ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Αγρονόμων & Τοπογράφων Μηχανικών-Μηχανικών Γεωπληροφορικής
Τομέας Τοπογραφίας
Εργαστήριο Φωτογραμμετρίας

ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ ΓΙΑ ΣΗΜΑΣΙΟΛΟΓΙΚΗ ΚΑΤΑΤΜΗΣΗ ΕΙΚΟΝΩΝ ΜΕ 3 ΚΑΝΑΛΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ηλίας Α. Τσαρπαλής

Επιβλέπων: καθ. Νικόλαος Δουλάμης

Αθήνα, Οκτώβριος 2021

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
School of Rural, Surveying and
Geoinformatics Engineering
Department of Topography
Laboratory of Photogrammetry

DEEP LEARNING TENCHIQUES FOR 3-CHANNEL IMAGE SEGMENTATION

DIPLOMA THESIS

Ilias A. Tsarpalis

Supervisor: prof. Nikolaos Doulamis

Athens, October 2021

Τεχνικές Βαθιάς Μάθησης για Σημασιολογική Κατάτμηση Εικόνων με 3 Κανάλια

Διπλωματική Εργασία

Ηλίας Α. Τσαρπαλής

Επιβλέπων: Δουλάμης Νικόλαος

Τσαρπαλής Η. Α. (2021).
Τεχνικές Βαθιάς Μάθησης για Σημασιολογική Κατάτμηση Εικόνων με 3 Κανάλια
Διπλωματική Εργασία
Εργαστήριο Φωτογραμμετρίας, Εθνικό Μετσόβιο Πολυτεχνείο, Αθήνα.

Tsarpalis I. A. (2021).
Deep Learning Techniques for 3-Channel Image Segmentation
Diploma Thesis
Laboratory of Photogrammetry, National Technical University of Athens, Greece

Στα αδέρφια μου Παναγιώτη και Δημήτρη,
μαζί με τις ευχές μου για συνεχή ανέλιξη
και επίτευξη των στόχων τους

Περιεχόμενα

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF RURAL, SURVEYING AND GEOINFORMATICS ENGINEERING
DEPARTMENT OF TOPOGRAPHY

DIPLOMA THESIS

Deep Learning Techniques for 3-Channel Image Segmentation

Tsarpalis I. A. (supervised by Doulamis N.)

Abstract

The development of new deep learning techniques is being prolific the last decade, with countless applications on a variety of scientific problems. Recently, big datasets are employed to train neural networks more efficiently and make predictions with high accuracy. Self-driving cars, medical imaging, and remote corrosion control on existing structures are only few of the problems that a neural network can deal with.

This thesis, with title "Deep Learning Techniques for 3 Channel Image Segmentation", is dealing with the problem of corrosion detection on part of a wall at the Agios Nikolaos fortress of Rhodes. In order to complete this task, a Fully Convolutional Neural Network architecture is proposed, which makes use of the Residual Blocks and Attention, called Residual Attention U-Net.

The aforementioned terms and the theoretical concepts are explained in the first chapter of the thesis, which are considered necessary to fully understand the nature of the problem and the proposed solution. Afterwards, the architecture of the proposed network is described, as the method to predict the image segmentation.

The data was first collected using the HyperView system, which can take both Visual and Near-infrared Images and then they were combined to produce a 3-channel Image. After the post-processing procedure, the images and their binary masks, are used to train the model and make prediction on them using a software application realized in Python3. To evaluate the results of the predictions, the accuracy, precision, recall, and F1 scores have been used.

Herein, an attempt of best prediction on the corroded stones of the wall was made, despite the small size of the data set, being 11 pictures in total. Nevertheless, the new deep learning technique of ResAttUNet (Residual Attention U-Net) could adequately predict the percentage of corroded stones on the wall, fact that makes it a great tool for the engineers for problems related to corrosion detection and displacement remote control. However, it is important to understand that although the neural network was able to predict well the percentage corrosion of the wall, more refined results could be achieved using a bigger data size.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ-
ΜΗΧΑΝΙΚΩΝ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τεχνικές Βαθιάς Μάθησης για Σημασιολογική Κατάτμηση Εικόνων με 3 Κανάλια

Τσαρπαλής Η. Α. (Επιβλέπων: Δουλάμης Ν.)

Περίληψη

Η ανάπτυξη τεχνικών βαθιάς μάθησης νευρωνικών δικτύων σημειώνει ραγδαία ανάπτυξη την τελευταία δεκαετία βρίσκοντας εφαρμογή σε πολλούς επιστημονικούς κλάδους. Η δημιουργία τεράστιων βάσεων δεδομένων και η εύκολη πρόσβαση σε αυτά επιτρέπει την αποτελεσματική εκπαίδευση νευρωνικών δικτύων με υψηλές ακρίβειες. Τα αυτοκινούμενα οχήματα, η ανίχνευση και η αξιολόγηση ιατρικών δεδομένων και η παρακολούθηση παραμορφώσεων σε υφιστάμενες κατασκευές είναι μόνο μερικά από τα προβλήματα που μπορούν να αντιμετωπιστούν με πολύ καλές αποδόσεις από τα νευρωνικά δίκτυα.

Η συγκεκριμένη εργασία με τίτλο «Τεχνικές Βαθιάς Μάθησης για Σημασιολογική Κατάτμηση Εικόνων με 3 Κανάλια», αφορά την ανίχνευση διαβρωμένων λίθων σε κομμάτι τείχους που βρίσκεται στο φρούριο του Αγίου Νικολάου στην Ρόδο. Η σημασιολογική κατάτμηση έγινε με την χρήση ενός νέου τύπου Πλήρως Συνελικτικού Νευρωνικού Δικτύου που κάνει χρήση των Residual Blocks και της Προσοχής (Attention), το Residual Attention U-Net.

Στα πρώτα κεφάλαια γίνεται ανάλυση όλων των παραπάνω όρων και γενικότερα βασικών εννοιών που η γνώση τους θεωρήθηκε απαραίτητη για την πλήρη κατανόηση του προβλήματος και της προτεινόμενης λύσης του. Στην συνέχεια περιγράφεται η δομή του προτεινόμενου μοντέλου, καθώς και η διαδικασία πρόβλεψης της σημασιολογικής κατάτμησης των RGB δεδομένων.

Τα δεδομένα συλλέχθηκαν με την βοήθεια του συστήματος HyperView που επιτρέπει την λήψη και την εξαγωγή ένωσης πραγματικών εικόνων και εικόνων υπέρυθρης ακτινοβολίας. Οι εικόνες αυτές ύστερα από κατάλληλη επεξεργασία, εισήλθαν στο νευρωνικό σαν δεδομένα για την εκπαίδευση και την εξαγωγή προβλέψεων σημασιολογικών μασκών σε περιβάλλον Python 3. Η αξιολόγηση των προβλέψεων έγινε με την χρήση των μεταβλητών accuracy, precision, recall και F1-score.

Στην παρούσα διπλωματική εργασία έγινε προσπάθεια βέλτιστης πρόβλεψης των διαβρωμένων λίθων του υπό εξέταση τείχους, χρησιμοποιώντας στο σύνολο 11 εικόνες για την εκπαίδευση και την αξιολόγηση του νευρωνικού δικτύου. Παρά τον περιορισμένο αριθμό δεδομένων, το καινοτόμο δίκτυο Residual Attention U-Net ήταν ικανό να προβλέψει με

επάρκεια το ποσοστό διάβρωσης του τείχους, γεγονός που το καθιστά ένα πολλά υποσχόμενο εργαλείο Μηχανικού για τον συνεχή έλεγχο βλαβών και μετατοπίσεων σε υφιστάμενες κατασκευές. Ωστόσο, θα πρέπει να τονιστεί ότι παρόλο που οι 11 εικόνες αρκούσαν για τη σωστή πρόβλεψη της ποσοστιαίας διάβρωσης, υψηλότερη ακρίβεια θα μπορούσε να επιτευχθεί με τη χρήση μεγαλύτερης βάσης δεδομένων.

Ευχαριστίες

Ολοκληρώνοντας τη διπλωματική μου εργασία, θα ήθελα να ευχαριστήσω όσους με βοήθησαν και με στήριξαν όλο αυτό το διάστημα.

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντά μου και αναπληρωτή καθηγητή Ε.Μ.Π. κ. Νικόλαο Δουλάμη για την ανάθεση του συγκεκριμένου θέματος, τη συνεχή καθοδήγησή του, το ενδιαφέρον του αλλά και τη βοήθειά του σε θέματα γνωστικού αντικειμένου

Ιδιαίτερες ευχαριστίες θα ήθελα να απευθύνω στον υποψήφιο διδάκτορα Ε.Μ.Π. Γιάννη Ράλλη, για τις συμβουλές του και την άψογη συνεργασία που έχουμε. Επίσης θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Ιάσονα Κατσαμένη και τον μεταδιδακτορικό ερευνητή Ευτύχιο Πρωτοπαπαδάκη για τον πολύτιμο χρόνο που αφιέρωσαν και την βοήθεια τους στην ολοκλήρωση της διπλωματικής μου εργασίας.

1 Εισαγωγή – Βασικοί όροι

1.1 Τεχνητή Νοημοσύνη (Artificial Intelligence)

Η Τεχνητή Νοημοσύνη (Artificial Intelligence (A.I.)) [1] αποτελεί το κομμάτι της επιστήμης του προγραμματισμού που ασχολείται με την σχεδίαση ενός υπολογιστικού συστήματος, τέτοιου που να παρουσιάζει χαρακτηριστικά που συνδέονται με την ευφυία που συναντάτε στην ανθρώπινη συμπεριφορά. Αυτό σημαίνει να καταλαβαίνει τις ανθρώπινες γλώσσες, να μαθαίνει, να αιτιολογεί, να επιλύει προβλήματα κλπ. Από την στιγμή που ο άνθρωπος άρχισε να ασχολείται με αυτή την επιστήμη αυτή, στα μέσα της δεκαετίας του 50', έχουν αναπτυχθεί πολλές προγραμματιστικές τεχνικές που να υποστηρίζουν τα παραπάνω χαρακτηριστικά.

Η Τεχνητή Νοημοσύνη στο σήμερα έχει εξελιχθεί σε τέτοιο βαθμό που ο υπολογιστής μπορεί να αντικαταστήσει τον άνθρωπο σε πολλά επιστημονικά και καθημερινά πεδία. Ενδεικτικά ο υπολογιστής μπορεί πλέον:

- Να επιλύσει σύνθετα προβλήματα χημείας, βιολογίας, γεωλογίας, μηχανικής, οικονομικών κλπ.
- Να χειριστεί ρομποτικές κατασκευές για να εκτελέσει σύνθετες λειτουργίες, όπως οδήγηση αυτοκινήτου κλπ.
- Να απαντήσει σε ερωτήσεις που γίνονται σε διάφορες διαλέκτους.

Το πιο σημαντικό πρόβλημα όμως που είχαν να αντιμετωπίσουν οι προγραμματιστές είναι να δώσουν στον υπολογιστή την δυνατότητα να μαθαίνει. Η εκπαίδευση ενός υπολογιστή λοιπόν περιλαμβάνει την απόκτηση νέων γνώσεων , την ανάπτυξη δεξιοτήτων είτε κινητικών είτε υπολογιστικών και τέλος την επίλυση των προβλημάτων που του δίνονται , είτε αυτό σημαίνει να οδηγήσει ένα αμάξι είτε αν σημαίνει να επιλύσει ένα επιστημονικό πρόβλημα μέσα από παρατήρηση και πειραματισμό. Η επιστήμη λοιπόν που ασχολείται με την ανάπτυξη υπολογιστικών συστημάτων τεχνητής νοημοσύνης που να μπορούν να μαθαίνουν, είναι αυτή της Μηχανικής Μάθησης.

1.2 Μηχανική Μάθηση (Machine Learning)

Η μηχανική μάθηση [2] είναι η επιστήμη του προγραμματισμού που ασχολείται με την αυτόματη βελτίωση του αλγορίθμου μέσα από την χρήση δεδομένων. Η επιστήμη της μηχανικής μάθησης αποτελεί κομμάτι του Artificial Intelligence (A.I.). Είναι ένας αλγόριθμος που κατασκευάζει ένα μοντέλο χρησιμοποιώντας δεδομένα εκπαίδευσης (training data) με σκοπό να κάνει προβλέπεις ή να λάβει αποφάσεις για προβλήματα χωρίς να είναι προγραμματισμένος να λύνει τα συγκεκριμένα προβλήματα ή να λαμβάνει τις συγκεκριμένες αποφάσεις

Οι αλγόριθμοι Μηχανικής Μάθησης χρησιμοποιούνται ευρέως στο σήμερα σε πολλές καθημερινές πτυχές της ζωής μας. Όταν για παράδειγμα γίνεται μια αναζήτηση στο Google, αυτό αποτελεί ένα δεδομένο για τον αλγόριθμο το οποίο στην συνέχεια χρησιμοποιείται σε

συνδυασμό με πολλά άλλα ήδη υπάρχοντα δεδομένα για να γίνει μια πρόβλεψη της αναζήτησης που θα κάνει στο μέλλον ο χρήστης. Δηλαδή θα προβλέψει τις προτιμήσεις του χρήστη σύμφωνα με το μοντέλο που έχει φτιάξει για αυτόν.

Κάποιες από τις εργασίες που μπορεί να επιτελέσει ένας αλγόριθμος μηχανικής μάθησης είναι οι παρακάτω:

- Classification: προβλέπει την «ταμπέλα» μιας κατηγορίας για ένα αντικείμενο.
- Regression: προβλέπει την πραγματική τιμή ενός αντικειμένου, για παράδειγμα την τιμή ενός οικονομικού δείκτη.
- Ranking: μαθαίνει να κατατάσσει τα αντικείμενα σε μια σειρά σύμφωνα με κάποια κριτήρια, για παράδειγμα αναζητήσεις στο google όπως αναφέρθηκε πριν.
- Clustering: διαιρεί ένα σύνολο στοιχείων σε ομοιογενή υποσύνολα, για παράδειγμα στα social media αναγνωρίζει κοινότητες ανάμεσα σε μεγάλο αριθμό ατόμων και μπορεί να προβλέψει για παράδειγμα ένα άτομο ως προτεινόμενο φίλο για τον χρήστη σύμφωνα με την κοινότητα που ανήκει.
- Dimensionality reduction or manifold learning: μετατρέπει την αρχική αναπαράσταση ενός αντικειμένου σε μια χαμηλότερων διαστάσεων αλλά διατηρώντας κάποιες από τις ιδιότητες της αρχικής, για παράδειγμα η προ-επεξεργασία που γίνεται σε μια εικόνα για να μπορεί να για την «δει» ο υπολογιστής.

Παρακάτω θα αναλύσουμε τα στάδια ενός αλγόριθμου μηχανικής μάθησης μέχρι να δώσει το αποτέλεσμα που του ζητείται από τους χρήστες. Για την πιο εύκολη κατανόηση θα χρησιμοποιηθεί σαν παράδειγμα την ανίχνευση spam μηνυμάτων σε κάποια τυχαία πλατφόρμα email:

- Examples: είναι το σύνολο των δεδομένων που θα χρησιμοποιηθούν για εκπαίδευση ή εκτίμηση (Training set – Testing set), δηλαδή το σύνολο των email που χρησιμοποιήθηκαν.
- Features: είναι ένα σύνολο χαρακτηριστικών που συνδέονται με ένα δεδομένο, στην συγκεκριμένη περίπτωση θα μπορούσε να είναι το μέγεθος του μηνύματος, το όνομα του αποστολέα κλπ.
- Labels: τιμές ή κατηγορίες που συνδέονται με τα δεδομένα, για παράδειγμα το αν ένα email είναι SPAM ή non-SPAM.
- Training sample: είναι το σύνολο των δεδομένων που χρησιμοποιούνται για την εκπαίδευση του αλγορίθμου μαζί με τις κατηγορίες στις οποίες ανήκουν, δηλαδή το σύνολο των email που χρησιμοποιούνται μαζί με την πληροφορία για το καθένα αν είναι SPAM ή όχι.
- Validation sample: είναι τα δεδομένα που χρησιμοποιούνται κατά την διάρκεια της εκπαίδευσης για να ρυθμίσουν τις παραμέτρους του αλγόριθμου.

- Test sample: είναι δεδομένα που χρησιμοποιούνται μετά την εκπαίδευση του μοντέλου για να δώσουν την επίδοση του αλγορίθμου, για παράδειγμα τα email που χρησιμοποιούνται από τον αλγόριθμο για να κάνει μια πρόβλεψη, τα όποια όμως δεν έχουν χρησιμοποιηθεί για την εκπαίδευση του.
- Loss function: είναι μια συνάρτηση που μετράει την διαφορά μεταξύ της εκτιμώμενης τιμής και της πραγματικής τιμής, δηλαδή κατά πόσο ο αλγόριθμος «έπεσε» έξω στην πρόβλεψη του αν το email είναι SPAM ή όχι.

Σε αυτό το σημείο θα γίνει ανάλυση των τρόπων εκπαίδευσης των αλγόριθμων μηχανικής μάθησης. Η κατηγορία αυτή εξαρτάται από τον τύπο των δεδομένων εκπαίδευσης, από τον τρόπο και την σειρά με την οποία αυτά θα δοθούν στον αλγόριθμο, καθώς και από τα δεδομένα εκτίμησης:

- Supervised learning: τα δεδομένα που δίνονται για εκπαίδευση είναι κατηγοριοποιημένα και γίνονται σε εκτιμήσεις σε δεδομένα που δεν έχουν χρησιμοποιηθεί στην εκπαίδευση.
- Unsupervised learning: τα δεδομένα που χρησιμοποιούνται στην εκπαίδευση δεν είναι κατηγοριοποιημένα και οι προβλέψεις γίνονται σε δεδομένα που δεν έχουν χρησιμοποιηθεί στην εκπαίδευση.
- Semi-supervised: εδώ το σύνολο των δεδομένων που δίνονται για την εκπαίδευση αποτελούνται από κατηγοριοποιημένα και μη-κατηγοριοποιημένα στοιχεία και γίνονται προβλέψεις σε δεδομένα που δεν έχουν χρησιμοποιηθεί στην εκπαίδευση του αλγορίθμου.
- Transductive inference: εδώ τα δεδομένα που δίνονται για εκπαίδευση είναι κατηγοριοποιημένα δεδομένα από το training sample καθώς και μη-κατηγοριοποιημένα δεδομένα εκτίμησης. Παρόλα αυτά ο στόχος είναι να γίνουν προβλέψεις κατηγοριών μόνο για τα δεδομένα εκτίμησης.
- Online learning: σε αυτή την περίπτωση η εκπαίδευση αποτελείται από πολλαπλούς γύρους όπου αποτελούνται από ενέργειες εκπαίδευσης και εκτίμησης. Σε κάθε γύρο ο αλγόριθμος παίρνει ένα δεδομένο εκπαίδευσης , κάνει μια πρόβλεψη γι' αυτό και τελικά το συγκρίνει με την πραγματική τιμή.
- Reinforcement learning: στην περίπτωση αυτή η εκπαίδευση γίνεται πάλι με πολλαπλές ενέργειες εκπαίδευσης και εκτίμησης. Ο αλγόριθμος σε αυτή την περίπτωση έρχεται σε ένα συνεχόμενο δίλλημα αν θα αλληλοεπιδράσει με το περιβάλλον του με σκοπό να συλλέξει κι άλλες πληροφορίες ή αν θα εκμεταλλευτεί τις ήδη δοσμένες-συλλεγμένες πληροφορίες που έχει.

1.3 Feedforward deep networks

Feedforward deep networks αποτελούν παραμετρικές συναρτήσεις οι οποίες ορίζονται από την σύνθεση πολλών παραμετρικών συναρτήσεων. Κάθε μια από αυτές τις συνιστώσες συναρτήσεις έχει πολλαπλές εισόδους και πολλαπλές εξόδους. Στην επιστήμη των νευρωνικών δικτύων οι συνιστώσες αυτές συναρτήσεις ονομάζονται layers και κάθε μια από τις πολλές

εξόδους τους ονομάζεται μονάδα ή χαρακτηριστικό (unit or feature). Παραπάνω ανάλυση θα γίνει στο Κεφάλαιο 2.

1.4 Όραση Υπολογιστή (Computer Vision)

Η όραση είναι η ικανότητα του ανθρώπου, όπως και πολλών άλλων ζώων, να βλέπουν τα αντικείμενα που βρίσκονται μπροστά τους και να αντιλαμβάνονται το σχήμα τους, την απόσταση στην οποία βρίσκονται από τους ίδιους, το χρώμα τους, την υφή τους κλπ. Η ικανότητα αυτή μπορεί να φαίνεται απλή για εμάς τους ίδιους σαν διαδικασία, αλλά στην περίπτωση της Όρασης Υπολογιστή η διαδικασία αυτή γίνεται πιο πολύπλοκη.

Η Όραση Υπολογιστή [3] περιλαμβάνει τις διαδικασίες της λήψης της εικόνας , της μετατροπής της σε μορφή που να είναι κατανοητή από τον υπολογιστή και τέλος την ανάλυση των αντικειμένων που απεικονίζονται σε αυτήν. Η Όραση Υπολογιστή βρίσκει πολλές εφαρμογές που βοηθούν στην επίλυση σύγχρονων προβλημάτων όπως είναι η αναγνώριση αντικειμένων σε μια εικόνα, που μπορεί να είναι ένα πρόσωπο, ένα αυτοκίνητο, μια ταμπέλα ακόμα και ζωτικά όργανα ή οστά σε μια ακτινογραφία, ακόμα και η τρισδιάστατη μοντελοποίηση κτηρίων ή αντικειμένων με δεδομένα μόνο εικόνες.

1.5 Σύγκριση τεχνικών βαθιάς μάθησης με ψηφιακής τηλεπισκόπησης

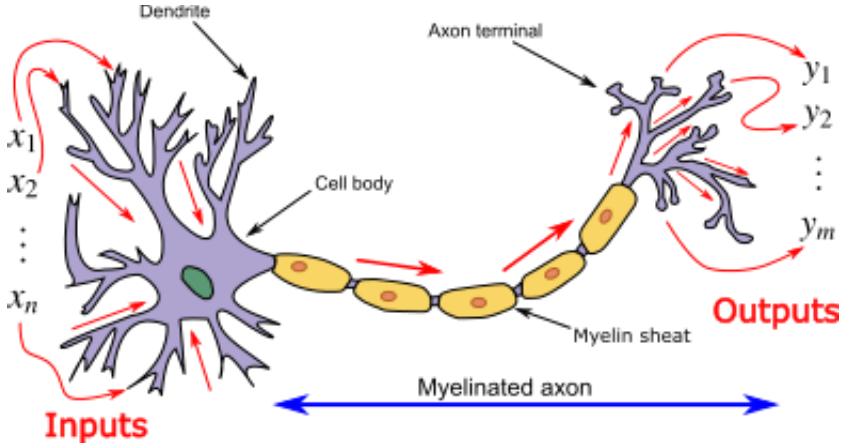
Οι παραδοσιακές τεχνικές της ψηφιακής τηλεπισκόπησης απαιτούν πολύωρη εργασία από άτομα προκειμένου να συλλέξουν, να οργανώσουν και να αναλύσουν τα δεδομένα για να καταλήξουν στο αποτέλεσμα που θέλουν. Η διαδικασία αυτή όμως κρύβει και πολλούς κινδύνους καθώς ο άνθρωπος είναι εύκολο να κάνει λάθη λόγω εξάντλησης , μη καλής κρίσης και διάφορων άλλων παραμέτρων.

Με την εισαγωγή των Νευρωνικών Δικτύων ως σύγχρονου εργαλείου της ψηφιακής τηλεπισκόπησης, παράγοντες όπως οι παραπάνω εξαλείφονται. Σήμερα διαδικασίες όπως η ανίχνευση ρωγμών ή διαβρώσεων στην επιφάνεια μιας κατασκευής είναι πολύ ευκολότερο και πολύ ταχύτερο από ότι ήταν με την χρήση των παλαιότερων μεθόδων. Αρχιτεκτονικές νευρωνικών δικτύων, όπως είναι τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) και ιδιαίτερα τα U-net , μπορούν να ταξινομήσουν εικονοστοιχεία RGB εικόνων με ιδιαίτερη ευκολία και σε μικρό χρονικό διάστημα, αυξάνοντας έτσι την ποιότητα των αποτελεσμάτων και μειώνοντας το κόστος τους.

2 Νευρωνικό Δίκτυο (Neural Network)

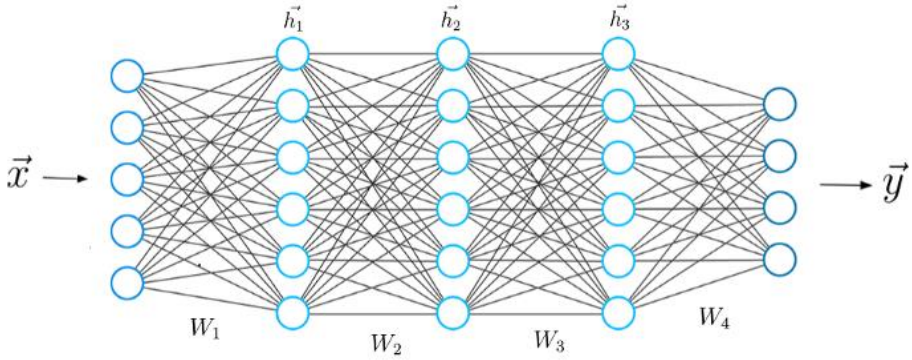
2.1 Εισαγωγή

Τα τεχνητά νευρωνικά δίκτυα αποτελούν το πιο θεμέλιο συστατικό της τεχνητής νοημοσύνης, είναι το εργαλείο με το οποίο αυτά μπορούν να αντιλαμβάνονται και να λαμβάνουν αποφάσεις. Η μορφή, δηλαδή η αρχιτεκτονική, ενός νευρωνικού δικτύου έχει εμπνευστεί από την αρχιτεκτονική , δηλαδή την δομή, του βιολογικού νευρώνα (Εικόνα 2.1).



Εικόνα 2.1: Βιολογικός νευρώνας

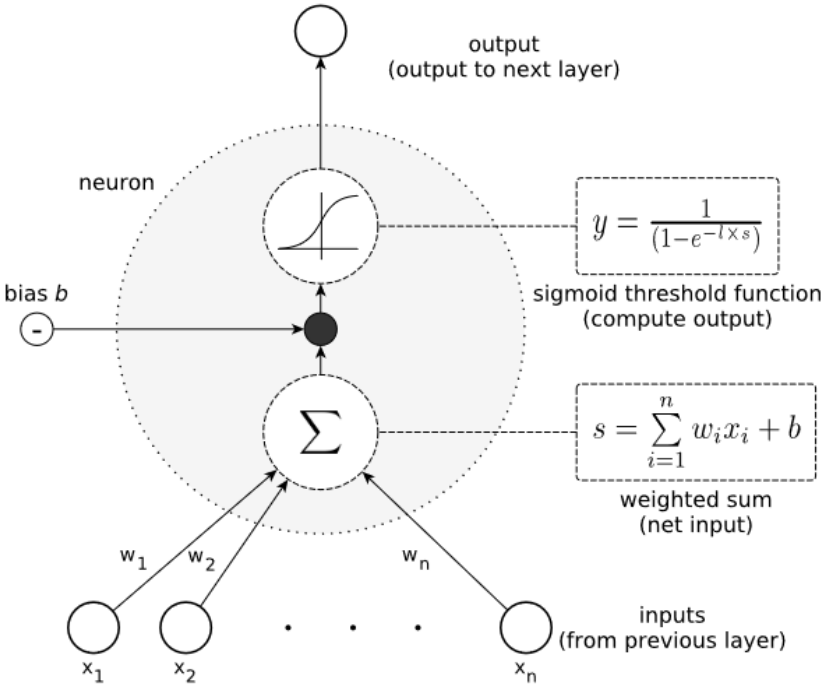
Μπορούμε να σκεφτούμε τους δενδρίτες του βιολογικού νευρικού κύτταρου, ως τα δεδομένα εισόδου x_i και τα δεδομένα εξόδου y_i του συστήματος μας. Τότε το σώμα και ο νευράξονας του κυττάρου αποτελούν όλες τις υπόλοιπες αποφάσεις και τις πράξεις που κάνει το σύστημα μας για να μπορεί αυτό να βγάλει ένα αποτέλεσμα. Το καθένα δεδομένο y_i δίνεται με την σειρά του στις απολήξεις ενός άλλου νευρικού κυττάρου το όποιο μετά από όλη την επεξεργασία που κάνει στο εσωτερικό του δίνει πληροφορίες στα επόμενα κύτταρα κλπ. Όλα αυτά τα κύτταρα μαζί σχηματίζουν ένα δίκτυο κυττάρων, τα οποία βοηθώντας το ένα το άλλο επιλύουν προβλήματα και λαμβάνουν αποφάσεις. Αν λοιπόν κάνουμε την αντιστοίχιση, το τεχνητό νευρωνικό μας δίκτυο καταλήγει με μορφή, λίγο πολύ ίδια με αυτή της παρακάτω εικόνας (Εικόνα 2.2).



Εικόνα 2.2: Δομή νευρωνικού δικτύου. Όπου x τα δεδομένα εισόδου, W_i τα βάρη μεταξύ των επιπέδων νευρώνων , h_i οι τιμές των νευρώνων στο κάθε επίπεδο και y τα δεδομένα εξόδου

2.2 Νευρώνας (node)

Ας εμβαθύνουμε όμως λίγο παραπάνω στο τι είναι ο νευρώνας [5][6]. Ποιες είναι δηλαδή οι διαδικασίες και ποιες οι αποφάσεις που εκτελούνται στο εσωτερικό του. Προκειμένου να καταλάβουμε τι γίνεται σε έναν νευρώνα πρέπει να δούμε ποια είναι τα βήματα που εκτελούνται στο εσωτερικό του από την στιγμή που λαμβάνει τα δεδομένα μέχρι την στιγμή που τα δίνει στους επόμενους νευρώνες (Εικόνα 2.3).



