ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEBINTELLIGENCE

Αναγνώριση μελανώματος από ιατρικές εικόνες χρησιμοποιώντας βαθιά νευρωνικά δίκτυα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΤΖΟΥΡΕΛΗ ΣΤΥΛΙΑΝΟΥ

Επιβλέπων : Κωνσταντίνος Ι. Διαμαντάρας
Καθηγητής, ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Σεπτέμβρης 2021

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Αναγνώριση μελανώματος από ιατρικές εικόνες χρησιμοποιώντας βαθιά νευρωνικά δίκτυα

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΤΖΟΥΡΕΛΗ ΣΤΥΛΙΑΝΟΥ

Επιβλέπων : Κωνσταντίνος Ι. Διαμαντάρας
Καθηγητής ΔΙ.ΠΑ.Ε.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις Choose a date.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο	Όνομα Επώνυμο	'Ονομα Επώνυμο
Choose an item. ΔΙ.ΠΑ.Ε.	Choose an item. ΔΙ.ΠΑ.Ε.	Choose an item. ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Choose a date

(Υπογραφή)

.....

Click here to enter text.

Click here to enter text.

Περίληψη

Το μελάνωμα θεωρείται από τους πιο σοβαρούς τύπους δερματικού καρκίνου και ο 19ος πιο συχνός καρκίνος με 287.723 χιλιάδες διαγνώσεις το 2018 παγκοσμίως . Υπολογίζεται ότι την τελευταία δεκαετία (2011 – 2021) οι περιπτώσεις επιθετικού μελανώματος έχουν αυξηθεί κατά 44% [1]. Είναι ευρέως γνωστό ότι ο έγκυρος και έγκαιρος εντοπισμός, καθώς και η πρόληψη, είναι πολύ σημαντικά για τον καρκίνο του δέρματος και πιο συγκεκριμένα για το μελάνωμα, αφού μειώνουν δραματικά το κόστος θεραπείας αλλά και τη θνησιμότητα.

Τα βαθιά νευρωνικά δίκτυα θα μπορούσαν να παίξουν ουσιαστικό ρόλο στην έγκαιρη διάγνωση του μελανώματος βοηθώντας στην κλινική εξέταση μειώνοντας την ανάγκη για επίπονες και ακριβές βιοψίες. Το πρόβλημα της αναγνώρισης κακοήθειας στο δέρμα δεν είναι καινούριο πρόβλημα, πάραυτα η διπλωματική εργασία αυτή εστιάζει σε δύο κύρια μέρη: 1) το νευρωνικό δίκτυο που αναπτύχθηκε να μπορεί αποτελεσματικά να γενικεύσει και να μην είναι αποτελεσματικό μόνο για ένα dataset, όπως είναι το ISIC 2019, 2020 ή MNIST-HAM, και 2) να μπορεί να εντοπίζει το μελάνωμα μέσα από φωτογραφίες που τραβήχτηκαν από κινητό συχνά με χαμηλό φωτισμό, ανάλυση, αλλά και να μπορεί να δίνει έγκυρη πρόγνωση αγνοώντας στοιχεία του σώματος όπως τις τρίχες.

Για τον εντοπισμό του μελανώματος χρησιμοποιήθηκε μοντέλο efficientnet στην έκδοση B0 με 5,330,571 παραμέτρους εκ των οποίων πολλοί από αυτούς ήταν παγωμένοι ώστε να μην επηρεαστούν τα batch normalization layers του συγκεκριμένου μοντέλου [2]. Στο μοντέλο αυτό προστέθηκε ένα attention mechanism [21] το οποίο αύξησε δραματικά την αποτελεσματικότητά του. Το μοντέλο πέτυχε 91,22% accuracy κατά μέσο όρο σε 4 dataset που δοκιμάστηκε (MNIST-HAM, ISIC 2019, ISIC 2020, benign vs malignant).

Λέξεις Κλειδιά: efficient-net, efficient-netb0, melanoma, CNN, malignant, transfer learning, blocks, attention mechanism.

Abstract

Melanoma is considered to be one of the most dangerous skin cancer types and the 19th most frequent type of cancer in the world with 287,723 diagnoses in 2018 alone. It is estimated that in the last decade (2011 – 2021) case of aggressive melanoma have risen by 44% [1]. It is widely accepted that early detection and prophylaxis are very important for skin cancer and especially for melanoma since they greatly reduce cost of treatment and fatality.

Deep neural networks could play a very significant role in early detection of melanoma assisting in the clinical assessment process while reducing the need for painful and costly biopsies. Classifying benign and malignant lesions is not a new topic but this thesis is focused in 2 specific topics that there is a lack of in other solutions: 1) the neural network is developed with generalization in mind across different datasets and not optimized for just one of the most known databases such as ISIC 2019,2020 and MNIST-HAM and 2) to be able to detect melanoma in conditions that exists in pictures captured from mobiles phone, such as but not limited to, low lighting, low resolution and skin characteristics such as hair.

For the detection of the melanoma an efficientnet model was used with transferred learning. We used the B0 variant with 5,330,571 parameters while we froze most of them to avoid ruining the batch normalization layers of the model [2]. Attached to model is an attention mechanism [21] which that greatly increased the model's accuracy. The model achieved an overall accuracy of 91,22% across 4 datasets (MNIST-HAM, ISIC 2019, ISIC 2020, benign vs malignant).

Keywords: efficient-net, efficient-netb0, melanoma, CNN, malignant, transfer learning, blocks, attention mechanism.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Νευρωνικά Δίκτυα στον τομέα της Υγείας

Οι διαγνώσεις στον τομέα της υγείας με την βοήθεια των υπολογιστών και επεξεργασία εικόνων είναι ένας τομέας που εξετάζεται εδώ και δύο δεκαετίες με την πρώτη επιστημονική δημοσίευση να δημοσιοποιείται το 1985 [3]. Αναλύοντας εικόνες προσπαθούμε να αναγνωρίσουμε στοιχεία, ώστε να τα χαρακτηρίσουμε είτε ως ασφαλή ή ως επικίνδυνα. Στην περίπτωση του μελανώματος, τα χαρακτηριστικά που ψάχνουμε είναι ασυμμετρία, πολλές διαφορές στο χρώμα καθώς και μεγάλη διάμετρος.

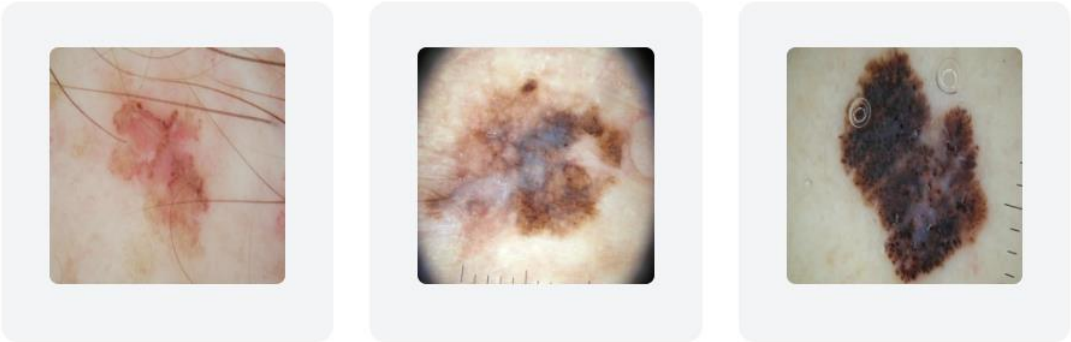
Τα πλεονεκτήματα των νευρωνικών δικτύων σε σχέση με τον παραδοσιακό προγραμματισμό είναι η δυνατότητά τους να λύνουν προβλήματα στα οποία δεν υπάρχει αλγοριθμική λύση ή αυτή είναι πολύ περίπλοκη για να βρεθεί. Στον τομέα της υγείας, τα νευρωνικά δίκτυα έχουν χρησιμοποιηθεί για κλινικές διαγνώσεις, ανάλυση εικόνων και εξαγωγή συμπερασμάτων από αυτές, καθώς και δημιουργία φαρμάκων [4].

1.2 Το Μελάνωμα και το Ανθρώπινο Δέρμα

Ο καρκίνος του δέρματος βρίσκεται ανάμεσα στους 20 πιο διαδεδομένους καρκίνους του κόσμου και χωρίζεται σε δύο κατηγορίες, το μελάνωμα (που είναι και η πιο πιθανή αιτία θανάτου από δερματικό καρκίνο [5]) και καρκίνους που δεν έχουν σχέση με αυτό. Όπως προαναφέρθηκε, η πρόληψη είναι η καλύτερη λύση για το μελάνωμα και η έγκαιρη διάγνωσή

του είναι σημαντική για την θεραπεία και την αντιμετώπισή του, διότι -αν και εμφανίζεται στο δέρμα- ο κίνδυνος μετάστασης είναι μεγάλος [6].

Η κλινική μέθοδος για τη διάγνωση του μελανώματος ξεκινάει με την οπτική εξέταση από έναν δερματολόγο και στη συνέχεια ακολουθεί η βιοψία και η ιστοπαθολογική ανάλυση. Το κυριότερο πρόβλημα που αντιμετωπίζουν οι δερματολόγοι είναι ότι πολλές φορές το μελάνωμα μπορεί να μιμείται την ανάπτυξη μιας καλοήθους δερματικής βλάβης (ή/και το αντίθετο), με αποτέλεσμα η εγκυρότητα της διάγνωσης των δερματολόγων να κυμαίνεται κατά μέσο όρο στο 80% [3].



Εικόνα 1. Παραδείγματα κακοήθειας

Στην εικόνα 1, μπορούμε να δούμε παραδείγματα από κακοήθειες που οι δερματολόγοι συνήθως προσπαθούν να αναγνωρίσουν κατά την οπτική εξέταση του δέρματος. Στην αριστερή εικόνα, διακρίνονται διάφορες αποχρώσεις του δέρματος και μη διακριτό όριο της αλλοίωσης. Στην κεντρική εικόνα, δεν υπάρχει συμμετρία, κάτι που μπορεί να παραπέμπει σε κακοήθεια. Τέλος, ο λόγος που πρέπει να παρακολουθούνται συχνά αυτές οι δερματικές αλλοιώσεις είναι ότι ενώ μπορεί να μην διακρίνονται κακοήθεις στην αρχή, υπάρχει η πιθανότητα τα μελανώματα να μεταλλάσσονται με πολύ γρήγορους ρυθμούς και να παρουσιάσουν στην πορεία.

Αυξάνοντας την πιθανότητα έγκυρου εντοπισμού της κακοήθειας ως εναλλακτική λύση στην βιοψία είναι η κυριότερη προτεραιότητα στη μάχη κατά του μελανώματος, επομένως αναπτύσσοντας τέτοιες μεθόδους μπορούμε να μειώσουμε δραστικά τα ποσοστά θνησιμότητας. Τα τελευταία χρόνια η ανάπτυξη της τεχνολογίας των προσωπικών υπολογιστών και των κινητών, καθώς και οι πρόοδοι που έχουν σημειωθεί στην τεχνητή νοημοσύνη, μας δίνουν μια μοναδική ευκαιρία να δώσουμε την δυνατότητα σε κάθε άνθρωπο να κάνει μόνος του έναν πρώιμο έλεγχο με αυτές τις συσκευές, πριν από την επίσκεψη στον δερματολόγο του.

1.3 Στόχος Εργασίας

Ο στόχος αυτής της διπλωματικής εργασίας είναι να αναπτυχθεί ένα νευρωνικό δίκτυο που θα μπορεί να εντοπίζει κακοήθειες στο ανθρώπινο δέρμα πριν αυτές αποβούν μοιραίες για τον/την ασθενή, ενώ ταυτόχρονα να μπορεί να λειτουργήσει σε δυσμενείς συνθήκες (όπως χαμηλή ανάλυση εικόνων, μέτριος φωτισμός), αλλά και να αγνοεί λάθη στα χρώματα που συνήθως συμβαίνουν σε φθηνούς αισθητήρες κινητών τηλεφώνων. Ταυτόχρονα, το δίκτυο αυτό θα πρέπει να έχει μέσο όρο ακρίβειας ίσο ή καλύτερο από τον μέσο όρο των δερματολόγων ώστε να θεωρηθεί αξιόπιστο και εύχρηστο.

Εκτός από το νευρωνικό δίκτυο, όπως προαναφέρθηκε, άλλος ένας στόχος της διπλωματικής είναι να κάνει προσβάσιμη αυτή την τεχνολογία σε κάθε χρήστη κινητού τηλεφώνου ή ηλεκτρονικού υπολογιστή, έτσι ώστε οι έλεγχοι να είναι συχνότεροι και ευκολότεροι, με αποτέλεσμα να αυξηθούν οι πιθανότητες να εντοπιστεί μια πιθανή κακοήθεια.

1.3.1 Συνεισφορά

1. Το πρώτο βήμα της διπλωματικής εργασίας ήταν η συλλογή των ερευνητικών δεδομένων. Συγκεντρώσαμε 46.078 φωτογραφίες συνολικά από διάφορα dataset, καθένα με τις δικές του ιδιαιτερότητες. Μεγαλώνοντας την ποικιλία των δεδομένων μας, σιγουρέψαμε ότι το μοντέλο θα γενικεύει καθολικά και ότι τα αποτελέσματά του δεν θα είναι εξειδικευμένα μόνο σε μια κατηγορία εικόνων ή κακοήθειας. Αξίζει να σημειωθεί ότι δεν χρησιμοποιήθηκαν όλες οι εικόνες στην εκπαίδευση του νευρωνικού δικτύου. Ωστόσο, όσες έλειπαν από την εκπαίδευση χρησιμοποιήθηκαν στην αξιολόγηση των αποτελεσμάτων.
2. Στην συνέχεια δοκιμάσαμε πολλές τεχνικές data augmentation προκειμένου να καταλήξουμε στην πιο αποτελεσματική για το σύνολο των δεδομένων μας. Ενδεικτικά δοκιμάστηκαν ρυθμίσεις όπως:
   - το χρώμα των εικόνων που δίνουμε στο μοντέλο κατά την διαδικασία της εκπαίδευσης (έγχρωμες, ασπρόμαυρες)
   - αυξομείωση του φωτισμού, καθώς και της αντίθεσης των εικόνων, για να προσομοιώσουμε συνθήκες λήψης από κινητό τηλέφωνο
   - προσομοίωση δερματικών τριχών επάνω από τις δερματικές αλλοιώσεις της εικόνας, έτσι ώστε το μοντέλο να μάθει να της αγνοεί.
3. Το επόμενο βήμα της διπλωματικής εργασίας ήταν να εκπαιδεύσουμε σε διάφορα ζευγάρια εικόνων το νευρωνικό δίκτυο, ώστε να συγκρίνουμε την ακρίβεια των μοντέλων ανάλογα με το dataset στο οποίο εκπαιδεύτηκε, αφαιρώντας έτσι εικόνες

που ήταν είτε *«η εξαίρεση στον κανόνα»* είτε λανθασμένα κατηγοριοποιημένες. Στη συνέχεια, επιλέξαμε το μοντέλο που μας έδινε τα πιο ακριβή αποτελέσματα.

4. Τέλος, αναπτύξαμε μια διαδικτυακή εφαρμογή χρησιμοποιώντας τις τελευταίες τεχνολογίες που είναι διαθέσιμες στο web development με το Laravel framework και αξιοποιήσαμε το νευρωνικό δίκτυο από τα προηγούμενα βήματα με σκοπό να δώσουμε την δυνατότητα στον εκάστοτε χρήστη να κάνει έναν βασικό έλεγχο των δερματικών αλλοιώσεών του.

1.4 Οργάνωση κειμένου

Αμέσως μετά την εισαγωγή σε αυτή την διπλωματική βρίσκονται τα εξής κεφάλαια:

- Το κεφάλαιο 2 περιλαμβάνει μια σύντομη ανασκόπηση της βιβλιογραφίας και διαφόρων δημοσιεύσεων που αφορούν στο μελάνωμα και στον εντοπισμό του από νευρωνικά δίκτυα, καθώς και σε προβλήματα που αντιμετωπίζουν αυτές οι τεχνικές.
- Στο κεφάλαιο 3 παρουσιάζουμε το θεωρητικό υπόβαθρο της διπλωματικής αυτής εργασίας και το τι πρέπει ο αναγνώστης να γνωρίζει πριν την ανάγνωσή της.
- Στο κεφάλαιο 4 παρουσιάζουμε τη διαδικασία συγκέντρωσης των δεδομένων και την επεξεργασία που ακολουθήθηκε πριν τα χρησιμοποιήσουμε.
- Στο κεφάλαιο 5 περιγράφουμε τη διαδικασία κατασκευής, αλλά και τη δομή του νευρωνικού δικτύου που χρησιμοποιήθηκε.
- Στο κεφάλαιο 6 αναλύουμε τα αποτελέσματα των πειραμάτων και τις παραμέτρους αξιολόγησης του μοντέλου.
- Στο κεφάλαιο 7 παρουσιάζουμε τις τεχνικές λεπτομέρειες της web εφαρμογής, του νευρωνικού δικτύου, καθώς και της γέφυρας που τα συνδέει.

