Deep Learning – Project Report

Alexandra Lakka

lakka20@aueb.gr

Code[1]: https://colab.research.google.com/drive/1UkW-1KT0ZLn0yQRWYJSrYblJx3aygxaO?usp=sharing

Pickles[2]: https://drive.google.com/drive/folders/13D3rdevkpyZs1H5GdjksYY1GvMxWYFpp?usp=sharing

Logs[3]: https://drive.google.com/drive/folders/1x1P3cYljb3tgXZd1JC7ckiVZa_1xjN-V?usp=sharing

1. Εισαγωγή

Από τις επιλογές που προσφέρθηκαν, το τελικό πρότζεκτ είναι πάνω στο MURA dataset, για αναγνώριση μια ιατρικής εικόνας ως φυσιολογικής(normal) ή μη φυσιολογικής(abnormal).

Το dataset που δόθηκε περιλάμβανε έτοιμα χωρισμένα τα δεδομένα σε train και validation. Επίσης, ένας περαιτέρω διαχωρισμός είναι η οργάνωση των εικόνων με βάση τη μυϊκή ομάδα στην οποία ανήκουν(XR_ELBOW, XR_FINGER, XR_FOREARM, XR_HAND, XR_HUMERUS, XR_SHOULDER, XR_WRIST). Μέσα σε κάθε μυϊκή ομάδα, οι εικόνες έχουν οργανωθεί ως προς τον ασθενή και με βάση την κατηγοριοποίηση ως normal ή abnormal(patient case study). Ακόμα, περιλαμβάνονταν 4 csv αρχεία, ένα με τα μονοπάτια για την κάθε εικόνα και ένα για τα μονοπάτια για κάθε case study.

Ο κώδικας γράφτηκε και έγινε tested στο Google Colab. Ο κώδικας μπορεί να βρεθεί στο πιο πάνω link. Συγκεκριμένα, στο [1] υπάρχει ο κώδικας και, σαν έξτρα, υπάρχει το link για τα pickle αρχεία[2], καθώς και ο φάκελος με όλα τα logs από τα αποτελέσματα του training, ώστε να γίνουν view στο TensorBoard[3].

2. Δεδομένα

Αρχικά, χρησιμοποιήθηκε το train_images_path.csv και το valid_images_path.csv, έτσι ώστε να οργανωθούν, σε πρώτη φάση, τα δεδομένα σε 2 dataframes. Και τα 2 dataframes περιλαμβάνουν τις εξής στήλες: image_path, patient_id, case και label. Κάθε εικόνα έχει γίνει labeled με βάση την κατηγορία στην οποία ανήκει, με 0 ή 1[έχουν δημιουργηθεί και dataframes, όπου το label είναι normal ή abnormal, για τις δοκιμές με την flow_from_dataframe() της ImageDataGenerator, εν τέλη όμως δεν χρησιμοποιήθηκε, καθώς η 1η εποχή, για όλες τις κατηγορίες έπαιρνε πάρα πολύ ώρα(φόρτωνε, πρώτα, όλες τις εικόνες στη RAM)].

Έπειτα, το dataframe χωρίστηκε με βάση τις μυϊκές ομάδες, ώστε να είναι πιο εύκολη, μετά, η επεξεργασία και η αποθήκευση. Να σημειωθεί ότι επιδιώκεται ο διαχωρισμός αυτός για να μπορεί, έπειτα, να γίνει εκπαίδευση και πρόβλεψη για κάθε κατηγορία. Κρατάμε και αποθηκεύουμε τα dataframes σε pickle αρχεία, για πρόσβαση αργότερα, τόσο για train όσο και για validation.

