ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ
ΠΜΣ «ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗ»

Εργασία στο μάθημα:

<<Τεχνητή Νοημοσύνη και Εφαρμογές στο Διαδίκτυο των Πραγμάτων.>>

Μορφιαδάκης Εμμανουήλ 21110
Ναβροζίδης Κυριάκος 21113

Περιεχόμενα

1. Εισαγωγή

1.1 Τίτλος εργασίας

Αναγνώριση πνευμονίας από ακτινογραφίες θώρακος (επιλογή ανάμεσα σε 3 κατηγορίες Normal, Bacterial Pneumonia, Viral Pneumonia).

1.2 Περιγραφή προβλήματος

Σήμερα ο κόσμος μαστίζεται από διάφορα άγχη,όπως εύρεση εργασίας,επίδοση στη δουλειά του,να φτιάξει το σπιτικό του, εξεταστικές κλπ.Υπάρχουν όμως και κάποιοι άνθρωποι που έχουν κάποιους πόνους στο στήθος και φοβούνται μην έχουν προβλήματα με τον πνεύμονα συν τα παραπάνω άγχη.Ο ακτινολόγος μέχρι να κάνει την διάγνωση θέλει κάποιο χρόνο και μπορεί ο ασθενής να αγχωθεί πολύ με την αναμονή.

Έτσι γίνεται προσπάθεια να εντοπιστεί μέσω ακτινογραφίας θώρακος, μια τεχνική που είναι ευρέως διαδεδομένη σε πολλές άλλες ασθένειες και χρησιμοποιείται πλέον παντού.
Οπότε το ενδιαφέρον και η σημαντικότητα έγκειται στο γεγονός ότι εάν ένα νευρωνικό δίκτυο καταφέρει να διακρίνει αποτελεσματικά την μορφή της πνευμονίας με μεγάλη επιτυχία.

1.3 Παραδείγματα εισόδου

Ως είσοδο δίνεται οι φωτογραφίες της ακτινογραφίας.

Ως έξοδο το σύστημα ταυτοποιεί αν κάποιος έχει πνευμονία 0 αν έχει viral ,1 αν έχει Bacterial ,και 2 αν δεν έχει πνευμονία.

1.4 Αξιολόγηση

Το σκεπτικό είναι να εξετάσουμε το confusion matrix. Τα δεδομένα που θα μας δώσουν το confusion matrix θα είναι ένα 10% του dataset, το οποίο δεν θα έχει δει καθόλου το μοντέλο μας. Για το συγκεκριμένο λοιπόν πρόβλημα αποφασίσαμε να χρησιμοποιήσουμε τη μετρική του accuracy.

1.5 Dataset

Τα δεδομένα και η αναφορά του προβλήματος είναι από το Kaggle Chest X-Ray Images (Pneumonia) στο παρακάτω Link.
https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Το dataset έχει 5856 εικόνες μεγέθους μεγαλύτερο από 900*900 χωρισμένες σε 2 κατηγορίες (Normal , Pneumonia).
Εξερευνώντας τα δεδομένα μας είδαμε πως στον φάκελο Pneumonia υπάρχουν 2 ισόποσες υποκατηγορίες οι οποίες ήταν η bacterial και η viral.

Έτσι αποφασίσαμε να χωρίσουμε το dataset σε (Normal (27%), Viral(25.5%), Bacterial(47.5%).

Η εργασία έχει υλοποιηθεί σε περιβάλλον Anaconda σε jupyter notebook.Επίσης χρησιμοποιούμε python, TensorFlow, Keras, scikitlearn και άλλων βιβλιοθηκών που χρησιμεύουν τόσο στην επεξεργασία όσο και στην οπτικοποίηση.

1.6 Μεθοδολογία

Αρχικά πειραματιστήκαμε με τις εικόνες έτσι όπως ήταν το αρχικό τους μέγεθος.Ομως η επεξεργασία καθυστερούσε πάρα πολύ.Αλλάξαμε το μέγεθος σε 400x400 και η διαδικασία ήταν εξίσου χρονοβόρα.Έτσι λοιπόν ξανά αλλάξαμε το μέγεθος και ασχοληθήκαμε στα πειράματα μας με εικόνες 224X224 , 200X200, 100X100.
Επίσης σε κάθε βήμα τα data τροποποιήθηκαν να παίρνουν τιμές από 0 μέχρι 1 σε float

Παρακάτω θα σας παρουσιάσουμε κάποια από τα σενάρια που τρέξαμε καθώς και τα αποτελέσματα τους.

