ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ – ΜΗΧΑΝΙΚΩΝ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ

Αναγνώριση ρωγμών με χρήση τεχνολογιών μη επιβλεπόμενης βαθιάς μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΚΟΤΡΩΝΑΚΗ ΙΩΑΝΝΑΣ

Επιβλέπων : Δουλάμης Νικόλαος

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2021

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Νικόλαο Δουλάμη, καθώς και τον υποψήφιο διδάκτορα κ. Νικόλαο Μπάκαλο, για την καθοδήγηση, τον χρόνο, τις πολύτιμες συμβουλές και γνώσεις που μου προσέφεραν σε όλη τη διάρκεια εκπόνησης της διπλωματικής εργασίας.

Πίνακας περιεχομένων

Περίληψη

Σκοπός της εργασίας είναι η αναγνώριση ρωγμών μέσω τεχνολογιών μη επιβλεπόμενης βαθιάς μηχανικής μάθησης. Η μηχανική μάθηση, αν και δεν αποτελεί νέο πεδίο μελέτης, είναι ένας συνεχώς εξελισσόμενος κλάδος της τεχνητής νοημοσύνης, με ποικίλες εφαρμογές όπως η αναγνώριση προφορικού λόγου, γραπτού λόγου και προσώπων αλλά και η πραγματοποίηση προβλέψεων. Μία από τις υποκατηγορίες της είναι και η μη επιβλεπόμενη μάθηση, στην οποία τα δεδομένα δεν είναι επισημασμένα, δηλαδή δεν υπάρχουν αληθείς ετικέτες, ενώ χρησιμοποιείται κυρίως σε εφαρμογές ομαδοποίησης, συσχέτισης και μείωσης διαστάσεων. Ο αυτόματος κωδικοποιητής, ο οποίος αποτελεί το κύριο μέσο αναγνώρισης ρωγμών στα πλαίσια της παρούσας εργασίας, είναι ένα μη επιβλεπόμενο νευρωνικό δίκτυο το οποίο επιτυγχάνει μέσω των επιπέδων του να αναπαράξει την είσοδό του στην έξοδό του, συμπιέζοντας τα δεδομένα στην διαδικασία κωδικοποίησης και αποσυμπιέζοντας τα στη διαδικασία αποκωδικοποίησης. Εφόσον το δίκτυο περιλαμβάνει πολλαπλά κρυμμένα επίπεδα, αποτελεί δίκτυο βαθιάς μάθησης.

Με σκοπό να επιτευχθεί αναγνώριση ρωγμών, χρησιμοποιήθηκαν εικόνες από τα τοιχώματα μίας σήραγγας, των οποίων η λήψη πραγματοποιήθηκε υπό διαφορετικό φωτισμό. Οι εικόνες αυτές κατακερματίστηκαν σε ομοιόμορφου μεγέθους τμήματα 45 × 45 εικονοστοιχείων. Για την εκπαίδευση του αυτόματου κωδικοποιητή αξιοποιήθηκαν αποκλειστικά τμήματα που δεν απεικόνιζαν ρωγμές, δηλαδή φυσιολογικές εικόνες. Τα τμήματα που απεικόνιζαν ρωγμές αποτελούν τις μη φυσιολογικές εικόνες και αξιοποιήθηκαν στη φάση του ελέγχου, σε συνδυασμό με ένα μέρος των φυσιολογικών εικόνων.

Η ταξινόμηση των εικόνων σε φυσιολογικές και μη φυσιολογικές πραγματοποιήθηκε μέσω υπολογισμού του μέσου τετραγωνικού σφάλματος της κάθε εικόνας με την αντίστοιχη παραγόμενη από το μοντέλο και προσδιορισμό ενός κατωφλίου. Οι εικόνες για τις οποίες το υπολογισμένο μέσο τετραγωνικό σφάλμα είναι μικρότερο από το κατώφλι θεωρούνται φυσιολογικές. Αντίθετα, οι εικόνες για τις οποίες το μέσο τετραγωνικό σφάλμα ξεπερνά το κατώφλι θεωρούνται μη φυσιολογικές και υποθέτεται ότι περιλαμβάνουν ρωγμή ή άλλο ανώμαλο συμβάν.

Εκτός των κανονικών δεδομένων εκπαίδευσης και ελέγχου, δημιουργήθηκαν σύνολα δεδομένων με εξομάλυνση και με θόρυβο τύπου αλάτι – πιπέρι, ακολουθώντας και για αυτά όμοια διαδικασία. Παράλληλα με τον αυτόματο κωδικοποιητή, επιχειρήθηκε να διαχωριστούν τα δεδομένα σε φυσιολογικές και μη φυσιολογικές εικόνες μέσω του αλγορίθμου K-means, του Gaussian Mixture Model και ενός επιβλεπόμενου νευρωνικού δικτύου, ώστε να υπάρχει δυνατότητα σύγκρισης.

Στα κανονικά δεδομένα ο αυτόματος κωδικοποιητής φάνηκε να υπερτερεί των υπόλοιπων αλγορίθμων πετυχαίνοντας συνολική ακρίβεια κοντά στο 94%, με το επιβλεπόμενο νευρωνικό δίκτυο να ακολουθεί. Στην περίπτωση των δεδομένων με εξομάλυνση όμως το επιβλεπόμενο νευρωνικό δίκτυο σημείωσε συνολική ακρίβεια της τάξης του 96%, ξεπερνώντας τον αυτόματο κωδικοποιητή, ο οποίος και πάλι κατάφερε να διαχωρίσει επιτυχημένα τα δεδομένα με ακρίβεια κοντά στο 91%. Η προσθήκη θορύβου επηρέασε αισθητά τους αλγορίθμους. Παρ' όλα αυτά ο αυτόματος κωδικοποιητής σημείωσε συνολική ακρίβεια κοντά στο 80%, ξεπερνώντας σημαντικά την απόδοση των υπόλοιπων αλγορίθμων.

Τελικά, συμπεραίνεται ότι ο αυτόματος κωδικοποιητής κατάφερε με επιτυχία να διαχωρίσει τις εικόνες και να εντοπίσει αυτές που περιλαμβάνουν ρωγμή, σημειώνοντας καλύτερη απόδοση και μεγαλύτερη σταθερότητα σε σύγκριση με τις υπόλοιπες μεθόδους με τις οποίες συγκρίθηκε.

Λέξεις Κλειδιά: "Μηχανική Μάθηση, Μη Επιβλεπόμενη Μάθηση, Αναγνώριση Ρωγμών, Αυτόματος Κωδικοποιητής"

Abstract

The purpose of this work is to identify cracks using unsupervised deep machine learning technologies. Machine learning, although not a new field of study, is a continuously evolving branch of artificial intelligence, with diverse applications such as voice and written text recognition, face recognition and making predictions. One of its subcategories is unsupervised learning, in which the data is not labelled, i.e. there are no true labels, and is mainly used in clustering, association and dimensionality reduction applications. The autoencoder, which is the main method of crack detection in the context of this work, is an unsupervised neural network which achieves through its layers to reproduce its input to its output by compressing the data in the encoding process and decompressing it in the decoding process. Since the network includes multiple hidden layers, it is a deep learning network.

In order to achieve crack detection, images of the walls of a tunnel, taken under different lighting conditions, were used. These images were segmented into uniformly sized segments of 45×45 pixels. For the training of the autoencoder, only segments that did not depict cracks, i.e. normal images, were used. The segments that depicted cracks constitute the abnormal images and were utilized in the test phase, in combination with a portion of the normal images.

The classification of images into normal and abnormal was performed by calculating the mean square error of each image with the corresponding generated by the model and determining a threshold. Images for which the calculated mean square error is less than the threshold are considered normal. Conversely, images for which the mean square error exceeds the threshold are considered abnormal and are assumed to depict a crack or other anomalous events.

In addition to the standard training and test data, smoothing and salt-and-pepper noise datasets were created and a similar procedure was applied to them. Along with the autoencoder, an attempt was made to separate the data into normal and abnormal images using the K-means algorithm, the Gaussian Mixture Model and a supervised neural network to enable comparison.

On standard data the autoencoder seemed to outperform the other algorithms, achieving an overall accuracy close to 94%, with the supervised neural network following. In the case of the smoothed data, however, the supervised neural network achieved an overall accuracy of 96%, outperforming the autoencoder, which again managed to successfully separate the data with an accuracy approaching 91%. The addition of noise significantly affected the algorithms. Nevertheless, the autoencoder achieved an overall accuracy close to 80%, significantly outperforming the other algorithms.

Finally, it is concluded that the autoencoder was able to successfully classify the images and identify those containing a crack, achieving better performance and greater stability compared to the other methods against which it was compared.

Keywords: “Machine Learning, Unsupervised Learning, Crack Detection, Autoencoder”

1 *Εισαγωγή*

Το πρόβλημα της αναγνώρισης ρωγμών σε δομικά στοιχεία και κατασκευές είναι ιδιαίτερα διαδεδομένο αφού η ύπαρξη ρωγμών αποτελεί καίριο ελάττωμα της κατασκευής και εμπεριέχει κινδύνους. Ιδιαίτερης σημασίας κρίνεται ο έγκαιρος εντοπισμός τους, αφού αν αγνοηθούν, η κατασκευή μπορεί να οδηγηθεί σε κατάρρευση. Έτσι, για να εξασφαλιστεί ότι η ρωγμή θα εντοπιστεί προτού υπάρξουν σημαντικές συνέπειες, απαιτείται παρακολούθηση και συχνός έλεγχος της κατασκευής. Ο έλεγχος αυτός πραγματοποιείται μέσω ανθρώπινης παρατήρησης από εξειδικευμένους παρατηρητές και σαν διαδικασία είναι ιδιαίτερα επίπονη και χρονοβόρα. Επιπλέον, ο κίνδυνος ανθρώπινου λάθους δεν πρέπει να αγνοηθεί, όπως και το γεγονός ότι ο έλεγχος δυσπρόσιτων σημείων δεν είναι πάντα εφικτός. Συνεπώς, δημιουργείται ανάγκη για αυτοματοποίηση της διαδικασίας.

Η μηχανική μάθηση είναι ένας συνεχώς εξελισσόμενος κλάδος της τεχνητής νοημοσύνης, ο οποίος γνώρισε άνθιση τα τελευταία χρόνια λόγω της εύκολης πρόσβασης σε μεγάλη υπολογιστική ισχύ και σε πληθώρα δεδομένων. Μέσω τεχνολογιών βαθιάς μηχανικής μάθησης είναι δυνατό να επιτευχθεί ακρίβεια που πλησιάζει τις ανθρώπινες ικανότητες ή μπορεί ακόμα και να τις ξεπεράσει. Με

αυτό τον τρόπο είναι εμφανές ότι η μηχανική μάθηση είναι δυνατό να εφαρμοστεί στο υπαρκτό πρόβλημα της αυτοματοποιημένης αναγνώρισης ρωγμών μέσω της χρήσης εικόνων και συγκεκριμένα, στα πλαίσια της παρούσας εργασίας, μέσω ασπρόμαυρων φωτογραφιών.

Η μεθοδολογία που επιλέχθηκε να εφαρμοστεί είναι η χρήση ενός αυτόματου κωδικοποιητή. Οι αυτόματοι κωδικοποιητές, όπως αναφέρεται και στη συνέχεια αναλυτικότερα, είναι μη επιβλεπόμενα νευρωνικά δίκτυα τα οποία χρησιμοποιούνται ευρέως για μείωση διαστάσεων και για εντοπισμό ακραίων τιμών, οπότε η χρήση ενός αυτόματου κωδικοποιητή για εντοπισμό ρωγμών, πράγμα που πραγματοποιείται στα πλαίσια της παρούσας εργασίας, παρουσιάζει ιδιαίτερο ενδιαφέρον.

Στα κεφάλαια που ακολουθούν αρχικά πραγματοποιείται βιβλιογραφική ανασκόπηση (Κεφάλαιο 2) στην οποία ορίζεται η έννοια της μηχανικής μάθησης, καθώς και οι τομείς που την απαρτίζουν. Στη συνέχεια του κεφαλαίου δίνεται περισσότερη έμφαση στον αυτόματο κωδικοποιητή και τις εφαρμογές του, αφού αποτελεί και το κύριο μέρος της παρούσας εργασίας.

Στο 3ο κεφάλαιο παρουσιάζεται η μεθοδολογία που ακολουθήθηκε για την αναγνώριση ρωγμών, δηλαδή αιτιολογείται η επιλογή του αυτόματου κωδικοποιητή για τη συγκεκριμένη εφαρμογή, παρουσιάζονται συνοπτικά τα βήματα που ακολουθήθηκαν και στη συνέχεια αναλύεται η μορφή του μοντέλου και τα επίπεδα που περιλαμβάνει.

Το 4ο κεφάλαιο αφορά την υλοποίηση της μεθοδολογίας σε γλώσσα προγραμματισμού Python και περιλαμβάνεται ο κώδικας που συντάχθηκε. Συγκεκριμένα αναφέρεται η διαδικασία προετοιμασίας των δεδομένων εκπαίδευσης και ελέγχου για την μετέπειτα χρήση τους, ο ορισμός και η εκπαίδευση του μοντέλου και στη συνέχεια η διαδικασία ελέγχου της απόδοσής του με τα διαθέσιμα δεδομένα. Τα δεδομένα εκπαίδευσης και ελέγχου που χρησιμοποιήθηκαν στη φάση αυτή στη συνέχεια αναφέρονται ως κανονικά δεδομένα.

Με σκοπό να γίνει πλήρης αξιολόγηση του αυτόματου κωδικοποιητή, στο 5ο κεφάλαιο δημιουργούνται δύο ακόμα σύνολα δεδομένων: τα δεδομένα με εξομάλυνση και τα δεδομένα με θόρυβο, τα οποία δημιουργήθηκαν εξομαλύνοντας και προσθέτοντας θόρυβο αντίστοιχα στα κανονικά δεδομένα. Έπειτα, τα νέα σύνολα δεδομένων χρησιμοποιούνται για την εκπαίδευση και τον έλεγχο του αυτόματου κωδικοποιητή και παρουσιάζονται τα αντίστοιχα αποτελέσματα. Παράλληλα, στα τρία σύνολα δεδομένων εφαρμόζονται οι αλγόριθμοι K-means, Mixture of Gaussians και ένα επιβλεπόμενο νευρωνικό δίκτυο και συγκρίνονται τα αποτελέσματα.

Τα παραπάνω λαμβάνονται υπόψιν στο 6ο και τελευταίο κεφάλαιο της παρούσας εργασίας, στο οποίο παρουσιάζονται συνολικά συμπεράσματα.

2 *Βιβλιογραφική ανασκόπηση*

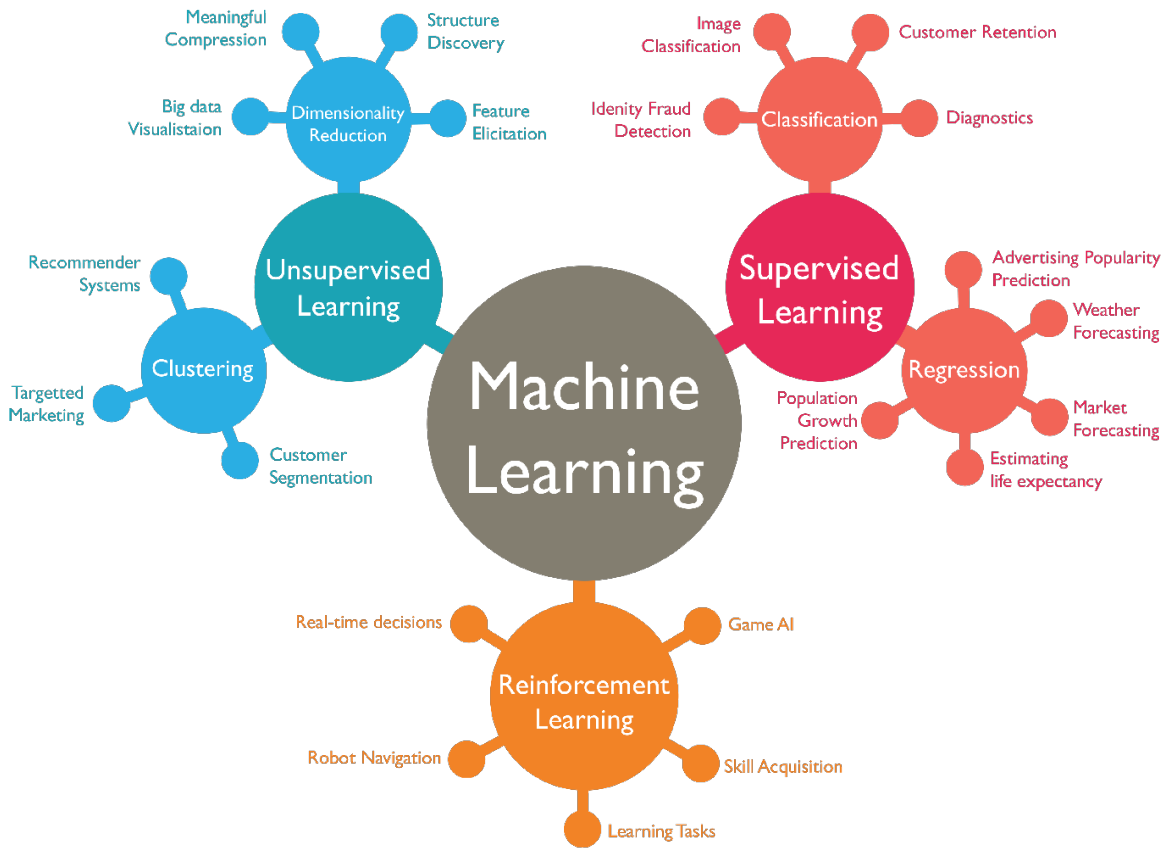
Το κεφάλαιο αυτό αφορά την ανασκόπηση και την αναφορά σε έννοιες που συνδέονται άμεσα με το αντικείμενο της εργασίας, όπως η μηχανική μάθηση και συγκεκριμένα η επιβλεπόμενη μάθηση, η μη επιβλεπόμενη μάθηση, η βαθιά μάθηση, η ενισχυτική μάθηση και τέλος οι αυτόματοι κωδικοποιητές και οι εφαρμογές τους.

2.1 Μηχανική μάθηση - Machine Learning

Η μηχανική μάθηση, στην τεχνητή νοημοσύνη, είναι ένας επιστημονικός κλάδος που ασχολείται με την υλοποίηση λογισμικού που μπορεί να μαθαίνει αυτόνομα. Μία από τις συνηθέστερες εφαρμογές της είναι η χρήση τεχνητών νευρωνικών δικτύων [27].

Συγκεκριμένα, μηχανική μάθηση είναι ο προγραμματισμός υπολογιστών με σκοπό τη βελτιστοποίηση ενός κριτηρίου απόδοσης χρησιμοποιώντας δεδομένα παραδείγματος ή προηγούμενη εμπειρία. Μία περίπτωση στην οποία η μηχανική μάθηση είναι απαραίτητη είναι στην επίλυση

προβλημάτων για τα οποία δεν υπάρχει ανθρώπινη εμπειρία ή οι άνθρωποι δεν μπορούν να εξηγήσουν την εμπειρία τους, όπως στο παράδειγμα της αναγνώρισης προφορικού λόγου. Επιπλέον, η χρήση μηχανικής μάθησης ενδείκνυται και στην περίπτωση που το πρόβλημα μεταβάλλεται με τον χρόνο ή εξαρτάται από το περιβάλλον του [1].



Εικόνα 1: Διακρίσεις μηχανικής μάθησης

Πηγή: *https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications*

Υπάρχουν ήδη πολλές επιτυχημένες εφαρμογές της μηχανικής μάθησης σε διάφορους τομείς όπως στην αναγνώριση ομιλίας, χειρογράφων και προσώπων, στην ανάλυση πωλήσεων και στη ρομποτική [1].

2.2 Επιβλεπόμενη μάθηση - Supervised learning

α) Ορισμός

Όπως φαίνεται και στην Εικόνα 1, η πρώτη υποκατηγορία της μηχανικής μάθησης είναι η επιβλεπόμενη μάθηση (supervised learning). Η επιβλεπόμενη μάθηση χρησιμοποιεί ένα σύνολο δεδομένων (training set) για να εκπαιδεύσει μοντέλα να αποδίδουν το επιθυμητό αποτέλεσμα. Το σύνολο δεδομένων εκπαίδευσης αποτελείται από δεδομένα εισόδου (inputs) και δεδομένα εξόδου (outputs), επιτρέποντας έτσι στο μοντέλο να εκπαιδευτεί με την πάροδο του χρόνου. Η ακρίβεια του αλγορίθμου μετριέται μέσω μιας συνάρτησης απωλειών και προσαρμόζεται μέχρι να επιτευχθεί η επιθυμητή ελαχιστοποίηση του σφάλματος. [28]

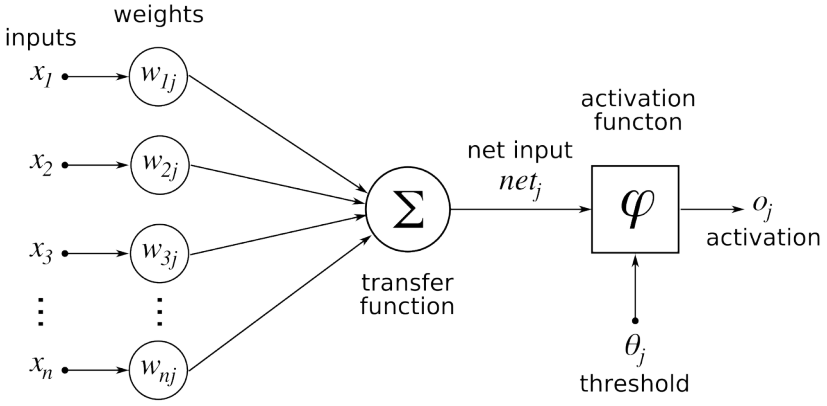
Ως data mining ορίζεται η εφαρμογή τεχνικών μηχανικής μάθησης σε μεγάλες βάσεις δεδομένων [1]. Έτσι, όταν εφαρμόζεται data mining, η επιβλεπόμενη μάθηση μπορεί να διαχωριστεί σε δύο τύπους προβλημάτων:

- Προβλήματα ταξινόμησης (classification). Στα προβλήματα αυτά χρησιμοποιείται ένας αλγόριθμος για να καταχωρηθούν δεδομένα ελέγχου (test data) σε συγκεκριμένες κατηγορίες. Ο αλγόριθμος αναγνωρίζει συγκεκριμένες οντότητες στο σύνολο δεδομένων και επιχειρεί να εξάγει κάποια συμπεράσματα σχετικά με τον τρόπο με τον οποίο οι συγκεκριμένες οντότητες πρέπει να επισημανθούν ή να οριστούν. Κάποιοι ευρέως χρησιμοποιούμενοι αλγόριθμοι ταξινόμησης είναι οι γραμμικοί ταξινομητές (linear classifiers), οι μηχανές διανυσμάτων υποστήριξης (support vector machines, SVM), τα δέντρα αποφάσεων (decision trees), ο k-κοντινότερος γείτονας (k-nearest neighbor) και το τυχαίο δάσος (random forest). [28]
- Προβλήματα παλινδρόμησης (regression). Η παλινδρόμηση χρησιμοποιείται για να κατανοηθούν οι σχέσεις μεταξύ εξαρτημένων και ανεξάρτητων μεταβλητών. Εφαρμόζεται συνήθως για την πραγματοποίηση προβλέψεων, όπως οι προβλέψεις εσόδων για μια επιχείρηση. Κάποιοι δημοφιλείς αλγόριθμοι παλινδρόμησης είναι η γραμμική παλινδρόμηση (linear regression), η λογιστική παλινδρόμηση (logistic regression) και η πολυωνυμική παλινδρόμηση (polynomial regression). [28]

β) Συχνά χρησιμοποιούμενοι αλγόριθμοι

Κάποιοι από τους πιο συχνά χρησιμοποιούμενους αλγορίθμους που υπάγονται στην επιβλεπόμενη μάθηση αναφέρθηκαν ονομαστικά παραπάνω. Ακολουθεί μια σύντομη επεξήγηση κάποιων ιδιαίτερα δημοφιλών αλγορίθμων που συνήθως εφαρμόζονται σε προγράμματα με χρήση γλωσσών προγραμματισμού όπως η Python και η R.

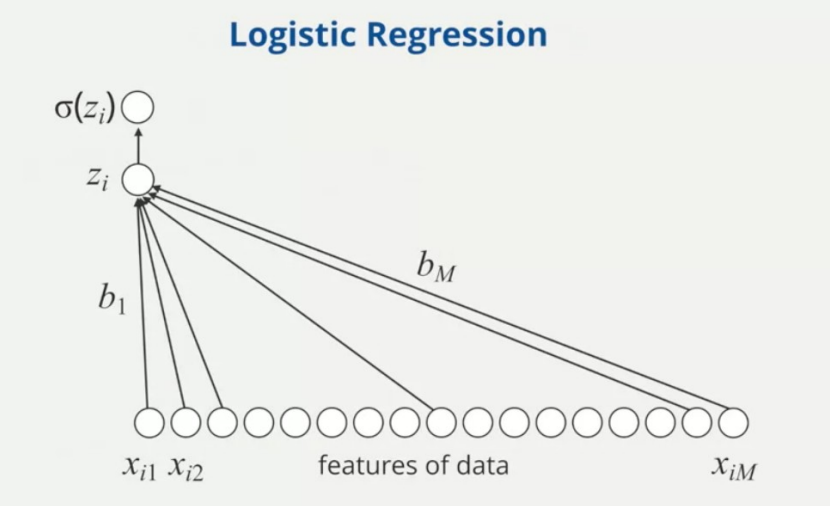
- Νευρωνικά δίκτυα (neural networks): τα νευρωνικά δίκτυα επεξεργάζονται τα δεδομένα εκπαίδευσης με τρόπο που μιμείται τη λειτουργία του ανθρώπινου εγκεφάλου, δηλαδή μέσω τεχνητών νευρώνων. Ένα από τα απλούστερα νευρωνικά δίκτυα είναι το multilayer perceptron το οποίο αποτελείται από ένα στρώμα εισόδου (input layer), ένα στρώμα εξόδου (output layer) και μερικά κρυμμένα στρώματα (hidden layers). Κάθε στρώμα έχει πολλαπλούς νευρώνες και τα γειτονικά στρώματα είναι πλήρως συνδεδεμένα. Κάθε νευρώνας υπολογίζει ένα σταθμισμένο άθροισμα όλων των εισόδων που δέχεται, πολλαπλασιάζοντας τις με τα αντίστοιχα βάρη και προσθέτοντας έναν αριθμό bias. Στη συνέχεια το σταθμισμένο άθροισμα περνάει από μια συνάρτηση ενεργοποίησης (activation function), η οποία καθορίζει τα σήματα εξόδου για το επόμενο επίπεδο. [3]



Εικόνα 2: Νευρωνικό δίκτυο

Πηγή:
https://upload.wikimedia.org/wikipedia/commons/6/60/ArtificialNeuronModel_english.png

- Μέθοδος Naive Bayes: η μέθοδος Naive Bayes είναι προσέγγιση ταξινόμησης η οποία υιοθετεί την αρχή της ανεξαρτησίας των κλάσεων υπό συνθήκες σύμφωνα με το θεώρημα του Bayes. Δηλαδή η παρουσία ενός χαρακτηριστικού δεν επηρεάζει την παρουσία του άλλου στην πιθανότητα ενός συγκεκριμένου αποτελέσματος. [28] Αν και η υπόθεση ανεξαρτησίας συχνά στην πράξη παραβιάζεται, η μέθοδος Naive Bayes προσφέρει ικανοποιητική ακρίβεια ταξινόμησης σε ορισμένες περιπτώσεις. [4]
- Γραμμική παλινδρόμηση (linear regression): η γραμμική παλινδρόμηση συνήθως χρησιμοποιείται για την πραγματοποίηση προβλέψεων σε σχέση με μελλοντικά αποτελέσματα μέσω του προσδιορισμού της σχέσης μεταξύ μιας εξαρτημένης και μιας ή περισσότερων ανεξάρτητων μεταβλητών. Αυτό επιτυγχάνεται επιδιώκοντας τη χάραξη μιας ευθείας γραμμής που προσαρμόζεται καλύτερα στα δεδομένα, μέσω της μεθόδου των ελαχίστων τετραγώνων. Ειδική περίπτωσή της είναι η απλή γραμμική παλινδρόμηση, η οποία αφορά την περίπτωση που υπάρχει μια ανεξάρτητη μεταβλητή και μια εξαρτημένη. Αν αυξηθεί ο αριθμός των ανεξάρτητων μεταβλητών, τότε είναι γνωστή ως πολλαπλή παλινδρόμηση. [28]
- Λογιστική παλινδρόμηση (logistic regression): σε αντίθεση με τη γραμμική παλινδρόμηση, η οποία χρησιμοποιείται στην περίπτωση που η εξαρτημένη μεταβλητή είναι συνεχής, η λογιστική παλινδρόμηση επιλέγεται όταν η εξαρτημένη μεταβλητή έχει δυαδικές εξόδους, όπως “αληθής” και “ψευδής” ή “ναι” και “όχι”. Έτσι η λογιστική παλινδρόμηση χρησιμοποιείται για την επίλυση προβλημάτων δυαδικής ταξινόμησης, όπως η αναγνώριση ανεπιθύμητων μηνυμάτων (spam). [28]



Εικόνα 3: Λογιστική παλινδρόμηση

Πηγή: *Lawrence Carin, “Introduction to Machine Learning”, Duke University, Coursera*

- Μηχανές διανυσμάτων υποστήριξης (support vector machines, SVM): η SVM αναπτύχθηκε από τον Vladimir Vapnik και χρησιμοποιείται για την ταξινόμηση αλλά και για την παλινδρόμηση δεδομένων. Σε προβλήματα ταξινόμησης συνήθως εφαρμόζεται ως ορίζοντας ένα υπερεπίπεδο, το οποίο είναι γνωστό ως όριο απόφασης, εκεί όπου η απόσταση μεταξύ των δύο κλάσεων είναι μέγιστη. Έτσι, διαχωρίζονται οι κλάσεις των δεδομένων σημείων εκατέρωθεν του υπερεπιπέδου [28]. Επιπλέον, η SVM μπορεί να μάθει να αναγνωρίζει χειρόγραφα αριθμητικά ψηφία εξετάζοντας σαρωμένες εικόνες, ενώ έχει εφαρμοστεί με επιτυχία σε ποικίλες εφαρμογές βιολογικού περιεχομένου. [5]

- Κ-κοντινότερος γείτονας (k-nearest neighbor, KNN): ο αλγόριθμος αυτός είναι μη παραμετρικός και ταξινομεί τα σημεία δεδομένων με βάση την εγγύτητα και τη συσχέτιση τους, θεωρώντας ότι σημεία που βρίσκονται κοντά είναι πιθανό να είναι παρόμοια. Έτσι, ταξινομεί τα δεδομένα υπολογίζοντας απόσταση, συνήθως ευκλείδεια, μεταξύ τους. Προτιμάται συχνά λόγω της ευκολίας χρήσης του και του χαμηλού χρόνου υπολογισμού, ο οποίος όμως αυξάνεται όσο αυξάνεται και η ποσότητα των δεδομένων. Ο KNN χρησιμοποιείται συχνά για μηχανές προτάσεων (recommendation engines) και αναγνώριση εικόνων (image recognition). [28]
- Τυχαίο δάσος (random forest): το τυχαίο δάσος είναι ένας απλός και ευέλικτος αλγόριθμος που μπορεί να χρησιμοποιηθεί για προβλήματα ταξινόμησης αλλά και παλινδρόμησης. Το "δάσος" αποτελείται από ένα σύνολο δέντρων αποφάσεων και η εκμάθηση του συνήθως πραγματοποιείται μέσω της μεθόδου "bagging". Σύμφωνα με τη μέθοδο αυτή, ο συνδυασμός μοντέλων εκμάθησης αυξάνει την ακρίβεια και τη σταθερότητα του τελικού αποτελέσματος. [29]

γ) Προκλήσεις επιβλεπόμενης μάθησης

Αν και η επιβλεπόμενη μάθηση προσφέρει κάποια πλεονεκτήματα, υπάρχουν ορισμένες προκλήσεις κατά τη δημιουργία τέτοιων μοντέλων όπως:

- Η δόμηση τέτοιων μοντέλων απαιτεί ένα συγκεκριμένο επίπεδο τεχνογνωσίας και εξειδίκευσης.
- Η εκπαίδευση επιβλεπόμενων μοντέλων μπορεί να είναι πολύ χρονοβόρα.
- Δεδομένου ότι απαιτούνται αληθείς ετικέτες στα δεδομένα εκπαίδευσης, υπάρχει μεγαλύτερη πιθανότητα ανθρώπινου σφάλματος.
- Σε αντίθεση με τα μη επιβλεπόμενα μοντέλα, η επιβλεπόμενη μάθηση δεν είναι δυνατό να ομαδοποιήσει ή να ταξινομήσει τα δεδομένα από μόνη της.

2.3 Μη επιβλεπόμενη μάθηση - Unsupervised learning

α) Ορισμός

Η μη επιβλεπόμενη μηχανική μάθηση (unsupervised learning) χρησιμοποιεί αλγορίθμους μηχανικής μάθησης, σε συνδυασμό με μη επισημασμένα δεδομένα (unlabeled data), επιτυγχάνοντας την ανάλυση και την ομαδοποίησή τους. Δηλαδή, οι αλγόριθμοι είναι σε θέση να ανακαλύψουν κρυμμένα μοτίβα ή ομαδοποιήσεις δεδομένων, χωρίς να υπάρχει η ανάγκη ανθρώπινης παρέμβασης, και μάλιστα πολύ γρηγορότερα σε σύγκριση με την ανθρώπινη παρατήρηση. Λόγω της ικανότητάς της να ανακαλύπτει ομοιότητες και διαφορές στα δεδομένα, η μη επιβλεπόμενη μάθηση είναι ιδανική λύση για πολλές εφαρμογές, όπως η διερευνητική ανάλυση δεδομένων, οι στρατηγικές διασταυρούμενων πωλήσεων, η κατηγοριοποίηση πελατών και η αναγνώριση εικόνων. [30]

β) Συνήθεις προσεγγίσεις μη επιβλεπόμενης μάθησης

Τα μοντέλα μη επιβλεπόμενης μάθησης χρησιμοποιούνται κυρίως για τρία ήδη εφαρμογών: για εφαρμογές ομαδοποίησης (clustering), εφαρμογές συσχέτισης (association) και εφαρμογές μείωσης διαστάσεων (dimesionality reduction).

1. Ομαδοποίηση (clustering)

Η ομαδοποίηση (clustering) είναι μια τεχνική data mining, η οποία ομαδοποιεί δεδομένα με βάση τις ομοιότητες και τις διαφορές τους. Οι αλγόριθμοι ομαδοποίησης αξιοποιούνται για την επεξεργασία πρωτογενών (raw) και μη ταξινομημένων (unclassified) δεδομένων και την ομαδοποίηση τους σε ομάδες που αντιπροσωπεύονται από συγκεκριμένες δομές και μοτίβα. Διακρίνονται τα παρακάτω είδη αλγορίθμων ομαδοποίησης: οι αποκλειστικοί (exclusive), οι επικαλυπτόμενοι (overlapping), οι ιεραρχικοί (hierarchical) και οι πιθανολογικοί (probabilistic). [30]

- Αποκλειστική ομαδοποίηση (exclusive clustering)

Σύμφωνα με την αποκλειστική ή "σκληρή" ομαδοποίηση, ένα σημείο δεδομένων είναι δυνατό να ανήκει μόνο σε μία ομάδα. Ένα παράδειγμα αποκλειστικής ομαδοποίησης είναι ο αλγόριθμος K-means, ο οποίος κατανέμει τα δεδομένα σε Κ ομάδες, όπου Κ ο αριθμός των ομάδων με βάση την απόσταση από το κεντροειδές κάθε ομάδας. Όσο μεγαλύτερος είναι ο αριθμός Κ, τόσο μικρότερες θα είναι οι ομάδες και τόσο μεγαλύτερη η λεπτομέρεια της ομαδοποίησης. Αντίθετα, για μικρή τιμή του Κ προκύπτουν μεγάλες ομάδες, άρα και μικρότερη λεπτομέρεια. Ο αλγόριθμος K-means χρησιμοποιείται ευρέως σε εφαρμογές τμηματοποίησης αγοράς (market segmentation), στην ομαδοποίηση εγγράφων (data clustering), στην τμηματοποίηση εικόνων (image segmentation) και στη συμπίεση εικόνων (image compression). [30]

- Επικαλυπτόμενη ομαδοποίηση (overlapping clustering)

Σε αρκετά σύνολα δεδομένων είναι απαραίτητο οι ομάδες να παρουσιάζουν επικάλυψη, όπως σε πολλές εφαρμογές βιολογικών συνόλων δεδομένων [6]. Η επικαλυπτόμενη ομαδοποίηση διαφέρει από την αποκλειστική στο ότι επιτρέπει στα σημεία δεδομένων να ανήκουν σε περισσότερες από μία ομάδες με διαφορετικούς βαθμούς συμμετοχής [30]. Μερικά παραδείγματα στα οποία η χρήση επικαλυπτόμενης ομαδοποίησης είναι προτιμώμενη είναι η περίπτωση των γονιδίων, τα οποία μπορεί να συμμετέχουν σε πολλές διαδικασίες του οργανισμού, και η περίπτωση των ταινιών, που μπορεί να ανήκουν σε πολλά διαφορετικά είδη. [6]

- Ιεραρχική ομαδοποίηση (hierarchical clustering)

Η ιεραρχική ομαδοποίηση ή ιεραρχική ανάλυση συστάδων (hierarchical cluster analysis, HCA) είναι ένας αλγόριθμος μη επιβλεπόμενης μάθησης, ο οποίος μπορεί να κατηγοριοποιηθεί με δύο τρόπους: μπορεί να είναι συσσωρευτικός (agglomerative) ή διαχωριστικός (divisive). [30]

Η συσσωρευτική ομαδοποίηση (agglomerative clustering) είναι μία "από κάτω προς τα πάνω" προσέγγιση ("bottoms-up"), αφού αρχικά τα σημεία δεδομένων θεωρούνται ξεχωριστές ομάδες και στη συνέχεια συγχωνεύονται επαναληπτικά με βάση την ομοιότητα. Οι μέθοδοι που χρησιμοποιούνται για τη μέτρηση της ομοιότητας είναι:

Η σύνδεση Ward (Ward's linkage): σύμφωνα με τη μέθοδο αυτή, η απόσταση μεταξύ δύο συστάδων ορίζεται σαν την αύξηση του αθροίσματος του τετραγώνου μετά τη συγχώνευση των συστάδων.

Η μέση σύνδεση (average linkage): σύμφωνα με τη μέθοδο αυτή, η ομοιότητα ορίζεται από τη μέση απόσταση δύο σημείων σε κάθε ομάδα.

Η πλήρης ή μέγιστη σύνδεση (complete or maximum linkage): σύμφωνα με τη μέθοδο αυτή, η ομοιότητα ορίζεται από τη μέγιστη απόσταση δύο σημείων σε κάθε ομάδα.