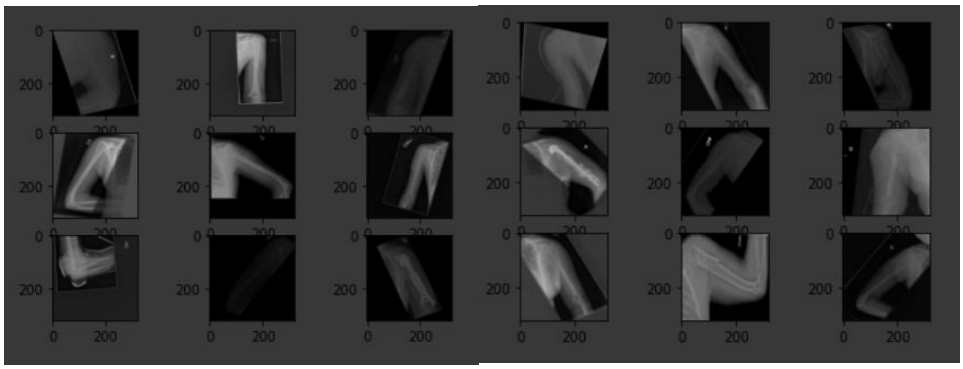
Στη συνέχεια, από κάθε dataframe, φτιάχτηκαν τα X_train, Y_train, X_val, Y_val. Συγκεκριμένα, οι εικόνες έγιναν resized, με μέγεθος 320x320, όπως και στο αρχικό paper για το dataset, και έγιναν από RGB(3 channels) GRAYSCALE(1 channel). Οι GRAYSCALE εικόνες έχουν μικρότερο μέγεθος και εξυπηρετούν πιο εύκολα με τους φυσικούς περιορισμούς που υπάρχουν(RAM). Αυτά και πάλι αποθηκεύονται σε pickle αρχεία, ώστε να μην γίνονται οι μετασχηματισμοί από την αρχή κάθε φορά, καθώς αυτό είναι πολύ χρονοβόρο. Πρέπει εδώ να σημειωθεί ότι η παραπάνω διαδικασία έχει πραγματοποιηθεί για όλες τις μυϊκές ομάδες, εκτός από την XR_WRIST. Λόγω του μεγάλου μεγέθους του(πάνω από 9000 εικόνες), η RAM στο Colab έσκαγε εκτελούταν το loop των μετασχηματισμών.

3. Preprocessing

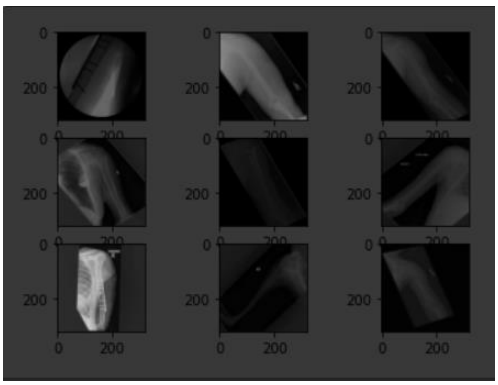
Σε επόμενη φάση, αυτό που έγινε ήταν η προετοιμασία των δεδομένων. Αυτό που χρησιμοποιήθηκε τελικά ήταν το normalization των pixel των εικόνων, διαιρώντας με το 255, ώστε οι τιμές να συγκεντρωθούν γύρω

από το διάστημα [0, 1]. Ωστόσο, έγιναν και άλλες δοκιμές, ώστε να αποφασιστεί αν υπάρχει καλύτερη preprocessing διαδικασία, που θα βοηθήσει με το overfitting πρόβλημα που παρουσιάστηκε στις δοκιμές για δημιουργία και εκπαίδευση του μοντέλου.

Η λογική ήταν, εφόσον το μοντέλο μαθαίνει "απέξω" τα δεδομένα και δεν μαθαίνει ουσιαστικά patterns, πρέπει να γίνουν κάποιοι μετασχηματισμοί, οι οποίοι θα το μπερδέψουν. Ένας από αυτούς ήταν να γίνεται, τυχαία, flip των εικόνων, οριζόντια. Σκοπός ήταν να διατηρηθεί η φορά των εικόνων, για αυτό δεν έγινε vertical flip. Υπήρξε και η σκέψη να γίνει rotation κάποιων εικόνων κατά πχ 30 μοίρες, ωστόσο κάποιες εικόνες είναι ήδη rotated, επομένως θεωρήθηκε ότι δεν χρειάστηκε κάτι παραπάνω. Κάτι, ακόμη, που δοκιμάστηκε ήταν να μειωθεί η φωτεινότητα της εικόνας κατά 20%. Σκοπός ήταν να χαθεί ένα μέρος της πληροφορίας, ώστε να αναγκαστεί το μοντέλο να μάθει. Είτε δοκιμάζοντας ξεχωριστά αυτές τις 2 τεχνικές είτε μαζί, δεν έβγαλαν καλύτερα αποτελέσματα από το αρχικό normalization, επομένως προτιμήθηκε η πιο απλή μέθοδος. Επίσης, και όταν οι τεχνικές συνδυάστηκαν δεν έδειξαν κάποιο καλύτερο αποτέλεσμα.