1η προσπάθεια:

- 1 Conv2d Layer:10 νευρώνων
- MaxPooling2d,Conv2d Layer 10 νευρώνων
- AveragePooling,Conv2d Layer 10 νευρώνων
- 5 artificial_layers με 30 νευρωνες+Softmax
- Τα βάρη τα αρχικοποιούμε στο εύρος [-1/sqrt(training_cases),1/sqrt(training_cases)]

Βάλαμε batch των 200 εικόνων.Εποχές όσες του μέγεθος του training δια τα batches επειδή νομίζαμε ότι σε ένα epoch αντιστοιχούσε ένα batch αλλά ένα epoch έχει όλα τα batches.
Το test accuracy βγήκε **75**%.

Μετά βάλαμε 10 artificial νευρώνες+Softmax με την λογική ότι έχουμε λίγα layers.
Το test accuracy βγήκε 45%.Τα πράγματα έγιναν χειρότερα.
Οπότε είπαμε να βάλουμε 3 artificial νευρώνες+Softmax μπας και ήθελε λιγότερους νευρώνες.
Το test accuracy βγήκε **78**%.Καλύτερα πήγαμε.

Κρατώντας το μοντέλο σκεφτήκαμε να πειραματιστούμε με τον αριθμό των νευρώνων.
Οι τιμές που δοκιμάσαμε ήταν[30,35,20,40,60].

Με **30** νευρώνες ήταν το καλύτερο.
Επειτα πειραματιστήκαμε με το learning rate με τις τιμές [0.01,0.001,0.0001]

Αυξήσαμε το learning rate στο 0.01 μήπως βρει καμιά καλύτερη λύση αλλά απ ότι φαίνεται δεν βρέθηκε λόγω παράλειψης καλύτερων τοπικών ελαχίστων στο Error Function.
Αφου δεν λειτούργησε αυτό το μειώσαμε στο 0.0001 μπας και κρύβεται ένα καλύτερο τοπικό ελάχιστο.Αλλά ούτε με χαμηλότερη ταχύτητα πήγαν καλύτερα τα πράγματα.Όπως διαπιστώσαμε με 0.001 είχαμε καλύτερα αποτελέσματα.

Μετά αποφασίσαμε να βάλουμε και validation data.Οπότε χρησιμοποιήσαμε 60% για training 20% για validation και 20% για test.

2η προσπάθεια:

- 1 Conv2d Layer:10 νευρώνων,
- MaxPooling2d,Conv2d Layer 10 νευρόνων,
- AveragePooling,
- Conv2d Layer 10 νευρόνων,
- 5 artificial_layers με 30 νευρωνες+Softmax
- learning rate=0.001,batches 200
- 46 epochs

training_score	test_score
82.65%	76.6%

Μετά βάλαμε 93 epochs

training_score	test_score
88%	76.9%

Μετά βάλαμε 23 epochs

training_score	test_score
73,5%	70.7%

Αφου τα πράγματα δεν πήγαν καλύτερα μειώσαμε το batch size στα 50 δειγματα με 46 epochs

training_score	test_score
85.67%	77.8%

Μετά βάλαμε 30 epochs

training_score	test_score
85.32%	75.9%

Με βάση τα παραπάνω το καλύτερο ήταν 46 εποχές και batches μεγέθους 50

Επειτα βάζουμε penalty L2=0.0001 και κρατήσαμε ενα Conv2dLayer και MaxPooling

training_score	test_score
99.44%	76.9%

Οπως φαίνεται εγινε overfitting

Αφού ο αλγόριθμος μάθαινε πολύ καλά το training set αποφασίσαμε να σταματήσουμε το overfitting.Οπότε δοκιμάσαμε cross validation με 4 διαχωρισμούς και batch 150 δείγματα και αφαιρέσαμε το L2 και πετύχαμε test_score **77.5**% και διαπιστώσαμε οτι ήταν λίγο καλύτερο ήταν αλλά πήρε πάρα πολύ ώρα.

Μετά καταλάβαμε πως λειτουργεί το epochs στο tensorflow και έτσι αρχίσαμε να πειραματιζόμαστε με τα epochs και το batch_len.

Αρχικά βάλαμε 30 epochs και batch_len:200

training_score	test_score
88.71%	76.6%

μετά μειώσαμε τα epochs στα 15 μήπως είναι πολλά

training_score	test_score
80%	76.45%

Αφού δεν γινόταν τίποτα βάλαμε 60 epochs αλλα batch_len 200

training_score	test_score
100%	75.6%

Το καλύτερο τελικά ήταν 30 epochs.