Η ενιαία ή ελάχιστη σύνδεση (single or minimum linkage): σύμφωνα με τη μέθοδο αυτή, η ομοιότητα ορίζεται από την ελάχιστη απόσταση δύο σημείων σε κάθε ομάδα. [30]

Η ευκλείδεια απόσταση είναι αυτή που χρησιμοποιείται συνήθως για τον υπολογισμό των παραπάνω αποστάσεων. Ωστόσο στη βιβλιογραφία αναφέρονται και άλλοι τρόποι υπολογισμού της απόστασης όπως η απόσταση Μανχάταν. [30]

Η διαχωριστική ομαδοποίηση (divisive clustering) ορίζεται σαν το αντίθετο της συσσωρευτικής ομαδοποίησης και ακολουθεί την προσέγγιση "από πάνω προς τα κάτω" ("top-down"). Έτσι, αρχικά υπάρχει μια ομάδα δεδομένων η οποία διαιρείται σύμφωνα με τις διαφορές τους. Σημειώνεται ότι η διαχωριστική ομαδοποίηση δεν εφαρμόζεται με την ίδια συχνότητα με την συσσωρευτική. [30]

- Πιθανολογική ομαδοποίηση (probabilistic clustering)

Η πιθανολογική ομαδοποίηση είναι η μη επιβλεπόμενη τεχνική σύμφωνα με την οποία τα σημεία δεδομένων ομαδοποιούνται με βάση την πιθανότητα να ανήκουν σε μια συγκεκριμένη κατανομή. Η προσέγγιση αυτή χρησιμοποιείται για την επίλυση προβλημάτων εκτίμησης πυκνότητας (density estimation) ή "μαλακής" ("soft") ομαδοποίησης. Μία από τις πιο συχνά εφαρμόσιμες μεθόδους είναι το Gaussian Mixture Model (GMM). [30]

Τα Gaussian Mixture Models (GMM) ανήκουν στα μοντέλα μίξης (mixture models), πράγμα που σημαίνει ότι αποτελούνται από έναν απροσδιόριστο αριθμό συναρτήσεων κατανομής πιθανότητας και αξιοποιούνται κυρίως για τον προσδιορισμό της κατανομής πιθανότητας (γκαουσιανή ή κανονική) στην οποία ανήκουν τα σημεία δεδομένων. Στα μοντέλα GMM δεν είναι γνωστή η μέση τιμή και η διασπορά, οπότε θεωρείται ότι υπάρχει μια κρυφή μεταβλητή σύμφωνα με την οποία πραγματοποιείται η ομαδοποίηση. [30]

2. Συσχέτιση (association)

Ο κανόνας συσχέτισης (association rule) είναι μια μέθοδος η οποία βασίζεται σε κανόνες με σκοπό την εύρεση σχέσεων μεταξύ μεταβλητών σε ένα σύνολο δεδομένων. Η μέθοδος αυτή εφαρμόζεται συχνά για την κατανόηση των καταναλωτικών συνηθειών πελατών, βοηθώντας έτσι τις επιχειρήσεις να αναπτύξουν στοχευμένες στρατηγικές πωλήσεων και μηχανές συστάσεων (recommendation engines). Αν και υπάρχουν πολλοί αλγόριθμοι που χρησιμοποιούν τη μέθοδο των κανόνων συσχέτισης, όπως ο Apriori, ο Eclat και ο FP-Growth, ο αλγόριθμος Apriori εφαρμόζεται συχνότερα [30] και προτάθηκε το 1994 από τους R. Agrawal και R. Srikant. Όπως φαίνεται και από το όνομα του αλγορίθμου, χρησιμοποιείται προηγούμενη γνώση των ιδιοτήτων των στοιχείων που εμφανίζονται συχνά και μέσω επαναληπτικής προσέγγισης αξιοποιούνται k σύνολα στοιχείων για την εύρεση k+1 στοιχείων. [31]

3. Μείωση διαστάσεων (dimensionality reduction)

Η μείωση διαστάσεων είναι μια τεχνική που χρησιμοποιείται όταν ο αριθμός των διαστάσεων ή των χαρακτηριστικών σε ένα σύνολο δεδομένων είναι πολύ υψηλός. Αν και συνήθως το μεγάλο πλήθος δεδομένων αυξάνει την ακρίβεια των αποτελεσμάτων, σε κάποιες περιπτώσεις μπορεί να δυσχεραίνει την οπτικοποίηση του συνόλου δεδομένων ή και να παρουσιάζεται το φαινόμενο της υπερβολικής προσαρμογής του μοντέλου στα δεδομένα (overfitting). [30]

Η μείωση διαστάσεων εφαρμόζεται συνήθως στο στάδιο της προεπεξεργασίας των δεδομένων και σκοπός της είναι να μειωθεί ο αριθμός των εισερχόμενων δεδομένων σε ένα μέγεθος που το μοντέλο μπορεί να διαχειριστεί, διατηρώντας όσο είναι δυνατό την ακεραιότητα του συνόλου δεδομένων. Υπάρχουν διάφοροι μέθοδοι με τις οποίες μπορεί να επιτευχθεί αυτό, όπως οι παρακάτω.

- Ανάλυση κύριων συνιστωσών (principal component analysis, PCA)

Ο αλγόριθμος αυτός επιτυγχάνει μείωση της πλεονάζουσας πληροφορίας και συμπίεση του συνόλου δεδομένων μέσω εξαγωγής χαρακτηριστικών (feature extraction), δηλαδή χρησιμοποιεί γραμμικό μετασχηματισμό για να αποδώσει ένα σύνολο "κύριων συνιστωσών" ("principal components"). Η πρώτη κύρια συνιστώσα είναι αυτή στην οποία μεγιστοποιείται η διασπορά του συνόλου δεδομένων, ενώ η δεύτερη συνιστώσα είναι κάθετη στην πρώτη συνιστώσα. Η διαδικασία αυτή επαναλαμβάνεται ανάλογα με τον αριθμό των διαστάσεων και η επόμενη κύρια συνιστώσα είναι η κάθετη στις προηγούμενες συνιστώσες με τη μεγαλύτερη διασπορά. [30]

- Αποσύνθεση μοναδιαίας τιμής (singular value decomposition, SVD)

Η αποσύνθεση μοναδιαίας τιμής (SVD) επιτυγχάνει μείωση των διαστάσεων μέσω της παραγοντοποίησης ενός πίνακα A σε τρεις πίνακες χαμηλής τάξης. Η μέθοδος αυτή μπορεί να περιγραφεί από την παρακάτω εξίσωση:

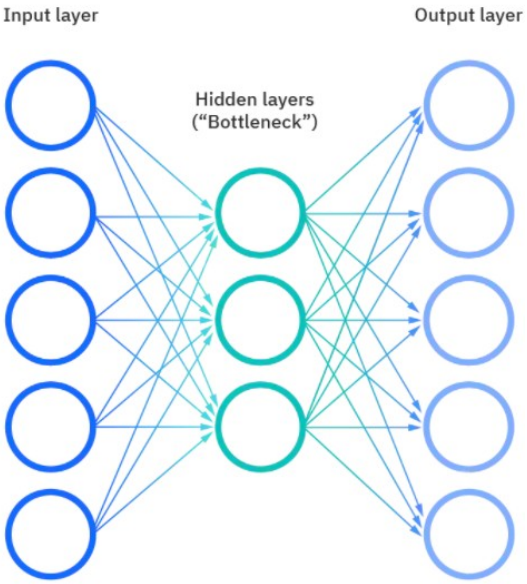
$$A = U \cdot S \cdot V^T \qquad (2.1)$$

Όπου U και V ορθογώνιοι πίνακες και S διαγώνιος πίνακας που περιλαμβάνει τα τετράγωνα των μη αρνητικών ιδιοτιμών του πίνακα A. Η μέθοδος αυτή, όπως και η μέθοδος PCA, εφαρμόζεται για να μειωθεί ο θόρυβος και να συμπιεστούν τα δεδομένα. [30]

- Αυτόματοι κωδικοποιητές (Autoencoders)

Οι αυτόματοι κωδικοποιητές είναι ένα είδος νευρωνικού δικτύου και χρησιμοποιούνται για να συμπιέσουν τα δεδομένα και στη συνέχεια να αναδημιουργήσουν μια αναπαράσταση των αρχικών δεδομένων. Όπως φαίνεται και παρακάτω στην Εικόνα 4, το κρυφό στρώμα συμπιέζει τα δεδομένα τα οποία ανακατασκευάζονται στο στρώμα εξόδου. Το τμήμα από το στρώμα εισόδου μέχρι το κρυφό στρώμα είναι γνωστό ως "κωδικοποίηση" ("encoding"), ενώ το τμήμα από το κρυφό στρώμα μέχρι το στρώμα εξόδου είναι γνωστό ως "αποκωδικοποίηση" ("decoding"). [30]

Στη συνέχεια του κεφαλαίου θα γίνει εκτενέστερη αναφορά στους αυτόματους κωδικοποιητές.



Εικόνα 4: Αυτόματος κωδικοποιητής

Πηγή:
https://www.ibm.com/cloud/learn/unsupervised-learning

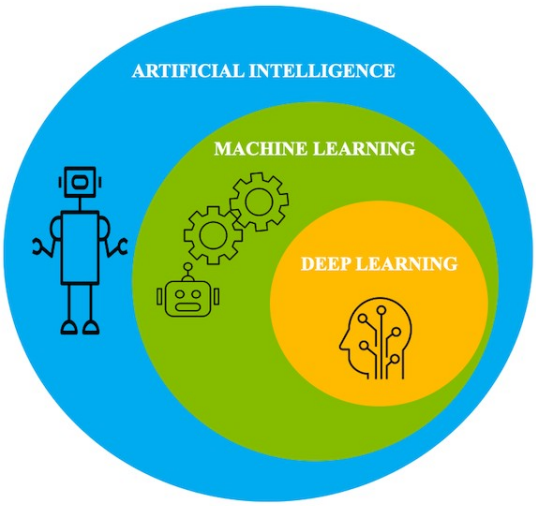
γ) Προκλήσεις μη επιβλεπόμενης μάθησης

Αν και η μη επιβλεπόμενη μάθηση έχει πολλά πλεονεκτήματα, η εκτέλεση μοντέλων μηχανικής μάθησης χωρίς ανθρώπινη παρέμβαση μπορεί να κρύβει και κάποιες προκλήσεις όπως:

- Η ύπαρξη μεγάλης υπολογιστικής πολυπλοκότητας λόγω του μεγάλου όγκου δεδομένων εκπαίδευσης.
- Η ανάγκη για μεγάλο χρόνο εκπαίδευσης.
- Η ύπαρξη μεγαλύτερου κινδύνου ανακρίβειας στα αποτελέσματα.
- Η ανάγκη ανθρώπινης παρέμβασης για επαλήθευση των αποτελεσμάτων.
- Η έλλειψη διαφάνειας ως προς τη διαδικασία ομαδοποίησης. [30]

2.4 Βαθιά μάθηση - Deep Learning

α) Ορισμός



Εικόνα 5: Συσχέτιση βαθιάς μάθησης με τη μηχανική μάθηση και την τεχνητή νοημοσύνη

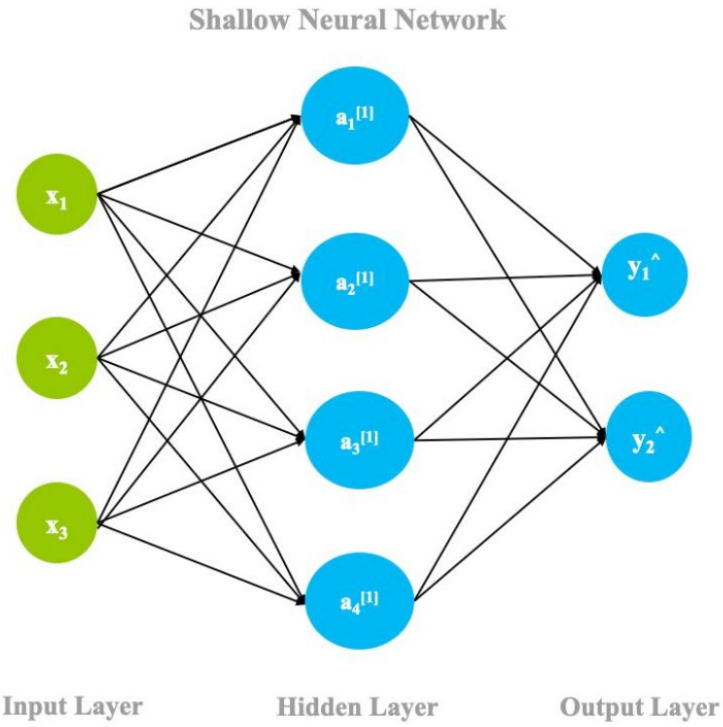
Πηγή:
https://developer.ibm.com/technologies/artificial-intelligence/articles/an-introduction-to-deep-learning/

Όπως φαίνεται και στην Εικόνα 5, η βαθιά μάθηση (deep learning) είναι ένας τομέας της μηχανικής μάθησης. Λόγω της όλο και μεγαλύτερης υπολογιστικής ισχύος των σύγχρονων υπολογιστών και της πληθώρας διαθέσιμων δεδομένων, οι αλγόριθμοι βαθιάς μάθησης επιτυγχάνουν να εντοπίζουν κρυμμένα μοτίβα στα δεδομένα ώστε να κάνουν προβλέψεις [33]. Τα μοντέλα βαθιάς μάθησης αποτελούνται από πολλά επίπεδα νευρωνικών δικτύων και η κατασκευή τους είναι εμπνευσμένη από τον σχεδιασμό του ανθρώπινου εγκεφάλου, σε απλοποιημένη μορφή. Κάθε στρώμα νευρώνων χρησιμοποιεί το αποτέλεσμα του προηγούμενου στρώματος ως είσοδο και το δίκτυο εκπαιδεύεται ως ενιαίο σύνολο [32].

Αφού η αρχιτεκτονική των μοντέλων βαθιάς μάθησης βασίζεται στην δομή του ανθρώπινου εγκεφάλου, είναι λογικό αρκετές θεμελιώδεις ορολογίες της βαθιάς μάθησης να μπορούν να αντιστοιχηθούν με τη νευρολογία. Έτσι, σε αντιστοίχιση με τους νευρώνες, οι οποίοι αποτελούν τα θεμελιώδη δομικά στοιχεία του εγκεφάλου, η αρχιτεκτονική της βαθιάς μάθησης περιλαμβάνει το perceptron, το οποίο είναι μια υπολογιστική μονάδα που επιτρέπει τη μοντελοποίηση μη γραμμικών συναρτήσεων. Όπως ένας νευρώνας στον ανθρώπινο εγκέφαλο μεταδίδει ηλεκτρικούς παλμούς σε όλο το νευρικό σύστημα, το perceptron λαμβάνει σήματα εισόδου και τα μετατρέπει σε σήματα εξόδου. Κάθε μέρος των δεδομένων που εισάγονται στο δίκτυο αναπαρίσταται από ένα ξεχωριστό επίπεδο από perceptrons, το οποίο ανιχνεύει την επαναλαμβανόμενη εμφάνιση τιμών. Το δίκτυο από perceptrons ονομάζεται νευρωνικό δίκτυο. [33]

β) Μορφές δικτύων βαθιάς μάθησης

Η πιο βασική μορφή νευρωνικού δικτύου περιλαμβάνει τρία στρώματα, το στρώμα εισόδου, ένα κρυμμένο στρώμα και το στρώμα εξόδου. Όταν το δίκτυο έχει αυτή την μορφή, η οποία φαίνεται και στην Εικόνα 6, τότε ονομάζεται ρηχό νευρωνικό δίκτυο (shallow neural network). [33]



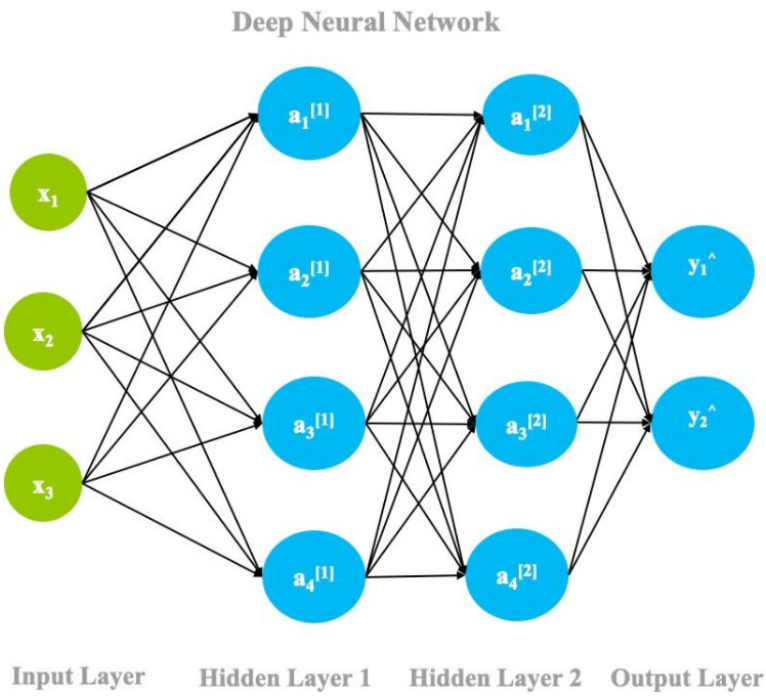
Εικόνα 6: Ρηχό νευρωνικό δίκτυο (γενική μορφή)

Πηγή: *https://developer.ibm.com/technologies/artificial-intelligence/articles/an-introduction-to-deep-learning/*

Το κάθε perceptron, όπως αναφέρθηκε και σε προηγούμενη ενότητα, υπολογίζει το σταθμισμένο άθροισμα των εισόδων που δέχεται, πολλαπλασιάζοντάς τες με τα αντίστοιχα βάρη και προσθέτοντας έναν αριθμό bias. Αυτό συμβαίνει σε όλα τα perceptrons του δικτύου, δηλαδή και στο στρώμα εξόδου. Ένα τέτοιο πέρασμα από το δίκτυο ονομάζεται προς τα εμπρός διάδοση (forward propagation). Αφού ολοκληρωθεί ένα πέρασμα προς τα εμπρός, το στρώμα εξόδου συγκρίνει τα αποτελέσματα με τις πραγματικές ετικέτες (labels) των δεδομένων και προσαρμόζει τα βάρη με βάση τις διαφορές τους. Αυτή η προς τα πίσω διαδικασία ονομάζεται οπισθοδιάδοση (backpropagation) [33]

Το αποτέλεσμα ενός περάσματος προς τα εμπρός διάδοσης (forward propagation) και οπισθοδιάδοσης (backpropagation) είναι μια αλλαγή στα βάρη των στρωμάτων και είναι ένα βήμα πιο κοντά στην μοντελοποίηση του συνόλου δεδομένων. Η διαδικασία σύγκλισης των παραμέτρων του νευρωνικού δικτύου ονομάζεται κάθοδος κλίσης (gradient descent), επειδή η διαδικασία χρησιμοποιεί την κλίση (gradient) για να ελαχιστοποιήσει το συνολικό σφάλμα. [33]

Το βαθύ νευρωνικό δίκτυο (deep neural network) ακολουθεί το πρότυπο του ρηχού νευρωνικού δικτύου, αλλά αυτή τη φορά υπάρχουν περισσότερα από ένα κρυφά στρώματα, όπως φαίνεται και στην Εικόνα 7. Η επιλογή του αριθμού των κρυφών στρωμάτων εξαρτάται από το πρόβλημα και το σύνολο δεδομένων που χρειάζεται να μοντελοποιηθεί. [33]



Εικόνα 7: Βαθύ νευρωνικό δίκτυο (γενική μορφή)

Πηγή: *https://developer.ibm.com/articles/an-introduction-to-deep-learning/*

γ) Εφαρμογές και προκλήσεις

Η βαθιά μηχανική μάθηση εφαρμόζεται ήδη σε πολλά πεδία και υπάρχουν προοπτικές για ακόμα πιο εκτεταμένη εφαρμογή της στο μέλλον. Ένας από τους τομείς στους οποίους εφαρμόζεται είναι ο τομέας της υγείας. Μέσω αναγνώρισης εικόνας η ανίχνευση καρκίνου από μαγνητικές τομογραφίες έχει ξεπεράσει τα ανθρώπινα επίπεδα ακρίβειας, λόγω της εύκολης πρόσβασης σε GPU (Graphics Processing Unit) και της πληθώρας διαθέσιμων δεδομένων. Ο τομέας της κατασκευής αυτόνομων οχημάτων επίσης στρέφεται σταδιακά προς την υλοποίηση, αφού μοντέλα που βασίζονται στην βαθιά μάθηση εκπαιδεύονται σε συνθήκες που περιλαμβάνουν από την αναγνώριση μιας πινακίδας μέχρι τον εντοπισμό πεζού. Επιπλέον εφαρμογές της βαθιάς μάθησης μπορούν να θεωρηθούν οι εξατομικευμένες συστάσεις προϊόντων στον τομέα του ηλεκτρονικού εμπορίου και οι προσωπικοί βοηθοί που βρίσκονται ενσωματωμένοι σε πολλές συσκευές και επιτυγχάνουν αναγνώριση φωνής, εξατομικευμένες συστάσεις και δημιουργία κειμένου. [33]

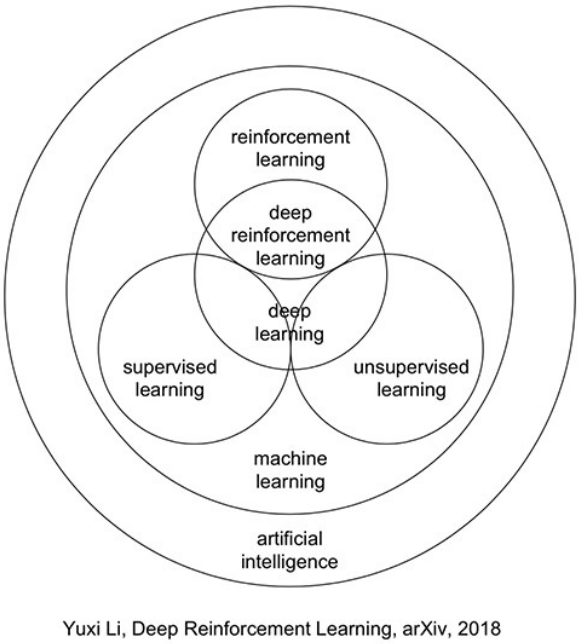
Αν και η βαθιά μάθηση εφαρμόζεται ευρέως τα τελευταία χρόνια, υπάρχουν και κάποιες προκλήσεις όπως η ανάγκη για δεδομένα. Τα μοντέλα έχουν ανάγκη για μεγάλο όγκο δεδομένων, ο οποίος δεν είναι διαθέσιμος σε πολλά πολύπλοκα προβλήματα, όπως η γλωσσική μετάφραση, με αποτέλεσμα να προκύπτουν αποτελέσματα χαμηλής ακρίβειας. Τεχνικές όπως η προσαρμογή σε τομείς (domain adaptation), δηλαδή η εφαρμογή γνώσεων που αποκτήθηκαν από συστήματα που υπάρχουν διαθέσιμοι πόροι σε συστήματα που οι πόροι είναι περιορισμένοι, φαίνεται να φέρνουν ενδιαφέροντα αποτελέσματα. [33]

Επίσης, παρόλο που οι αλγόριθμοι βαθιάς μάθησης φαίνεται να ξεπερνούν την ανθρώπινη ακρίβεια, δεν υπάρχει σαφής τρόπος σκέψης πίσω από κάθε πρόβλεψη που γίνεται. Αυτό καθιστά

δύσκολη τη χρήση τους σε εφαρμογές που απαιτείται η παροχή αιτιολογίας, όπως η έγκριση ή η απόρριψη ενός δανείου. [33]

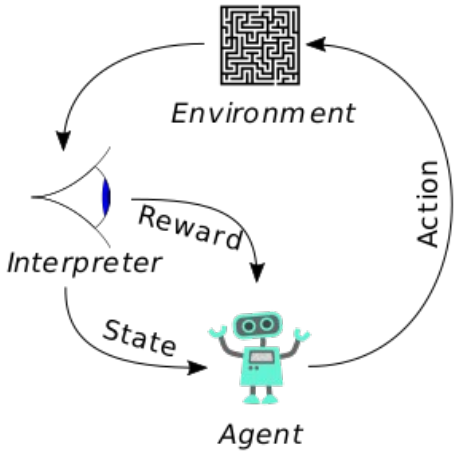
2.5 Ενισχυτική μάθηση - Reinforcement Learning

α) Ορισμός



Εικόνα 8: Συσχέτιση ενισχυτικής μάθησης με τη μηχανική και τη βαθιά μάθηση

Πηγή: *https://www.yash.com/blog/an-introduction-to-reinforcement-learning-part-1/*



Εικόνα 9: Διάγραμμα ενισχυτικής μάθησης

Πηγή: *https://www.unite.ai/what-is-deep-reinforcement-learning/*

Στη μηχανική μάθηση το σύστημα μαθαίνει εντοπίζοντας μοτίβα και προσφέροντας προβλέψεις. Η βαθιά μάθηση, όπως αναφέρθηκε και παραπάνω είναι ένα υποσύνολο της μηχανικής μάθησης της οποίας ο στόχος είναι και πάλι να επισημανθούν ή να ομαδοποιηθούν δεδομένα αλλά αυτή τη φορά με χρήση μοντέλων που περιλαμβάνουν βαθύ νευρωνικό δίκτυο. Σε αντίθεση με τα προηγούμενα ο στόχος της ενισχυτικής μάθησης δεν είναι η επισήμανση ή η ομαδοποίηση δεδομένων αλλά η επίτευξη ενός επιθυμητού στόχου [34]. Η σχέση της ενισχυτικής μάθησης με τη μηχανική μάθηση και τις υποκατηγορίες της φαίνεται στην Εικόνα 8. Πέρα από την κανονική μορφή ενισχυτικής μάθησης η οποία υπάρχει εδώ και δεκαετίες, η βαθιά ενισχυτική μάθηση μπορεί να οδηγήσει σε εντυπωσιακά αποτελέσματα δεδομένου ότι συνδυάζει χαρακτηριστικά της ενισχυτικής αλλά και της βαθιάς μάθησης. Δεν πρέπει να συγχέονται οι έννοιες της βαθιάς ενισχυτικής μάθησης και της βαθιάς μάθησης αφού στην περίπτωση της πρώτης οι είσοδοι μεταβάλλονται συνεχώς, πράγμα που δεν συμβαίνει στην κανονική βαθιά μάθηση. [35]

Η ενισχυτική μάθηση είναι η εκπαίδευση μοντέλων μηχανικής μάθησης με σκοπό τη λήψη μιας ακολουθίας αποφάσεων. Ο πράκτορας (agent), ο οποίος μπορεί να είναι μέρος του λογισμικού όπως

στην περίπτωση των παιχνιδιών ή μέρος του υλικού όπως ένα ρομπότ ή ένα αυτόνομο αυτοκίνητο [2], αντιμετωπίζει μια κατάσταση που μοιάζει με παιχνίδι και μαθαίνει να επιτυγχάνει ένα στόχο σε ένα πολύπλοκο περιβάλλον. Χρησιμοποιεί μια διαδικασία δοκιμής και σφάλματος για να βρει μια λύση στο πρόβλημα, ενώ λαμβάνει είτε ανταμοιβές, είτε ποινές για την ενέργειά του. Ο στόχος του είναι να μεγιστοποιήσει τη συνολική ανταμοιβή. [32]

Παρόλο που οι κανόνες που αφορούν τις ανταμοιβές είναι καθορισμένοι, δεν δίνονται στο μοντέλο υποδείξεις ή προτάσεις σχετικά με τη λύση του προβλήματος και έτσι το μοντέλο ξεκινάει από τυχαίες δοκιμές και καταλήγει σε υπεράνθρωπες ικανότητες. Σε αντίθεση με τους ανθρώπους, αν ένας αλγόριθμος ενισχυτικής μάθησης εκτελείται σε υπολογιστή με επαρκή υπολογιστική ισχύ, είναι δυνατό να συλλέξει εμπειρία εκτελώντας χιλιάδες παράλληλες δοκιμές με σκοπό τη μεγιστοποίηση της ανταμοιβής. [32] Σε αντίθεση με τη μηχανική μάθηση και τη βαθιά μάθηση, η ενισχυτική μάθηση δεν προϋποθέτει την ύπαρξη συλλεγμένων δεδομένων, αφού ο αλγόριθμος εκπαιδεύεται από μόνος του στο περιβάλλον του. [34]

β) Εφαρμογές και διακρίσεις

Η εκπαίδευση μοντέλων που ελέγχουν αυτόνομα αυτοκίνητα είναι μια πιθανή εφαρμογή της ενισχυτικής μάθησης, στην οποία ο υπολογιστής δεν θα λαμβάνει οδηγίες για την οδήγηση του αυτοκινήτου αλλά θα μαθαίνει σύμφωνα με τα δικά του λάθη. Σε μια ιδανική κατάσταση, το μόνο που θα είναι απαραίτητο να καθοριστεί θα είναι η συνάρτηση ανταμοιβών [32]. Γενικά οι αλγόριθμοι ενισχυτικής μάθησης μπορούν να εφαρμοστούν σε πληθώρα εφαρμογών αφού μπορούν να μάθουν δυναμικά από το περιβάλλον τους και να εντοπίσουν πιθανές ενέργειες [35].

Στην περίπτωση της βαθιάς ενισχυτικής μάθησης, το περιβάλλον αναπαρίσταται συνήθως με εικόνες, με την κάθε εικόνα να είναι μια αποτύπωση του περιβάλλοντος μια συγκεκριμένη χρονική στιγμή. Ο πράκτορας (agent) πρέπει να αναλύσει τις εικόνες και να εξάγει πληροφορίες από αυτές για να αποφασίσει ποια ενέργεια είναι η κατάλληλη. Οι δύο διαφορετικές τεχνικές με τις οποίες πραγματοποιείται συνήθως η βαθιά ενισχυτική μάθηση είναι η μάθηση με βάση τιμές (value-based learning) και η μάθηση με βάση πολιτικές (policy-based learning).

- Μάθηση με βάση τιμές (value-based learning): Οι αλγόριθμοι μάθησης με βάση τιμές χρησιμοποιούν τεχνικές όπως τα συνελικτικά νευρωνικά δίκτυα και τα Deep-Q-Networks και λειτουργούν μετατρέποντας την εικόνα σε κλίμακα του γκρι και περικόπτοντας τα περιττά μέρη της. Η εικόνα στη συνέχεια περνάει από στάδια συνέλιξης (convolution) και συγκέντρωσης (pooling) με σκοπό να εξαχθούν τα σημαντικά τμήματα της εικόνας. Αυτά χρησιμοποιούνται για τον υπολογισμό της τιμής Q για τις διάφορες ενέργειες που ο πράκτορας θα μπορούσε να κάνει και αυτές οι τιμές χρησιμοποιούνται για να προσδιοριστεί η καλύτερη πορεία δράσης. Οι τιμές Q είναι η συνολική ανταμοιβή που δίνεται στον πράκτορα στο τέλος μιας ακολουθίας ενεργειών. Αφού υπολογιστούν οι αρχικές τιμές Q, πραγματοποιείται οπισθοδιάδοση (backpropagation) για να προσδιοριστούν πιο ακριβείς τιμές Q [35].
- Μάθηση με βάση πολιτικές (policy-based learning): οι αλγόριθμοι μάθησης με βάση πολιτικές χρησιμοποιούνται όταν ο αριθμός των ενεργειών που μπορεί να εκτελέσει ο πράκτορας είναι πολύ μεγάλος, πράγμα που συμβαίνει σε προβλήματα του πραγματικού κόσμου. Σε αυτές τις περιπτώσεις δεν είναι δυνατό να εφαρμοστεί η προηγούμενη μέθοδος αφού ο υπολογισμός όλων των τιμών Q για τις επιμέρους ενέργειες δεν είναι ρεαλιστικός. Έτσι, αντί να υπολογίζουν τις επιμέρους τιμές, υιοθετούν πολιτικές μέσω τεχνικών που ονομάζονται κλίσεις πολιτικής (policy gradients). Οι κλίσεις πολιτικής λειτουργούν λαμβάνοντας μια κατάσταση,

υπολογίζοντας πιθανότητες για τις ενέργειες με βάση τις προηγούμενες εμπειρίες του πράκτορα και έτσι τελικά επιλέγεται η πιθανότερη ενέργεια. Η διαδικασία επαναλαμβάνεται μέχρι το τέλος της περιόδου αξιολόγησης και την απόδοση ανταμοιβών στον πράκτορα. Στη συνέχεια οι παράμετροι του δικτύου ενημερώνονται με οπισθοδιάδοση (backpropagation). [35]

γ) Προκλήσεις ενισχυτικής μάθησης

Αφού οι αλγόριθμοι ενισχυτικής μάθησης μαθαίνουν με βάση το περιβάλλον τους, όπως είναι λογικό πρέπει να δοθεί ιδιαίτερη σημασία στην προετοιμασία του περιβάλλοντος προσομοίωσης, το οποίο εξαρτάται από την εργασία που είναι επιθυμητό να εκτελεστεί. Όταν για παράδειγμα ο στόχος είναι να δημιουργηθεί ένας αλγόριθμος που θα φτάσει σε υπεράνθρωπες ικανότητες σε ένα παιχνίδι όπως το σκάκι, τότε το περιβάλλον προσομοίωσης είναι σχετικά εύκολο να δημιουργηθεί. Από την άλλη, όταν ο σκοπός είναι η κατασκευή ενός μοντέλου το οποίο θα οδηγεί ένα αυτόνομο αυτοκίνητο, η επίδοση του μοντέλου έχει ιδιαίτερη σημασία αφού προκύπτουν ζητήματα ασφαλείας. Έτσι, όπως είναι λογικό το μοντέλο αυτό πρέπει να μάθει πώς να εκτελεί βασικές ενέργειες, όπως η ακινητοποίηση του αυτοκινήτου ή η αποφυγή εμποδίου, σε ένα ασφαλές και ελεγχόμενο περιβάλλον πριν κυκλοφορήσει στον δρόμο. [32]

Επιπλέον, ιδιαίτερη σημασία έχει και η διαδικασία μεταφοράς του μοντέλου από το περιβάλλον εκπαίδευσης στον πραγματικό κόσμο και όπως και η κλιμάκωση και η προσαρμογή του νευρωνικού δικτύου που ελέγχει τον πράκτορα. Ακόμη, το γεγονός ότι δεν υπάρχει άλλος τρόπος επικοινωνίας με τον πράκτορα πέρα από το σύστημα ανταμοιβών και ποινών μπορεί να οδηγήσει σε καταστροφική λήθη (catastrophic forgetting), δηλαδή τη διαγραφή παλιάς γνώσης όταν αποκτάται νέα. [32].

Το φαινόμενο αυτό παρατηρείται όταν ένα τεχνητό νευρωνικό δίκτυο εκπαιδεύεται σε μια μη σταθερή κατανομή δεδομένων, όπως στην περίπτωση δυο διαδοχικών εργασιών. Στην περίπτωση της ενισχυτικής μάθησης η κατανομή των δεδομένων που συγκεντρώνονται καθώς ο πράκτορας αλληλεπιδρά με το περιβάλλον είναι συχνά μη σταθερή κατά τη διάρκεια εκπαίδευσης για μια μεμονωμένη εργασία ή και μεταξύ εργασιών. Έτσι, μπορεί να εμφανιστεί καταστροφική λήθη δεδομένου ότι οι εμπειρίες του πράκτορα συσχετίζονται χρονικά και η πολιτική του αλλάζει καθώς μαθαίνει. Ένας τρόπος αντιμετώπισης του προβλήματος είναι η αποθήκευση των γνώσεων σε μια βάση δεδομένων και η χρήση της στην εκπαίδευση. Ωστόσο η λύση αυτή δεν είναι ικανοποιητική αφού δημιουργεί μεγάλο υπολογιστικό όγκο και επιπλέον δεν εξηγεί πως ο ανθρώπινος εγκέφαλος επιτυγχάνει συνεχή μάθηση χωρίς καταστροφική λήθη [7].

Μία ακόμη πρόκληση που μπορεί να αντιμετωπιστεί είναι η επίτευξη τοπικού βέλτιστου, δηλαδή ο πράκτορας να εκτελεί την εργασία αλλά όχι με τον βέλτιστο ή επιθυμητό τρόπο. Ένα τέτοιο παράδειγμα θα μπορούσε να είναι η περίπτωση ενός συστήματος που φτάνει στον τελικό του προορισμό πραγματοποιώντας άλματα και όχι περπατώντας. Τέλος, υπάρχει και η περίπτωση πρακτόρων που επιτυγχάνουν να βελτιστοποιήσουν τις ανταμοιβές χωρίς όμως να επιτύχουν τον τελικό στόχο. Για παράδειγμα σε ένα παιχνίδι με αγώνες, το όχημα μπορεί να εξερευνεί την περιοχή μαζεύοντας ανταμοιβές χωρίς όμως να επιδιώκει να ολοκληρώσει τον αγώνα. [32]

2.6 Αυτόματοι κωδικοποιητές – Autoencoders

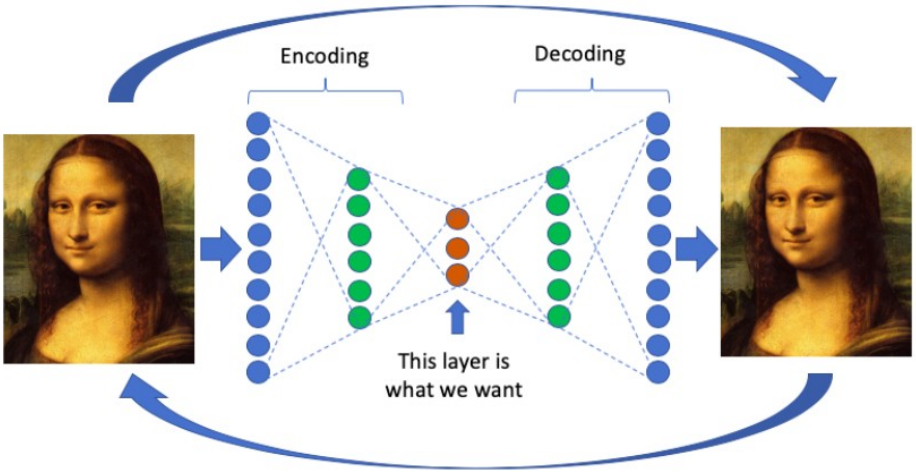
α) Ορισμός

Στην ενότητα της μη επιβλεπόμενης μάθησης έγινε μια σύντομη αναφορά στους αυτόματους κωδικοποιητές (autoencoders). Στη συνέχεια θα αναλυθούν εκτενέστερα αφού αποτελούν σημαντικό μέρος της παρούσας εργασίας.