Εικόνα 2.3: Δομή νευρώνα

Ο κάθε νευρώνας λοιπόν δέχεται από το προηγούμενο επίπεδο (layer) κάποιες τιμές-δεδομένα, τα όποια έρχονται σε αυτό έτοιμα «πακεταρισμένα» το καθένα από αυτά με το δικό του «ζευγάρι» βάρος. Η πρώτη πράξη που κάνει ο νευρώνας είναι να αθροίσει όλα αυτά τα γινόμενα ζευγαριών δεδομένων-βαρών.

$$s = \sum_{i=1}^{n} w_i \times x_i \qquad \text{Εξ. (2.1)}$$

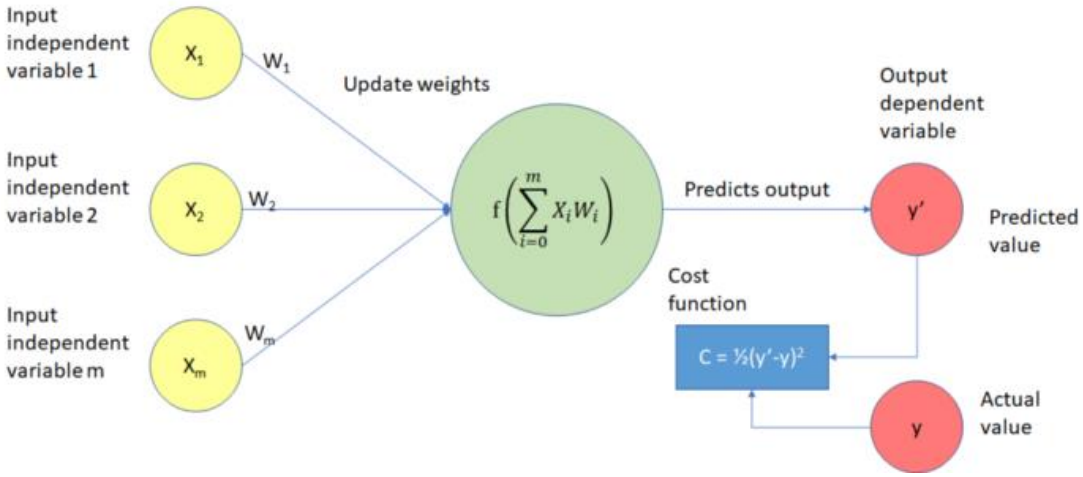
Στο άθροισμα αυτό στην συνέχεια προστίθεται μια τιμή b (bias) η οποία είναι ξεχωριστή για κάθε νευρώνα και υπάρχει για να επηρεάζει το πόσο «δυνατά» συνδέεται ένας νευρώνας με τους νευρώνες του προηγούμενου επιπέδου. Οπότε το τελικό άθροισμα προκύπτει ως:

$$s = \sum_{i=1}^{n} w_i \times x_i + b \qquad \text{Εξ. (2.2)}$$

Το άθροισμα αυτό μπορεί να πάρει θεωρητικά οποιαδήποτε τιμή.

Στην συνέχεια η τιμή αυτή περνάει στην συνάρτηση ενεργοποίησης f() [9], η οποία ορίζεται από την αρχιτεκτονική του δικτύου, και αυτή αναλόγως την τιμή ενεργοποιείται ή όχι. Στην

παραπάνω εικόνα έχει χρησιμοποιηθεί σαν παράδειγμα η σιγμοειδής συνάρτηση η οποία παίρνει τιμές από 1 (“ON”) έως 0 (“OFF”). Οπότε το αποτέλεσμα περνάει από αυτήν την συνάρτηση και σε συνδυασμό με την πραγματική τιμή, δηλαδή αυτή που θα θέλαμε το δίκτυο μας να έχει μαντέψει, δίνονται σε μια συνάρτηση απώλειας (loss function ή cost function), η οποία μας δίνει το πόσο «καλή» ή πόσο «κακή» επίδοση είχε το δίκτυο.



Εικόνα 2.4: Λειτουργία μεμονωμένου νευρώνα

Στην παραπάνω εικόνα (Εικόνα 2.4) έχει χρησιμοποιηθεί σαν παράδειγμα η συνάρτηση απώλειας του μέσου τετραγωνικού σφάλματος.

$$MSE = \frac{1}{2}(y_i - \hat{y_i})^2 s \qquad \text{Εξ. (2.3)}$$

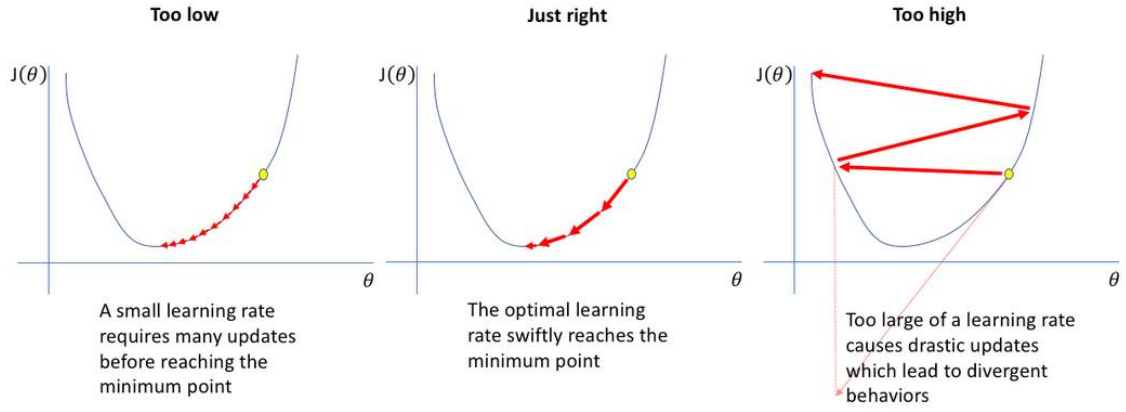
Και επί του συνόλου των νευρώνων.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 s \qquad \text{Εξ. (2.4)}$$

Αν υποθέσουμε λοιπόν ότι τα βάρη που συνδέουν το ένα επίπεδο με το επόμενο κατανέμονται αρχικά τυχαία, τότε πρέπει να υπάρχει ένας τρόπος με τον οποίο κατά την διάρκεια της εκπαίδευσης το νευρωνικό δίκτυο βρίσκει έναν τρόπο να αλλάζει τα βάρη προκειμένου αυτά να δίνουν όλο και καλύτερο αποτέλεσμα. Πρέπει δηλαδή να υπάρχει κάποιος αλγόριθμος που να βελτιώνει τα αποτελέσματα, δίνοντας συνεχώς και «καλύτερα» βάρη, ο αλγόριθμος αυτός είναι ο αλγόριθμος βελτιστοποίησης (optimizer)[13].

Ουσιαστικά ο βελτιστοποιητής είναι μια συνάρτηση που συνδέει το κόστος με τα βάρη, αυτό σημαίνει ότι υπάρχει ένα βάρος για το οποίο η συνάρτηση, δηλαδή το κόστος, παίρνει την μικρότερη τιμή, δηλαδή έχουμε το καλύτερο αποτέλεσμα. Τότε όσο πλησιάζουμε το κατώτερο αυτό σημείο της συνάρτησης τόσο βελτιώνουμε το αποτέλεσμα, δηλαδή όσο ο ρυθμός μεταβολής του κόστους είναι αρνητικός, δηλαδή η παράγωγος της συνάρτησης, θα βελτιώνουμε το αποτέλεσμα μας. Ο βελτιστοποιητής επίσης ορίζει πόσο μεγάλα ή πόσο μικρά θα είναι τα βήματα που κάνουμε στην αλλαγή του βάρους προκειμένου αυτό να δώσει

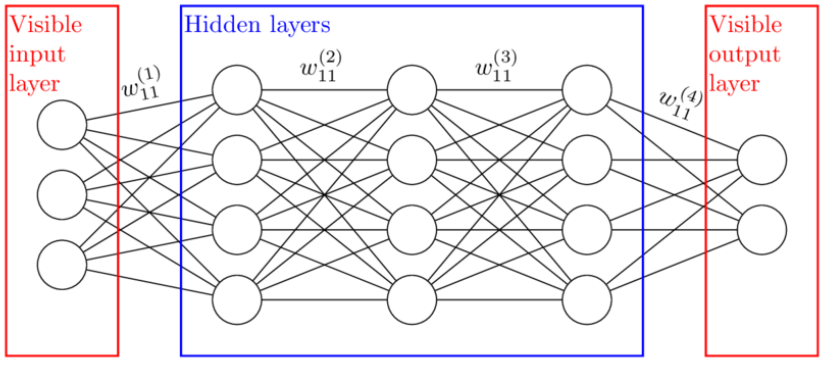
μικρότερο κόστος από ότι έδωσε στην προηγούμενη προσπάθεια. Το πόσο μεγάλα ή πόσο μικρά είναι αυτά τα βήματα ονομάζεται learning rate (Εικόνα 2.5).



Εικόνα 2.5: Ρυθμός μάθησης

2.3 Επίπεδα (Layers)

Φυσικά οι παραπάνω διαδικασίες που περιγράψαμε αφορούν την λειτουργία ενός νευρώνα και ο τρόπος με τον οποίο βελτιστοποιείτε το βάρος αφορά μόνο την βελτιστοποίηση ενός βάρους. Στην πραγματικότητα ένα νευρωνικό δίκτυο αποτελείται από πολλά επίπεδα (layers), στο καθένα από τα οποία υπάρχουν πολλοί νευρώνες που ο καθένας δέχεται πολλά βάρη και πολλές τιμές και επιστρέφει πολλά βάρη και πολλές τιμές στους νευρώνες του επόμενου επιπέδου. Το επίπεδο στο οποίο δίνονται τα αρχικά δεδομένα ονομάζεται input layer και το επίπεδο στο όποιο βλέπουμε τα αποτελέσματα μας ονομάζεται output layer. Τα ενδιάμεσα επίπεδα, στα οποία γίνονται οι περισσότερες ενέργειες για την βελτίωση του δικτύου και την πραγματοποίηση της εκπαίδευσης ονομάζονται κρυφά επίπεδα (hidden layers) (Εικόνα 2.6).



Εικόνα 2.6: Κρυφά επίπεδα

Από την παραπάνω εικόνα συμπεραίνουμε ότι όλα τα επίπεδα του δικτύου μπορούν να γραφτούν με την μορφή πινάκων. Έτσι τα αρχικά δεδομένα x_i μπορούν να γραφτούν ως:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Και τα βάρη που περνάνε από το k επίπεδο στο επόμενο επίπεδο k+1 μπορούν να γραφτούν ως:

$$W = \begin{bmatrix} w_{0,0}^k & w_{0,1}^k & \cdots & w_{0,n}^k \\ w_{1,0}^k & w_{1,1}^k & & w_{1,n}^k \\ \vdots & & \ddots & \vdots \\ w_{j,0}^k & w_{j,1}^k & \cdots & w_{j,n}^k \end{bmatrix}$$

Όπου j, το σύνολο των νευρώνων στο επίπεδο k+1 και n το σύνολο των νευρώνων στο επίπεδο k.

Επίσης έστω οι τιμές των νευρώνων $\alpha_j^{(k)}$ και οι τιμές των bias b_n:

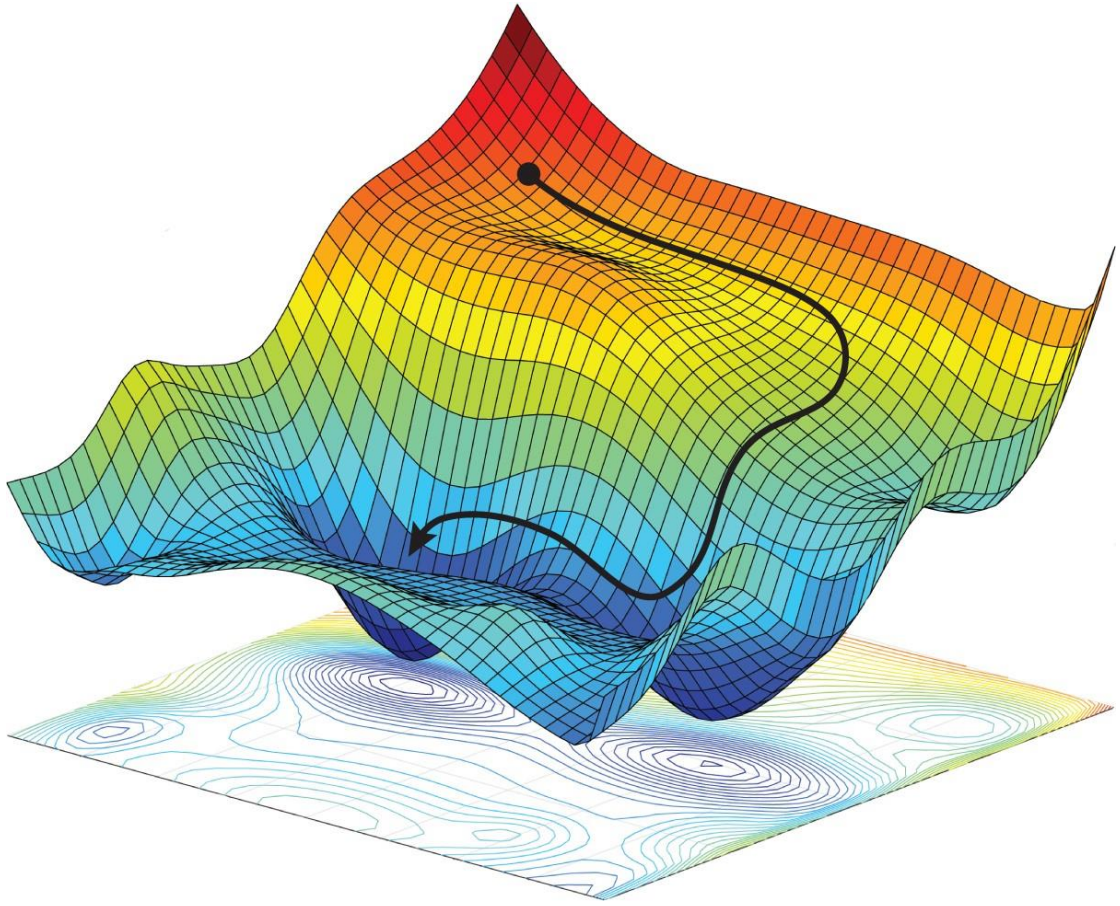
$$a = \begin{bmatrix} a_0^{(k)} \\ a_1^{(k)} \\ \vdots \\ a_j^{(k)} \end{bmatrix} \text{ και } b^{(k)} = \begin{bmatrix} b_0{}^{(k)} \\ b_1{}^{(k)} \\ \vdots \\ b_j{}^{(k)} \end{bmatrix}$$

Τότε οι τιμές των επόμενων νευρώνων θα δίνονται ως εξής:

$$a^{(k+1)} = \sigma\left(W^{(k)} * a^{(k)} + b^{(k)}\right) \qquad \text{Εξ. (2.5)}$$

Αν λοιπόν θέλουμε να εξετάσουμε ένα δίκτυο ίδιων διαστάσεων με αυτό της (Εικόνα 2.6), τότε θα καταλάβουμε ότι μεταξύ του πρώτου επιπέδου και του δεύτερου έχουμε 3 x4 βάρη και 4 bias , μεταξύ του δεύτερου και του τρίτου, έχουμε 4x4 βάρη και 4 bias, το τρίτο με το τέταρτο, έχουν 4x4 βάρη και 4 bias και τέλος μεταξύ του τέταρτου και του τελευταίου επίπεδου έχουμε 4x2 βάρη και 2 bias. Όλα αυτά μαζί μας δίνουν 66 παραμέτρους, το οποίο σημαίνει ότι το τελικό μας κόστος επηρεάζεται από 66 παραμέτρους.

Όπως αναφέραμε στην προηγούμενη παράγραφο (§2.2), ο βελτιστοποιητής ορίζει μια καμπύλη-συνάρτηση με την οποία βρίσκει πόσο πρέπει να μεταβληθεί η τιμή μιας μεταβλητής του δικτύου για να έχει αυτό μικρότερο κόστος. Στην πραγματικότητα όμως το δίκτυο μπορεί να έχει πολλές μεταβλητές, για παράδειγμα 66. Αυτό σημαίνει ότι υπάρχουν και πολλές καμπύλες που πρέπει να ακολουθούν οι βελτιώσεις των μεταβλητών του δικτύου που επηρεάζουν το κόστος, ο βελτιστοποιητής λοιπόν στην πραγματικότητα ορίζει μια επιφάνεια, η οποία περιγράφει τις αλλαγές των τιμών του κόστους ανάλογα με τις τιμές των μεταβλητών και προσπαθεί να βρει το πιο «βαθύ» σημείο της επιφάνειας (Εικόνα 2.7).



Εικόνα 2.7: Εύρεση μηδενικού ρυθμού μεταβολής

Για να βρεθεί αυτό το σημείο χρειάζεται αρχικά να βρούμε τον ρυθμό μεταβολής του κόστους του δικτύου και ακολουθώντας το «μονοπάτι» όπου αυτός είναι συνέχεια αρνητικός βρίσκουμε το πιο «βαθύ» σημείο. Σε μια συνάρτηση όπως η συνάρτηση του κόστους που έχει πολλές μεταβλητές ο ρυθμός μεταβολής της ονομάζεται βαθμίδα και συμβολίζεται με το γνωστό σύμβολο ανάδελτα ∇.

Αν λοιπόν το κόστος ενός δικτύου είναι C τότε η βαθμίδα αυτής της συνάρτησης είναι ∇C και μας δίνει το πόσο πρέπει να αλλάξουν οι μεταβλητές του δικτύου για να βελτιωθεί το κόστος. Αυτή η διαδικασία αλλαγής των μεταβλητών για να βελτιωθεί το κόστος ονομάζεται οπισθοδιάδοση (backpropagation)[7][8].

2.4 Βελτίωση Αποτελεσμάτων

Σε συνδυασμό με την οπισθοδιάδοση υπάρχουν και άλλοι τρόποι για να βελτιωθούν τα αποτελέσματα ενός νευρωνικού δικτύου. Αρχικά το κόστος ενός δικτύου επηρεάζετε από τα δεδομένα εισόδου, από τα βάρη και από τα bias. Υπάρχει λοιπόν η επιλογή να αλλάξουν όποιες από αυτές τις μεταβλητές χρειάζεται προκειμένου να μειωθεί το κόστος.

2.4.1 Αρχικοποιητής (Initializer)

Ο initializer ή αλλιώς αρχικοποιητής ορίζει τα βάρη σε ένα μοντέλο νευρωνικού δικτύου, αυτά μπορεί να είναι τυχαία είτε μπορεί να ακολουθούν κάποιους κανόνες, το οποίο εξαρτάται από τον αρχιτέκτονα του μοντέλου. Αυτό συμβαίνει ανάμεσα σε όλες τις φορές που

θα τρέξει το δίκτυο, όποτε σε περιπτώσεις όπως τους αλγόριθμους οπισθοδιάδοσης αυτά εκτός της πρώτης φοράς που ορίζονται από τον initializer μετά ορίζονται από το ίδιο το δίκτυο.

2.4.2 Τακτοποιητής (Regularizer)

Η Τακτοποίηση (Regularization) [11] είναι η διαδικασία που ακολουθείτε για να ελαττωθεί το σφάλμα και πετυχαίνετε προσαρμόζοντας μια συνάρτηση στα δεδομένα εισόδου για αποφευχθεί η υπερπροσαρμογή (overfitting).

Έστω ένα σύνολο δεδομένων εκπαίδευσης X= (x_1, x_2, x_3 …x_n) όπου είναι τυχαίες τιμές στο διάστημα [0,1]. Τότε ορίζουμε μια συνάρτηση f, έστω f= sin(2x), στην οποία δίνουμε τιμές από το σύνολο X και μας επιστρέφει άλλες τιμές, στις οποίες προσθέτουμε και έναν θόρυβο , συνήθως Γκαουσιανό (Gaussian Noise) δηλαδή που ακολουθεί την κατανομή του Gauss. Η κάθε τιμή αυτή που προκύπτει από την συνάρτηση f και στην οποία έχει προστεθεί ο θόρυβος, αποτελεί μια μεταβλητή στόχο (target variable).

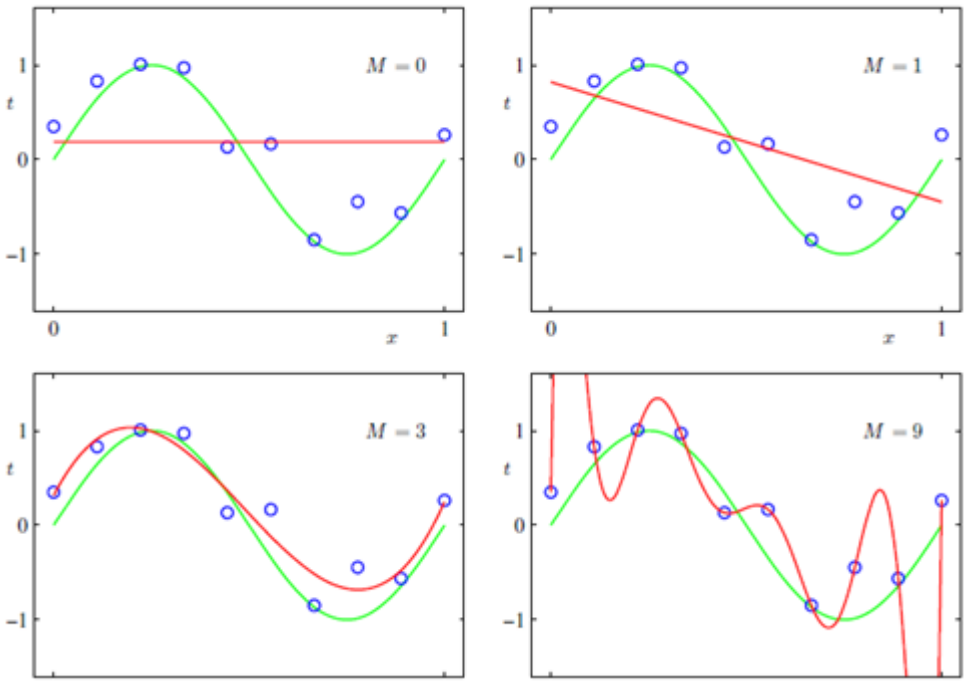
Ο στόχος είναι να βρεθεί μια συνάρτηση την οποία ακολουθούν οι μεταβλητές στόχοι για να είναι δυνατόν μετά να προβλεφθούν οι μεταβλητές στόχοι ενός καινούργιου συνόλου τιμών X2. Το πρόβλημα που υπάρχει σε αυτό το παράδειγμα είναι ότι τα δεδομένα στόχοι του αρχικού συνόλου έχουν «αλλοιωθεί» με την προσθήκη του θορύβου, οπότε είναι δύσκολο να προσαρμόσουμε τα καινούργια δεδομένα στην συνάρτηση f, στο παράδειγμα μας sin(2x).

Αρχικά προσπαθούμε να βρούμε ένα πολυώνυμο που να προσαρμόζεται στα δεδομένα μας ώστε:

$$Y(X,W) = w_0 + w_1 * x + w_2 * x^2 + \cdots + w_n * x^n \qquad \text{Εξ. (2.6)}$$

Η παραπάνω συνάρτηση είναι μη-γραμμική για X και γραμμική για W. Οπότε θα εκπαιδεύουμε τα δεδομένα μας σε αυτή την συνάρτηση ώστε να μας δώσει τα βάρη w για τα οποία η συνάρτηση θα ελαχιστοποιεί το σφάλμα στις προβλέψεις των μεταβλητών στόχων. Έστω η συνάρτηση σφάλματος είναι η συνάρτηση μέσου τετραγωνικού σφάλματος (Εξ. (2.4)). Τότε στο σημείο που η παράγωγος της συνάρτησης σφάλματος είναι μηδέν θα βρίσκεται η τιμή του βάρους w που ελαχιστοποιεί το σφάλμα.

Το πρόβλημα πλέον είναι τι βαθμός να χρησιμοποιηθεί στο παραπάνω πολυώνυμο (Εξ. (2.6)) ώστε να προσαρμόζεται καλύτερα στα δεδομένα εκπαίδευσης. Παρακάτω (Εικόνα 2.8) παρουσιάζεται διάγραμμα στο οποίο είναι οι τιμές στόχοι συναρτήσει των τιμών εκπαίδευσης. Οι μπλε κύκλοι αποτελούν τα δεδομένα εκπαίδευσης X2. Η πράσινη καμπύλη είναι το αναμενόμενο πολυώνυμο που θέλουμε να προβλέψουμε, δηλαδή είναι οι τιμές στόχοι που προέκυψαν για τις τιμές εκπαίδευσης X. Η κόκκινη καμπύλη είναι το πολυώνυμο που προσπαθούμε να προσαρμόσουμε στις τιμές X2 για διάφορους βαθμούς (M) του πολυωνύμου.



Εικόνα 2.8: Υπερπροσαρμογή

Από την Εικόνα (Εικόνα 2.8) συμπεραίνουμε ότι παρόλο που για M=9 η συνάρτηση προσαρμόζεται στα δεδομένα εκπαίδευσης τέλεια, το αποτέλεσμα δεν μοιάζει καθόλου με την πράσινη καμπύλη που είχαμε ορίσει στην αρχή. Αυτό συνέβη επειδή η συνάρτηση εκπαιδεύτηκε για όλες τις τιμές των μεταβλητών στόχους συμπεριλαμβανομένου και του θορύβου και γι' αυτό και επέτυχε να κάνει την σωστή πρόβλεψη. Αυτή η συνάρτηση θα δώσει μηδενικό σφάλμα στην εκπαίδευση αλλά θα δώσει μεγάλα σφάλματα στην πρόβλεψη τιμών. Αυτή η υπερβολική ταύτιση με τις μεταβλητές στόχους ονομάζεται υπερπροσαρμογή

Για να αποφευχθεί αυτό το σφάλμα χρησιμοποιείται ο Τακτοποιητής (Regularizer). Η τακτοποίηση είναι μια τεχνική που χρησιμοποιείται για ρυθμίσει την συνάρτηση, προσθέτοντας έναν ακόμα όρο στην συνάρτηση απώλειας. Αυτός ο έξτρα όρος βοηθάει την συνάρτηση να μην υπερπροσαρμόζεται, δηλαδή οι συντελεστές να μην παίρνουν υπερβολικές τιμές. Υπάρχουν 2 ειδών όροι που μπορούν να προστεθούν στην συνάρτηση απώλειας, αυτοί είναι ο L1 Lasso Regression και ο L2 Ridge Regression [12].

- L1 Lasso Regression, δηλαδή Least Absolute Shrinkage and Selection Operator, λιγότερο απόλυτος συρρικνωτής και διαχειριστής επιλογών. Ο τελεστής L1 (ο δεύτερος όρος της Εξ. (2.7)) ονομάζεται έτσι επειδή στην ουσία διαλέγει τις λιγότερο «σημαντικές» τιμές των μεταβλητών και τις μηδενίζει, αφήνοντας τις υπόλοιπες τιμές ανεπηρέαστες.

$$COST\ FUNCTION + \lambda * \sum_{j=1}^{p} |\beta_j| \qquad \text{Εξ. (2.7)}$$

- L2 Ridge Regression, ο τελεστής αυτός επηρεάζει όλες τις τιμές των μεταβλητών σε κάποιον βαθμό για να τις προσαρμόσει καλύτερα και να μειώσει τις πιθανότητες να γίνει

υπερπροσαρμογή. Παρακάτω δίνεται ο τελεστής ως επιπλέον όρος στην συνάρτηση απώλειας (ο δεύτερος όρος της Εξ. (2.8)).

$$COST\ FUNCTION + \lambda * \sum_{j=1}^{p} \beta_j^2 \qquad \text{Εξ. (2.8)}$$

Όταν οι τελεστές L1 και L2 υπάρχουν ταυτόχρονα σαν επιπλέον όροι της συνάρτησης απώλειας, τότε λέμε ότι έχουμε ελαστική τακτοποίηση δικτύου (Elastic net regularization).

2.4.3 Κανονικοποιητής (Normalizer)

Η κανονικοποίηση (Normalization) [10] είναι μια τεχνική που εφαρμόζεται στα δεδομένα εισόδου ενός επιπέδου του νευρωνικού δικτύου. Ο σκοπός της κανονικοποίησης είναι να αλλάξει τις τιμές των δεδομένων εισόδου με ομοιόμορφο τρόπο, χωρίς να επηρεάζει δηλαδή τις διαφορές στα εύρη τιμών, προκειμένου να βελτιώσει διάφορα προβλήματα που προκύπτουν στο δίκτυο ανάλογα με τον κανονικοποιητή (normalizer).

2.5 Βήματα εκπαίδευσης

Η εκπαίδευση, δηλαδή η βελτίωση των παραμέτρων του δικτύου, μπορεί να γίνει σε βήματα ή μπορεί να γίνει κατευθείαν. Η κάθε μια από τις 2 περιπτώσεις έχει πλεονεκτήματα και μειονεκτήματα. Η εκπαίδευση του δικτύου χωρίζεται σε παρτίδες (batch) και εποχές (epochs), όπου το πρώτο αποτελεί μέρος του δεύτερου.

2.5.1 Παρτίδα (Batch):

Η παρτίδα είναι μια υπερπαράμετρος του δικτύου που υποδηλώνει το σύνολο των δεδομένων εισόδου με τα οποία θα δουλέψει το δίκτυο προτού βελτιώσει τις τιμές των μεταβλητών του. Η παρτίδα μπορεί να παρομοιαστεί με μια for{ } λούπα προβλέψεων, που επαναλαμβάνεται για ένα ή περισσότερα δεδομένα. Στο τέλος κάθε παρτίδας, οι προβλέψεις συγκρίνονται με τις πραγματικές τιμές, υπολογίζεται η απώλεια και ο αλγόριθμος βελτιώνει τις μεταβλητές του.

2.5.2 Εποχή (Epoch):

Η εποχή είναι επίσης μια υπερπαράμετρος του δικτύου που δηλώνει τις φορές που θα περάσει ο αλγόριθμος μάθησης από ολόκληρη την βάση δεδομένων εκπαίδευσης. Μια εποχή ουσιαστικά σημαίνει ότι ένα δεδομένο εισόδου είχε την «ευκαιρία» του να βελτιώσει τις παραμέτρους του δικτύου. Η αντίστοιχη παρομοίωση που έγινε και για τις παρτίδες θα ήταν, αν πάλι σκεφτούμε την εποχή σαν μια λούπα for{ } που περνάει όσες φόρες ορίσει ο αρχιτέκτονας του δικτύου από ολόκληρη την βάση δεδομένων εισόδου. Στο εσωτερικό αυτής της for{ } υπάρχει άλλη μια λούπα for{ } η οποία περνάει από έναν αριθμό δεδομένων, δηλαδή η παρτίδα.

Ο αριθμός των παρτίδων πρέπει να είναι συγκεκριμένος και εξαρτάται από το μέγεθος της παρτίδας (batch size) που έχει ορίσει πάλι ο αρχιτέκτονας και το σύνολο των δεδομένων. Αντίθετα ο αριθμός των εποχών εξαρτάται μόνο από τον αρχιτέκτονα και μπορεί να είναι μεταξύ του 1 και του άπειρου. Δηλαδή, κάποιος μπορεί να «περάσει» από ολόκληρη την βάση δεδομένων εισόδου όσες φορές θέλει, τα κριτήρια με τα οποία θα επιλέξει να

σταματήσει είναι διάφορα και ο τρόπος με τον οποίο θα σταματήσει είναι μια συνάρτηση που ονομάζεται early_stopping.