Εφαρμογή και των 2 μετασχηματισμών *Horizontal Flip*



Brightness minus 20%

Άλλη preprocessing τεχνική που δοκιμάστηκε ήταν να γίνει normalization, βγάζοντας το μέσο όρο των pixels να είναι κοντά στο 0 και διαιρώντας με την τυπική απόκλιση. Και πάλι, δεν έδειξε σημαντική βελτίωση έναντι της πρώτης περίπτωσης.

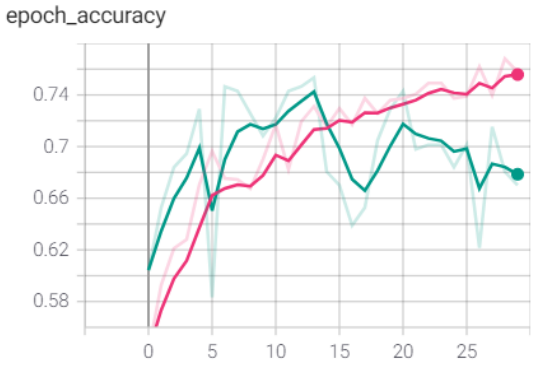
4. Models

Τώρα, στο στάδιο επιλογής του μοντέλου, έγιναν δοκιμές σε 2 pretrained μοντέλα και σε ένα μοντέλο φτιαγμένο από την αρχή.

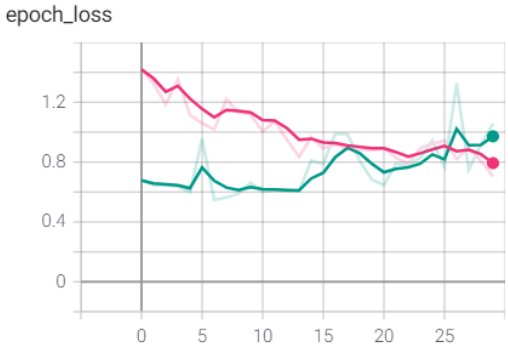
4.1. Pretrained

Όσον αφορά τα pretrained μοντέλα, βασικός λόγος που δοκιμάστηκαν ήταν για να χρησιμοποιηθούν τα weights του ImageNet, με σκοπό να βελτιώσουν την εκπαίδευση. Δοκιμάστηκαν τα DenseNet169 και VGG16. Το πρώτο επιλέχθηκε, ώστε να γίνει μια απόπειρα αναπαραγωγής των δοκιμών από το αρχικό paper. Φορτώθηκε το μοντέλο, με τα βάρη από το ImageNet και "πάγωσε" εκεί η εκπαίδευση. Αυτό μετά χρησιμοποιήθηκε σαν βάση για ένα Sequential μοντέλο, στο οποίο προστέθηκε ένα Flatten layer, για να διαμορφώσει την είσοδο του DenseNet, ένα Dropout, ώστε να βοηθήσει στη βελτίωση των αποτελεσμάτων

και, τέλος ένα Dense layer μεγέθους 1(έχουμε binary classification, μόνο μια έξοδος θα υπάρχει). Από τις εικόνες φαίνεται ότι το loss σταθεροποιείται σε αρκετή υψηλή τιμή και το validation accuracy δεν φαίνεται να βελτιώνεται μετά τις 20 εποχές(**train για 30 εποχές, train -> ροζ, validation -> πράσινο**).