Μετά αποφασίσαμε να αυξήσουμε τα convolutional neurons στα 40 και επιστρέψαμε στα 2 conv2D Layer.Μετά από κάθε layer βάζαμε MaxPooling2D.

Ξαναβάλαμε 60 epochs και batch_len 200

training_score	test_score
100%	73.55%

Επειτα σκεφτήκαμε πως το νευρωνικό δικτυο δέχεται πολλές παραμέτρους οπότε βάλαμε 4 convLayers με maxPooling και τα artificialLayers τα μειώσαμε στα 3+Softmax και βάλαμε 46 epoch και 50 batch_len.

training_score	test_score

91.65%	76.96%

είπαμε να μειώσουμε τις εποχές στις 23 και να αυξήσουμε το μέγεθος του batch στο 100

training_score	test_score
80.9%	77%

Για να γίνει ακόμα καλύτερο σκεφτήκαμε να βάλουμε 30 εποχές κρατώντας το batch_len στο 100

training_score	test_score
84.23%	77.8%

Στη συνεχεια αποκτήσαμε μια νέα γνώση το stop training αν δεν υπάρχει άνοδος στο score η μείωση.

Βάλαμε patience 4 για validation_loss

training_score	test_score
79.88%	78.2%

Μετά είπαμε να βάλουμε 4 conv Layers και 5 artificial Layers+Softmax Layer.

training_score	test_score

79.9%	77.56%

αφού πήγαν πολύ λίγο καλύτερα τα πράγματα
βάλαμε 5 Conv_Layers+4 artificial_Layers+SoftmaxLayer

training_score	test_score
79.16%	76.45%

Μετά είπαμε να μειώσουμε κατα 1 τα ArtificialLayers μήπως τα τόσα πολλά επίπεδα επηρεάζουν την εκπαίδευση.

training_score	test_score
80.44%	78.07%

Το καλύτερο μοντέλο που πήραμε τελικά είναι 5 convLayers 3 Artificial_Layers+softmax Layer.Μετά σε κάθε τεχνητό νευρώνα αυξάναμε το εύρος των αρχικών βαρων στην ανάλογη δύναμη και όπως είδαμε τελικά τα πράγματα ήταν χειρότερα.

training_score	test_score
78.92%	76.79%

Μετά αντί της δύναμης μετά από κάθε νευρώνα πολλαπλασιάζαμε με το μέσο αριθμό των νευρόνων για κάθε προηγούμενο Layer και μετά από κάθε artificial_Layer εκτός του τελευταίου λόγω softmax βάλαμε πιθανότητα απόριψης νευρώνα προηγούμενο Layer 20% για καλύτερη γενίκευση.Και για τους convLayers πιθανότητα 10%.

training_score	test_score
72.06%	73.3%

Οπότε βγάλαμε το Dropout από τους convLayers

training_score	test_score
77.21%	78.5%

Μετά λόγω ότι υπάρχουν λιγότερα outputs μετά από κάθε artificial Layer για κάθε layers ορίσαμε δύναμη του 10.

Στον 1ο 10^(-1),2ο 10^(-2) κλπ

training_score	test_score
82.04%	77.98%

Κρατώντα ίδια την μεθοδολογία αλλάξαμε τα epochs από 30 σε 60 μήπως με περισσότερες επαναλήψεις πετύχουμε καλύτερα αποτελέσματα.

training_score	test_score
84.89%	79.5%

Τελικά βγήκε αποτελεσματικό.Έπειτα βάλαμε σε όλα τα artificial Layers πλην του τελευταίου penalty 0.01 και στον τελευταίο 0.001 λόγω softmax δεν είμαστε πολύ αυστηροί.

training_score	test_score

92.1%	77.39%

Μετά κάναμε αλλαγή και είπαμε να βάλουμε patience 4 με monitor το val_accuracy βάλαμε σε όλους τους artificial neurons L2=0.01 .Είχαμε 40 νευρώνες σε όλα τα layers πλην του softmax και βάλαμε 60 epochs 50 batch_size 4 convLayers και 3 artificialLayers+ softmax Layer.

training_score	test_score
78.12%	79.1%

Μετά μειώνουμε το learning_rate στο 0.00001 μήπως βρεθεί καλύτερη μέθοδος.

training_score	test_score
70.64%	69.88%

Μετά αλλάξαμε το dataset και βάλαμε εικόνες μεγέθους 200x200.Επιστρέψαμε στο default learning 0.001
και βάλαμε ένα παραπάνω Conv Layer και βάλαμε batch_len 100 και epoch_num 60.