2.6 Εφαρμογές βαθιάς μάθησης για ταξινόμηση εικόνων

Η Βαθιά Μάθηση είναι ένας τύπος Μηχανικής Μάθησης που είναι εμπνευσμένος από την δομή του ανθρώπινου εγκεφάλου. Οι αλγόριθμοι βαθιάς μάθησης προσπαθούν να καταλήξουν στα ίδια αποτελέσματα που θα κατέληγε και ένας άνθρωπος αναλύοντας συνεχώς τα δεδομένα μεσα από μια δοσμένη λογική δομή. Για να το καταφέρουν αυτό χρησιμοποιούν μια πολυεπίπεδη δομή αλγορίθμων που ονομάζεται Νευρωνικό Δίκτυο.

Ο σχεδιασμός των νευρωνικών δικτύων βασίζεται στη δομή του ανθρώπινου εγκεφάλου. Ακριβώς όπως χρησιμοποιούμε τον εγκέφαλό μας για τον εντοπισμό αντικειμένων και την ταξινόμηση διαφορετικών τύπων πληροφοριών, τα νευρωνικά δίκτυα μπορούν να διδαχθούν να εκτελούν τις ίδιες εργασίες στα δεδομένα.

Τα επιμέρους στρώματα νευρωνικών δικτύων μπορούν επίσης να θεωρηθούν ως ένα είδος φίλτρου που λειτουργεί ξεχωρίζοντας τα χονδροειδή λάθη από τα σωστά συμπεράσματα, αυξάνοντας την πιθανότητα ανίχνευσης και εξόδου ενός σωστού αποτελέσματος.

Τα νευρωνικά δίκτυα μας επιτρέπουν να εκτελέσουμε πολλές εργασίες, όπως ομαδοποίηση, ταξινόμηση ή παλινδρόμηση. Με τα νευρωνικά δίκτυα, μπορούμε να ομαδοποιήσουμε ή να ταξινομήσουμε δεδομένα χωρίς ετικέτα σύμφωνα με τις ομοιότητες μεταξύ των δειγμάτων σε αυτά τα δεδομένα. Ή στην περίπτωση της ταξινόμησης, μπορούμε να εκπαιδεύσουμε το δίκτυο σε ένα σύνολο δεδομένων με ετικέτα και να ταξινομήσουμε τα δείγματα σε αυτό το σύνολο δεδομένων σε διαφορετικές κατηγορίες.

Υπάρχουν διάφορες μέθοδοι με τις οποίες μπορούν να αναγνωριστούν αντικείμενα σε μια εικόνα από υπολογιστή και σε διαφορετικό επίπεδο αναγνώρισης:

- Image Classification

Η πιο βασική εφαρμογή της Όρασης Υπολογιστή είναι η ταξινόμηση εικόνων με βάση την ετικέτα τους, η οποία προβλέπεται από τον ίδιο και υποδηλώνει το αντικείμενο που απεικονίζεται σε αυτήν. Στην Ταξινόμηση εικόνων υποθέτουμε ότι στην εικόνα απεικονίζεται μόνο ένα αντικείμενο και όχι πολλαπλά. Για να είναι δυνατή η ταξινόμηση χρειάζεται ένα συγκεκριμένο είδος βαθιού νευρωνικού δικτύου το οποίο είναι το συνελικτικό δίκτυο, το οποίο συνήθως αναφέρεται ως CNN ή ConvNet [18][19]. Είναι ένα βαθύ, feed-forward τεχνητό νευρωνικό δίκτυο. Τα μοντέλα ονομάζονται "feed-forward" επειδή οι πληροφορίες κυλούν μέσα από το μοντέλο, δηλαδή η πληροφορία δεν επιστρέφει και ξανά εισάγεται σαν δεδομένο στο ίδιο το μοντέλο.

- Classification with Localization

Κατά τον εντοπισμό του αντικειμένου με βάση το οποίο γίνεται η ταξινόμηση των εικόνων [16], αναμένουμε επίσης ότι ο υπολογιστής θα εντοπίσει πού ακριβώς υπάρχει το αντικείμενο στην εικόνα. Αυτός ο εντοπισμός τυπικά εφαρμόζεται χρησιμοποιώντας ένα πλαίσιο οριοθέτησης το οποίο μπορεί να οριστεί με ορισμένες αριθμητικές παραμέτρους σε σχέση

με το όριο εικόνας. Και σε αυτή την περίπτωση, η υπόθεση είναι να έχουμε μόνο ένα αντικείμενο ανά εικόνα.

- Object Detection

Η ανίχνευση αντικειμένου είναι ουσιαστικά το επόμενο στάδιο του εντοπισμού, στο οποίο δεν είναι απαραίτητο ότι η εικόνα περιέχει μόνο ένα αντικείμενο, αλλά μπορεί να περιέχει πολλά. Ο σκοπός σε αυτή την διαδικασία είναι η ταξινόμηση και ο εντοπισμός όλων των αντικειμένων στην εικόνα.

Οι άνθρωποι ρίχνουν μια ματιά σε μια εικόνα και γνωρίζουν αμέσως που βρίσκονται αντικείμενα σε αυτήν, τι είναι, πόσα είναι και πώς αλληλοεπιδρούν με το περιβάλλον τους. Το ανθρώπινο οπτικό σύστημα είναι γρήγορο και ακριβές, επιτρέποντας μας να εκτελούμε σύνθετες λειτουργίες, όπως η οδήγηση, με λίγη συγκέντρωση σε αυτό που κάνουμε. Γρήγοροι, ακριβείς αλγόριθμοι για τον εντοπισμό αντικειμένων θα επιτρέψουν στους υπολογιστές να οδηγούν αυτοκίνητα χωρίς ειδικούς αισθητήρες.

Τα τρέχοντα συστήματα ανίχνευσης επαναχρησιμοποιούν τους ταξινομητές για να καταφέρουν να κάνουν ανίχνευση. Για την ανίχνευση ενός αντικειμένου, αυτά τα συστήματα λαμβάνουν ένα ταξινομητή για αυτό το αντικείμενο και τον εφαρμόζει σε διάφορες τοποθεσίες της εικόνας και σε διαφορετικές κλίμακες. Συστήματα όπως το Deformable Parts Models (DPM) χρησιμοποιούν μια προσέγγιση συρόμενου παραθύρου, όπου ο ταξινομητής εκτελείται σε ομοιόμορφα τοποθετημένα κομμάτια σε ολόκληρο το σύνολο της εικόνας.

Ακόμα και πιο σύγχρονες προσεγγίσεις του προβλήματος , όπως τα συστήματα R-CNN , πρώτα παράγουν πιθανές θέσεις που μπορεί να υπάρχουν τα αντικείμενα στην εικόνα και μετά χρησιμοποιούν τον ταξινομητή στις θέσεις αυτές. Μετά την ταξινόμηση , χρησιμοποιούνται μέθοδοι με τις οποίες γίνεται ένας επανακαθορισμός των πλαισίων οριοθέτησης (bounding-boxes), επαλείφονται οι διπλές ανιχνεύσεις και επαναστοχεύονται οι θέσεις των αντικειμένων. Όμως η διαδικασία αυτή είναι αργή και δύσκολη καθώς για κάθε αντικείμενο πρέπει να εκπαιδευτεί ξεχωριστά το νευρωνικό.

Το YOLO (You Only Look Once) [17] είναι ένα ενιαίο συνελικτικό δίκτυο που προβλέπει ταυτόχρονα πολλαπλά πλαίσια οριοθέτησης και πιθανές κλάσης για αυτά τα πλαίσια. Το YOLO εκπαιδεύεται σε πλήρεις εικόνες και βελτιστοποιεί άμεσα την απόδοση ανίχνευσης. Αυτό το ενοποιημένο μοντέλο έχει πολλά οφέλη έναντι των παραδοσιακών μεθόδων ανίχνευσης αντικειμένων.

Αρχικά, το YOLO είναι πολύ γρήγορο, εφόσον η ανίχνευση γίνεται σε μια στιγμή σε μια εικόνα δεν χρειάζονται πολύπλοκοι κώδικες, απλά χρησιμοποιείται ένα νευρωνικό για να κάνει την ανίχνευση εκείνη την στιγμή. Δεύτερον, το YOLO κάνει προβλέψεις σε ολόκληρη την εικόνα. Αντί των παραδοσιακών τεχνικών που βλέπανε την εικόνα κομμάτι κομμάτι και κάνανε στο καθένα από αυτά ανίχνευση αντικειμένων, το YOLO βλέπει ολόκληρη την εικόνα με αποτέλεσμα , λάθη των παλιών τεχνικών όπως η ταξινόμηση κομματιών του φόντο των αντικειμένων ως άλλα αντικείμενα δεν συμβαίνουν. Τέλος το YOLO μπορεί να εκπαιδευτεί με πραγματικές εικόνες και να κάνει αναγνώριση σε έργα τέχνης. Επειδή ακριβώς το

YOLO βλέπει ολόκληρη την εικόνα , είναι πολύ πιο δύσκολο να καταλήξει σε σφάλμα αν τα δεδομένα που του δίνονται δεν είναι τα αναμενόμενα.

- Semantic Segmentation

Ο στόχος της σημασιολογικής κατάτμησης είναι να κατηγοριοποίηση όλα τα εικονοστοιχεία μιας εικόνας σύμφωνα με την κατηγορία που ανήκει το αντικείμενο το οποίο αναπαριστά το κάθε εικονοστοιχείο. Επειδή η ταξινόμηση γίνεται σε επίπεδο εικονοστοιχείου, η διαδικασία αυτή συχνά ονομάζεται και πυκνή πρόβλεψη. Σε αντίθεση με τις προηγούμενες διαδικασίες, η σημασιολογική κατάτμηση δεν δίνει σαν αποτέλεσμα απλά κατηγορίες ή πλαίσια οριοθέτησης. Το αποτέλεσμα σε αυτή την περίπτωση είναι μια εικόνα, συνήθως ίδιων διαστάσεων με την αρχική εικόνα, όπου κάθε εικονοστοιχείο είναι κατηγοριοποιημένο σε μια συγκεκριμένη κατηγορία.

Το πρόβλημα της σημασιολογικής κατάτμησης συνήθως αντιμετωπίζεται με την χρήση ενός πλήρους συνελικτικού δικτύου (Fully Convolutional Network (FCN)), όπως το U-NET .

2.7 Συνελικτικά δίκτυα

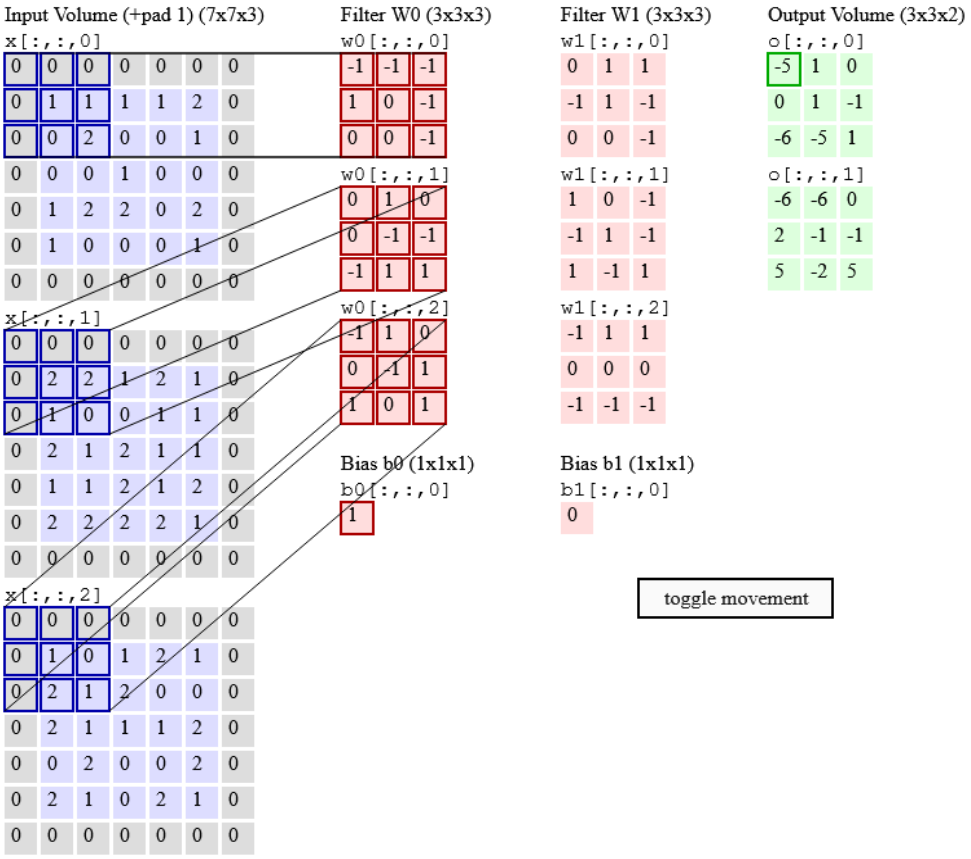
Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) ή CNN [18][19], είναι εμπνευσμένα από τον βιολογικό οπτικό φλοιό. Ο φλοιός έχει μικρές περιοχές κυττάρων που είναι ευαίσθητα στις συγκεκριμένες περιοχές του οπτικού πεδίου. Αυτή η ιδέα επεκτάθηκε μετά από ένα πείραμα που έγινε από τους Hubel και Wiesel το 1962. Σε αυτό το πείραμα, οι ερευνητές έδειξαν ότι ορισμένοι μεμονωμένοι νευρώνες στον εγκέφαλο ενεργοποιήθηκαν ή πυροδοτήθηκαν μόνο παρουσία ακμών συγκεκριμένου προσανατολισμού όπως κάθετες ή οριζόντιες ακμές. Για παράδειγμα, ορισμένοι νευρώνες πυροδοτούνται όταν εκτίθενται σε κάθετες πλευρές και μερικοί όταν εμφανίζονται σε οριζόντια ακμή. Ο Hubel και ο Wiesel διαπίστωσαν ότι όλοι αυτοί οι νευρώνες ήταν καλά ταξινομημένοι με στήλη και ότι μαζί ήταν σε θέση να παράγουν οπτική αντίληψη. Αυτή η ιδέα των εξειδικευμένων εξαρτημάτων μέσα σε ένα σύστημα που έχει συγκεκριμένες εργασίες είναι αυτή που χρησιμοποιούν επίσης οι μηχανές και που μπορούμε επίσης να βρούμε πίσω από τα CNN. Ουσιαστικά η διαφορά είναι ότι τα συνελικτικά δίκτυα έχουν 2 διαφορετικά layer, το Convolutional Layer και το Pool Layer.

2.7.1 Συνελικτικό επίπεδο (Convolutional layer)

Τα Συνελικτικά Δίκτυα είναι ένας ειδικός τύπος νευρωνικών δικτύων σχεδιασμένος ειδικά για δισδιάστατα δεδομένα εισόδου, αν και μπορούν να χρησιμοποιηθούν και για μονοδιάστατα ή τρισδιάστατα δεδομένα. Το πιο βασικό στοιχείο ενός συνελικτικού δικτύου είναι το συνελικτικό επίπεδο στο οποίο συμβαίνει η διαδικασία της συνέλιξης. Στην περίπτωση των συνελικτικών δικτύων, συνέλιξη είναι η γραμμική διαδικασία που περιλαμβάνει τον πολλαπλασιασμό ενός σετ βαρών και δεδομένων εισόδου. Επειδή η τεχνική αυτή είναι σχεδιασμένη για 2 διαστάσεων δεδομένα, ο πολλαπλασιασμός γίνεται μεταξύ ενός πίνακα που αποτελεί τα δεδομένα και ενός δισδιάστατου πίνακα βαρών, που ονομάζεται φίλτρο ή kernel.

Οι υπολογιστές αντιλαμβάνονται τις εικόνες ως ένα σύνολο εικονοστοιχείων. Τα εικονοστοιχεία στις εικόνες συνήθως σχετίζονται μεταξύ τους. Για παράδειγμα, μια συγκεκριμένη

ομάδα εικονοστοιχείων μπορεί να συμβολίζει ένα άκρο σε μια εικόνα ή κάποιο άλλο μοτίβο, τα συνελικτικά δίκτυα χρησιμοποιούν τέτοια μοτίβα για την ταξινόμηση εικόνων. Μια συνέλιξη πολλαπλασιάζει ένα πίνακα ,με κελιά τις τιμές των pixel, με ένα φίλτρο ή «πυρήνα» και αθροίζει τις τιμές που προκύπτουν από τον πολλαπλασιασμό, δίνοντας στην συνέχεια αυτήν την τιμή σε κελί πίνακα διαστάσεων n_{out}. Στη συνέχεια, η συνέλιξη μετακινείται στο επόμενο εικονοστοιχείο και επαναλαμβάνει την ίδια διαδικασία μέχρι να καλυφθούν όλα τα εικονοστοιχεία της εικόνας. Αυτή η διαδικασία απεικονίζεται παρακάτω.



Εικόνα 2.9: Διαδικασία συνέλιξης

Η διαδικασία της συνέλιξης χρειάζεται 2 δεδομένα εισόδου:

- Ένα τρισδιάστατο δεδομένο (εικόνα) διαστάσεων $n \times n \times channels$
- Ένα σετ φίλτρων kernel (k filters) το καθένα διαστάσεων $f \times f \times channels$, όπου f τυπικά είναι ίσο με 3 ή 5.

Το αποτέλεσμα της διαδικασίας της συνέλιξης είναι ένας επίσης τρισδιάστατος πίνακας διαστάσεων $n_{out} \times n_{out} \times c$, όπου n_{out} δίνεται από την παρακάτω σχέση:

$$n_{out} = \left[\frac{n_{in} + 2p - k}{s}\right] + 1$$

όπου,

n_{in} οι διαστάσεις της δεδομένης εικόνας

p το μέγεθος του περιγράμματος που θα προστεθεί σε κάθε διάσταση της εικόνας

k η διάσταση του φίλτρου

s το πόσο ολισθαίνει το φίλτρο πάνω στην εικόνα

c ο αριθμός των φίλτρων στο σετ

Το φίλτρο αυτό μπορεί όπως είπαμε να είναι σχεδιασμένο για την ανίχνευση οποιουδήποτε μοτίβου, όπως μια ακμής. Χρησιμοποιώντας ένα φίλτρο που είναι μικρότερο από την αρχική εικόνα, δίνεται η ευκαιρία σε αυτό να ανακαλύψει στο καθένα από τα κομμάτια της εικόνας που εφαρμόζεται κάθε φόρα αν υπάρχει κάποιο μοτίβο που ενδιαφέρετε το δίκτυο ή όχι. Το προϊόν της διαδικασίας αυτής είναι ένας δισδιάστατος πίνακας που ονομάζεται χάρτης χαρακτηριστικών (Feature Map).

2.7.2 Επίπεδο συγκέντρωσης (Pool layer)

Το συνελικτικό επίπεδο στην ουσία δείχνει την ύπαρξη ενός χαρακτηριστικού, για παράδειγμα μιας ακμής, σε μια δεδομένη εικόνα. Το πρόβλημα που δημιουργείτε όμως είναι ότι ο χάρτης χαρακτηριστικών, που δημιουργείτε σαν παράγωγο της συνέλιξης, είναι πολύ ευαίσθητος στις αλλαγές των χαρακτηριστικών στην δεδομένη εικόνα. Ένας τρόπος να προσπελαστεί αυτό το πρόβλημα είναι με την Δειγματοληψία (Down Sampling), η οποία μέθοδος βοηθάει τον παράγωγο χάρτη χαρακτηριστικών να είναι πιο «ανθεκτικός» σε αλλαγές στο σημείο στο οποίο βρίσκονται τα χαρακτηριστικά.

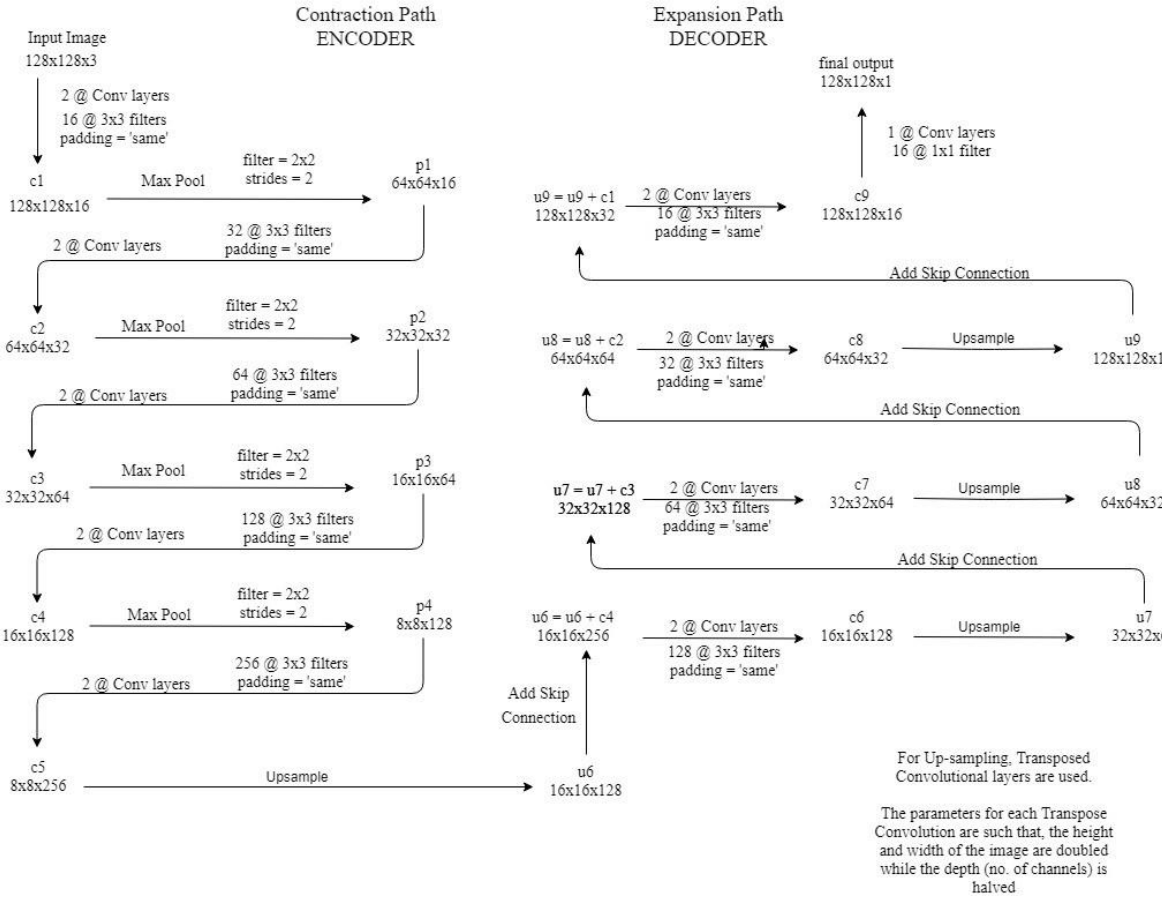
Το μειονέκτημα των χαρτών χαρακτηριστικών είναι ότι καταγράφουν την συγκεκριμένη θέση των χαρακτηριστικών στα δεδομένα εισόδου, αυτό έχει σαν αποτέλεσμα ότι μικρές αλλαγές στην θέση των χαρακτηριστικών έχουν σαν αποτέλεσμα έναν καινούργιο διαφορετικό χάρτη χαρακτηριστικών. Αυτό μπορεί να συμβεί με το κόψιμο, την στροφή, την μετακίνηση και τον κατοπτρισμό της εικόνας εισόδου. Η δειγματοληψία λοιπόν δημιουργεί έναν νέο χάρτη χαρακτηριστικών μικρότερης ανάλυσης που όμως συγκρατεί ακόμη την βασική δομή του, χωρίς όμως την ακρίβεια που είχε ο προηγούμενος, η οποία όμως πολλές φορές δεν είναι χρήσιμη. Αυτή η δειγματοληψία μπορεί να επιτευχθεί και με την αλλαγή της ολίσθησης στο συνελικτικό επίπεδο, όμως η πιο σύγχρονη αντιμετώπιση του προβλήματος είναι η προσθήκη του επίπεδου συγκέντρωσης (pool layer).

Στο επίπεδο συγκέντρωσης κάθε χάρτης χαρακτηριστικών, αντικαθιστάτε από έναν καινούργιο χάρτη μικρότερων διαστάσεων. Κατά την διαδικασία της συγκέντρωσης διαλέγετε ένας τελεστής, κάτι σαν φίλτρο που θα εφαρμοστεί στον χάρτη. Το φίλτρο αυτό είναι μικρότερων διαστάσεων από ότι ο χάρτης, στις περισσότερες περιπτώσεις διαλέγετε να είναι διαστάσεων 2×2 εικονοστοιχείων, αυτό σημαίνει ότι το επίπεδο συγκέντρωσης θα μικραίνει πάντα τον δεδομένο χάρτη. Αν δηλαδή έχουμε έναν χάρτη 6 x6 εικονοστοιχεία και το φίλτρο μας είναι 2x2 εικονοστοιχεία, τότε το αποτέλεσμα θα είναι ένας χάρτης 3 x3 εικονοστοιχεία.

2.8 Πλήρως Συνελικτικά Δίκτυα UNET (Fully Convolutional Neural Networks – UNET)

Τα συνελικτικά δίκτυα είναι πολύ καλά στην αναγνώριση της ύπαρξης ενός αντικειμένου σε μια εικόνα, όμως το πρόβλημα είναι ότι δεν μπορούν να αναγνωρίσουν που ακριβώς βρίσκεται η πληροφορία. Αυτό συμβαίνει γιατί κατά την διάρκεια της δειγματοληψίας το δίκτυο καταλαβαίνει μεν καλύτερα «ΤΙ ΕΙΔΟΥΣ» πληροφορία υπάρχει στην εικόνα, αλλά χάνει την πληροφορία για το «ΠΟΥ» ακριβώς βρίσκεται η πληροφορία. Στην περίπτωση της σημασιολογικής κατάτμησης όμως , όπως ήδη αναφέρθηκε, δεν φτάνει να δοθεί μια ετικέτα ή ένα πλαίσιο οριοθέτησης. Το αποτέλεσμα πρέπει να είναι μια εικόνα υψηλής ανάλυσης που να είναι ταξινομημένα όλα τα εικονοστοιχεία της αρχικής εικόνας. Στην περίπτωση λοιπόν αυτή χρειαζόμαστε και το «ΤΙ» πληροφορία έχουμε, αλλά και το «ΠΟΥ» βρίσκεται η πληροφορία αυτή πάνω στην αρχική εικόνα. Για να γίνει αυτό χρειάζεται μια ακόμη μέθοδος αυτή είναι η μέθοδος UpSampling. Ουσιαστικά το UpSampling Layer αυξάνει τις διαστάσεις μιας δεδομένης εικόνας προσθέτοντας εικονοστοιχεία σε αυτήν.

Το UNET [20] δημιουργήθηκε από τον Olaf Ronneberger για την σημασιολογική κατάτμηση ιατρικών εικόνων το 2015. Η αρχιτεκτονική του δικτύου αυτού αποτελείται από 2 διαδρομές, την διαδρομή συστολής (contraction path – ENCODER) και την συμμετρική διαδρομή διαστολής (symmetric expanding path – DECODER). Η πρώτη διαδρομή χρησιμοποιείτε για να βρει το περιεχόμενο της εικόνας και δεν είναι τίποτα άλλο από επίπεδα συνέλιξης και συγκέντρωσης το ένα μετά το άλλο. Η δεύτερη διαδρομή χρησιμοποιείται για τον ακριβή εντοπισμό του αντικειμένου στην εικόνα χρησιμοποιώντας συνελικτικό μετασχηματισμό. Το UNET λοιπόν είναι ένα πλήρες συνελικτικό δίκτυο, γιατί όλα του τα επίπεδα είναι συνελικτικά το οποίο επιτρέπει να δέχεται εικόνες όλων των μεγεθών. Στο παρακάτω σχήμα (Εικόνα 2.10) φαίνεται ένα δίκτυο UNET στο οποίο στο Encoder Path έχει χρησιμοποιηθεί για την διαδικασία της συγκέντρωσης το Max Pool (βλ. Κεφάλαιο 4) και για την διαδικασία του Up-sample ένα επίπεδο Transposed Convolutional Layer (βλ. Κεφάλαιο 4). Αυτό που συμπεραίνουμε είναι ότι σε κάθε βήμα που γίνεται στην διαδρομή διαστολής χρησιμοποιείται η συνάρτηση Skip Connection με την οποία ενώνουμε το αποτέλεσμα του επιπέδου up-sampling με τον χάρτη χαρακτηριστικών από την διαδρομή συστολής στο ίδιο ακριβώς επίπεδο. Το συμμετρικό σχήμα που έχει λόγο των 2 διαδρομών είναι αυτό που του έδωσε το όνομα UNET.



Εικόνα 2.10: Παράδειγμα αρχιτεκτονικής δικτύου UNET

3 Περιγραφή προβλήματος και δεδομένων

3.1 Εισαγωγή – Περιγραφή του προβλήματος

Στην παρούσα εργασία καλούμαστε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο ώστε να αναγνωρίζει στην επιφάνεια ενός τείχους ποια σημεία είναι διαβρωμένα και ποια όχι. Το τείχος ανήκει στο Φρούριου του Αγίου Νικολάου στην μεσαιωνική πόλη της νήσου Ρόδος. Το Φρούριο του Αγίου Νικολάου είναι ιπποτικό μνημείο στη Ρόδο, σημαντικό οχυρό για την άμυνα της πόλης και των λιμανιών, που κτίστηκε από το Μεγάλο Μάγιστρο Zacosta γύρω στα 1464-67 στο μόλο του Μαντρακιού. Στη θέση του ως το 1460 υπήρχε μόνο η Εκκλησία του Αγίου Νικολάου. Από την πολιορκία του 1480 και το σεισμό του 1481 υπέστη ζημιές και ανοικοδομήθηκε από τον M. D' Aubusson, που την ενίσχυσε με το νέο ισχυρό συγκρότημα που το περιβάλλει. Από τα μέσα του 17ου αιώνα εγκαταστάθηκε στη στέγη του φάρος, λειτουργία που συνεχίζεται μέχρι σήμερα. Από το 1998 άρχισαν οι εργασίες για την αποκατάστασή του και είναι επισκέψιμο σαν εργοτάξιο.