Ένας αυτόματος κωδικοποιητής είναι ένα νευρωνικό δίκτυο το οποίο εκπαιδεύεται με σκοπό να αντιγράψει την είσοδο στην έξοδό του και δεν απαιτεί επισημασμένα δεδομένα αφού ανήκει στην μη επιβλεπόμενη μάθηση. Έχει τουλάχιστον ένα κρυφό επίπεδο το οποίο χρησιμοποιείται για να αναπαραστήσει τα δεδομένα εισόδου. Στο δίκτυο μπορεί να διακριθεί ένας κωδικοποιητής (encoder) και ένας αποκωδικοποιητής (decoder). Ένας αυτόματος κωδικοποιητής που πετυχαίνει απλά να αντιγράφει την είσοδό του στην έξοδο δεν είναι ιδιαίτερα χρήσιμος αφού στην πραγματικότητα παρουσιάζει ιδιαίτερο ενδιαφέρον το κρυμμένο στρώμα στον πυρήνα του δικτύου. [36] Συνήθως σχεδιάζονται ώστε να επιτρέπουν να αντιγραφεί μόνο κατά προσέγγιση η είσοδος στην έξοδο, αφού ο αριθμός των νευρώνων στα κρυφά στρώματα είναι μικρότερος από το πλήθος τους στα στρώματα εισόδου και εξόδου, με σκοπό το μοντέλο να αναγκαστεί να δώσει προτεραιότητα στα κομμάτια της εισόδου που πρέπει να αντιγραφούν, μαθαίνοντας έτσι χρήσιμες ιδιότητες για τα δεδομένα. [8]

Όπως αναφέρθηκε, τα κρυφά στρώματα πρέπει να έχουν λιγότερες διαστάσεις από τα στρώματα εισόδου και εξόδου. Στην περίπτωση που δεν ισχύει αυτό, το δίκτυο θα μπορούσε απλώς να αντιγράψει τις τιμές εισόδου στις τιμές εξόδου, πράγμα που δεν είναι επιθυμητό αφού δεν εξάγεται ουσιαστική πληροφορία. [36]

Η Εικόνα 10 δείχνει την διαδικασία κωδικοποίησης και αποκωδικοποίησης. Η διαδικασία κωδικοποίησης συμπιέζει τα δεδομένα εισόδου ώστε να φτάσουν στο κεντρικό στρώμα, το οποίο είναι το στρώμα με τους λιγότερους νευρώνες. Η διαδικασία αποκωδικοποίησης είναι συμμετρική με τη διαδικασία κωδικοποίησης ως προς τον αριθμό στρωμάτων και νευρώνων, πράγμα που εφαρμόζεται συνήθως και στην πράξη. Η διαδικασία αποκωδικοποίησης επαναφέρει τις διαστάσεις στην αρχική τους μορφή, έτσι ώστε η είσοδος και η έξοδος να είναι συγκρίσιμες. [36]



Εικόνα 10: Αυτόματος κωδικοποιητής

Πηγή: *https://towardsdatascience.com/anomaly-detection-with-autoencoder-b4cdce4866a6*

β) Εφαρμογές

Οι αυτόματοι κωδικοποιητές αρχικά εφαρμόστηκαν, όπως αναφέρθηκε και σε προηγούμενη ενότητα, για μείωση διαστάσεων [36]. Ο Geoffrey Hinton έδειξε το 2006 ότι ένας εκπαιδευμένος

αυτόματος κωδικοποιητής προσφέρει μικρότερο σφάλμα και καλύτερο διαχωρισμό συστάδων σε σχέση με τις 30 πρώτες συνιστώσες μιας ανάλυσης κύριων συνιστωσών (PCA). [9] Αυτό συμβαίνει επειδή οι αυτόματοι κωδικοποιητές, έχοντας μη γραμμική συνάρτηση ενεργοποίησης και πολλαπλά στρώματα, είναι δυνατό να εκτελέσουν μη γραμμικούς μετασχηματισμούς, ενώ η PCA χρησιμοποιεί γραμμική άλγεβρα. Οι αυτόματοι κωδικοποιητές επίσης χρησιμοποιούνται σε εφαρμογές όρασης υπολογιστών και στην επεξεργασία εικόνας, για παράδειγμα για τη μετατροπή μιας ασπρόμαυρης εικόνας σε έγχρωμη και την αφαίρεση θορύβου [36].

Εκτός των παραπάνω, οι αυτόματοι κωδικοποιητές εφαρμόζονται και για ανίχνευση ακραίων τιμών (outlier detection). Η ανίχνευση ακραίων τιμών μπορεί να εφαρμοστεί στο στάδιο της προεπεξεργασίας των δεδομένων με σκοπό να εντοπιστούν πιθανά προβλήματα με τα ίδια τα δεδομένα. [11] Οι ακραίες τιμές είναι σημεία δεδομένων που διαφέρουν σημαντικά από τα υπόλοιπα [10] και συχνά εμφανίζουν ανομοιομορφία και υψηλή διακύμανση οπότε είναι δύσκολο να μοντελοποιηθούν [13].

Σε πολλές εφαρμογές οι ακραίες τιμές παρουσιάζουν ιδιαίτερο ενδιαφέρον ή ακόμα και μεγαλύτερο από τις φυσιολογικές. Ένα κλασικό παράδειγμα τέτοιας περίπτωσης είναι η περίπτωση ανίχνευσης απάτης. Οι ακραίες τιμές που ανιχνεύονται μπορεί να υποδεικνύουν άτομα ή πελάτες μιας επιχείρησης τα οποία παρουσιάζουν συμπεριφορά εκτός του φυσιολογικού εύρους. [11]

Οι μέθοδοι που χρησιμοποιούνται συχνά για ανίχνευση ακραίων τιμών μπορούν να ταξινομηθούν είτε ως βασισμένες στην κατανομή είτε ως βασισμένες στην απόσταση, ενώ υπάρχει περίπτωση αυτές οι δύο κατηγορίες να επικαλύπτονται. Εναλλακτικά οι μέθοδοι ανίχνευσης ακραίων τιμών μπορούν να βασιστούν στο αν η μέθοδος παρέχει κάποια βαθμολογία (score) ακραίων τιμών ή αν προσδιορίζει την ακραία τιμή από τον όγκο των δεδομένων ή από μια επιφάνεια παλινδρόμησης. Στις μεθόδους που βασίζονται στην κατανομή περιλαμβάνονται μοντέλα όπως το SmartSifter και το κάθε σημείο δεδομένων βαθμολογείται ανάλογα με τον βαθμό που διαταράσσει το τρέχον μοντέλο που εκπαιδεύεται. Στις μεθόδους που βασίζονται στην απόσταση, χρησιμοποιούνται αποστάσεις όπως η ευκλείδεια και η απόσταση Mahalanobis. [11]

Μια άλλη ιδιαίτερα χρήσιμη τεχνική για τον εντοπισμό ακραίων τιμών είναι η χρήση αλγορίθμου ομαδοποίησης, όπως ο CURE ή ο BIRCH, και στη συνέχεια ο χαρακτηρισμός ως ακραίες τιμές των ομάδων ή δεδομένων που απέχουν πολύ από τις υπόλοιπες ομάδες. Παράλληλα υπάρχουν και μέθοδοι οπτικοποίησης, στις οποίες η απόσταση μεταξύ σημείων προβάλλεται σε ένα δισδιάστατο επίπεδο, επιτρέποντας στους χρήστες να εντοπίζουν τις ακραίες τιμές, αν και στην εφαρμογή τους εμφανίζονται προβλήματα υποκειμενικότητας και κλιμάκωσης. Ένα παράδειγμα διαθέσιμου εργαλείου οπτικοποίησης είναι το xgobi. Τέλος, μια μέθοδος στην οποία αξιοποιείται νευρωνικό δίκτυο για την ανίχνευση ακραίων τιμών είναι η προσέγγιση του νευρωνικού δικτύου του Sykacek, στην οποία χρησιμοποιείται ένα πολυεπίπεδο perceptron (multilayer perceptron) ως μοντέλο παλινδρόμησης και οι ακραίες τιμές αντιμετωπίζονται ως υπόλοιπα εκτός των ράβδων σφάλματος. [11]

Οι μέθοδοι βαθιάς μάθησης έχουν επιδείξει σημαντικά πλεονεκτήματα στην εκμάθηση χαρακτηριστικών και έχουν αποδειχθεί ιδιαίτερα αποτελεσματικές σε διάφορες εφαρμογές όρασης υπολογιστών και στην ανίχνευση ανωμαλιών. Ο D. Xu και οι συνεργάτες του προτείνουν έναν στοιβαγμένο αυτόματο κωδικοποιητή (stacked autoencoder) για την αυτόματη εκμάθηση της αναπαράστασης χαρακτηριστικών και χρησιμοποιούν μοντέλα SVM για την πρόβλεψη του βαθμού ανωμαλίας (anomaly score). Η μέθοδος αυτή εξάγει τμήματα της εικόνας και στη συνέχεια τα μετατρέπει σε μονοδιάστατα διανύσματα, τα οποία χρησιμοποιούνται σαν είσοδος στον αυτόματο κωδικοποιητή, αφαιρώντας όμως έτσι τη χωρική πληροφορία. Ο M. Hassan και οι συνεργάτες του προτείνουν έναν πλήρως συνελικτικό αυτόματο κωδικοποιητή για την εκμάθηση της χωροχρονικής

κανονικότητας. Αφού οι πράξεις συνέλιξης γίνονται μόνο χωρικά, η χρονική πληροφορία δεν διατηρείται.[13] Η μέθοδος εντοπισμού ακραίων τιμών και γενικά ανώμαλων συμβάντων με χρήση αυτόματου κωδικοποιητή μπορεί συνολικά να συνοψιστεί ως η μέθοδος στην οποία το μοντέλο εκπαιδεύεται στα φυσιολογικά δείγματα δεδομένων και στη συνέχεια μπορεί να αναγνωρίζει τα μη φυσιολογικά ως αυτά που αποκλίνουν από την κανονικότητα [12].

Στα πλαίσια της παρούσας εργασίας χρησιμοποιείται αυτόματος κωδικοποιητής για να εντοπιστούν ρωγμές. Αφού οι ρωγμές μπορούν γενικά να θεωρηθούν ασυνέχειες ή ανώμαλα φαινόμενα σε σχέση με την ομοιομορφία του μη ρηγματωμένου τοίχου, είναι δυνατό να εφαρμοστούν όσα αναφέρθηκαν συνοπτικά παραπάνω σε σχέση με την ανίχνευση ακραίων τιμών.

3 *Μεθοδολογία*

Στο κεφάλαιο αυτό θα αναλυθούν οι λόγοι για τους οποίους επιλέχθηκε να χρησιμοποιηθεί αυτόματος κωδικοποιητής, η λειτουργία του και τα χαρακτηριστικά του συγκεκριμένου αυτόματου κωδικοποιητή που δημιουργήθηκε στα πλαίσια της παρούσας εργασίας.

3.1 Αιτιολόγηση επιλογής αυτόματου κωδικοποιητή

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, αυτόματος κωδικοποιητής (autoencoder) είναι ένα νευρωνικό δίκτυο, το οποίο αποτελείται από ένα στρώμα εισόδου, ένα στρώμα εξόδου και μερικά κρυμμένα στρώματα. Σκοπός του είναι να αναπαράξει τα δεδομένα εισόδου στην έξοδο του, μαθαίνοντας όμως και κάποιες χρήσιμες πληροφορίες για αυτά. Αυτό επιτυγχάνεται μέσω της αρχιτεκτονικής του. Συγκεκριμένα τα κρυμμένα στρώματα έχουν μικρότερο αριθμό νευρώνων από τα στρώματα εισόδου και εξόδου, αναγκάζοντας έτσι το μοντέλο να κρατήσει τις απαραίτητες πληροφορίες για τα δεδομένα αντί απλώς να τα αντιγράψει στην έξοδο.

Ο σκοπός της παρούσας εργασίας είναι η αναγνώριση ρωγμών σε εικόνες με χρήση τεχνολογιών βαθιάς μηχανικής μάθησης. Για να επιτευχθεί ο σκοπός αυτός θα ήταν δυνατό να χρησιμοποιηθούν τεχνικές επιβλεπόμενης ή μη επιβλεπόμενης μάθησης.

Εξ ορισμού, για να εφαρμοστεί επιβλεπόμενη μάθηση είναι απαραίτητο, εκτός των δεδομένων, να υπάρχει και η γνώση της πραγματικής κατάστασης με τη μορφή ετικετών (labels), δηλαδή θα πρέπει να είναι γνωστό αν στην πραγματικότητα η εικόνα που εισάγεται στο μοντέλο περιλαμβάνει ρωγμή. Με αυτό τον τρόπο το μοντέλο, μέσω της εκπαίδευσης του και της προσαρμογής των παραμέτρων του για την πρόβλεψη των δεδομένων εκπαίδευσης, είναι σε θέση να πραγματοποιήσει προβλέψεις για νέα δεδομένα τα οποία δεν έχει ήδη αντιμετωπίσει. Οι ετικέτες αυτές είναι αναγκαίο να δημιουργηθούν μέσω ανθρώπινης παρατήρησης, πράγμα που είναι ιδιαίτερα επίπονο και χρονοβόρο, ιδιαίτερα όταν ο αριθμός των στοιχείων του συνόλου δεδομένων είναι μεγάλος. Επίσης, υπάρχει η πιθανότητα ανθρώπινου λάθους, το οποίο θα επηρεάσει συνεπώς την ποιότητα των προβλέψεων από το μοντέλο. Επιπλέον υπάρχει ο κίνδυνος της υπερβολικής προσαρμογής στα δεδομένα (overfitting) με αποτέλεσμα το μοντέλο να είναι σε θέση να πραγματοποιήσει ακριβείς προβλέψεις για τα δεδομένα εκπαίδευσης αλλά να μην μπορεί να γενικεύσει τις γνώσεις του σε δεδομένα που δεν έχει αντιμετωπίσει ξανά.

Αντίθετα, στην περίπτωση της μη επιβλεπόμενης μάθησης δεν χρειάζονται labels οπότε δεν είναι απαραίτητο να είναι εκ των προτέρων γνωστό αν η εκάστοτε εικόνα περιλαμβάνει ρωγμή. Το γεγονός αυτό εξαλείφει τον κίνδυνο ανθρώπινου λάθους κατά τη διάρκεια της επισήμανσης των δεδομένων εκπαίδευσης. Ακόμη, το μοντέλο που αξιοποιείται είναι ένας αυτόματος κωδικοποιητής (autoencoder) ο οποίος, σύμφωνα με τη μεθοδολογία που ακολουθείται για την αναγνώριση ανώμαλων συμβάντων, εκπαιδεύεται με δεδομένα τα οποία δεν περιλαμβάνουν ρωγμές. Η εξασφάλιση ότι τα δεδομένα εκπαίδευσης δεν περιλαμβάνουν ρωγμές είναι αρκετά εύκολο να πραγματοποιηθεί ακόμα και χωρίς ανθρώπινη παρατήρηση, φωτογραφίζοντας το δομικό στοιχείο που επιθυμείται να ελεγχθεί μόλις ολοκληρωθεί η κατασκευή του. Με αυτό τον τρόπο το μοντέλο εκπαιδεύεται στην υγιή κατάσταση ώστε μελλοντικά να είναι σε θέση να αναγνωρίσει συμβάντα που αποκλίνουν από αυτήν. Συνεπώς, για τη συγκεκριμένη εφαρμογή προτιμάται η χρήση μη επιβλεπόμενης μάθησης και συγκεκριμένα ενός αυτόματου κωδικοποιητή, αφού είναι επιθυμητό να υπάρχει καλή απόδοση χωρίς την ανάγκη για επισημασμένα δεδομένα.

3.2 Ανάλυση της μεθοδολογίας

Όπως αναφέρθηκε παραπάνω, για την εφαρμογή της αναγνώρισης ρωγμών επιλέγεται η χρήση μη επιβλεπόμενης μάθησης και συγκεκριμένα ενός αυτόματου κωδικοποιητή. Τα δεδομένα εκπαίδευσης, για τα οποία θα γίνει εκτενέστερη αναφορά στο επόμενο κεφάλαιο, είναι φωτογραφίες που απεικονίζουν τα τοιχώματα μιας σήραγγας στην υγιή τους κατάσταση, δηλαδή χωρίς ρωγμές. Η συλλογή των δεδομένων αυτών μπορεί να γίνει φωτογραφίζοντας τα τοιχώματα μόλις ολοκληρωθεί η κατασκευή της σήραγγας, εξασφαλίζοντας έτσι την απουσία ρωγμών.

Εκπαιδεύοντας τον αυτόματο κωδικοποιητή με τις φωτογραφίες της υγιούς κατάστασης της σήραγγας, μαθαίνει να αναπαράγει στην έξοδό του την υγιή αυτή κατάσταση, δηλαδή το ομοιόμορφο τοίχωμα. Με αυτό τον τρόπο η περίπτωση της υπερβολικής προσαρμογής στα δεδομένα δεν είναι ιδιαίτερα ανησυχητική, αφού το μοντέλο δεν προσπαθεί να αντιληφθεί μια τάση που μπορεί να υπάρχει στα δεδομένα και να πραγματοποιήσει προβλέψεις σύμφωνα με αυτή αλλά απλώς να μοντελοποιήσει την ομοιομορφία τους.

Αφού το μοντέλο εκπαιδευτεί και είναι σε θέση να αναπαράξει την ομοιόμορφη κατάσταση, είναι δυνατό να υπολογιστεί μια συνάρτηση απωλειών για να ελεγχθεί πόσο επιτυχημένα πραγματοποιείται η αναπαραγωγή της κάθε εικόνας που εισάγεται σε αυτό. Στην περίπτωση αυτή επιλέχθηκε το μέσο τετραγωνικό σφάλμα το οποίο υπολογίζεται από την παρακάτω σχέση: [37]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \qquad (3.1)$$

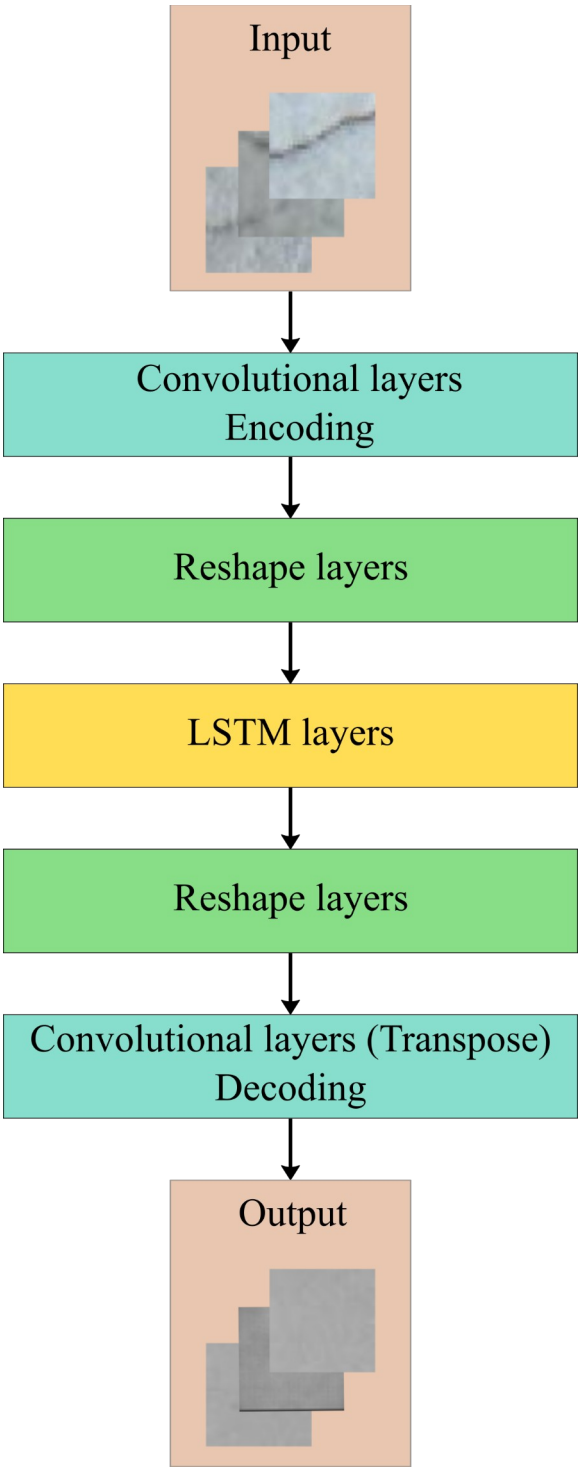
Όπου n το πλήθος των εικονοστοιχείων της κάθε εικόνας, Y ο πίνακας με τις τιμές της εικόνας που χρησιμοποιήθηκε σαν είσοδος στο μοντέλο και $\hat{Y}$ ο πίνακας με τις τιμές της εικόνας που παράχθηκε από το μοντέλο.

Αν στο εκπαιδευμένο μοντέλο ζητηθεί να αναπαράξει μια εικόνα που δεν περιλαμβάνει ρωγμή, δηλαδή μοιάζει με τα δεδομένα εκπαίδευσης, τότε θα καταφέρει να την αναπαράξει ικανοποιητικά και το μέσο τετραγωνικό σφάλμα αναμένεται να είναι μικρό. Αντίθετα, αν στο μοντέλο εισαχθεί μια εικόνα που περιλαμβάνει ρωγμή ή κάποιο άλλο χαρακτηριστικό που απέχει από την ομοιομορφία των δεδομένων εκπαίδευσης, το μέσο τετραγωνικό σφάλμα θα είναι μεγαλύτερο και έτσι θα είναι εμφανές ότι στη συγκεκριμένη εικόνα είναι πιθανό να υπάρχει ρωγμή. Έτσι, είναι δυνατό να οριστεί ένα κατώφλι στο μέσο τετραγωνικό σφάλμα, σύμφωνα με το οποίο διαχωρίζονται οι εικόνες σε αυτές που απεικονίζουν υγιή τοίχο και σε αυτές που απεικονίζουν κάποιο ανώμαλο συμβάν που είναι πιθανό να είναι ρωγμή.

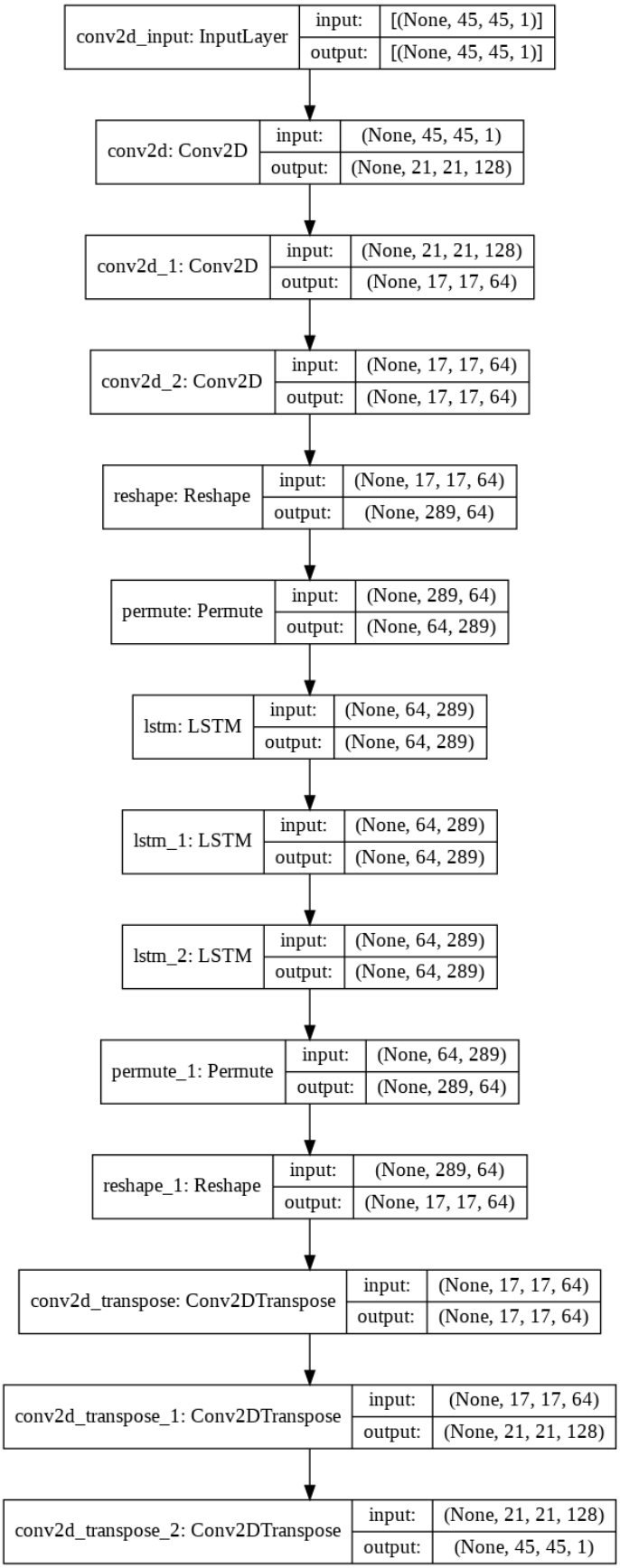
3.3 Περιγραφή μοντέλου

Το μοντέλο που δημιουργήθηκε για τις ανάγκες της παρούσας εργασίας είναι ένας συνελικτικός αυτόματος κωδικοποιητής σε χωροχρονικά πρότυπα (convolutional spatiotemporal autoencoder). Αποτελείται συνολικά από 13 στρώματα από τα οποία τα τρία πρώτα πραγματοποιούν δισδιάστατη συνέλιξη, τα επόμενα δύο αφορούν μετατροπή διαστάσεων και τα επόμενα τρία είναι στρώματα long-short term memory (LSTM), δηλαδή στρώματα μακροπρόθεσμης βραχυπρόθεσμης μνήμης. Στο κεντρικό στρώμα LSTM είναι δυνατό να θεωρηθεί ότι υπάρχει ένας νοητός άξονας συμμετρίας και έτσι τα στρώματα που ακολουθούν είναι συμμετρικά ως προς αυτά που προηγήθηκαν.

Ακολουθούν στην επόμενη σελίδα σχηματικές αναπαραστάσεις του μοντέλου. Στην Εικόνα 11 φαίνονται οι διαστάσεις των δεδομένων εισόδου (45 × 45 εικονοστοιχεία και ένα κανάλι με τόνους του γκρι), όλα τα επιμέρους στρώματα του μοντέλου και οι διαστάσεις της εισόδου και εξόδου στο εκάστοτε στρώμα. Παρατηρείται η συμμετρία που αναφέρθηκε παραπάνω ως προς το κεντρικό στρώμα LSTM, καθώς και το γεγονός ότι οι διαστάσεις των δεδομένων κατά την εισαγωγή τους και κατά την εξαγωγή τους από το μοντέλο είναι ίδιες ώστε η είσοδος και η έξοδος να είναι συγκρίσιμες. Επίσης παρατηρείται ότι οι διαστάσεις στο εσωτερικό του μοντέλου γίνονται κατά πολύ μικρότερες από τις αρχικές (17×17 αντί για 45×45) σύμφωνα με την κλασική αρχιτεκτονική ενός autoencoder. Στην Εικόνα 12 απεικονίζεται συνοπτικά το μοντέλο, ομαδοποιώντας τα όμοια επίπεδα.



Εικόνα 12: Περιληπτική σχηματική αναπαράσταση του συνελικτικού αυτόματου κωδικοποιητή σε χωροχρονικά πρότυπα



Εικόνα 11: Σχηματική αναπαράσταση του συνελικτικού αυτόματου κωδικοποιητή σε χωροχρονικά πρότυπα

Η πλειονότητα των στρωμάτων του παραπάνω αυτόματου κωδικοποιητή είναι στρώματα που είτε πραγματοποιούν συνέλιξη, είτε μεταθετική συνέλιξη (transposed convolution). Η μεταθετική συνέλιξη είναι ένας μετασχηματισμός ο οποίος έχει αντίθετη κατεύθυνση από την κανονική συνέλιξη [39]. Έτσι, αφού η συνέλιξη αποτελεί τόσο σημαντικό τμήμα του δικτύου, είναι απαραίτητο να αναλυθεί εκτενέστερα.

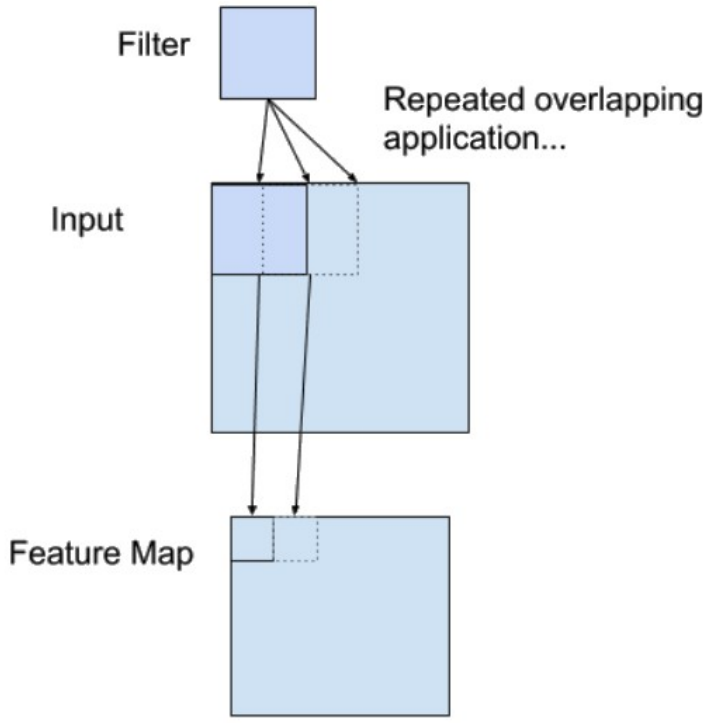
α) Συνέλιξη

Η συνέλιξη είναι μια γραμμική λειτουργία η οποία περιλαμβάνει τον πολλαπλασιασμό ενός δισδιάστατου πίνακα βαρών με την είσοδο, η οποία επίσης έχει τη μορφή δισδιάστατου πίνακα. Ο δισδιάστατος πίνακας βαρών ονομάζεται φίλτρο (filter) ή πυρήνας (kernel) και οι τιμές των βαρών του παραμένουν σταθερές κατά την εφαρμογή του φίλτρου σε μια εικόνα. Όπως είναι λογικό το φίλτρο είναι μικρότερο από τα δεδομένα εισόδου και μεταξύ του εκάστοτε τμήματος των δεδομένων και του φίλτρου πραγματοποιείται εσωτερικό γινόμενο, το οποίο συνοψίζεται στην παρακάτω σχέση.

$$\alpha \cdot \beta = \sum_{i=1}^{n} \alpha_i \beta_i \qquad (3.2)$$

Όπως φαίνεται και στην εξίσωση 3.2, πραγματοποιείται πολλαπλασιασμός μεταξύ των στοιχείων του φίλτρου με τα αντίστοιχα στοιχεία του κάθε τμήματος της εισόδου και στη συνέχεια τα γινόμενα αυτά αθροίζονται, με αποτέλεσμα για κάθε θέση του φίλτρου να προκύπτει μια ενιαία τιμή.

Η χρήση ενός φίλτρου μικρότερων διαστάσεων από την είσοδο είναι σκόπιμη αφού επιτρέπει στο φίλτρο να πολλαπλασιαστεί πολλές φορές με την είσοδο σε διαφορετικά σημεία της. Συγκεκριμένα, το φίλτρο εφαρμόζεται σε κάθε επικαλυπτόμενο τμήμα μεγέθους ίσου με το μέγεθος του φίλτρου από αριστερά προς τα δεξιά και από πάνω προς τα κάτω, όπως φαίνεται στην διπλανή εικόνα. [38]



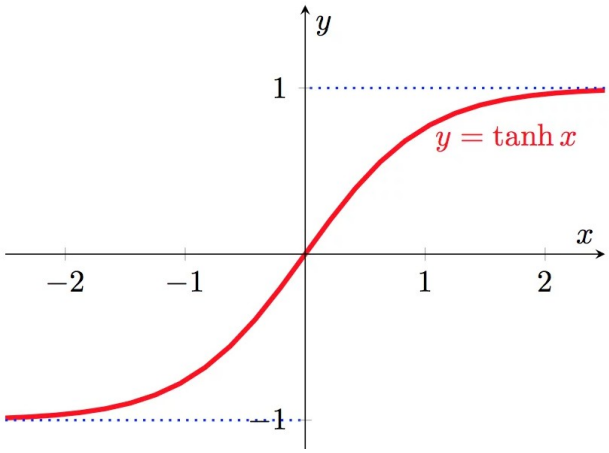
Εικόνα 13: Παράδειγμα εφαρμογής συνέλιξης

Πηγή: *https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/*

Εκτός από το μέγεθος του φίλτρου, το αποτέλεσμα της συνέλιξης εξαρτάται από το βήμα του φίλτρου πάνω στην εικόνα, το οποίο ονομάζεται stride και επηρεάζει το μέγεθος της εξόδου, αλλά και από το padding. Το padding εφαρμόζεται όταν οι διαστάσεις της εικόνας (ή γενικά του πίνακα στον οποίο εφαρμόζεται συνέλιξη) δεν είναι πολλαπλάσιες των διαστάσεων του φίλτρου, λαμβάνοντας βέβαια υπόψιν και την τιμή του stride, και έτσι παρατηρείται μια απόκλιση στα άκρα. Το padding μπορεί να είναι μηδενικό, δηλαδή η τελευταία συνέλιξη να απορρίπτεται αν οι διαστάσεις δεν συμπίπτουν, ίδιο, δηλαδή να

εξασφαλίζεται ότι το στρώμα εξόδου έχει ίδιο μέγεθος με το στρώμα εισόδου, ή πλήρες προσθέτοντας μηδενικά στα όρια της εισόδου ώστε να εναρμονίζονται οι διαστάσεις. [41]

Η εφαρμογή του φίλτρου, το οποίο μπορεί να χαρακτηριστεί και σαν ανιχνευτής χαρακτηριστικών, σε όλη την δεδομένη εικόνα δίνει σαν αποτέλεσμα έναν δισδιάστατο πίνακα ο οποίος ονομάζεται χάρτης χαρακτηριστικών (feature map). Αφού δημιουργηθεί ο χάρτης χαρακτηριστικών, η κάθε τιμή του περνάει από μια μη γραμμική συνάρτηση ενεργοποίησης όπως η ReLU (Rectified Linear Unit) ή η σιγμοειδής (sigmoid). Στην περίπτωση του αυτόματου κωδικοποιητή που χρησιμοποιήθηκε στα πλαίσια της παρούσας εργασίας χρησιμοποιήθηκε μέγεθος φίλτρου 5×5 και 3×3 εικονοστοιχείων, ενώ σαν συνάρτηση ενεργοποίησης εφαρμόστηκε η υπερβολική εφαπτομένη (tanh), της οποία η γραφική παράσταση φαίνεται στη διπλανή εικόνα. [38] Όπως φαίνεται στο γράφημα, οι έντονα θετικές είσοδοι της συνάρτησης απεικονίζονται κοντά στο 1, ενώ οι έντονα αρνητικές κοντά στο -1. Το γεγονός αυτό καθιστά την υπερβολική εφαπτομένη ιδανική επιλογή συνάρτησης ενεργοποίησης σε προβλήματα δυαδικής ταξινόμησης [40].

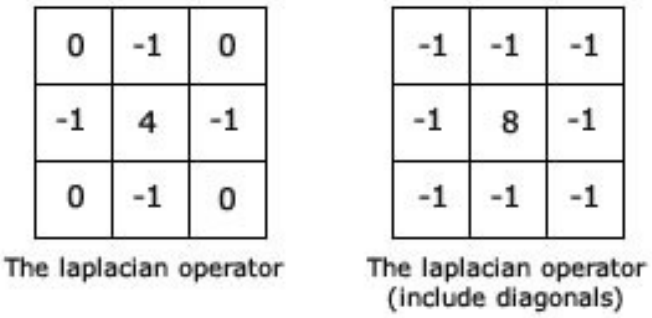


Εικόνα 14: Διάγραμμα υπερβολικής εφαπτομένης

Πηγή: *https://www.journaldev.com/49083/tanh-activation-function*

Αν το φίλτρο που εφαρμόζεται σε μια εικόνα έχει σχεδιαστεί για να ανιχνεύει κάποιο χαρακτηριστικό στα δεδομένα εισόδου τότε η συστηματική εφαρμογή του φίλτρου σε ολόκληρη την εικόνα δίνει την ευκαιρία στο φίλτρο να ανακαλύψει την ύπαρξη του χαρακτηριστικού στην εικόνα και να απεικονίσει την ανακάλυψη αυτή στον χάρτη χαρακτηριστικών.

Η εφαρμογή συνέλιξης σε δεδομένα εικόνων είναι συνήθης τεχνική στην όραση υπολογιστών, η οποία είναι ένας τομέας της τεχνητής νοημοσύνης που επιτρέπει στους υπολογιστές να αντλούν σημαντικές πληροφορίες από ψηφιακές εικόνες, βίντεο ή άλλα εποπτικά μέσα και με βάση τις πληροφορίες αυτές να λαμβάνονται αποφάσεις και να πραγματοποιούνται συστάσεις [41]. Στο παρελθόν τα φίλτρα σχεδιάζονταν χειροκίνητα και εφαρμόζοντάς τα στην εικόνα προέκυπτε χάρτης χαρακτηριστικών ο οποίος βοηθούσε στην ανάλυση και στην ερμηνεία των εικόνων [38]. Ένα χαρακτηριστικό τέτοιο παράδειγμα είναι η ανίχνευση ακμών με το φίλτρο του Laplace, του οποίου η μορφή φαίνεται στην Εικόνα 15 και τα αποτελέσματα εφαρμογής του σε μια ασπρόμαυρη εικόνα φαίνονται παρακάτω στην Εικόνα 16.



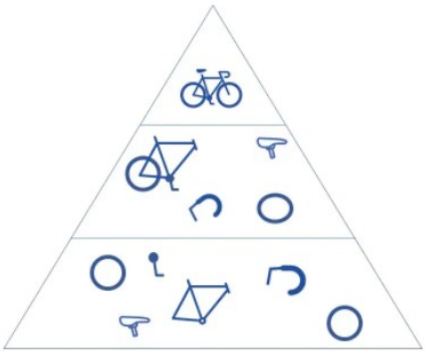
Εικόνα 15: Φίλτρο Laplace για ανίχνευση ακμών

Πηγή: https://aishack.in/tutorials/sobel-laplacian-edge-detectors/

Εικόνα 16: Αποτέλεσμα εφαρμογής του φίλτρου Laplace σε εικόνα

Πηγή: *https://aishack.in/tutorials/sobel-laplacian-edge-detectors/*

Η καινοτομία που έφερε η χρήση συνέλιξης σε νευρωνικά δίκτυα είναι ότι τα βάρη των φίλτρων, αντί για χειροκίνητα, προσδιορίζονται μέσω της εκπαίδευσης του μοντέλου. [38] Επιπλέον, στα συνελικτικά νευρωνικά δίκτυα δεν μαθαίνεται μόνο ένα φίλτρο, αλλά είναι δυνατό να χρησιμοποιηθούν πολλαπλά φίλτρα για την εξαγωγή πολλαπλών χαρακτηριστικών σε μια δεδομένη είσοδο [38]. Στην περίπτωση του συνελικτικού αυτόματου κωδικοποιητή που δημιουργήθηκε στα πλαίσια της παρούσας εργασίας, στα στρώματα συνέλιξης όπως φαίνεται ορίστηκαν 128 ή 64 φίλτρα. Έτσι μετά από κάθε συνελικτικό στρώμα δημιουργούνται 128 ή 64 χάρτες χαρακτηριστικών αντίστοιχα.