2

Σχετικές εργασίες

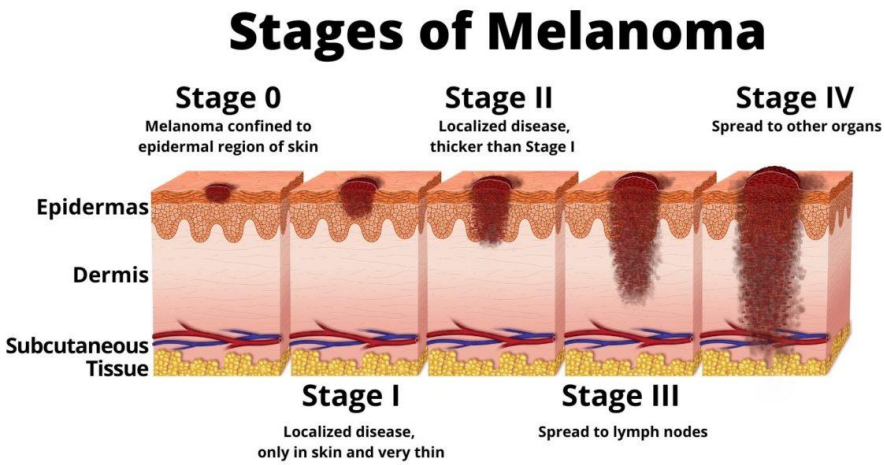
Το μελάνωμα, όπως και οι τεχνικές εντοπισμού του είτε με παραδοσιακές μεθόδους είτε με πιο σύγχρονους τρόπους όπως είναι τα νευρωνικά δίκτυα, δεν είναι ένα καινούριο ζήτημα, επομένως οι επιστημονικές βιβλιογραφικές αναφορές είναι αρκετά πλούσιες και με πολλές υποενότητες. Ωστόσο, το πρώτο θέμα με το οποίο ασχοληθήκαμε στην παρούσα εργασία είναι η δομή του ανθρώπινου δέρματος και τα χαρακτηριστικά του μελανώματος, προκειμένου να κατανοήσουμε καλύτερα το πρόβλημα που έχουμε να αντιμετωπίσουμε. Έπειτα βέβαια, συγκεντρώσαμε εργασίες σχετικές με το μελάνωμα που εστιάζουν στον εντοπισμό του χρησιμοποιώντας νευρωνικά δίκτυα.

2.1 Μελάνωμα και το Ανθρώπινο Δέρμα

Για το ανθρώπινο δέρμα και τις λειτουργίες του, αλλά και για το μελάνωμα, έχουν δημοσιευθεί πολλές εργασίες τις τελευταίες δεκαετίες από τις οποίες μπορούμε να εξάγουμε συμπεράσματα για να κατανοήσουμε τις βασικές παραμέτρους του προβλήματος αυτού. Το ανθρώπινο δέρμα αποτελείται από τρία κυρίως μέρη:

- την επιδερμίδα (το εξωτερικό στρώμα),
- το χόριο ή κυρίως δέρμα,
- και ο υποδόριος ιστός.

Κάθε ένα από αυτά τα στρώματα έχει τον δικό του ρόλο. Στο σύνολό του το ανθρώπινο δέρμα είναι από τα σημαντικότερα όργανα στο σώμα μας και είναι υπεύθυνο για σημαντικές λειτουργίες όπως η ρύθμιση της θερμοκρασίας, η αποβολή και απορρόφηση διαφόρων ουσιών, καθώς και η αποθήκευση λίπους στο κυρίως δέρμα [8].



Εικόνα 2. Στάδια Μελανώματος.

Η επιδερμίδα είναι υπεύθυνη για την προστασία του δέρματος και αποτελείται από ειδικά κύτταρα που ονομάζονται μελανοκύτταρα. Τα μελανοκύτταρα παράγουν μια ουσία που ονομάζεται μελανίνη όταν δεχθούν υπεριώδη ακτινοβολία από τον ήλιο ή από άλλες πηγές. Η διαδικασία αυτή είναι υπεύθυνη για το μαύρισμα του δέρματος και έχει ως σκοπό την προστασία της επιδερμίδας [9]. Στην εικόνα 2 μπορούμε να δούμε τα αποτελέσματα της υπερβολικής υπεριώδους ακτινοβολίας, καθώς και τα στάδια του μελανώματος στο δέρμα. Είναι λοιπόν ξεκάθαρο γιατί ο έγκαιρος εντοπισμός και η επικείμενη αφαίρεση του μελανώματος είναι σημαντικά για την επιβίωση του ασθενούς.

2.2 Εντοπισμός του Μελανώματος με Νευρωνικά Δίκτυα

Ο εντοπισμός του μελανώματος από νευρωνικά δίκτυα είναι ένα θέμα που έχει εξετασθεί πολλές φορές σε πολλούς διαγωνισμούς όπως τον ετήσιο International Skin Imaging Collaboration (ISIC) που διεξήχθησε το καλοκαίρι του 2020 και το dataset του διαγωνισμού χρησιμοποιήθηκε και στη διπλωματικής μας. Παρ’ όλα αυτά, λόγω της περιπλοκότητας του προβλήματος υπάρχουν πολλά περιθώρια βελτίωσης στον τομέα.

Η πρώτη εργασία που εξετάσαμε είναι η νικητήρια του διαγωνισμού του ISIC-2020 με πολύ εντυπωσιακά αποτελέσματα. Η εργασία πέτυχε 0,96 AUC (Area Under The Curve) στον διαγωνισμό με τα δημόσια δεδομένα, ενώ το τελικό αποτέλεσμα ήταν 0,94 AUC. Για να επιτευχθεί το αποτέλεσμα αυτό χρησιμοποιήθηκαν 7 διαφορετικά μοντέλα ανάμεσά τους και το EfficentNet [11] στις εκδόσεις B3-B7, αλλά και 2 εκδόσεις του μοντέλου ResNet το ResNeSt [12] και το ResNeXt [13] με 5-fold στα δεδομένα εκπαίδευσης [10]. Τα αποτελέσματα της εργασίας αυτής -αν και αξιοσημείωτα- είναι διαμορφωμένα μόνο στα δεδομένα του ISIC-2020

και δεν αποδίδουν αποτελεσματικά σε άλλα dataset. Τέλος, λόγω των πολλών μοντέλων που χρησιμοποιήθηκαν για να παραχθούν οι προβλέψεις, είναι δύσκολο να χρησιμοποιηθούν σε μια εφαρμογή όπου ο τελικός χρήστης θα περίμενε άμεσα αποτελέσματα για τη δερματική του αλλοίωση.

2.3 Αφαίρεση Τριχών Από Εικόνες Μελανώματος

Η αφαίρεση τριχών είναι μια πολύ βασική προεργασία στην επεξεργασία εικόνων για τον εντοπισμό του μελανώματος καθώς επιτρέπει στους δερματολόγους να παρακάμπτουν την διαδικασία ξυρίσματος της περιοχής που θέλουν να εξετάσουν. Ο πρώτος αλγόριθμος αφαίρεσης παρουσιάστηκε το 1997 από τον Lee et al. (Dullazor) και αφορούσε εικόνες που έχουν τραβηχτεί από δερματοσκόπιο. Μετά την εμφάνιση αυτού του αλγορίθμου έχουν γίνει αρκετές παραλλαγές και βελτιώσεις επάνω σε αυτόν και χρησιμοποιείται μέχρι και σήμερα. Ο αλγόριθμός λειτουργεί με 3 βήματα:

1) Εντοπίζει την περιοχή που περιέχει τις τρίχες χρησιμοποιώντας ειδικά φίλτρα.

2) Αντικαθιστά τα pixel που περιέχουν την τρίχα με γειτονικά.

3) Χρησιμοποιεί ειδικό αλγόριθμο για να εξομαλύνει την εικόνα και να φτάσει στο τελικό αποτέλεσμα [25].

Σε πιο πρόσφατες έρευνες ο Abass et al. [26] πρότεινε έναν αυτόματο αλγόριθμο αφαίρεσης τριχών από εικόνες δερματοσκοπίου, όπου διατηρεί τα στοιχεία του μελανώματος που βρίσκονται κάτω από αυτές. Ο αλγόριθμος αυτός αξιολογήθηκε σε 100 εικόνες και τα αποτελέσματά του συγκρίθηκαν με 2 παραδοσιακές μεθόδους αφαίρεσης και εξομάλυνσης, τις οποίες ξεπέρασε με ακρίβεια 93%.

Η αφαίρεση των τριχών παραμένει ως πρόβλημα στον εντοπισμό του μελανώματος, διότι καμία από τις τεχνικές που είναι διαθέσιμες δεν το έχει λύσει πλήρως. Ακόμη, η αποτελεσματικότητα των αλγορίθμων αυτών εξαρτάται από τα δεδομένα που τους δίνουμε, καθώς αν εφαρμοστεί τεχνική αφαίρεσης σε εικόνα που δεν περιέχει τρίχες, θα αλλοιωθεί η εικόνα προς εξέταση. Η διπλωματική αυτή, αντί για την αφαίρεση τριχών, κάνει προσθήκη προσομοιωμένων επάνω στις εικόνες με σκοπό το νευρωνικό δίκτυο να μάθει να τις αγνοεί, μια λύση που είναι ανεξάρτητη από τα δεδομένα εισόδου.

3

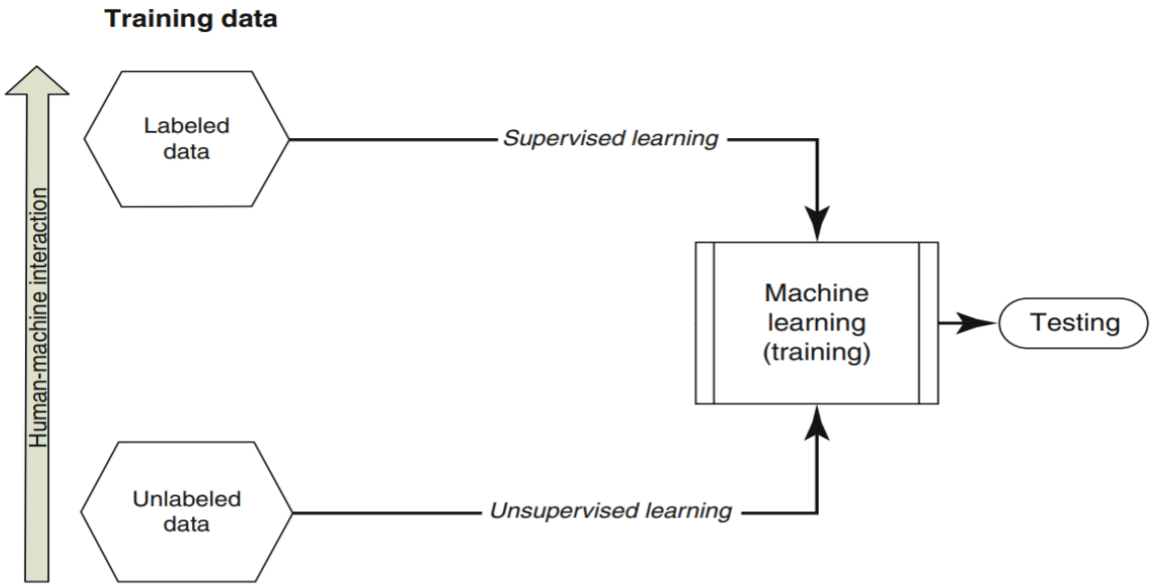
Θεωρητικό υπόβαθρο

3.1 Μηχανική Μάθηση

Η μηχανική μάθηση σύμφωνα με τον El Naqa είναι ένας συνεχώς εξελισσόμενος κλάδος των υπολογιστικών αλγορίθμων με σκοπό να μιμηθούν την ανθρώπινη ευφυΐα μαθαίνοντας από το περιβάλλον τους. Ένας αλγόριθμος μηχανικής μάθησης διαφέρει από τον κλασικό αλγόριθμο, καθώς ο προγραμματιστής δεν χρειάζεται να βρει την λύση στο πρόβλημά του, αλλά να δώσει τα δεδομένα στον αλγόριθμο και αυτός με την σειρά του μέσω της επανάληψης να ρυθμιστεί κατάλληλα για τα επιθυμητά αποτελέσματα [14]. Όπως φαίνεται στην εικόνα 3, η μηχανική μάθηση χωρίζεται σε 2 κατηγορίες όσον αφορά τα δεδομένα εκπαίδευσης:

- Στην μάθηση με επίβλεψη τα δεδομένα εκπαίδευσης δίνονται στον αλγόριθμο με ετικέτες που υποδεικνύουν σε ποια ομάδα ανήκουν οι εικόνες. Έτσι, ο αλγόριθμος ρυθμίζει τον εαυτό του ώστε να αναγνωρίζει τα χαρακτηριστικά της κάθε ομάδας.
- Στην μάθηση χωρίς επίβλεψη τα δεδομένα δίνονται στον αλγόριθμο χωρίς ετικέτες με αποτέλεσμα να τα χωρίζει από μόνος του σε ομάδες.

Όπως φαίνεται και στην εικόνα 3, η επίδραση του ανθρώπου με τον αλγόριθμο αυξάνεται στη μηχανική μάθηση με επίβλεψη.



Εικόνα 3. Κατηγορίες μηχανικής μάθησης ανάλογα με τα δεδομένα εκπαίδευσης.

3.2 Ταξινόμηση

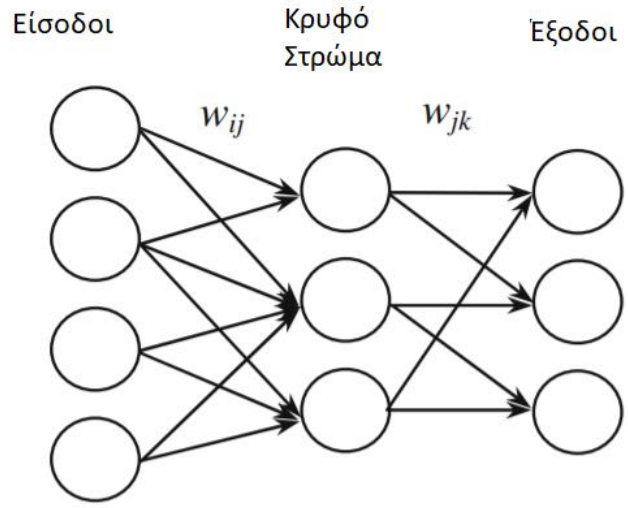
Η ταξινόμηση είναι η πιο σύνηθης διαδικασία που αναθέτουμε σε ευφυή συστήματα όπως είναι και τα νευρωνικά δίκτυα. Κατά τη διαδικασία της ταξινόμησης, ο αλγόριθμος πρέπει να καταλήξει σε ένα συμπέρασμα από τα δεδομένα που του έχουν δοθεί κατά την εκπαίδευση, ώστε να χτίσει έναν κατανεμητή κατάλληλο να προβλέπει καινούρια δεδομένα. Διάφοροι τρόποι ταξινόμησης παραθέτονται παρακάτω, ωστόσο αξίζει να σημειωθεί ότι εμείς χρησιμοποιήσαμε ένα νευρωνικό δίκτυο.

- Δέντρα αποφάσεων
- Ταξινομητές βασισμένοι σε κανόνες
- Δίκτυα Bayes

3.3 Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα αποτελούνται από έναν μεγάλο αριθμό τεχνητών νευρώνων που συνδέονται μεταξύ τους με ένα συνδυασμό συνδέσεων όπως φαίνεται και στην εικόνα 4. Οι νευρώνες χωρίζονται σε 3 γενικές κατηγορίες. Τους νευρώνες εισαγωγής που δέχονται τα δεδομένα με βάση τα οποία θα πρέπει να επεξεργαστούν [15]. Του νευρώνες εξόδου που

παράγουν το αποτέλεσμα ανάλογα με την είσοδο και τέλος, τους ενδιάμεσους κρυφούς νευρώνες.



Εικόνα 4. Δομή Νευρωνικού δικτύου.

Το μέγεθος του κρυφού στρώματος ρυθμίζεται από τον προγραμματιστή και εξαρτάται από το πρόβλημα που έχει να λύσει. Είναι σημαντικό να τονίσουμε πως θέλει πολλή προσοχή, καθώς ακόμα και ένα μικρό κρυφό στρώμα μπορεί να σημαίνει ότι το δίκτυο δεν μπορεί να βρει λύση στο πρόβλημά μας, ενώ ένα μεγάλο κρυφό στρώμα μπορεί να σημαίνει ότι το δίκτυο προσαρμόζεται μόνο στα δεδομένα εισόδου και δεν μπορεί να προβλέψει αποτελεσματικά τα δεδομένα που βλέπει για πρώτη φορά.

Ένα νευρωνικό δίκτυο, όπως αυτό της εικόνας 4, εξαρτάται από 3 πράγματα, τα δεδομένα εισόδου και τον αλγόριθμο ενεργοποίησης, την αρχιτεκτονική του δικτύου και τα βάρη των συνδέσεων μεταξύ των νευρώνων. Κατά τη διαδικασία εκπαίδευσης, τα βάρη αυτών των συνδέσεων συνήθως παίρνουν τυχαίες τιμές και στην συνέχεια δίνουμε στο δίκτυο τα δεδομένα εκπαίδευσης και συγκρίνουμε την έξοδο του δικτύου με το επιθυμητό αποτέλεσμα. Έπειτα, τα βάρη αυτών των συνδέσεων αλλάζουν, έτσι ώστε η έξοδος του δικτύου να πλησιάζει περισσότερο την έξοδο που θέλουμε να πετύχουμε. Η διαδικασία αυτή επαναλαμβάνεται πολλές φορές μέχρι τα αποτελέσματα του δικτύου να κριθούν επαρκή [15].