Εικόνα 3.1: Το φρούριο του Αγίου Νικολάου

3.1.1 Παλιές μέθοδοι ανίχνευσης διαβρώσεων σε κατασκευές

Παλαιότερα για την ανίχνευση διαβρώσεων σε πέτρινες κατασκευές ήταν απαραίτητη η αφιέρωση πολλών εργατοωρών. Το συνεργείο που θα αναλάμβανε την εξέταση της κατασκευής θα έπρεπε να καταγράψει όλα τα τμήματα της που θεωρούνταν σύμφωνα με αυτό διαβρωμένα. Μετά την καταγραφή των σημείων της κατασκευής που θεωρούνταν διαβρωμένα, το συνεργείο καλούταν να κάνει μετρήσεις και ακριβή αποτύπωση των σημείων-επιφανειών που είχαν καταγραφή. Οι μετρήσεις αυτές συνήθως ήταν ένας συνδυασμός γεωδαιτικών και φωτογραμμετρικών – τηλεσκοπικών μεθόδων. Για παράδειγμα για την

παρακολούθηση μια ρωγμής στην πέτρινη κατασκευή απαιτούταν η μέτρηση των μικρομετακινήσεων με μεθόδους μετρήσεων ακριβείας των σημείων περιμετρικά της ρωγμής. Για την μελέτη καθιζήσεων και άλλων μετακινήσεων της κατασκευής συνολικά πάλι ήταν απαραίτητες συνεχής μετρήσεις ακριβείας. Τέλος σε ότι αφορά την μελέτη των διαβρώσεων επιφανειών και της εξαγωγής πληροφοριών από αυτές, όπου η περίπτωση αυτή αφορά και την παρούσα εργασία, ήταν απαραίτητη η μέτρηση των επιφανειών και η εξαγωγή συμπερασμάτων από αυτές. Αυτό σημαίνει επίσης και συνεχής παρακολούθηση των περαιτέρω διαβρώσεων στις επιφάνειες αυτές.

3.1.2 Κίνδυνοι παλαιών μεθόδων – κόστος

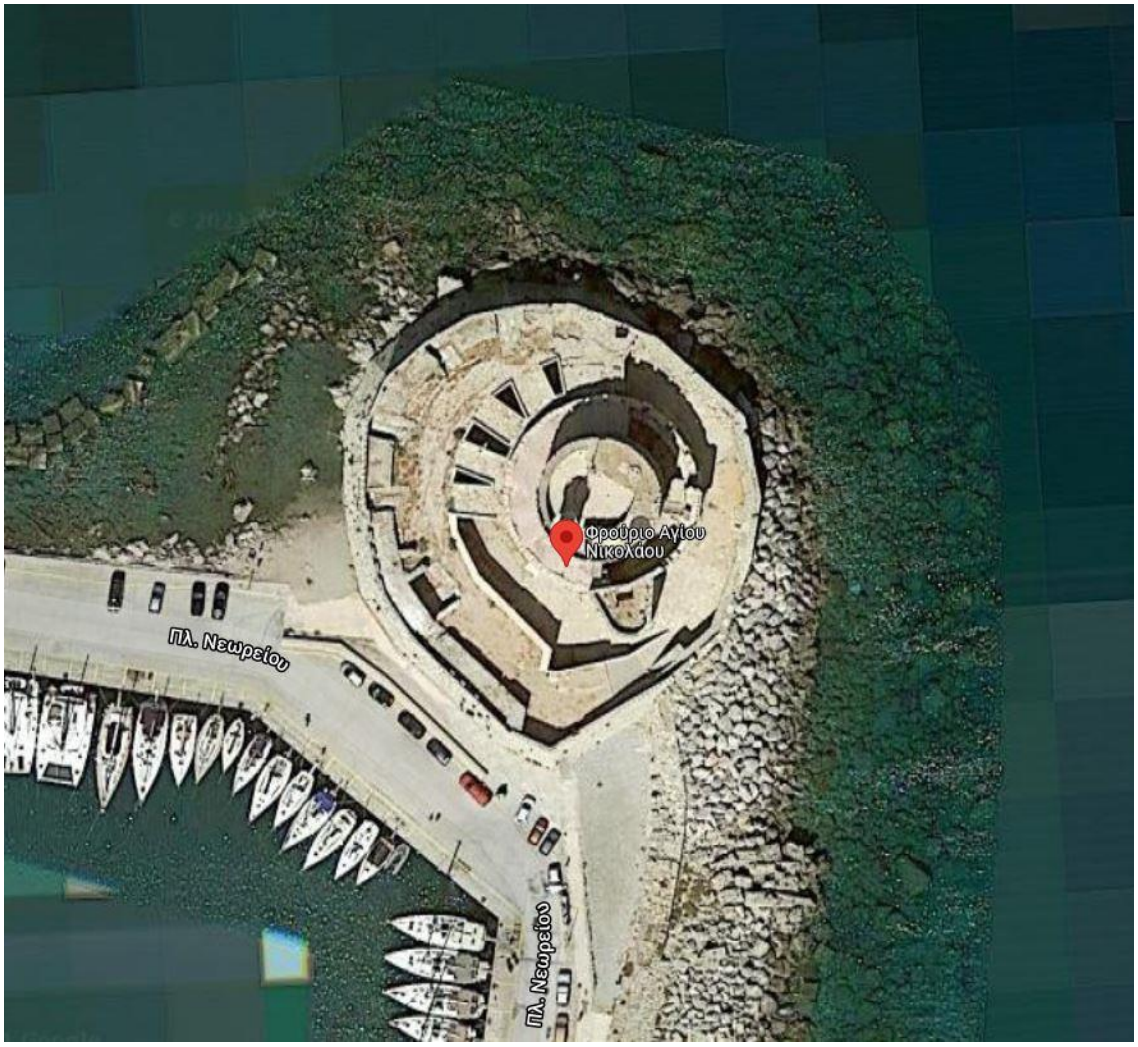
Οι παλιές μέθοδοι παρακολούθησης διαβρώσεων σε κατασκευές απαιτούσαν την συνεχή παρουσία μελών του συνεργείου στον χώρο ώστε να μελετάνε με ακρίβεια την οποιαδήποτε αλλαγή συμβαίνει σε αυτή. Η παρακολούθηση όμως δεν γινόταν πάντα υπό τις ίδιες συνθήκες, που σημαίνει ότι ούτε τα άτομα που τις κάνανε θα μπορούσαν να είναι συνέχεια τα ίδια, αλλά και ούτε τα όργανα. Αυτό οδηγεί στην προβληματική εξαγωγή συμπερασμάτων που αφορούν τα συστηματικά σφάλματα που επηρέαζαν τις μετρήσεις. Δηλαδή δεν υπήρχαν αντικειμενικά κριτήρια με βάση τα οποία κάποιος μπορούσε να κατανοήσει τι αποτελεί συστηματικό σφάλμα και τι όχι, γιατί δεν χρησιμοποιούνταν πάντα τα ίδια συστήματα ανίχνευσης. Για παράδειγμα, η αντίληψη του ανθρώπου για το τι είναι και τι όχι διάβρωση , καθώς και η ικανότητα του να τις αναγνωρίζει πάνω στην επιφάνεια του τοίχου δεν θα μπορούσαν να είναι αντικειμενικές, καθώς αυτές επηρεάζονται από πολύπλευρους παράγοντες, όπως το πόσο ξεκούραστος ή όχι είναι ο μελετητής την ώρα της παρατήρησης. Ακόμα στην περίπτωση των άμεσων μετρήσεων απευθείας πάνω στην επιφάνεια των τοίχων, πάλι δεν ήταν αντικειμενικές καθώς επηρεάζονται από το σφάλμα στόχευσης λόγο του ανθρώπινου ματιού καθώς και της κατάστασης του ίδιου του οργάνου. Βέβαια ακόμα και αν η εξαγωγή συμπερασμάτων συνέβαινε με τη μελέτη εικόνων των επιφανειών, οι ίδιες αβεβαιότητες για την ποιότητα των μετρήσεων πάλι θα ίσχυαν.

3.1.3 Λύσεις με την χρήση νευρωνικών δικτύων

Η επίλυση αυτού του προβλήματος με την εκπαίδευση νευρωνικών δικτύων και την πραγματοποίηση προβλέψεων με αυτά, αποτελεί πιο αξιόπιστη μέθοδο καθώς πολλές παράμετροι που οδηγούσαν σε αύξηση του σφάλματος έχουν εξαλειφθεί. Τέλος η χρήση νευρωνικών δικτύων για την επίλυση αυτού του προβλήματος υπόσχεται μείωση του κόστους υλοποίησης της εργασίας, καθώς μειώνει τον αριθμό των ατόμων που χρειάζονται για την υλοποίηση της , καθώς και τον αριθμό των εργαλείων που είναι απαραίτητα για την διεξαγωγή της. Με την χρήση μιας φωτογραφικής κάμερας και ενός ηλεκτρονικού υπολογιστή, η εργασία μπορεί να διεξαχθεί με ακόμα καλύτερα αποτελέσματα και σε λιγότερο χρονικό διάστημα.

3.2 Περιγραφή δεδομένων

Το φρούριο του Αγίου Νικολάου στην ρόδο, αποτελεί την περιοχή από την οποία συλλέχθηκαν δεδομένα για την εκπαίδευση του νευρωνικού δικτύου που θα αναλυθεί στο επόμενο κεφάλαιο. Συνολικά έγινε λήψη 11 φωτογραφιών από το εξωτερικό τείχος του φρουρίου (Εικόνα 3.2).



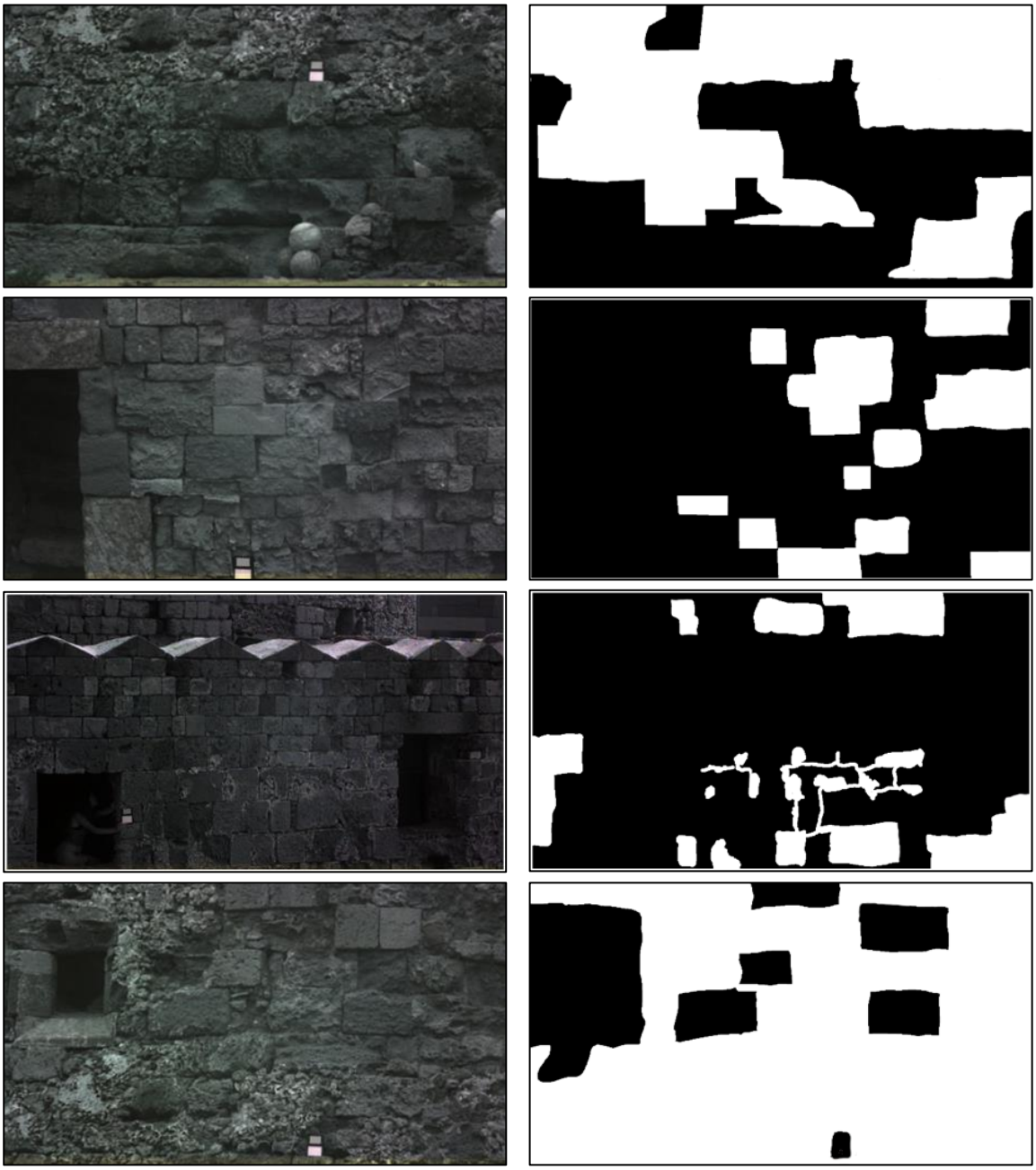
Εικόνα 3.2: Δορυφορική εικόνα φρουρίου Αγίου Νικολάου

Οι εικόνες αυτές είναι διαστάσεων 1859x1044 εικονοστοιχείων και η κάθε μια έχει 3 κανάλια, καθώς επίσης έχουν προκύψει ως ένωση των NIR (Near-Infrared Spectroscopy) και των VIS (Visible Image RGB) εικόνων. Οι εικόνες NIR είναι εικόνες που μετατρέπουν την υπέρυθρη ακτινοβολία, που ανακλάται από την επιφάνεια του εικονιζόμενου αντικειμένου και δεν είναι ορατή στον άνθρωπο, από 0.7μm σε 1.4μm που είναι ορατή στον άνθρωπο. Η ένωση των 2 εικόνων έγινε με την χρήση του συστήματος HyperView που χρησιμοποιεί 2 κάμερες μια για την λήψη της NIR εικόνας και η άλλης για την λήψη της VIS εικόνας. Οι δύο αυτές κάμερες του συστήματος HyperView μπορούν να λειτουργήσουν είτε και οι 2 μαζί είτε και ξεχωριστά η κάθε μια:

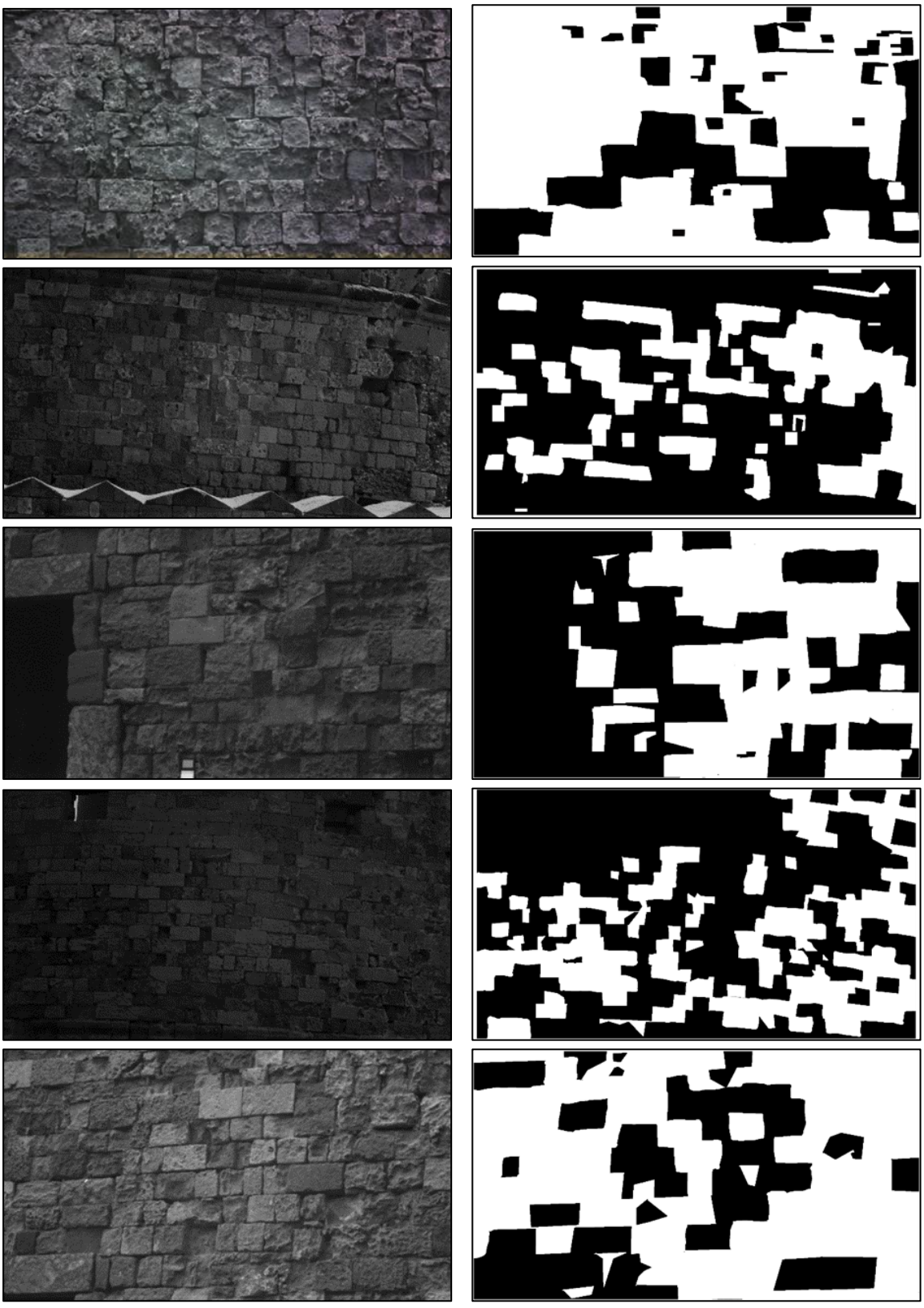
- 1 λήψη, κάμερα VIS – 470-620nm, 16 κανάλια
- 1 λήψη, κάμερα NIR – 600-975nm, 25 κανάλια
- 1 λήψη, κάμερες VIS και NIR μαζί – 470-975nm, 41 κανάλια

Επίσης κάθε μια από αυτές τις εικόνες συνοδεύεται από μια αντίστοιχη μάσκα της εικόνας ίδιων διαστάσεων 1859x1044 εικονοστοιχείων, αλλά 2 καναλιών. Όπου στο κάθε εικονοστοιχείο κάθε μιας από τις εικόνες – μάσκες αυτές, έχει δοθεί η τιμή 0 (μαύρο) αν το

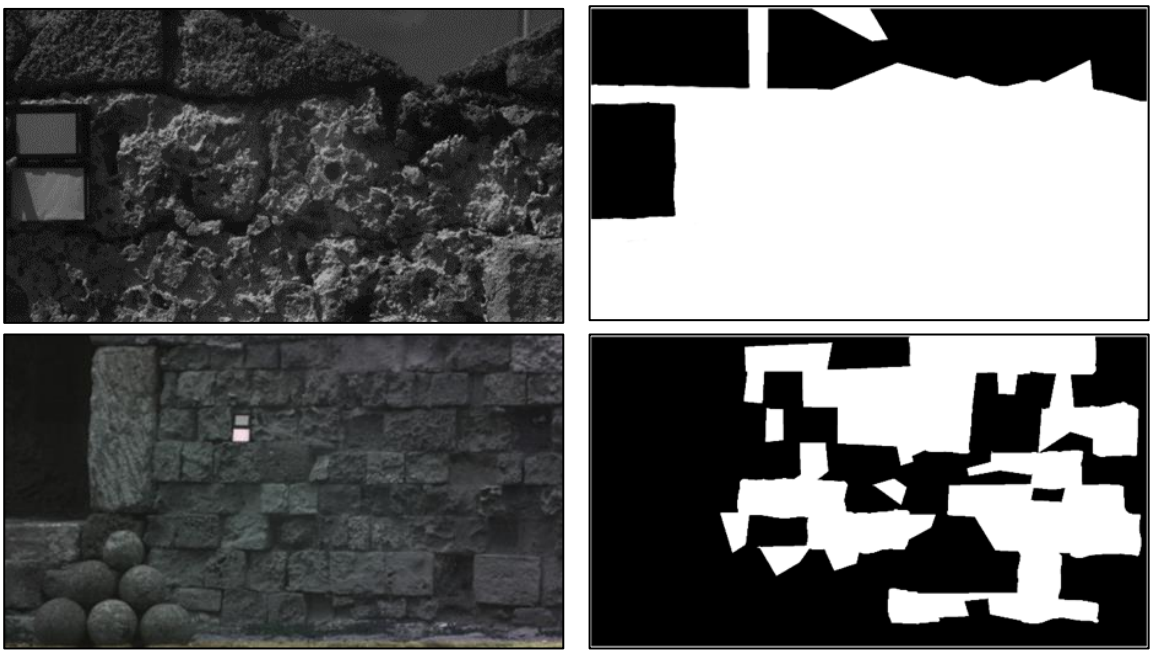
εικονοστοιχείο απεικονίζει κομμάτι του τείχους που δεν έχει διαβρωθεί και η τιμή 255 (λευκό) αν απεικονίζει κομμάτι που έχει διαβρωθεί. Παρακάτω φαίνονται οι εικόνες σε σύγκριση με τις αντίστοιχες μάσκες τους (Εικόνα 3.3).



Εικόνα 3.3: Δεδομένες εικόνες και αντίστοιχες μάσκες εικόνων (συνεχίζεται στην επόμενη σελίδα)



Εικόνα 3.3: Δεδομένες εικόνες και αντίστοιχες μάσκες εικόνων (συνεχίζεται στην επόμενη σελίδα)



Εικόνα 3.3: Δεδομένες εικόνες και αντίστοιχες μάσκες εικόνων

3.3 Προεπεξεργασία δεδομένων

Πριν οι εικόνες εισαχθούν στο δίκτυο για να γίνει η εκπαίδευση του πρέπει πρώτα να επεξεργαστούν έτσι ώστε να βελτιωθεί το αποτέλεσμα του δικτύου, αλλά και για να αλλάξουν στην κατάλληλη μορφή για να μπορούν να εκπαιδεύσουν το δίκτυο.

3.3.1 Ταξινόμηση δεδομένων

Ένα δίκτυο εκπαιδεύεται (training process) και αφού εκπαιδευτεί κάνει προβλέψεις για να ελεγχθούν ο αποδόσεις του (testing process). Το πρώτο πράγμα που πρέπει να σκεφτεί κάποιος, όταν ξεκινάει την διαδικασία της εκπαίδευσης ενός δικτύου, είναι πόσα από τα συνολικά δεδομένα πρέπει να δώσει στο δίκτυο για εκπαίδευση (training) και πόσα στο πρέπει να δώσει για δοκιμές (testing). Τα δεδομένα που δίνονται για εκπαίδευση και αυτά που δίνονται για δοκιμή πρέπει να είναι διαφορετικά για να εξασφαλιστεί η αντικειμενικότητα του δικτύου, δηλαδή να μην έχει «ξαναδεί» το δίκτυο τις εικόνες πάνω στις οποίες θα δοκιμαστεί. Γι' αυτόν τον λόγο δίνουμε ένα ποσοστό των δεδομένων στο σετ εκπαίδευσης (training set) και ένα ποσοστό των δεδομένων στο σετ δοκιμής (testing set). Για το παρών δίκτυο λόγο των λίγων δεδομένων που ήταν διαθέσιμα επιλέχθηκε να δοθούν λίγα δεδομένα στο σετ δοκιμής προκειμένου να είναι όσο το δυνατόν καλύτερα τα αποτελέσματα της εκπαίδευσης. Το ποσοστό αυτό είναι 90% TRAINING και 10% στο TESTING, δηλαδή 10 και 1 εικόνες αντιστοίχως.

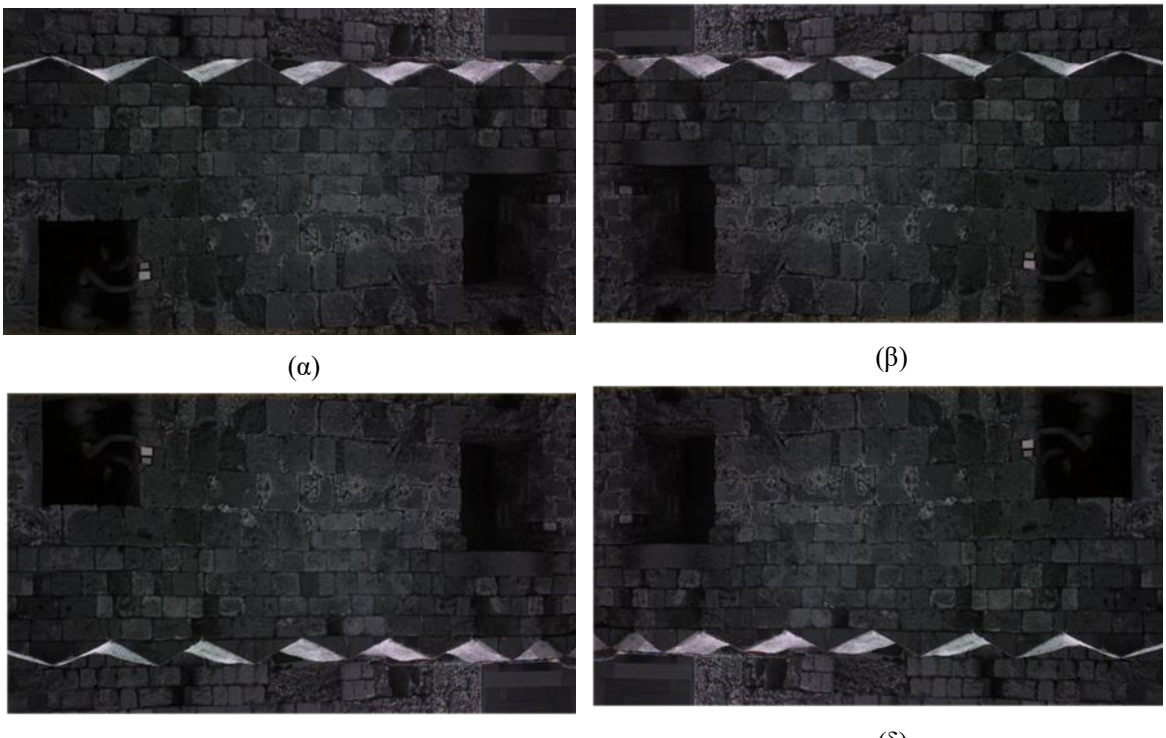
Κατά την διαδικασία της εκπαίδευσης συμβαίνει και μια άλλη διαδικασία που ονομάζεται Επικύρωση (Validation). Αυτό σημαίνει ότι το δίκτυο ενώ προσπαθεί να προσαρμόσει τις μεταβλητές του για να μάθει τα μοτίβα όσο το δυνατόν καλύτερα, χρησιμοποιεί ένα ποσοστό των δεδομένων εκπαίδευσης για να κάνει προβλέψεις ταυτόχρονα με την εκπαίδευση ώστε να μάθει τι ποσοστά επιτυχίας έχει και να καθοδηγήσει ύστερα την διαδικασία την ανανέωσης των μεταβλητών του δικτύου καλύτερα. Για να συμβεί αυτό σημαίνει ότι εικόνες που

θα χρησιμοποιηθούν για την επικύρωση του δικτύου δεν πρέπει να είναι οι ίδιες που χρησιμοποιήθηκαν για την ανανέωση των μεταβλητών, προκειμένου πάλι να είναι αντικειμενική η επικύρωση. Άρα τα δεδομένα από το σετ εκπαίδευση πρέπει να υποστούν άλλον έναν διαχωρισμό, ένα ποσοστό πρέπει να παραμείνει στο σετ εκπαίδευσης (training set), αλλά ένα άλλο πρέπει να δοθεί για την διαδικασία την επικύρωσης, αυτό είναι το σετ επικύρωσης (validation set). Στο παρών δίκτυο το ποσοστό αυτό είναι 80% στο TRAINING και 20% στο VALIDATION, δηλαδή 8 και 2 εικόνες αντιστοίχως.

Επί του συνόλου τα δεδομένα έχουν χωριστεί συνολικά σε 3 σετ, TRAIN SET, VALIDATION SET και TEST SET, όπου το καθένα έχει ποσοστά 72% (8 εικόνες) , 18% (2 εικόνες) και 10% (1 εικόνα) αντιστοίχως επί του συνόλου. Η αντίστοιχη διαδικασία έγινε και για τις αντίστοιχες μάσκες χαρακτηριστικών των εικόνων με τα ίδια ακριβώς ποσοστά καθώς είναι ίδιες σε αριθμό. Η βάση δεδομένων των πραγματικών εικόνων X ονομάστηκε INPUT_SET και η βάση δεδομένων Y των μασκών ονομάστηκε OUTPUT_SET.

3.3.2 Αύξηση δεδομένων (data augmentation)

Η αύξηση δεδομένων είναι μια τεχνική που χρησιμοποιείται στην εκπαίδευση των νευρωνικών δικτύων για αυξηθούν οι επιδώσεις τους. Στην αύξηση δεδομένων γίνεται επεξεργασία των δεδομένων φωτογραφιών και αλλάζονται σε μικρό βαθμό έτσι ώστε η πληροφορία που μεταδίδεται να είναι η ίδια για εμάς, αλλά για τον υπολογιστή να είναι ένα τελείως καινούργιο μοτίβο που πρέπει να μάθει. Οι αλλαγές αυτές μπορεί να είναι ο κατοπτρισμός, η περιστροφή, η περικοπή, η αλλαγή κλίμακας και η παραμόρφωση. Για την αύξηση των δεδομένων εκπαίδευσης του παρών δικτύου χρησιμοποιήθηκε η τεχνική του κατοπτρισμού σε 1 και 2 διευθύνσεις όπως φαίνεται στην παρακάτω εικόνα (Εικόνα 3.4).



Εικόνα 3.4: (α) Αρχική εικόνα (β) Κατοπτρισμός εικόνας στον κατακόρυφο άξονα (γ) Κατοπτρισμός εικόνας στον οριζόντιο άξονα (δ) Κατοπτρισμός εικόνας και στους δύο άξονες

Η διαδικασία αυτή έγινε για όλες τις πραγματικές εικόνες αλλά και για τις αντίστοιχες μάσκες των εικόνων των βάσεων δεδομένων TRAIN SET και VALIDATION SET. Άρα η 1 φωτογραφία με την τεχνική αυτή γίνεται 4 φωτογραφίες, άρα στο σύνολο οι 8 εικόνες του TRAIN SET θα γίνουν 32 εικόνες και 2 του VALIDATION SET θα γίνουν 8. Το TEST SET που είναι μια εικόνα δεν έχει νόημα να του εφαρμόσουμε αυτή την τεχνική καθώς η εικόνα αυτή θα χρησιμοποιηθεί για δοκιμή και δεν θα προσφέρει κάτι παραπάνω ο κατοπτρισμός της.