Εικόνα 17: Ιεραρχική δομή εντοπισμού ποδηλάτου με συνελικτικό νευρωνικό δίκτυο

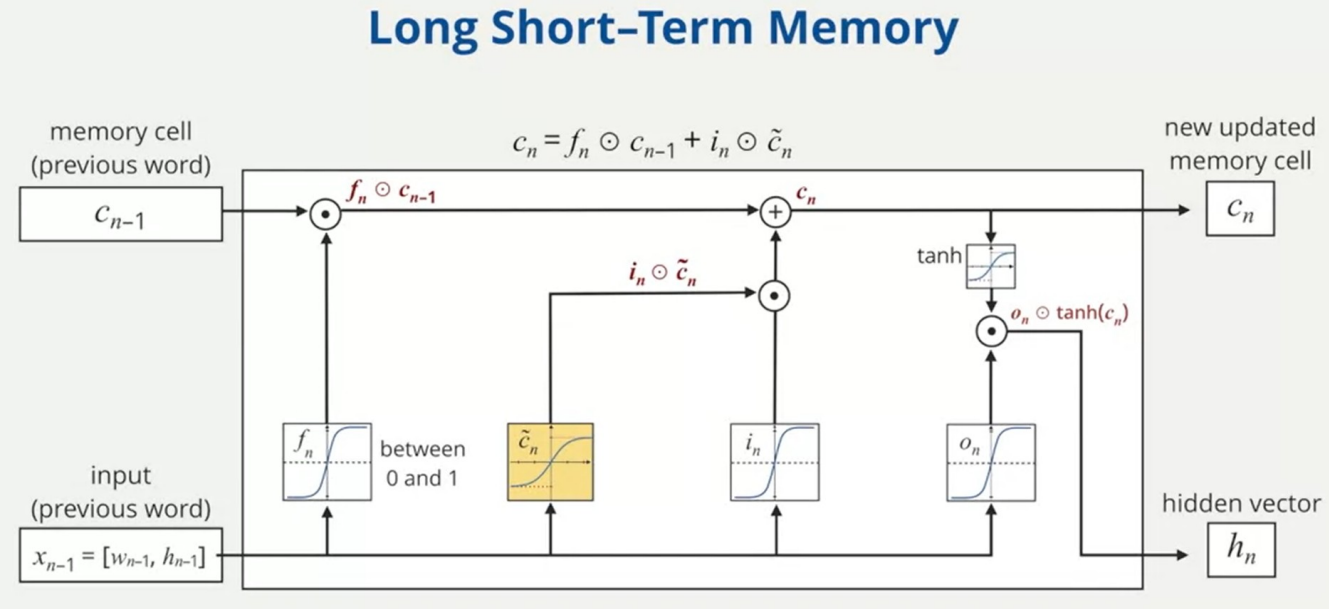
Πηγή: https://www.ibm.com/cloud/learn/convolutional-neural-networks

Σε ένα συνελικτικό νευρωνικό δίκτυο είναι συχνό φαινόμενο να στοιβάζονται πολλαπλά στρώματα συνέλιξης. Όταν συμβαίνει αυτό, όπως και στην περίπτωση της παρούσας εργασίας, η δομή του δικτύου γίνεται ιεραρχική, αφού τα μεταγενέστερα στρώματα δέχονται σαν είσοδο την έξοδο των προηγούμενων και τα χαρακτηριστικά που αυτά έχουν ήδη εντοπίσει. Για παράδειγμα για τον εντοπισμό ενός ποδηλάτου, είναι δυνατό να θεωρηθεί ότι ένα ποδήλατο είναι άθροισμα κάποιων μερών όπως το τιμόνι και οι τροχοί. Στη διπλανή εικόνα, στη βάση της πυραμίδας φαίνεται κάθε μεμονωμένο μέρος του ποδηλάτου και η άνοδος σε υψηλότερα επίπεδα, δηλαδή σε μεταγενέστερα συνελικτικά στρώματα, αντιπροσωπεύει τον συνδυασμό των μερών αυτών μέχρι να επιτευχθεί ο εντοπισμός ενός ολοκληρωμένου ποδηλάτου [41].

β) Μακροπρόθεσμη βραχυπρόθεσμη μνήμη (LSTM)

Όπως αναφέρθηκε παραπάνω και όπως φαίνεται στην Εικόνα 11, εκτός από στρώματα συνέλιξης, ο αυτόματος κωδικοποιητής περιλαμβάνει και τρία στρώματα μακροπρόθεσμης βραχυπρόθεσμης μνήμης (long short-term memory), ή εν συντομία LSTM.

Η αρχιτεκτονική LSTM παρουσιάστηκε από τους S. Hochreiter και J. Schmidhuber ως λύση στο πρόβλημα της εξαφανιζόμενης κλισης (vanishing gradient). Η εργασία τους [14] ασχολείται με την αντιμετώπιση του προβλήματος των μακροχρόνιων εξαρτήσεων (long-term dependencies), δηλαδή το πρόβλημα της αδυναμίας ακριβούς πρόβλεψης της τρέχουσας κατάστασης από το νευρωνικό δίκτυο αν η προηγούμενη κατάσταση που την επηρεάζει δεν ανήκει στο πρόσφατο παρελθόν. [42] Για να μετριαστεί το πρόβλημα αυτό, το LSTM περιλαμβάνει κύτταρα μνήμης σε κάθε κόμβο του κρυφού στρώματος και έτσι οι πληροφορίες είναι δυνατό να αποθηκευτούν και να προσπελαστούν για μεγάλο χρονικό διάστημα. Τα δίκτυα LSTM χρησιμοποιούνται ευρέως στην πρόβλεψη χρονοσειρών όπως στην περίπτωση της αυτόματης μετάφρασης, της πρόβλεψης φαινομένων όπως η ατμοσφαιρική ρύπανση και της αναγνώρισης ομιλίας. [16]



Εικόνα 18: Διαγραμματική μορφή LSTM

Πηγή: *Lawrence Carin, David Carlson, Timothy Dunn, Kevin Liang, "Introduction to Machine Learning", Duke University, Coursera*

Στην Εικόνα 18 φαίνεται η αρχιτεκτονική του LSTM. Ένα στρώμα LSTM στην περίπτωση του παραδείγματος που απεικονίζεται στην εικόνα δέχεται σαν είσοδο ένα διάνυσμα x_{n-1} ενώ με c συμβολίζεται το κύτταρο μνήμης. Το στρώμα ελέγχεται συνολικά από τέσσερα νευρωνικά δίκτυα. [15]

Τα τρία νευρωνικά δίκτυα ονομάζονται δίκτυα ελέγχου (control neural networks). Το διάνυσμα δεδομένων x_{n-1} εισάγεται στα δίκτυα αυτά και η έξοδός τους έχει και πάλι τη μορφή διανύσματος και συμβολίζεται αντίστοιχα στην εικόνα ως f_n, i_n και o_n. Όλα τα δίκτυα ελέγχου έχουν την ίδια βασική μορφή η οποία είναι η μορφή του πολυστρωματικού perceptron (multilayer perceptron), ενώ σαν συνάρτηση ενεργοποίησης χρησιμοποιείται η σιγμοειδής (sigmoid). Όπως είναι γνωστό από τις ιδιότητες της σιγμοειδούς, οι τιμές της κυμαίνονται στο διάστημα μεταξύ του 0 και του 1, πράγμα που

συνεπάγεται ότι όλα τα στοιχεία των διανυσμάτων f_n, i_n και o_n είναι αριθμοί που ανήκουν στο ίδιο διάστημα. [15]

Όπως αναφέρθηκε, η λειτουργία του LSTM ελέγχεται συνολικά από τέσσερα νευρωνικά δίκτυα με τα τρία από αυτά να είναι τα δίκτυα ελέγχου. Το τέταρτο νευρωνικό δίκτυο είναι αυτό που ελέγχει το κύτταρο μνήμης και η συνάρτηση ενεργοποίησης που εφαρμόζεται σε αυτή την περίπτωση είναι η υπερβολική εφαπτομένη (tanh). Συνεπώς, η έξοδος του δικτύου αυτού, η οποία στην εικόνα συμβολίζεται ως $\widetilde{c_n}$, είναι ένα διάνυσμα που περιλαμβάνει αριθμούς στο διάστημα -1 έως 1. [15]

Αφού υπολογιστούν τα τέσσερα διανύσματα εισάγοντας το διάνυσμα δεδομένων στα τέσσερα νευρωνικά δίκτυα, υπολογίζεται η νέα τιμή του κυττάρου μνήμης. Η τιμή αυτή υπολογίζεται από την παρακάτω σχέση:

$$c_n = f_n \odot c_{n-1} + i_n \odot \widetilde{c_n} \qquad (3.3)$$

Σημειώνεται ότι το σύμβολο $\odot$ δείχνει ότι πραγματοποιείται πολλαπλασιασμός του κάθε στοιχείου του ενός διανύσματος με το αντίστοιχο στοιχείο του άλλου, με αποτέλεσμα το διάνυσμα που προκύπτει να είναι συνδυασμός των δύο προγενέστερων. Παρατηρώντας την εξίσωση αυτή φαίνεται ότι το διάνυσμα f_n ελέγχει τον βαθμό που η ήδη αποθηκευμένη στο κύτταρο μνήμης πληροφορία θα επηρεάσει την νέα του τιμή, δηλαδή ελέγχει ποια μέρη του διανύσματος c_{n-1} θα διατηρηθούν και ποια θα "ξεχαστούν". Αυτό επιτυγχάνεται μέσω των τιμών του διανύσματος f_n αφού σε κάθε πολλαπλασιασμό που εκτελείται, αν η τιμή του στοιχείου του f_n είναι κοντά στο 0, τότε και το αποτέλεσμα του πολλαπλασιασμού με το αντίστοιχο στοιχείο του διανύσματος c_{n-1} θα είναι κοντά στο 0. Αντίθετα, αν η τιμή του στοιχείου του f_n είναι κοντά στο 1, τότε το αποτέλεσμα του πολλαπλασιασμού με το αντίστοιχο στοιχείο του διανύσματος c_{n-1} θα είναι κοντά στην αρχική τιμή του στοιχείου του c_{n-1}. [15]

Παράλληλα, το διάνυσμα i_n μέσω του πολλαπλασιασμού καθορίζει τη συμβολή της νέας εκτίμησης για το κύτταρο μνήμης $\widetilde{c_n}$ με όμοιο τρόπο. Δηλαδή μέσω του πολλαπλασιασμού, αν η τιμή του διανύσματος i_n είναι κοντά στο 0, τότε η συμβολή του αντίστοιχού στοιχείου του διανύσματος $\widetilde{c_n}$ είναι μικρή. Από την άλλη πλευρά, αν η τιμή του i_n είναι κοντά στο 1, τότε η αρχική τιμή του διανύσματος $\widetilde{c_n}$ σχεδόν διατηρεί την ακεραιότητα της. [15]

Αφού υπολογιστεί το νέο διάνυσμα που αποθηκεύεται στο κύτταρο μνήμης (c_n), τότε αυτό εισάγεται σε μία υπερβολική εφαπτομένη (tanh) με αποτέλεσμα να λαμβάνει τιμές που κυμαίνονται μεταξύ του -1 και του 1. Όπως φαίνεται στην εικόνα, το διάνυσμα που εξάγεται από το στρώμα LSTM υπολογίζεται από την εξίσωση:

$$h_n = o_n \odot \tanh(c_n) \qquad (3.4)$$

Συνολικά, ο βαθμός συμμετοχής των προηγούμενων δεδομένων καθορίζεται μέσω του f_n, ο βαθμός εισαγωγής νέων δεδομένων καθορίζεται μέσω του i_n και τέλος το διάνυσμα o_n ελέγχει τον βαθμό στον οποίο το κύτταρο μνήμης επηρεάζει την έξοδο. Σημειώνεται ότι όλες οι παράμετροι του LSTM γίνονται γνωστές μέσω μη επισημασμένων δεδομένων, αφού χρησιμοποιείται προηγούμενη γνώση για το πρόσφατο παρελθόν, όπως ένα σετ εικόνων αντί για μία εικόνα τη φορά, για την παραγωγή αποτελέσματος.

γ) Στρώματα αναδιαμόρφωσης (reshape) και αντιμετάθεσης (permute)

Αφού καλύφθηκαν οι περιπτώσεις των στρωμάτων συνέλιξης και LSTM, τα υπόλοιπα τέσσερα στρώματα του δικτύου έχουν βοηθητικό ρόλο και, όπως είναι λογικό, ακολουθούν τη συμμετρική μορφή που παρατηρείται συνολικά στο δίκτυο.

Τα συνελικτικά στρώματα δέχονται σαν είσοδο την εκάστοτε εικόνα και οι διαστάσεις της εξόδου είναι της μορφής (batch size, rows, columns, filters). Παράλληλα τα στρώματα LSTM δέχονται σαν είσοδο δεδομένα της μορφής (batch size, timesteps, features). Έτσι, μέσω της αναδιαμόρφωσης (reshape) και της αντιμετάθεσης (permute) επιτυγχάνεται τα στρώματα LSTM που επιθυμούν τα χαρακτηριστικά που έχουν εξαχθεί (features) να τα λάβουν με διάσταση ίση με το γινόμενο των γραμμών και των στηλών.

4 *Μελέτη εφαρμογής*

Στο κεφάλαιο αυτό πρόκειται να περιγραφούν εκτενέστερα τα δεδομένα, να παρουσιαστεί ο κώδικας που συντάχθηκε στη γλώσσα προγραμματισμού Python, ενώ στη συνέχεια θα παρουσιαστούν τα αποτελέσματα που προέκυψαν από την εφαρμογή του στα δεδομένα ελέγχου.

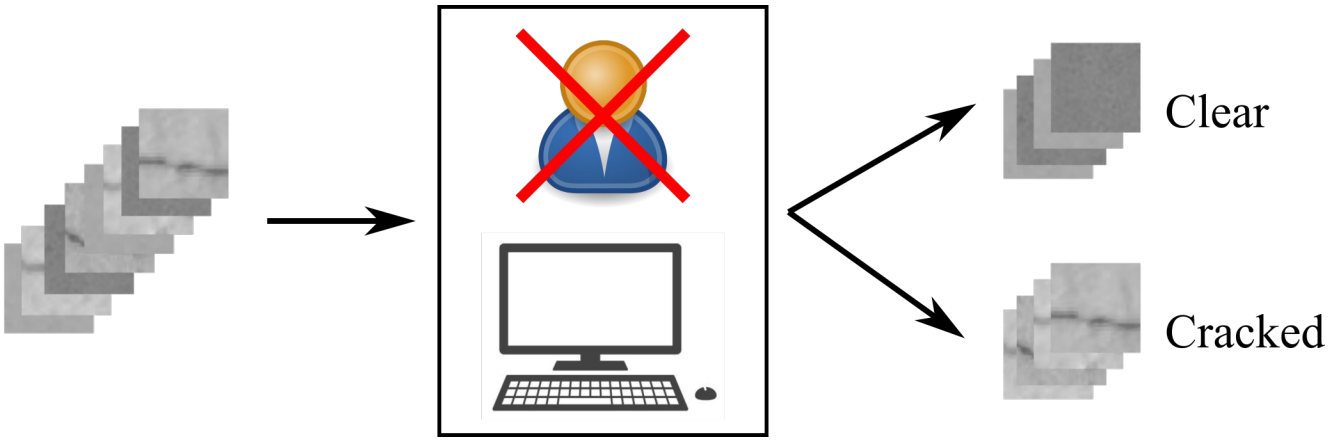
4.1 Περιγραφή προβλήματος

Όπως αναφέρθηκε και σε προηγούμενη ενότητα ο εντοπισμός ρωγμών σε μια κατασκευή είναι καίριας σημασίας αφού η αντοχή της κατασκευής μπορεί να εξαρτηθεί από τον έγκαιρο εντοπισμό τους. Ο έλεγχος της κατασκευής μέσω ανθρώπινης παρατήρησης, όπως είναι λογικό, είναι μια διαδικασία χρονοβόρα, επίπονη και επιρρεπής στα ανθρώπινα λάθη. Έτσι δεν είναι δυνατό να πραγματοποιείται εντατική παρακολούθηση της κατασκευής με αυτό τον τρόπο και υπάρχει ανάγκη για αυτοματοποίηση της διαδικασίας.

Με σκοπό να αυτοματοποιηθεί η διαδικασία εντοπισμού ρωγμών, είναι δυνατό να εφαρμοστούν αλγόριθμοι μηχανικής μάθησης σε εικόνες της κατασκευής. Ένα εκπαιδευμένο μοντέλο

επιβλεπόμενης μάθησης είναι δυνατό να ταξινομήσει με μεγάλη ακρίβεια εικόνες ανάλογα με το περιεχόμενό τους, δηλαδή να αναγνωρίσει αν στην εκάστοτε εικόνα περιλαμβάνεται ρωγμή ή αν απεικονίζεται υγιής τοίχος. Το γεγονός αυτό φαίνεται και στο επόμενο κεφάλαιο της παρούσας εργασίας, στο οποίο χρησιμοποιείται και ένα επιβλεπόμενο νευρωνικό δίκτυο για να ταξινομήσει τα δεδομένα. Αν και το μοντέλο επιβλεπόμενης μάθησης αποτελεί μια λύση η οποία μπορεί να αυτοματοποιήσει τη διαδικασία εντοπισμού ρωγμών, έχει κάποια βασικά μειονεκτήματα. Συγκεκριμένα, για την εκπαίδευση του μοντέλου σε μία εφαρμογή αναγνώρισης ρωγμών απαιτούνται δεδομένα εκπαίδευσης που περιλαμβάνουν φυσιολογικές εικόνες (εικόνες που απεικονίζουν υγιή τοίχο) και μη φυσιολογικές εικόνες (εικόνες που απεικονίζουν ρωγμή) και τις αντίστοιχες αληθείς ετικέτες τους. Δηλαδή είναι απαραίτητο να είναι γνωστό εκ των προτέρων αν η κάθε εικόνα του συνόλου δεδομένων εκπαίδευσης είναι φυσιολογική ή μη φυσιολογική. Η παραγωγή των αληθών ετικετών για τα δεδομένα πραγματοποιείται μέσω ανθρώπινης παρατήρησης, πράγμα που συνεπάγεται μεγάλο κόστος, απαιτήσεις χρόνου και την πιθανότητα ανθρώπινου λάθους, στο οποίο το μοντέλο θα προσαρμοστεί κατά την εκπαίδευσή του.

Η προσέγγιση της μη επιβλεπόμενης μάθησης από την άλλη πλευρά παρέχει λύση σε πολλά από τα παραπάνω μειονεκτήματα. Ο αυτόματος κωδικοποιητής που χρησιμοποιείται στα πλαίσια της παρούσας εργασίας, ο οποίος είναι ένα νευρωνικό δίκτυο μη επιβλεπόμενης μάθησης, δεν χρειάζεται αληθείς ετικέτες για την εκπαίδευσή του, μειώνοντας κατά πολύ το κόστος της διαδικασίας. Παράλληλα, απαιτούνται για την εκπαίδευσή του μόνο φυσιολογικά δεδομένα, δηλαδή δεδομένα που απεικονίζουν την υγιή κατάσταση. Η εξασφάλιση ότι τα δεδομένα είναι φυσιολογικά μπορεί να επιτυγχάνεται εύκολα αν η συλλογή τους πραγματοποιηθεί με την ολοκλήρωση της κατασκευής του δομικού στοιχείου που επιθυμείται να ελέγχεται. Έτσι, συνολικά το πρόβλημα που επιλύεται στη συνέχεια του κεφαλαίου είναι ο εντοπισμός ρωγμών με αυτοματοποιημένο τρόπο μέσω μη επιβλεπόμενης μάθησης και συνοψίζεται σχηματικά στην παρακάτω εικόνα.



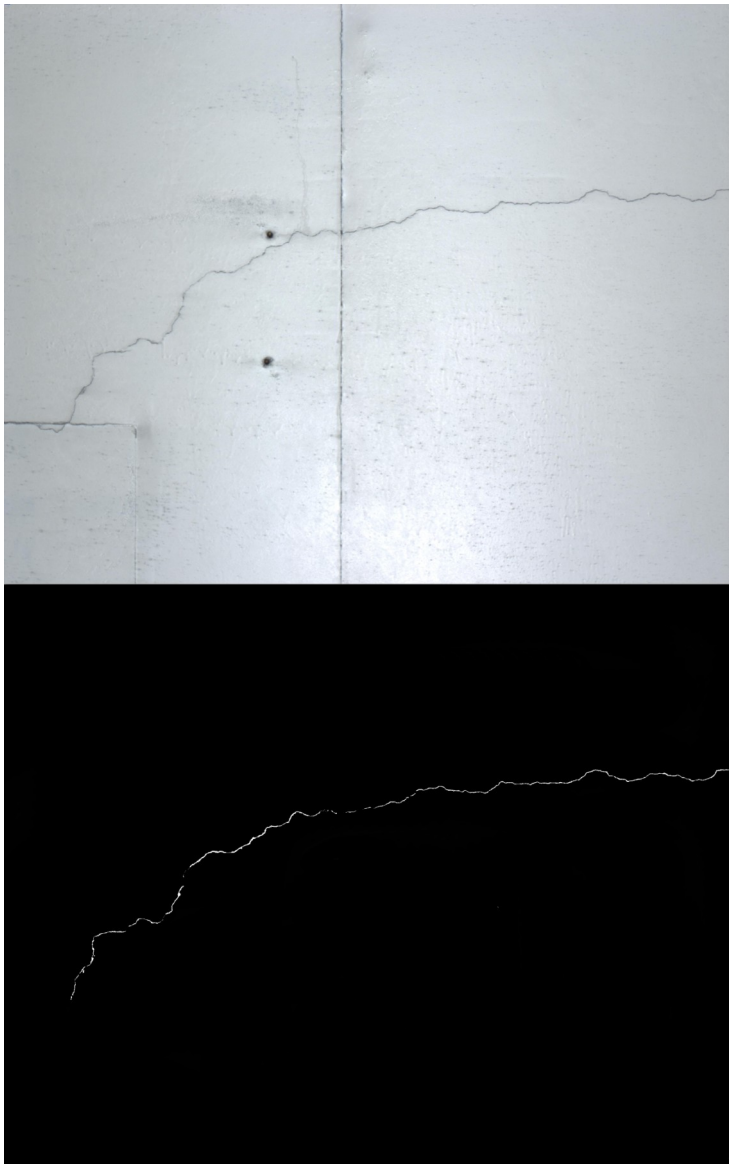
Εικόνα 19: Σχηματική περιγραφή προβλήματος

4.2 Επεξεργασία δεδομένων

α) Περιγραφή δεδομένων

Το σύνολο δεδομένων που χρησιμοποιήθηκε στα πλαίσια της παρούσας εργασίας αποτελείται από 80 έγχρωμες εικόνες, οι οποίες έχουν ληφθεί σε τρεις διαφορετικές σήραγγες. Κατά την λήψη τους, τοποθετήθηκε μονάδα LED στην κορυφή της κάμερας ώστε ο φωτισμός του αντικειμένου να είναι επαρκής. Παρ' όλα αυτά οι συνθήκες σε κάθε σήραγγα διέφεραν με αποτέλεσμα τα δεδομένα να παρουσιάζουν χαρακτηριστική ποικιλομορφία σε σχέση με την υφή της επιφάνειας, τον φωτισμό και την μορφή των ατελειών.

Οι διαστάσεις της πλειονότητας των εικόνων είναι 3376 × 2704 εικονοστοιχεία, περιλαμβάνοντας όμως και μερικές εικόνες μεγέθους 4288 × 2848 εικονοστοιχείων. Χαρακτηριστικό των συγκεκριμένων εικόνων είναι η σπανιότητα των ρωγμών καθώς και η ποικιλομορφία χαρακτηριστικών όπως αναφέρθηκε και παραπάνω.

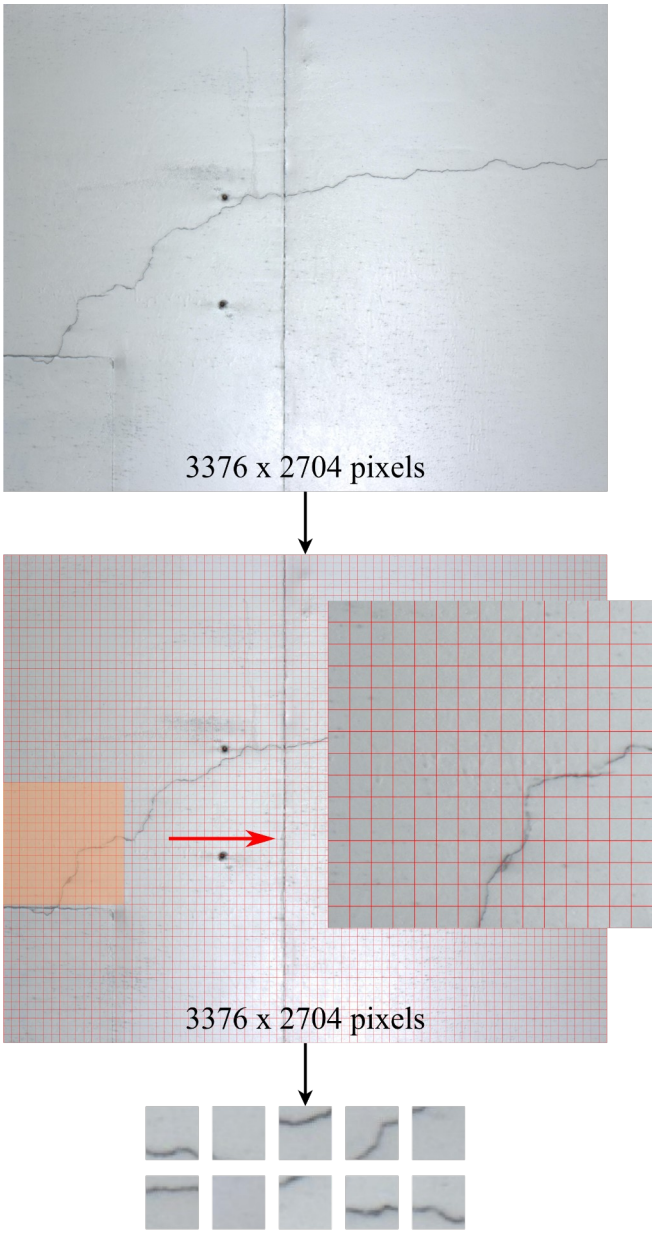


Εικόνα 20: Μία από τις εικόνες του συνόλου δεδομένων και η αντίστοιχη ασπρόμαυρη

Για κάθε μία από τις 80 εικόνες του συνόλου δεδομένων υπάρχει και μια ασπρόμαυρη εικόνα στην οποία οι ρωγμές σημαίνονται με άσπρο χρώμα σε μαύρο φόντο. Αν και όπως αναφέρθηκε σε προηγούμενα κεφάλαια για τη χρήση τεχνικών μη επιβλεπόμενης μάθησης δεν χρειάζονται επισημασμένα δεδομένα, στην περίπτωση της παρούσας εργασίας οι ασπρόμαυρες εικόνες είναι απαραίτητες για να εξασφαλιστεί ότι η εκπαίδευση του μοντέλου θα πραγματοποιηθεί μόνο με εικόνες που απεικονίζουν τη σήραγγα σε υγιή κατάσταση αλλά και για να δημιουργηθεί ένα σύνολο εικόνων ελέγχου με φυσιολογικές και μη εικόνες για την αξιολόγηση της απόδοσης του μοντέλου μετά την εκπαίδευση του.

β) Προετοιμασία δεδομένων για τη μετέπειτα χρήση τους

Οι διαστάσεις των δεδομένων εικόνων όπως αναφέρθηκε παραπάνω αποτελούνται από μεγάλο αριθμό εικονοστοιχείων. Με σκοπό να είναι ευκολότερα διαχειρίσιμες από το μοντέλο επιλέχθηκε να διαμελιστούν σε επιμέρους εικόνες μεγέθους 45 × 45 εικονοστοιχείων. Το μέγεθος αυτό, πέρα από το πλεονέκτημα της εύκολης διαχείρισης από το μοντέλο, είναι αρκετό ώστε να εντοπιστεί κάποιο πιθανό ανώμαλο συμβάν και παράλληλα προσφέρει ομοιομορφία στα δεδομένα αφού η κατάτμηση καταλήγει σε εικόνες ίδιου μεγέθους, ανεξάρτητα από τις διαστάσεις της αρχικής εικόνας. Στη διπλανή εικόνα φαίνεται σχηματικά η κατάτμηση μιας εικόνας σε επιμέρους εικόνες μεγέθους 45 × 45 εικονοστοιχείων.



Εικόνα 21: Παράδειγμα κατάτμησης εικόνας

Η ανάπτυξη κώδικα για τη διαδικασία κατάτμησης αλλά και για τις διαδικασίες που θα περιγραφούν στη συνέχεια του κεφαλαίου πραγματοποιήθηκε σε γλώσσα προγραμματισμού Python και χρησιμοποιήθηκε το περιβάλλον του Google Colaboratory, το οποίο επιτρέπει τη συγγραφή και την εκτέλεση κώδικα μέσω προγράμματος περιήγησης.

```
# import packages
import cv2
import numpy as np
import os
import PIL
from PIL import Image
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
# paths
INPUT_PATH = 'drive/My Drive/diplomatiki_colab/dedomena_unzipped/tunnel-crack-images/'
OUTPUT_PATH = 'drive/My Drive/diplomatiki_colab/dedomena_unzipped/tunnel-crack-images/DataCrop'
```

Ένα πλεονέκτημα του περιβάλλοντος του Google Colaboratory είναι ότι είναι δυνατό να προσπελαστούν αρχεία που βρίσκονται αποθηκευμένα στο Google Drive, καθώς και να αποθηκευτούν νέα αρχεία σε αυτό. Η δυνατότητα αυτή αξιοποιείται παραπάνω αφού φαίνεται να φορτώνονται οι

απαραίτητες για τη συνέχεια βιβλιοθήκες και έπειτα να ορίζεται η διαδρομή (path) στην οποία βρίσκονται αποθηκευμένα τα δεδομένα, καθώς και η διαδρομή στην οποία θα αποθηκευτούν οι νέες εικόνες μετά την κατάτμηση.

```
# size
h=45
# list directories
_, subCategoryDirectoriesInputSet, _ = next(os.walk(INPUT_PATH))
# list images in each directory
SubcategoryFiles = [None] * len(subCategoryDirectoriesInputSet)
for directory in range(0, len(subCategoryDirectoriesInputSet)):
  tmpTrainOrTestPath = INPUT_PATH + subCategoryDirectoriesInputSet[directory]
  _, _, SubcategoryFiles[directory] = next(os.walk(tmpTrainOrTestPath))
```

Αφού ορίστηκε η διαδρομή στην οποία βρίσκονται τα δεδομένα, αρχικά ορίζεται μια μεταβλητή h η οποία ισούται με την επιθυμητή διάσταση των εικόνων που θα παραχθούν. Στη συνέχεια, δημιουργήθηκε μια λίστα με τους φακέλους που βρίσκονται στην ορισμένη διαδρομή. Στη συγκεκριμένη περίπτωση βρέθηκαν δύο φάκελοι: ο φάκελος που περιλαμβάνει τις έγχρωμες εικόνες και ο φάκελος που περιλαμβάνει τις ασπρόμαυρες επισημασμένες εικόνες (annotated). Τα στοιχεία της λίστας αυτής χρησιμοποιούνται για να δημιουργηθεί με πιο αυτοματοποιημένο τρόπο στη συνέχεια ένας πίνακας που περιλαμβάνει τις ονομασίες όλων των εικόνων που περιλαμβάνονται στους δύο φακέλους. Αυτό επιτυγχάνεται μέσω μιας επαναληπτικής διαδικασίας η οποία επισυνάπτει διαδοχικά το όνομα του εκάστοτε φακέλου που εντοπίστηκε στο τέλος της διαδρομής που είχε ήδη οριστεί και διαβάζει τα ονόματα των αρχείων που περιλαμβάνονται στη νέα αυτή διαδρομή.

```
def binarize(image, threshold):
  output_image = image.convert("L")
  for i in range(image.width):
      for j in range(image.height):
          if output_image.getpixel((i,j)) < threshold:
                output_image.putpixel((i,j), 0)
          else:
                output_image.putpixel((i, j), 255)
  return output_image
```

Παρατηρώντας τις annotated εικόνες φαίνεται ότι δεν περιλαμβάνουν μόνο τις τιμές φωτεινότητας 0 και 255 αλλά και κάποιες ενδιάμεσες τιμές. Με σκοπό να διευκολυνθεί στη συνέχεια ο διαχωρισμός των εικόνων σε αυτές που περιλαμβάνουν ρωγμή και στις φυσιολογικές, θεωρήθηκε απαραίτητο να εφαρμοστεί ένα κατώφλι (threshold). Έτσι, ορίστηκε η παραπάνω συνάρτηση η οποία δέχεται σαν ορίσματα μια εικόνα και μια τιμή για το κατώφλι και ελέγχει αν η τιμή του κάθε εικονοστοιχείου της εικόνας είναι μεγαλύτερη ή μικρότερη από το κατώφλι. Αν η τιμή είναι μικρότερη,

φαίνεται ότι το εικονοστοιχείο πλησιάζει περισσότερο στο μαύρο, οπότε η τιμή του αντικαθίσταται με την τιμή 0. Αντίθετα, αν η τιμή του εικονοστοιχείου είναι μεγαλύτερη από το κατώφλι, φαίνεται ότι το εικονοστοιχείο αυτό διαφοροποιείται σημαντικά από το μαύρο, οπότε η τιμή αντικαθίσταται με το 255 και είναι πλέον άσπρο.

```
mask_tuple = (0, 0, h, h)
count = 0
for i in range(len(SubcategoryFiles[0])):
  RGB_path = INPUT_PATH + subCategoryDirectoriesInputSet[0] + '/' +
  SubcategoryFiles[0][i]
  RGB_img = cv2.imread(RGB_path)

  if SubcategoryFiles[0][i].endswith(".jpg"):
    Annotated_path = INPUT_PATH + subCategoryDirectoriesInputSet[1] + '/' +
    SubcategoryFiles[0][i][:-4] + "_BG.jpg"
  else:
    Annotated_path = INPUT_PATH + subCategoryDirectoriesInputSet[1] + '/' +
    SubcategoryFiles[0][i][:-4] + "GT_BG.JPG"
  Annotated_img = cv2.imread(Annotated_path)

  if RGB_img is None:
    print('\n .. unable to load RGB image')
  elif Annotated_img is None:
    print('\n .. unable to load Annotated image')
  else:
    print('\n .. images successfully loaded.')
    #number of cropped images
    width = RGB_img.shape[1]
    height = RGB_img.shape[0]
    number_x = int(width/h)
    number_y = int(height/h)

    #cropping images
    count = count + 1
    print("Counter: " + str(count))
    print("... working on RGB image " + SubcategoryFiles[0][i])
    RGB_img = Image.fromarray(RGB_img)
    Annotated_img = Image.fromarray(Annotated_img)

    clear_count = 0
```

```
cracked_count = 0
for j in range(0, ((number_x)*h), h):
  for k in range(0, ((number_y)*h), h):
    RGB_cropped = RGB_img.crop((j, k, j+h, k+h))
    Annot_cropped = Annotated_img.crop((j,k,j+h,k+h))

    annot_gray_version = Annot_cropped.convert('L')
    annot_binary = binarize(annot_gray_version, 15)
    # convert image to array
    annot_binary = np.asarray(annot_binary)
    RGB_cropped = np.asarray(RGB_cropped)
    if cv2.countNonZero(annot_binary) == 0:
      clear_count = clear_count + 1
      RGB_output_path = OUTPUT_PATH + '/RGB/Clear/' + SubcategoryFiles[0][i][:-
      4] + '_clear_' + str(clear_count) + '.jpg'
      cv2.imwrite(RGB_output_path, RGB_cropped)
    else:
      cracked_count = cracked_count + 1
      RGB_output_path = OUTPUT_PATH + '/RGB/Cracked/'+ SubcategoryFiles[0][i]
      [:-4] + '_cracked_' + str(cracked_count) + '.jpg'
      annot_output_path = OUTPUT_PATH + '/Annotated/Cracked/' +
      SubcategoryFiles[0][i][:-4] + '_cracked_BG_' + str(cracked_count) +
      '.jpg'
      cv2.imwrite(RGB_output_path, RGB_cropped)
      cv2.imwrite(annot_output_path, annot_binary)

print(".... Saved " + str(clear_count) + " clear images")
print(".... Saved " + str(cracked_count) + " cracked images")
```

Η κατάτμηση των εικόνων σε επιμέρους εικόνες μεγέθους 45 × 45 εικονοστοιχείων επιτυγχάνεται μέσω μιας επαναληπτικής διαδικασίας. Έχοντας ήδη σε μορφή πίνακα τα ονόματα όλων των εικόνων, σε κάθε επανάληψη διαβάζεται μια έγχρωμη εικόνα και στη συνέχεια εντοπίζεται και διαβάζεται η αντίστοιχη annotated ενημερώνοντας την αντίστοιχη διαδρομή. Αφού ελεγχθεί ότι οι δύο εικόνες έχουν βρεθεί και ανοιχθεί, μέσω επιμέρους επαναληπτικής διαδικασίας διατρέχεται όλη η εικόνα κατά μήκος και κατά πλάτος με βήμα όσο η μεταβλητή h που ορίστηκε παραπάνω, μέσω των δεικτών j και k οι οποίοι, με αφετηρία το 0, φτάνουν μέχρι το τελευταίο ακέραιο πολλαπλάσιο του 45 εντός της εικόνας. Επειδή δεν είναι εξασφαλισμένο ότι οι διαστάσεις της κάθε εικόνας θα είναι ακέραιο πολλαπλάσιο του 45, η εικόνα διατρέχεται μέχρι το τελευταίο εικονοστοιχείο στο οποίο ισχύει ότι η αντίστοιχη διάσταση είναι πολλαπλάσιο του 45, περικόπτοντας έτσι μερικά εικονοστοιχεία στα άκρα.

Αφού διαχωριστεί το τμήμα της έγχρωμης εικόνας μεγέθους 45 × 45 εικονοστοιχείων και το αντίστοιχο τμήμα της annotated, πρέπει να ελεγχθεί αν η εικόνα στο σημείο αυτό περιλαμβάνει ρωγμή. Αυτό εξετάζεται μέσω του ελέγχου αν η αντίστοιχη annotated στη συγκεκριμένη περιοχή περιλαμβάνει

εικονοστοιχείο άσπρου χρώματος. Μετά την εφαρμογή της συνάρτησης “binarize” στην annotated εικόνα με κατώφλι την τιμή 15, αν το πλήθος των μη μηδενικών εικονοστοιχείων είναι 0, φαίνεται ότι όλα τα εικονοστοιχεία του παραθύρου 45 × 45 εικονοστοιχείων που διαχωρίστηκε έχουν τιμή φωτεινότητας 0. Το γεγονός αυτό δείχνει ότι το αντίστοιχο κομμάτι 45 × 45 εικονοστοιχείων της έγχρωμης εικόνας δεν περιλαμβάνει ρωγμή, οπότε αποθηκεύεται σε μορφή jpg σε έναν φάκελο με την ονομασία “Clear” με το όνομα της έγχρωμης εικόνας από την οποία προέρχεται, τον χαρακτηρισμό clear και έναν αύξων αριθμό. Ομοίως στην περίπτωση που εντοπιστεί μη μηδενικό εικονοστοιχείο στο τμήμα της annotated εικόνας, θεωρείται ότι η αντίστοιχη έγχρωμη εικόνα περιλαμβάνει ρωγμή και έτσι αποθηκεύεται σε μορφή jpg και πάλι σε έναν φάκελο με την ονομασία “Cracked” με το όνομα της έγχρωμης εικόνας από την οποία προέρχεται, τον χαρακτηρισμό cracked και έναν αύξων αριθμό.