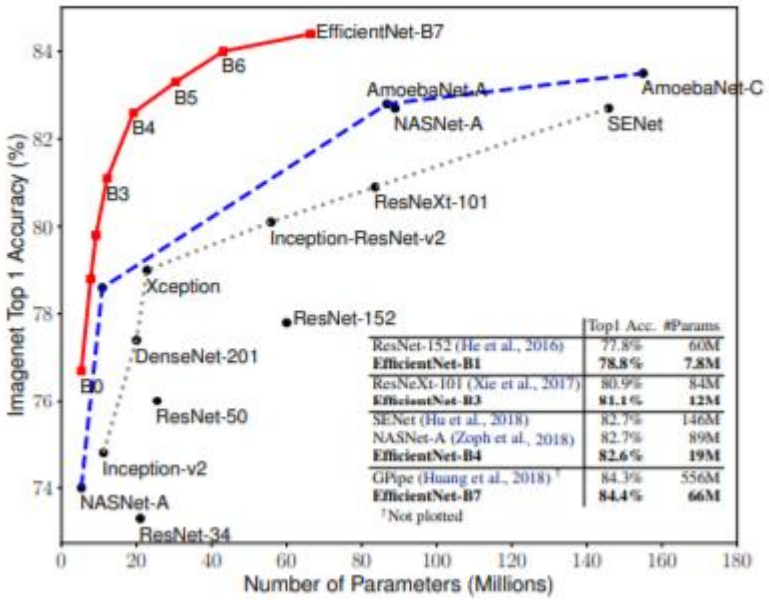
3.3.1 Convolution Neural Networks (CNNs)

Στη μηχανική μάθηση τα CNN είναι μια κλάση νευρωνικών δικτύων που συνήθως χρησιμοποιούνται για ανάλυση εικόνων ή video. Τα CNN διαφέρουν από τα απλά νευρωνικά δίκτυα, γιατί αντί να έχουν πολλούς τεχνητούς νευρώνες που συνδέονται μεταξύ τους, βασίζονται σε ιεραρχικά στρώματα που μικραίνουν όσο βαθύτερα βρίσκονται με αποτέλεσμα

να λειτουργούν σαν φίλτρα. Τα CNN είναι εμπνευσμένα από τις βιολογικές διαδικασίες καθώς οι συνδέσεις τους θυμίζουν πολύ το ζωικό οπτικό φλοιό [22].

3.3.2 Efficient Net

Το Efficient Net είναι ένα είδος νευρωνικού δικτύου το οποίο είναι σχεδιασμένο με τρόπο ώστε να είναι πολύ πιο αποδοτικό σε σχέση με τα υπόλοιπα. Δημοσιεύτηκε το 2019 από την Google και στην εικόνα 5 μπορούμε να δούμε πως η αποτελεσματικότητά του συγκρίνεται σε σχέση με τα υπόλοιπα νευρωνικά δίκτυα που χρησιμοποιούνται κυρίως για ταξινόμηση εικόνων. Το efficient net με λιγότερες παραμέτρους (στρώματα) σε σχέση με τα υπόλοιπα δίκτυα επιτυγχάνει μεγαλύτερη ακρίβεια στο dataset Imagenet (dataset για την αναγνώριση αντικειμένων που συνήθως χρησιμοποιείται για την αξιολόγηση νευρωνικών δικτύων) [11].



Εικόνα 5. Σύγκριση του EfficientNet σε σχέση με άλλα νευρωνικά δίκτυα.

3.3.3 Attention Mechanism

Τα νευρωνικά δίκτυα είναι αποτελεσματικά στο να παίρνουν χαρακτηριστικά χρήσιμα για την ταξινόμηση από φωτογραφίες. Υπάρχουν περιπτώσεις όμως που οι κλάσεις που θέλουμε να ταξινομήσουμε έχουν μικρές διαφορές μεταξύ τους και είναι δύσκολο για τον αλγόριθμο να εξάγει τα χαρακτηριστικά που είναι απαραίτητα. Ο μηχανισμός προσοχής (Attention Mechanism) είναι μια γρήγορη και αποτελεσματική μέθοδος που μιμείται τον τρόπο που οι άνθρωποι εξάγουν και ερμηνεύουν χαρακτηριστικά από εικόνες [21].

3.3.4 Optimizers

Optimizers είναι μαθηματικοί αλγόριθμοι που χρησιμοποιούνται κατά την διαδικασία της εκπαίδευσης του νευρωνικού δικτύου. Ο σκοπός τους είναι να μειώνουν όσο το δυνατόν περισσότερο το λάθος κατά την εκπαίδευση και να αυξάνουν την απόδοση του αλγορίθμου επηρεάζοντας τα βάρη των συνδέσεων μεταξύ των τεχνητών νευρώνων. Ο Adam optimizer είναι ένας από τους πιο γνωστούς optimizers καθώς, χωρίς να απαιτεί ιδιαίτερη ρύθμιση από τον χρήστη, μπορεί να εφαρμοστεί στα περισσότερα προβλήματα μηχανικής μάθησης.

3.4 Python

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου γενικής χρήσης, ενώ παράλληλα δίνει στον προγραμματιστή τα εργαλεία που χρειάζεται για να γράψει καθαρό και σαφή κώδικα. Ξεκίνησε να αναπτύσσετε το 1980 από τον Guido van Rossum, ενώ η έκδοση 3.8 έγινε διαθέσιμη το 2019. Η Python έχει μια από τις μεγαλύτερες βιβλιοθήκες και αυτό θεωρείται ως ένα από τα μεγαλύτερα πλεονεκτήματά της, γι’ αυτό και θεωρείται από τις πιο δημοφιλείς γλώσσες προγραμματισμού [16].

3.4.1 Tensorflow-Keras

Η Tensorflow είναι μια δωρεάν και ανοικτού λογισμικού βιβλιοθήκη για την Python, κατάλληλη για λειτουργίες όπως η μηχανική μάθηση. Δημιουργήθηκε από την Google το 2011 για εσωτερική χρήση και στη συνέχεια έγινε διαθέσιμη στο κοινό το 2015. Η βιβλιοθήκη κάνει χρήση του επεξεργαστή για να εκπαιδεύσει το νευρωνικο δίκτυο ή, αν ο υπολογιστής πληρεί κάποιες προϋποθέσεις, χρησιμοποιεί την κάρτα γραφικών για πιο γρήγορα αποτελέσματα [17].

3.5 PHP

Η PHP είναι μια γενική γλώσσα προγραμματισμού που χρησιμοποιείται κυρίως για την κατασκευή ιστοσελίδων. Κατασκευάστηκε το 1994 από τον δανό προγραμματιστή Rasmus Lerdorf, ενώ πλέον αναπτύσσεται από το PHP Group. Η PHP συνήθως συνδυάζεται από κάποιον web server που εκτελεί τον κώδικα που έχει γράψει ο προγραμματιστής, ενώ παράλληλα μπορεί να χρησιμοποιηθεί και κατευθείαν από την γραμμή εντολών του λειτουργικού. Είναι μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού για την κατασκευή ιστοσελίδων, ενδεικτικά ο οργανισμός W3Techs δήλωσε τον Απρίλιο του 2021 ότι

από όλες τις ιστοσελίδες που γνωρίζουν την γλώσσα με την οποία έχουν γραφτεί, το 79,2% ήταν γραμμένο σε PHP [18].

3.5.1 Laravel Framework

Το Laravel είναι ένα Framework ανοικτού κώδικα και δωρεάν γραμμένο σε PHP. Δημιουργήθηκε από τον Taylor Otwell και χρησιμοποιεί την αρχιτεκτονική προγραμματισμού Model-View-Controller. Η πρώτη έκδοση έγινε διαθέσιμη το 2011 και ανά τα χρόνια κέρδιζε όλο και περισσότερους χρήστες, ενώ ήδη από την έκδοση 5 που έγινε διαθέσιμη το 2015 θεωρούταν από τα καλύτερα framework που υπήρχαν για την PHP. Η τελευταία έκδοση (8.0) έγινε διαθέσιμη τον Σεπτέμβριο του 2020 [19].

4

Δεδομένα

4.1 Συγκέντρωση Δεδομένων

Το πρώτο βήμα που πρέπει να κάνουμε προκειμένου να δημιουργήσουμε ένα μοντέλο που να μπορεί να ταξινομεί εικόνες σε επικίνδυνες και μη είναι να συγκεντρώσουμε δεδομένα για να το εκπαιδεύσουμε. Τα δεδομένα αποτελούνται από εικόνες που περιλαμβάνουν καλοήθεις και κακοήθεις εικόνες από πολλές οπτικές γωνίες, διαφορές στο φωτισμό, στην ανάλυση κ.ά. Τα δεδομένα συγκεντρώθηκαν από ιστοσελίδες δερματολογίας όπως ο ISIC (https://www.isic-archive.com) και από ιστοσελίδες που διεξάγουν διαγωνισμούς μηχανικής μάθησης όπως το Kaggle (https://www.kaggle.com).

Όλα τα dataset που χρησιμοποιήσαμε χωρίζονται σε 2 σκέλη. Το πρώτο είναι ο φάκελος με τις εικόνες που είναι χωρισμένες είτε ανάλογα με την κλάση στην οποία ανήκουν (π.χ. κακοήθεις) είτε είναι όλες σε έναν φάκελο ανεξάρτητα με την κατηγορία. Το δεύτερο μέρος είναι συνήθως ένα αρχείο CSV (comma separated values) που περιέχει το όνομα της εικόνας και την κλάση της εικόνας. Σε μερικές περιπτώσεις -όπως στον ISIC 2020- το αρχείο αυτό περιέχει και επιπλέον πληροφορίες, όπως την τοποθεσία της δερματικής βλάβης επάνω στο σώμα. Ένα παράδειγμα τέτοιου αρχείου φαίνεται στην εικόνα 6.

```
image_name,patient_id,sex,age_approx,anatom_site_general_challenge,diagnosis,benign_malignant,target
ISIC_2637011,IP_7279968,male,45.0,head/neck,unknown,benign,0
ISIC_0015719,IP_3075186,female,45.0,upper extremity,unknown,benign,0
ISIC_0052212,IP_2842074,female,50.0,lower extremity,nevus,benign,0
ISIC_0068279,IP_6890425,female,45.0,head/neck,unknown,benign,0
ISIC_0074268,IP_8723313,female,55.0,upper extremity,unknown,benign,0
ISIC_0074311,IP_2950485,female,40.0,lower extremity,unknown,benign,0
ISIC_0074542,IP_4698288,male,25.0,lower extremity,unknown,benign,0
ISIC_0075663,IP_6017204,female,35.0,torso,unknown,benign,0
ISIC_0075914,IP_7622888,male,30.0,torso,unknown,benign,0
ISIC_0076262,IP_5075533,female,50.0,lower extremity,unknown,benign,0
ISIC_0076545,IP_9802602,male,55.0,upper extremity,unknown,benign,0
ISIC_0076742,IP_2318163,male,75.0,upper extremity,unknown,benign,0
ISIC_0076995,IP_2235340,female,55.0,torso,nevus,benign,0
ISIC_0077472,IP_3691360,female,40.0,torso,unknown,benign,0
ISIC_0077735,IP_1109756,male,70.0,torso,unknown,benign,0
ISIC_0078703,IP_7279968,male,45.0,torso,unknown,benign,0
ISIC_0078712,IP_2189124,male,40.0,lower extremity,unknown,benign,0
ISIC_0079038,IP_5295861,male,70.0,torso,unknown,benign,0
ISIC_0080512,IP_1870306,male,75.0,torso,unknown,benign,0
ISIC_0080752,IP_2613684,male,50.0,torso,unknown,benign,0
ISIC_0080817,IP_7318404,male,50.0,lower extremity,unknown,benign,0
ISIC_0081956,IP_2010919,female,50.0,upper extremity,unknown,benign,0
ISIC_0082348,IP_7684360,male,55.0,torso,unknown,benign,0
ISIC_0082543,IP_9463965,female,30.0,torso,unknown,benign,0
ISIC_0082934,IP_6572129,male,65.0,torso,unknown,benign,0
ISIC_0083035,IP_5805281,male,50.0,torso,unknown,benign,0
ISIC_0084086,IP_4023055,male,60.0,lower extremity,nevus,benign,0
ISIC_0084270,IP_2961528,male,40.0,lower extremity,nevus,benign,0
ISIC_0084395,IP_0175539,female,45.0,torso,nevus,benign,0
ISIC_0085172,IP_1705144,female,50.0,lower extremity,unknown,benign,0
```

Εικόνα 6 CSV αρχείο από τον ISIC 2020.

Για την εκπαίδευση και την αξιολόγηση του μοντέλου χρησιμοποιήθηκαν 4 dataset:

- ISIC-2019-2020 που περιλαμβάνει 32.542 φωτογραφίες καλοηθών και 584 κακοηθών αλλοιώσεων.
- Mnist-HAM που περιέχει 7.470 φωτογραφίες συνολικά από τις οποίες το 67% είναι καλοήθεις και το 11% αφορούν μελάνωμα. Οι υπόλοιπες αφορούν άλλες αλλοιώσεις, είτε επικίνδυνες είτε όχι για τον άνθρωπο. Σε αντίθεση με τα άλλα dataset, οι φωτογραφίες είναι χωρισμένες με την ονομασία της πάθησης και όχι με το αν είναι επικίνδυνες η όχι. Προκειμένου να τις χρησιμοποιήσουμε, μετατρέψαμε το αρχείο CSV σε καλοήθεις και κακοήθεις ανάλογα με την κατηγορία που ανήκουν.
- Benign vs Malignant είναι το τελευταίο dataset που χρησιμοποιήθηκε από το Kaggle. Ο λόγος που επιλέχθηκε είναι ότι -σε αντίθεση με τα υπόλοιπα- δεν περιέχει μόνο ιατρικές εικόνες άλλα και εικόνες τραβηγμένες από χρήστες που περιλαμβάνουν και άλλα χαρακτηριστικά και ιδιαιτερότητες, όπως background που ο αλγόριθμος θα πρέπει να μάθει να αγνοεί αφού τον προορίζουμε για χρήση μέσα από κινητά τηλέφωνα.

4.2 Ενίσχυση των δεδομένων

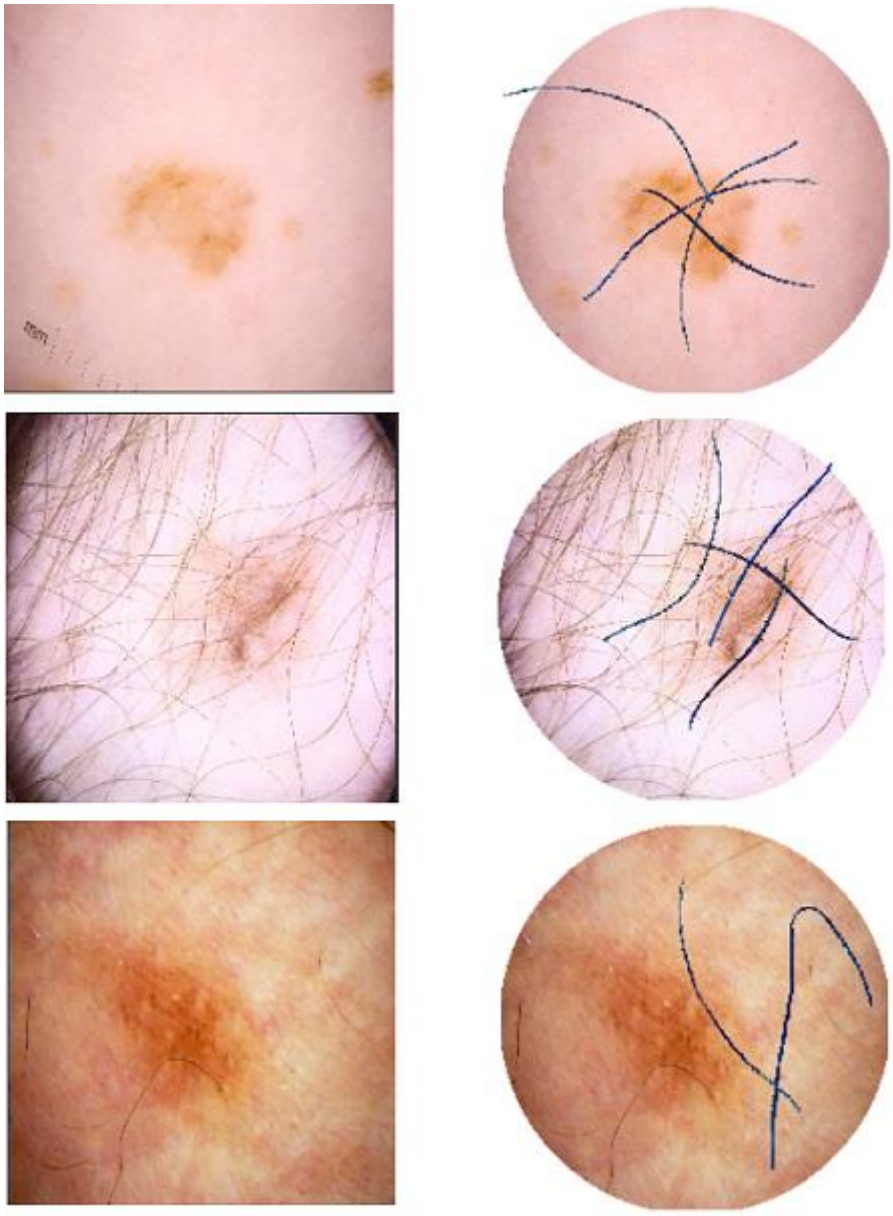
Όπως αναφέρθηκε νωρίτερα, τα δεδομένα που χρησιμοποιούμε αποτελούνται από κυρίως ιατρικές εικόνες αλλοιώσεων που σκοπεύουμε να δώσουμε στο νευρωνικό δίκτυο προκειμένου να εκπαιδευτεί. Στα μοντέλα νευρωνικών δικτύων είναι πολύ σημαντικό να αυξάνουμε την ποικιλότητα των δεδομένων μας, για να μπορούν να αναγνωρίσουν τα

χαρακτηριστικά που είναι σημαντικά στην ταξινόμηση. Κατά την εκπαίδευση του μοντέλου διακρίναμε τον παρακάτω συνδυασμό μεθόδων για την ενίσχυση του dataset μας [20].