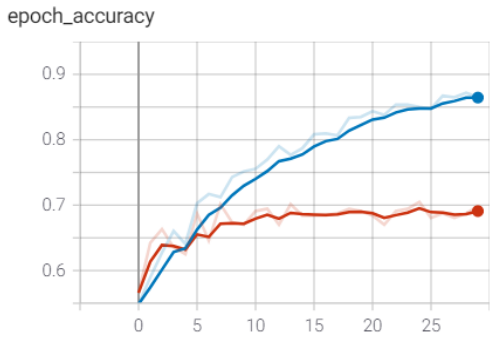


DenseNet169 – accuracy

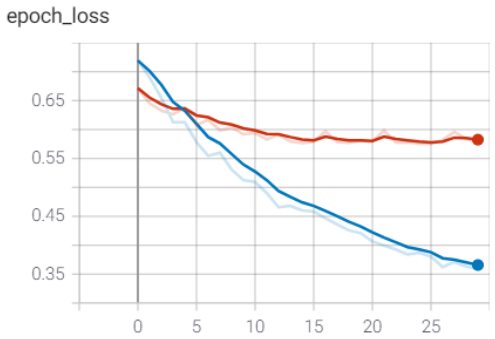


DenseNet169 – loss

Ίδια βήματα ακολουθήθηκαν και για το VGG16. Ο λόγος που επιλέχθηκε ήταν για να χρησιμοποιηθεί ένα μικρότερο μοντέλο από το προηγούμενο, μήπως η πολυπλοκότητα του προηγούμενου λειτουργούσε αρνητικά. Εδώ, η ψαλίδα μεταξύ train και validation είναι μεγαλύτερη και το validation loss πάλι είναι υψηλό. Φαίνεται να δημιουργείται overfitting από τις 15 εποχές και μετά(accuracy)(**train για 30 εποχές, train -> μπλε, validation -> κόκκινο**).



VGG16 – accuracy



VGG16 – loss

Τα αποτελέσματα των εικόνων προκύπτουν από το **XR_HUMERUS**. Επειδή τα pretrained μοντέλα παίρνουν σαν είσοδο εικόνες με 3 channels, έγινε προσπάθεια να δημιουργηθούν τα X και Y, κάνοντας μόνο resize και διατηρώντας τα αρχικά κανάλια. Ωστόσο, για τις περισσότερες κατηγορίες, δεν ολοκληρωνόταν η διαδικασία, καθώς γέμιζε η RAM του Colab. Επομένως, για πρακτικούς κυρίως λόγους, εφόσον δεν μπορούσε να εξεταστεί η αποτελεσματικότητα των μοντέλων για τις υπόλοιπες κατηγορίες, απορρίφθηκε η χρήση ενός pretrained μοντέλο.

4.2. Custom

Στην περίπτωση του custom μοντέλου, έγιναν πολλές δοκιμές όσον αφορά τα επίπεδα που θα χρησιμοποιηθούν, το μέγεθος κ.α. Βασικός στόχος ήταν να μειωθεί το overfitting, όσο το δυνατό περισσότερο, που παρατηρούνταν στην εκπαίδευση. Παρακάτω θα αναλυθεί η τελική μορφή του μοντέλου.

Κάτι σημαντικό που πρέπει να τονιστεί αφορά τη χρήση της activation function στο output layer. Αρχικά, είχε γίνει χρήση της softmax. Ωστόσο, ανεξαρτήτως από το μέγεθος του μοντέλου και τις διάφορες τεχνικές που μπορεί να χρησιμοποιούνταν(πχ Dropout), τα train και validation accuracies παρέμεναν σταθερά. Για αυτό, τελικά, επιλέχθηκε η sigmoid, η οποία εξάλειψε αυτό το πρόβλημα.

Χρησιμοποιήθηκαν ένα Input layer, για την είσοδο της εικόνας, 6 2D Convolutional layers και ένα Flatten layer, για να μετατραπεί η πολυδιάστατη έξοδος του τελευταίου Convolutional σε μια είσοδο 1-D, για το τελικό Dense layer, το οποίο πραγματοποιεί και το classification της εικόνας(normal, abnormal).