Μετά την ολοκλήρωση της διαδικασίας αυτής, προέκυψαν συνολικά 358232 εικόνες μεγέθους 45 × 45 εικονοστοιχείων οι οποίες δεν περιλαμβάνουν ρωγμή και 6223 εικόνες ίδιου μεγέθους οι οποίες απεικονίζουν ρηγματωμένο τοίχωμα. Από τις φυσιολογικές εικόνες επιλέχθηκαν 20000 εικόνες με τυχαίο τρόπο οι οποίες θα χρησιμοποιηθούν στη συνέχεια για αξιολόγηση της εκτίμησης του μοντέλου σε συνδυασμό με τις ρηγματωμένες. Οι υπόλοιπες φυσιολογικές εικόνες αποτελούν το σύνολο δεδομένων εκπαίδευσης για τον αυτόματο κωδικοποιητή που δημιουργήθηκε στη συνέχεια.

Παρατηρώντας τις εικόνες φαίνεται ότι οι φυσιολογικές εικόνες που θα χρησιμοποιηθούν για την εκπαίδευση, αν και δεν απεικονίζουν ρωγμές, περιλαμβάνουν πληθώρα ανώμαλων συμβάντων όπως ενώσεις ή άλλου είδους ασυνέχειες στις οποίες το μοντέλο δεν είναι επιθυμητό να προσαρμοστεί. Αντίθετα, το μοντέλο χρειάζεται να μάθει μια κατάσταση ομοιομορφίας για να αναγνωρίσει τις αποκλίσεις από αυτήν. Έτσι, σαν σύνολο δεδομένων εκπαίδευσης, επιλέχθηκε ένα υποσύνολο 1000 εικόνων οι οποίες απεικονίζουν την επιθυμητή ομοιομορφία και το πλήθος τους είναι ικανό για να εκπαιδευτεί το μοντέλο.

Όπως είναι λογικό, όμοια προβλήματα παρατηρήθηκαν και στις 20000 φυσιολογικές εικόνες που κρατήθηκαν σαν σύνολο δεδομένων ελέγχου. Οι εικόνες αυτές περιλάμβαναν πολλά χαρακτηριστικά τα οποία το μοντέλο θα ήταν δυνατό να αναγνωρίσει σαν ανώμαλα συμβάντα. Συνεπώς θεωρήθηκε απαραίτητο να κρατηθούν μόνο 1415 φυσιολογικές εικόνες για έλεγχο, με σκοπό να δοθεί η ευκαιρία στο μοντέλο να αναγνωρίσει ρωγμές, χωρίς να αποπροσανατολίζεται από άλλου είδους ανώμαλα συμβάντα.

Αντίθετα, πολλές από τις εικόνες που αποθηκεύτηκαν σαν ρηγματωμένες για να αξιοποιηθούν για τον έλεγχο του μοντέλου παρουσίαζαν ιδιαίτερη ομοιομορφία αφού κάποιες ρωγμές ήταν δύσκολα ορατές. Το γεγονός αυτό, σε συνδυασμό με την ποικιλομορφία των φυσιολογικών εικόνων, οδηγούσε το μοντέλο σε λανθασμένα συμπεράσματα. Γι αυτό το λόγο επιλέχθηκαν 2317 εικόνες οι οποίες περιλαμβάνουν εμφανείς ρωγμές με σκοπό το μοντέλο να καταφέρει να τις εντοπίσει.

Οπότε, αξιοποιήθηκαν συνολικά 1000 φυσιολογικές εικόνες για την εκπαίδευση του μοντέλου και 1415 φυσιολογικές σε συνδυασμό με 2317 ρηγματωμένες για τον έλεγχο της απόδοσής του. Οι εικόνες αυτές αποθηκεύτηκαν στο Google Drive σε συμπιεσμένους φακέλους.

4.3 Ορισμός και εκπαίδευση μοντέλου

Στο προηγούμενο κεφάλαιο έγινε περιγραφή του μοντέλου και των στρωμάτων που περιλαμβάνει, καθώς και του τρόπου λειτουργίας των στρωμάτων αυτών. Στο κεφάλαιο αυτό θα παρουσιαστεί ο κώδικας με τον οποίο υλοποιήθηκε το μοντέλο και η διαδικασία εκπαίδευσής του.

```
# import packages
import numpy as np
import cv2
import tensorflow as tf
from tensorflow import keras
import os
from tensorflow.keras.layers import Conv2D, LSTM, Conv2DTranspose, Reshape, Permute
from tensorflow.keras.models import Sequential
from zipfile import ZipFile
from PIL import Image
from google.colab.patches import cv2_imshow
# connect to google drive
from google.colab import drive
drive.mount('/content/drive')
# define input path
input_path = 'drive/My Drive/diplomatiki_colab/dedomena_unzipped/tunnel-crack-
images/DataCrop/RGB/'
```

Αρχικά απαιτείται να φορτωθούν οι απαραίτητες για τη συνέχεια βιβλιοθήκες και να πραγματοποιηθεί η σύνδεση με το Google Drive στο οποίο είναι αποθηκευμένες οι εικόνες που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου στη συνέχεια. Επίσης, καθορίζεται η διαδρομή (path) στην οποία βρίσκεται ο φάκελος με τις αποθηκευμένες εικόνες από το προηγούμενο στάδιο.

Αφού καθοριστούν αυτά, είναι απαραίτητο να διαβαστούν τα δεδομένα εκπαίδευσης από τον φάκελο.

```
zf = ZipFile(input_path + 'Clear_for_training2.zip', 'r')
file_list = ZipFile.namelist(zf)
clear_images = []
i=0
for f in file_list:
    print(f)
    ifile = zf.open(f)
    if ifile is None:
        print('File can not be loaded')
        break
    temp = Image.open(ifile)
    temp = temp.convert('L')
    temp = np.array(temp)
    #NORMALIZE
```

```
    temp = temp/255
    clear_images.append(temp)
    if clear_images[i] is None:
        print('Image can not be loaded')
        break
    i = i + 1
```

Οι εικόνες που αποτελούν τα δεδομένα εκπαίδευσης βρίσκονται αποθηκευμένες σε συμπιεσμένο φάκελο, ο οποίος διαβάζεται και δημιουργείται μια λίστα με τα ονόματα των εικόνων που περιλαμβάνει. Μέσω μιας επαναληπτικής διαδικασίας, η κάθε εικόνα ανοίγεται και ελέγχεται αν έχει ανοιχθεί σωστά. Στη συνέχεια, η εικόνα, η οποία μέχρι και αυτό το στάδιο είναι έγχρωμη, μετατρέπεται σε τόνους του γκρι, αφού για να εντοπιστεί κάποια ασυνέχεια δεν είναι απαραίτητη η ύπαρξη χρωμάτων. Έπειτα, η εικόνα περνάει από μια διαδικασία κανονικοποίησης η οποία επιτυγχάνεται διαιρώντας την τιμή κάθε εικονοστοιχείου της με τον αριθμό 255. Με αυτό τον τρόπο όλες οι τιμές του πίνακα που αποτελεί την εικόνα βρίσκονται στο διάστημα [0, 1] χωρίς όμως να χάνονται πληροφορίες ή να διαστρεβλώνονται οι διαφορές στα εύρη τιμών. Αφού ολοκληρωθεί και η κανονικοποίηση, η εκάστοτε εικόνα μπαίνει στη λίστα “clear_images” με σκοπό το σύνολο των δεδομένων εκπαίδευσης να είναι οργανωμένο και εύκολα διαχειρίσιμο.

```
# define model
def autoencoder():
  model = Sequential()


   model.add(Conv2D(filters = 128, kernel_size = (5,5), strides = (2,2), padding =
  'valid', input_shape = (45,45,1), activation = 'tanh', data_format =
  'channels_last'))
   model.add(Conv2D(filters = 64, kernel_size = (5,5), strides = (1,1), padding =
  'valid', activation = 'tanh', data_format = 'channels_last'))
   model.add(Conv2D(filters = 64, kernel_size = (3,3), strides = (1,1), padding =
  'same', activation = 'tanh', data_format = 'channels_last'))


  model.add(Reshape((-1,64,)))
  model.add(Permute((2,1), input_shape=(-1, 289, 64)))


  model.add(LSTM(289, dropout=0.4, recurrent_dropout=0.3, return_sequences=True))
  model.add(LSTM(289, dropout=0.3, return_sequences=True))
  model.add(LSTM(289, return_sequences=True, dropout=0.5))


  model.add(Permute((2,1), input_shape=(-1,64, 289)))
  model.add(Reshape((-1,17,64)))
```

```
    model.add(Conv2DTranspose(filters = 64, kernel_size = (3,3), strides = (1,1),
    padding = 'same', input_shape = (45,45,1), activation = 'tanh'))
    model.add(Conv2DTranspose(filters = 128, kernel_size = (5,5), strides = (1,1),
    padding = 'valid', activation = 'tanh'))
    model.add(Conv2DTranspose(filters = 1, kernel_size = (5,5), strides = (2,2),
    padding = 'valid', activation = 'tanh'))

    model.compile(optimizer='adam',loss='mean_squared_error',metrics=[tf.keras.metric
    s.MeanSquaredError()])
    return model


model1 = autoencoder()
model1.summary()
```

Όπως φαίνεται στο παραπάνω τμήμα κώδικα, ο αυτόματος κωδικοποιητής ορίστηκε σαν μία συνάρτηση με το όνομα "autoencoder" η οποία όταν καλείται επιστρέφει το μοντέλο. Στο εσωτερικό της συνάρτησης διακρίνονται τα στρώματα που αναφέρθηκαν στο προηγούμενο κεφάλαιο, δηλαδή τα στρώματα που εκτελούν συνέλιξη, LSTM και αντίστροφη συνέλιξη. Παράλληλα ορίζονται οι παράμετροι που απαιτούνται για το κάθε στρώμα όπως ο αριθμός των φίλτρων, το μέγεθος του φίλτρου, το stride, το padding και η συνάρτηση ενεργοποίησης. Επιπλέον ορίζεται ο τρόπος με τον οποίο θα πραγματοποιηθεί η βελτιστοποίηση των βαρών του μοντέλου κατά την εκπαίδευσή του, οποίος στη συγκεκριμένη περίπτωση είναι ο αλγόριθμος Adam (adaptive moment estimation). Ο αλγόριθμος αυτός είναι απλός στην υλοποίηση, υπολογιστικά αποδοτικός, έχει μικρές απαιτήσεις μνήμης και είναι κατάλληλος για προβλήματα με μεγάλες απαιτήσεις δεδομένων και πολλές παραμέτρους [17]. Για να υπάρχει μια εκτίμηση της ποιότητας της αναπαραγωγής της εισόδου στην έξοδο, κατά τη διάρκεια της εκπαίδευσης υπολογίζεται και το μέσο τετραγωνικό σφάλμα.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 21, 21, 128)	3328
conv2d_1 (Conv2D)	(None, 17, 17, 64)	204864
conv2d_2 (Conv2D)	(None, 17, 17, 64)	36928
reshape (Reshape)	(None, 289, 64)	0
permute (Permute)	(None, 64, 289)	0
lstm (LSTM)	(None, 64, 289)	669324
lstm_1 (LSTM)	(None, 64, 289)	669324
lstm_2 (LSTM)	(None, 64, 289)	669324
permute_1 (Permute)	(None, 289, 64)	0
reshape_1 (Reshape)	(None, 17, 17, 64)	0
conv2d_transpose (Conv2DTran	(None, 17, 17, 64)	36928
conv2d_transpose_1 (Conv2DTr	(None, 21, 21, 128)	204928
conv2d_transpose_2 (Conv2DTr	(None, 45, 45, 1)	3201

Total params: 2,498,149
Trainable params: 2,498,149
Non-trainable params: 0

Εικόνα 22: Περίληψη μοντέλου

Η συνάρτηση "autoencoder" καλείται στη συνέχεια και δημιουργείται το μοντέλο "model1". Η μορφή του μοντέλου φαίνεται στην Εικόνα 11 αλλά και στην περίληψη που φαίνεται στην Εικόνα 22 στην οποία φαίνονται όλα τα στρώματα, οι διαστάσεις της εξόδου του κάθε στρώματος καθώς και οι παράμετροι που χρειάζεται να προσδιορίσει το μοντέλο μέσω της εκπαίδευσής του.

```
# images to array
X_train = np.array(clear_images)


# add number of channels
X_train = np.reshape(X_train, (1000, 45, 45, 1))
```

Το τελευταίο βήμα πριν την εκπαίδευση του μοντέλου είναι η μετατροπή της λίστας "clear_images", η οποία περιλαμβάνει τα δεδομένα εκπαίδευσης, σε numpy array. Numpy array είναι ένας πίνακας ο οποίος, σε αντίθεση με μία απλή λίστα, μπορεί να διαχειριστεί μαθηματικές πράξεις ενώ είναι και αποδοτικότερος στην αποθήκευση μεγάλων ποσοτήτων δεδομένων, αφού τα αποθηκεύει με συμπαγή τρόπο. [43]

Όπως ορίστηκε, οι διαστάσεις της κάθε εικόνας που δέχεται σαν είσοδο το μοντέλο πρέπει να είναι (45, 45, 1), δηλαδή (rows, columns, channels). Αν οι εικόνες είχαν παραμείνει έγχρωμες οι διαστάσεις τους θα ήταν (45, 45, 3) αφού η κάθε έγχρωμη εικόνα περιλαμβάνει κανάλια RGB για να συντεθούν τα χρώματά της. Από την άλλη πλευρά, οι διαστάσεις που θεωρείται ότι έχει μια ασπρόμαυρη εικόνα είναι (45, 45), δηλαδή παραλείπεται ο αριθμός των καναλιών αφού στη συγκεκριμένη περίπτωση είναι 1 και αφορά τιμές φωτεινότητας. Για το λόγο αυτό γίνεται αναδιαμόρφωση (reshape) του numpy array, το οποίο ονομάζεται "X_train" στη συγκεκριμένη περίπτωση, ώστε να περιλαμβάνει και τον αριθμό των καναλιών. Τελικά, οι διαστάσεις του "X_train" είναι (1000, 45, 45, 1) με το 1000 να αντιστοιχεί στο πλήθος των εικόνων, τις διαστάσεις της εικόνας να ακολουθούν και τέλος τον αριθμό 1 που δείχνει το πλήθος των καναλιών.

```
# fit
model1.fit(X_train, X_train, batch_size=10, epochs=5)
```

```
Epoch 1/5
100/100 [==============================] - 17s 113ms/step - loss: 0.0326 - mean_squared_error: 0.0326
Epoch 2/5
100/100 [==============================] - 11s 113ms/step - loss: 0.0127 - mean_squared_error: 0.0127
Epoch 3/5
100/100 [==============================] - 11s 114ms/step - loss: 0.0120 - mean_squared_error: 0.0120
Epoch 4/5
100/100 [==============================] - 11s 112ms/step - loss: 0.0175 - mean_squared_error: 0.0175
Epoch 5/5
100/100 [==============================] - 11s 112ms/step - loss: 0.0114 - mean_squared_error: 0.0114
<tensorflow.python.keras.callbacks.History at 0x7f01e2578950>
```

Εικόνα 23: Αποτελέσματα εκπαίδευσης μοντέλου

Μέσω της εντολής "fit" το μοντέλο εκπαιδεύεται, δηλαδή προσαρμόζει τις παραμέτρους του στα δεδομένα εκπαίδευσης. Γενικά, η μορφή της εντολής είναι (x, y, batch_size, epochs) με όπου x τα δεδομένα εκπαίδευσης τα οποία έχουν τη μορφή numpy array ενώ όπου y οι ετικέτες που δείχνουν τον στόχο που προσπαθεί το μοντέλο να πετύχει μέσω της εκπαίδευσής του. Στην περίπτωση του αυτόματου κωδικοποιητή είναι επιθυμητό το μοντέλο να μάθει να αναπαράγει την είσοδό του στην έξοδό του. Γι αυτό το λόγο όπου x εισάγεται το "X_train" αλλά και όπου y επίσης εισάγεται το

“X_train”. Έτσι, το μοντέλο ελέγχει τον βαθμό με τον οποίο επιτυγχάνει να αναπαράξει τα δεδομένα, αφού ο στόχος του είναι τα ίδια τα δεδομένα. Η παράμετρος “batch_size” δείχνει το πλήθος των στοιχείων του συνόλου δεδομένων εκπαίδευσης που εισάγονται στο μοντέλο και στη συνέχεια ενημερώνονται τα βάρη του μοντέλου. Δηλαδή στη συγκεκριμένη περίπτωση το μοντέλο εκπαιδεύεται με 10 εικόνες και ενημερώνει τις παραμέτρους του πριν εισαχθούν σε αυτό οι επόμενες 10 εικόνες. Θέτοντας έναν αριθμό “batch_size” μειώνονται οι απαιτήσεις μνήμης για την εκπαίδευση του μοντέλου. Τέλος, ορίζεται και ο αριθμός των εποχών (epochs), ο οποίος καθορίζει το πλήθος των επαναλήψεων στο δεδομένο σύνολο x και y, δηλαδή πόσες φορές το μοντέλο θα αντιμετωπίσει το σύνολο των δεδομένων εκπαίδευσης. Στη συγκεκριμένη περίπτωση το μοντέλο εκπαιδεύτηκε για 5 εποχές, αφού δεν φαίνεται να υπάρχει σημαντική περαιτέρω μείωση του μέσου τετραγωνικού σφάλματος από εποχή σε εποχή. [44]

```
# save model
save_path = input_path + 'model1_normalized.h5'
model1.save(save_path)

# load trained model
save_path = input_path + 'model1_normalized.h5'
model1 = keras.models.load_model(save_path)
```

Ένα βήμα το οποίο δεν απαιτείται για τη συνέχεια αλλά είναι ιδιαίτερα βοηθητικό είναι η αποθήκευση του εκπαιδευμένου μοντέλου στο Google Drive σε μορφή h5, αφού έτσι δίνεται η δυνατότητα να φορτωθεί και πάλι οποιαδήποτε στιγμή, χωρίς να απαιτείται να επαναληφθεί η εκπαίδευσή του.

4.4 Προσδιορισμός κατωφλίου

Σκοπός της παρούσας εργασίας είναι η αυτόματη αναγνώριση ρωγμών χρησιμοποιώντας το μοντέλο που δημιουργήθηκε και εκπαιδεύτηκε στην προηγούμενη ενότητα. Αυτό επιτυγχάνεται με τη χρήση μιας συνάρτησης απωλειών και τον ορισμό ενός κατωφλίου. Η συνάρτηση απωλειών που χρησιμοποιείται στη συγκεκριμένη περίπτωση είναι το μέσο τετραγωνικό σφάλμα, το οποίο ποσοτικοποιεί τον βαθμό επιτυχίας στην αναπαραγωγή της εισόδου του μοντέλου στην έξοδό του. Υπενθυμίζεται ότι το μοντέλο είναι εκπαιδευμένο με εικόνες που δεν περιλαμβάνουν ρωγμή ή κάποιο άλλο ανώμαλο συμβάν με σκοπό να μάθει να αναπαράγει τη φυσιολογική κατάσταση. Αν ζητηθεί στο μοντέλο να αναπαράξει μια εικόνα που περιλαμβάνει κάποιο ανώμαλο συμβάν αναμένεται να προκύψει μεγαλύτερο μέσο τετραγωνικό σφάλμα σε σχέση με αυτό που θα προέκυπτε αν εισαγόταν στο μοντέλο μια εικόνα που απεικονίζει τη φυσιολογική κατάσταση. Έτσι, είναι επιθυμητό να προσδιοριστεί μια τιμή του μέσου τετραγωνικού σφάλματος (κατώφλι), η οποία αν ξεπεραστεί κατά την αναπαραγωγή κάποιας εικόνας θεωρείται ότι η εικόνα περιλαμβάνει ρωγμή ή κάποιο άλλο ανώμαλο συμβάν.

```
def mse(im1, im2):
  diff = im1 - im2
```

```
  squared_diff = diff**2
  sum = squared_diff.sum()
  error = np.sqrt(sum)
  mean_error = error/(im1.shape[0]*im2.shape[1])
  return mean_error
```

Για να προσδιοριστεί η τιμή του κατωφλίου αρχικά απαιτείται να δημιουργηθεί μια συνάρτηση η οποία θα υπολογίζει το μέσο τετραγωνικό σφάλμα. Η συνάρτηση αυτή ονομάστηκε “mse” και δέχεται σαν ορίσματα δύο εικόνες, δηλαδή δύο πίνακες. Το μέσο τετραγωνικό σφάλμα υπολογίζεται από την εξίσωση (3.1) και όπως φαίνεται αρχικά υπολογίζεται η διαφορά του κάθε εικονοστοιχείου της αρχικής εικόνας με το κάθε εικονοστοιχείο της εικόνας που αναπαράχθηκε από το μοντέλο. Η κάθε διαφορά στη συνέχεια υψώνεται στο τετράγωνο, όλες οι επιμέρους τιμές αθροίζονται και προκύπτει μια ενιαία τιμή για την κάθε εικόνα. Έπειτα, υπολογίζεται η τετραγωνική ρίζα της τιμής αυτής και τέλος διαιρείται με το γινόμενο των διαστάσεων της εικόνας, δηλαδή με το πλήθος των εικονοστοιχείων της. Έτσι, είναι εμφανές ότι προκύπτει μία ενιαία τιμή μέσου τετραγωνικού σφάλματος για κάθε αναπαραγωγή.

Με σκοπό να φανεί αν το μοντέλο επιτυγχάνει τον διαχωρισμό φυσιολογικών και μη φυσιολογικών εικόνων, επιλέγονται δειγματοληπτικά 10 φυσιολογικές εικόνες και 10 μη φυσιολογικές και ζητείται από το μοντέλο να πραγματοποιήσει πρόβλεψη για αυτές. Ο τρόπος που οι εικόνες αυτές διαβάζονται και προετοιμάζονται για την είσοδο στο μοντέλο είναι όμοιος με τον τρόπο που ακολουθήθηκε στην περίπτωση των δεδομένων εκπαίδευσης.

```
# test images mixed
clear_test_path = 'drive/My Drive/diplomatiki_colab/dedomena_unzipped/tunnel-crack-images/DataCrop/RGB/10_clear_for_testing.zip'
cracked_test_path = input_path + '10_cracked_for_testing.zip'

# load 10 clear images for testing
zf2 = ZipFile(clear_test_path, 'r')
file_list_clear_test = ZipFile.namelist(zf2)
test_images = []
i=0
for f in file_list_clear_test:
    print(f)
    ifile = zf2.open(f)
    if ifile is None:
        print('File can not be loaded')
        break

    temp = Image.open(ifile)
    temp = temp.convert('L')
```

```
    temp = np.array(temp)
    #NORMALIZE
    temp = temp/255
    test_images.append(temp)
    if test_images[i] is None:
        print('Image can not be loaded')
        break
    i = i + 1

# load 10 cracked images
zf3 = ZipFile(cracked_test_path, 'r')
file_list_cracked_test = ZipFile.namelist(zf3)
i=0
for f in file_list_cracked_test:
    print(f)
    ifile = zf3.open(f)
    if ifile is None:
        print('File can not be loaded')
        break

    temp = Image.open(ifile)
    temp = temp.convert('L')
    temp = np.array(temp)
    #NORMALIZE
    temp = temp/255
    test_images.append(temp)
    if test_images[i] is None:
        print('Image can not be loaded')
        break
    i = i + 1
```

Συγκεκριμένα δημιουργείται μία λίστα με εικόνες η οποία ονομάζεται “test_images”. Η λίστα αυτή τελικά αποτελείται από 20 εικόνες, οι οποίες μετατρέπονται σε ασπρόμαυρες και κανονικοποιούνται, με τις πρώτες 10 (0 έως 9) να αντιστοιχούν σε φυσιολογικές εικόνες και τις υπόλοιπες (10 έως 19) να αντιστοιχούν σε μη φυσιολογικές εικόνες.

```
test_images = np.array(test_images)
test_images = np.reshape(test_images, (20, 45, 45, 1))
```

Στη συνέχεια, η λίστα αυτή μετατρέπεται σε numpy array και διαμορφώνεται ώστε να έχει διαστάσεις (20, 45, 45, 1), με όμοια λογική με την περίπτωση των δεδομένων εκπαίδευσης. Από τις διαστάσεις αυτές φαίνεται ότι το numpy array αποτελείται από 20 εικόνες μεγέθους 45 × 45 εικονοστοιχείων οι οποίες είναι ασπρόμαυρες.

```
prediction = model1.predict(test_images)
losses = []
for i in range(20):
  losses.append(mse(test_images[i], prediction[i]))
```

Μέσω της εντολής “predict” οι 20 αυτές εικόνες, οι οποίες βρίσκονται με τη μορφή πίνακα στο numpy array “test_images”, εισάγονται στο εκπαιδευμένο μοντέλο και αυτό προσπαθεί να τις αναπαράξει στην έξοδό του. Έτσι, δημιουργείται ένας πίνακας “prediction”, ο οποίος περιλαμβάνει τις προβλέψεις του μοντέλου για κάθε επιμέρους εικόνα. Στη συνέχεια δημιουργείται μια λίστα “losses” η οποία περιλαμβάνει το μέσο τετραγωνικό σφάλμα που υπολογίζεται αν εισαχθούν στη συνάρτηση “mse” η αρχική εικόνα και η αντίστοιχη πρόβλεψη του μοντέλου. Η τιμή του μέσου τετραγωνικού σφάλματος για κάθε εικόνα φαίνεται παρακάτω στην Εικόνα 24.

```
image 0 -> error: 0.0016201485242227202
image 1 -> error: 0.001626808083077765 4
image 2 -> error: 0.0015513077769545337
image 3 -> error: 0.0019474793775229044
image 4 -> error: 0.0020041206443677296
image 5 -> error: 0.0019202477949436172
image 6 -> error: 0.001833169487138034
image 7 -> error: 0.001965774665585238
image 8 -> error: 0.002011717683656676
image 9 -> error: 0.0019252865757123827
image 10 -> error: 0.0034007738395468758
image 11 -> error: 0.003544255743941867
image 12 -> error: 0.0036117060899074657
image 13 -> error: 0.003852928463587508
image 14 -> error: 0.0029699998972474127
image 15 -> error: 0.003671247442468411
image 16 -> error: 0.003481763133917342
image 17 -> error: 0.0033501426603863224
image 18 -> error: 0.0035075430464864606
image 19 -> error: 0.003466598830779402
```

Εικόνα 24: Τιμές μέσου τετραγωνικού σφάλματος για το δείγμα 20 εικόνων

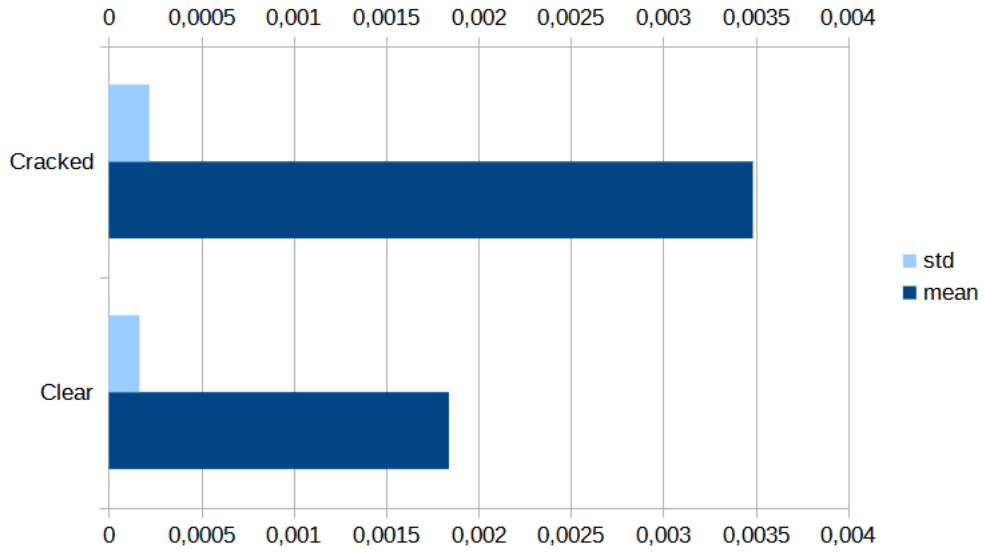
Με μία πρώτη ματιά στις τιμές αυτές φαίνεται ότι οι τιμές του μέσου τετραγωνικού σφάλματος που αφορούν φυσιολογικές εικόνες ξεπερνούν ελάχιστα την τιμή 0.0020, ενώ οι τιμές που αφορούν μη φυσιολογικές εικόνες ξεπερνούν την τιμή 0.0029 και έτσι φαίνεται ότι είναι δυνατό να διαχωριστούν. Με σκοπό να προκύψει μια καλή εκτίμηση του κατωφλίου υπολογίζεται η μέση τιμή και η τυπική απόκλιση του σφάλματος αναπαραγωγής των φυσιολογικών και των μη φυσιολογικών εικόνων.

```
# mean clear
clear_losses = losses[0:10]
sum = 0
for i in range(10):
  sum = sum + clear_losses[i]
clear_mean = sum/10
print("mean(clear) = " + str(clear_mean))

# mean cracked
cracked_losses = losses[10:20]
sum = 0
for i in range(10):
  sum = sum + cracked_losses[i]
cracked_mean = sum/10
print("mean(cracked) = " + str(cracked_mean))

#std clear
clear_std = np.std(clear_losses)
print("std(clear)="+str(clear_std))
#std_cracked
cracked_std = np.std(cracked_losses)
print("std(cracked)="+str(cracked_std))
```

Οι τιμές που προέκυψαν φαίνονται στον παρακάτω πίνακα και σχηματικά στην Εικόνα 24.

Μέση τιμή (mean)	Φυσιολογικές (clear)	0.00184060606131816 ≃ 0.00184
	Μη φυσιολογικές (cracked)	0.003485695914826907 ≃ 0.00349
Τυπική απόκλιση (standard deviation)	Φυσιολογικές (clear)	0.0001657343620876616 ≃ 0.00017
	Μη φυσιολογικές (cracked)	0.0002193132966425681 ≃ 0.00022

Πίνακας 1: Μέση τιμή και τυπική απόκλιση του δείγματος 20 εικόνων



Εικόνα 25: Διαγραμματική απεικόνιση μέσης τιμής και τυπικής απόκλισης του δείγματος 20 εικόνων

Έχοντας υπολογίσει τη μέση τιμή και την τυπική απόκλιση, η τιμή του κατωφλίου η οποία αντιπροσωπεύει τα δεδομένα υπολογίζεται αν στη μέση τιμή των φυσιολογικών εικόνων προστεθεί η αντίστοιχη τυπική απόκλιση. Έτσι, η τιμή που προκύπτει ("thres") είναι η παρακάτω:

```
thres = clear_mean + clear_std
```

$$thres = 0.002006340423405822 \simeq 0.00201$$

4.5 Έλεγχος της απόδοσης του μοντέλου

α) Έλεγχος με χρήση των φυσιολογικών δεδομένων ελέγχου

Αφού προσδιορίστηκε η τιμή του κατωφλίου σύμφωνα με την οποία διαχωρίζονται οι εικόνες που εισάγονται στο μοντέλο σε φυσιολογικές ή μη φυσιολογικές, χρησιμοποιούνται οι φυσιολογικές εικόνες που δεν χρησιμοποιήθηκαν για την εκπαίδευση για να ελεγχθεί αν επιτυγχάνει το μοντέλο να τις ταξινομήσει σαν φυσιολογικές.

```
# load images
clear_test_path = 'drive/My Drive/diplomatiki_colab/dedomena_unzipped/tunnel-crack-images/DataCrop/RGB/Clear_for_testing_cleared.zip'
zf2 = ZipFile(clear_test_path, 'r')
file_list_clear_test = ZipFile.namelist(zf2)
clear_test_images = []
i=0
for f in file_list_clear_test:
    print(f)
    ifile = zf2.open(f)
```

```
    if ifile is None:
        print('File can not be loaded')
        break

    temp = Image.open(ifile)
    temp = temp.convert('L')
    temp = np.array(temp)
    #NORMALIZE
    temp = temp/255
    clear_test_images.append(temp)
    if clear_test_images[i] is None:
        print('Image can not be loaded')
        break
    i=i+1
```

Αρχικά, με όμοιο τρόπο με τα δεδομένα εκπαίδευσης, διαβάζονται οι εικόνες από τον αντίστοιχο συμπιεσμένο φάκελο, μετατρέπονται σε ασπρόμαυρες, κανονικοποιούνται και εισάγονται σε μία λίστα που ονομάζεται “clear_test_images”.

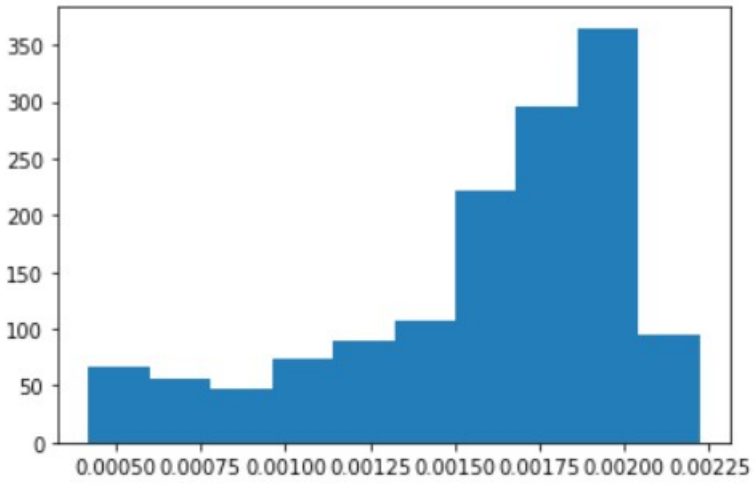
```
clear_test_images = np.array(clear_test_images)
clear_test_images = np.reshape(clear_test_images, (len(clear_test_images), 45, 45, 1))
```

Στη συνέχεια, πριν εισαχθούν οι εικόνες στο μοντέλο, η λίστα που τις περιλαμβάνει μετατρέπεται σε numpy array και έπειτα αναδιαμορφώνεται ώστε ο πίνακας που αντιστοιχεί στην κάθε εικόνα να έχει διαστάσεις (45, 45, 1).

```
clear_prediction = model1.predict(clear_test_images)
clear_losses = []
for i in range(len(clear_test_images)):
  clear_losses.append(mse(clear_test_images[i], clear_prediction[i]))
```

Όπως και στην προηγούμενη ενότητα, μέσω της εντολής “predict” το εκπαιδευμένο μοντέλο προσπαθεί να αναπαράξει τις εικόνες του “clear_test_images” και στη συνέχεια για κάθε πρόβλεψη υπολογίζεται το μέσο τετραγωνικό σφάλμα ώστε να υπάρχει μια εκτίμηση της ποιότητας της αναπαραγωγής.

Στην Εικόνα 26 στον οριζόντιο άξονα φαίνονται οι τιμές του μέσου τετραγωνικού σφάλματος και στον κατακόρυφο άξονα φαίνεται το πλήθος εμφανίσεων της κάθε τιμής. Σημειώνεται ότι το κατώφλι που προσδιορίστηκε στην προηγούμενη ενότητα είναι κοντά στην τιμή 0,002 και για τις εικόνες που θεωρούνται φυσιολογικές αναμένεται το μέσο τετραγωνικό σφάλμα να μην ξεπερνά την τιμή αυτή. Παρατηρώντας το ιστόγραμμα φαίνεται ότι οι εικόνες στο διάστημα από 0,00200 έως 0,00225 είναι λίγες αν ληφθεί υπόψιν το σύνολο των φυσιολογικών εικόνων ελέγχου. Από το γεγονός αυτό αναμένεται το μοντέλο να είναι σε θέση να χαρακτηρίσει με επιτυχία μια φυσιολογική εικόνα.



Εικόνα 26: Ιστόγραμμα φυσιολογικών εικόνων

```
clear_correct_count = 0
for i in range(len(clear_losses)):
  if clear_losses[i]<=thres:
    clear_correct_count = clear_correct_count + 1


clear_percentage = (clear_correct_count/(len(clear_losses)))*100
print("Correct(clear): " + str(clear_percentage) + " %")
```

Έχοντας μία εποπτική εκτίμηση για την ακρίβεια των προβλέψεων, εφαρμόζεται το κατώφλι ώστε να προσδιοριστεί ποσοτικά το ποσοστό επιτυχίας του μοντέλου. Αρχικά δημιουργείται ένας μετρητής ο οποίος αρχικοποιείται στην τιμή 0. Μέσω μιας επανάληψης, ελέγχεται αν το μέσο τετραγωνικό σφάλμα που προέκυψε για κάθε φυσιολογική εικόνα είναι μικρότερο ή ίσο με το κατώφλι. Αν είναι μικρότερο, η φυσιολογική αυτή εικόνα αποτελεί μια σωστή πρόβλεψη και αυξάνεται ο μετρητής κατά 1. Αφού ολοκληρωθεί η επανάληψη, προσδιορίζεται το ποσοστό βρίσκοντας ποιο μέρος του συνόλου αποτελεί το πλήθος των σωστών προβλέψεων. Τελικά το ποσοστό επιτυχίας που προέκυψε είναι:

$$Correct(clear): 88.90459363957596\,\% \simeq 88.90\,\%$$

β) Έλεγχος με χρήση των μη φυσιολογικών δεδομένων ελέγχου

Αφού ελέγχθηκε η απόδοση του μοντέλου στον εντοπισμό της φυσιολογικής κατάστασης, πρέπει να ελεγχθεί και ο βαθμός επιτυχίας του στον εντοπισμό ανώμαλων συμβάντων. Για τον σκοπό αυτό χρησιμοποιούνται οι εικόνες ρηγματωμένου τοιχώματος που εντοπίστηκαν μετά την κατάτμηση των αρχικών δεδομένων. Ομοίως με τη διαδικασία που ακολουθήθηκε για τις φυσιολογικές εικόνες, ζητείται από το μοντέλο να πραγματοποιήσει πρόβλεψη για το σύνολο των μη φυσιολογικών εικόνων και ελέγχεται το μέσο τετραγωνικό σφάλμα που προκύπτει σε σχέση με το καθορισμένο κατώφλι.

```
# load cracked images
cracked_test_path = input_path + 'Cracked_cleared.zip'
zf3 = ZipFile(cracked_test_path, 'r')
file_list_cracked_test = ZipFile.namelist(zf3)
i=0
cracked_test_images = []
for f in file_list_cracked_test:
    print(f)
    ifile = zf3.open(f)
    if ifile is None:
        print('File can not be loaded')
        break

    temp = Image.open(ifile)
    temp = temp.convert('L')
    temp = np.array(temp)
    #NORMALIZE
    temp = temp/255
    cracked_test_images.append(temp)
    if cracked_test_images[i] is None:
        print('Image can not be loaded')
        break
    i = i + 1


cracked_test_images = np.array(cracked_test_images)
cracked_test_images = np.reshape(cracked_test_images, (len(cracked_test_images),
45, 45, 1))
```

Με βάση τη διαδικασία που ακολουθήθηκε για να διαβαστούν τα δεδομένα εκπαίδευσης αλλά και οι φυσιολογικές εικόνες ελέγχου, οι μη φυσιολογικές εικόνες ελέγχου διαβάζονται από τον αντίστοιχο συμπιεσμένο φάκελο, μετατρέπονται σε ασπρόμαυρες, κανονικοποιούνται και εισάγονται

στη λίστα “cracked_test_images”. Η λίστα αυτή στη συνέχεια μετατρέπεται σε numpy array με την ίδια ονομασία και αναδιαμορφώνεται ώστε οι διαστάσεις που αντιστοιχούν στον πίνακα της κάθε εικόνας να είναι (45, 45, 1).