Η πρώτη ενίσχυση που κάνουμε είναι μια τυχαία αυξομείωση σε γενικές ρυθμίσεις των εικόνων, όπως η φωτεινότητα, η αντίθεση, ο χρωματισμός και ο κορεσμός μιας εικόνας. Όλες αυτές οι ρυθμίσεις είναι σχετικές με το βασικό πρόβλημα που έχουμε να λύσουμε. Όλες οι εικόνες που θα δίνονται στο δίκτυο θα είναι τραβηγμένες από διαφορετικά μοντέλα κινητών τηλεφώνων και προσωπικών υπολογιστών, κάτω από διαφορετικές συνθήκες. Με αυτό τον τρόπο δημιουργούμε μια ποικιλία στα δεδομένα μας, ώστε να μην επηρεάζεται από αυτές τις παραμέτρους η διαδικασία της ταξινόμησης.

Η δεύτερη τροποποίηση που υλοποιήσαμε στα δεδομένα μας είναι η κυκλική κοπή των εικόνων. Όπως φαίνεται και στην εικόνα 7, η πλειοψηφία των δεδομένων προέρχονται από φωτογραφίες που έχουν τραβηχτεί από δερματολόγους οι οποίοι τείνουν να χρησιμοποιούν ένα είδος μικροσκοπίου με σκοπό να δώσουν έμφαση στο κέντρο της εικόνας και να εστιάσουν επάνω στη δερματική αλλοίωση. Έτσι λοιπόν, εφαρμόσαμε σε όλες τις εικόνες μια κυκλική περικοπή που προσομοιώνει αυτή των δερματολόγων. Επιπρόσθετα, η περικοπή αυτή δεν επηρεάζει την ποιότητα των εικόνων, αφού στις περισσότερες περιπτώσεις η δερματική αλλοίωση βρίσκεται στο κέντρο τους.

Η τελευταία τροποποίηση που εφαρμόστηκε στα δεδομένα μας είναι η προσομοίωση τριχών του δέρματος επάνω στις εικόνες. Οι δερματικές τρίχες είναι ένα βασικό πρόβλημα στον εντοπισμό του μελανώματος, λόγω του ότι πολλές φορές επικαλύπτουν την περιοχή που εξετάζεται. Επίσης, η αφαίρεση των τριχών θα προκαλούσε παραμόρφωση στην αλλοίωση με αποτέλεσμα την πιθανή λάθος ταξινόμηση σε κακοήθεις ή καλοήθεις. Επομένως, αντί για την αφαίρεση, αποφασίσαμε να προσθέσουμε έναν τυχαίο αριθμό (από καμία έως τέσσερις) τεχνητών τριχών επάνω στις εικόνες με σκοπό να προσομοιώσουμε το πρόβλημα και να μάθουμε στο μοντέλο να τις αγνοεί και να εκτελεί την ταξινόμηση χωρίς να επηρεάζεται από αυτές. Στην εικόνα 6 φαίνονται παραδείγματα από αυτές τις τεχνητές τρίχες, οι οποίες εσκεμμένα ποικίλουν σε χρώμα κατά το μήκος τους.



Εικόνα 7. Εικόνες πριν και μετά την τροποποίηση.

5

Ανάπτυξη Λογισμικού

Σε αυτό το κεφάλαιο θα περιγράψουμε την διαδικασία που ακολουθήσαμε για την κατασκευή του νευρωνικού δικτύου, το μοντέλο που επιλέχθηκε καθώς και το attention mechanism [21] που προσθέσαμε σε αυτό. Επίσης θα παρουσιάσουμε την διαδικασία εκπαίδευσης του καθώς και τις λεπτομέρειες των διαφόρων τεχνικών που ακολουθήθηκαν κατά την διάρκεια της ώστε να φτάσουμε στο καλύτερο δυνατό αποτέλεσμα.

5.1 Κατασκευή Νευρωνικού Δικτύου.

Το μοντέλο το οποίο χρησιμοποιήσαμε σαν βάση είναι το Efficient-Net στην έκδοση B0 που αποτελείται από 5,3 εκατομμύρια παραμέτρους και δέχεται σαν είσοδο εικόνες που ξεκινούν από ανάλυση 224x224. Η επιλογή του Efficinet-Net έγινε μετά την δοκιμή του συνόλου της βιβλιοθήκης του tensorflow στο ίδιο dataset και έπειτα από αξιολόγηση των αποτελεσμάτων. Ενδεικτικά τα μοντέλα που πλησίασαν τα αποτελέσματα του efficient net ή το ξεπέρασαν με ποσοστό ακρίβειας έως και 1.5% είναι τα εξής:

Μοντέλο	**Αριθμός Παραμέτρων**	**Χρόνος ανά βήμα εκπαίδευσης στην CPU (ms)**	**Μέγεθος (MB)**
InceptionResNetV2	55,873,736	130.19	215
NASNetLarge	88,949,818	344.51	343
VGG16	138,357,544	69.50	549
InceptionV3	23,851,784	42.25	92
EfficientNetB0	**5,330,571**	**46**	**29**

Όπως φαίνεται και στον παραπάνω πίνακα όλα τα μοντέλα που πλησιάζουν το efficientnet B0 είτε είναι πολύ μεγαλύτερα είτε πιο αργά κατά την διαδικασία της εκπαίδευσης με αποτέλεσμα να μην δικαιολογείται η επιλογή τους. Επίσης, μια μελλοντική επέκταση αυτής της διπλωματικής θα μπορούσε να είναι η εξαγωγή της διαδικτυακής εφαρμογής σε κανονική εφαρμογή που δεν χρειάζεται το internet για να εκτελέσει διαγνώσεις. Σε αυτή την περίπτωση μεγάλα δίκτυα που χρησιμοποιούν πολλούς πόρους από τον επεξεργαστή ή την μνήμη ενός κινητού τηλεφώνου θα ήταν απαγορευτικά.

Η αρχικοποίηση του efficientnet γίνεται πολύ εύκολα με την βιβλιοθήκη tensorflow με τις παρακάτω γραμμές:

```
base = EfficientNetB0(
    input_shape = (width, height, 3),
    weights = "imagenet",
    include_top = False,
    drop_connect_rate = 0.3
)
```

Σε αυτό το σημείο υποδεικνύουμε στην βιβλιοθήκη να αρχικοποιήσει το EfficientNet με έκδοση B0 με τις προκαθορισμένες διαστάσεις και να φορτώσει τα βάρη που δίνονται από την Google. Αξίζει να σημειωθεί ότι τα βάρη του Imagenet που φορτώνονται με την βιβλιοθήκη tensorflow είναι αρκετά παλιά, επομένως ακολουθήσαμε την διαδικασία που περιγράφεται στο documentation για την ενημέρωσή τους κατεβάζοντας τα απαραίτητα script από το repository της Google.

Το επόμενο βήμα είναι να κατασκευάσουμε το attention mechanism που αναφέρθηκε προηγουμένως με τον παρακάτω κώδικα:

```
# call so we can get the output shape
base(tf.keras.Input((width, height, 3)))
pt_depth = base.get_output_shape_at(0)[-1]
pt_features = base(inp)
bn_features = BatchNormalization()(pt_features)

### Attention Mechanism ###
attn_layer = Conv2D(64, kernel_size = (1, 1), padding = "same",
activation = "relu")(Dropout(0.35)(bn_features))
attn_layer = Conv2D(32, kernel_size = (1, 1), padding = "same",
activation = "relu")(attn_layer)
attn_layer = Conv2D(16, kernel_size = (1, 1), padding = "same",
activation = "relu")(attn_layer)
attn_layer = Conv2D(8, kernel_size = (1, 1), padding = "same",
activation = "relu")(attn_layer)
attn_layer = Conv2D(classes, kernel_size = (1, 1), padding = "valid",
activation = "sigmoid")(attn_layer)

mask_features = multiply([attn_layer, bn_features])
gap_features = GlobalAveragePooling2D()(mask_features)
gap_mask = GlobalAveragePooling2D()(attn_layer)
```

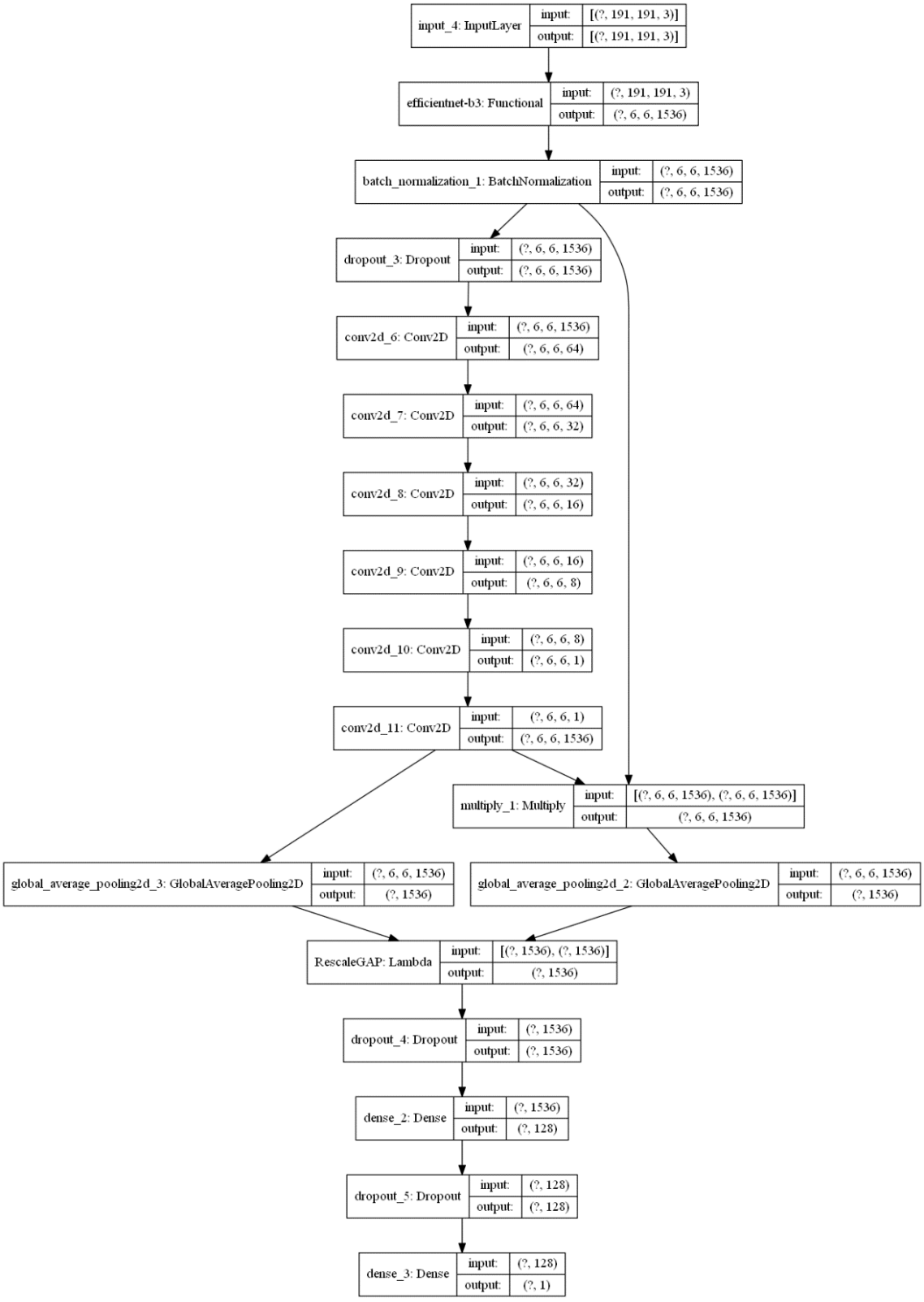
```
gap = Lambda(lambda x: x[0] / x[1], name =
"RescaleGAP")([gap_features, gap_mask])
gap_dr = Dropout(0.25)(gap)
dr_steps = Dropout(0.25)(Dense(128, activation = "relu")(gap_dr))

x = Dense(classes, activation = 'sigmoid')(dr_steps)
model = Model(inputs = inp, outputs = x)
model.compile(optimizer = Adam(learning_rate=1e-3), loss =
'binary_crossentropy',
                  metrics = ["accuracy"])

return model
```

Στο σημείο αυτό παίρνουμε την έξοδο του efficient net και την ενώνουμε με τον μηχανισμό προσοχής. Παρατηρούμε ότι όσο πιο βαθιά στα στρώματα αυτού του μηχανισμού πηγαίνουμε, τα φίλτρα του και εν συνεχεία το μέγεθός του μικραίνει με σκοπό να λειτουργεί σαν φίλτρο για τα σημαντικότερα χαρακτηριστικά που θέλουμε να αναγνωρίσει ο αλγόριθμος. Στη συνέχεια, ενώνουμε το attention mechanism με την έξοδο του efficient net με ένα custom layer που ονομάζουμε RescaleGAP (Global Average Pooling) για να επιτευχθεί αυτό χρησιμοποιούμε την κλάση lambda keras. Τέλος ο συνδυασμός των εξόδων ενώνονται με ένα τελευταίο στρώμα με σιγμοειδή ενεργοποίηση, που σημαίνει ότι οι τιμές που θα λαμβάνουμε είναι μεταξύ 0 και 1. Πριν επιστρέψουμε το μοντέλο πρέπει να καλέσουμε την μέθοδο compile που χτίζει τα στρώματα που έχουμε προσδιορίσει σε ένα νευρωνικό δίκτυο και χρησιμοποιούμε σαν optimizer τον Adam. Η τελευταία εντολή επιστρέφει το μοντέλο και τερματίζει την μέθοδο getModel, σε αυτό το σημείο το μοντέλο είναι έτοιμο για εκπαίδευση.

Στην εικόνα 8 με την βοήθεια της βιβλιοθήκης vis_utils μπορούμε να δούμε την τελική μορφή του μοντέλου. Μια σημαντική σημείωση για την βιβλιοθήκη είναι ότι το βασικό μοντέλο φαίνεται σαν ένα μοναδικό στρώμα το οποίο δεν είναι αληθές αλλά γίνεται για να δώσουμε έμφαση στα κομμάτια τα οποία δημιουργήσαμε εμείς και όχι στο efficient net το οποίο είναι μια κλασική υλοποίηση του μοντέλου.



Εικόνα 8. Η τελική μορφή του νευρωνικού δικτύου.

5.2 *Εκπαίδευση Δικτύου*

Αμέσως μετά την κατασκευή του μοντέλου, ακολουθεί η διαδικασία της εκπαίδευσης η οποία έχει πολλές παραμέτρους αλλά και ανάλογα με το dataset μας και τον εξοπλισμό που διαθέτουμε μπορεί να κρατήσει αρκετές ώρες. Για το μοντέλο που είδαμε στην προηγούμενη ενότητα ακολουθήσαμε την μέθοδο της εκπαίδευσης με επίβλεψη, που σημαίνει ότι μαζί με την φωτογραφία που δίναμε στο δίκτυο του παρείχαμε και την κλάση στην οποία ανήκει (καλοήθης ή κακοήθης).

Πολλές φορές κατά την εκπαίδευση στα δεδομένα μας υπάρχουν περιπτώσεις όπου εικόνες αντί να βοηθούν το μοντέλο να γενικεύει καλύτερα έχουν το αντίθετο αποτέλεσμα, αυτό μπορεί να συμβεί είτε γιατί κάποια φωτογραφία είναι καταταγμένη σε λάθος κλάση είτε γιατί πρόκειται για κάποια σπάνια περίπτωση π.χ. κακοήθης μελάνωμα που μιμείται καλοήθης δερματική αλλοίωση. Για να λύσουμε το συγκεκριμένο πρόβλημα και εφόσον έχουμε πολλές εικόνες διαθέσιμες χρησιμοποιήσαμε την μέθοδο 10-fold validation. Πρακτικά αυτό σημαίνει ότι χωρίσαμε τα δεδομένα μας σε 10 τυχαία κομμάτια και εκπαιδεύσαμε το μοντέλο σε όλα ξεχωριστά έτσι ώστε να δούμε πιο από αυτά έχει τα καλύτερα δυνατά αποτελέσματα. Στην βιβλιοθήκη sklrean αυτό επιτυγχάνεται πολύ εύκολα με τις παρακάτω γραμμές κώδικα:

```
from sklearn.model_selection import KFold
for fold,(train_index,test_index) in
enumerate(KFold(n_splits=FOLDS,shuffle=True,random_state=SEED).split(
train_df)):
```

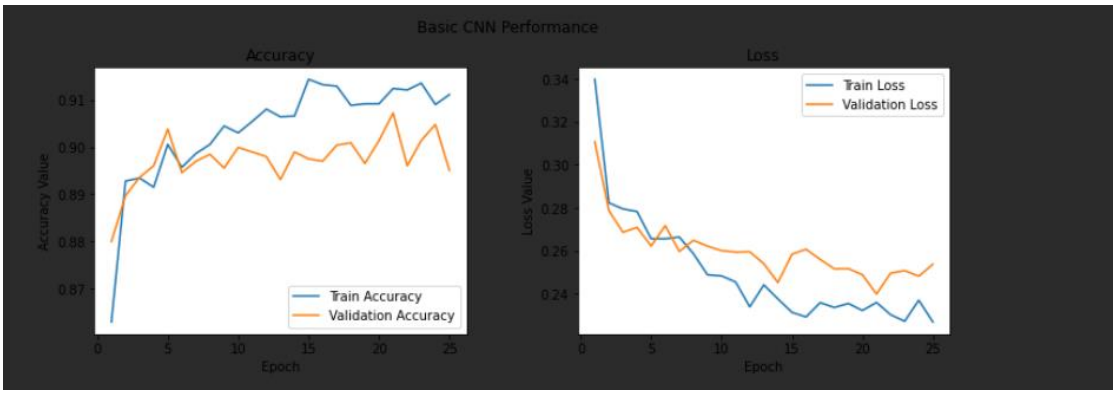
Οι μόνοι παράμετροι που έχουμε να ρυθμίσουμε είναι σε πόσα fold θέλουμε να χωρίσουμε το dataset μας και στη συνέχεια ξεκινάει μια επανάληψη μέσα στην οποία εκπαιδεύουμε το μοντέλο.