Πιο ειδικά, τώρα, χρησιμοποιήθηκαν σε κάθε layer τα παρακάτω:

Conv2D: Το πρώτο τέτοιο επίπεδο ξεκινάει με μικρό αριθμό filters(16), με μεγάλο μέγεθος(7x7). Όσο αυξάνονται τα επίπεδα, ο αριθμός των φίλτρων αυξάνεται(φτάνει μέχρι 128), ενώ το kernel_size μειώνεται(φτάνει το 3x3). Ο αριθμός των φίλτρων αυξάνεται, ώστε να κάνουμε extract όσα περισσότερα features γίνεται και, τελικά, να πάρουμε περισσότερη πληροφορία που θα βοηθήσει στην αναγνώριση abstract αντικειμένων[πχ το πώς είναι ένα κανονικό χέρι και πώς ένα με κάποιο πρόβλημα(μπορεί να υπάρχει κάποιο σπασμένο κόκαλο)], ενώ το μέγεθος μειώνεται, καθώς η έξοδος κάθε επιπέδου θα είναι μικρότερη από την είσοδο του.

Activation function: Χρησιμοποιείται η ReLU(η πιο κλασσική περίπτωση).

Batch normalization: Ένας τρόπος που χρησιμοποιήθηκε για να μειωθεί το overfitting ήταν η χρήση του batch normalization, σε κάθε layer. Αυτό που κάνει είναι να κάνει normalize το output του προηγούμενου επιπέδου(υπολογίζει mean και standard deviation) και μπορεί να βοηθήσει τις παραγώγους να έχουν πιο σταθερή συμπεριφορά.

MaxPooling: έχει μέγεθος 2x2 και είναι η πιο κλασσική περίπτωση pooling, για image classification, καθώς διαλέγει τα πιο έντονα features στο patch που ελέγχει κάθε φορά.

Dropout: Το dropout χρησιμοποιήθηκε μόνο κάθε δεύτερο layer, καθώς πολλά dropouts χειροτερεύουν το training. Παρόλο που με το batch normalization δεν χρειάζεται το dropout, η χρήση του έδινε λίγο καλύτερα αποτελέσματα, από ότι χωρίς.

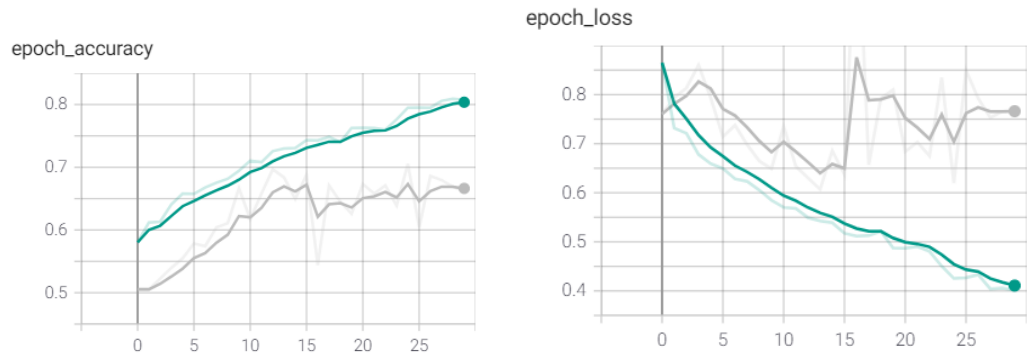
Μετά το τελικό hidden layer, γίνεται ένα **Flatten** της εξόδου και, για να βοηθήσει με το πρόβλημα του overfitting, προστέθηκε λίγος Gaussian θόρυβος στην είσοδο του Dense layer, ώστε να μπερδέψει το μοντέλο.