training_score	test_score
78.09%	75.26%

Μετά ξαναβάζουμε στον τελευταίο artificial νευρώνα(softmax) 0.001 και είχαμε αυτό

training_score	test_score

80.28%	78.41%

Μετά αλλάξαμε των αριθμό των νευρώνων που θα έχει κάθε artificial layer και τους κάναμε 30 κρατώντας τον αριθμό των νευρώνων στους convolutional Layers στους 40.
Το batch_len το βάλαμε στα 50 cases και χρησιμοποιήσαμε 5 convLayers,3 artificial Layers+Softmax Layer και σε κάθε layer αρχικοποιήσαμε τα βάρη των artificial layers στο διάστημα

Στο 1ο number of inputs δώσαμε τις διαστάσεις των εικόνων(100x100)
[-sqrt(number_inputs)^(-1),sqrt(number_inputs)^(-1)]

training_score	test_score
78.52%	77.05%

Μετά ξανά εκτελέσαμε την ίδια μεθοδολογία αλλάζοντας το 1ο input στον αριθμό των νευρώνων που παράγονται μετά το
Flatten layer (2560)

training_score	test_score
78.41%	76.02%

Αφού τα πράγματα δεν είχαν βελτίωση αποφασίσαμε να αυξήσουμε το batch_size με το σκεπτικό ότι μπορεί να ταν υπερβολικά μικρό και το κάναμε 200.

training_score	test_score
79.66%	77.39%

Τελικά από όσο καταλάβαμε έπαιξε σημαντικό ρόλο.Μετά δοκιμάσαμε άλλη τεχνική για τα βάρη αντί για την ρίζα των inputs βάλαμε qrt(2.0/number_of_inputs)*random_number(1,1000) και τελικά τα πράγματα δεν πήγαν καλά.

training_score	test_score
46.68%	47.35%

Αφου δεν πέτυχαν όσο θέλαμε αυτά αποφασίσαμε να βάλουμε στο παιχνίδι και το data augmentation.
Για κάθε εικόνα του dataset βάλαμε 3 augmented data με περιστροφή ,zoom out και επικέντρωση στα training και validation sets και βάλαμε monitor val_loss με patience 4 ,4 conv Layers των 40 νευρώνων και 3 Dense Layers των 30 νευρώνων με L2=0.01 + Softmax χωρίς L2

training_score	test_score
80.39%	78.16%

Μετά αλλάξαμε το batch size σε 150 σε περίπτωση που βάλαμε μεγάλο batch και αφαιρέσαμε το L2 από τους Dense νευρώνες και βάλαμε validation split το 33% των training data λόγω augmentation.

training_score	test_score
58.01%	78.07%

Μετά σκεφτήκαμε ότι οι εικόνες μεγέθους 200x200 μπορούν να δώσουν καλύτερα αποτελέσματα.Οπότε κάναμε resize τις αρχικές εικόνες σε 200x200.

Αφού είχαμε μεγαλύτερες διαστάσεις προσθέσαμε ένα convolutional layer και δοκιμάσαμε batches των 150,300,500 και τελικά μόνο των 150 κατάφεραν να τρέξουν,τα άλλα δεν μπόρεσαν λόγω μεγέθους batch και διαστάσεων.

training_score	test_score
80.47%	76.79%

Αφού δεν είδαμε βελτίωση και δεν μπορούσαμε να βάλουμε υψηλότερα batches επιστρέψαμε στο μέγεθος 100x100 και σε αυτά δοκιμάσαμε batches των 300 και 500.

300

training_score	test_score
81.27%	77.82%

500

training_score	test_score
81.12%	76.79%

Μετά αυξήσαμε τον αριθμό των εποχών στις 200 μήπως έχουμε καλύτερα αποτελέσματα και βάλαμε το batch_size στο 150 μήπως χρειάζεται μικρότερο batch_size και το μεγαλύτερο δημιουργεί προβλήματα.

training_score	test_score
80.14%	74.66%

Αφού τα πράγματα έγιναν χειρότερα βάλαμε batch_size 300 και epochs=100 μήπως χρειαζόνταν λιγότερες epochs και μεγαλύτερο batch_size.

training_score	test_score

80.77%	76.88%

Έπειτα κάναμε weight initialization σε όλα τα layers μαζί με τα convolutional_layers και τελικά δεν είχαμε καλά αποτελέσματα.