Μέσα σε αυτή την επανάληψη εκπαιδεύσαμε το δίκτυο σε κάθε εκδοχή του dataset για 25 εποχές. Αξίζει να σημειωθεί ότι το βασικό μοντέλο (efficient net) και τα στρώματα του ήταν παγωμένα κατά την διαδικασία της εκπαίδευσης και τα μόνα μέρη που εκπαιδεύονταν στην αρχική διαδικασία ήταν ο μηχανισμός προσοχής (attention mechanism) καθώς και το τελευταίο στρώμα που μας δίνει την κλάση της εικόνας. Το πάγωμα των στρωμάτων επιτυγχάνεται πολύ εύκολα με μια γραμμή στον κώδικα :

```
base.trainable = False
```

Με αυτό τον τρόπο ακολουθούμε την μεθοδολογία του transfer learning, εκμεταλλευόμαστε δηλαδή την εκπαίδευση που ήδη έχει γίνει στο μοντέλο στο Imagenet και το προσαρμόζουμε στο καινούριο dataset [23].

Η αξιολόγηση των 10 διαφορετικών εκδόσεων των μοντέλων έγινε στο ίδιο dataset που ήταν μια μικρή έκδοση και συνδυασμός των 4 dataset που αναφερθήκαμε στο προηγούμενο κεφάλαιο, με μεγαλύτερη έμφαση στο benign vs malignant το οποίο περιέχει περισσότερες εικόνες από τραβηγμένες από κινητά και το αποτέλεσμα φαίνεται στην εικόνα 9.



Εικόνα 9. Αποτελέσματα πρώτης φάσης εκπαίδευσης.

Παρατηρούμε ότι το μοντέλο έχει ξεπεράσει το 80% ακρίβειας που είχαμε βάλει σαν αρχικό στόχο από το πρώτο στάδιο που είναι ένα καλό αρχικό σημάδι.

Το επόμενο στάδιο της εκπαίδευσης είναι να συνεχίσουμε με την εκπαίδευση του μοντέλου αφού πρώτα επιλέξουμε αυτό που είχε τα καλύτερα αποτελέσματα. Το πρώτο μας βήμα είναι να φορτώσουμε τα βάρη του μοντέλου που αξιολογήσαμε ως καλύτερο. Η tensorflow έχει προκαθορισμένο σύστημα για αποθήκευση και επαναφορά βαρών ενός δικτύου έτσι ώστε να μπορούμε να συνεχίσουμε την εκπαίδευση είτε μετά από κάποια διακοπή είτε (όπως και στην δική μας περίπτωση) μετά την επιλογή του καλύτερου μοντέλου:

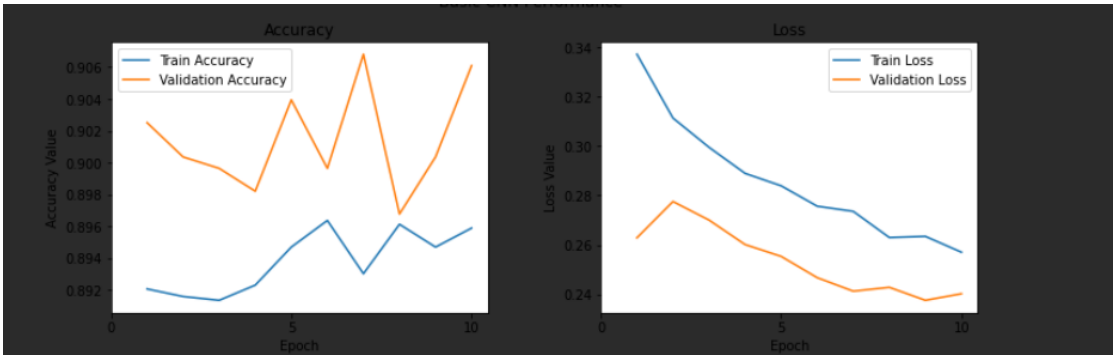
```
model.load_weights('best_model.h5')
```

Αφού επαναφέρουμε τα βάρη του μοντέλου που είχε την καλύτερη ακρίβεια στο dataset της αξιολόγησης, ξεκινάμε την διαδικασία fine tuning. Κατά την διάρκεια του fine tuning ξεπαγώνουμε τις συνδέσεις των νευρώνων (είτε όλου του δικτύου είτε μέρος του) και μειώνουμε κατά πολύ τον ρυθμό μάθησης του μοντέλου. Η διαδικασία αυτή έχει ως σκοπό την προσαρμογή του δικτύου και των βαρών του στα νέα δεδομένα χωρίς όμως να είναι αναγκαία και αποτελεσματική σε όλες τις περιπτώσεις [24]. Στην δική μας περίπτωση ξεπαγώσαμε τα πρώτα 20 στρώματα του βασικού μοντέλου (efficient net) αφήνοντας τα batch normalization layers παγωμένα και στην συνέχεια ξεπαγώσαμε ένα ολόκληρο block του efficient net (το block_4a)[2]. Η επιλογή του block_4a έγινε μετά από δοκιμή όλων των block και αξιολόγηση των αποτελεσμάτων.

```
for layer in model.layers[-20:]:
    if not isinstance(layer, BatchNormalization):
        layer.trainable = True
```

```
for layer in model.layers:
    if(blockFreeze in layer.name and not isinstance(layer,
BatchNormalization)):
        layer.trainable = True
```

Στο παραπάνω κομμάτι κώδικα βλέπουμε τον τρόπο με τον οποίο έγινε το ξεπάγωμα των στρωμάτων. Στην ουσία προσπελάσαμε όλα τα στρώματα του νευρωνικού δικτύου και ελέγχαμε αν δεν είναι Batch-Normalization layer και αν ανήκει στο block που μας ενδιαφέρει, σε αυτή την περίπτωση μαρκάραμε το στρώμα σαν έτοιμο για εκπαίδευση. Το fine tuning έγινε για 10 εποχές και τα αποτελέσματα αυτής της διαδικασίας φαίνονται στην εικόνα 10. Είχαμε μια μικρή αύξηση στην ακρίβεια του μοντέλου σε σχέση με το πρώτο βήμα της εκπαίδευσης.



Εικόνα 10. Αποτελέσματα του fine tuning.

5.3 *Κατασκευή Διαδικτυακής Εφαρμογής*

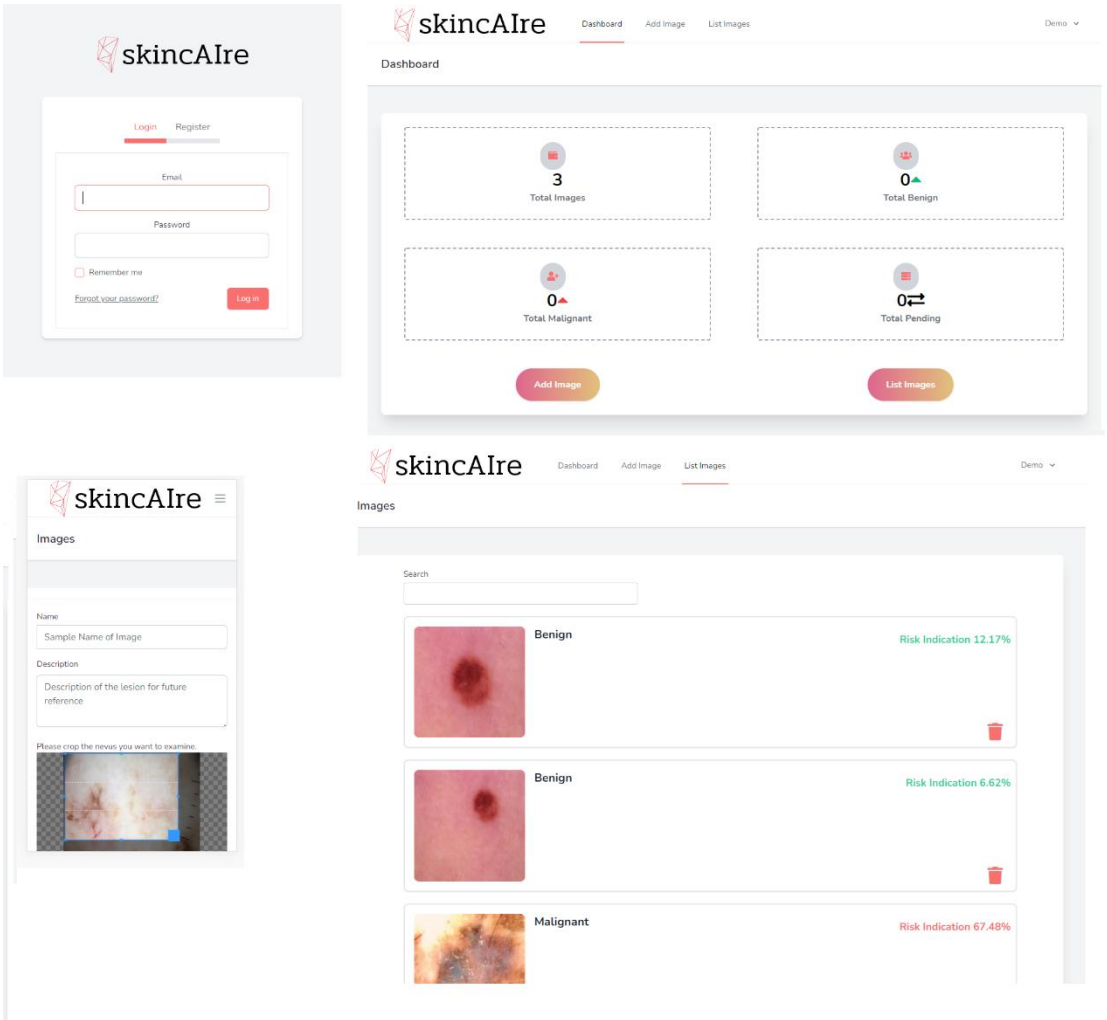
Μετά την κατασκευή του νευρωνικού δικτύου το επόμενο βήμα είναι η κατασκευή μιας εφαρμογής προκειμένου να δώσουμε πρόσβαση σε αυτό στους χρήστες που θέλουν να κάνουν τα δικά τους test. Για να επιτύχουμε αυτό το αποτέλεσμα αποφασίσαμε όπως προαναφέρθηκε την κατασκευή ιστοσελίδας, έτσι ώστε να έχουμε την μεγαλύτερη δυνατή κάλυψη συσκευών καθώς όλα τα smartphone και laptop έχουν υποστηριζόμενους από την εφαρμογή HTML 5 browsers.

Όπως περιγράψαμε σε προηγούμενο κεφάλαιο για την κατασκευή αυτής της εφαρμογής επιλέχτηκε η γλώσσα προγραμματισμού PHP σε Laravel framework με συνδυασμό livewire για το frontend. Ο κάθε χρήστης που ενδιαφέρεται μπορεί να κάνει εγγραφή στην ιστοσελίδα όπως φαίνεται και στην εικόνα 11 βάζοντας το όνομα του, το email του και τον κωδικό που επιθυμεί.

Μετά την εγγραφή ο χρήστης μεταφέρεται στην αρχική σελίδα της εφαρμογής όπου μπορεί να δει τα συνολικά στατιστικά του, όπως πόσες εικόνες έχει ανεβάσει για να ελεγχθούν,

πόσες είναι καλοήθης και πόσες κακοήθης και από πόσες περιμένει αποτελέσματα. Στην ίδια σελίδα ο χρήστης έχει 2 επιλογές την λίστα εικόνων όπου μπορεί να δει παλιές διαγνώσεις που έχουν γίνει από το μοντέλο και την δυνατότητα να ανεβάσει μια καινούρια εικόνα.

Στην προσθήκη εικόνας ο χρήστης μπορεί εύκολα να ανεβάσει μια υπάρχουσα είτε να τραβήξει μια καινούρια από το κινητό του ή τον προσωπικό του υπολογιστή. Αφού επιλέξει την εικόνα που θέλει να πάρει αποτελέσματα δίνεται η δυνατότητα στον χρήστη να κάνει περικοπή της περιοχής που θέλει να διαγνωσθεί. Αυτό γίνεται γιατί τα περισσότερα σύγχρονα κινητά τηλέφωνα έχουν κάμερες με πολλά megapixel και είναι δύσκολο να εστιάσουν σε μια πολύ κοντινή περιοχή. Στη συνέχεια ο χρήστης συμπληρώνει το όνομα καθώς και προαιρετικά την περιγραφή της εικόνας και πατάει αποθήκευση. Τέλος ο χρήστης μεταφέρεται στην σελίδα με όλες τις εικόνες που έχει ανεβάσει και περιμένει τα αποτελέσματα τα οποία εμφανίζονται μετά από μερικά δευτερόλεπτα όπως φαίνεται και στην εικόνα 11.



Εικόνα 11. Από αριστερά προς τα δεξιά: α) Σελίδα εγγραφής β) Κύρια Σελίδα Χρήστη γ) Σελίδα Προσθήκης εικόνας δ) Σελίδα Αποτελεσμάτων.

6

Αξιολόγηση

Όπως αναφέρθηκε σε προηγούμενα κεφάλαια, στόχος του μοντέλου είναι να δώσει στο ευρύ κοινό τη δυνατότητα να εξετάζεται γρήγορα και εύκολα με μοναδικό εργαλείο το κινητό του. Έτσι, για την αξιολόγηση του νευρωνικού δικτύου θέσαμε σαν όριο να έχουμε ίδια ή καλύτερη ακρίβεια από τον μέσο όρο των δερματολόγων κατά την εξέταση για αναγνώριση κακοηθειών (80%). Για την αξιολόγηση του αλγορίθμου χρησιμοποιήθηκε το 20% από τα dataset ISIC-2020, MNIST-HAM, benign vs malignant που δεν χρησιμοποιήθηκαν κατά την διάρκεια της εκπαίδευσης. Στο σύνολο χρησιμοποιήσαμε 8.000 φωτογραφίες από τα 3 dataset κατά την διάρκεια της αξιολόγησης. Τέλος, λόγω της φύσης του προβλήματος, δίνουμε παραπάνω σημασία στη μείωση των ψευδώς αρνητικών αποτελεσμάτων που θα ήταν και τα πιο επιζήμια για τον ασθενή.

6.1 Παράμετροι αξιολόγησης

Η πρώτη παράμετρος που χρησιμοποιήθηκε είναι η ακρίβεια (accuracy) του μοντέλου. Η ακρίβεια μετριέται από όλα τα αληθινά -θετικά και αρνητικά- αποτελέσματα που ταξινόμησε το μοντέλο και προς όλες τις εικόνες που εξέτασε.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Instances}}$$

Η ακρίβεια αν και μια καλή πρώτη ένδειξη για την απόδοση του μοντέλου έχει ένα βασικό πρόβλημα. Σε dataset όπως το ISIC-2020 όπου οι καλοήθειες είναι πολύ μεγαλύτερες σε αριθμό από τις κακοήθειες (32.542 έναντι 584) το μοντέλο μπορεί να δείξει μια ακρίβεια της τάξης του 98% ταξινομώντας όλες τις εικόνες σαν καλοήθεις. Επίσης, σε προβλήματα όπου θέλουμε να ταξινομήσουμε ή να διαγνώσουμε ασθένειες, μας ενδιαφέρει περισσότερο να μην έχουμε

ψευδή αρνητικά αποτελέσματα, να ταξινομήσουμε δηλαδή μια δερματική αλλοίωση σαν καλοήθη ενώ πρόκειται για μελάνωμα.

Η δεύτερη παράμετρος αξιολόγησης που χρησιμοποιήσαμε είναι η ευαισθησία (sensitivity), η οποία μετράει πόσο αποτελεσματικό είναι το μοντέλο στο να εντοπίζει κακοήθειες, δηλαδή τα πραγματικά θετικά.

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Η ευαισθησία είναι μια πολύ σημαντική παράμετρος αξιολόγησης για το πρόβλημά μας, καθώς είναι ανεξάρτητη από τον λόγο θετικών/αρνητικών του dataset.

Η επόμενη παράμετρος που χρησιμοποιήθηκε είναι η ιδιαιτερότητα (specificity), όπου μετράει τα ψευδώς θετικά ταξινομημένα αποτελέσματα.