3.3.3 Αύξηση δεδομένων με περικοπή των αρχικών εικόνων

Στην συνέχεια για να αυξήσουμε κι άλλο τα δεδομένα θα χωρίσουμε την κάθε μια από τις εικόνες μεγέθους 1859x1044 σε εικόνες διαστάσεων 256x256. Προκειμένου το δίκτυο να συλλέξει όσο το δυνατόν περισσότερη πληροφορία από τις δεδομένες εικόνες, αυτές χωρίστηκαν μεν σε εικόνες διαστάσεων 256x256, αλλά ορίστηκε ποσοστό οριζόντιας και κατακόρυφης επικάλυψης μεταξύ των κομματιών της κάθε μιας, ίσο με 30%. Αν προσπαθήσουμε τώρα να χωρέσουμε τα κομμάτια διαστάσεων 256x256 στο πλαίσιο 1859x1044 της αρχικής εικόνας θα βγάλουμε το συμπέρασμα ότι δεν υπάρχει ακριβής αριθμός κομματιών στα οποία μπορεί να χωριστεί η αρχική μας εικόνα. Ας εξετάσουμε πρώτα την πρώτη γραμμή κομματιών της αρχικής εικόνας 1859x1044. Επειδή η μία εικόνα με την επόμενη της έχουν επικάλυψη 30%, με την κάθε μια θα «περισσεύει» μόνο το 70% της κάθε μιας, δηλαδή 0,7*256=179 εικονοστοιχεία εκτός της τελευταίας που είναι ολόκληρη. Τα 179 εικονοστοιχεία αυτά χωράνε στην διάσταση 1859-256=1603 εικονοστοιχεία, 1603/179=9 φορές. Άρα 1859-(9*179+256) θα έπρεπε να ισούται με 0, αλλά στην πραγματικότητα δίνει -8 εικονοστοιχεία που σημαίνει ότι το τελευταίο κομμάτι θα πρέπει να έχει κατά 8 εικονοστοιχεία παραπάνω επικάλυψη, δηλαδή 8/256=0.03 που συνεπάγεται 33% επικάλυψη (Εικόνα 3.5).

Ας εξετάσουμε τώρα την κατακόρυφη διάσταση της αρχικής εικόνας με διαστάσεις 1044 εικονοστοιχείων, για να ανακαλύψουμε πόσα κομμάτια χωράνε σε αυτήν. Με την ίδια λογική όπως το παράδειγμα για την οριζόντια επικάλυψη, πάλι η κατακόρυφη επικάλυψη είναι 30%, οι πράξεις φαίνονται παρακάτω.

$$256 * 0{,}7 = 179\ pixels$$

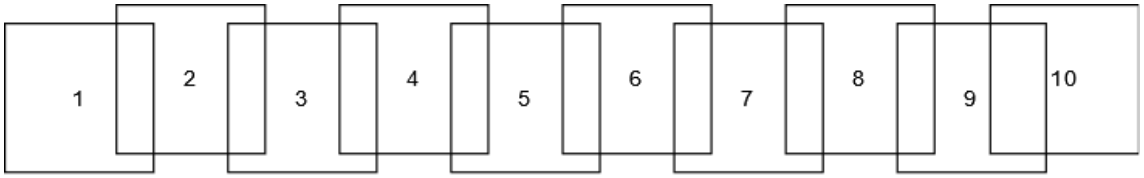
$$1044 - 256 = 788\ pixels$$

$$\frac{788}{179} = 4.4\ που\ γίνεται\ 5\ φορές$$

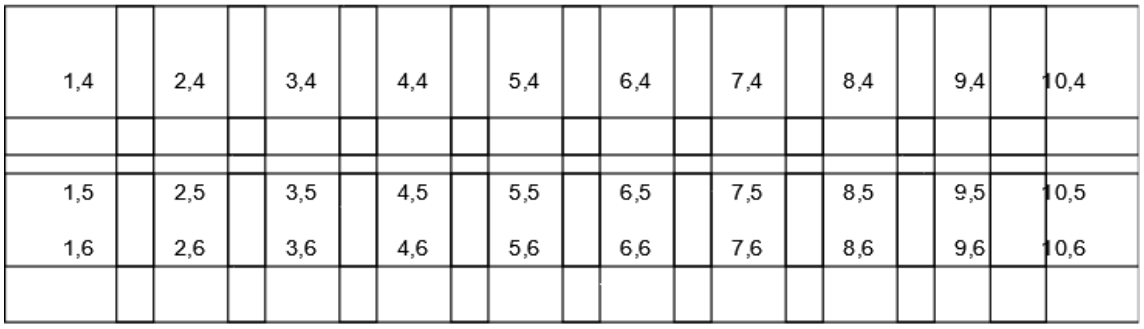
$$1044 - (5 * 179 + 256) = -107\ pixels$$

$$-\frac{107}{256} = -0.4 \Rightarrow 70\%\ επικάλυψη\ μεταξύ\ της\ προτελευταίας\ γραμμής\ και\ της\ τελευταίας.$$

Άρα η τελευταία γραμμή έχει παραπάνω επικάλυψη επίσης (Εικόνα 3.5), αν παρατηρήσουμε το τελευταίο κομμάτι της τελευταίας γραμμής και τελευταίας στήλης, δηλαδή την κάτω δεξιά γωνία, τότε θα συμπεράνουμε ότι το κομμάτι αυτό έχει παραπάνω επικάλυψη από ότι τα άλλα και στην οριζόντια και στην κατακόρυφη διεύθυνση.



Οριζόντια Επικάλυψη σε μια γραμμή



Κατακόρυφη Επικάλυψη μεταξύ των 3 τελευταίων γραμμών

Εικόνα 3.5: Κατακόρυφη και οριζόντια επικάλυψη ανά εικόνα

Στο σύνολο η κάθε μια εικόνα θα χωρίζεται σε 10 στήλες και 6 γραμμές, συνολικά δηλαδή από κάθε μια φωτογραφία θα παράγονται 10*6=60 εικόνες διαστάσεων 256x256 και 3 καναλιών. Η ίδια διαδικασία θα γίνει για όλες τις εικόνες και τις αντίστοιχες μάσκες τους στις βάσεις δεδομένων TRAIN SET και VALIDATION SET. Στο σύνολο λοιπόν, η βάση δεδομένων TRAIN SET θα καταλήξει με 32*60=1920 εικόνες οι πραγματικές και άλλες 1920 εικόνες μάσκες. Η βάση δεδομένων VALIDATION SET θα καταλήξει με 8*60=480 πραγματικές εικόνες και άλλες 480 εικόνες μάσκες. Στην συνέχεια οι εικόνες αυτές προκειμένου να δοθούν στο δίκτυο για εκπαίδευση θα πρέπει να μετατραπούν σε διανύσματα τύπου np.array με την βοήθεια της βιβλιοθήκης NumPy. Το διάνυσμα των πραγματικών εικόνων της βάσης εκπαίδευσης ονομάζεται X_train και έχει διαστάσεις X_train.(1920,256,256,3) και το διάνυσμα των εικόνων μασκών ονομάζεται Y_train και έχει διαστάσεις Y_train.(1920,256,256,2). Αντίστοιχα, το διάνυσμα των πραγματικών εικόνων της βάσης επικύρωσης θα ονομάζεται X_val και θα έχει διαστάσεις X_val.(480,256,256,3) και το διάνυσμα των εικόνων μασκών της βάσης επικύρωσης θα ονομάζεται Y_val και θα έχει διαστάσεις Y_val.(480,256,256,2).

4 Προτεινόμενο μοντέλο – ResAttUNet

Παρακάτω παρουσιάζονται τα χαρακτηριστικά του δικτύου UNET που χρησιμοποιήθηκε στην παρών εργασία για την ανίχνευση διαβρώσεων σε κομμάτι τείχους του φρουρίου του Αγίου Νικολάου στην Πόλη της Ρόδου. Στην παρών εργασία το μοντέλο που χρησιμοποιήθηκε είναι γνωστό ως ResAttUNet, που φτιάχτηκε το 2019 [34], και αποτελεί έναν συνδυασμό των μοντέλων UNET και ResNET με την προσθήκη μπλοκ Προσοχής (Attention).

4.1 Συναρτήσεις Ενεργοποίησης (Activation Function)

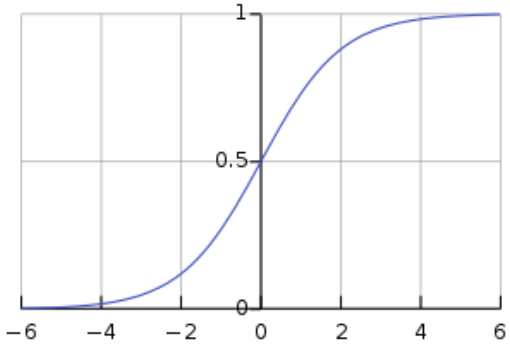
Στην συγκεκριμένη εργασία χρησιμοποιήθηκαν δυο είδη συναρτήσεων ενεργοποίησης, η Συνάρτηση Διορθωμένης Γραμμικής Μονάδας (Rectified Linear Unit – ReLU) και η Σιγμοειδής Συνάρτηση.

4.1.1 Σιγμοειδής Συνάρτηση

Η σιγμοειδής συνάρτηση είναι μια μαθηματική συνάρτηση που έχει το σχήμα του αγγλικού γράμματος S. Της περισσότερες φορές ως σιγμοειδής συνάρτηση αναφέρεται η ειδική περίπτωση της λογιστικής παλινδρόμησης που δίνεται από τον παρακάτω τύπο.

$$S(x) = \frac{1}{1 + e^{-x}}, \text{με πεδίο τιμών } [0,1] \qquad \text{Εξ. (4.1)}$$

Όπως φαίνεται και στο παρακάτω σχήμα (Εικόνα 4.1), η σιγμοειδής συνάρτηση ουσιαστικά δέχεται κάποιο δεδομένα εισόδου και το μετατρέπει σε μια τιμή από το 0 έως 1. Οι τιμές οι οποίες είναι πολύ μεγαλύτερες του 0 μετατρέπονται στην τιμή 1 και οι τιμές οι οποίες είναι πολύ μικρότερες του 0 παίρνουν την τιμή 0.

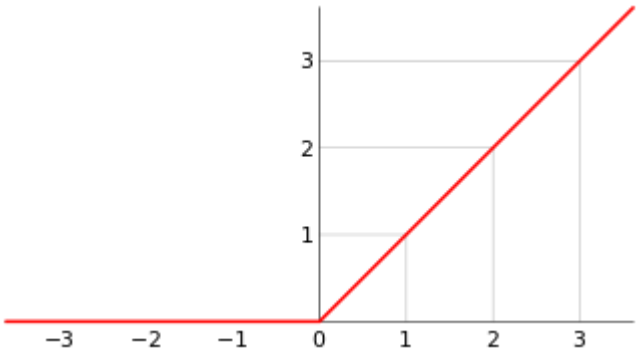


Εικόνα 4.1: Διάγραμμα σιγμοειδούς συνάρτησης

4.1.2 Rectified Linear Unit (ReLU)

Προκειμένου να ξεπεραστεί το πρόβλημα της σιγμοειδούς συνάρτησης, δηλαδή ότι είτε μηδενίζει είτε δίνει την τιμή 1 στις πολύ μικρές ή πολύ μεγάλες τιμές αντίστοιχα, δημιουργήθηκε η ανάγκη για μια συνάρτηση η οποία είναι γραμμική, δηλαδή αφήνει την πληροφορία να περάσει αναλλοίωτη, αλλά ταυτόχρονα δεν είναι γραμμική αφήνοντας το δίκτυο να μάθει πολύπλοκα μοτίβα. Η συνάρτηση αυτή είναι η συνάρτηση ReLU και δίνεται από την εξίσωση (Εξ. (4.2)), ενώ περιγράφεται από το παρακάτω σχήμα (Εικόνα 4.2).

$$f(x) = \begin{cases} x\,, για\ x > 0 \\ 0 \end{cases}$$ Εξ. (4.2)



Εικόνα 4.2: Διάγραμμα συνάρτησης ReLU

Αυτές οι ιδιότητες της συνάρτησης ReLU δίνουν το πλεονέκτημα σε σχέση με την χρήση της σιγμοειδούς γιατί επιτρέπει στους νευρώνες με πολύ χαμηλές τιμές να μην ενεργοποιούνται καθιστώντας το δίκτυο μας πιο «ελαφρύ».

4.2 Συνάρτηση Απώλειας – Dice Coefficient Loss

Η συνάρτηση απώλειας που χρησιμοποιήθηκε στο παρόν μοντέλο είναι η συνάρτηση Dice Coefficient Loss [21] γνωστή και ως Dice Loss και δίνεται από τον παρακάτω τύπο:

$$DL(y,p) = 1 - \frac{2 * y * p + 1}{y + p + 1}$$ Εξ. (4.3)

Όπου y η πραγματική τιμή και p η τιμή που προκύπτει από την συνάρτηση ενεργοποίησης.

4.3 Βελτιστοποιητής Adam (Adam Optimizer)

Ο Βελτιστοποιητής Adam (Adam Optimizer) [22][23][24] δημοσιεύτηκε πρώτη φορά το 2014 [ΑΝΑΦΟΡΑ] ως ένας αλγόριθμος βελτιστοποίησης που προσαρμόζει τον ρυθμό μάθησης (learning rate), ειδικά σχεδιασμένος για εκπαίδευση βαθιών νευρωνικών δικτύων (deep neural network). Ο βελτιστοποιητής αυτός χρησιμοποιεί τα τετράγωνα της κλίσης (gradient) για να μετρήσει το ρυθμό μάθησης και ταυτόχρονα εκμεταλλεύεται τις ροπές αλλάζοντας τον μέσω όρο των κλίσεων αντί για την ίδια την κλίση.

Ο Adam είναι όπως αναφέραμε μια μέθοδος προσαρμογής των ρυθμών μάθησης, ο οποίος όμως υπολογίζει ξεχωριστά το ρυθμό μάθησης για κάθε μεταβλητή. Το όνομα του βελτιστοποιητή Adam πηγάζει από την ονομασία Προσαρμοστική Εκτίμηση Ροπών (ADAptive Moment estimation). Ο λόγος για τον οποίο ονομάζεται έτσι είναι επειδή ο Adam χρησιμοποιεί εκτιμήσεις της πρώτης και της δεύτερης ροπής της κλίσης (Μέση Τιμή και Διακύμανση) για να προσαρμόσει το ρυθμό μάθησης του κάθε βάρους του δικτύου. Ως n-ιοστή ροπή μια τυχαίας μεταβλητής x ορίζετε το αποτέλεσμα που δίνει η εκτιμήτρια συνάρτηση για τον n-ιοστό βαθμό αυτής της μεταβλητής (Εξ. (4.4)).

$$m_n = E[x^n] \quad \text{Εξ. (4.4)}$$

Η κλίση μια συνάρτησης κόστους μπορεί να θεωρηθεί τυχαία καθώς βασίζεται σε τυχαίο δείγμα των δεδομένων. Η πρώτη ροπή λοιπόν είναι η μέση τιμή του κόστους και η δεύτερη ροπή είναι διακύμανση του κόστους. Για να εκτιμήσει τις ροπές, ο Adam, χρησιμοποιεί εκθετικά εναλλασσόμενους μέσους όρους της ροπής, που υπολογίζονται από την κλίση του τρέχοντος δείγματος (Εξ. (4.5), Εξ. (4.6)).

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \quad \text{Εξ. (4.5)}$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \quad \text{Εξ. (4.6)}$$

Όπου m και v είναι οι μέσες τιμές των ροπών, g η κλίση στο τρέχων δείγμα και $\beta_{1,2}$ είναι καινούργιες παράμετροι που συνήθως είναι 0.9 και 0.999 αντιστοίχως. Τα διανύσματα των μεταβαλλόμενων μέσων τιμών αρχικοποιούνται στο 0 στην πρώτη επανάληψη. Για να δούμε πως ανταποκρίνονται αυτές οι τιμές στις ροπές, θα πρέπει να ισχύει η παρακάτω σχέση, εφόσον τα m και v είναι εκτιμήσεις της πρώτης και της δεύτερης ροπής.

$$E[m_t] = E[g_t] \quad \text{Εξ. (4.7)}$$

$$E[v_t] = E[g_t^2] \quad \text{Εξ. (4.8)}$$

Οι προβλεπόμενες τιμές των εκτιμήσεων θα έπρεπε να είναι ίσες με τις παραμέτρους που προσπαθούμε να εκτιμήσουμε, αλλά η παράμετρος στην περίπτωση αυτή είναι επίσης εκτιμώμενη τιμή. Αν η παραπάνω σχέσεις (Εξ. (4.7), Εξ. (4.8)) ισχύουν τότε λέμε ότι έχουμε μια αμερόληπτη εκτίμηση. Όμως, όπως θα δούμε η παραπάνω σχέσεις δεν επαληθεύονται για την δική μας περίπτωση, επειδή οι μέσοι όροι αρχικοποιήθηκαν στο 0. Για να το κάνουμε αυτό θα πρέπει να βρούμε πρώτα κάποιο τύπο για το m.

$$m_0 = 0$$

$$m_1 = \beta_1 * m_0 + (1-\beta_1) * g_1 = (1-\beta_1) * g_1$$

$$m_2 = \beta_1 * m_1 + (1-\beta_1) * g_2 = \beta_1 * (1-\beta_1) * g_1 + (1-\beta_1) * g_2$$

$$m_3 = \beta_1 * m_2 + (1-\beta_1) * g_3 = \beta_1^2 * (1-\beta_1) * g_1 + \beta_1 * (1-\beta_1) * g_2 + (1-\beta_1) * g_3$$

Οπότε αν συνεχίσουμε με την ίδια λογική καταλήγουμε στον παρακάτω τύπο.

$$m_t = (1 - \beta_1) * \sum_{i=0}^{t} \beta_1^{t-i} * g_i \quad \text{Εξ. (4.9)}$$

Αν ξαναγράψουμε την αναμενόμενη τιμή της τιμής m προκύπτει ο παρακάτω τύπος.

$$E[m_t] = E\left[(1-\beta_1) * \sum_{i=1}^{t} \beta_1^{t-i} * g_i\right] = E[g_i] * (1-\beta_1) * \sum_{i=1}^{t} \beta_1^{t-i} + \zeta$$
$$\Longrightarrow E[m_t] = E[g_i] * (1-\beta_1^t) + \zeta \qquad \text{Εξ. (4.10)}$$

Όπου το g_i αντικαταστάθηκε με g_t και ζ είναι το σφάλμα λόγω αυτής της αντικατάστασης, αντίστοιχα θα προκύψουν και για v_t. Τώρα πρέπει να διορθώσουμε την εκτιμήτρια συνάρτηση σύμφωνα με τις αναμενόμενες τιμές που θέλουμε. Οι τύποι που προκύπτουν είναι οι παρακάτω.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \text{Εξ. (4.11)}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \qquad \text{Εξ. (4.12)}$$

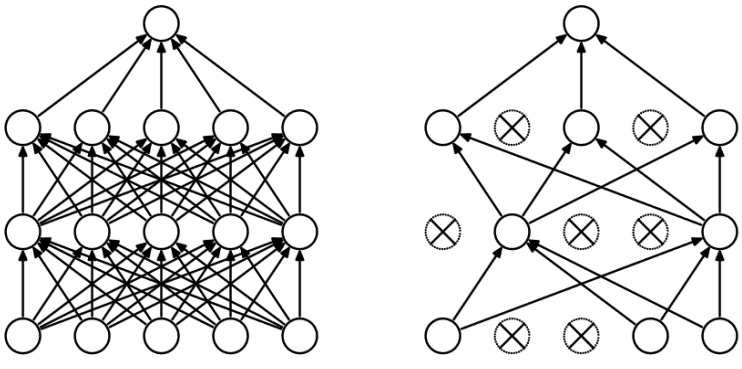
Τέλος, πρέπει να βρεθεί ο τύπος που δίνει το ρυθμό μάθησης κάθε παραμέτρου χρησιμοποιώντας τους μεταβαλλόμενους μέσους όρους. Ο τρόπος λοιπόν με τον οποίο ο βελτιστοποιητής Adam διορθώνει τα βάρη του δικτύου είναι:

$$w_t = w_{t-1} - \eta * \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + e} \qquad \text{Εξ. (4.13)}$$

Όπου w τα βάρη και η το βήμα.

4.4 Τακτοποιητής – απόσυρση (Regularizer – dropout)

Ο Τακτοποιητής Απόσυρση (Dropout) [27][28] στην ουσία διαλέγει κάποιους νευρώνες κατά την διάρκεια της εκπαίδευσης (τυχαία) και δεν τους λαμβάνει υπόψη του κατά την διάρκεια της μεταφοράς δεδομένων από το ένα επίπεδο στο άλλο. Δηλαδή κατά την διάρκεια κάθε στάδιού της εκπαίδευσης κάποιοι νευρώνες του δικτύου αποσύρονται, με πιθανότητα απόσυρσης 1-p, και κάποιοι κρατιούνται, με πιθανότητα p. Παρακάτω (Εικόνα 4.3) φαίνεται ένα κανονικό νευρωνικό δίκτυο σε σχέση με ένα δίκτυο που έχει εφαρμοστεί ο τακτοποιητής απόσυρση.



(a) Standard Neural Net (b) After applying dropout.

Εικόνα 4.3: (α) Κανονικό νευρωνικό δίκτυο (β) δίκτυο που έχει εφαρμοστεί ο τακτοποιητής απόσυρσης

Ο τακτοποιητής απόσυρση βοηθάει στην μείωση της υπερπροσαρμογής και παρέχει έναν αποτελεσματικό τρόπο για τον συνδυασμό πολλών διαφορετικών αρχιτεκτονικών νευρωνικού δικτύου. Με τον όρο «απόσυρση» εννοούμε την προσωρινή αφαίρεση κάποιον νευρώνων από το δίκτυο και όλων των συνδέσεων εισόδου και εξόδου με αυτόν. Στην πιο απλή των περιπτώσεων στον κάθε νευρώνα δίνεται μια πιθανότητα p να κρατηθεί στο δίκτυο, η οποία δεν εξαρτάται από τους άλλους νευρώνες. Η υπερπαράμετρος p μπορεί να επιλεχθεί με βάση το validation set ή μπορεί να θεωρηθεί ίσος με 0.5. Για τους νευρώνες που βρίσκονται στο επίπεδο εισόδου, συνήθως η πιθανότητα αυτή είναι πιο κοντά στο 1 απ' ότι στο 0.5.

Η χρήση του τακτοποιητή απόσυρσης σε ένα νευρωνικό δίκτυο είναι σαν να διαλέγουμε ένα πιο «λεπτό» νευρωνικό δίκτυο από το ήδη υπάρχων. Το πιο λεπτό αυτό νευρωνικό δίκτυο περιέχει όλους τους νευρώνες που «επιβίωσαν», οπότε αν ένα αρχικό δίκτυο έχει n νευρώνες τότε υπάρχουν 2^n πιθανά «λεπτά» νευρωνικά δίκτυα. Για κάθε βήμα της εκπαίδευσης του νευρωνικού και κάθε εφαρμογή του τακτοποιητή απόσυρση δημιουργείτε ουσιαστικά βήμα βήμα ένα καινούργιο νευρωνικό δίκτυο. Οπότε με την χρήση του τακτοποιητή απόσυρση είναι σαν να έχουμε μια συλλογή από 2^n νευρωνικά, όπου καθένα από αυτά εκπαιδεύεται πολύ σπάνια ή και ποτέ.

4.5 Κανονικοποίηση παρτίδας (Batch normalization)

Η τεχνική της Κανονικοποίησης Παρτίδας ανακαλύφθηκε το 2015 [25][26], η ιδέα ήταν αντί για την κανονικοποίηση όλων των δεδομένων εισόδου στο δίκτυο, να γίνεται κανονικοποίηση των δεδομένων εισόδου σε κάθε επίπεδο του δικτύου. Το όνομα της τεχνικής αυτής, «κανονικοποίηση παρτίδας», έχει ακριβώς αυτό το όνομα επειδή κανονικοποιεί το αποτέλεσμα της συνάρτησης ενεργοποίησης του προηγούμενου επιπέδου, πριν αυτό περάσει στο επόμενο επίπεδο. Η κανονικοποίηση γίνεται με τέτοιο τρόπο ώστε η μέση τιμή της συνάρτησης ενεργοποίησης προσεγγίζει το 0 και να υπάρχουν λίγες αποκλίσεις που να προσεγγίζουν την τιμή 1.

Η εκπαίδευση ενός νευρωνικού δικτύου είναι πολύπλοκη επειδή τα δεδομένα εισόδου σε κάθε επίπεδο του δικτύου επηρεάζονται από τις παραμέτρους του προηγούμενου επιπέδου, οπότε μικρές αλλαγές στις παραμέτρους του δικτύου μεγαλώνουν όσο το δίκτυο γίνεται πιο «βαθύ». Η αλλαγή στην κατανομή των δεδομένων εισόδου στο κάθε επίπεδο δημιουργεί ένα πρόβλημα επειδή πρέπει τα επίπεδα να προσαρμόζονται συνέχεια στις καινούργιες κατανομές. Όταν αλλάζει η κατανομή των δεδομένων εισόδου σε ένα σύστημα εκπαίδευσης, λέμε ότι οι μεταβλητές του μετατοπίστηκαν. Αναφερόμαστε λοιπόν στην αλλαγή της κατανομής των δεδομένων εισόδου στους εσωτερικούς νευρώνες του δικτύου κατά την εκπαίδευση, ως Εσωτερική Μετατόπιση Μεταβλητών (Internal Covariate Shift). Η απαλοιφή αυτής υπόσχεται πιο γρήγορη εκπαίδευση και εκτελείται με την Κανονικοποίηση Παρτίδας (Batch Normalization). Ο κανονικοποιητής αυτός επιτυγχάνει την απαλοιφή με την εφαρμογή κανονικοποίησης σε κάθε βήμα από επίπεδο σε επίπεδο, που καθορίζει τις μέσες τιμές και τις αποκλίσεις των δεδομένων εισόδου. Αντί να αφαιρέσουμε τη συσχέτιση των χαρακτηριστικών κάθε δεδομένου εισόδου και εξόδου ανά επίπεδο, την κανονικοποιούμε ξεχωριστά για κάθε βαθμωτό χαρακτηριστικό, θέτοντας την μέση τιμή του 0 και την διακύμανση

1. Για ένα επίπεδο με d-διάστατο δεδομένο εισόδου X = ($X^{(1)}$…$X^{(d)}$), κανονικοποιούμε κάθε διάσταση όπως φαίνεται παρακάτω:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \qquad \text{Εξ. (4.14)}$$

Όπου E[ ] είναι η αναμενόμενη τιμή της τυχαίας μεταβλητής και Var[ ] είναι η διακύμανση. Όμως υπάρχει η πιθανότητα λόγω της κανονικοποίησης κάθε δεδομένου εισόδου να αλλάξει το χαρακτηριστικό που αντιπροσωπεύει το επίπεδο. Για να αντιμετωπιστεί αυτό πρέπει να βεβαιωθούμε ότι ο μετασχηματισμός που γίνεται στο δίκτυο μπορεί να αντιπροσωπεύει τον μετασχηματισμό του χαρακτηριστικού. Για να το πετύχουμε αυτό εισαγάγουμε για κάθε δεδομένο εισόδου $x^{(k)}$ ένα ζεύγος μεταβλητών $\gamma^{(k)}$ και $\beta^{(k)}$ οι οποίες κλιμακώνουν και μετατοπίζουν την κανονικοποιημένη τιμή και το αποτέλεσμα δίνεται στην παρακάτω εξίσωση.

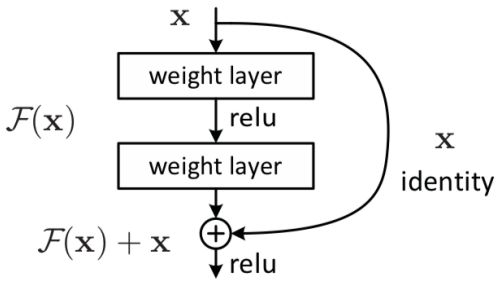
$$y^{(k)} = \gamma^{(k)} * \hat{x}^{(k)} + \beta^{(k)} \qquad \text{Εξ. (4.15)}$$

Αυτές οι νέες παράμετροι μαθαίνονται από το δίκτυο μαζί με τις αρχικές παραμέτρους του και επαναφέρουν την αντιπροσώπευση του δικτύου. Δηλαδή αν ορίσουμε όπου $\gamma^{(k)} = \sqrt{Var[x^{(k)}]}$ και όπου $\beta^{(k)} = E[x^{(k)}]$ τότε μας επιστρέφει την αρχική τιμή πριν την κανονικοποίηση.

4.6 Residual Block

Το ResAttUNet αποτελείτε και αυτό από μια διαδρομή συστολής, μια γέφυρα και μια διαδρομή διαστολής, όπως ακριβώς και το UNET. Το πρόβλημα που υπήρχε ήταν ότι στα δίκτυα UNET όσα περισσότερα επίπεδα προσπαθούσες να βάλεις στο δίκτυο τόσο πιο εύκολο γινόταν να μηδενιστεί ο ρυθμός μεταβολής του κόστους του δικτύου, δηλαδή ο τελεστής ∇ , γιατί λόγο των συνεχόμενων μετατροπών που γίνονταν μέσα στα επίπεδα του δικτύου η αρχική πληροφορία χανόταν και το δίκτυο σταματούσε να εκπαιδεύεται. Για την αντιμετώπιση του προβλήματος της εκμηδένισης του τελεστή ∇, «δανειστήκαμε» ένα χαρακτηριστικό των ResNET το οποίο ονομάζεται Residual Block [29][30].

Έστω ένα μπλοκ νευρωνικού δικτύου το οποίο έχει δεδομένα εισόδου X και θέλουμε να μάθουμε ποια θα ήταν η τιμή της εξόδου αν όλες οι μεταβλητές ήταν οι βέλτιστες, το οποίο θα συμβολίσουμε H(x). Τότε θα ορίσουμε την τιμή της διαφοράς των δυο παραπάνω τιμών ως F(x)= H(x)-x $\Rightarrow$ H(x)= F(x) + x (έστω ότι και τα δεδομένα εισόδου και τα δεδομένα εξόδου έχουν τις ίδιες διαστάσεις). Αντί λοιπόν να προσπαθούμε να προσαρμόσουμε τις τιμές των μεταβλητών του μπλοκ ώστε να βγάζουν αποτέλεσμα όσο πιο κοντά στο H(x) γίνεται, τώρα προσπαθούμε να αναγκάσουμε τα επίπεδα του μπλοκ να προσεγγίσουν την συνάρτηση F(x), η οποία ονομάζεται Residual Function. Αυτό το καταφέρνουμε χρησιμοποιώντας μια παράκαμψη (shortcut) για να μεταφέρουμε την τιμή του x στην έξοδο του μπλοκ όπως φαίνεται στην (Εικόνα 4.4).