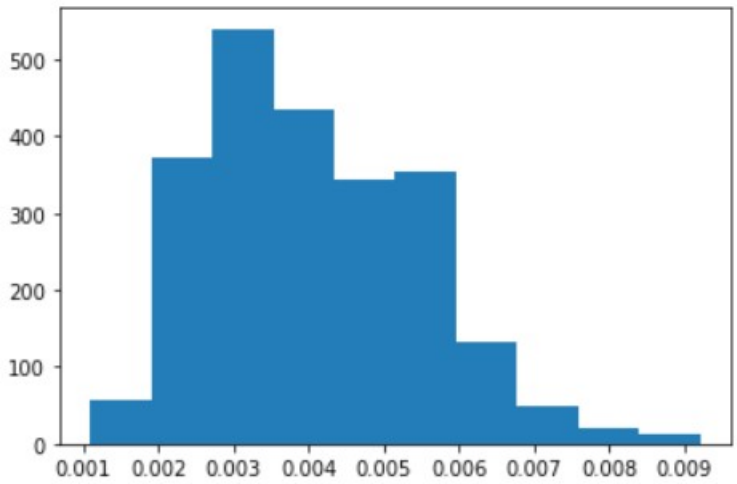
```
cracked_prediction = model1.predict(cracked_test_images)
cracked_losses = []
for i in range(len(cracked_test_images)):
  cracked_losses.append(mse(cracked_test_images[i], cracked_prediction[i]))
```

Έπειτα το numpy array που περιλαμβάνει τους πίνακες των εικόνων εισάγεται στο μοντέλο το οποίο προσπαθεί να το αναπαράξει στην έξοδό του. Για κάθε εικόνα στο “cracked_test_images” υπολογίζεται το μέσο τετραγωνικό σφάλμα το οποίο δείχνει με ποσοτικό τρόπο πόσο επιτυχημένα το μοντέλο κατάφερε να αναδημιουργήσει την εικόνα.

Στην Εικόνα 27 φαίνεται ένα τυχαίο παράδειγμα μη φυσιολογικής εικόνας ελέγχου στην οποία παρατηρείται μία εμφανής ρωγμή. Αφού το μοντέλο είναι εκπαιδευμένο στην ομοιόμορφη και φυσιολογική κατάσταση, η εικόνα που παράχθηκε δεν θυμίζει την αρχική καθώς έχει εξαλειφθεί πλήρως η ρωγμή. Αντίθετα, η έξοδος του μοντέλου θυμίζει περισσότερο τα δεδομένα εκπαίδευσης ή τις φυσιολογικές εικόνες ελέγχου. Συνεπώς, όταν υπολογίζεται το μέσο τετραγωνικό σφάλμα για μια εικόνα που περιλαμβάνει κάποιο ανώμαλο συμβάν, όπως η ρωγμή στη συγκεκριμένη περίπτωση, αυτό αναμένεται να είναι μεγαλύτερο από το αντίστοιχο σφάλμα για μία φυσιολογική εικόνα αφού τα εικονοστοιχεία της εξόδου έχουν εμφανείς διαφορές με τα εικονοστοιχεία της εισόδου.

Εικόνα 27: Σύγκριση αρχικής μη φυσιολογικής εικόνας με την αντίστοιχη έξοδο του μοντέλου

Πράγματι υπολογίζοντας το μέσο τετραγωνικό σφάλμα παρατηρήθηκε ξεκάθαρη διαφοροποίηση των μη φυσιολογικών εικόνων από τις φυσιολογικές. Στην Εικόνα 28 ο οριζόντιος άξονας απεικονίζει τις τιμές του μέσου τετραγωνικού σφάλματος, ενώ ο κατακόρυφος τη συχνότητα εμφάνισης της κάθε τιμής. Με την τιμή του κατωφλίου κοντά στο 0,002, φαίνεται ότι η πλειονότητα των εικόνων ξεπερνάει το κατώφλι με αποτέλεσμα να ταξινομείται ορθώς στην κατηγορία των μη φυσιολογικών. Οι εικόνες για τις οποίες προέκυψε μέσο τετραγωνικό σφάλμα μικρότερο από το κατώφλι με αποτέλεσμα να θεωρούνται λανθασμένα φυσιολογικές είναι λίγες σε σχέση με τον αριθμό των εικόνων που περιλαμβάνει το σύνολο των μη φυσιολογικών εικόνων ελέγχου.



Εικόνα 28: Ιστόγραμμα μη φυσιολογικών εικόνων

Με αυτό τον τρόπο φαίνεται εποπτικά ότι μέσω του μοντέλου επιτυγχάνεται διαχωρισμός φυσιολογικών και μη φυσιολογικών εικόνων με ικανοποιητική ακρίβεια.

```
cracked_correct_count = 0
for i in range(len(cracked_losses)):
  if cracked_losses[i]>=thres:
    cracked_correct_count = cracked_correct_count + 1

cracked_percentage = (cracked_correct_count/(len(cracked_losses)))*100
print("Correct(cracked): " + str(cracked_percentage) + " %")
```

Με σκοπό να υπάρχει και ποσοτική εκτίμηση της απόδοσης του μοντέλου στις μη φυσιολογικές εικόνες ορίστηκε και πάλι ένας μετρητής με αρχική τιμή το 0. Μέσω μιας επανάληψης ελέγχεται αν το μέσο τετραγωνικό σφάλμα για κάθε μη φυσιολογική εικόνα ξεπερνά το κατώφλι. Αν αυτό ισχύει, η εικόνα θεωρήθηκε ορθώς μη φυσιολογική και ο μετρητής αυξάνεται κατά 1. Στη συνέχεια, οι σωστές προβλέψεις εκφράζονται σαν ποσοστό του συνόλου των μη φυσιολογικών εικόνων και προκύπτει το παρακάτω ποσοστό επιτυχίας για τις μη φυσιολογικές εικόνες.

$$Correct(cracked): 96.460940871817\% \simeq 96.46\%$$

Σύμφωνα με τα παραπάνω ποσοστά, το μοντέλο είναι σε θέση να διαχωρίσει επιτυχημένα τις φυσιολογικές και τις μη φυσιολογικές εικόνες κάνοντας ελάχιστα λάθη. Αφού ελέγχθηκε η απόδοση του ξεχωριστά σε φυσιολογικές και μη φυσιολογικές είναι χρήσιμο να προσδιοριστεί κάποιο μέτρο της απόδοσής του σε όλα τα δεδομένα ελέγχου ανεξάρτητα από τη φύση τους.

γ) Υπολογισμός συνολικού ποσοστού επιτυχίας

Μέσω του ελέγχου με χρήση φυσιολογικών εικόνων είναι γνωστό το πλήθος των σωστών προβλέψεων μέσω του μετρητή “clear_correct_count”, ο οποίος προσδιορίστηκε με εφαρμογή του κατωφλίου. Παράλληλα με όμοιο τρόπο προσδιορίστηκε και ο μετρητής “cracked_correct_count” ο οποίος δείχνει το πλήθος των σωστών προβλέψεων στις μη φυσιολογικες εικόνες.

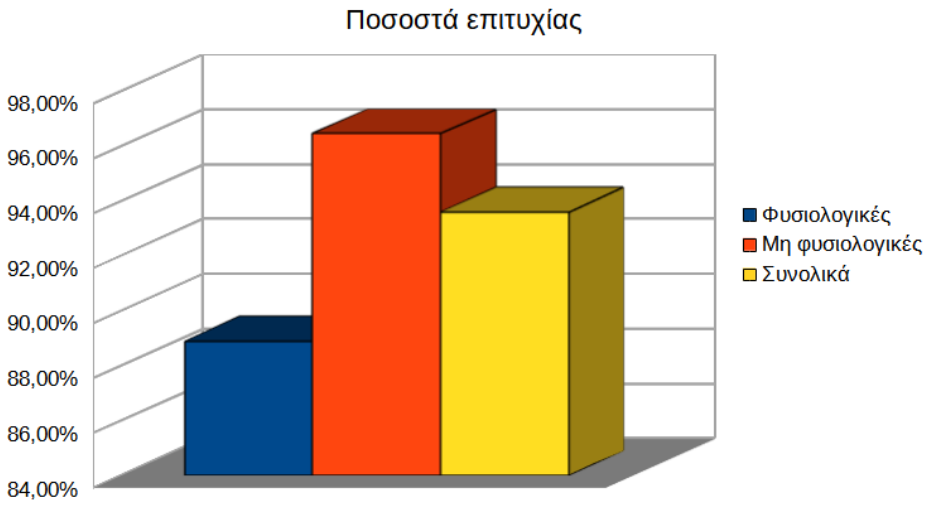
```
total_correct_count = cracked_correct_count + clear_correct_count
total_len = len(cracked_losses) + len(clear_losses)
```

Συνεπώς είναι δυνατό να προσδιοριστεί ο δείκτης “total_correct_count”, ο οποίος δείχνει το πλήθος των σωστών προβλέψεων στις φυσιολογικές και στις μη φυσιολογικές εικόνες συνολικά. Ο δείκτης αυτός υπολογίζεται ως το άθροισμα των “cracked_correct_count” και “clear_correct_count”. Επίσης, προσθέτοντας το πλήθος των φυσιολογικών και των μη φυσιολογικών εικόνων προκύπτει ο δείκτης “total_len”, ο οποίος δείχνει το συνολικό πλήθος των εικόνων ελέγχου.

```
total_percentage = (total_correct_count/(total_len))*100
print("Total correct: " + str(total_percentage) + " %")
```

$$Total\,correct: 93.59592711682744\% \simeq 93.60\%$$

Εκφράζοντας το πλήθος των σωστών προβλέψεων σαν ποσοστό του συνολικού πλήθους των εικόνων φαίνεται ότι το μοντέλο καταφέρνει με μεγάλη επιτυχία να προσδιορίσει αν μια εικόνα περιλαμβάνει κάποιο ανώμαλο συμβάν ή αν απεικονίζει τη φυσιολογική κατάσταση.



Εικόνα 29: Διαγραμματική απεικόνιση των παραγόμενων ποσοστών

δ) Accuracy, Precision, Recall και F1 Score

Προτού υπολογιστούν οι παράμετροι accuracy, precision, recall και F1 score είναι απαραίτητο να οριστούν οι έννοιες των αληθινά θετικών (true positive), αληθινά αρνητικών (true negative), ψευδώς θετικών (false positive) και ψευδώς αρνητικών (false negative).

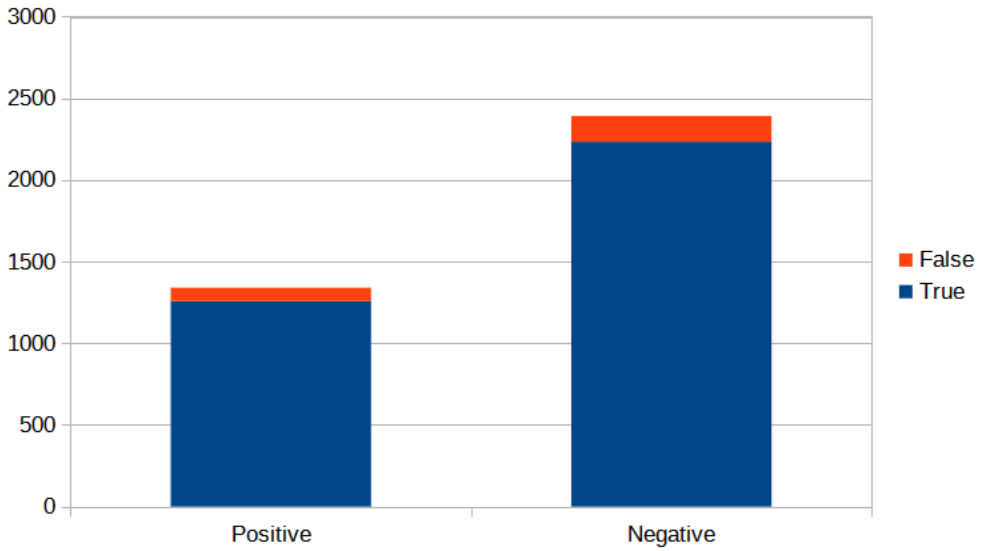
- Αληθινά θετικά (TP): Με τον όρο αυτό χαρακτηρίζονται τα στοιχεία των οποίων η πραγματική τιμή είναι θετική και η προβλεπόμενη τιμή είναι επίσης θετική. Στη συγκεκριμένη εφαρμογή, TP θεωρούνται οι εικόνες που είναι φυσιολογικές και μέσω του μοντέλου χαρακτηρίζονται και πάλι φυσιολογικές. [45]
- Αληθινά αρνητικά (TN): Αληθινά αρνητικά θεωρούνται τα στοιχεία που η πραγματική τους τιμή είναι αρνητική αλλά και η προβλεπόμενη τιμή είναι αρνητική. Στη συγκεκριμένη εφαρμογή, TN θεωρούνται οι εικόνες που είναι μη φυσιολογικές και μέσω του μοντέλου χαρακτηρίζονται σαν μη φυσιολογικές. [45]
- Ψευδώς θετικά (FP): Ψευδώς θετικά χαρακτηρίζονται τα στοιχεία που η πραγματική τους τιμή είναι αρνητική αλλά η προβλεπόμενη είναι θετική. Στη συγκεκριμένη εφαρμογή, FP θεωρούνται οι εικόνες που είναι μη φυσιολογικές αλλά χαρακτηρίζονται από το μοντέλο σαν φυσιολογικές. [45]
- Ψευδώς αρνητικά (FN): Ψευδώς αρνητικά χαρακτηρίζονται τα στοιχεία που η πραγματική τους τιμή είναι θετική αλλά η προβλεπόμενη είναι αρνητική. Στη συγκεκριμένη εφαρμογή, FN θεωρούνται οι εικόνες που είναι φυσιολογικές αλλά χαρακτηρίζονται από το μοντέλο σαν μη φυσιολογικές. [45]

```
TP = clear_correct_count
TN = cracked_correct_count
FP = len(cracked_losses) - cracked_correct_count
FN = len(clear_losses) - clear_correct_count
```

Σύμφωνα με τα παραπάνω, TP ονομάζεται ο μετρητής επιτυχημένων προβλέψεων για τις φυσιολογικές εικόνες και TN ονομάζεται ο μετρητής επιτυχημένων προβλέψεων για τις μη φυσιολογικές εικόνες. Παράλληλα, η παράμετρος FP υπολογίζεται σαν τη διαφορά του συνολικού αριθμού των μη φυσιολογικών εικόνων και του μετρητή επιτυχημένων προβλέψεων για τις μη φυσιολογικές εικόνες. Ομοίως, η παράμετρος FN υπολογίζεται σαν τη διαφορά του συνολικού αριθμού των φυσιολογικών εικόνων και του μετρητή επιτυχημένων προβλέψεων για τις φυσιολογικές εικόνες. Οι αριθμητικές τιμές των παραμέτρων αυτών παρουσιάζονται στον παρακάτω πίνακα και σχηματικά στο παρακάτω διάγραμμα.

True Positive (TP)	1258
False Positive (FP)	82
True Negative (TN)	2235
False Negative (FN)	157

Πίνακας 2: Τιμές των παραμέτρων TP, FP, TN, FN



Εικόνα 30: Διαγραμματική παρουσίαση των τιμών TP, FP, TN, FN

Αφού υπολογίστηκαν οι παραπάνω παράμετροι είναι δυνατό να υπολογιστούν τα μέτρα απόδοσης accuracy, precision, recall και F1 score.

- Accuracy

Η παράμετρος accuracy υπολογίζεται σαν τον λόγο των σωστά προβλεπόμενων παρατηρήσεων προς το σύνολο των παρατηρήσεων και είναι ουσιαστικά το συνολικό ποσοστό που υπολογίστηκε στην προηγούμενη ενότητα. Η παράμετρος αυτή είναι ιδιαίτερα χρήσιμο μέτρο απόδοσης όταν το σύνολο δεδομένων είναι συμμετρικό, δηλαδή όταν οι τιμές των ψευδώς θετικών και των ψευδώς αρνητικών στοιχείων είναι σχεδόν ίδιες. [45]

```
accuracy = (TP + TN)/(TP + FP + FN + TN)
```

Η παράμετρος αυτή υπολογίζεται όπως φαίνεται παραπάνω και προκύπτει η παρακάτω τιμή.

$$Accuracy = 0.9359592711682744 \simeq 0.9360$$

- Precision

Η παράμετρος precision είναι ο λόγος των σωστά προβλεπόμενων θετικών στοιχείων, δηλαδή των σωστά προβλεπόμενων φυσιολογικών εικόνων, προς το σύνολο των προβλεπόμενων θετικών παρατηρήσεων, δηλαδή προς το σύνολο των προβλεπόμενων φυσιολογικών εικόνων. Μεγάλη τιμή του precision σχετίζεται με το χαμηλό ποσοστό ψευδώς θετικών αποτελεσμάτων. [45]

```
precision = TP/(TP + FP)
```

$$Precision = 0.9388059701492537 \simeq 0.9388$$

- Recall

Η παράμετρος recall είναι ο λόγος των σωστά προβλεπόμενων θετικών παρατηρήσεων προς το σύνολο των πραγματικά θετικών παρατηρήσεων. [45] Δηλαδή είναι ο λόγος των προβλεπόμενων φυσιολογικών εικόνων προς το σύνολο των φυσιολογικών εικόνων. Το σύνολο των φυσιολογικών εικόνων είναι το άθροισμα των προβλεπόμενων φυσιολογικών εικόνων με τις ψευδώς μη φυσιολογικές εικόνες. Η παράμετρος αυτή υπολογίστηκε και στην προηγούμενη ενότητα σαν το ποσοστό επιτυχίας των φυσιολογικών εικόνων.

```
recall = TP/(TP + FN)
```

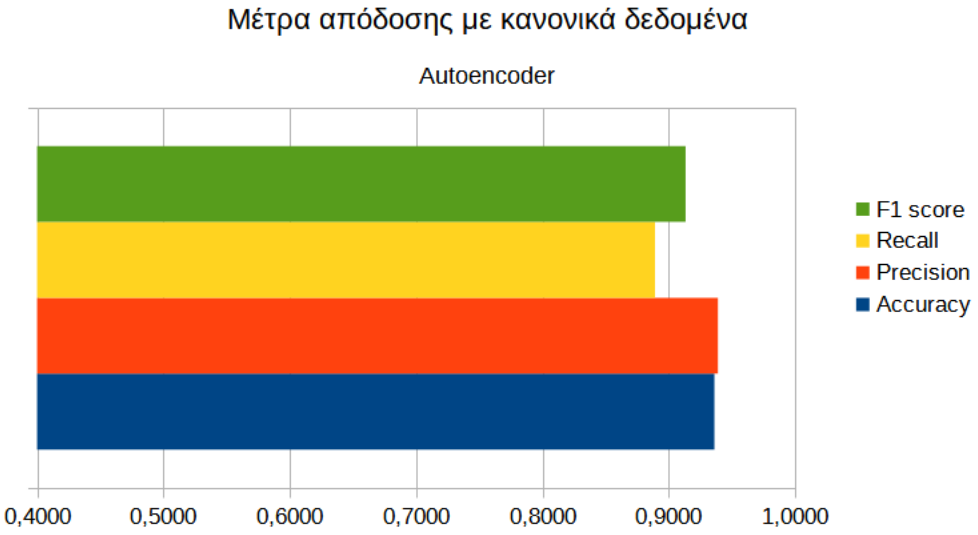
$$Recall = 0.8890459363957597 \simeq 0.8890$$

- F1 score

Η παράμετρος F1 score είναι ο σταθμισμένος μέσος όρος των παραμέτρων precision και recall. Με αυτό τον τρόπο λαμβάνει υπόψιν τόσο τα ψευδώς θετικά όσο και τα ψευδώς αρνητικά στοιχεία. Είναι ιδιαίτερα χρήσιμη όταν η κατανομή των κλάσεων είναι ανομοιόμορφη [45].

```
f1_score = 2*(recall*precision)/(recall + precision)
```

$$F1 \quad score = 0.9132486388384755 \simeq 0.9132$$



Εικόνα 31: Διαγραμματική παρουσίαση μέτρων απόδοσης

Όλες οι παραπάνω παράμετροι φαίνεται να πλησιάζουν πολύ στο 1, πράγμα που δείχνει ότι το μοντέλο παρουσιάζει πολύ καλή απόδοση στην ταξινόμηση των δεδομένων ελέγχου που αντιμετώπισε.

ε) Καμπύλη ROC – AUC

Η καμπύλη ROC (Receiver Operating Characteristics) – AUC (Area Under the Curve), η οποία είναι επίσης γνωστή και ως AUROC (Area Under the Receiver Operating Characteristics) είναι ένας από τους σημαντικότερους τρόπους αξιολόγησης της απόδοσης οποιουδήποτε μοντέλου ταξινόμησης και εφαρμόζεται σε διάφορες ρυθμίσεις κατωφλίου. Η ROC είναι μια καμπύλη πιθανοτήτων και η AUC αντιπροσωπεύει το βαθμό διαχωρισιμότητας μεταξύ των κλάσεων, δηλαδή το βαθμό που το μοντέλο είναι ικανό να διαχωρίσει τις κλάσεις. Ιδανικά η τιμή του AUC πρέπει να πλησιάζει το 1. Αν η τιμή του AUC είναι κοντά στο 0, το μοντέλο πραγματοποιεί αντίστροφες προβλέψεις. Δηλαδή στην εφαρμογή της παρούσας εργασίας, οι μη φυσιολογικές εικόνες θα είχαν θεωρηθεί σαν φυσιολογικές και το αντίστροφο. Παράλληλα, στην περίπτωση που η παράμετρος AUC είναι 0,5, το μοντέλο δεν έχει την ικανότητα διαχωρισμού κλάσεων. [46]

```
# y_test, losses
y_test = []
losses = []

for i in range(len(clear_losses)):
  y_test.append(0) #clear
  losses.append(clear_losses[i])

for i in range(len(cracked_losses)):
```

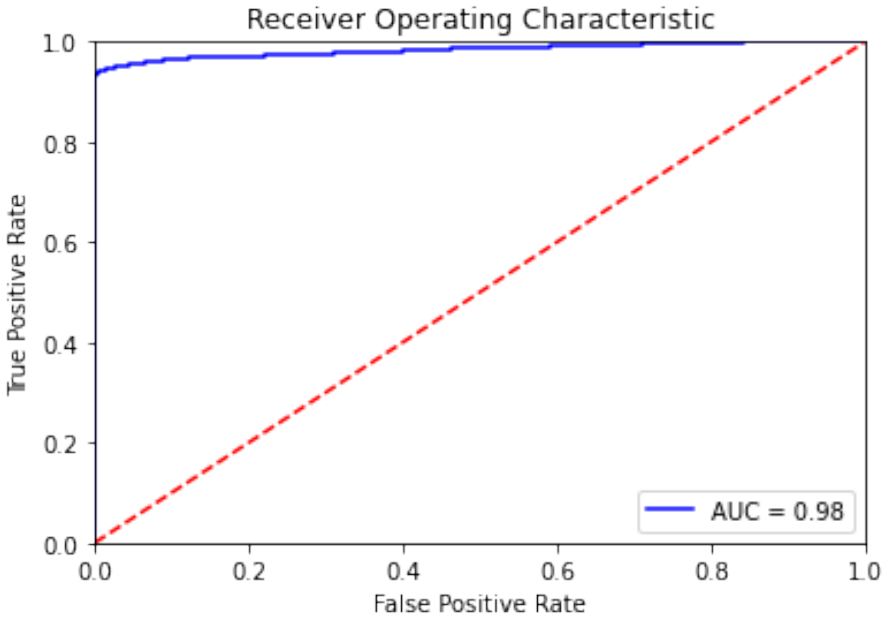
```
  y_test.append(1) #cracked
  losses.append(cracked_losses[i])

y_test = np.array(y_test)
losses = np.array(losses)
```

Για να προσδιοριστεί η καμπύλη ROC και η τιμή του AUC χρειάζεται να δημιουργηθούν labels για τις εικόνες ελέγχου με το 0 να αντιστοιχεί στις φυσιολογικές εικόνες και το 1 στις μη φυσιολογικές. Έτσι, δημιουργείται ένα ενιαίο numpy array με τις ετικέτες φυσιολογικών και μη φυσιολογικών εικόνων (“y_test”) και ένα numpy array με τα αντίστοιχα μέσα τετραγωνικά σφάλματα όλων των εικόνων (“losses”).

```
import sklearn.metrics as metrics
import matplotlib.pyplot as plt

fpr, tpr, threshold = metrics.roc_curve(y_test,losses)
roc_auc = metrics.auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Τα δύο numpy arrays που δημιουργήθηκαν εισάγονται στη συνάρτηση “roc_curve” και υπολογίζονται οι παράμετροι “fpr” (false positive rate), “tpr” (true positive rate) και “threshold”. Στη συνέχεια οι παράμετροι “fpr” και “tpr” εισάγονται στην συνάρτηση “auc” και τελικά παράγεται το διάγραμμα της Εικόνας 32.

Όπως φαίνεται στην Εικόνα 32, η παράμετρος AUC ισούται με 0,98 και είναι πολύ κοντά στο 1. Το γεγονός αυτό δείχνει ότι το μοντέλο μπορεί να διαχωρίσει με μεγάλη επιτυχία τις δύο αυτές κλάσεις και έχει γενικά πολύ καλή απόδοση στα δεδομένα ελέγχου, πράγμα που ήταν εμφανές και από τις παραμέτρους accuracy, precision, recall και F1 score.



Εικόνα 32: Καμπύλη ROC - AUC

5 *Πειραματικά αποτελέσματα*

Αφού παρουσιάστηκαν τα αποτελέσματα εφαρμογής του αυτόματου κωδικοποιητή στα δεδομένα ελέγχου και περιγράφηκε η διαδικασία, στο κεφάλαιο αυτό ελέγχεται η απόδοσή του σε δεδομένα στα οποία έχει εφαρμοστεί φίλτρο εξομάλυνσης και φίλτρο προσθήκης θορύβου αντίστοιχα. Παράλληλα, η απόδοση του αυτόματου κωδικοποιητή συγκρίνεται με τα αποτελέσματα εφαρμογής άλλων αλγορίθμων, του K-means, του Mixture of Gaussians και ενός επιβλεπόμενου νευρωνικού δικτύου, στα ίδια δεδομένα ελέγχου.

5.1 Εξομάλυνση και προσθήκη θορύβου στα δεδομένα

α) Διαδικασία εξομάλυνσης

Όπως αναφέρθηκε παραπάνω, με σκοπό να αξιολογηθεί επιτυχώς η απόδοση του αυτόματου κωδικοποιητή, θεωρήθηκε απαραίτητο, εκτός των δεδομένων που χρησιμοποιήθηκαν στο προηγούμενο κεφάλαιο, να χρησιμοποιηθούν και δεδομένα που έχουν υποστεί εξομάλυνση. Εφαρμόζοντας

εξομάλυνση αλλάζει η μορφή των δεδομένων με αποτέλεσμα το έργο του αυτόματου κωδικοποιητή να δυσχεραίνεται, αφού δυσδιάκριτες ρωγμές ή άλλου είδους ασυνέχειες εναρμονίζονται ακόμα περισσότερο με το περιβάλλον τους.

Υπάρχουν πολλά παραδείγματα φίλτρων εξομάλυνσης όπως το φίλτρο μέσου όρου (box filter), το φίλτρο διαμέσου (median filter) και το φίλτρο Gauss, το οποίο επιλέχθηκε να χρησιμοποιηθεί στα πλαίσια της παρούσας εργασίας.

Το φίλτρο Gauss εφαρμόζεται μέσω δισδιάστατης συνέλιξης και χρησιμοποιείται για να εξομαλύνει εικόνες, να αφαιρέσει λεπτομέρειες και θόρυβο. Χρησιμοποιεί μια γκαουσιανή συνάρτηση για τον υπολογισμό του μετασχηματισμού που εφαρμόζεται σε κάθε εικονοστοιχείο της εικόνας. Θεωρητικά, η γκαουσιανή συνάρτηση είναι μη μηδενική σε κάθε σημείο της εικόνας στην οποία εφαρμόζεται και έτσι στον αντίστοιχο υπολογισμό για κάθε εικονοστοιχείο θα έπρεπε να συμπεριληφθεί ολόκληρη η εικόνα. Πρακτικά, κατά τον υπολογισμό μιας προσέγγισης της γκαουσιανής συνάρτησης, τα εικονοστοιχεία που απέχουν απόσταση μεγαλύτερη από το τριπλάσιο της τυπικής απόκλισης έχουν τόσο μικρή επιρροή που μπορεί να αγνοηθεί. Σε μία διάσταση ο τύπος της συνάρτησης είναι:

$$G(x)=\frac{1}{\sqrt{(2\pi\sigma^2)}}e^{-\frac{x^2}{2\sigma^2}} \qquad (5.1)$$

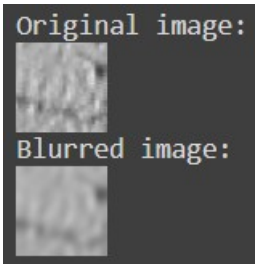
Στη συγκεκριμένη εφαρμογή, η συνάρτηση εφαρμόζεται σε μία εικόνα, η οποία έχει δύο διαστάσεις. Έτσι, η αντίστοιχη συνάρτηση είναι:

$$G(x,y)=\frac{1}{\sqrt{(2\pi\sigma^2)}}e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (5.2)$$

Όπου x η απόσταση από την αρχή στον οριζόντιο άξονα, y η απόσταση από την αρχή στον κατακόρυφο άξονα και σ η τυπική απόκλιση της γκαουσιανής κατανομής. [47]

Το φίλτρο Gauss εφαρμόστηκε σε όλες τις εικόνες, δηλαδή στα δεδομένα εκπαίδευσης και στα δεδομένα ελέγχου, με τη χρήση της συνάρτησης "GaussianBlur" που περιλαμβάνεται στη βιβλιοθήκη "OpenCV". Η συνάρτηση αυτή δέχεται σαν ορίσματα την εικόνα στην οποία θα εφαρμοστεί η εξομάλυνση, το μέγεθος του πυρήνα του φίλτρου και την τιμή της τυπικής απόκλισης. Με σκοπό να είναι εμφανής η εξομάλυνση στις εικόνες μετά την εφαρμογή του φίλτρου, επιλέχθηκε μέγεθος πυρήνα 11 × 11 εικονοστοιχεία και τυπική απόκλιση 1,6.

Όπως είναι λογικό, στις φυσιολογικές εικόνες δεν παρατηρείται σημαντική διαφοροποίηση, αφού απεικονίζουν ομοιόμορφο τοίχωμα. Στις εικόνες όμως που περιλαμβάνουν ρωγμές ή άλλα ανώμαλα συμβάντα, η επίδραση της εξομάλυνσης είναι εμφανής. Στην Εικόνα 33 είναι δυνατό να γίνει σύγκριση της αρχικής μη φυσιολογικής εικόνας με αυτή που παράχθηκε από την εφαρμογή του φίλτρου. Στην αρχική εικόνα, εκτός της ρωγμής που φαίνεται εύκολα, είναι εμφανής και η υφή του τοίχου. Από την άλλη πλευρά, στη δεύτερη εικόνα, η υφή του τοίχου έχει σχεδόν εξαφανιστεί και η ρωγμή αναμειγνύεται περισσότερο με το περιβάλλον της.



Εικόνα 33: Παράδειγμα εφαρμογής φίλτρου Gauss

Οι εικόνες που παράχθηκαν με τον τρόπο αυτό χρησιμοποιήθηκαν στη συνέχεια για την εκπαίδευση και τον έλεγχο του αυτόματου κωδικοποιητή και του επιβλεπόμενου νευρωνικού δικτύου καθώς και για την εφαρμογή των αλγορίθμων K-means και Mixture of Gaussians και έπειτα συγκρίθηκαν τα αποτελέσματα.

β) Διαδικασία εισαγωγής θορύβου

Όπως και στην περίπτωση της εξομάλυνσης, υπάρχουν πολλά διαδεδομένα φίλτρα που εισάγουν θόρυβο σε μία εικόνα. Στα πλαίσια της παρούσας εργασίας επιλέχθηκε να εισαχθεί στις εικόνες θόρυβος τύπου αλάτι – πιπέρι, ο οποίος μετατρέπει σε άσπρο και μαύρο διασκορπισμένα εικονοστοιχεία στην επιφάνεια της εικόνας.

Εικόνα 35: Εισαγωγή θορύβου σε φυσιολογική εικόνα

Εικόνα 34: Εισαγωγή θορύβου σε μη φυσιολογική εικόνα

Ο θόρυβος εισάγεται σε όλες τις διαθέσιμες εικόνες πριν εισαχθούν στους αντίστοιχους αλγορίθμους και, σε αντίθεση με την περίπτωση της εξομάλυνσης, δεν επηρεάζει μόνο τις μη φυσιολογικές εικόνες. Συγκεκριμένα, ο θόρυβος είναι εμφανής και στις φυσιολογικές εικόνες, διαταράσσοντας έτσι την ομοιομορφία τους. Στις Εικόνες 34 και 35 φαίνονται παραδείγματα εφαρμογής των παραπάνω σε φυσιολογικές και μη φυσιολογικές εικόνες αντίστοιχα.

Παρατηρώντας τις διαφορές των αρχικών εικόνων με τις παραγόμενες φαίνεται ότι, στην περίπτωση της φυσιολογικής εικόνας, ο θόρυβος δημιουργεί την εντύπωση ότι ο τοίχος είναι τραχύς, ενώ στην περίπτωση της μη φυσιολογικής εικόνας, η ρωγμή εν μέρη κρύβεται από τον διασκορπισμένο θόρυβο.

Η εισαγωγή θορύβου πραγματοποιήθηκε μέσω ορισμού της συνάρτησης “sp_noise” η οποία δέχεται σαν ορίσματα την εικόνα στην οποία θα εισαχθεί ο θόρυβος και έναν αριθμό “prob” ο οποίος δείχνει την πιθανότητα εμφάνισης του θορύβου.

```
import random
def sp_noise(image,prob):
    output = np.zeros(image.shape,np.uint8)
    thres = 1 - prob
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random.random()
            if rdn < prob:
                output[i][j] = 0
            elif rdn > thres:
                output[i][j] = 255
            else:
                output[i][j] = image[i][j]
    return output
```

Στο εσωτερικό της συνάρτησης διατρέχεται η εικόνα κατά μήκος και κατά πλάτος και χρησιμοποιείται η συνάρτηση “random” η οποία επιστρέφει έναν τυχαίο δεκαδικό αριθμό μεταξύ του 0 και του 1. Για κάθε εικονοστοιχείο της εικόνας υπολογίζεται ο τυχαίος αριθμός “rnd” και ελέγχεται αν

είναι μικρότερος του αριθμού “prob”. Αν αυτό ισχύει, η τιμή του εικονοστοιχείου γίνεται 0 και έτσι είναι πλέον μαύρο. Ο αριθμός “prob” στη συγκεκριμένη εφαρμογή ορίστηκε ίσος με 0,05. Αν δεν ικανοποιείται η προηγούμενη συνθήκη, τότε ελέγχεται αν ο αριθμός “rnd” είναι μεγαλύτερος από την τιμή $1-prob=0{,}95$ και αν αυτό ισχύει τότε το εικονοστοιχείο λαμβάνει την τιμή 255 και είναι πλέον άσπρο. Αν και οι δύο προηγούμενες συνθήκες απορριφθούν, το εικονοστοιχείο διατηρεί την τιμή του.

Όπως και με τις εικόνες που έχουν υποστεί εξομάλυνση, οι εικόνες με θόρυβο χρησιμοποιήθηκαν για την εκπαίδευση και τον έλεγχο του αυτόματου κωδικοποιητή και του επιβλεπόμενου νευρωνικού δικτύου καθώς και για την εφαρμογή των αλγορίθμων K-means και Mixture of Gaussians.

5.2 Εφαρμογή αυτόματου κωδικοποιητή μετά από εξομάλυνση και εισαγωγή θορύβου

α) Εξομάλυνση

Όπως αναφέρθηκε παραπάνω, με σκοπό να ελεγχθεί η απόδοση του αυτόματου κωδικοποιητή σε ποικίλα δεδομένα, πραγματοποιήθηκε εκπαίδευση και έλεγχος χρησιμοποιώντας τα δεδομένα που παράχθηκαν με εφαρμογή του φίλτρου Gauss. Συγκρίνοντας τα δεδομένα αυτά με την αρχική τους μορφή, για τις φυσιολογικές εικόνες δεν παρατηρείται σημαντική μεταβολή. Αντίθετα για τις μη φυσιολογικές εικόνες παρατηρείται απώλεια μέρους της υφής των αντικειμένων, ενώ οι υπάρχουσες ρωγμές έγιναν λιγότερο ορατές. Έτσι, αναμένεται ο αυτόματος κωδικοποιητής να μην μπορεί να διαχωρίσει τις εικόνες με εξίσου μεγάλη επιτυχία, αφού οι μη φυσιολογικές εικόνες που έχουν υποστεί εξομάλυνση αναμένεται να παράξουν μικρότερο μέσο τετραγωνικό σφάλμα από τις αρχικές μη φυσιολογικές εικόνες.

Ομοίως με την διαδικασία που ακολουθήθηκε στο προηγούμενο κεφάλαιο με τα αρχικά δεδομένα, διαβάστηκαν οι εικόνες από τους αντίστοιχους συμπιεσμένους φακέλους και αποθηκεύτηκαν σε μορφή λίστας. Στη συνέχεια, στην κάθε εικόνα, δηλαδή στα δεδομένα εκπαίδευσης και στα δεδομένα ελέγχου, έγινε εξομάλυνση μέσω του φίλτρου Gauss με χρήση της συνάρτησης “GaussianBlur” της βιβλιοθήκης “OpenCV”. Έτσι, πλέον υπάρχει ένα νέο σύνολο δεδομένων εκπαίδευσης, το οποίο ονομάστηκε “b_x_train”, και χρησιμοποιήθηκε για την εκπαίδευση του αυτόματου κωδικοποιητή. Σημειώνεται ότι και πάλι ο αυτόματος κωδικοποιητής εκπαιδεύεται μόνο με φυσιολογικά δεδομένα.

```
b_x_train = np.array(blurred_clear_train)
b_x_train = np.reshape(b_x_train, (-1, 45, 45, 1))
b_model1 = autoencoder()
b_model1.summary()
b_model1.fit(b_x_train, b_x_train, batch_size=10, epochs=5)
```

Αφού τα δεδομένα εκπαίδευσης μετατράπηκαν σε numpy array και αναδιαμορφώθηκαν ώστε να έχουν τις κατάλληλες διαστάσεις, έγινε η εκπαίδευση του μοντέλου για 5 εποχές, όπως και με τα αρχικά δεδομένα.