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{Fulse Positives} + \text{True Negatives}}$$

Η ιδιαιτερότητα, όπως και η ευαισθησία, δεν επηρεάζονται από μη ισορροπημένα δεδομένα μεταξύ των κλάσεων που εξετάζουμε.

Η τελευταία παράμετρος που χρησιμοποιήσαμε είναι η καμπύλη ROC (Receiver Operating Characteristic). Η ROC είναι μια γραμμική παράσταση που μας δείχνει την συσχέτιση μεταξύ ιδιαιτερότητας και ευαισθησίας. Η γραμμική παράσταση σχεδιάζεται χρησιμοποιώντας τα πραγματικά θετικά σε σχέση με ψευδώς θετικά. Η περιοχή κάτω από την γραμμική παράσταση (AUC, Area Under the Curve) χρησιμοποιείται πολύ συχνά για την αξιολόγηση δυαδικών μοντέλων, όπως το δικό μας, που προορίζεται να προβλέπει μεταξύ καλοήθειας και κακοήθειας. Οι τιμές του AUC μετριούνται μεταξύ του 0.5 έως του 1, με το 1 να αποτελεί τον τέλειο ταξινομητή ενώ το 0.5 έναν τελείως τυχαίο αλγόριθμο.

6.2 Οργάνωση πειραμάτων

Για να αξιολογήσουμε τα αποτελέσματα του νευρωνικού δικτύου, χρησιμοποιήσαμε εργαλεία που δίνονται από τη βιβλιοθήκη sklearn, καθώς και τις παραμέτρους που αναφέρθηκαν προηγουμένως. Αρχικά, από την διαδικασία της εκπαίδευσης χωρίσαμε τα δεδομένα μας σε «*εκπαίδευσης*» και «*αξιολόγησης*», έτσι ώστε να έχουμε ένα αρκετά μεγάλο dataset στο οποίο θα μπορούσαμε να αξιολογήσουμε αξιόπιστα τα αποτελέσματα. Ο διαχωρισμός του dataset έγινε με τον παρακάτω κώδικα στα πρώτα στάδια της εκπαίδευσης του αλγορίθμου.

```
from sklearn.model_selection import train_test_split
train_df, test_df = train_test_split(df, test_size=0.2)
```

Όπως φαίνεται και στο παρακάτω στιγμιότυπο, από την εκτέλεση του αλγορίθμου η αξιολόγηση των δεδομένων έγινε σε 8197 εικόνες στο σύνολο, από τις οποίες οι 6500 ήταν καλοήθης ενώ οι 1697 ήταν κακοήθης.

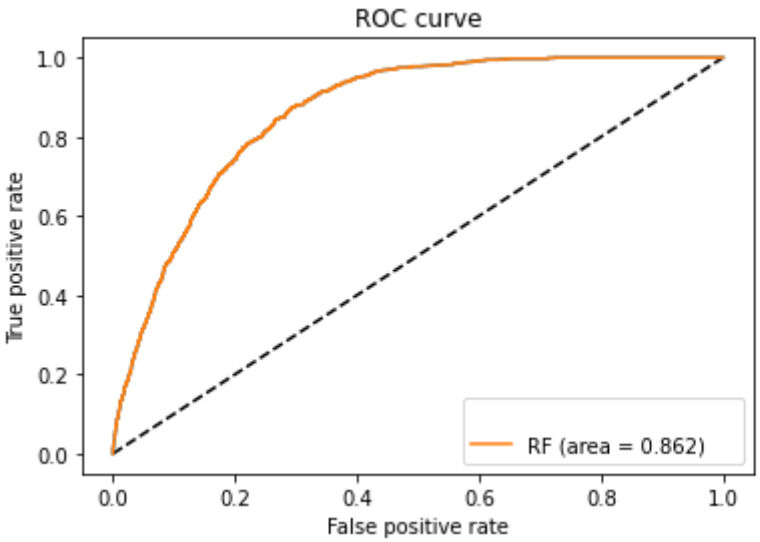
```
Name: benign_malignant, dtype: int64
benign       6500
malignant    1697
```

Η διαδικασία της αξιολόγησης διήρκησε περίπου 20 λεπτά, διότι χρησιμοποιήσαμε την κάρτα γραφικών του υπολογιστή που έτρεχε το νευρωνικό δίκτυο (Nvidia GTX 1660 TI) .

6.3 Αποτελέσματα

Τα αποτελέσματα του δικτύου μας είναι αρκετά ικανοποιητικά, καθώς στο dataset αξιολόγησης πέτυχε 91,22% ακρίβεια. Στον παρακάτω πίνακα φαίνονται και οι τιμές των υπόλοιπων παραμέτρων που μας δίνουν καλύτερη εικόνα για την συνολική απόδοση του αλγορίθμου, όπως και η γραφική παράσταση του AUC.

Παράμετρος	**Ποσοστό**
Specificity	87.34%
Sensitivity	90.79%
Accuracy	91.22 %
ROC/AUC	86.24 %



Εικόνα 12. Γραμμική παράσταση ROC

6.4 Σύνοψη συμπερασμάτων αξιολόγησης

Από την παραπάνω διαδικασία αξιολόγησης συνοψίζουμε ότι το νευρωνικό δίκτυο επιτυγχάνει το στόχο του 80% ακρίβειας και φαίνεται να τον ξεπερνάει στο dataset που αξιολογήθηκε. Σαν τελικό αποτέλεσμα και σημείο αναφοράς θα κρατήσουμε το 86% AUC, το οποίο σε θέματα διάγνωσης -όπως και στο δικό μας- είναι η πιο σημαντική μετρική για τον μέσο όρο ακρίβειας του μοντέλου, Επίσης, σημαντικό κρίνουμε και το 87% της ιδιαιτερότητας για τα πραγματικά αρνητικά (διαγνώσεις υγειούς δερματικής αλλοίωσης).

7

Τεχνικές λεπτομέρειες

Σε αυτό το κεφάλαιο θα αναφέρουμε τις τεχνικές λεπτομέρειες που αφορούν την εν λόγω διπλωματική εργασία, είτε αυτές αναφέρονται στην ανάπτυξη του νευρωνικού δικτύου, είτε στην web εφαρμογή που το συνοδεύει. Θα γίνει επίσης αναφορά στη γέφυρα που συνδέει τα 2 τμήματα μεταξύ τους, αλλά και στα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξή τους.

7.1 Δημιουργία Μοντέλου με Keras/Tensorflow

Ένα από τα βασικότερα κομμάτια κώδικα της εργασίας είναι η function που ονομάζεται getModel. Είναι η υπορουτίνα που είναι υπεύθυνη για τη δημιουργία του μοντέλου και έχει σαν παραμέτρους την ανάλυση των εικόνων που δέχεται σαν είσοδο (224x224 στην δική μας περίπτωση). Η ίδια function είναι υπεύθυνη για τη δημιουργία του attention mechanism που προσθέτει στο βασικό μοντέλο που επιλέξαμε (efficient net). Η function ξεκινάει με την αρχικοποίηση του βασικού μοντέλου και το πάγωμα των στρωμάτων του. Στη συνέχεια, δημιουργούμε τον μηχανισμό προσοχής και τον προσκολλάμε στο βασικό μοντέλο. Τέλος, πριν επιστρέψουμε το νευρωνικό δίκτυο που δημιουργήσαμε, εμφανίζουμε στον χρήστη μια σύνοψη που περιλαμβάνει τον αριθμό παραμέτρων του μοντέλου, καθώς και τις εισόδους και εξόδους κάθε στρώματός του.

```
def getModel(width,height):

    # Input
    inp = Input(shape = (width, height, 3))
    # Base EfficientNet pretrained model
    base = EfficientNetB0(
        input_shape = (width, height, 3),
        weights = "imagenet",
        include_top = False,
        drop_connect_rate = 0.3
    )
    base.trainable = False

    # call so we can get the output shape
```

```
    base(tf.keras.Input((width, height, 3)))
    pt_depth = base.get_output_shape_at(0)[-1]
    pt_features = base(inp)
    bn_features = BatchNormalization()(pt_features)

    ### Attention Mechanism ###
    attn_layer = Conv2D(64, kernel_size = (1, 1), padding = "same",
activation = "relu")(Dropout(0.35)(bn_features))
    attn_layer = Conv2D(32, kernel_size = (1, 1), padding = "same",
activation = "relu")(attn_layer)
    attn_layer = Conv2D(16, kernel_size = (1, 1), padding = "same",
activation = "relu")(attn_layer)
    attn_layer = Conv2D(8, kernel_size = (1, 1), padding = "same",
activation = "relu")(attn_layer)
    attn_layer = Conv2D(1, kernel_size = (1, 1), padding = "valid",
activation = "sigmoid")(attn_layer)

    # Fan it out to all of the channels
    up_c2_w = np.ones((1, 1, classes, pt_depth))
    up_c2 = Conv2D(
        pt_depth, kernel_size = (1, 1),
        padding = "same",
        activation = "linear",
        use_bias = False,
        weights = [up_c2_w]
    )
    up_c2.trainable = False
    attn_layer = up_c2(attn_layer)

    mask_features = multiply([attn_layer, bn_features])
    gap_features = GlobalAveragePooling2D()(mask_features)
    gap_mask = GlobalAveragePooling2D()(attn_layer)

    # To account for missing values from the attention model
    gap = Lambda(lambda x: x[0] / x[1], name =
"RescaleGAP")([gap_features, gap_mask])
    gap_dr = Dropout(0.25)(gap)
    dr_steps = Dropout(0.25)(Dense(128, activation = "relu")(gap_dr))

    x = Dense(1, activation = 'sigmoid')(dr_steps)
    model = Model(inputs = inp, outputs = x)
    model.compile(optimizer = Adam(learning_rate=1e-3), loss =
'binary_crossentropy',
                  metrics = ["accuracy"])

    # from keras.utils.vis_utils import plot_model
    # plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
    # os.System.exit(1)
    model.summary()
    return model
```

7.2 Dataset Augmentation

Η δεύτερη υπορουτίνα που θα πρέπει να αναφέρουμε είναι η preprocess η οποία είναι υπεύθυνη για την ενίσχυση των δεδομένων μας, ένα πολύ σημαντικό κομμάτι στην εκπαίδευση του νευρωνικού δικτύου. Η ανάλυση της θα γίνει σε 2 μέρη το πρώτο είναι της βασικής

υπορουτίνας που είναι υπεύθυνη για το σύνολο τον τροποποιήσεων της εικόνας και το δεύτερο είναι της κλάσης που είναι υπεύθυνη για την προσθήκη τυχαίων τριχών στις εισόδους.

7.2.1 Function Preprocess

Η function που ονομάζεται preprocess εκτελείται σε κάθε εικόνα πριν εισέλθει στο νευρωνικό δίκτυο. Αυτό επιτυγχάνεται εύκολα δηλώνοντάς την στην κλάση ImageDatagenerator του Keras.

```
image_generator_train = ImageDataGenerator(
    rescale=1/255,
    preprocessing_function=preprocess,
    horizontal_flip=True,
)
```

Στη συνέχεια, όπως φαίνεται και στο παρακάτω μέρος του κώδικα, το πρώτο βήμα είναι να κάνουμε γενικές τροποποιήσεις στην εικόνα, όπως να αυξομειώσουμε την φωτεινότητα, την αντίθεση ή και τα χρώματα της εικόνας. Έπειτα, εκτελούμε την κυκλική περικοπή για να προσομοιώσουμε τις φωτογραφίες που τραβάνε συνήθως οι δερματολόγοι με το δερματοσκόπιο. Η τελευταία τροποποίηση που κάνουμε στην εικόνα πριν την επιστρέψουμε για να εισαχθεί είναι η προσθήκη τυχαίου αριθμού τριχών με την κλάση AdvancedHairAugmentation που θα δούμε λεπτομερώς στην επόμενη υποενότητα.

```
def preprocess(img):
    img = tf.image.random_saturation(img, 0.7, 1.3)
    img = tf.image.random_contrast(img, 0.9, 1.1)
    img = tf.image.random_brightness(img, 0.1)
    img = tf.image.random_hue(img, 0.01)

    # # circle cropping
    img = array_to_img(img)
    h,w=img.size
    alpha = Img.new('L', img.size,0)
    draw = ImageDraw.Draw(alpha)
    draw.pieslice([0,0,h,w],0,360,fill=255)
    npAlpha=np.array(alpha)
    npImage=np.dstack((img,npAlpha))
    npImage = array_to_img(npImage)
    background = Img.new("RGB", img.size, (255, 255, 255))
    background.paste(npImage, mask=npImage.split()[3]) # 3 is the
alpha channel
    img = array_to_img(background).convert('RGB')
    # end circle cropping

    img = hairClass.apply(np.array(img))

    return  np.array(img).astype('float64')
```

7.2.2 Advanced Hair Augmentation

Ένα ακόμη σημαντικό κομμάτι κώδικα είναι η κλάση AdvancedHairAugmentation η οποία είναι υπεύθυνη για την προσθήκη τυχαίου αριθμού προσομοιωμένων τριχών επάνω στην εικόνα εισόδου. Όπως φαίνεται στο παρακάτω κομμάτι κώδικα, η κλάση αρχικοποιείται με 2 παραμέτρους, τον μέγιστο αριθμό τριχών που θέλουμε να προσθέσουμε και τον φάκελο με τις εικόνες που βρίσκονται οι προσομοιωμένες εικόνες. Στην πορεία, με την βοήθεια της random αποφασίζουμε τον αριθμό που θέλουμε να επαναληφθεί η διαδικασία. Ο αλγόριθμος βρίσκει ένα τυχαίο σημείο επάνω στην εικόνα και κάνει τυχαίες περιστροφές στην τρίχα έτσι ώστε να έχουμε ένα σχετικά μοναδικό αποτέλεσμα κάθε φορά. Τέλος, η υπορουτίνα apply επιστρέφει την εικόνα.

```
class AdvancedHairAugmentation:
    def __init__(self, hairs: int = 4, hairs_folder: str = ""):
        self.hairs = hairs
        self.hairs_folder = hairs_folder
        self.hair_images = [im for im in
os.listdir(self.hairs_folder) if 'png' in im]

    def apply(self, img):
        n_hairs = random.randint(1, self.hairs)

        if not n_hairs:
            return img

        height, width, _ = img.shape  # target image width and height

        for _ in range(n_hairs):
            hair = cv2.imread(os.path.join(self.hairs_folder,
random.choice(self.hair_images)))
            hair = cv2.flip(hair, random.choice([-1, 0, 1]))
            hair = cv2.rotate(hair, random.choice([0, 1, 2]))

            h_height, h_width, _ = hair.shape  # hair image width and
height
            roi_ho = random.randint(0, img.shape[0] - hair.shape[0])
            roi_wo = random.randint(0, img.shape[1] - hair.shape[1])
            roi = img[roi_ho:roi_ho + h_height, roi_wo:roi_wo +
h_width]

            img2gray = cv2.cvtColor(hair, cv2.COLOR_BGR2GRAY)
            ret, mask = cv2.threshold(img2gray, 10, 255,
cv2.THRESH_BINARY)
            mask_inv = cv2.bitwise_not(mask)
            img_bg = cv2.bitwise_and(roi, roi, mask=mask_inv)
            hair_fg = cv2.bitwise_and(hair, hair, mask=mask)

            dst = cv2.add(img_bg, hair_fg)
            img[roi_ho:roi_ho + h_height, roi_wo:roi_wo + h_width] =
dst

        return img
```

7.3 Γραμμικές Παραστάσεις

Κατά τη διάρκεια της εκπαίδευσης του μοντέλου αλλά και της αξιολόγησής του είναι πολύ σημαντικό να κατανοούμε τα νούμερα που μας επιστρέφονται από την βιβλιοθήκη keras προκειμένου να παίρνουμε αποφάσεις για τα επόμενα βήματα που θα ακολουθήσουμε. Πολλές φορές αυτό επιτυγχάνεται με γραμμικές παραστάσεις. Έτσι όπως φαίνεται και στο κομμάτι κώδικα που ακολουθεί αναπτύξαμε την plotAndSaveHistory function, όπου ο ρόλος της είναι να δέχεται σαν παράμετρο το ιστορικό εκπαίδευσης του νευρωνικού δικτύου και να παράγει την γραμμική παράσταση για 2 σημαντικές μετρικές, την ακρίβεια και το λάθος. Το αποτέλεσμα της υπορουτίνας φαίνεται στις εικόνες 10 και 11 όπου είναι αποτελέσματα της εκπαίδευσης και του fine tuning του μοντέλου.

```
def plotAndSaveHistory(history):
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
    t = f.suptitle('Basic CNN Performance', fontsize=12)
    f.subplots_adjust(top=0.85, wspace=0.3)

    epoch_list = list(range(1,len(history.epoch)+1))

    ax1.plot(epoch_list, history.history['accuracy'], label='Train
Accuracy')
    ax1.plot(epoch_list, history.history['val_accuracy'],
label='Validation Accuracy')
    ax1.set_xticks(np.arange(0, len(history.epoch)+1, 5))
    ax1.set_ylabel('Accuracy Value')
    ax1.set_xlabel('Epoch')
    ax1.set_title('Accuracy')
    l1 = ax1.legend(loc="best")

    ax2.plot(epoch_list, history.history['loss'], label='Train Loss')
    ax2.plot(epoch_list, history.history['val_loss'],
label='Validation Loss')
    ax2.set_xticks(np.arange(0, len(history.epoch)+1, 5))
    ax2.set_ylabel('Loss Value')
    ax2.set_xlabel('Epoch')
    ax2.set_title('Loss')
    l2 = ax2.legend(loc="best")

    plt.savefig('plot_fold_.png')
    return
```

Η δεύτερη γραμμική παράσταση που χρησιμοποιήθηκε είναι η ROC/AUC κατά τη διάρκεια της αξιολόγησης του μοντέλου. Η function αυτή λειτουργεί με τον ίδιο τρόπο που λειτουργούσε η προηγούμενη, με μοναδική διαφορά ότι σαν παραμέτρους δέχεται το AUC score του μοντέλου, τα αληθινά θετικά αποτελέσματα και τα ψευδώς θετικά. Το αποτέλεσμα αυτού του κώδικα φαίνεται στην εικόνα 12.

```
def plotROCCurve(auc_keras,nn_fpr_keras,nn_tpr_keras):
    plt.figure(1)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(nn_fpr_keras, nn_tpr_keras, label='Keras (area =
```

```
{:.3f})'.format(auc_keras))
    plt.plot(nn_fpr_keras, nn_tpr_keras, label='RF (area =
{:.3f})'.format(auc_keras))
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.title('ROC curve')
    plt.legend(loc='best')
    plt.show()
    return plt
```

7.4 *Web Εφαρμογή*

Όπως προαναφέρθηκε, μαζί με το νευρωνικό δίκτυο αναπτύχθηκε και μια web εφαρμογή για να δοθεί η δυνατότητα σε όσους το επιθυμούν να ελέγξουν μια δερματική αλλοίωση που τους ανησυχεί, εύκολα και γρήγορα με το κινητό τους τηλέφωνο. Για τον σκοπό αυτό χρησιμοποιήθηκε το framework Laravel, ενώ ακολουθήσαμε την component based λογική για την ανάπτυξή της με το πακέτο livewire. Σε αυτό το κεφάλαιο ακολουθεί μια ανάλυση του κώδικα για τις πιο σημαντικές σελίδες της εφαρμογής.