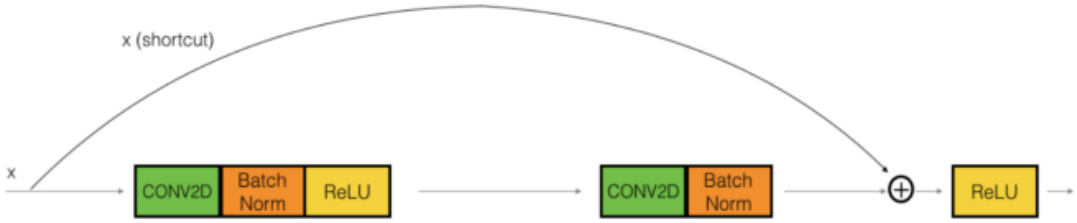


Εικόνα 4.4: Δομή residual block

Στην πραγματικότητα υπάρχουν 2 τύποι υπό-μπλοκ που χρησιμοποιούνται για την παράκαμψη και την μεταφορά αρχικών δεδομένων κατευθείαν στην έξοδο του μπλοκ, το Identity Block και το Convolutional Block.

4.6.1 Identity Block

Στην περίπτωση αυτή χρησιμοποιείται μια identity function στην παράκαμψη για την μεταφορά της αρχικής τιμής X. Το identity block αποτελεί το πιο συχνά χρησιμοποιήσιμο μπλοκ στα δίκτυα ResNET και αφορά την περίπτωση όπου η είσοδος έχει τις ίδιες διαστάσεις με την έξοδο. Στην (Εικόνα 4.5) φαίνεται ένα τέτοιο μπλοκ.

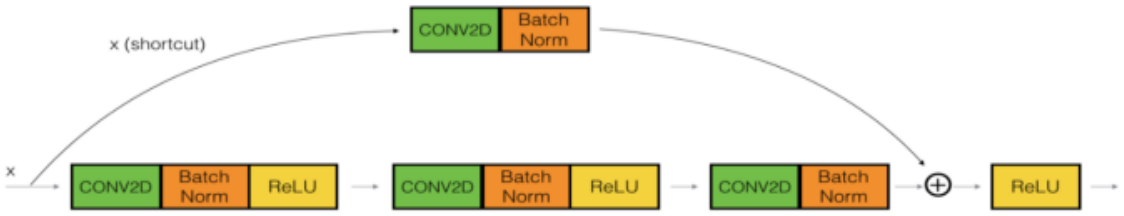


Εικόνα 4.5: Identity block

Η συνάρτηση identity function δίνεται από τον τύπο f(x)=x και στην ουσία μεταφέρει την τιμή αναλλοίωτη.

4.6.2 Convolutional Block

Χρησιμοποιούμε αυτού του είδος το μπλοκ όταν η είσοδος και η έξοδος δεν έχουν τις ίδιες διαστάσεις. Στην ουσία η μόνη διαφορά είναι ότι αντί για την συνάρτηση identity function χρησιμοποιείται η συνάρτηση συνέλιξης Conv2D όπως φαίνεται και στην (Εικόνα 4.6).



Εικόνα 4.6: Convolutional block

4.7 Παράλειψη με Προσοχή (Skip-connection Attention)

Η Προσοχή (Attention) [32][33] είναι μια μέθοδος των νευρωνικών δικτύων που προσπαθεί να μιμηθεί την ανθρώπινη προσοχή. Η εφαρμογή της προσοχής ενισχύει τα σημαντικά δεδομένα εισόδου και εξασθενεί αυτά που δεν θεωρούνται τόσο σημαντικά. Η σκέψη πίσω από αυτή την μέθοδο είναι ότι ο υπολογιστής θα έπρεπε να αφιερώνει περισσότερη υπολογιστική δύναμη στα λιγότερα αλλά πιο σημαντικά δεδομένα.

Με όλα τα δεδομένα που φτάνουν στα χαμηλότερα επίπεδα του δικτύου από τα υψηλότερα, το δίκτυο καταλήγει να έχει πολλές πληροφορίες διαθέσιμες προκειμένου να εκτελέσει τις λειτουργίες του. Ωστόσο πολύ αχρείαστη πληροφορία ή σύγχυση μπορεί να αποδειχθεί τροχοπέδη για τη αποτελεσματικότητα του μοντέλου. Το μοντέλο της προσοχής βοηθάει τον υπολογιστή να μάθει ποια κομμάτια των δεδομένων εικόνων πρέπει να «κοιτάξει» καθώς και σε ποια χαρακτηριστικά πρέπει να δώσει παραπάνω σημασία κατά την εκτέλεση της λειτουργίας του. Χρησιμοποιούμαι την μαλακή επιπρόσθετη προσοχή (soft additive attention) η οποία ζυγίζει τους χάρτες χαρακτηριστικών και ενισχύει τις συγγενείς κατηγορίες χαρακτηριστικών , σε αντίθεση με την σκληρή προσοχή (hard attention) η οποία κοιτάει μόνο ένα σημείο κάθε φορά και μηδενίζει όλα τα υπόλοιπα. Η εξίσωση (Εξ. (4.16) δίνει την προσοχή του μοντέλου.

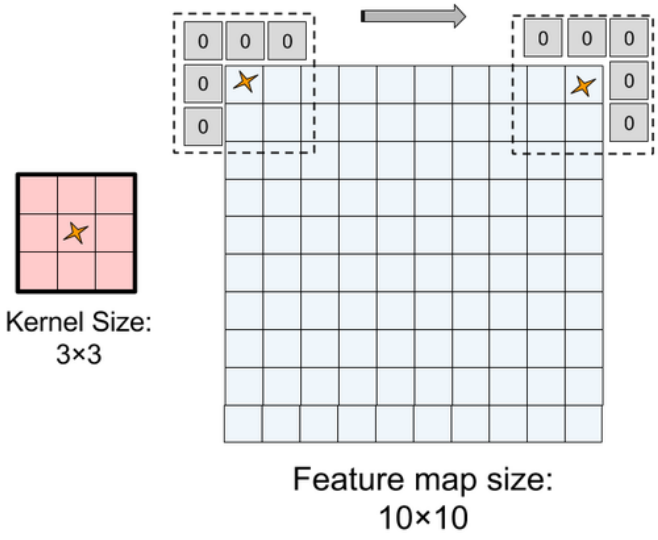
$$g_{att} = \sigma\left(W_p * x_p + W_s * x_s + b\right) \qquad \text{Εξ. (4.16)}$$

Όπου x_p είναι το δεδομένο εισόδου από το προηγούμενο επίπεδο, x_s είναι το δεδομένο εισόδου από την συνάρτηση skip-connection, b είναι η μεροληψία, σ αναφέρεται στην σιγμοειδή συνάρτηση και W_p και W_s οι παράμετροι της γραμμικής συνάρτησης μετασχηματισμού που γίνονται κατά την συνέλιξη. Το αποτέλεσμα του μοντέλου παράλειψης με προσοχή g_{att} έχει τις ίδιες χωρικές διαστάσεις με τα δεδομένα εισόδου και περιέχει βάρη από το 0 έως το 1 που υποδηλώνουν το πόσο σημαντικό είναι ένα δεδομένο στην συγκεκριμένη χωρική τοποθεσία.

4.8 Upsampling layers

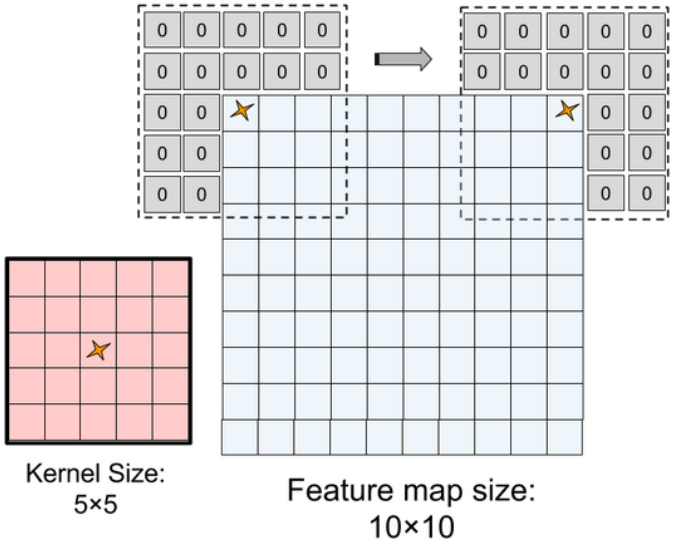
4.8.1 Μέγεθος επένδυσης συνέλιξης

Στο συγκεκριμένο δίκτυο χρησιμοποιούμε για την επίτευξη της συνέλιξης φίλτρα Kernel διαστάσεων 3x3 ή 5x5. Το ζητούμενο εδώ είναι η συνέλιξη να μην αλλάζει το μέγεθος της εικόνας που παίρνει σαν δεδομένο εισόδου, γι' αυτόν τον λόγο και το μέγεθος της επένδυσης (pad size) ορίστηκε ως «same». Η παράμετρος «same» padding σημαίνει ότι το μοντέλο προσθέτει γύρω από την εικόνα όσες «στοίβες» , δηλαδή στήλες ή γραμμές αναλόγως την διεύθυνση, χρειάζεται ώστε όλα τα εικονοστοιχεία του δεδομένου χάρτη χαρακτηριστικών να υπάρξουν μια φορά κέντρο του φίλτρου. Αυτή η διαδικασία προϋποθέτει ότι θα γίνετε ολίσθηση του φίλτρου στην εικόνα μόνο κατά μια θέση κάθε φορά , δηλαδή stride=1. Το μέγεθος της επένδυσης για ένα φίλτρο k x k θα δίνεται από τον τύπο $p = \frac{k-1}{2}$. Όπως φαίνεται στην (Εικόνα 4.7) αν έχουμε έναν χάρτη χαρακτηριστικών μεγέθους 10x10 και του εφαρμόσουμε ένα φίλτρο 3x3 τότε προκειμένου το προϊόν χάρτης της συνέλιξης να είναι επίσης 10x10 θα πρέπει να προστεθεί μια «στοίβα» μηδενικόν γύρο από τον χάρτη.



Εικόνα 4.7: Επένδυση για φίλτρο Kernel 3x3

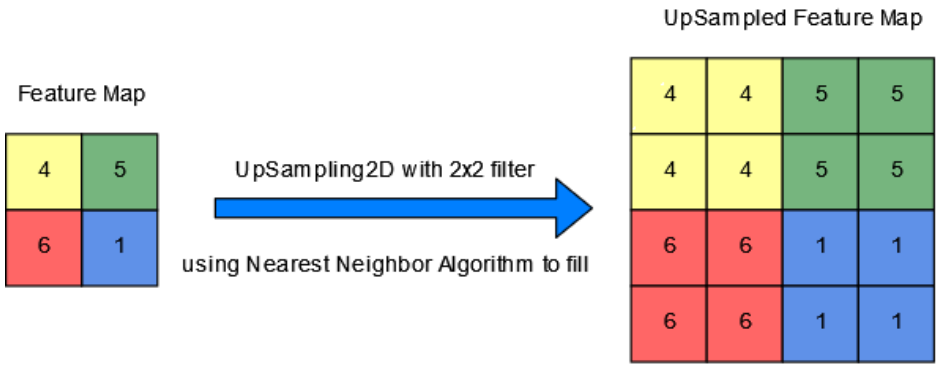
Στην περίπτωση που χρησιμοποιούσαμε φίλτρο Kernel διαστάσεων 5x5, τότε το μέγεθος της επένδυσης θα έπρεπε να είναι $p = \frac{5-1}{2} = 2$, όπως φαίνεται και στην (Εικόνα 4.8).



Εικόνα 4.81: Επένδυση για φίλτρο Kernel 5x5

4.8.2 UpSampling2D Layer

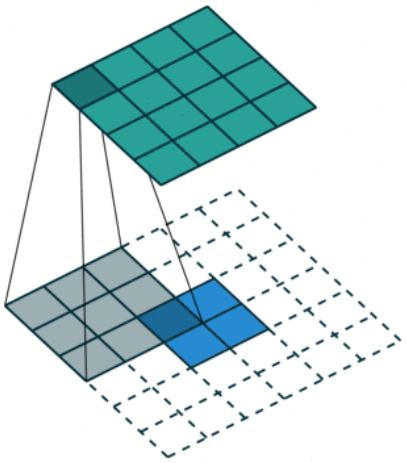
Η συνάρτηση UpSampling2D της βιβλιοθήκης Keras είναι μια συνάρτηση υπερδειγματοληψίας, δηλαδή παίρνει δεδομένο χάρτη χαρακτηριστικών και αυξάνει τις διαστάσεις συνήθως κατά δυο φορές , το φίλτρο που εφαρμόζεται δηλαδή είναι 2x2 διαστάσεων. Για κάθε ένα κελί του δεδομένου πίνακα θα δώσει πίσω σαν αποτέλεσμα έναν πίνακα διαστάσεων 2x2 όπου τα κελιά μπορεί να έχουν συμπληρωθεί με διάφορες μεθόδους, όπως αυτης που φαινέται στην παρακάτω εικόνα (Εικόνα 4.9), δηλαδή του κοντινότερου γείτονα (Nearest Neighbor).



Εικόνα 4.9: Αποτέλεσμα επίπεδου UpSampling2D Layer

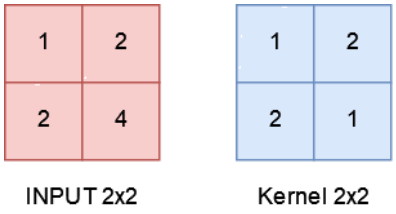
4.8.3 Conv2DTranspose Layer

Η συνάρτηση Conv2DTranspose της βιβλιοθήκης Keras δημιουργεί ένα επίπεδο συνέλιξης μετασχηματισμού (Transposed Convolution). Το επίπεδο συνέλιξης μετασχηματισμό είναι εκ φύσεως του ένα επίπεδο υπερδειγματοληψίας, που χρησιμοποιείται από τα δίκτυα τύπου UNET στην διαδρομή της διαστολής (Decoder Path). Στην συνέλιξη μετασχηματισμού, αντί το δεδομένο εισόδου να είναι μεγαλύτερο από το δεδομένο εξόδου συμβαίνει το αντίστροφο. Όπως φαίνεται στην (Εικόνα 4.10) μπορούμε να σκεφτούμε ότι προσθέτουμε αρκετό περίγραμμα στο δεδομένο εισόδου ώστε κατά την εφαρμογή των φίλτρων Kernel, η γωνία του φίλτρου να αγγίζει ίσα ίσα την γωνία του δεδομένου εισόδου.

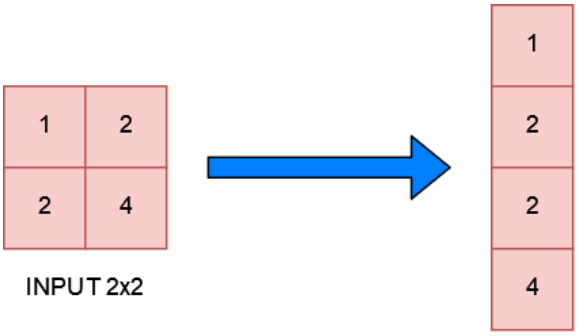


Εικόνα 4.10: Αποτέλεσμα επίπεδου Conv2DTranspose Layer

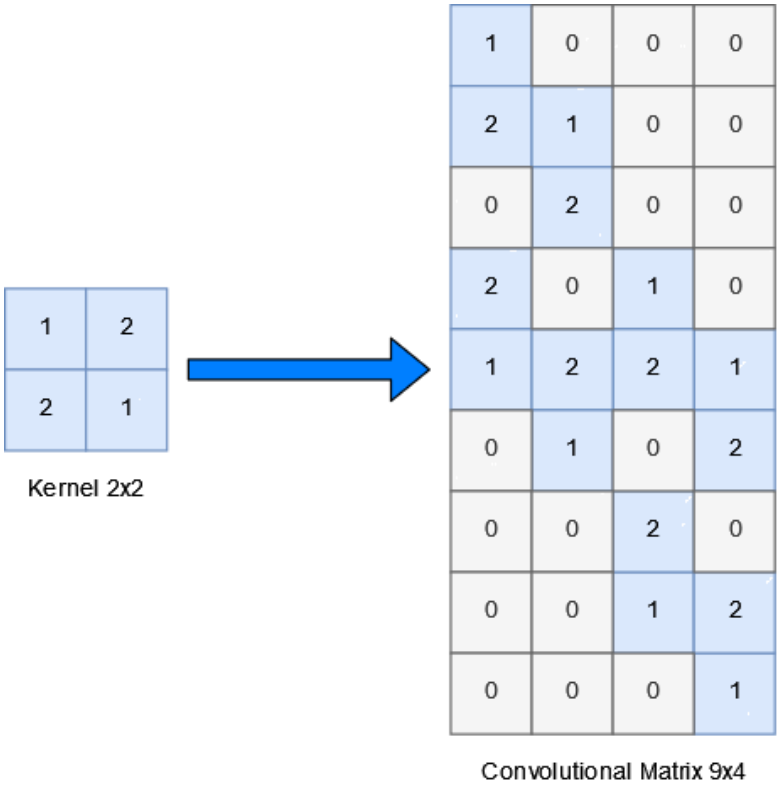
Για παράδειγμα , έστω ότι έχουμε έναν πίνακα δεδομένων εισόδου INPUT διαστάσεων 2X2 και θέλουμε να το μετατρέψουμε σε έναν πίνακα 3x3 με την μέθοδο της συνέλιξης μετασχηματισμού. Τότε χρειάζεται να εφαρμόσουμε ένα φίλτρο kernel διαστάσεων 2x2 με επένδυση p=1 και ολίσθηση s=1. Ο πίνακας εφαρμογής της συνέλιξης προκειμένου να δώσει αποτέλεσμα πίνακα διαστάσεων 3x3 με εφαρμογή σε πίνακα διαστάσεων 2x2 πρέπει να είναι διαστάσεων 9x4.



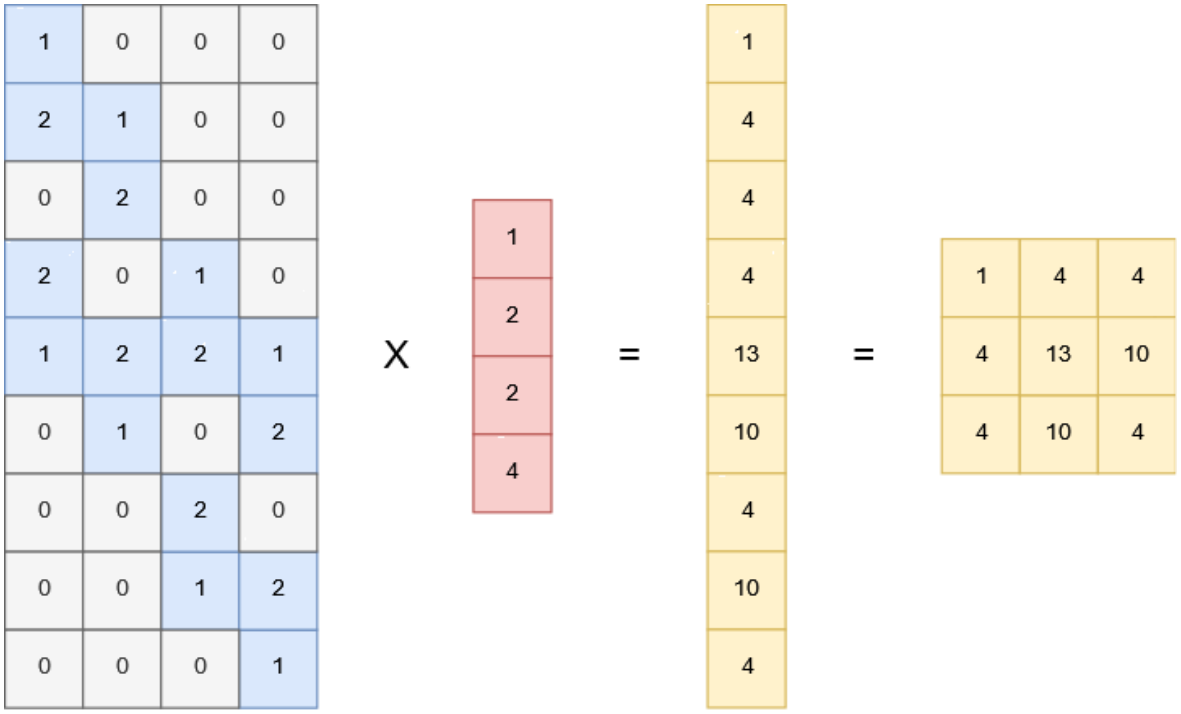
Αρχικά τα δεδομένα εισόδου μετατρέπονται σε πίνακα διαστάσεων 4x1.



Μετά το φίλτρο Kernel μετατρέπεται σε πίνακα συνέλιξης.



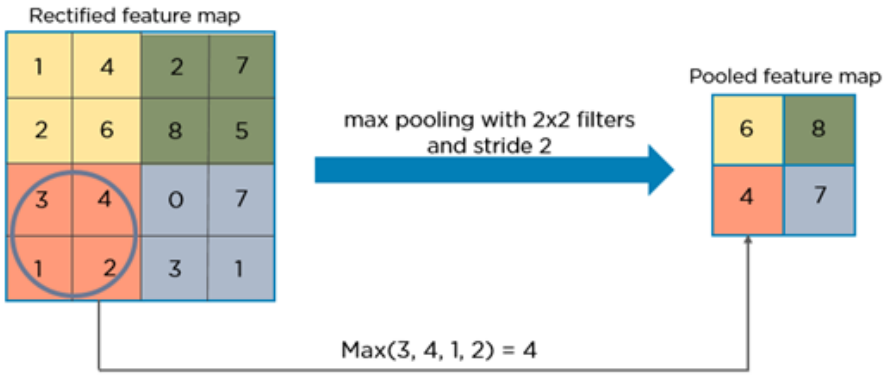
Τέλος ο πίνακας συνέλιξης 9x4 πολλαπλασιάζεται με τον πίνακα εισόδου 4x1 και μας δίνουν το αποτέλεσμα της συνέλιξης μετασχηματισμού, δηλαδή τον πίνακα 9x1 που θα μετατραπεί στον αντίστοιχο 3x3.



4.9 Downsampling layers

4.9.1 Pool layer – Max pool

Ο τύπος του επίπεδου συγκέντρωσης που χρησιμοποιήθηκε στο παρών δίκτυο είναι αυτό της Μέγιστης Συγκέντρωσης (Max Pooling). Η διαδικασία της Μέγιστης συγκέντρωσης περιγράφεται στην (Εικόνα 4.11).



Εικόνα 4.11: Αποτέλεσμα επίπεδου max pool

5 Εκπαίδευση - πρόβλεψη - σχολιασμός αποτελεσμάτων

5.1 Εκπαίδευση

Στο Κεφάλαιο 4 περιγράψαμε τα επίπεδα και όλα τα εργαλεία που χρησιμοποιήθηκαν στο νευρωνικό δίκτυο που χρησιμοποιήθηκε στην παρούσα εργασία. Όλα αυτά τα στοιχεία χρησιμοποιήθηκαν πολλές φορές σχηματίζοντας μπλοκ όπως περιγράφηκε και προηγουμένως, αναλυτικός πίνακας με την σύνοψη του δικτύου υπάρχει στο Παράρτημα Α. Το μοντέλο «AttentionResUNet» όπως και ονομάστηκε, εκπαιδεύτηκε με μέγεθος παρτίδας batch_size=8, επειδή στο σύνολο τα δεδομένα μας για την εκπαίδευση είναι 1920 εικόνες, θα έχουμε τις εξής παρτίδες ανά εποχή, $1920/8 \Rightarrow 240$ batches/Epoch. Επίσης στην συνάρτηση Early_stopping της βιβλιοθήκης keras, η οποία ορίζει σε ποιο σημείο θα σταματάει η εκπαίδευση, ορίστηκε να σταματάει μετά από 50 εποχές που δεν έχει βελτιωθεί η τιμή της μεταβλητής Validation Loss (patience=30). Στο σύνολο τους οι εποχές είναι 100 (Epochs=100), αλλά όπως αναφέραμε το πόσες συνολικά θα εκτελεστούν εξαρτάται από την τιμή της απώλειας που προκύπτει από την εφαρμογή του νευρωνικού στην βάση Επίδοσης (VALIDATION SET). Επίσης με την βοήθεια της συνάρτησης ModelCheckpoint της βιβλιοθήκης keras, το μοντέλο αποθηκευόταν κάθε φορά που βελτιωνόταν η τιμή της μεταβλητής validation loss. Τα αποτελέσματα της εκπαίδευσης σε κάθε εποχή παρουσιάζονται σε πίνακα στο Παράρτημα Α.

Πίνακας 5.1: Ενδεικτικά αποτελέσματα αποθηκευμένου μοντέλου

αποτελέσματα αποθηκευμένου μοντέλου	
Εποχές	11/100
Loss	-0.78076
Validation Loss	-0.7895

5.2 Διαδικασία δοκιμής δικτύου (Testing Process)

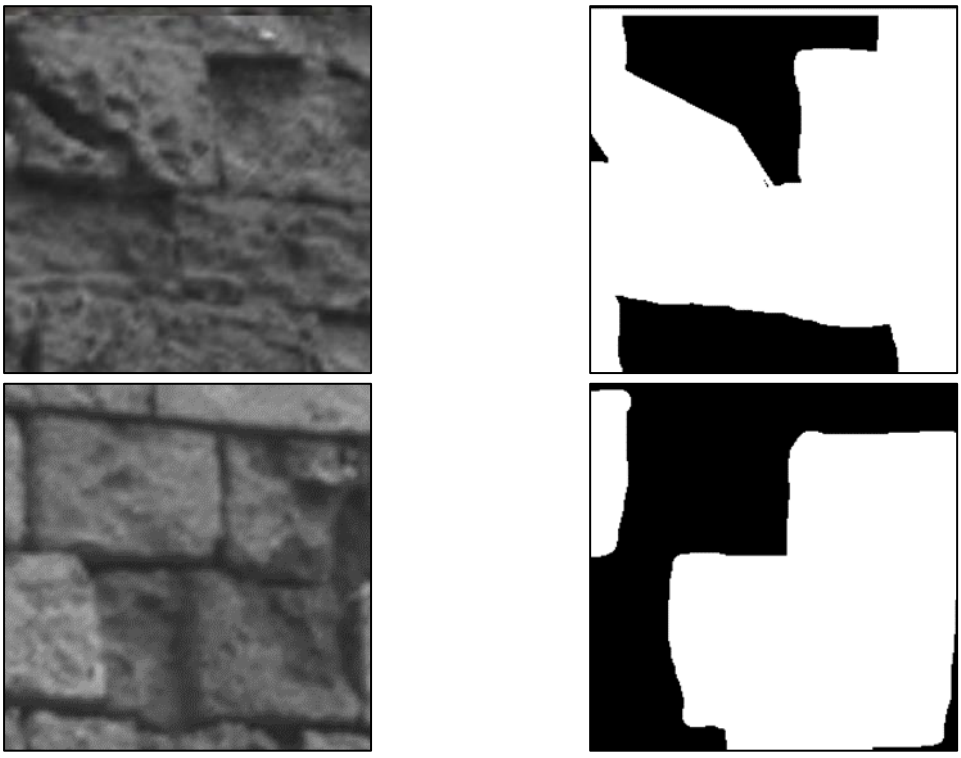
5.2.1 Προεπεξεργασία δεδομένων δοκιμής (testing set preprocessing)

Τα δεδομένα τις βάσης Δοκιμής, όπως εξηγήθηκε στο Κεφάλαιο 3, αποτελείται από μια μόνο εικόνα. Η εικόνα αυτή προκειμένου να είναι ίδιων διαστάσεων και περιέχει όσο το δυνατών παρόμοια πληροφορία με τις εικόνες που χρησιμοποιήθηκαν για την εκπαίδευση, χρειάστηκε και αυτή να κοπεί σε κομμάτια διαστάσεων 256x256 εικονοστοιχεία. Τα κομμάτια αυτά όμως δεν κόπηκαν, όπως εξηγήθηκε για τις εικόνες των άλλων βάσεων, σε κομμάτια με επικαλυπτόμενο μέρος η μια με την επόμενη της. Ο χωρισμός έγινε όπως φαίνεται στην §3.3.3 χωρίς επικαλυπτόμενο μέρος, εκτός από τα κομμάτια της εικόνας που βρίσκονται στην τελευταία στήλη και αυτά που βρίσκονται στην τελευταία γραμμή. Αυτό συνέβη, όπως εξηγήθηκε και στην παράγραφο, για να μπορέσει να υπάρξει ακέραιος αριθμός κομματιών στα οποία θα χωριστεί η αρχική εικόνα.