training_score	test_score
47.71%	49.32%

Αφαιρέσαμε τα weight initialization και για batch_size βάλαμε 200.

training_score	test_score
83.47%	76.62%

Μετά βάλαμε monitor το accuracy loss για να δούμε αν ο αλγόριθμος γίνεται να κάνει overfitting και βάλαμε batch 300.

training_score	test_score
77.90%	77.39%

Αφαιρέσαμε το monitor μήπως γίνονται χειρότερα τα πράγματα εξαιτίας του λόγω του ότι παγιδεύεται κάπου το score για κάμποσα βήματα.

training_score	test_score
97.44%	76.37%

Αφού είχαμε χειρότερα αποτελέσματα είπαμε να βάλουμε 32 εποχές

training_score	test_score
83.08%	77.79%

Μετά ξανά επαναφέραμε το patience=4 με monitor val_loss και βάλαμε Augmentation αποκλέιστικά στο Training dataset επειδή με validation τα augmented data το score δεν ανέβαινε καλά.
Και τα ποσοστά που βάλαμε ήταν **training:64% validation:16% test:20%.**

Βάλαμε ως input εικόνες των 200x200 βάλαμε batch_size=150 ,5 conv Layers των 40 νευρώνων και βάλαμε.

3 dense Layers των 30 νευρώνων + Softmax και 100epochs.

training_score	test_score
82.69%	78.41%

Διακρίναμε ότι υπήρχαν κάμποσα val_accuracy άνω του 80% σε κάποια epochs.
Για κάθε κατηγορία(Normal,Bacterial,Viral) βάλαμε 64% training 16%validation και 20% test για να έχουμε κατα κάποιον τρόπο μια ομοιομορφία.

training_score	test_score
79.71%	77.9%

Μετά σκεφτήκαμε και μια άλλη καινοτομία για κάθε layer να βάλουμε διαφορετικό αριθμό νευρώνων 3 conv Layers με 32 64 96 νευρώνες αντίστοιχα.
Σε όλους τους ConvLayers BatchNormalization 2 Dense Layers με 50 νευρώνες +Softmax. Dense Layers είχαν L2=0.0001 με monitor val_accuracy με patience 4 batch_size=150 και epochs=100.

training_score	test_score
81.63%	79.69%

Όπως φάνηκε ο μεταβλητός αριθμός νευρώνων καθώς και λιγότερα Dense Layers φέρνουν πολύ καλύτερα αποτελέσματα.

Επόμενο που κάναμε ήταν να χρησιμοποιήσουμε εικόνες μεγέθους 200x200 με batch_size=100 και 100 epochs
4 ConvLayers με 32,32*2,32*4,32*8 νευρώνες αντίστοιχα
2 Dense Layers με 40 και 30 νευρώνες αντίστοιχα+Softmax
Ομως απ όλο το training set αφαιρέσαμε τα μισά δείγματα λόγω μεγάλων καθυστερήσεων στους υπολογισμούς.

training_score	test_score
56.46%	77.39%

Επειδή συνήθως τέτοιοι ConvLayers είναι καλοί απ όσο ξέρουμε σκεφτήκαμε να μειώσουμε το batch_size στο 75

training_score	test_score
54.69%	75.43%

Τελικά δεν είναι σωστό να αφαιρούμε data.

Μετά για να μην υπερφορτωθεί το σύστημα αποφασίσαμε από τα παραπάνω augmentations να κρατάει 1 στην τύχη.
Επίσης μετά από 2 convLayers να γίνεται MaxPooling και όχι μετά από έναν
Το padding το κάναμε same ετσι ώστε να μην μειώνονται οι διαστάσεις σε κάθε convLayer παρα μόνο τα κενά να συμπληρόνονται με τον κοντινότερο γείτονα
για κάθε DenseLayer εκτός από το softmax βάλαμε Dropout=0.2.Για όλα τα DenseLayers βάλαμε L2=10^(-6).

training_score	test_score
54.03%	58.62%

Κοιτώντας όμως παραδείγματα της αρχιτεκτονικής VGG16 αποφασίσαμε να χρησιμοποιήσουμε εικόνες 200x200 να βάλουμε patience 20 με monitor val_accuracy και να κρατήσουμε τα βάρη από το καλύτερο epoch

Για να μην τρέχει για μέρες δεν βάλαμε Augmented Data και στους 2 πρώτους ConvoLutional Layers κάναμε MaxPooling ανά Layer.Μετά ανά 2 Layers

Βάλαμε batch_size=50 100 εποχές L2=10^(-7) στους Dense Layers

6 ConvLayers με 32,64,128,128,256,256 νευρώνες αντίστοιχα
2 DenseLayers με 40 και 30 νευρώνες αντίστοιχα με Dropout=0.2 και τον DenseLayer του SoftMax.

