG16 που μελετήσαμε παραπάνω, ενώ το test score: 81%. Ο χρόνος εκπαίδευσης είναι πιο μικρός από του VGG16, οπώς μπορούμε να παρατηρήσουμε και στον πίνακα της επόμενης ενότητας. Επίσης, αύτη η τεχνική έχει ως αποτέλεσμα να μειώνει και την μνήμη που απαιτείται για την εκπαίδευση και το testing του μοντέλου.

3.2.1 VGG19

Έχοντας εκπαιδεύσει αρκετά μοντέλα με transfer learning του VGG-19, παρατηρήσαμε πως τα αποτελέσματά του ήταν σε γενικές γραμμές λίγο χειρότερα από του VGG-16. Στο σημείο αυτό, με χρήση ίδιων υπερπαραμέτρων με του VGG16 παραπάνω, παραθέτουμε τα learning curves στο σχήμα που ακολουθεί.



Το test score που προέκυψε ήταν 78%. Σημειώνουμε πως το μοντέλο δεν έχει υποστεί data augmentation. Παρολά αυτά, ακόμα και με data augmentation δεν ξεπερνάει το 83% που είδαμε παραπάνω. Ωστόσο, το μοντέλο αυτό είναι πιο περίπλοκο από το VGG16, έχοντας παραπάνω παραμέτρους, αλλά δίχως κάποιο ουσιαστικό αποτελέσμα στην μελέτη μας.

4. Συμπεράσματα

Στο σημείο αυτό, καλούμαστε να αναπαραστήσουμε τα αποτελέσματα που παρουσιάσαμε παραπάνω σε μία πιο γενική και κατανοητή μορφή. Για αυτό τον λόγο, παρουσιάζουμε με την μορφή πινάκων τα αποτελέσματα από τις αρχιτεκτονικές που μελετήσαμε.

Αρχιτεκτονική	Αρ. Παραμέτρων	Χρόνος Εκπ. (s)	Αρ. Εποχών	Test Accuracy
CNN3	61,460	48.96	60	**63%**
VGG16 (init)	14,724,948	66.56	25	**79%**
CNN4	395,476	736.42	800	**75%**
CNN6	1,133,652	725.34	300	**74%**
VGG16	14,987,604	331.84	40	**83%**
VGG19	20,338,340	218.94	40	**78%**
VGG16 (s)	14,987,604	226.40	40	**81%**

Παρατηρούμε πως τον ελάχιστο αριθμό παραμέτρων από τα βελτιστοποιημένα μοντέλα τον εμφανίζει το CNN4, που ωστόσο χρειάζεται τον μεγαλύτερο χρόνο εκπαίδευσης. Από την αλλή το μεγαλύτερο test score accuracy το εμφανίζει το VGG16 με **83%**, έχοντας σχετικά μικρό χρόνο εκτέλεσης, δηλαδή περίπου 100 seconds πιο πολύ από το VGG16(s) που είναι πιο βέλτιστο στο χρόνο. Αν και τα μη βελτιστοποιημένα μοντέλα έχουν αρκετά μικρότερο χρόνο εκπαίδευσης, αυτό είναι αρκετά υποκειμενικό, αφού έχουν πολύ λιγότερες εποχές συγκριτικά με τις βελτιστοποιημένες υλοποιήσεις τους. Ακόμη, ο αριθμός των εποχών ίσως σε κάποια μοντέλα να είναι υπέρ αρκετός, αλλά, παρόλα αυτά, με την χρήση του Checkpoint, καταφέρνουμε να καταλήγουμε κάθε φορά με το επιθυμητό εκπαιδευμένο μοντέλο.

5. What's next ?

Αφού έχουμε ολοκληρώσει την μελέτη μας στο πλαίσιο της τρίτης εργαστηριακής άσκησης, θα ήταν άξιο μελέτης να μελετήσουμε την προσαρμοστικότητα και τη συνέπεια των αρχιτεκτονικών μας σε ολόκληρο το CIFAR-100 dataset, στο οποίο έχει παρατηρηθεί μέχρι στιγμής επίδοση μέχρι και 92%.

6. References

[1] https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
[2] https://datascience.stackexchange.com/questions/29719/how-to-set-batch-size-steps-per-epoch-and-validation-steps
[3] https://stackoverflow.com/questions/49922252/choosing-number-of-steps-per-epoch
[4] https://stats.stackexchange.com/questions/140811/how-large-should-the-batch-size-be-for-stochastic-gradient-descent
[5] https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/
[6] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
[7] https://scholar.smu.edu/cgi/viewcontent.cgi?article=1091&context=datasciencereview
[8] https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/
[9] https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/
[10] http://marubon-ds.blogspot.com/2017/09/vgg16-fine-tuning-model.html
[11] https://benchmarks.ai/cifar-100
[12]https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/18_TFRecords_Dataset_API.ipynb