```
Epoch 1/5
100/100 [==============================] - 28s 215ms/step - loss: 0.0330 - mean_squared_error: 0.0330
Epoch 2/5
100/100 [==============================] - 21s 214ms/step - loss: 0.0119 - mean_squared_error: 0.0119
Epoch 3/5
100/100 [==============================] - 21s 214ms/step - loss: 0.0120 - mean_squared_error: 0.0120
Epoch 4/5
100/100 [==============================] - 22s 215ms/step - loss: 0.0122 - mean_squared_error: 0.0122
Epoch 5/5
100/100 [==============================] - 21s 214ms/step - loss: 0.0108 - mean_squared_error: 0.0108
<keras.callbacks.History at 0x7f615e9f6c50>
```

Εικόνα 36: Εκπαίδευση αυτόματου κωδικοποιητή με ομαλοποιημένα δεδομένα

Αφού εκπαιδεύτηκε το μοντέλο, πρέπει να ελεγχθεί η απόδοσή του. Έτσι, χρησιμοποιήθηκαν 20 εικόνες, 10 φυσιολογικές και 10 μη φυσιολογικές, οι οποίες εξομαλύνθηκαν, για τον προσδιορισμό του κατωφλίου. Συγκεκριμένα, όπως και με τα αρχικά δεδομένα, ζητήθηκε από το μοντέλο να κάνει πρόβλεψη για τα δεδομένα αυτά και στη συνέχεια υπολογίστηκε το μέσω τετραγωνικό σφάλμα της κάθε εικόνας με την αντίστοιχη αναπαράστασή της από το μοντέλο. Υπολογίζοντας τη μέση τιμή και την τυπική απόκλιση για τις φυσιολογικές και τις μη φυσιολογικές εικόνες, το κατώφλι προκύπτει σαν άθροισμα της μέσης τιμής των φυσιολογικών με την τυπική απόκλιση των φυσιολογικών.

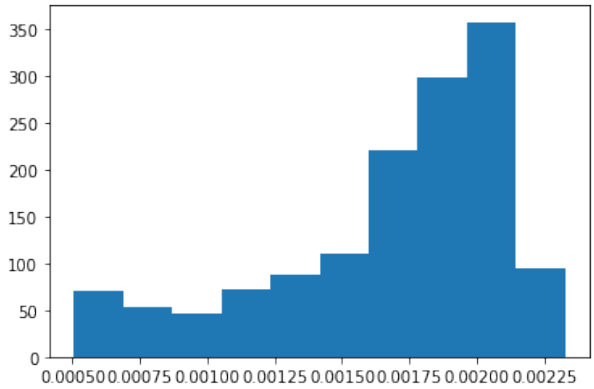
Μέση τιμή (mean)	Φυσιολογικές (blurred clear)	0.0019412049019994662 ≃ 0.00194
	Μη φυσιολογικές (blurred cracked)	0.003183104370643338 ≃ 0.00318
Τυπική απόκλιση (standard deviation)	Φυσιολογικές (blurred clear)	0.00016873154421474638 ≃ 0.00017
	Μη φυσιολογικές (blurred cracked)	0.00021699324366006588 ≃ 0.00022

Πίνακας 3: Μέση τιμή και τυπική απόκλιση του δείγματος 20 ομαλοποιημένων εικόνων

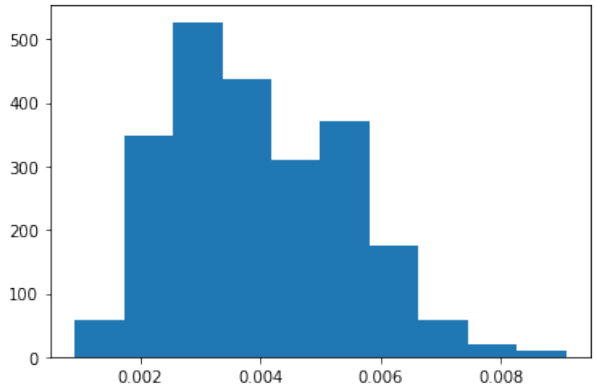
Με βάση τα παραπάνω, το κατώφλι υπολογίζεται ίσο με:

$$threshold_{blurred} = mean_{blurred-clear} + std_{blurred-clear} = 0.0021099364462142125 \simeq 0.00211$$

Έχοντας την τιμή του κατωφλίου, ζητείται από το μοντέλο να πραγματοποιήσει πρόβλεψη, δηλαδή να αναπαράξει, όλες τις εικόνες ελέγχου και υπολογίζεται το μέσο τετραγωνικό σφάλμα σύμφωνα με το οποίο θα διαχωριστούν οι εικόνες σε φυσιολογικές και μη φυσιολογικές.



Εικόνα 37: Ιστόγραμμα μέσου τετραγωνικού σφάλματος για τις φυσιολογικές εικόνες με εξομάλυνση



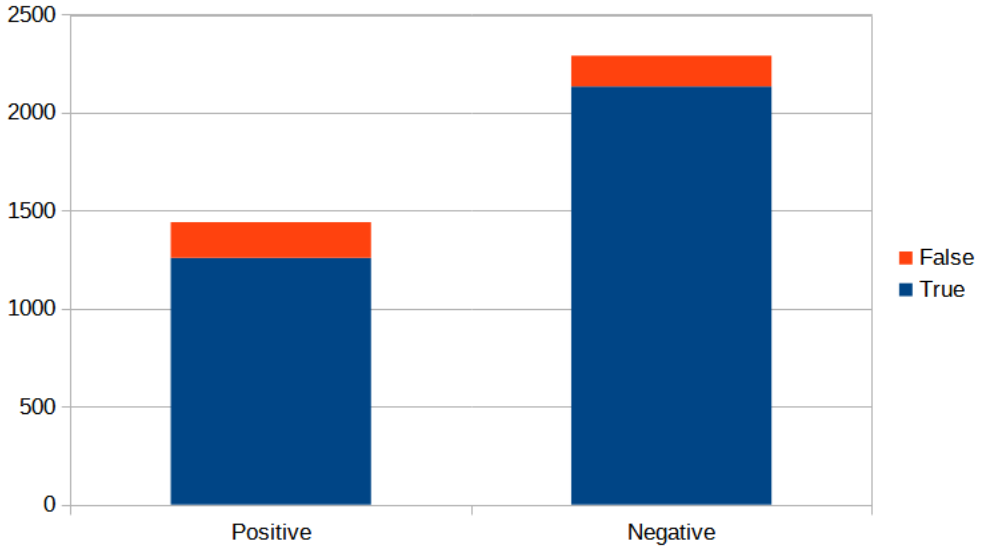
Εικόνα 38: Ιστόγραμμα μέσου τετραγωνικού σφάλματος για τις μη φυσιολογικές εικόνες με εξομάλυνση

Στις Εικόνες 37 και 38 φαίνονται τα ιστογράμματα μέσου τετραγωνικού σφάλματος για τις φυσιολογικές και τις μη φυσιολογικές εικόνες. Στον άξονα x'x φαίνονται οι τιμές του μέσου τετραγωνικού σφάλματος και στον άξονα y'y το πλήθος εμφάνισης της κάθε τιμής. Το κατώφλι ορίστηκε κοντά στην τιμή 0,0021, οπότε φαίνεται ότι ένα μικρό τμήμα των φυσιολογικών εικόνων θα θεωρηθεί ότι ανήκει στις μη φυσιολογικές και ομοίως ένα μικρό τμήμα των μη φυσιολογικών θα ταξινομηθεί στις φυσιολογικές. Το γεγονός αυτό, το οποίο φαίνεται σχηματικά από τα παραπάνω ιστογράμματα, ποσοτικοποιείται παρακάτω υπολογίζοντας τις τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών.

Όπως και στο προηγούμενο κεφάλαιο, στα αληθινά θετικά (TP) ανήκουν οι εικόνες που είναι φυσιολογικές και μέσω του μοντέλου χαρακτηρίστηκαν και πάλι φυσιολογικές, στα αληθινά αρνητικά (TN) ανήκουν οι εικόνες που είναι μη φυσιολογικές και χαρακτηρίστηκαν στη συνέχεια μη φυσιολογικές. Παράλληλα, τα ψευδώς θετικά στοιχεία είναι οι εικόνες που είναι στην πραγματικότητα μη φυσιολογικές αλλά χαρακτηρίστηκαν φυσιολογικές, ενώ τα ψευδώς αρνητικά στοιχεία είναι οι εικόνες που είναι φυσιολογικές αλλά χαρακτηρίστηκαν μη φυσιολογικές.

True Positive (TP)	1259
False Positive (FP)	183
True Negative (TN)	2134
False Negative (FN)	156

Πίνακας 4: Πίνακας παραμέτρων TP, FP, TN, FN



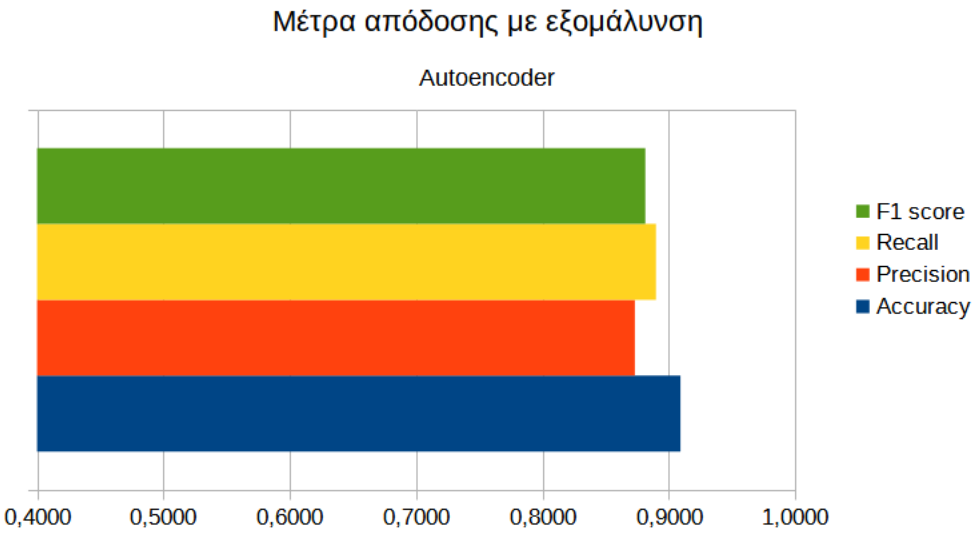
Εικόνα 39: Διαγραμματική παρουσίαση των τιμών TP, FP, TN, FN

Συγκρίνοντας τις τιμές αυτές με τις αντίστοιχες τιμές των ίδιων παραμέτρων στην περίπτωση των κανονικών δεδομένων παρατηρείται ότι το πλήθος των αληθινά θετικών και των ψευδώς αρνητικών έμεινε πρακτικά ίδιο, πράγμα που σημαίνει ότι το μοντέλο δεν συνάντησε επιπλέον δυσκολία στον εντοπισμό των φυσιολογικών δεδομένων, όπως αναμενόταν. Αντίθετα, φαίνεται το πλήθος των ψευδώς θετικών να υπερδιπλασιάστηκε. Το γεγονός αυτό δείχνει ότι πολλές μη φυσιολογικές εικόνες θεωρήθηκαν φυσιολογικές λόγω της εξομάλυνσης, η οποία απέκρυψε τις υπάρχουσες ρωγμές και την υπάρχουσα τραχύτητα.

Με βάση τις τιμές των παραπάνω παραμέτρων υπολογίζονται τα μέτρα απόδοσης accuracy, precision, recall και F1 score.

Accuracy	0,9092
Precision	0,8731
Recall	0,8898
F1 Score	0,8813

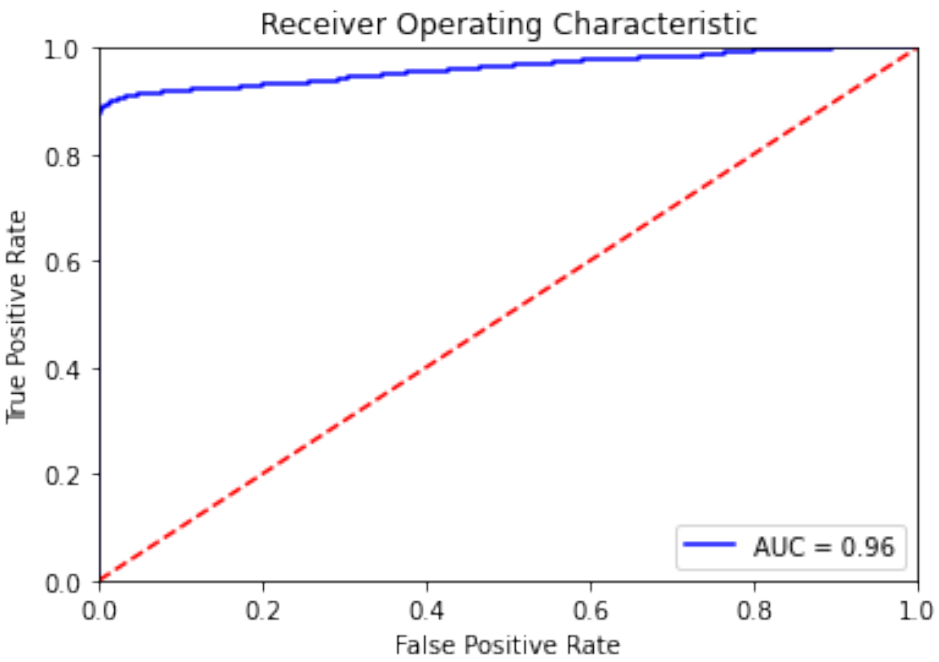
Πίνακας 5: Μέτρα απόδοσης με εξομάλυνση



Εικόνα 40: Διαγραμματική παρουσίαση των μέτρων απόδοσης με εξομάλυνση

Όπως αναμενόταν παρατηρείται μια μικρή μείωση στις τιμές των accuracy, precision και F1 score, ενώ το recall παρέμεινε σχεδόν στην ίδια τιμή αφού εξαρτάται από την επιτυχία στην πρόβλεψη των φυσιολογικών εικόνων. Παρ' όλο που οι τιμές μειώθηκαν, διατηρήθηκαν σε αρκετά υψηλά επίπεδα και έτσι θεωρείται ότι ο αυτόματος κωδικοποιητής διαχώρισε επιτυχώς τις εικόνες που έχουν υποστεί εξομάλυνση.

Το συμπέρασμα αυτό επιβεβαιώνεται και από την καμπύλη ROC που φαίνεται παρακάτω αφού η παράμετρος Area Under the Curve (AUC) ισούται με 0,96. Μειώθηκε δηλαδή μόνο κατά 0,02 σε σχέση με την αντίστοιχη τιμή για τα κανονικά δεδομένα, δείχνοντας έτσι ότι οι δύο κλάσεις (φυσιολογικές και μη φυσιολογικές εικόνες) είναι ξεκάθαρα διαχωρίσιμες ακόμα και μετά την εξομάλυνση.



Εικόνα 41: Καμπύλη ROC μετά την εξομάλυνση

β) Θόρυβος

Σε συμφωνία με τα προηγούμενα, στην ενότητα αυτή εξετάζεται η απόδοση του αυτόματου κωδικοποιητή αν εκπαιδευτεί και ελεγχθεί με δεδομένα που περιέχουν θόρυβο, συγκεκριμένα θόρυβο τύπου αλάτι – πιπέρι. Ο θόρυβος εισάγεται στις εικόνες μέσω της συνάρτησης "sp_noise" η οποία περιγράφηκε σε προηγούμενη ενότητα.

Οι αρχικές εικόνες εκπαίδευσης, οι οποίες είναι και πάλι αποκλειστικά φυσιολογικές, δόθηκαν σαν είσοδο στη συνάρτηση και έτσι προέκυψε το σύνολο εκπαίδευσης που θα χρησιμοποιηθεί στην παρούσα φάση. Το σύνολο αυτό, αφού μετατράπηκε από απλή λίστα σε numpy array και αναδιαμορφώθηκε ώστε να έχει τις κατάλληλες διαστάσεις, αξιοποιήθηκε στην εκπαίδευση του μοντέλου, η οποία πραγματοποιήθηκε και πάλι σε 5 εποχές.

```
n_x_train = np.array(noisy_clear_train)
n_x_train = np.reshape(n_x_train, (-1, 45, 45, 1))
n_model1 = autoencoder()
n_model1.fit(n_x_train, n_x_train, batch_size=10, epochs=5)
```