1,1	1,2	1,3	1,4	1,5	1,6	1,7		1,8
2,1	2,2	2,3	2,4	2,5	2,6	2,7		2,8
3,1	3,2	3,3	3,4	3,5	3,6	3,7		3,8
4,1	4,2	4,3	4,4	4,5	4,6	4,7		4,8
5,1	5,2	5,3	5,4	5,5	5,6	5,7		5,8

Εικόνα 5.1: Διαδικασία διαχωρισμού εικόνας χωρίς επικάλυψη

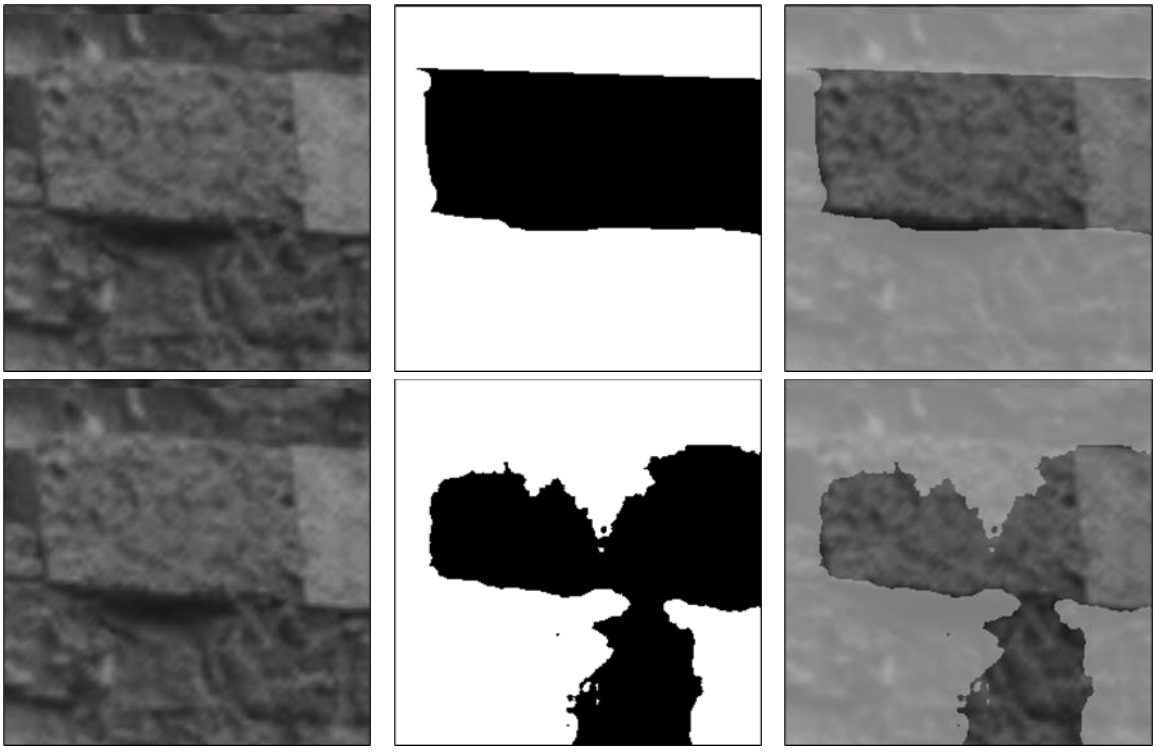
Η εικόνα λοιπόν της βάσης δοκιμής θα χωριστεί σε 8 στήλες και 5 γραμμές, στο σύνολο θα υπάρχουν 40 κομμάτια της πραγματικής εικόνας και 40 της αντίστοιχης μάσκας της. Για να μπορέσουν να γίνουν προβλέψεις πάνω στην πραγματική εικόνα και ύστερα να συγκριθούν με τις πραγματικές μάσκες, θα πρέπει αυτά να μετατραπούν σε διανύσματα τύπου NumPy Array. Το διάνυσμα των πραγματικών εικόνων ονομάζεται X_test και έχει διαστάσεις X_test.(40,256,256,3) και το διάνυσμα των αντίστοιχων μασκών ονομάζεται Y_test και έχει διαστάσεις Y_test.(40,256,256,2). Κάποια παραδείγματα κομματιών της εικόνας και των αντίστοιχων μασκών φαίνονται στις παρακάτω εικόνες.



Εικόνα 5.2: Παραδείγματα κομματιών της εικόνας δοκιμής

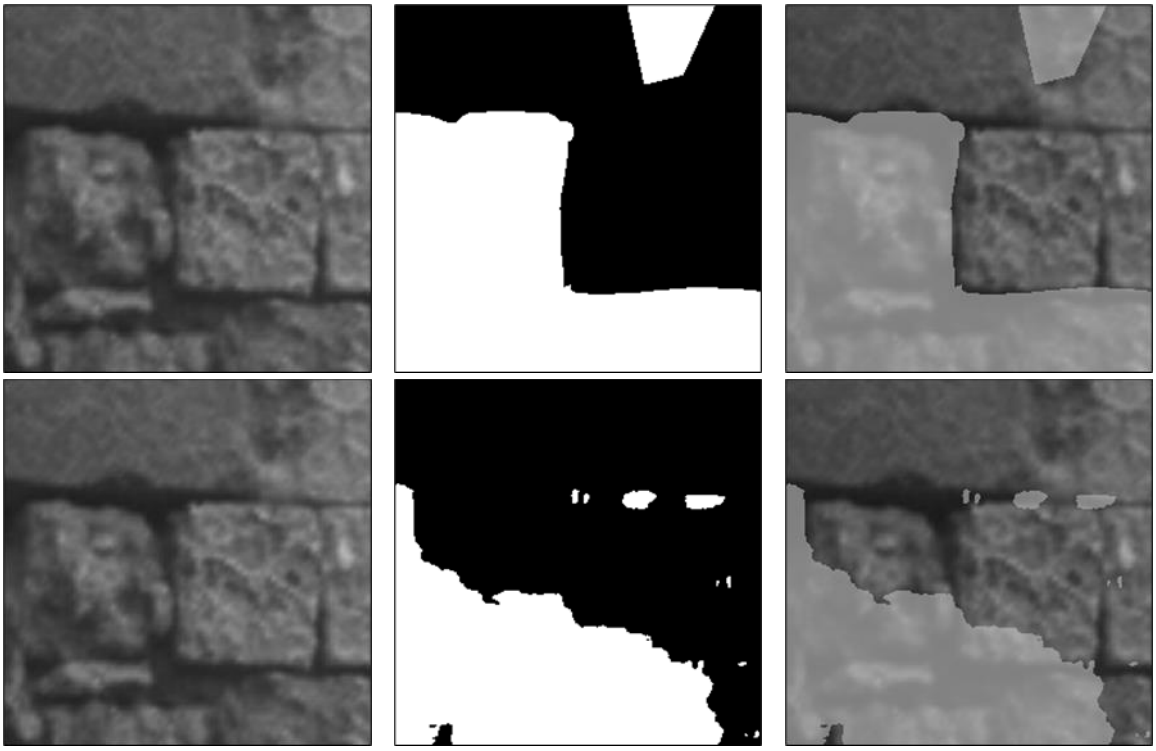
5.2.2 Δοκιμή προβλέψεων στα δεδομένα δοκιμής

Αφού τα δεδομένα δοκιμής έρθουν στην κατάλληλη μορφή, ύστερα χρησιμοποιείται το ήδη εκπαιδευμένο μοντέλο που προέκυψε από την παραπάνω διαδικασία εκπαίδευσης (§5.1). Ο αλγόριθμος διαβάζει με την σειρά κάθε διάνυσμα i της μορφής (i,256,256,3) για i=[0,40) και κάνει μια πρόβλεψη για την σημασιολογική κατάτμηση των εικονοστοιχείων της εικόνας. Αποτέλεσμα της πρόβλεψης είναι μια εικόνα ίδιων διαστάσεων (256,256), όπου όλα τα εικονοστοιχεία του έχουν τιμές 0 και 1 ανάλογα με το αν ο αλγόριθμος προέβλεψε ότι το εικονοστοιχείο απεικονίζει περιοχή του τείχους που είναι διαβρωμένο (1) ή είναι μη-διαβρωμένο (0). Κάθε ένα από αυτά τα κομμάτια στην συνέχεια αποθηκεύονται σε ένα συνολικό διάνυσμα μασκών Y_pred το οποίο έχει διαστάσεις Y_pred.(40,256,256). Ύστερα τα κομμάτια των δεδομένων μασκών Y_test μετατρέπονται σε κατάλληλη μορφή (i,256,256) και συγκρίνονται με αντίστοιχα κομμάτια του διανύσματος Y_pred.(i,256,256) με βάση τις μεταβλητές Accuracy, Precision, Recall-score και F1-score [35][36]. Παρακάτω επίσης φαίνονται κάποια παραδείγματα των αποτελεσμάτων του αλγόριθμου Predict πάνω στα κομμάτια της πραγματικής εικόνας X_test.(i,256,256,3).



(α)

Εικόνα 5.3: Τα (α) και (β) αποτελούν παραδείγματα σύγκρισης της πραγματικής μάσκας με την μάσκα που προέκυψε από την πρόβλεψη (συνεζίζεται στην επόμενη σελίδα)



(β)

Εικόνα 5.3: Τα (α) και (β) αποτελούν παραδείγματα σύγκρισης της πραγματικής μάσκας με την μάσκα που προέκυψε από την πρόβλεψη

➢ True, False/Positive, Negative

Τα δεδομένα που καλούμαστε να προβλέψουμε είναι 1 τιμή για κάθε εικονοστοιχείο μεταξύ των αριθμών 0 και 1, αντίστοιχα έχουμε και μια τιμή μεταξύ 0 και 1 για κάθε εικονοστοιχείο της δεδομένης μάσκας. Η τιμή 1 είναι η κατηγορία «YES» και η τιμή 0 είναι η κατηγορία «NO».

		Predicted Class	
Actual Class		Class=Yes	Class=No
	Class=Yes	True Positive	False Negative
	Class=No	False Positive	True Negative

- Η τιμή True Positive (TP) δίνεται όταν η πραγματική τιμή είναι 1 και ο αλγόριθμος πρόβλεψης έδωσε την τιμή 1.
- Η τιμή False Positive (FP) δίνεται όταν η πραγματική τιμή είναι 0 και ο αλγόριθμος πρόβλεψης έδωσε την τιμή 1.
- Η τιμή True Negative (TN) δίνεται όταν η πραγματική τιμή είναι 0 και ο αλγόριθμος πρόβλεψης έδωσε την τιμή 0.
- Η τιμή False Negative (FN) δίνεται όταν η πραγματική τιμή είναι 1 και ο αλγόριθμος πρόβλεψης έδωσε την τιμή 0.

- Ακρίβεια (Accuracy)

Η ακρίβεια (accuracy) δίνεται από τον παρακάτω τύπο:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad \text{Εξ. (5.1)}$$

Η ακρίβεια είναι το πιο διαισθητικό μέτρο απόδοσης και είναι απλώς ένας λόγος σωστά προβλεπόμενης παρατήρησης προς το σύνολο των παρατηρήσεων. Κάποιος μπορεί να σκεφτεί ότι, αν έχουμε υψηλή ακρίβεια, τότε το μοντέλο μας είναι το καλύτερο. Ναι, η ακρίβεια είναι ένα μεγάλο μέτρο, αλλά μόνο όταν έχουμε συμμετρικά σύνολα δεδομένων όπου οι τιμές των ψευδώς θετικών και ψευδώς αρνητικών είναι σχεδόν ίδιες. Επομένως, πρέπει να εξετάσουμε και άλλες παραμέτρους για την αξιολόγηση της απόδοσης του μοντέλου μας.

- Ακριβολογία (Precision)

Η ακριβολογία (precision) δίνεται από τον παρακάτω τύπο:

$$Precision = \frac{TP}{TP + FP} \quad \text{Εξ. (5.2)}$$

Η ακριβολογία είναι ο λόγος των σωστά προβλεπόμενων θετικών παρατηρήσεων προς το σύνολο των προβλεπόμενων θετικών παρατηρήσεων. Η υψηλή ακριβολογία σχετίζεται με το χαμηλό ψευδώς θετικό ποσοστό

- Ανάκλαση (Recall)

Η ανάκλαση (Recall) δίνεται από τον παρακάτω τύπο

$$Recall = \frac{TP}{TP + FN} \quad \text{Εξ. (5.3)}$$

Η ανάκληση ή ευαισθησία (Recall) είναι ο λόγος των σωστά προβλεπόμενων θετικών παρατηρήσεων προς όλες τις παρατηρήσεις στην πραγματική κλάση που είναι «YES».

- F1-score

Το F1-score δίνεται από τον παρακάτω τύπο:

$$F1 - score = \frac{2 * (Recall * Precision)}{Recall + Precision} \quad \text{Εξ. (5.4)}$$

Το F1-score είναι ο σταθμισμένος μέσος όρος ακριβολογίας και ανάκλησης. Επομένως, αυτή η μεταβλητή λαμβάνει υπόψη και τα ψευδώς θετικά και τα ψευδώς αρνητικά. Το F1-score είναι συνήθως πιο χρήσιμο από την ακρίβεια, ειδικά αν υπάρχει άνιση κατανομή κλάσεων. Η ακρίβεια λειτουργεί καλύτερα εάν τα ψευδώς θετικά και τα ψευδώς αρνητικά έχουν παρόμοιο κόστος. Εάν το κόστος των ψευδώς θετικών και των ψευδώς αρνητικών είναι πολύ διαφορετικό, είναι καλύτερο να εξεταστεί τόσο η ακριβολογία όσο και η ανάκληση.

Αναλυτικά τα αποτελέσματα των μεταβλητών accuracy, precision, recall και F1-score για κάθε ζεύγος εικόνων Y_pred[i,256,256] και Y_test[i,256,256] βρίσκονται στο Παράρτημα Α.

5.2.3 Επανένωση κομματιών που προβλέφθηκαν

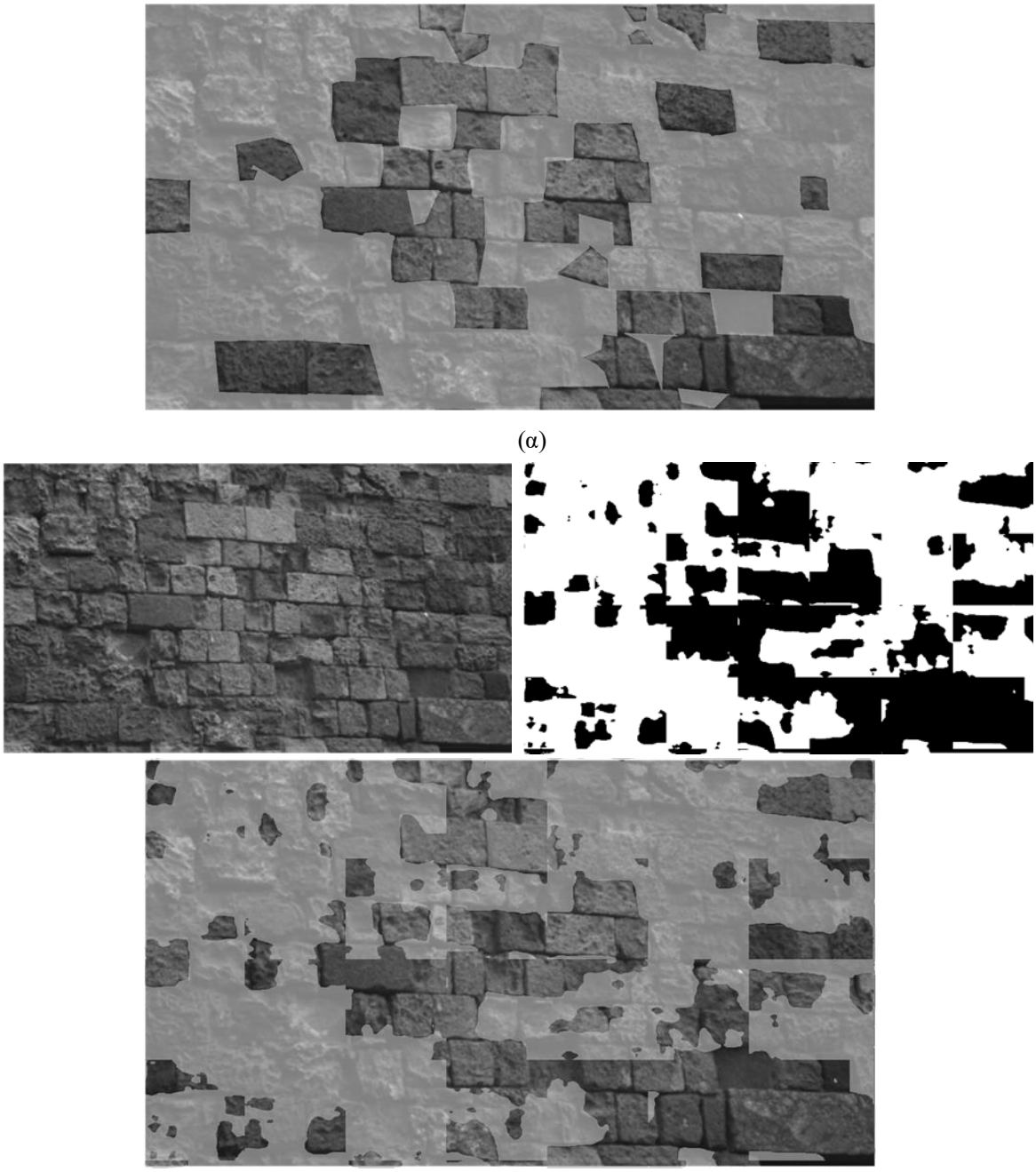
Προκειμένου να αντιληφθούμε και οπτικά το αποτέλεσμα της πρόβλεψης, οι προβλέψεις των μασκών των κομματιών της πραγματικής εικόνας θα πρέπει να ξαναενωθούν σε μια συνολική, η οποία θα συγκριθεί με την πραγματική μάσκα για να γίνει σχολιασμός του αποτελέσματος. Η επανένωση των εικόνων είναι η αντίστροφη διαδικασία της διαδικασίας διαχωρισμού που περιγράφηκε στην §5.2.1. Ιδιαίτερη προσοχή πρέπει να δοθεί στο γεγονός ότι στα κομμάτια που βρίσκονται στην τελευταία στήλη και στην τελευταία γραμμή της εικόνας, έχουν ποσοστό επικάλυψης μεταξύ τους. Μετά την επανένωση των κομματιών το αποτέλεσμα παρουσιάζεται στην παρακάτω εικόνα.

Εικόνα 5.4: Πρόβλεψη μάσκας ολόκληρης της εικόνας δοκιμής

5.2.4 Σύγκριση μάσκας πρόβλεψης και πραγματικών εικόνων

Σε αυτή την παράγραφο γίνεται σύγκριση των εικόνων (Εικόνα 5.5) που προέκυψαν από την διαδικασία της πρόβλεψης και των πραγματικών εικόνων και πραγματικών μασκών, καθώς επίσης παρουσιάζονται και οι τιμές των μεταβλητών accuracy, precision, recall και F1-score για την σύγκριση των εικόνων.

Εικόνα 5.5: (α) Πραγματική μάσκα εφαρμοσμένη πάνω σε πραγματική εικόνα (β) Μάσκα πρόβλεψης εφαρμοσμένη πάνω σε πραγματική εικόνα (συνεχίζεται)



(α)

(β)

Εικόνα 5.5: (α) Πραγματική μάσκα εφαρμοσμένη πάνω σε πραγματική εικόνα (β) Μάσκα πρόβλεψης εφαρμοσμένη πάνω σε πραγματική εικόνα

Από ότι μπορούμε να παρατηρήσουμε η πρόβλεψη της μάσκας δεν ταυτίζεται με την πραγματική μάσκα. Στην συνέχεια δίνονται οι τιμές των μεταβλητών που αναφέρθηκε προηγουμένως για την σύγκριση των εικόνων.

Πίνακας 5.2: Αποτελέσματα αξιολόγησης για ολόκληρη την εικόνα

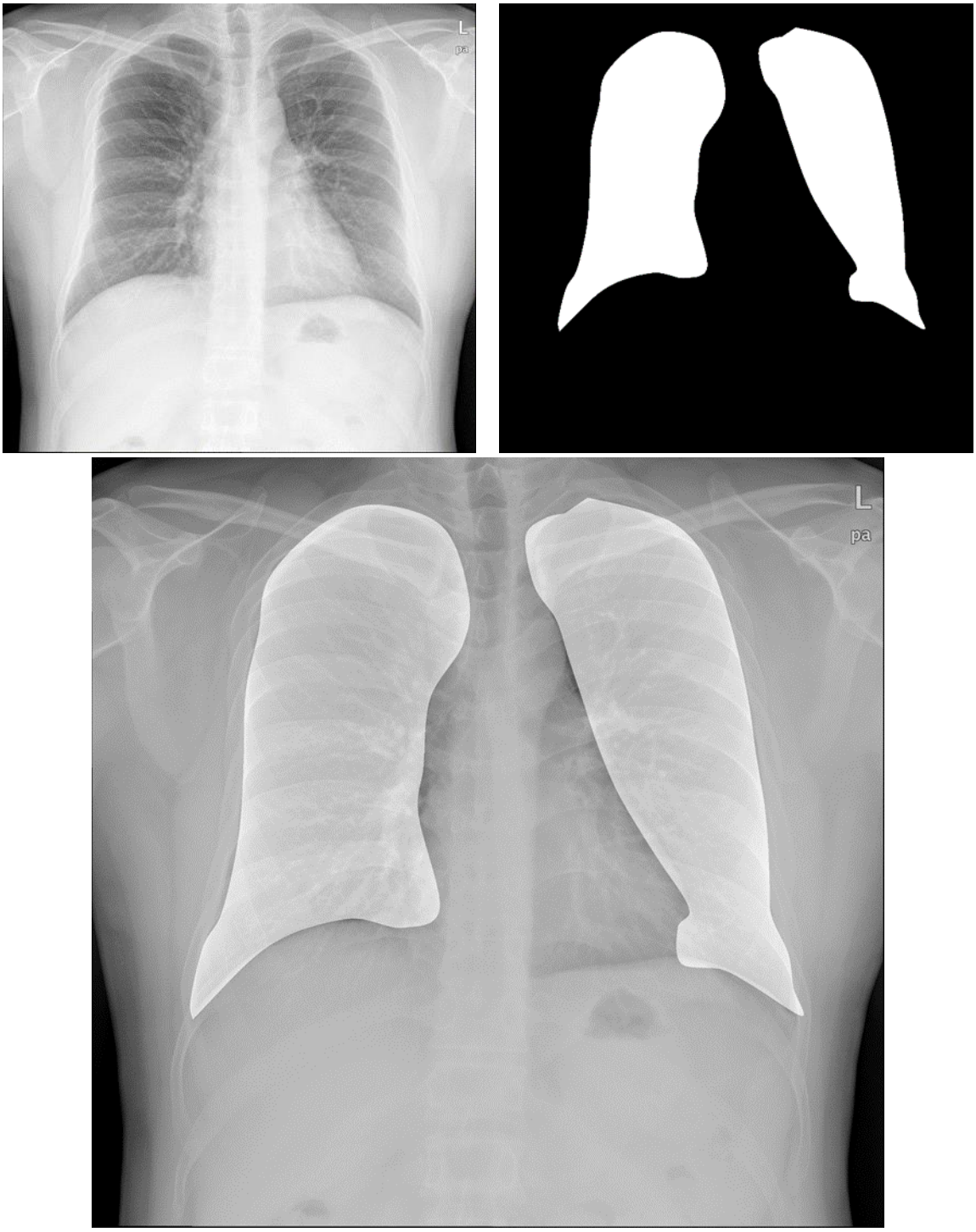
	Predicted Mask vs Real Mask
Accuracy	0.7505
Precision	0.7485
Recall	0.7505
F1-score	0.7480

5.3 Έλεγχος Δικτύου με χρήση άλλων δεδομένων

Για τον έλεγχο της σωστής λειτουργίας του κώδικα, ο ίδιος ακριβώς αλγόριθμος χρησιμοποιήθηκε για την σημασιολογική κατάτμηση εικονοστοιχείων εικόνων που προέρχονται από κοινόχρηστη βάση δεδομένων που βρέθηκε στον ιστότοπο Kaggle. Οι καινούργιες εικόνες αυτές, αποτελούν ακτινογραφίες θώρακα ανθρώπων και κάθε μια από αυτές συνοδεύεται από μια αντίστοιχη μάσκα που απεικονίζει με λευκά εικονοστοιχεία τους πνεύμονες και με μαύρα την υπόλοιπη εικόνα. Η βάση δεδομένων όπως και πριν αποτελείται από τον φάκελο INPUT που περιέχει τις πραγματικές εικόνες και από τον φάκελο OUTPUT που περιέχει τις αντίστοιχες πραγματικές μάσκες των πραγματικών εικόνων. Ο κάθε φάκελος INPUT και OUTPUT περιέχει ο καθένας 3 φακέλους TRAIN, VALIDATION και TEST, όπου αυτοί περιέχουν 50, 20 και 10 εικόνες αντιστοίχως. Όλες οι εικόνες των φακέλων TRAIN και VALIDATION κόπηκαν σε 100 κομμάτια με επικάλυψη 30% μεταξύ τους και όλες οι εικόνες των φακέλων TEST κόπηκαν σε 56 κομμάτια με 0% επικάλυψη μεταξύ τους. Τα διανύσματα που δημιουργήθηκαν για την εκπαίδευση και την δοκιμή είναι τα εξής, X_train[5000,256,256,3], Y_train[5000,256,256,2], X_val[2000,256,256,3], Y_val[2000,256,256,2] και X_test[560,256,256,3], Y_test[560,256,256,2]. Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα της εκπαίδευσης με την χρήση της δοκιμαστικής βάσης δεδομένων και στην Εικόνα 5.6 φαίνεται ενδεικτικά ένα αποτέλεσμα της πρόβλεψης της μάσκας για μια εικόνα από την βάση που περιεγράφηκε παραπάνω

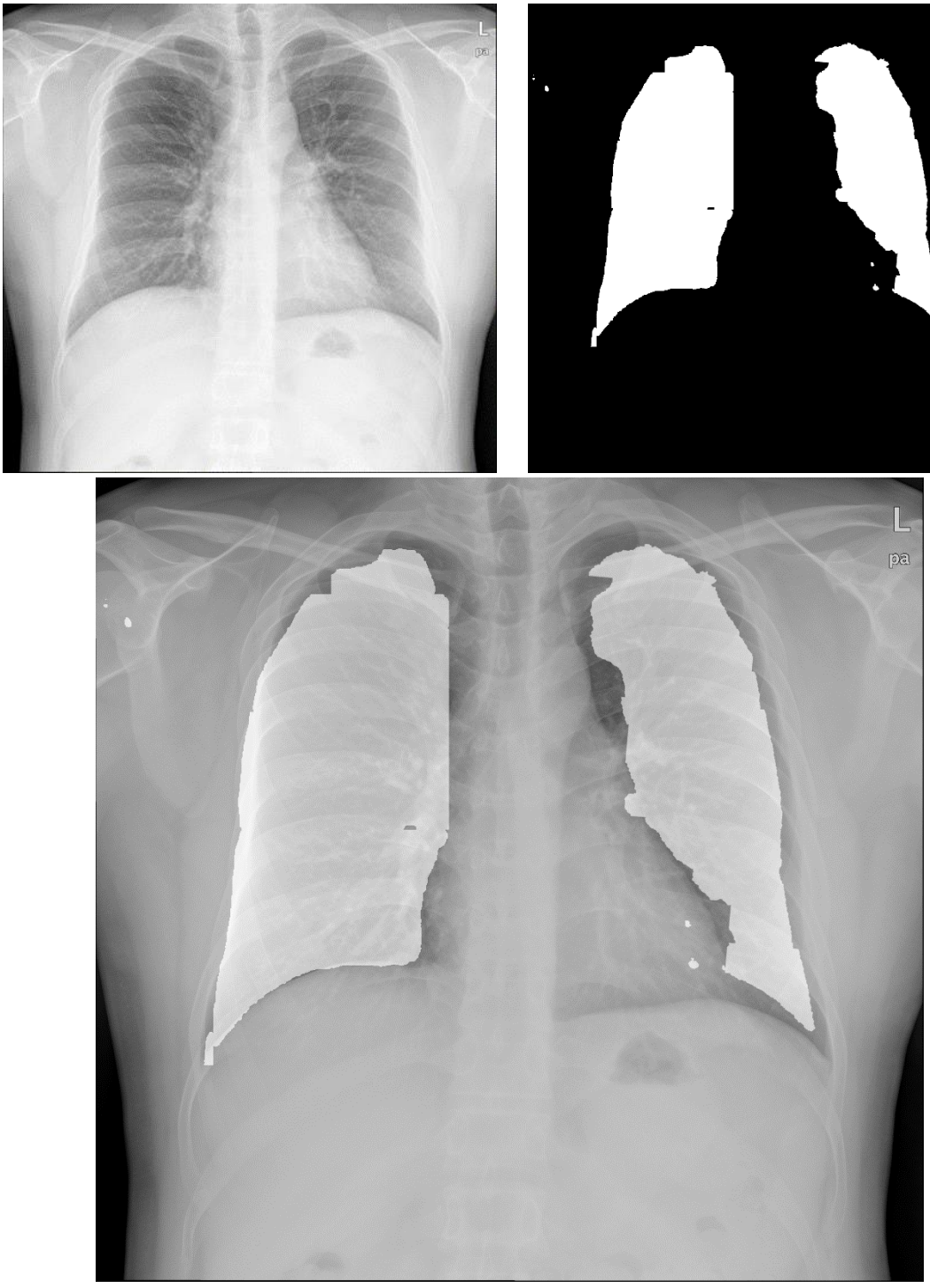
Πίνακας 5.3: Αποτελέσματα Εκπαίδευσης με την χρήση ιατρικής βάσης δεδομένων

Epochs	17
Loss	-0,9602
Validation Loss	-0,9709



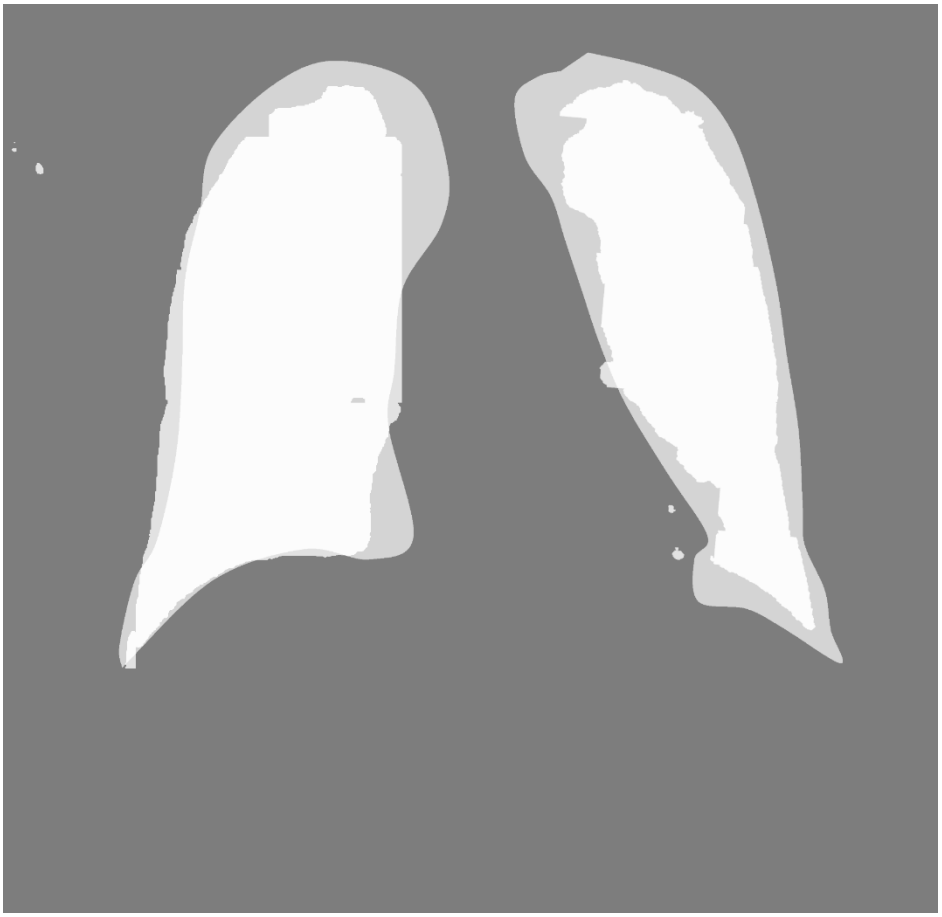
(α)

Εικόνα 5.6: (α) Πραγματική μάσκα εφαρμοσμένη πάνω σε πραγματική εικόνα (β) Μάσκα πρόβλεψης εφαρμοσμένη πάνω σε πραγματική εικόνα (γ)Πραγματική μάσκα σε σύγκριση με την μάσκα πρόβλεψης (συνεχίζεται)



(β)

Εικόνα 5.6: (α) Πραγματική μάσκα εφαρμοσμένη πάνω σε πραγματική εικόνα (β) Μάσκα πρόβλεψης εφαρμοσμένη πάνω σε πραγματική εικόνα (γ)Πραγματική μάσκα σε σύγκριση με την μάσκα πρόβλεψης (συνεχίζεται)



(γ)

Εικόνα 5.6: (α) Πραγματική μάσκα εφαρμοσμένη πάνω σε πραγματική εικόνα (β) Μάσκα πρόβλεψης εφαρμοσμένη πάνω σε πραγματική εικόνα (γ)Πραγματική μάσκα σε σύγκριση με την μάσκα πρόβλεψης

Από την παραπάνω διαδικασία συμπεραίνουμε ότι ο αλγόριθμος βγάζει σχετικά καλά αποτελέσματα με τις μεταβλητές accuracy, precision, recall και f1 να έχουν κατά μέσο όρο τις παρακάτω τιμές.