Μέσα σε 8 με 10 ώρες είδαμε αυτά τα αποτελέσματα.

training_score	test_score
83.4%	80.7%

Επίσης ασχοληθήκαμε και τον τον RESNET αλλα δεν είχαμε κάποια καλά αποτελέσματα.

1.7 Συμπεράσματα

Τελικά και μόνο με softmax να λειτουργήσει το σίγουρο είναι πως μετά από 300 εποχές το πολύ αν δεν βάλουμε παραμέτρους για να σταματήσουμε το overfitting(L2,Dropout,Augmentation,ConvLayers,DenseLayers,etc) το σίγουρο είναι ότι το error στα training data θα είναι τουλάχιστον απειροελάχιστο.
Γι αυτό πρέπει να συμμετέχουν και τα παραπάνω που σταματούν το overfitting η το επιβραδύνουν πολύ μειώνοντας τις τιμές των βαρών για να μειωθεί το error.Μπορεί έτσι να υπάρχουν λιγότερες επιλογές για τιμές βαρών αλλα θα κρατηθούν τιμές που βοηθάνε να μαθαίνει καλύτερα ο αλγόριθμος αγνωστα data.
Επίσης κάποιες φορές μπορεί το validation accuracy να ναι μεγάλο αλλά στα περισσότερα παραδείγματα τουλάχιστον 1 φορά θα ναι τυχαίο.
Ακόμη δεν σημαινει οτι το val_accuracy αν σταματήσει να αυξάνεται για λίγο έφτασε στο μέγιστο.Μπορεί άνετα να αυξηθει ξανά μετά από 10 με 20 epochs αν προσεγγίσουμε την επίλυση του προβλήματος.
Τέλος με τις παραμέτρους για να αποφύγουμε το Overfitting δεν πρέπει να το παρακάνουμε
Για pneumonia detection μέθοδος για να μην γίνει underfitting:

- **DenseLayers:3**

- **Dropout:Το πολύ 0.2 και μόνο σε DenseLayers εκτός του SoftMax φυσικά**
- **Augmentation:3 augmentations αρκούν.Παραπάνω δεν χρειάζεται**
- **L2=Οχι πάνω από 10^(-5)**

1.8 Προβλήματα που συναντήσαμε

Θα μπορούσαμε να βάλουμε περισσότερους convolutional Layers και μεγαλύτερες διαστάσεις εικόνας όπως 800x800 αλλά οι υπολογιστικοί πόροι που έχουμε δεν μας το επιτρέπουν λόγω του ότι θα υπάρχει υπερβολική καθυστέρηση(4-5 ημέρες τουλάχιστον).Θεωρούμε ότι το πλή δεδομένα του dataset δεν

Επιπλέον θα μπορούσαμε να ερευνήσουμε ποιο εξονυχιστικα την υλοποίηση με resnet50 όπου οι πρώτοι μας πειραματισμοί δεν μας έφεραν καλά αποτελέσματα.

1.9 Κώδικας

Έχει δημιουργηθεί ένα git repo με σκοπό την επικοινωνία των μελών της ομάδας και για την παρουσίαση του κώδικα μας.
https://github.com/ManosMorf97/Rapid_Diagnosis_of_Pneumonia

Επίσης Υπάρχει και στο Google drive όπου υπάρχει πρόσβαση για χρήστες με hua mail
Rapid Diagnosis of Pneumonia

1.10 Δημοσιεύσεις

- https://www.tandfonline.com/doi/full/10.1080/00051144.2021.1973297
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8423280
- https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

1.11 Εκκίνηση

Κανουμε download και install το Anaconda
Ανοίγουμε anaconda cmd και γράφουμε την εντολή

```
jupyter notebook –generate-config
```

Μπαίνουμε στο αρχείο C:\Users\yourname\.jupyter\jupyetr_notebook_config.py
κάνουμε 2 declarations

c.NotebookApp.notebook_dir = ‘το path που αποθηκεύτηκε η εργασία’
c.NotebookApp.iopub_data_rate_limit = 1.0e10
για επεξεργασία μεγάλου όγκου δεδομένων(τοπικά)
Ανοίγουμε το Anaconda και διαλέγουμε το Jupyter Notebook