7.4.1 *Αρχική Σελίδα Χρήστη*

Η αρχική σελίδα του χρήστη είναι μία από τις πιο απλές της εφαρμογής. Ο σκοπός του κώδικα που βλέπουμε παρακάτω είναι να μαζέψει πληροφορίες για την κατάσταση του χρήστη, όπως τον αριθμό των εικόνων που περιμένει διάγνωση, τις συνολικές εικόνες καθώς και τον αριθμό καλοήθων και κακοήθων εικόνων που έχει ανεβάσει στην εφαρμογή.

```
//Get user stats for dashboard home page
$data['total'] = Image::where('user_id',auth()->user()->id)->count();

$data['totalBenign'] = Image::where('user_id',auth()->user()->id)
                            ->where('benign_malignant','<','0.50')
                            ->where('status','finishes')->count();

$data['totalMalignant'] = Image::where('user_id',auth()->user()->id)
                            ->where('benign_malignant','>=','0.50')
                            ->where('status','finishes')->count();

$data['totalPending'] = Image::where('user_id',auth()->user()->id)
    ->where('status','uploaded')->count();

return view('dashboard.home',['data' => $data]);
```

7.4.2 *Σελίδα Προθήκης Εικόνας*

Μια από τις πιο σημαντικές σελίδες είναι η προσθήκη νέας εικόνας. Η σελίδα αυτή δίνει την δυνατότητα στον χρήστη να ανεβάσει μια καινούρια εικόνα και, αφού προσθέσει το όνομά της και προαιρετικά μια περιγραφή, μπορεί να την αποθηκεύσει ώστε να μπει στη σειρά για

διάγνωση. Ο κώδικας που φαίνεται παρακάτω είναι το μέρος που τρέχει στο server και όχι στη συσκευή του χρήστη. Αποτελείται από 3 υπορουτίνες, η updated είναι εκεί για να ενημερώνει το frontend κομμάτι της σελίδας (αυτό που εκτελείται δηλαδή στη συσκευή του χρήστη) όταν ο χρήστης ενημερώνει την εικόνα. Η submit εκτελείται όταν ο χρήστης συμπληρώσει όλα τα πεδία της φόρμας και πατήσει *αποθήκευση*. Η υπορουτίνα αυτή δημιουργεί μια νέα εικόνα και την αποθηκεύει στη βάση δεδομένων, ενώ στη συνέχεια κάνει την περικοπή της εικόνας με βάση την επιλογή του χρήστη. Η κομμένη εικόνα αποθηκεύεται στο server και στη συνέχεια εμφανίζεται στον χρήστη στη σελίδα με όλες τις εικόνες που έχει ανεβάσει στο παρελθόν.

```
class AddImage extends Component
{
    use WithFileUploads;

    public $photo;
    public $name;
    public $description;
    public $bbox;

    protected $rules = [
        'name' => 'required|min:5',
        'description' => 'nullable|max:255',
        'photo' => 'image|max:10000',
    ];

    public function updated($name,$value)
    {
        //If the image is changed during cropping, emit event to
inform cropperjs
        if($name == 'photo')
            $this->emitSelf('updatedPhoto',$value->temporaryUrl());
    }

    public function submit()
    {
        $this->validate();

        //Create new image model and fill info
        $image = new Image();
        $image->name = $this->name;
        $image->description = $this->description;
        $image->original_name = $this->photo-
>getClientOriginalName();
        $image->status = 'uploaded';
        $image->file_path = $this->photo->store('photos');

        //Disable orientation and cropping during unit tests (open
issue with ImageIntervention)
        if(!App::runningUnitTests() && (int)$this->bbox['width'] != 0
&& (int)$this->bbox['height'] != 0) {
            //Orientate image if exif exists, crop it based on user
selection and save it
            $interventionImage =
ImageIntervention::make(storage_path('app/'.$image->file_path))-
>setFileInfoFromPath(storage_path('app/'.$image->file_path));
            $interventionImage->orientate()->crop((int)$this-
>bbox['width'], (int)$this->bbox['height'], (int)$this->bbox['x'],
```

```
(int)$this->bbox['y']);
            $interventionImage->save(storage_path('app/'.$image-
>file_path),100);
        }

        //TODO decide if guests will be able to use service.
        $image->user_id = Auth::user()->id;

        $image->save();

        return redirect()->to('/imageList');
    }

    public function render()
    {
        return view('components.images.add-image');
    }
}
```

7.4.2.1 *Frontend Προσθήκης Εικόνας*

Ένα ακόμα σημαντικό κομμάτι της σελίδας αυτής είναι το frontend το οποίο εκτελείται στη συσκευή του χρήστη και όχι στον server. Το κομμάτι κώδικα που παραθέτουμε παρακάτω είναι γραμμένο σε γλώσσα Javascript και χρησιμοποιούμε την AlpineJs σαν framework, η οποία έρχεται συνδεδεμένη με την livewire. Ο συγκεκριμένος κώδικας είναι υπεύθυνος για την επιλογή του κομματιού της εικόνας που θέλει ο χρήστης να εξεταστεί από το νευρωνικό δίκτυο ή αλλιώς την διαδικασία crop. Για να το πετύχουμε αυτό χρησιμοποιούμε την βιβλιοθήκη cropperjs, η οποία μας επιστρέφει συντεταγμένες ενός ορθογωνίου που είναι σχεδιασμένο επάνω στην εικόνα. Οι συντεταγμένες αυτές αποθηκεύονται στη μεταβλητή bbox, την οποία έχουμε μαρκάρει με τη λέξη entangle που μας επιτρέπει να στέλνουμε τις τιμές στον server κάθε φορά που αλλάζουν στη συσκευή του χρήστη.

```
<script>
    function cropper()
    {
        return {
            bbox: @entangle('bbox'),
            cropperInstance: null,
            updateBbox() {
                let data = this.cropperInstance.getData()
                this.bbox= {
                    x: data['x'],
                    y: data['y'],
                    width: data['width'],
                    height: data['height'],
                }
                console.log('new bbox')

console.log(data['x'],data['y'],data['width'],data['height'])
            },
            initCroperjs() {
                let self = this;
                const image = document.getElementById('img-preview');
                this.cropperInstance = new Cropper(image, {
                    viewMode:2,
```

```
                    rotatable: true,
                    minCropBoxWidth:24,
                    minCropBoxHeight:24,
                    zoomable:false,
                    cropend: function (e) {
                        self.updateBbox();
                    },
                });
                this.updateBbox();
            },
            mount() {
                let self = this;
                this.$wire.on('updatedPhoto', (photo) => {
                    self.cropperInstance.replace(photo);
                })
                this.initCroperjs();
            }
        }
    }
</script>
```

7.4.3 Σελίδα Λίστας Εικόνων

Μετά την αποθήκευση της καινούριας εικόνας ο χρήστης μεταφέρεται στη σελίδα με την λίστα από όλες τις φωτογραφίες που έχει ανεβάσει στην πλατφόρμα. Η κλάση και ο κώδικάς της που είναι υπεύθυνη για τη λειτουργία αυτή φαίνεται παρακάτω. Η πρώτη function που βλέπουμε είναι η loadImages η οποία είναι εσκεμμένα άδεια. Ο σκοπός της είναι να την καλέσει το frontend αφού η σελίδα έχει ήδη γίνει ορατή στον χρήστη και αμέσως μετά να ξεκινήσει να φορτώνει την λίστα με τις εικόνες. Αυτό γίνεται έτσι ώστε ο χρήστης να μην έχει μεγάλες αναμονές κατά την διάρκεια της περιήγησης. Η επόμενη function είναι η deleteImage, η οποία δέχεται σαν παράμετρο το μοναδικό αριθμό μιας εικόνας και στη συνέχεια, αν αυτή η εικόνα ανήκει στο χρήστη την διαγράφει πρώτα από τον δίσκο του server και έπειτα από τη βάση δεδομένων. Η τρίτη υπορουτίνα δέχεται την ίδια παράμετρο με την διαγραφή εικόνας και είναι υπεύθυνη για την εμφάνιση του μηνύματος επιβεβαίωσης διαγραφής εικόνας. Ο διάλογος επιβεβαίωσης που καλείται είναι δυναμικός και χρησιμοποιείται σε αρκετά σημεία της ιστοσελίδας (θα αναλυθεί σε επόμενη υποενότητα). Προκειμένου λοιπόν να τον καλέσουμε, πρέπει να συμπληρώσουμε τις παραμέτρους που είναι αναγκαίες για αυτόν. Στην πορεία, συναντάμε την clickedImage, η οποία -όπως και οι 2 προηγούμενες- δέχεται σαν παράμετρο τον μοναδικό αριθμό μιας εικόνας. Σαν αποτέλεσμα έχει την εμφάνιση ενός μικρού παραθύρου με το αποτέλεσμα της διάγνωσης της εικόνας ανάλογα με την ένδειξη κινδύνου. Τέλος, η function render είναι υπεύθυνη για την σελιδοποίηση των αποτελεσμάτων στον χρήστη.

```
class ImageList extends Component
{
    use WithPagination;
```

```
    public $searchInput = '';

    protected $listeners = ['deleteImage'];

    /*
     * An empty callback for livewire in order to defer load the
image list.
     */
    public function loadImages()
    {

    }

    /**
     * Tries to delete image with specific id after checking policy
     * @param $imageId
     */
    public function deleteImage($imageId)
    {
        $image = Image::findOrFail($imageId);
        if(!Auth::user()->can('delete',$image))
            return abort(403);

        try {
            unlink(storage_path('app/' . $image->file_path));
        } catch (\Exception $e) {
        }
        $image->delete();
        $this->emit('resetModal');
        $this->render();
    }

    /**
     * Shows modal and parses dynamic action to delete specific image
id
     * @param $imageId
     */
    public function modalDeleteImage($imageId)
    {
        $modalData = [];
        $modalData['showOk'] = true;
        $modalData['showCancel'] = true;
        $modalData['modalTitle'] = 'Notice';
        $modalData['modalMessage'] = 'Are you sure you want to delete
this image?';
        $modalData['modalAction'] = ['action' => 'deleteImage',
'parameters' => [$imageId]];
        $this->emitTo('modal', 'callModal', $modalData);
    }

    /**
     * Shows image info for imageId by calling modal.
     * @param $imageId
     */
    public function clickedImage($imageId)
    {
        //TODO ditch modal and make a new view for image showing with
feature extraction from CNN
        $image = Image::findOrFail($imageId);
        if ($image->status == 'uploaded') {
            $modalData = [];
```

```
            $modalData['showOk'] = true;
            $modalData['showCancel'] = false;
            $modalData['modalTitle'] = 'Notice';
            $modalData['modalMessage'] = 'This image is still being
assessed by our AI.';
            $this->emitTo('modal', 'callModal', $modalData);
        } elseif ($image->status == 'finished') {
            $modalData = [];
            $modalData['showOk'] = true;
            $modalData['showCancel'] = false;
            $modalData['modalTitle'] = 'Results';

            if ($image->benign_malignant <= 45)
                $modalData['modalMessage'] = "Your skin lesion was
classified by our AI as <span class=\"text-green-400 font-
bold\">safe</span>. This doesn't mean that you shouldn't visit your
dermatologist regularly. ";
            else if ($image->benign_malignant < 50)
                $modalData['modalMessage'] = "Your skin lesion was
classified by our AI as <span class=\"text-green-400 font-
bold\">safe</span> but it is within margin of error. This doesn't
mean that you shouldn't visit your dermatologist regularly.";
            else if ($image->benign_malignant < 65)
                $modalData['modalMessage'] = "Your skin lesion was
classified by our AI as <span class=\"text-red-400 font-
bold\">malignant</span> but it is within margin of error. You should
visit a dermatologist for a check up.";
            else
                $modalData['modalMessage'] = "Your skin lesion was
classified by our AI as <span class=\"text-red-400 font-
bold\">malignant</span> with a certain level of certainty. You should
visit a dermatologist as soon as possible for a check up.";

            $this->emitTo('modal', 'callModal', $modalData);
        } elseif ($image->status == 'failed') {
            $modalData = [];
            $modalData['showOk'] = true;
            $modalData['showCancel'] = false;
            $modalData['modalTitle'] = 'Error';
            $modalData['modalMessage'] = "Our AI failed to classify
your image as malignant or benign. Our support team will look into
that and inform you as soon as possible.";
            $this->emitTo('modal', 'callModal', $modalData);
        }
    }

    public function render()
    {
        $data['images'] = array();
        if (Auth::user()) {
            $data['images'] = Image::where('user_id', Auth::user()-
>id);

            // If user has typed into search bar get related images.
            if (!empty($this->searchInput))
                $data['images'] = $data['images']->where('name',
'like', "%" . $this->searchInput . "%")
                    ->orWhere('original_name', 'like', "%" . $this-
>searchInput . "%")
                    ->orWhere('description', 'like', "%" . $this-
>searchInput . "%");
```

```
            $data['images'] = $data['images']-
>orderByDesc('created_at')->paginate(10);
        }


        return view('components.images.list-image', $data);
    }
}
```

7.4.4 Διάλογος Επιβεβαίωσης

Ο διάλογος επιβεβαίωσης είναι ένα τμήμα της εφαρμογής που χρησιμοποιείται σε αρκετά σημεία, όπως η επιβεβαίωση διαγραφής μιας εικόνας, η λήψη αποτελεσμάτων από την εφαρμογή κ.ά. Είναι δυναμικός στο σχεδιασμό έτσι ώστε να μην χεριάζεται να αντιγράφουμε το ίδιο κομμάτι κώδικα σε πολλά σημεία της εφαρμογής. Επίσης, ακούει σε 2 events (callModal που τον αρχικοποιεί και resetModal που τον επαναφέρει στην αρχική του κατάσταση και τον κρύβει από τον χρήστη), έτσι ώστε να μπορούμε να τον καλέσουμε από οποιοδήποτε σημείο. Ο διάλογος είναι παραμετροποιήσιμος και μερικές από τις επιλογές που μας δίνονται είναι οι ακόλουθες:

- showOk: Αν το κουμπί Ok θα εμφανίζεται στον χρήστη.
- showCancel: Αν το κουμπί Cancel θα εμφανίζεται στον χρήστη.
- modalTitle: Ο τίτλος του διαλόγου.
- modalMessage: Το μήνυμα που θα εμφανίζεται στον χρήστη.
- modalAction: Η ενέργεια που θα εκτελείται από τον διάλογο όταν ο χρήστης πατήσει το κουμπί επιβεβαίωσης. Η προκαθορισμένη ενέργεια είναι να κρύψει το παράθυρο του διαλόγου.

```
class Modal extends Component
{
    public $showModal = false;
    public $showOk = false;
    public $showCancel = false;
    public $modalTitle = '';
    public $modalMessage = '';
    public $modalAction = ['action' => 'resetModal','parameters' =>
[]];
    public $actionParameters = '';

    protected $listeners = ['callModal','resetModal'];

    /**
     * When event callModal is called set the parameters from the
array that was called.
     * @param $modalData
     */
    public function callModal($modalData)
    {
        foreach ($modalData as $parameter => $value)
```

```
            $this->{$parameter} = $value;

        $this->showModal = true;
    }

    /**
     * Reset modal to it's original state
     */
    public function resetModal()
    {
        $this-
>reset(['showModal','modalTitle','modalMessage','modalAction','showOk
','showCancel','actionParameters']);
    }

    public function render()
    {
        $this->actionParameters = "'".implode("','",$this-
>modalAction['parameters'])."'";
        return view('components.modal');
    }
}
```