Μια ακόμη τεχνική που χρησιμοποιήθηκε για να μειωθεί το overfitting ήταν η αρχικοποίηση των βαρών. Εφόσον, για αυτό το μοντέλο, δεν μπορούν να χρησιμοποιηθούν τα βάρη του ImageNet, πρώτα έγινε προσπάθεια αρχικοποίησης με τα Glorot Uniform και Normal. Δεν έκαναν κάποια ιδιαίτερη διαφορά, επομένως ένας άλλος τρόπος ήταν να ελέγχονται οι τιμές των βαρών σε κάθε layer. Οι τιμές των βαρών ανήκουν στο διάστημα [0, 1]. Για μεγαλύτερο(πχ [0, 3]) ή συμμετρικό διάστημα(πχ[-1, 1]), τα αποτελέσματα χειροτερεύουν συγκριτικά με το επιλεγμένο διάστημα.

Μερικά παραδείγματα από το training φαίνονται στις παρακάτω εικόνες:

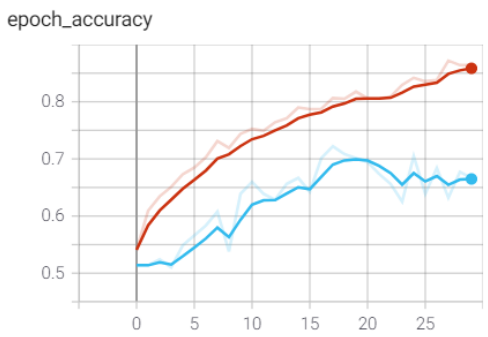
XR_ELBOW:

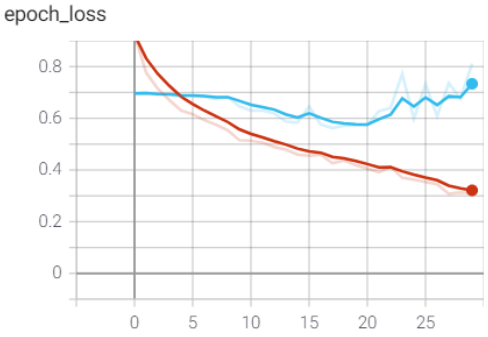
- **train για 30 εποχές, train -> πράσινο, validation -> γκρι**



XR_HUMERUS:

- **train για 30 εποχές, train -> κόκκινο, validation -> μπλε**



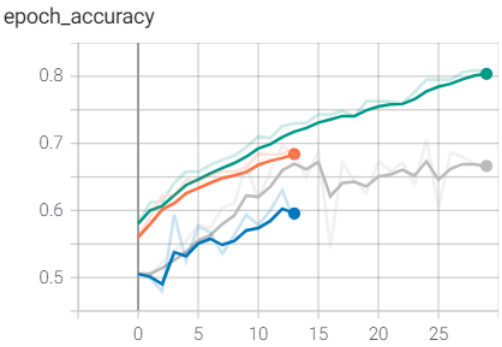


Για να βοηθήσει στο θέμα του diverge που οφείλεται στο overfitting ή όταν το μοντέλο δεν μπορεί να αυξήσει ή να μειώσει το accuracy ή το loss, αντίστοιχα, γίνεται χρήση του Early Stopping, για το validation loss, με patience 10.

Για παράδειγμα, φαίνεται παρακάτω ένα παράδειγμα με Early Stopping:

XR_ELBOW με Early Stopping:

- **train -> πορτοκαλί, validation -> μπλε**



5. Prediction

Αφού γίνει η εκπαίδευση του μοντέλου, μετά μπορεί να γίνει η πρόβλεψη αν ένα case είναι normal ή abnormal. Ο τρόπος με τον οποίο γίνεται είναι ελέγχοντας, ξεχωριστά την κάθε εικόνα σε ένα φάκελο, σε ποια κατηγορία ανήκει. Το threshold για να θεωρηθεί abnormal έχει τεθεί να είναι πάνω από 0.5. Ίσο ή κάτω από αυτή την τιμή θεωρείται normal. Κρατώντας 2 counters, ελέγχουμε ποιος είναι μεγαλύτερος και ανάλογα με το αποτέλεσμα χαρακτηρίζεται και το case. Σε περίπτωση που αυτοί είναι ίσοι, θεωρείται ότι η υπόθεση πρέπει να επανεξεταστεί.