Πίνακας 5.4: Μέσοι όροι μεταβλητών αξιολόγησης

	Real Lung Mask VS Predicted Lung Mask
Mean Accuracy	0.9780
Mean Precision	0.9607
Mean Recall	0.9928
Mean F1-score	0.9765

6 Συμπεράσματα και προοπτικές εξέλιξης

Στην παρούσα Διπλωματική Εργασία παρουσιάζεται η εκπαίδευση ενός νευρωνικού δικτύου και η χρησιμοποίησή του για την πραγματοποίηση προβλέψεων. Συγκεκριμένα πραγματοποιήθηκαν δύο εφαρμογές η πρώτη στο φρούριο του Αγίου Νικολάου στην Ρόδο και η δεύτερη σε ιατρικά δεδομένα. Από την αξιολόγηση των αποτελεσμάτων των δύο εφαρμογών αρχικά συμπεραίνουμε ότι ο αλγόριθμος που χρησιμοποιήθηκε για την εκπαίδευση του νευρωνικού δικτύου λειτούργησε ικανοποιητικά. Παρ' όλα αυτά, από την σύγκριση των αποτελεσμάτων, παρατηρούμε ότι του Αγίου Νικολάου δεν είναι τόσο ικανοποιητικά όσο των Ιατρικών δεδομένων. Αυτό μπορεί να οφείλεται σε δύο λόγους:

- Η βάση δεδομένων από το φρούριο του Αγίου Νικολάου είναι πολύ μικρότερη από την ιατρική βάση δεδομένων, οπότε και η εκπαίδευση με την πρώτη είναι πολύ πιο δύσκολη.
- Τα μοτίβα που έχει να μάθει το νευρωνικό δίκτυο στην περίπτωση του τείχους είναι πολύ πιο πολύπλοκα απ' ότι τα μοτίβα που αφορούν την αναγνώριση των πνευμόνων στην δεύτερη περίπτωση.

Από τα παραπάνω προκύπτει ότι η χρήση περισσότερων δεδομένων για την εκπαίδευση του νευρωνικού δικτύου είναι απαραίτητη, προκειμένου αυτό να μπορεί να αναγνωρίσει κάτι τόσο πολύπλοκο όσο είναι η διάβρωση των λίθων στο τείχος.

Για την αποτελεσματικότερη εκπαίδευση του νευρωνικού δικτύου, θα μπορούσαν να εφαρμοστούν επιπλέον πρακτικές. Για παράδειγμα η χρήση πολυφασματικών εικόνων, όπως προκύπτουν από την χρήση του συστήματος HyperView δηλαδή διαστάσεων 1859x1044x41 θα βοηθούσε στην ανάλυση περισσότερων δεδομένων και τελικά στην καλύτερη εκπαίδευση του μοντέλου. Επίσης ο διαχωρισμός των εικόνων μασκών κάθε πραγματικής εικόνας σε περισσότερες κατηγορίες διάβρωσης, όπως Πολύ Καλή Κατάσταση , Καλή Κατάσταση, Μέτρια Κατάσταση, Κακή Κατάσταση και Πολύ Κακή Κατάσταση. Αυτό θα ήταν δυνατό να συμβεί με την ύπαρξη αρκετών δεδομένων εικόνων, ώστε να είναι δυνατή η σημασιολογική κατάτμηση των υπερφασματικών δεδομένων σε πολλαπλές σημασιολογικές κατηγορίες.

7 Βιβλιογραφικές αναφορές

[1] A. Barr, E.A. Feigenbaum (2014). "*The Handbook of Artificial Intelligence: Volume 1*", ISBN: 9781483214375

[2] M. Mohri, A. Rostamizadeh, A. Talwalkar (2018). "*Foundations of Machine Learning*", ISBN: 9780262351362

[3] I. Goodfellow, Y. Bengio, A. Courville (2016). "*Deep Learning*", ISBN: 9780262337373

[4] R. Szeliski (2010). "*Computer Vision: Algorithms and Applications*", ISBN: 9781848829350

[5] R.C. Staudemeyer, E.R. Morris (2019), "*Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks*"

[6] https://towardsdatascience.com/classical-neural-network-what-really-are-nodes-and-layers-ec51c6122e09

[7] https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7

[8] https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

[9] https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

[10] https://towardsdatascience.com/understand-data-normalization-in-machine-learning-8ff3062101f0

[11] https://towardsdatascience.com/regularization-an-important-concept-in-machine-learning-5891628907ea

[12] https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

[13] https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0

[14] https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python?utm_source=adwords_ppc&utm_campaignid=898687156&utm_adgroupid=48947256715&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=229765585183&utm_targetid=dsa-473406581915&utm_loc_interest_ms=&utm_loc_physical_ms=9067699&gclid=Cj0KCQjwytOEBhD5ARIsANnRjVhthHCEBDR1D-1mIh-HEkKnbqtKrMRYJgRDunYk8CECDlEDGoDwBu88aAtZNEALw_wcB

[15] https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[16] C.L. Zitnick, P. Dollár (2014). "*Edge Boxes: Locating Object Proposals from Edges*". In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) ECCV 2014: Computer Vision –

ECCV 2014. Lecture Notes in Computer Science, vol 8693, p. 391-405 Springer, Cham, doi: 10.1007/978-3-319-10602-1_26

[17] J. Redmon, S. Divvala, R. Girshick, A. Farhadi (2016). "*You Only Look Once: Unified, Real-Time Object Detection*"

[18] Fukushima K. (1980). *"Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position"*, Biological Cybernetics, **36**(4), p. 193-202, doi: 10.1007/BF00344251

[19] R. Yamashita, M. Nishio, R.K.G. Do, K. Togashi (2018). "*Convolutional neural networks: an overview and application in radiology*", Insights into Imaging, **9**(4), p. 611-629, doi: 10.1007/s13244-018-0639-9

[20] O. Ronneberger, P. Fischer, T. Brox (2015). "*U-Net: Convolutional Networks for Biomedical Image Segmentation*", LNCS, vol. 9351, p.234-241, doi: 10.1007/978-3-319-24574-4_28

[21] S. Jadon (2020). "*A survey of loss functions for semantic segmentation*", IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, p.1-7, doi: 10.1109/CIBCB48159.2020.9277638

[22] D.P. Kingma, J. Ba (2017). "*Adam: A Method for Stochastic Optimization*"

[23] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[24] https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[25] S. Ioffe, C. Szegedy (2015). "*Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*".

[26] https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014), "*Dropout: A Simple Way to Prevent Neural Networks from Overfitting*", Journal of Machine Learning Research, **15**(56), p. 1929-1958

[28] https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[29] K. He, X. Zhang, S. Ren and J. Sun (2016), "Deep Residual Learning for Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 770-778, doi: 10.1109/CVPR.2016.90.

[30] https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec

[31] Z. Zhang, Q. Liu and Y. Wang (2018). "Road Extraction by Deep Residual U-Net," in IEEE Geoscience and Remote Sensing Letters, **15**(5), pp. 749-753, doi: 10.1109/LGRS.2018.2802944.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin (2017). "*Attention is All you Need*", Curran Associates, Inc., **30**

[33] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio (2015). "*Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*", PMLR, **37**, p. 2048-2057

[34] Y. Liu, D. S. Tan, J. Chen, W. Cheng, K. Hua (2019). *"Segmenting Hepatic Lesions Using Residual Attention U-Net with an Adaptive Weighted Dice Loss."*, IEEE International Conference on Image Processing (ICIP), p. 3322-3326, doi: 10.1109/ICIP.2019.8803471.

[35] https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[36] http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

[37] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis (2018). "*Deep Learning for Computer Vision: A Brief Review*", Computational Intelligence and Neuroscience, vol. 2018, Article ID 7068349, p. 13, doi: 10.1155/2018/7068349

[38] K. Makantasis, K. Karantzalos, A. Doulamis and N. Doulamis (2015). "*Deep supervised learning for hyperspectral data classification through convolutional neural networks*", *IEEE* International Geoscience and Remote Sensing Symposium *(IGARSS)*, p. 4959-4962, doi: 10.1109/IGARSS.2015.7326945.

7 Βιβλιογραφικές αναφορές

Παράρτημα Α: Αρχιτεκτονική προτεινόμενου μοντέλου

Model: "AttentionResUNet"			
Layer(type)	Output shape	Parameters	Connected to
input_1 (InputLayer)	(None, 256, 256, 3)	0	
lambda (Lambda)	(None, 256, 256, 3)	0	input_1[0][0]
conv2d (Conv2D)	(None, 256, 256, 64)	4864	lambda[0][0]
batch_normalization (BatchNorm)	(None, 256, 256, 64)	256	conv2d[0][0]
activation (Activation)	(None, 256, 256, 64)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 256, 256, 64)	102464	activation[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 64)	256	lambda[0][0]
batch_normalization_1 (BatchNorm)	(None, 256, 256, 64)	256	conv2d_1[0][0]
batch_normalization_2 (BatchNorm)	(None, 256, 256, 64)	256	conv2d_2[0][0]
activation_1 (Activation)	(None, 256, 256, 64)	0	batch_normalization_1[0][0]
add (Add)	(None, 256, 256, 64)	0	batch_normalization_2[0][0] activation_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	add[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 128)	204928	max_pooling2d[0][0]
batch_normalization_3 (BatchNorm)	(None, 128, 128, 128)	512	conv2d_3[0][0]
activation_2 (Activation)	(None, 128, 128, 128)	0	batch_normalization_3[0][0]
conv2d_4 (Conv2D)	(None, 128, 128, 128)	409728	activation_2[0][0]
conv2d_5 (Conv2D)	(None, 128, 128, 128)	8320	max_pooling2d[0][0]
batch_normalization_4 (BatchNorm)	(None, 128, 128, 128)	512	conv2d_4[0][0]
batch_normalization_5 (BatchNorm)	(None, 128, 128, 128)	512	conv2d_5[0][0]
activation_3 (Activation)	(None, 128, 128, 128)	0	batch_normalization_4[0][0]

add_1 (Add)	(None, 128, 128, 128)	0	batch_normalization_5[0][0] activation_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 128)	0	add_1[0][0]
conv2d_6 (Conv2D)	(None, 64, 64, 256)	819456	max_pooling2d_1[0][0]
batch_normalization_6 (BatchNorm)	(None, 64, 64, 256)	1024	conv2d_6[0][0]
activation_4 (Activation)	(None, 64, 64, 256)	0	batch_normalization_6[0][0]
conv2d_7 (Conv2D)	(None, 64, 64, 256)	1638656	activation_4[0][0]
conv2d_8 (Conv2D)	(None, 64, 64, 256)	33024	max_pooling2d_1[0][0]
batch_normalization_7 (BatchNorm)	(None, 64, 64, 256)	1024	conv2d_7[0][0]
batch_normalization_8 (BatchNorm)	(None, 64, 64, 256)		conv2d_8[0][0]
activation_5 (Activation)	(None, 64, 64, 256)	1024	batch_normalization_7[0][0]
add_2 (Add)	(None, 64, 64, 256)	0	batch_normalization_8[0][0] activation_5[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 256)	0	add_2[0][0]
conv2d_9 (Conv2D)	(None, 32, 32, 512)	0	max_pooling2d_2[0][0]
batch_normalization_9 (BatchNorm)	(None, 32, 32, 512)	3277312	conv2d_9[0][0]
activation_6 (Activation)	(None, 32, 32, 512)	2048	batch_normalization_9[0][0]
conv2d_10 (Conv2D)	(None, 32, 32, 512)	0	activation_6[0][0]
conv2d_11 (Conv2D)	(None, 32, 32, 512)	6554112	max_pooling2d_2[0][0]
batch_normalization_10 (BatchNorm)	(None, 32, 32, 512)	131584	conv2d_10[0][0]
batch_normalization_11 (BatchNorm)	(None, 32, 32, 512)	2048	conv2d_11[0][0]
activation_7 (Activation)	(None, 32, 32, 512)	2048	batch_normalization_10[0][0]

add_3 (Add)	(None, 32, 32, 512)	0	batch_normalization_11[0][0] activation_7[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 512)	0	add_3[0][0]
conv2d_12 (Conv2D)	(None, 16, 16, 1024)	0	max_pooling2d_3[0][0]
batch_normalization_12 (BatchNorm)	(None, 16, 16, 1024)	13108224	conv2d_12[0][0]
activation_8 (Activation)	(None, 16, 16, 1024)	4096	batch_normalization_12[0][0]
conv2d_13 (Conv2D)	(None, 16, 16, 1024)	0	activation_8[0][0]
conv2d_14 (Conv2D)	(None, 16, 16, 1024)	26215424	max_pooling2d_3[0][0]
batch_normalization_13 (BatchNorm)	(None, 16, 16, 1024)	525312	conv2d_13[0][0]
batch_normalization_14 (BatchNorm)	(None, 16, 16, 1024)	4096	conv2d_14[0][0]
activation_9 (Activation)	(None, 16, 16, 1024)	4096	batch_normalization_13[0][0]
add_4 (Add)	(None, 16, 16, 1024)	0	batch_normalization_14[0][0] activation_9[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 512)	0	add_4[0][0]
batch_normalization_15 (BatchNorm)	(None, 16, 16, 512)	524800	conv2d_15[0][0]
activation_10 (Activation)	(None, 16, 16, 512)	2048	batch_normalization_15[0][0]
conv2d_17 (Conv2D)	(None, 16, 16, 512)	0	activation_10[0][0]
conv2d_transpose (Conv2DTranspo	(None, 16, 16, 512)	262656	conv2d_17[0][0]
conv2d_16 (Conv2D)	(None, 16, 16, 512)	2359808	add_3[0][0]
add_5 (Add)	(None, 16, 16, 512)	1049088	conv2d_transpose[0][0] conv2d_16[0][0]
activation_11 (Activation)	(None, 16, 16, 512)	0	add_5[0][0]
conv2d_18 (Conv2D)	(None, 16, 16, 1)	0	activation_11[0][0]
activation_12 (Activation)	(None, 16, 16, 1)	513	conv2d_18[0][0]

up_sampling2d (UpSampling2D)	(None, 32, 32, 1)	0	activation_12[0][0]
lambda_1 (Lambda)	(None, 32, 32, 512)	0	up_sampling2d[0][0]
multiply (Multiply)	(None, 32, 32, 512)	0	lambda_1[0][0] add_3[0][0]
conv2d_19 (Conv2D)	(None, 32, 32, 512)	0	multiply[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 32, 32, 1024)	262656	add_4[0][0]
batch_normalization_16 (BatchNorm)	(None, 32, 32, 512)	0	conv2d_19[0][0]
concatenate (Concatenate)	(None, 32, 32, 1536)	2048	up_sampling2d_1[0][0] batch_normalization_16[0][0]
conv2d_20 (Conv2D)	(None, 32, 32, 512)	0	concatenate[0][0]
batch_normalization_17 (BatchNorm)	(None, 32, 32, 512)	19661312	conv2d_20[0][0]
activation_13 (Activation)	(None, 32, 32, 512)	2048	batch_normalization_17[0][0]
conv2d_21 (Conv2D)	(None, 32, 32, 512)	0	activation_13[0][0]
conv2d_22 (Conv2D)	(None, 32, 32, 512)	6554112	concatenate[0][0]
batch_normalization_18 (BatchNorm)	(None, 32, 32, 512)	786944	conv2d_21[0][0]
batch_normalization_19 (BatchNorm)	(None, 32, 32, 512)	2048	conv2d_22[0][0]
activation_14 (Activation)	(None, 32, 32, 512)	2048	batch_normalization_18[0][0]
add_6 (Add)	(None, 32, 32, 512)	0	batch_normalization_19[0][0] activation_14[0][0]
conv2d_23 (Conv2D)	(None, 32, 32, 256)	0	add_6[0][0]
batch_normalization_20 (BatchNo	(None, 32, 32, 256)	131328	conv2d_23[0][0]
activation_15 (Activation)	(None, 32, 32, 256)	1024	batch_normalization_20[0][0]
conv2d_25 (Conv2D)	(None, 32, 32, 256)	0	activation_15[0][0]

conv2d_transpose_1 (Conv2DTrans)	(None, 32, 32, 256)	65792	conv2d_25[0][0]
conv2d_24 (Conv2D)	(None, 32, 32, 256)	590080	add_2[0][0]
add_7 (Add)	(None, 32, 32, 256)	262400	conv2d_transpose_1[0][0] conv2d_24[0][0]
	(None, 32, 32, 256)	0	add_7[0][0]
activation_16 (Activation)	(None, 32, 32, 1)	0	activation_16[0][0]
conv2d_26 (Conv2D)	(None, 32, 32, 1)	257	conv2d_26[0][0]
activation_17 (Activation)	(None, 64, 64, 1)	0	activation_17[0][0]
up_sampling2d_2 (UpSampling2D)	(None, 64, 64, 256)	0	up_sampling2d_2[0][0]
lambda_2 (Lambda)	(None, 64, 64, 256)	0	lambda_2[0][0] add_2[0][0]
multiply_1 (Multiply)	(None, 64, 64, 256)	0	multiply_1[0][0]
conv2d_27 (Conv2D)	(None, 64, 64, 512)	65792	add_6[0][0]
up_sampling2d_3 (UpSampling2D)	(None, 64, 64, 256)	0	conv2d_27[0][0]
batch_normalization_21 (BatchNorm)	(None, 64, 64, 768)	1024	up_sampling2d_3[0][0] batch_normalization_21[0][0]
concatenate_1 (Concatenate)	(None, 64, 64, 256)	0	concatenate_1[0][0]
conv2d_28 (Conv2D)	(None, 64, 64, 256)	4915456	conv2d_28[0][0]
batch_normalization_22 (BatchNorm)	(None, 64, 64, 256)	1024	batch_normalization_22[0][0]
activation_18 (Activation)	(None, 64, 64, 256)	0	activation_18[0][0]
conv2d_29 (Conv2D)	(None, 64, 64, 256)	1638656	concatenate_1[0][0]
conv2d_30 (Conv2D)	(None, 64, 64, 256)	196864	conv2d_29[0][0]
batch_normalization_23 (BatchNorm)	(None, 64, 64, 256)	1024	conv2d_30[0][0]
batch_normalization_24 (BatchNorm)	(None, 64, 64, 256)	1024	batch_normalization_23[0][0]

activation_19 (Activation)	(None, 64, 64, 256)	0	batch_normalization_24[0][0] activation_19[0][0]
add_8 (Add)	(None, 64, 64, 128)	0	add_8[0][0]
conv2d_31 (Conv2D)	(None, 64, 64, 128)	32896	conv2d_31[0][0]
batch_normalization_25 (BatchNo	(None, 64, 64, 128)	512	batch_normalization_25[0][0]
activation_20 (Activation)	(None, 64, 64, 128)	0	activation_20[0][0]
conv2d_33 (Conv2D)	(None, 64, 64, 128)	16512	conv2d_33[0][0]
conv2d_transpose_2 (Conv2DTrans)	(None, 64, 64, 128)	147584	add_1[0][0]
conv2d_32 (Conv2D)	(None, 64, 64, 128)	65664	conv2d_transpose_2[0][0] conv2d_32[0][0]
add_9 (Add)	(None, 64, 64, 128)	0	add_9[0][0]
activation_21 (Activation)	(None, 64, 64, 1)	0	activation_21[0][0]
conv2d_34 (Conv2D)	(None, 64, 64, 1)	129	conv2d_34[0][0]
activation_22 (Activation)	(None, 128, 128, 1)	0	activation_22[0][0]
up_sampling2d_4 (UpSampling2D)	(None, 128, 128, 128)	0	up_sampling2d_4[0][0]
lambda_3 (Lambda)	(None, 128, 128, 128)	0	lambda_3[0][0] add_1[0][0]
multiply_2 (Multiply)	(None, 128, 128, 128)	0	multiply_2[0][0]
conv2d_35 (Conv2D)	(None, 128, 128, 256)	16512	add_8[0][0]
up_sampling2d_5 (UpSampling2D)	(None, 128, 128, 128)	0	conv2d_35[0][0]
batch_normalization_26 (BatchNorm)	(None, 128, 128, 384)	512	up_sampling2d_5[0][0] batch_normalization_26[0][0]
concatenate_2 (Concatenate)	(None, 128, 128, 128)	0	concatenate_2[0][0]
conv2d_36 (Conv2D)	(None, 128, 128, 128)	1228928	conv2d_36[0][0]
batch_normalization_27 (BatchNorm)	(None, 128, 128, 128)	512	batch_normalization_27[0][0]

activation_23 (Activation)	(None, 128, 128, 128)	0	activation_23[0][0]
conv2d_37 (Conv2D)	(None, 128, 128, 128)	409728	concatenate_2[0][0]
conv2d_38 (Conv2D)	(None, 128, 128, 128)	49280	conv2d_37[0][0]
batch_normalization_28 (BatchNorm)	(None, 128, 128, 128)	512	conv2d_38[0][0]
batch_normalization_29 (BatchNorm)	(None, 128, 128, 128)	512	batch_normalization_28[0][0]
activation_24 (Activation)	(None, 128, 128, 128)	0	batch_normalization_29[0][0] activation_24[0][0]
add_10 (Add)	(None, 128, 128, 64)	0	add_10[0][0]
conv2d_39 (Conv2D)	(None, 128, 128, 64)	8256	conv2d_39[0][0]
batch_normalization_30 (BatchNorm)	(None, 128, 128, 64)	256	batch_normalization_30[0][0]
activation_25 (Activation)	(None, 128, 128, 64)	0	activation_25[0][0]
conv2d_41 (Conv2D)	(None, 128, 128, 64)	4160	conv2d_41[0][0]
conv2d_transpose_3 (Conv2DTrans)	(None, 128, 128, 64)	36928	add[0][0]
conv2d_40 (Conv2D)	(None, 128, 128, 64)	16448	conv2d_transpose_3[0][0] conv2d_40[0][0]
add_11 (Add)	(None, 128, 128, 64)	0	add_11[0][0]
activation_26 (Activation)	(None, 128, 128, 1)	0	activation_26[0][0]
conv2d_42 (Conv2D)	(None, 128, 128, 1)	65	conv2d_42[0][0]
activation_27 (Activation)	(None, 256, 256, 1)	0	activation_27[0][0]
up_sampling2d_6 (UpSampling2D)	(None, 256, 256, 64)	0	up_sampling2d_6[0][0]
lambda_4 (Lambda)	(None, 256, 256, 64)	0	lambda_4[0][0] add[0][0]
multiply_3 (Multiply)	(None, 256, 256, 64)	0	multiply_3[0][0]
conv2d_43 (Conv2D)	(None, 256, 256, 128)	4160	add_10[0][0]

up_sampling2d_7 (UpSampling2D)	(None, 256, 256, 64)	0	conv2d_43[0][0]
batch_normalization_31 (BatchNorm)	(None, 256, 256, 192)	256	up_sampling2d_7[0][0] batch_normalization_31[0][0]
concatenate_3 (Concatenate)	(None, 256, 256, 64)	0	concatenate_3[0][0]
conv2d_44 (Conv2D)	(None, 256, 256, 64)	307264	conv2d_44[0][0]
batch_normalization_32 (BatchNorm)	(None, 256, 256, 64)	256	batch_normalization_32[0][0]
activation_28 (Activation)	(None, 256, 256, 64)	0	activation_28[0][0]
conv2d_45 (Conv2D)	(None, 256, 256, 64)	102464	concatenate_3[0][0]
conv2d_46 (Conv2D)	(None, 256, 256, 64)	12352	conv2d_45[0][0]
batch_normalization_33 (BatchNorm)	(None, 256, 256, 64)	256	conv2d_46[0][0]
batch_normalization_34 (BatchNorm)	(None, 256, 256, 64)	256	batch_normalization_33[0][0]
activation_29 (Activation)	(None, 256, 256, 64)	0	batch_normalization_34[0][0] activation_29[0][0]
add_12 (Add)	(None, 256, 256, 2)	0	add_12[0][0]
conv2d_47 (Conv2D)	(None, 256, 256, 2)	130	conv2d_47[0][0]
batch_normalization_35 (BatchNorm)	(None, 256, 256, 2)	8	batch_normalization_35[0][0]
Total Parameters: 94,864,654 Trainable Parameters: 94,843,146 Non-trainable Parameters: 21,508			

Παράρτημα Β: Αποτελέσματα εκπαίδευσης ανά εποχή

Training Process				
Epochs	Loss	Accuracy	Validation Loss	Validation Accuracy
1	-0.5873	0.6671	-0.3439	0.3057
2	-0.6470	0.7245	-0.4766	0.4474
3	-0.6731	0.7290	-0.6711	0.6870
4	-0.6988	0.7504	-0.6983	0.7118
5	-0.7192	0.7609	-0.7291	0.7684
6	-0.7360	0.7742	-0.7192	0.7360
7	-0.7467	0.7786	-0.7054	0.7352
8	-0.7618	0.7918	-0.7486	0.7542
9	-0.7685	0.7943	-0.7410	0.7468
10	-0.7760	0.8002	-0.7390	0.7554
11	-0.7895	0.8120	-0.7808	0.7974
12	-0.7898	0.8063	-0.7752	0.7833
13	-0.7946	0.8081	-0.7231	0.7261
14	-0.8125	0.8278	-0.7633	0.7690
15	-0.8176	0.8321	-0.5929	0.5646
16	-0.8265	0.8393	-0.7149	0.7102
17	-0.8464	0.8609	-0.7722	0.7774
18	-0.8608	0.8777	-0.7522	0.7547
19	-0.8630	0.8806	-0.7637	0.7697
20	-0.8638	0.8813	-0.7532	0.7565
21	-0.8769	0.8969	-0.7631	0.7698
22	-0.8812	0.9017	-0.7672	0.7735
23	-0.8793	0.9003	-0.7652	0.7712
24	-0.8803	0.9025	-0.7616	0.7661
25	-0.8821	0.9031	-0.7599	0.7651
26	-0.8779	0.8990	-0.7573	0.7619
27	-0.8823	0.9037	-0.7567	0.7613
28	-0.8844	0.9057	-0.7556	0.7601
29	-0.8850	0.9069	-0.7555	0.7599
30	-0.8837	0.9043	-0.7546	0.7590
31	-0.8814	0.9029	-0.7564	0.7613
32	-0.8773	0.8991	-0.7593	0.7643

33	-0.8861	0.9072	-0.7585	0.7634
34	-0.8811	0.9021	-0.7575	0.7624
35	-0.8809	0.9024	-0.7587	0.7637
36	-0.8803	0.9031	-0.7569	0.7618
37	-0.8835	0.9050	-0.7594	0.7648
38	-0.8782	0.8993	-0.7593	0.7645
39	-0.8871	0.9088	-0.7573	0.7622
40	-0.8803	0.9014	-0.7571	0.7621
41	-0.8818	0.9027	-0.7584	0.7634

	Epoch in which val_loss improved		Early_stopping application

Παράρτημα Γ: Αποτελέσματα αξιολόγησης προβλέψεων

	PREDICTED MASKS SCORES for TEST IMAGE			
	Accuracy	Precision	Recall	F1-score
1	0.9635	0.9960	0.9673	0.9814
2	0.9343	0.9686	0.9635	0.9660
3	0.7094	0.6571	0.8896	0.7559
4	0.8333	0.9010	0.7027	0.7896
5	0.7955	0.9095	0.8553	0.8816
6	0.2891	0.0000	0.0000	0.0000
7	0.8490	0.8774	0.8775	0.8774
8	0.8150	0.8372	0.8401	0.8386
9	0.9093	0.9503	0.9451	0.9477
10	0.7392	0.7510	0.9611	0.8432
11	0.5829	0.4863	0.8068	0.6068
12	0.6077	0.7778	0.4677	0.5842
13	0.8923	0.7976	0.9087	0.8496
14	0.7806	0.8044	0.9439	0.8686
15	0.5732	0.9980	0.5424	0.7029
16	0.5840	0.9917	0.5558	0.7123
17	0.9356	0.9678	0.9579	0.9628
18	0.9122	0.9957	0.9100	0.9509
19	0.7937	0.9650	0.5905	0.7327
20	0.7983	0.8777	0.6867	0.7706
21	0.6831	0.7042	0.8004	0.7492
22	0.7727	0.7976	0.9383	0.8622
23	0.6151	0.7614	0.7551	0.7583
24	0.7859	0.8853	0.8734	0.8793
25	0.9266	0.9428	0.9736	0.9580
26	0.6243	0.5599	0.9929	0.7161
27	0.7995	0.8370	0.9470	0.8886
28	0.7621	0.9806	0.7204	0.8306
29	0.6650	0.9245	0.3583	0.5164
30	0.8139	0.6489	0.3663	0.4683
31	0.8517	0.0000	0.0000	0.0000
32	0.8860	0.4368	0.7502	0.5521
33	0.8313	0.9488	0.8494	0.8964

34	0.6635	0.5867	0.9977	0.7389
35	0.7128	0.8213	0.8433	0.8322
36	0.7553	0.9862	0.7263	0.8365
37	0.6760	0.9694	0.3190	0.4801
38	0.8307	0.6796	0.4625	0.5504
39	0.9038	0.3810	0.2874	0.3277
40	0.8856	0.4525	0.8470	0.5899