7.5 *Script Πρόγνωσης*

Στον server της εφαρμογής -πέρα από την ιστοσελίδα- τρέχει και ένα script γραμμένο σε python. Αυτό λειτουργεί σαν γέφυρα μεταξύ του νευρωνικού δικτύου και της εφαρμογής και είναι υπεύθυνο για την παραγωγή αποτελεσμάτων κάθε φορά που ο χρήστης ανεβάζει μια εικόνα σε αυτήν. Ο κώδικας αυτός μοιράζεται πολλά κοινά με το script αξιολόγησης και εκπαίδευσης του μοντέλου, όπως την δημιουργία του μοντέλου και την φόρτωση των βαρών. Ωστόσο, υπάρχει ένα σημείο που διαφέρει και αυτό είναι ότι δεν γίνεται κάποια εκπαίδευση ή αξιολόγηση, αλλά πρόγνωση. Πρώτα συνδεόμαστε στη βάση που τρέχει η εφαρμογή, όπου οι χρήστες ανεβάζουν τις εικόνες, έτσι ώστε να πάρουμε όλες αυτές που περιμένουν αποτελέσματα. Για κάθε μια από αυτές τις εικόνες κάνουμε Test Time Augmentation (TTA), δηλαδή περνάμε τις εικόνες από τη function proprocess που αναλύθηκε νωρίτερα. Η διαδικασία αυτή επαναλαμβάνεται 10 φορές και παίρνουμε τον μέσο όρο πρόβλεψης του μοντέλου για την συγκεκριμένη εικόνα. Στο τέλος, αποθηκεύουμε τα αποτελέσματα στη βάση έτσι ώστε να γίνουν ορατά στον χρήστη που την ανέβασε.

```
while(True):
    mydb = mysql.connector.connect(
        host="localhost",
        user=username,
        password=password,
        database=database
    )
```

```
    df_test = pd.DataFrame({"image_id":[],
                            "benign_malignant":[]})
    finalResults=pd.DataFrame({"image_name":[],
                               "benign_malignant":[]})
    try:
        mycursor = mydb.cursor(dictionary=True)
        mycursor.execute("SELECT * FROM images WHERE
status='uploaded'")
        images = mycursor.fetchall()

        for image in images:
            df_test =
df_test.append({"image_name":image['file_path'].split('photos/')[-1],

"benign_malignant":0},ignore_index=True)
    except:
        e = sys.exc_info()[0]
        print(e)

    finally:
        mycursor.close()
        mydb.close()

    df_test=df_test.reset_index()

    for index, row in df_test.iterrows():
        single_df = pd.DataFrame({"image_name":[row['image_name']],
                                  "benign_malignant":[0]})

        test_generator = test_datagen.flow_from_dataframe(
            dataframe = single_df,
            directory = test_dir,
            x_col="image_name",
            target_size=(im_size,im_size),
            color_mode="rgb",
            batch_size=1,
            shuffle=False,
            class_mode=None
        )

        pred = 0;
        for i in range(0,ttaTimes):
            if(classes == 1):
                pred += saved_model.predict(
                    test_generator,
                    steps=1,
                    verbose=0
                )[0]
            else:
                pred += saved_model.predict(
                    test_generator,
                    steps=1,
                    verbose=0
                )
        pred /= ttaTimes
        # print(pred)
        if(classes > 1):
            pred = [np.argmax(pred)]
        finalResults =
finalResults.append({"image_name":row['image_name'],
```

```
"benign_malignant":pred[0]},ignore_index=True)

    # print(finalResults)
    # import sys
    # sys.exit(1)
    mydb = mysql.connector.connect(
        host="localhost",
        user=username,
        password=password,
        database=database
    )
    try:
        mycursor = mydb.cursor(dictionary=True)
        for result,row in finalResults.iterrows():
            mycursor.execute("UPDATE images SET
benign_malignant='"+str(round(row['benign_malignant']*100,2))+"',stat
us='finished' WHERE file_path LIKE '%"+row['image_name']+"'")
            print("UPDATE images SET
benign_malignant='"+str(round(row['benign_malignant']*100,2))+"',stat
us='finished' WHERE file_path LIKE '%"+row['image_name']+"'")
            mydb.commit()
    except:
        e = sys.exc_info()[0]
        print(e)

    finally:
        mycursor.close()
        mydb.close()

    del mydb,df_test
    time.sleep(5)
```

7.6 *Πλατφόρμες και προγραμματιστικά εργαλεία*

Η διπλωματική αυτή εργασία αναπτύχθηκε σε περιβάλλον Windows 10 και σαν IDE χρησιμοποιήθηκε το InteliJ Idea, τόσο για το νευρωνικό δίκτυο όσο και για την web εφαρμογή. Ο συνδυασμός του αυτή την στιγμή βρίσκεται online στο https://skincaire.eu σε server με περιβάλλον Ubuntu 18.04 LTS. Προκειμένου να στηθεί η web εφαρμογή και το νευρωνικό δίκτυο χρειαζόμαστε τα εξής:

- Server με πάνω από 4GB ram
- MySql 5.7+
- Web Server με PHP 8.0+
- Python 3.7+
- Tensorflow 2.4+
- Composer 2.0+

Στη συνέχεια ακολουθούν οδηγίες εγκατάστασης της εφαρμογής και του νευρωνικού δικτύου σε ubuntu 18.04 που είναι και η έκδοση η οποία προτείνεται αυτή τη στιγμή.

7.6.1 Οδηγίες Εγκατάστασης εφαρμογής

Για την εγκατάσταση της web εφαρμογής χρειαζόμαστε έναν web server που να υποστηρίζει PHP σε εκδόσεις μεγαλύτερες από 7.2.33 και μια βάση δεδομένων συμβατή με το Laravel (προτείνετε η MySql 5.7). Εφόσον έχουμε αυτά, τότε το πρώτο μας βήμα είναι να δημιουργήσουμε ένα αρχείο .env στον κύριο φάκελο της εφαρμογής με την εξής μορφή:

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:kXKfA1ecaj2Tzj3pO9zfCBPSoiDrVcmnehPXJiL0t2c=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=melanoma_detection
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=database
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1
```

```
MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Οι βασικότερες μεταβλητές που πρέπει να ρυθμιστούν είναι η σύνδεση με την βάση δεδομένων.

Στη συνέχεια πρέπει να τρέξουμε την παρακάτω εντολή στη γραμμή εντολών:

```
composer install
```

Η εντολή αυτή θα εγκαταστήσει όλα τα απαραίτητα πακέτα για το Laravel ώστε να μπορεί να τρέξει για να προχωρήσουμε στο επόμενο βήμα που είναι η εγκατάσταση της βάσης. Η δημιουργία της βάσης γίνεται πολύ εύκολα: αφού έχουμε βεβαιωθεί ότι το .env αρχείο μας είναι σωστά ρυθμιζόμενο, τότε τρέχουμε στην γραμμή εντολών την ακόλουθη εντολή:

```
php artisan migrate
```

Με την σειρά της η εντολή αυτή θα ανοίξει όλα τα αρχεία που έχουμε μέσα στο φάκελο database/migrations και θα δημιουργήσει τη βάση που είναι απαραίτητη για την web εφαρμογή.

Σε αυτό το σημείο η εφαρμογή πρέπει να είναι προσβάσιμη και να λειτουργεί κανονικά.

7.6.2 Οδηγίες Εγκατάστασης Νευρωνικού Δικτύου

Προκειμένου να εγκαταστήσουμε το νευρωνικό δίκτυο όπως προαναφέρθηκε, πρέπει να υπάρχει εγκατεστημένη η Python και η βιβλιοθήκη Tensorflow και να έχουμε διαθέσιμα 2 αρχεία από την εργασία:

- Predictor.py Το αρχείο python που είναι υπεύθυνο για την πρόβλεψη των εικόνων
- Weights.h5 Το αρχείο βαρών του νευρωνικού δικτύου

Εφόσον έχουμε αυτά τα 2 αρχεία στον ίδιο φάκελο, η εγκατάσταση είναι αρκετά εύκολη: ανοίγουμε το αρχείο python και ρυθμίζουμε τις εξής παραμέτρους:

```
test_dir = r"C:\xampp\htdocs\www\melanoma-
detection\storage\app\photos"
hair_dir = r"C:\Users\ \Documents\Dataset\hairs"
ttaTimes = 10
im_size = 224
classes = 1
username="root"
database="melanoma_detection"
password=""
```

Ο φάκελος test_dir είναι ο φάκελος της web εφαρμογής, εκεί δηλαδή που οι χρήστες ανεβάζουν τις φωτογραφίες τους. Ο φάκελος hair_dir δίνεται μαζί με τα αρχεία της εφαρμογής και περιλαμβάνει τις προσομοιωμένες τρίχες για το data augmentation. Οι παράμετροι ttaTimes, im_size, classes δεν πρέπει να αλλάξουν. Τέλος, οι παράμετροι username, database και password αναφέρονται στα στοιχεία της βάσης δεδομένων και πρέπει να είναι ίδιες με αυτά

που ρυθμίστηκαν στο αρχείο .env της web εφαρμογής. Στη συνέχεια, για να τρέξουμε το script στο background γράφουμε στην γραμμή εντολών το εξής:

```
nohup python predictor.py &
```

Αν έχουμε ακολουθήσει όλες τις παραπάνω οδηγίες, τότε η εφαρμογή πρέπει να είναι πλήρως λειτουργική, να επιτρέπει στους χρήστες να ανεβάσουν εικόνες και μετά από μερικά δευτερόλεπτα να βγάζει αποτελέσματα για αυτές.

8

Επίλογος

Σε αυτό το κεφάλαιο συνοψίζουμε τα αποτελέσματα και τους στόχους τις διπλωματικής, ενώ παράλληλα προτείνουμε επεκτάσεις που μπορούν να γίνουν, τόσο στο νευρωνικό δίκτυο όσο και στην web εφαρμογή που το πλαισιώνει.

8.1 Σύνοψη και συμπεράσματα

Η διπλωματική αυτή είχε ως στόχο να αναπτύξει ένα νευρωνικό δίκτυο όπου μπορεί να αναγνωρίσει αποτελεσματικά και αξιόπιστα κακοήθειες στο ανθρώπινο δέρμα. Τα αποτελέσματα των πειραμάτων αξιολόγησης μάς δείξανε ότι το μοντέλο μας είναι αρκετά αξιόπιστο στο να εντοπίζει μελανώματα με συνολική ακρίβεια 91.22% . Το αποτέλεσμα αυτό είναι αρκετά υψηλότερο από τον αρχικό στόχο του 80% που αποτελεί τον μέσο όρο αναγνώρισης μελανώματος από δερματολόγο.

Ο δεύτερος στόχος της διπλωματικής ήταν να δώσει σε άτομα που θέλουν να ελέγξουν μια δερματική αλλοίωση έναν εύκολο και γρήγορο τρόπο να το κάνουν με τη βοήθεια του κινητού τους ή του προσωπικού τους υπολογιστή. Αυτός ο στόχος επιτεύχθηκε με την ανάπτυξη μιας διαδικτυακής εφαρμογής, βασισμένη στις τελευταίες διαθέσιμες τεχνολογίες του web development.

8.2 Μελλοντικές επεκτάσεις

Οι μελλοντικές επεκτάσεις που μπορούν να γίνουν επάνω σε αυτή την διπλωματική χωρίζονται σε 2 μέρη:

- Το πρώτο μέρος είναι οι επεκτάσεις που αφορούν στο νευρωνικό δίκτυο. Η πρώτη επέκταση που θα μπορούσαμε να προτείνουμε είναι να δοκιμαστούν τα καινούρια δίκτυα NFNet και να εξεταστεί το κατά πόσο αυτά αποδίδουν καλύτερα ή χειρότερα αποτελέσματα στο πρόβλημα εντοπισμού του

μελανώματος. Η επόμενη επέκταση επάνω στο δίκτυο θα ήταν η μετατροπή του από δυαδικό σε πολύ-κλασικό δίκτυο, δηλαδή εκτός από το αν η εικόνα είναι καλοήθης ή κακοήθης να ενημερώνει τον χρήστη και για την συγκεκριμένη πάθηση αν αυτή υπάρχει. Τέλος, θα πρέπει να ερευνηθεί κατά πόσο η μέθοδος χωρίς επίβλεψη θα βοηθούσε ή θα έβλαπτε τα αποτελέσματα του δικτύου.

- Το δεύτερο μέρος της επέκτασης αφορά την web εφαρμογή που συνοδεύει το νευρωνικό δίκτυο. Εφόσον επεκταθεί το μοντέλο ώστε να αναγνωρίζει πολλαπλές αλλοιώσεις και να ταξινομεί τις φωτογραφίες, αντίστοιχες ενημερώσεις και τροποποιήσεις πρέπει να γίνουν και στην εφαρμογή για να ενημερώνει καταλλήλως τον χρήστη. Μια ακόμα αλλαγή που θα μπορούσαμε να προτείνουμε είναι η τροποποίηση του συστήματος έτσι ώστε να μην χρειάζεται εγγραφή στο σύστημα αλλά να επιτρέπεται και στους επισκέπτες να ανεβάζουν εικόνες. Τέλος, μια σημαντική επέκταση θα ήταν να υπάρχει τύπος χρήστη (δερματολόγου/απλού χρήστη) όπου ο δερματολόγος θα μπορεί να καταχωρεί στοιχεία ασθενή και πολλαπλές εικόνες του ταυτόχρονα.

9

Βιβλιογραφία

[1] Cancer.org, 2021. [Online]. Available: https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2021/cancer-facts-and-figures-2021.pdf. [Accessed: 6- Jul- 2021].

[2] K. Team, "Keras documentation: Image classification via fine-tuning with EfficientNet", Keras.io, 2021. [Online]. Available: https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/. [Accessed: 6- Jul- 2021].

[3] G. Day and R. Barbour, "Automated melanoma diagnosis: where are we at?", Skin Research and Technology, vol. 6, no. 1, pp. 1-5, 2000. Available: 10.1034/j.1600-0846.2000.006001001.x.

[4] A. Miller, "A review of neural network applications in Astronomy", Vistas in Astronomy, vol. 36, pp. 141-161, 1993. Available: 10.1016/0083-6656(93)90118-4.

[5] M. Sneyd, C. Cameron and B. Cox, "Individual risk of cutaneous melanoma in New Zealand: developing a clinical prediction aid", BMC Cancer, vol. 14, no. 1, 2014. Available: 10.1186/1471-2407-14-359.

[6] C. Garbe et al., "European consensus-based interdisciplinary guideline for melanoma. Part 1: Diagnostics – Update 2019", European Journal of Cancer, vol. 126, pp. 141-158, 2020. Available: 10.1016/j.ejca.2019.11.014 .

[7] A. Ali and T. Deserno, "A systematic review of automated melanoma detection in dermatoscopic images and its ground truth data", Medical Imaging

2012: Image Perception, Observer Performance, and Technology Assessment, 2012. Available: 10.1117/12.912389.

[8] "Rook's Textbook of Dermatology", 2004. Available: 10.1002/9780470750520.

[9] J. Lai-Cheong and J. McGrath, "Structure and function of skin, hair and nails", Medicine, vol. 41, no. 6, pp. 317-320, 2013. Available: 10.1016/j.mpmed.2013.04.017.

[10] Q. Ha, B. Liu and F. Liu, "Identifying Melanoma Images using EfficientNet Ensemble: Winning Solution to the SIIM-ISIC Melanoma Classification Challenge", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/2010.05351.

[11] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/1905.11946.

[12] H. Zhang et al., "ResNeSt: Split-Attention Networks", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/2004.08955. [Accessed: 12- Sep- 2021].

[13]]S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated Residual Transformations for Deep Neural Networks", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/1611.05431v2.

[14] I. El Naqa and M. Murphy, "What Is Machine Learning?", Machine Learning in Radiation Oncology, pp. 3-11, 2015. Available: 10.1007/978-3-319-18305-3_1.

[15] S. Kotsiantis, I. Zaharakis and P. Pintelas, "Machine learning: a review of classification and combining techniques", Artificial Intelligence Review, vol. 26, no. 3, pp. 159-190, 2006. Available: 10.1007/s10462-007-9052-3.

[16] "Python (programming language) - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language).

[17] "TensorFlow - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow.

[18] "PHP - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/PHP#PHP_8.

[19] "Laravel - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Laravel.

[20] S. Kitada and H. Iyatomi, "Skin lesion classification with ensemble of squeeze-and-excitation networks and semi-supervised learning", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/1809.02568.

[21] R. Xu, Y. Tao, Z. Lu and Y. Zhong, "Attention-Mechanism-Containing Neural Networks for High-Resolution Remote Sensing Image Classification", Remote Sensing, vol. 10, no. 10, p. 1602, 2018. Available: 10.3390/rs10101602.

[22] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional networks and applications in vision", Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010. Available: 10.1109/iscas.2010.5537907.

[23] S. Pan and Q. Yang, "A Survey on Transfer Learning", IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, 2010. Available: 10.1109/tkde.2009.191 [Accessed 12 September 2021].

[24] K. Team, "Keras documentation: Transfer learning & fine-tuning", Keras.io, 2021. [Online]. Available: https://keras.io/guides/transfer_learning/.

[25] T. Lee, V. Ng, R. Gallagher, A. Coldman and D. McLean, "Dullrazor®: A software approach to hair removal from images", Computers in Biology and Medicine, vol. 27, no. 6, pp. 533-543, 1997. Available: 10.1016/s0010-4825(97)00020-6.

[26] Q. Abbas, M. Emre Celebi, I. Garcia and W. Ahmad, "Melanoma recognition framework based on expert definition of ABCD for dermoscopic images", Skin Research and Technology, vol. 19, no. 1, pp. e93-e102, 2012. Available: 10.1111/j.1600-0846.2012.00614.x.