```
Epoch 1/5
100/100 [==============================] - 26s 217ms/step - loss: 0.0544 - mean_squared_error: 0.0544
Epoch 2/5
100/100 [==============================] - 22s 217ms/step - loss: 0.0370 - mean_squared_error: 0.0370
Epoch 3/5
100/100 [==============================] - 22s 218ms/step - loss: 0.0353 - mean_squared_error: 0.0353
Epoch 4/5
100/100 [==============================] - 22s 218ms/step - loss: 0.0357 - mean_squared_error: 0.0357
Epoch 5/5
100/100 [==============================] - 22s 217ms/step - loss: 0.0357 - mean_squared_error: 0.0357
<keras.callbacks.History at 0x7f60ee0f9f90>
```

Εικόνα 42: Διαδικασία εκπαίδευσης αυτόματου κωδικοποιητή με δεδομένα με θόρυβο

Ολοκληρώνοντας την εκπαίδευση του μοντέλου, ακολουθείται όμοια διαδικασία με τα προηγούμενα για τον έλεγχο της απόδοσής του, προσθέτοντας όμως θόρυβο τύπου αλάτι – πιπέρι στα δεδομένα ελέγχου. Αφού διαβαστούν οι εικόνες και προστεθεί ο θόρυβος, 20 από αυτές (10 φυσιολογικές και 10 μη φυσιολογικές) χρησιμοποιούνται για τον υπολογισμό του κατωφλίου. Συγκεκριμένα, ζητείται από το μοντέλο να πραγματοποιήσει πρόβλεψη για τις εικόνες αυτές και στη συνέχεια υπολογίζεται το μέσο τετραγωνικό σφάλμα της κάθε εικόνας του συνόλου ελέγχου και της αναπαράστασής της από το μοντέλο. Με αυτό τον τρόπο είναι δυνατό να υπολογιστούν η μέση τιμή και η τυπική απόκλιση για τις φυσιολογικές και τις μη φυσιολογικές εικόνες και να υπολογιστεί το κατώφλι.

Μέση τιμή (mean)	Φυσιολογικές (noisy clear)	0.005014607364026715 ≃ 0.00501
	Μη φυσιολογικές (noisy cracked)	0.004554817323029707 ≃ 0.00455
Τυπική απόκλιση (standard deviation)	Φυσιολογικές (noisy clear)	0.0001638047247538428 ≃ 0.00016
	Μη φυσιολογικές (noisy cracked)	0.0000897155369755263 ≃ 0.00009

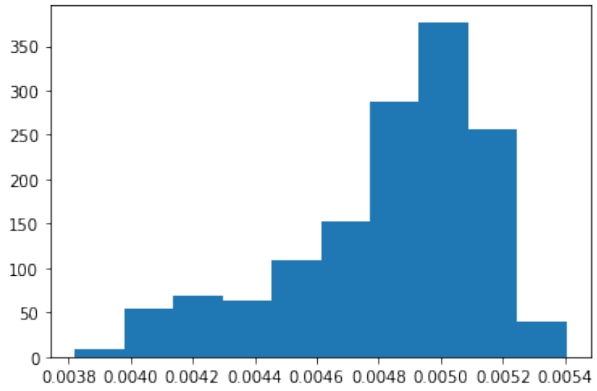
Πίνακας 6: Μέση τιμή και τυπική απόκλιση του δείγματος 20 εικόνων με θόρυβο

Παρατηρώντας τις τιμές αυτές φαίνεται ότι η μέση τιμή του μέσου τετραγωνικού σφάλματος των μη φυσιολογικών εικόνων είναι μικρότερη από την αντίστοιχη τιμή των φυσιολογικών εικόνων, ενώ θα αναμενόταν το αντίθετο. Από το γεγονός αυτό φαίνεται ότι η τελική ακρίβεια της ταξινόμησης θα είναι σίγουρα μειωμένη σε σχέση με τις προηγούμενες περιπτώσεις.

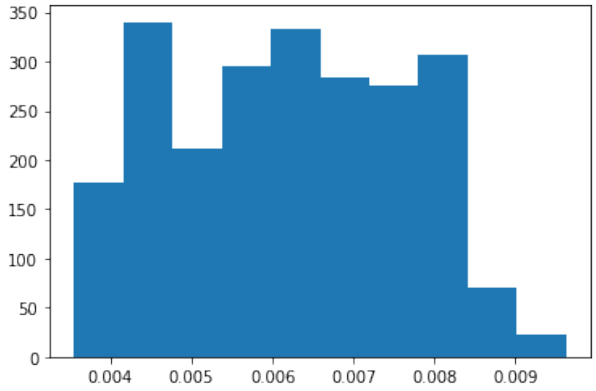
Από τις παραπάνω τιμές υπολογίζεται η τιμή του κατωφλίου:

$$threshold_{noisy} = mean_{noisy-clear} + std_{noisy-clear} = 0.005178412088780558 \simeq 0.00518$$

Αφού προσδιορίστηκε η τιμή του κατωφλίου, εισάγονται στο μοντέλο όλες οι εικόνες ελέγχου, ζητείται να αναπαραχθούν και στη συνέχεια υπολογίζεται το μέσο τετραγωνικό σφάλμα για κάθε εικόνα. Παρακάτω φαίνονται τα αντίστοιχα ιστογράμματα.



Εικόνα 43: Ιστόγραμμα μέσου τετραγωνικού σφάλματος για τις φυσιολογικές εικόνες με θόρυβο

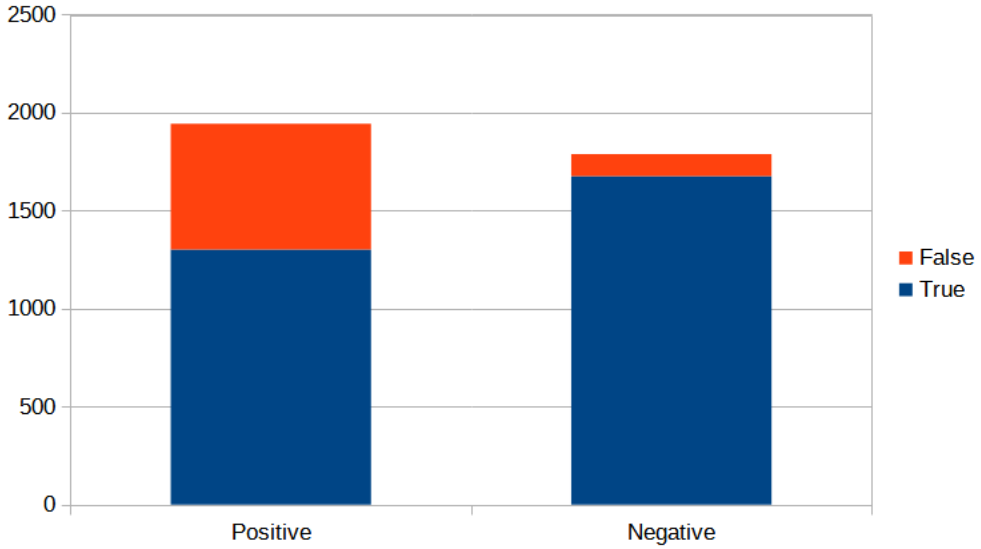


Εικόνα 44: Ιστόγραμμα μέσου τετραγωνικού σφάλματος για τις μη φυσιολογικές εικόνες με θόρυβο

Με την τιμή του κατωφλίου να βρίσκεται κοντά στο 0,0052, παρατηρώντας τα ιστογράμματα φαίνεται ότι ένα σημαντικό μέρος των μη φυσιολογικών εικόνων ταξινομείται στις φυσιολογικές. Παράλληλα, το μέρος των φυσιολογικών που φαίνεται να ταξινομούνται στις μη φυσιολογικές είναι σημαντικό αλλά αρκετά μικρότερο. Αυτό φαίνεται και από τις τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών που φαίνονται στον παρακάτω πίνακα.

True Positive (TP)	1302
False Positive (FP)	641
True Negative (TN)	1676
False Negative (FN)	113

Πίνακας 7: Πίνακας παραμέτρων TP, FP, TN, FN

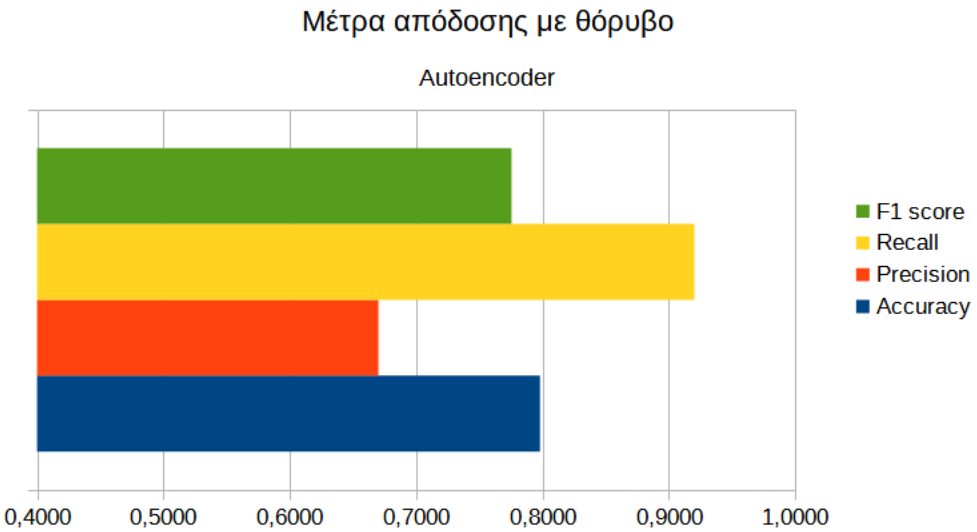


Εικόνα 45: Διαγραμματική παρουσίαση των τιμών TP, FP, TN, FN

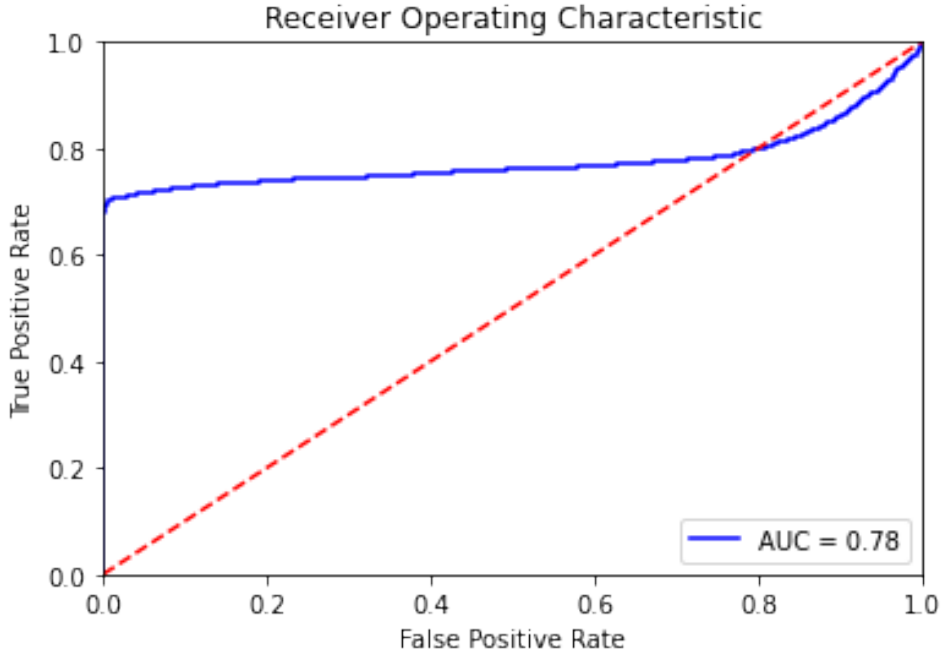
Παρατηρώντας το παραπάνω διάγραμμα φαίνεται ότι σχεδόν το $\frac{1}{3}$ των εικόνων που χαρακτηρίστηκαν θετικές (δηλαδή φυσιολογικές), είναι ψευδώς θετικές. Οι εικόνες αυτές αποτελούν περιπτώσεις ρωγμών και άλλων ασυνεχειών που δεν εντοπίστηκαν από το μοντέλο, πράγμα που δείχνει ότι με την εισαγωγή του θορύβου, η ικανότητα του μοντέλου να εντοπίσει τις ρωγμές μειώθηκε σημαντικά. Το γεγονός αυτό απεικονίζεται και στις παραμέτρους accuracy, precision, recall και F1 score, καθώς και στην καμπύλη ROC στην οποία το AUC μειώθηκε σημαντικά, δείχνοντας έτσι τη μειωμένη ικανότητα διάκρισης μεταξύ των δύο κλάσεων.

Accuracy	0,9092
Precision	0,8731
Recall	0,8898
F1 Score	0,8813

Πίνακας 8: Μέτρα απόδοσης με θόρυβο

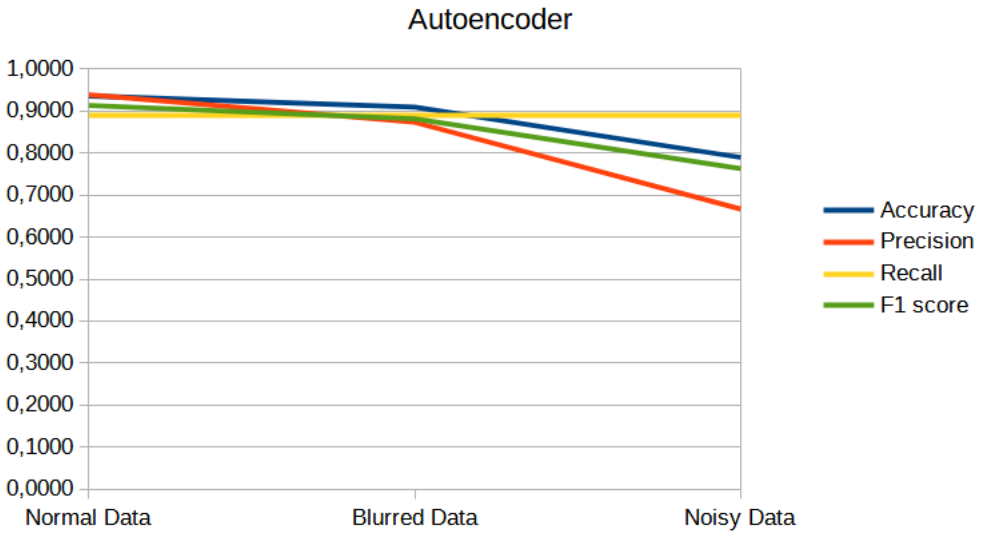


Εικόνα 46: Διαγραμματική παρουσίαση των μέτρων απόδοσης με θόρυβο



Εικόνα 47: Καμπύλη ROC μετά την προσθήκη θορύβου

γ) Σύγκριση αποτελεσμάτων για τον αυτόματο κωδικοποιητή



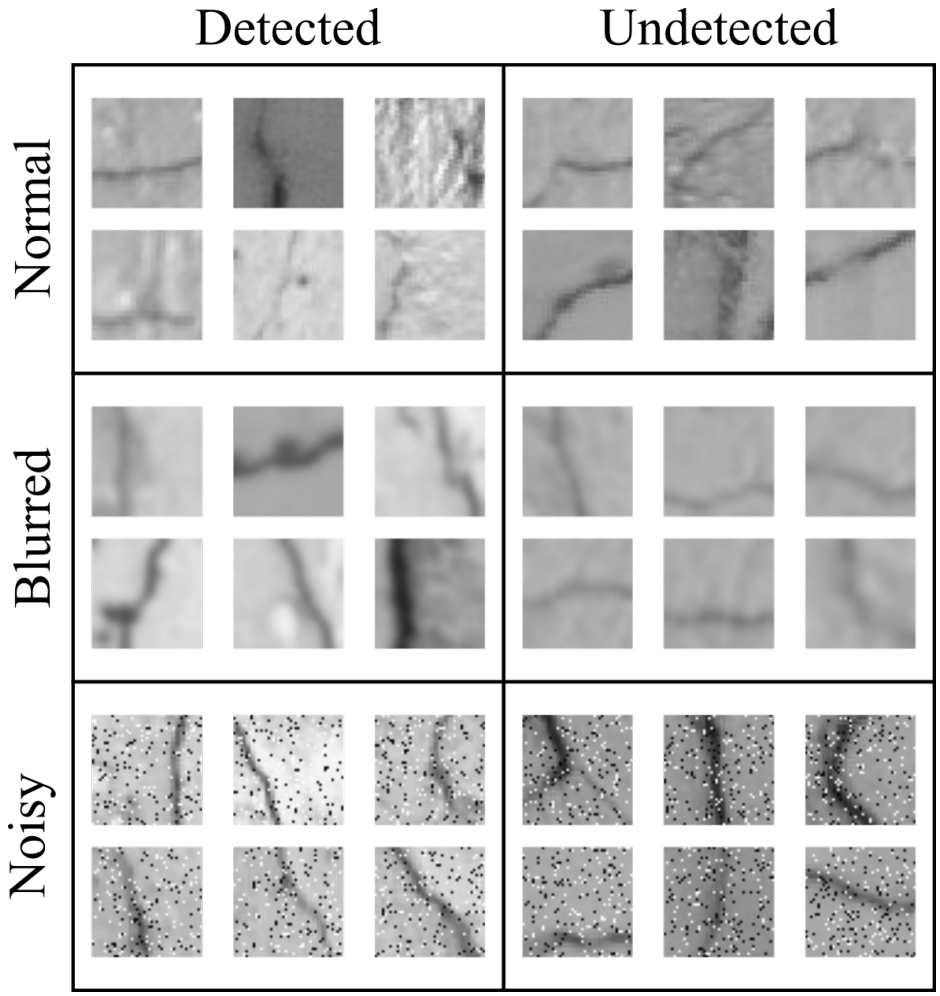
Εικόνα 48: Συγκεντρωτικό διάγραμμα μέτρων απόδοσης για τον αυτόματο κωδικοποιητή

Στο παραπάνω διάγραμμα παρουσιάζονται τα μέτρα απόδοσης accuracy, precision, recall και F1 score στην περίπτωση των κανονικών δεδομένων, στην περίπτωση των δεδομένων που έχουν υποστεί εξομάλυνση και στην περίπτωση των δεδομένων με θόρυβο.

- Accuracy: Η παράμετρος accuracy λαμβάνει υπόψιν για τον υπολογισμό της τα αληθινά θετικά, τα αληθινά αρνητικά, τα ψευδώς θετικά και τα ψευδώς αρνητικά στοιχεία με αποτέλεσμα να προσφέρει σφαιρική εκτίμηση της ποιότητας του διαχωρισμού. Η μέγιστη τιμή της παραμέτρου παρατηρείται με χρήση των κανονικών δεδομένων, όπως ήταν αναμενόμενο, αφού στα κανονικά δεδομένα οι διαφορές μεταξύ φυσιολογικών και μη φυσιολογικών εικόνων είναι εύκολα εμφανείς. Στην περίπτωση των δεδομένων που έχουν υποστεί εξομάλυνση, η τιμή του accuracy δεν σημείωσε σημαντική πτώση, αν και με την εξομάλυνση οι μη φυσιολογικές εικόνες τείνουν να μοιάζουν περισσότερο με τις φυσιολογικές. Αντίθετα, στην περίπτωση των δεδομένων με θόρυβο παρατηρείται σημαντική πτώση η οποία όπως φάνηκε στην προηγούμενη ενότητα οφείλεται στην λανθασμένη κατηγοριοποίηση μεγάλου μέρους των μη φυσιολογικών εικόνων.
- Precision: Η παράμετρος precision λαμβάνει υπόψιν τα στοιχεία που έχουν κατηγοριοποιηθεί σαν θετικά (φυσιολογικά), είτε αληθώς, είτε ψευδώς. Η μέγιστη τιμή της παρατηρείται και πάλι στα κανονικά δεδομένα και παρουσιάζει μικρή πτώση στην περίπτωση της εξομάλυνσης που οφείλεται στην αύξηση των ψευδώς θετικών στοιχείων. Η σημαντικότερη πτώση παρατηρείται στην περίπτωση των δεδομένων με θόρυβο και οφείλεται και πάλι στον σημαντικό αριθμό ψευδώς θετικών στοιχείων.
- Recall: Η παράμετρος recall είναι η μοναδική που δεν φαίνεται να παρουσιάζει μείωση. Αυτό οφείλεται στο γεγονός ότι εξαρτάται από την επιτυχία εντοπισμού των φυσιολογικών εικόνων, αφού λαμβάνει υπόψιν τα αληθινά θετικά και τα ψευδώς αρνητικά στοιχεία. Η ικανότητα του μοντέλου να διακρίνει φυσιολογικές εικόνες δεν μεταβλήθηκε με την αλλαγή δεδομένων, οπότε είναι λογικό να παρατηρείται σταθερότητα στην τιμή του recall.

- F1 score: το F1 score υπολογίζεται με χρήση της τιμής του precision και της τιμής του recall., λαμβάνοντας με αυτό τον τρόπο υπόψιν και τους τέσσερις αριθμούς TP, TN, FP, FN και προσφέροντας μια γενική εκτίμηση της ποιότητας της ταξινόμησης, όπως στην περίπτωση του accuracy. Το F1 score φαίνεται να ακολουθεί παράλληλη πορεία με το accuracy, με τη μέγιστη τιμή του να παρατηρείται στα κανονικά δεδομένα, μία μικρή πτώση στην περίπτωση της εξομάλυνσης και έντονη πτώση στην περίπτωση του θορύβου λόγω του μεγάλου αριθμού ψευδώς θετικών στοιχείων.

Στην εικόνα που ακολουθεί φαίνονται ενδεικτικά μερικές εικόνες στις οποίες εντοπίστηκαν επιτυχημένα ρωγμές και μερικές τις οποίες το μοντέλο αγνόησε. Στην περίπτωση των κανονικών δεδομένων, το μοντέλο φαίνεται να αγνόησε την ύπαρξη ρωγμών σε εικόνες στις οποίες το περιβάλλον της ρωγμής είναι ιδιαίτερα ομοιόμορφο, με αποτέλεσμα το αντίστοιχο μέσο τετραγωνικό σφάλμα να προκύπτει μικρότερο από το κατώφλι. Παράλληλα, στην περίπτωση των δεδομένων με εξομάλυνση, οι εικόνες στις οποίες αγνοήθηκαν οι ρωγμές φαίνεται να είναι οι εικόνες στις οποίες αυτές είναι δυσδιάκριτες και δεν ξεχωρίζουν ιδιαίτερα από το περιβάλλον τους. Τέλος, στην περίπτωση ύπαρξης θορύβου, στις εικόνες που εντοπίστηκαν ρωγμές αλλά και σε αυτές που δεν εντοπίστηκαν, οι ρωγμές είναι εμφανείς, επιβεβαιώνοντας το συμπέρασμα ότι σε κάποιες περιπτώσεις ο θόρυβος οδήγησε το μοντέλο σε λανθασμένη ταξινόμηση.



Εικόνα 49: Ενδεικτικές εικόνες με ρωγμές που εντόπισε και που δεν εντόπισε ο αυτόματος κωδικοποιητής

5.3 K-means

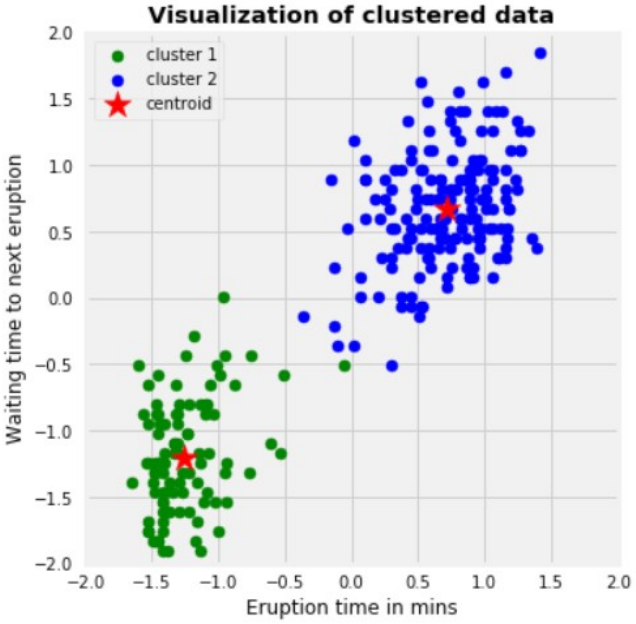
α) Γενικά

Ομαδοποίηση είναι η διαδικασία εντοπισμού υποομάδων στα δεδομένα ώστε τα σημεία δεδομένων που ανήκουν στην ίδια υποομάδα να είναι πολύ παρόμοια, ενώ τα σημεία δεδομένων που ανήκουν σε διαφορετικές υποομάδες να είναι πολύ διαφορετικά. Έτσι, ο σκοπός είναι να βρεθούν ομοιογενείς υποομάδες μέσα στα δεδομένα ώστε τα σημεία δεδομένων να είναι όσο το δυνατόν πιο παρόμοια με βάση ένα μέτρο ομοιότητας, όπως η ευκλείδεια απόσταση. Η ομαδοποίηση είναι μη επιβλεπόμενη μέθοδος μάθησης και ένας από τους πιο συχνά χρησιμοποιούμενους αλγορίθμους είναι ο K-means. [48]

Ο αλγόριθμος K-means είναι ένας επαναληπτικός αλγόριθμος, ο οποίος προσπαθεί να χωρίσει το σύνολο δεδομένων σε K διακριτές και μη επικαλυπτόμενες υποομάδες (clusters), με το κάθε σημείο δεδομένων να ανήκει μόνο σε μία υποομάδα. Σκοπός του είναι να αναθέσει τα σημεία δεδομένων σε υποομάδες έτσι ώστε το άθροισμα της τετραγωνικής απόστασης μεταξύ των σημείων δεδομένων και του κεντροειδούς της ομάδας να είναι το ελάχιστο. [48]

Ένα μειονέκτημα του αλγορίθμου είναι ότι το πλήθος των κλάσεων K θα πρέπει να είναι εκ των προτέρων γνωστό. Ο τρόπος λειτουργίας του K-means ακολουθεί τα παρακάτω βήματα:

1. Θεωρούνται τυχαία K κεντροειδή, με K το πλήθος των κλάσεων.
2. Τα δεδομένα ομαδοποιούνται ως προς τα κεντροειδή αυτά με βάση κάποιο μέτρο ομοιότητας, όπως η ευκλείδεια απόσταση.
3. Για κάθε ομάδα που δημιουργήθηκε υπολογίζεται το νέο κεντροειδές
4. Αν τα νέα κεντροειδή ταυτίζονται με τα προηγούμενα, τότε η διαδικασία ολοκληρώνεται. Αλλιώς, η διαδικασία επαναλαμβάνεται με αφετηρία το δεύτερο βήμα.



Εικόνα 50: Παράδειγμα ομαδοποιημένων δεδομένων

Πηγή: *https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a*

β) Εφαρμογή K-means στα κανονικά δεδομένα

Με σκοπό να ελεγχθεί η απόδοση του αυτόματου κωδικοποιητή, κρίθηκε απαραίτητο να εφαρμοστούν άλλοι αλγόριθμοι, όπως ο K-means, στα ίδια δεδομένα και να συγκριθούν οι αποδόσεις τους. Στην ενότητα αυτή ο K-means εφαρμόζεται στα δεδομένα που χρησιμοποιήθηκαν για τον έλεγχο του αυτόματου κωδικοποιητή στο 4ο κεφάλαιο της παρούσας εργασίας. Τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση του αυτόματου κωδικοποιητή δεν θα αξιοποιηθούν στην παρούσα φάση, αφού ο αλγόριθμος δεν χρειάζεται εκπαίδευση και είναι επιθυμητό να εξεταστεί η απόδοσή του στα δεδομένα με τα οποία αξιολογήθηκε το μοντέλο στις προηγούμενες ενότητες.

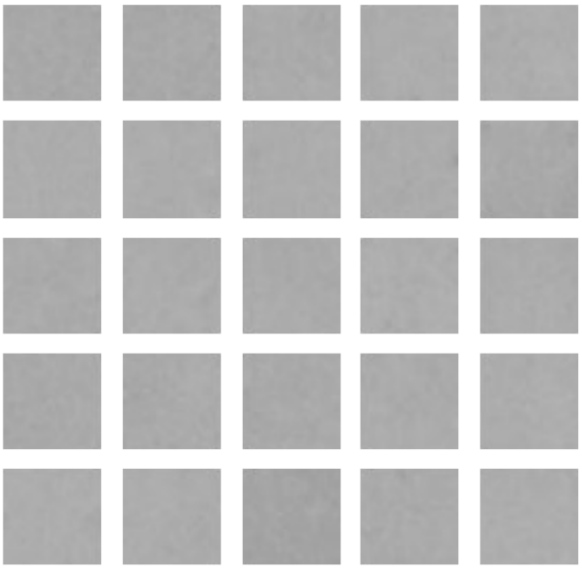
Αρχικά τα δεδομένα διαβάστηκαν από τους αντίστοιχους συμπιεσμένους φακέλους με όμοιο τρόπο με τα προηγούμενα, χωρίς αυτή τη φορά να εφαρμοστεί κανονικοποίηση. Δημιουργήθηκαν ετικέτες για την αξιολόγηση της απόδοσης του μοντέλου με το 0 να αντιστοιχεί στις φυσιολογικές εικόνες και το 1 να αντιστοιχεί στις μη φυσιολογικές. Όλες οι εικόνες, φυσιολογικές και μη φυσιολογικές, οργανώθηκαν σε μία λίστα που ονομάστηκε "data" και η λίστα αυτή στη συνέχεια μετατράπηκε σε numpy array.

Οι διαστάσεις του numpy array ήταν αρχικά (3732, 45, 45), με τον πρώτο αριθμό να υποδεικνύει το πλήθος των εικόνων και τους επόμενους τις διαστάσεις τους. Στη συνέχεια, το array αναδιαμορφώθηκε ώστε οι διαστάσεις του να είναι (3723, 2025). Με αυτό τον τρόπο η κάθε εικόνα μετατράπηκε σε μία γραμμή που περιλαμβάνει 2025 στοιχεία. Η κάθε σειρά αποτελεί ένα αντικείμενο και οι ιδιότητες του αντικειμένου είναι οι τόνοι του γκρι σε κάθε εικονοστοιχείο του.

Η ομαδοποίηση των δεδομένων πραγματοποιήθηκε μέσω της συνάρτησης "KMeans" της βιβλιοθήκης "Scikit-learn".

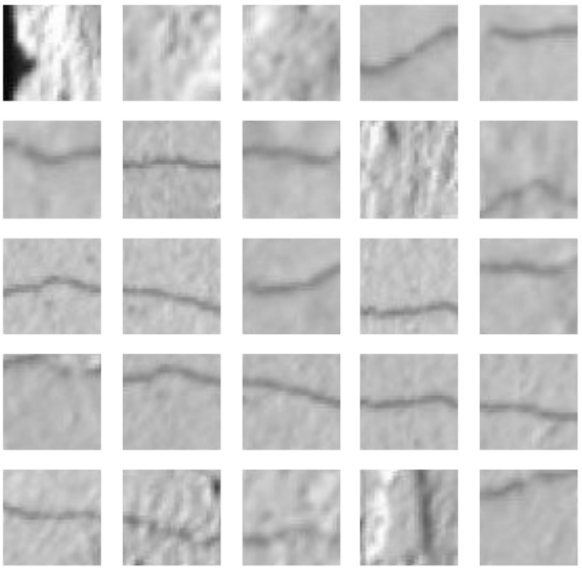
```
n = 2 # number of clusters
kmeans = KMeans(n_clusters=n,init='random')
kmeans.fit(data)
Z = kmeans.predict(data)
```

Αρχικά, ορίστηκε το πλήθος των κλάσεων, το οποίο στη συγκεκριμένη περίπτωση είναι 2, και ορίστηκε ότι η αρχικοποίηση θα πραγματοποιηθεί με τυχαίο τρόπο. Στη συνέχεια, έγινε εισαγωγή των δεδομένων στη συνάρτηση "KMeans" και ζητήθηκε να πραγματοποιηθούν προβλέψεις. Η ανάθεση των ετικετών 0 και 1 στην εκάστοτε κλάση από τον αλγόριθμο πραγματοποιείται με τυχαίο τρόπο, δηλαδή η κλάση 0 που προέκυψε μετά την εφαρμογή του αλγορίθμου δεν ταυτίζεται απαραίτητα με τα φυσιολογικά δεδομένα. Μετά από σχετικό έλεγχο διαπιστώθηκε ότι στη συγκεκριμένη περίπτωση όντως η κλάση 0 του αλγορίθμου αντιστοιχεί στα φυσιολογικά δεδομένα.



Εικόνα 51: Κλάση 0 (φυσιολογικές εικόνες) K-means

Στις διπλανές εικόνες απεικονίζονται μερικές από τις εικόνες που ανατέθηκαν στην κάθε κλάση.



Εικόνα 52: Κλάση 1 (μη φυσιολογικές εικόνες) K-means

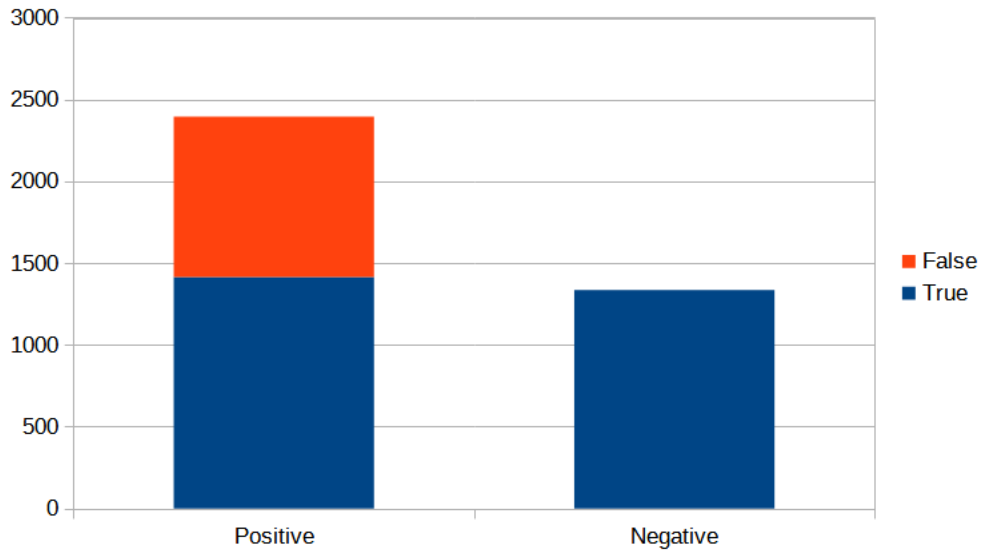
Παρατηρώντας αυτό το μικρό υποσύνολο των εικόνων φαίνεται ο διαχωρισμός να έχει επιτευχθεί και ο αλγόριθμος να κατάφερε να ομαδοποιήσει τις φυσιολογικές και τις μη φυσιολογικές εικόνες αντίστοιχα. Έχοντας τις πραγματικές ετικέτες των εικόνων είναι δυνατό να ελεγχθεί με ποσοτικό τρόπο η επιτυχία της ομαδοποίησης. Συγκεκριμένα, είναι

δυνατό να υπολογιστεί το πλήθος των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών στοιχείων. Όπως και στα προηγούμενα, αληθινά θετικά στοιχεία θεωρούνται οι εικόνες που είναι φυσιολογικές και ομαδοποιήθηκαν στην κλάση των φυσιολογικών, αληθινά αρνητικά στοιχεία είναι οι εικόνες που είναι μη φυσιολογικές και ομαδοποιήθηκαν στην κλάση των μη φυσιολογικών, ενώ ψευδώς θετικά και ψευδώς αρνητικά είναι τα στοιχεία που δεν κατέληξαν στις σωστές κλάσεις.

True Positive (TP)	1415
False Positive (FP)	981
True Negative (TN)	1336
False Negative (FN)	0

Πίνακας 9: Πίνακας παραμέτρων TP, TN, FP, FN για εφαρμογή K-means σε κανονικά δεδομένα

Από τις τιμές των TP, FP, TN, FN που φαίνονται στον παραπάνω πίνακα και σχηματικά στο παρακάτω διάγραμμα, φαίνεται ότι ο αλγόριθμος κατάφερε με μεγάλη επιτυχία να ομαδοποιήσει μαζί όλες τις φυσιολογικές εικόνες χωρίς κανένα λάθος, πράγμα που φαίνεται από τη μηδενική τιμή των ψευδώς αρνητικών στοιχείων. Αντίθετα, όσον αφορά τις μη φυσιολογικές εικόνες δεν παρατηρείται εξίσου εντυπωσιακή απόδοση, αποτυγχάνοντας στην επιλογή κλάσης για τα $\frac{2}{5}$ των μη φυσιολογικών εικόνων, οι οποίες συσσωρεύτηκαν σε μία κλάση μαζί με τις φυσιολογικές. Σημειώνεται ότι το πλήθος των ψευδώς θετικών στην περίπτωση αυτή ξεπερνάει κατά πολύ το αντίστοιχο πλήθος στην περίπτωση του αυτόματου κωδικοποιητή, ακόμα και στην περίπτωση των δεδομένων με θόρυβο, όπου και παρουσίασε τη χειρότερη απόδοσή του.



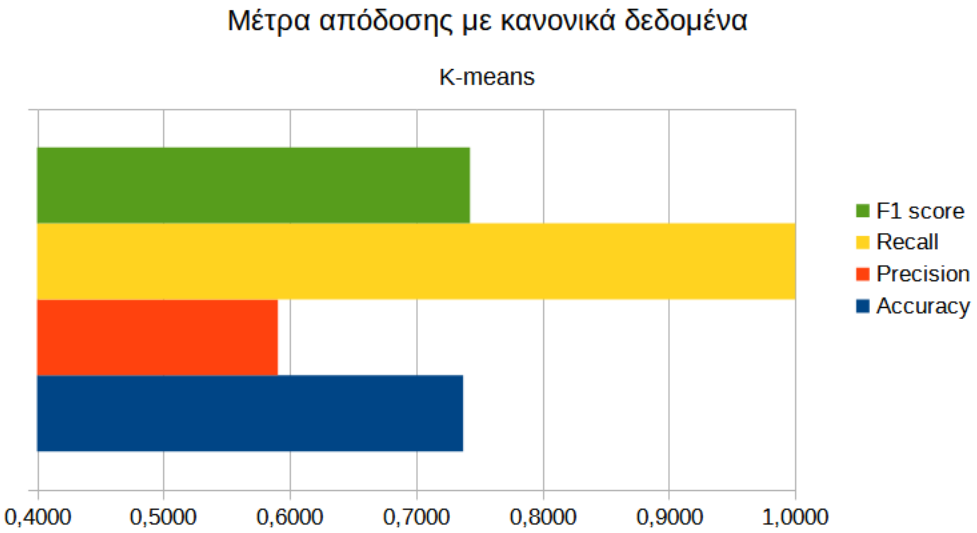
Εικόνα 53: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή K-means σε κανονικά δεδομένα

Από τις τιμές των TP, TN, FP και FN, όπως και στα προηγούμενα, υπολογίζονται τα μέτρα απόδοσης accuracy, precision, recall και F1 score, τα οποία συνοψίζουν με ποσοτικό τρόπο τις παραπάνω παρατηρήσεις.

Accuracy	0,7371
Precision	0,5906
Recall	1,000
F1 Score	0,7426

Πίνακας 10: Μέτρα απόδοσης K-means με κανονικά δεδομένα

Συγκεκριμένα από τον παραπάνω πίνακα και το διάγραμμα που ακολουθεί επιβεβαιώνονται όσα παρατηρήθηκαν παραπάνω. Αρχικά, βλέποντας το accuracy και το F1 score κοντά στο 74%, δίνεται η εντύπωση ότι ο αλγόριθμος είχε μέτρια απόδοση και σε γενικές γραμμές επιτεύχθηκε ικανοποιητική ομαδοποίηση. Λαμβάνοντας όμως υπόψιν την τιμή του precision φαίνεται ότι ο αλγόριθμος δεν κατάφερε να ταξινομήσει στη σωστή κλάση ένα μεγάλο μέρος των μη φυσιολογικών δεδομένων, ο εντοπισμός των οποίων είναι ο βασικός σκοπός της παρούσας εργασίας. Έτσι, αφού ο αλγόριθμος K-means απέτυχε στην ομαδοποίηση σχεδόν των μισών μη φυσιολογικών εικόνων με τα κανονικά δεδομένα, η λύση του αυτόματου κωδικοποιητή δείχνει να είναι η πλέον συμφέρουσα. Στη συνέχεια ο αλγόριθμος K-means θα δοκιμαστεί σε συνθήκες εξομάλυνσης και θορύβου.



Εικόνα 54: Διαγραμματική παρουσίαση των μέτρων απόδοσης του K-means με κανονικά δεδομένα

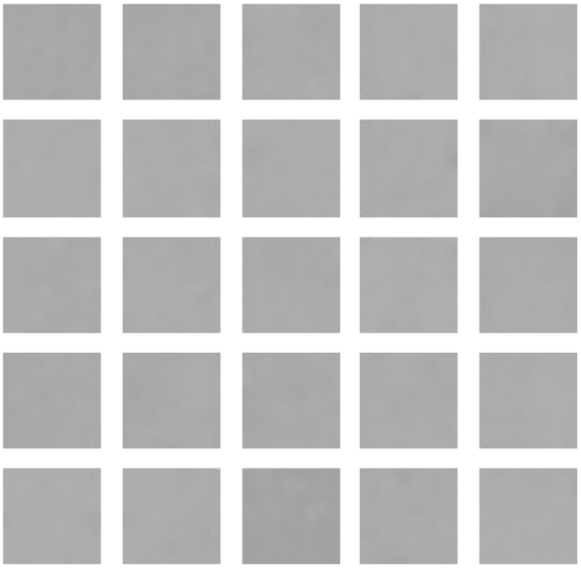
γ) Εφαρμογή K-means σε δεδομένα με εξομάλυνση

Για τη δοκιμή του αλγορίθμου K-means σε δεδομένα με εξομάλυνση χρησιμοποιήθηκαν τα ίδια δεδομένα στα οποία δοκιμάστηκε και ο αυτόματος κωδικοποιητής. Τα δεδομένα αυτά δηλαδή είναι τα

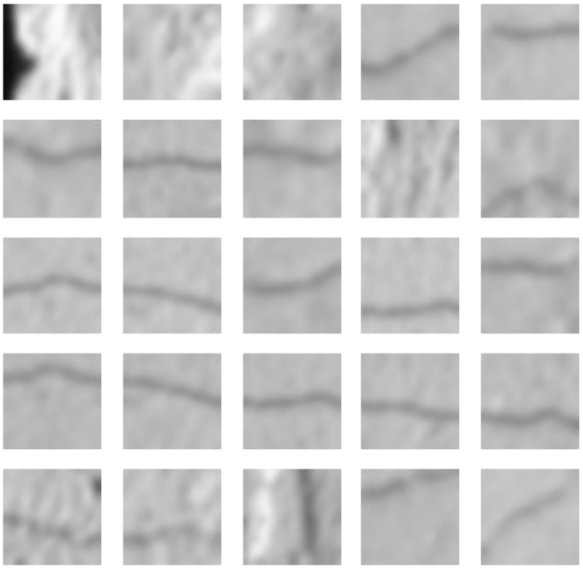
κανονικά δεδομένα ελέγχου, στα οποία όμως έγινε εξομάλυνση μέσω του φίλτρου Gauss, με τις ίδιες παραμέτρους με την περίπτωση του αυτόματου κωδικοποιητή.

Αφού διαβάστηκαν οι εικόνες από τον αντίστοιχο φάκελο και εφαρμόστηκε το φίλτρο Gauss, οργανώθηκαν στη λίστα “b_data” και δημιουργήθηκαν οι αντίστοιχες αληθείς ετικέτες. Στη συνέχεια η λίστα μετατράπηκε σε numpy array με διαστάσεις (3732, 45, 45) και αναδιαμορφώθηκε ώστε οι νέες διαστάσεις να είναι (3732, 2025). Έπειτα, με χρήση της συνάρτησης “KMeans”, με όμοιο τρόπο με τα προηγούμενα, ζητήθηκε από τον αλγόριθμο να ομαδοποιήσει τα δεδομένα αυτά. Αυτή τη φορά παρατηρήθηκε ότι η ετικέτα 1 αντιστοιχεί στην κλάση που περιλαμβάνει τα φυσιολογικά δεδομένα, ενώ η ετικέτα 0 αντιστοιχεί στην κλάση που περιλαμβάνει τα μη φυσιολογικά δεδομένα, σε αντίθεση με τις αληθείς ετικέτες.

Οι εικόνες που ακολουθούν απεικονίζουν ένα μικρό υποσύνολο των δύο κλάσεων, στο οποίο φαίνεται να έχει πραγματοποιηθεί επιτυχημένα η ομαδοποίηση, αφού στην πρώτη εικόνα περιλαμβάνονται μόνο φυσιολογικές εικόνες, ενώ στη δεύτερη εικόνα μόνο μη φυσιολογικές. Αντικειμενική εκτίμηση της ποιότητας της ομαδοποίησης προκύπτει στη συνέχεια με ποσοτικό τρόπο αφού υπολογίζεται το πλήθος των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών στοιχείων καθώς και τα μέτρα απόδοσης accuracy, precision, recall και F1 score, όπως και στις προηγούμενες περιπτώσεις.



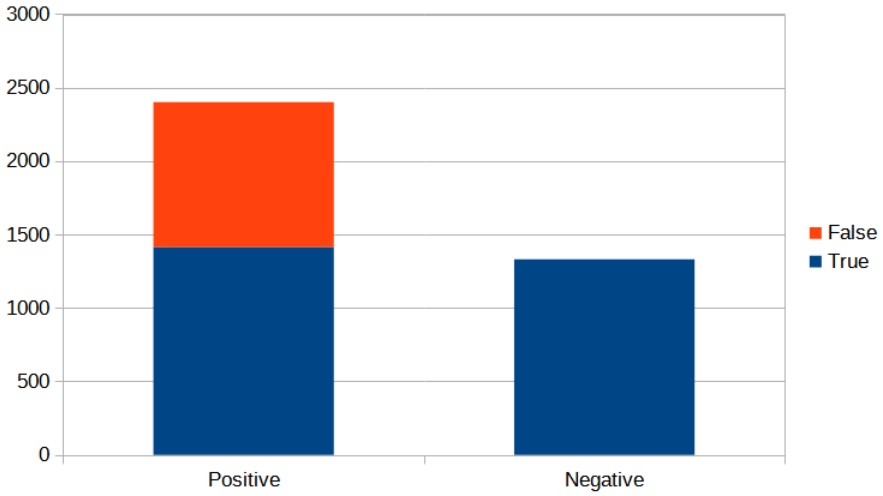
Εικόνα 55: Κλάση 1 (φυσιολογικές εικόνες) K-means με εξομάλυνση



Εικόνα 56: Κλάση 0 (μη φυσιολογικές εικόνες) K-means με εξομάλυνση

True Positive (TP)	1415
False Positive (FP)	987
True Negative (TN)	1330
False Negative (FN)	0

Πίνακας 11: Πίνακας TP, TN, FP, FN για εφαρμογή K-means σε δεδομένα με εξομάλυνση

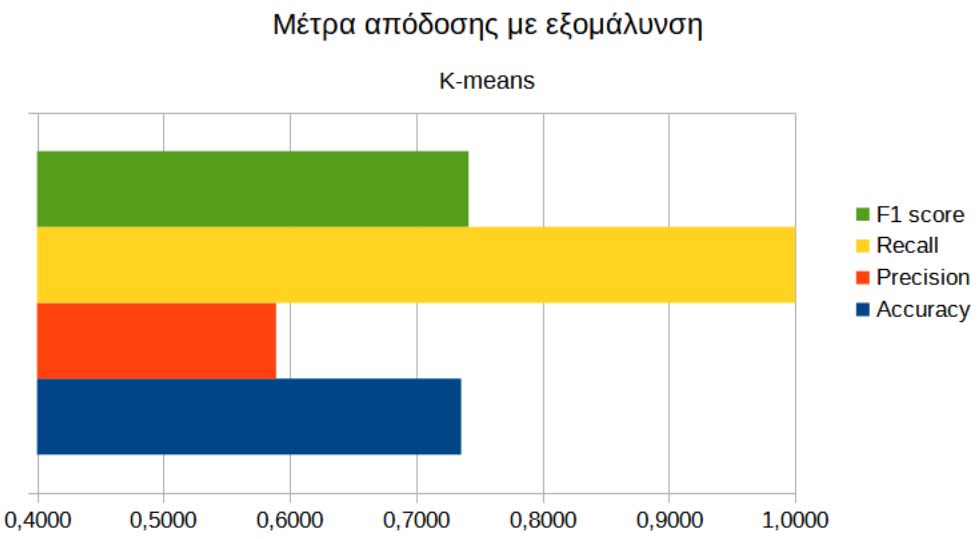


Εικόνα 57: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή K-means με εξομάλυνση

Παρατηρώντας τις τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών φαίνεται ότι ο αλγόριθμος K-means ομαδοποίησε με σχεδόν την ίδια επιτυχία τις κανονικές εικόνες και τις εικόνες με εξομάλυνση. Συγκεκριμένα, φαίνεται και πάλι ότι οι φυσιολογικές εικόνες ομαδοποιήθηκαν όλες μαζί, όπως και στην προηγούμενη περίπτωση. Αντίθετά, μεγάλο μέρος των μη φυσιολογικών εικόνων θεωρήθηκε ότι ανήκει στις φυσιολογικές, με αποτέλεσμα να μην αναγνωριστούν σχεδόν 1000 μη φυσιολογικές εικόνες. Το γεγονός αυτό αποτελεί αποτυχία του αλγορίθμου αφού ο σκοπός είναι η αναγνώριση ρωγμών και έχει ιδιαίτερη σημασία το πλήθος των ψευδώς θετικών στοιχείων.

Accuracy	0,7355
Precision	0,5991
Recall	1,000
F1 Score	0,7414

Πίνακας 12: Μέτρα απόδοσης K-means με εξομάλυνση



Εικόνα 58: Διαγραμματική παρουσίαση των μέτρων απόδοσης του K-means με εξομάλυνση

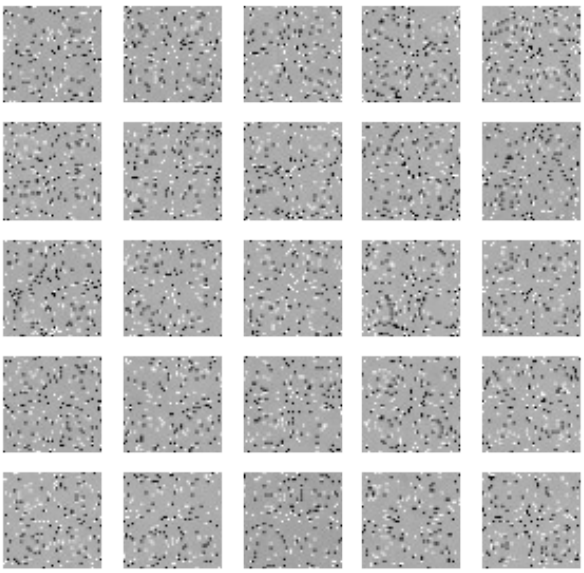
Τα παραπάνω φαίνονται και μέσω του υπολογισμού των accuracy, precision, recall και F1 score, των οποίων οι τιμές μεταβλήθηκαν ελάχιστα με την εφαρμογή εξομάλυνσης. Γενικά φαίνεται και πάλι ότι ο αλγόριθμος αναγνώρισε τις ομοιότητες μεταξύ των φυσιολογικών εικόνων, αναθέτοντας τες στην ίδια κλάση, αλλά αγνόησε την ύπαρξη μεγάλου μέρους ανώμαλων συμβάντων στις μη φυσιολογικές εικόνες. Αυτό είναι εμφανές μέσω της τιμής του precision, το οποίο λαμβάνει υπόψιν στον υπολογισμό του το πλήθος των ψευδών θετικών στοιχείων και προέκυψε κοντά στο 59%. Η ύπαρξη εξομάλυνσης δεν φάνηκε να επηρεάζει ιδιαίτερα τη διαδικασία, αφού υπήρξε ελάχιστη μεταβολή του πλήθους των αληθινά αρνητικών και ψευδώς θετικών.

Σημειώνεται ότι στα ίδια δεδομένα ο αυτόματος κωδικοποιητής εμφάνισε accuracy κοντά στο 91% και precision κοντά στο 87%, κάνοντας έτσι σαφές ότι επιτεύχθηκε καλύτερος διαχωρισμός φυσιολογικών και μη φυσιολογικών εικόνων.

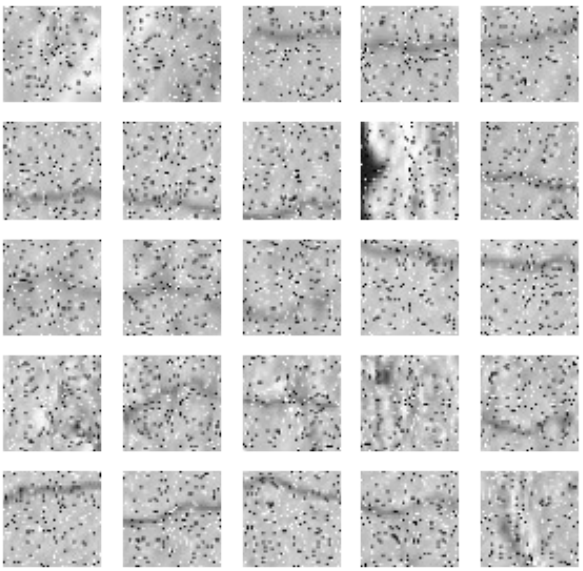
δ) Εφαρμογή K-means σε δεδομένα με θόρυβο

Για την εφαρμογή του αλγορίθμου K-means σε δεδομένα με θόρυβο, όπως είναι λογικό, χρησιμοποιήθηκαν τα δεδομένα ελέγχου στα οποία ελέγχθηκε ο αυτόματος κωδικοποιητής και έγινε εισαγωγή θορύβου τύπου αλάτι – πιπέρι. Η εισαγωγή του θορύβου επιτεύχθηκε μέσω της συνάρτησης “sp_noise”, η οποία παρουσιάστηκε σε προηγούμενη ενότητα.

Όμοια διαδικασία με τα προηγούμενα εφαρμόστηκε και για την εφαρμογή του αλγορίθμου, αφού χρησιμοποιήθηκε η συνάρτηση “KMeans” στην οποία ζητήθηκε να ομαδοποιήσει τα δεδομένα με θόρυβο. Μετά την ομαδοποίηση φαίνεται ότι η κλάση 1, που προέκυψε με την εφαρμογή του αλγορίθμου, αντιστοιχεί στις φυσιολογικές εικόνες, ενώ η κλάση 0 στις μη φυσιολογικές εικόνες. Παρακάτω φαίνονται ενδεικτικά μερικές εικόνες που ανήκουν στην κάθε κλάση.



Εικόνα 59: Κλάση 1 (φυσιολογικές εικόνες) K-means με θόρυβο



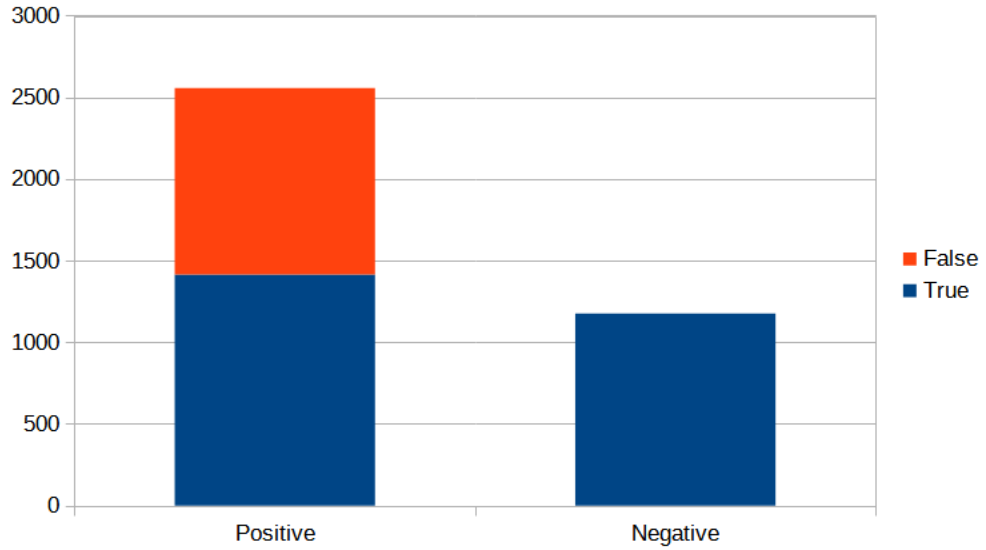
Εικόνα 60: Κλάση 0 (μη φυσιολογικές εικόνες) K-means με θόρυβο

Παρατηρώντας τις εικόνες αυτές, φαίνεται ότι για τα συγκεκριμένα υποσύνολα των κλάσεων έχει γίνει επιτυχημένη ομαδοποίηση με την κλάση 1 να περιλαμβάνει φυσιολογικές εικόνες με θόρυβο και την κλάση 0 να περιλαμβάνει μη φυσιολογικές εικόνες με θόρυβο.

Στον παρακάτω πίνακα και στο διάγραμμα που ακολουθεί φαίνονται οι τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών, από τις οποίες μπορούν να εξαχθούν αντικειμενικά συμπεράσματα για την ομαδοποίηση.

True Positive (TP)	1415
False Positive (FP)	1141
True Negative (TN)	1176
False Negative (FN)	0

Πίνακας 13: Πίνακας TP, TN, FP, FN για εφαρμογή K-means σε δεδομένα με θόρυβο



Εικόνα 61: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή K-means με θόρυβο

Ακόμα και με την ύπαρξη θορύβου, ο αλγόριθμος φαίνεται να μπορεί με μεγάλη επιτυχία να αναθέσει στην ίδια κλάση όλες τις φυσιολογικές εικόνες αναγνωρίζοντας τις ομοιότητές τους. Αντίθετα, στις μη φυσιολογικές εικόνες η απόδοσή του ήταν εντελώς διαφορετική. Στις προηγούμενες περιπτώσεις, δηλαδή στην περίπτωση των κανονικών δεδομένων και στην περίπτωση των δεδομένων με εξομάλυνση, ο αλγόριθμος αγνόησε μεγάλο μέρος των μη φυσιολογικών εικόνων θεωρώντας ότι ανήκουν στην ίδια κλάση με τις φυσιολογικές. Με την εισαγωγή θορύβου, το πρόβλημα φαίνεται να επιδεινώθηκε, αφού κατάφερε να διαχωρίσει λιγότερες από τις μισές μη φυσιολογικές εικόνες και οι υπόλοιπες ανατέθηκαν στην κλάση των φυσιολογικών.

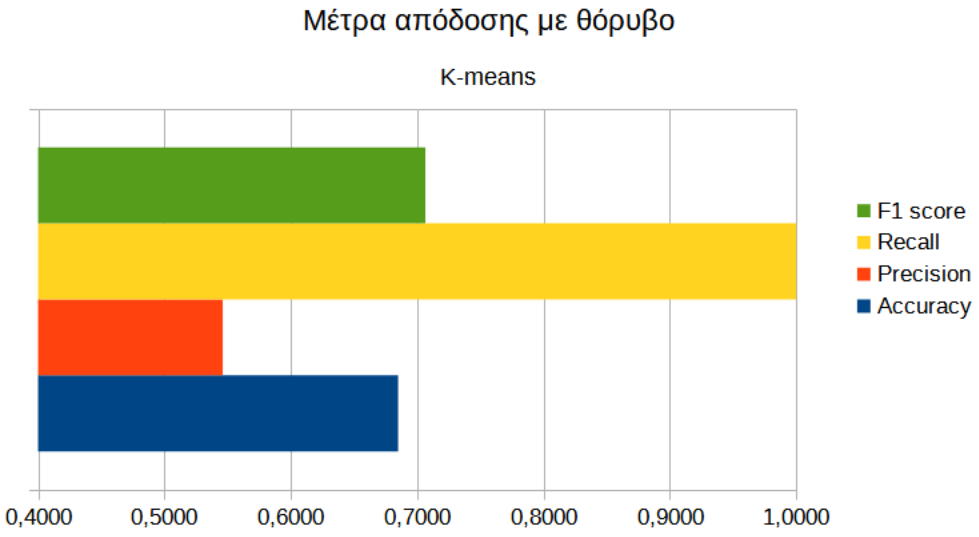
Φυσικά, το φαινόμενο αυτό απεικονίζεται και στα μέτρα απόδοσης, σημειώνοντας τα παρακάτω αποτελέσματα.

Accuracy	0,6849
Precision	0,5461
Recall	1,000
F1 Score	0,7064

Πίνακας 14: Μέτρα απόδοσης K-means με θόρυβο

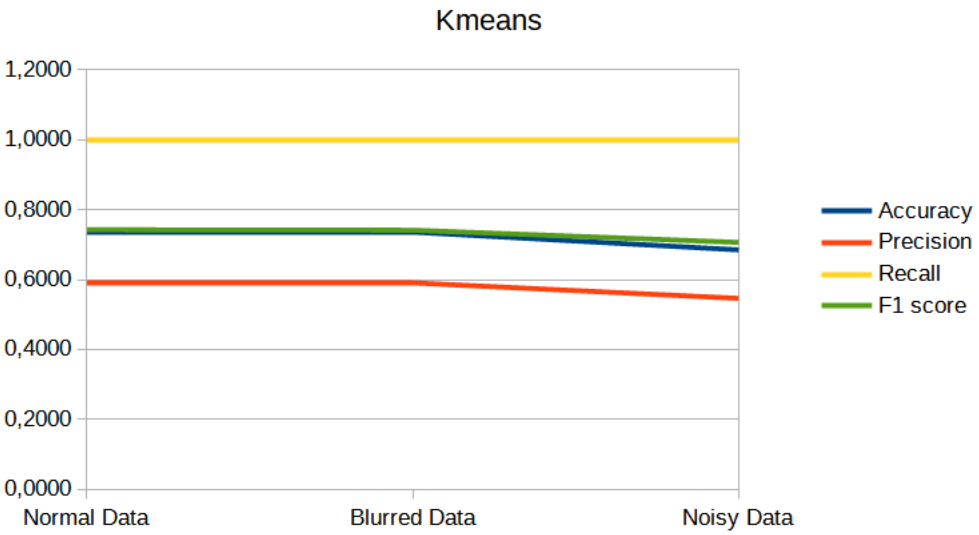
Παρατηρώντας τις τιμές του παραπάνω πίνακα αλλά και τη διαγραμματική τους αναπαράσταση στο παρακάτω διάγραμμα, φαίνεται οι τιμές των accuracy, precision και F1 score να παρουσιάζουν πτώση σε σχέση με τις προηγούμενες περιπτώσεις. Η τιμή του recall παραμένει σταθερή στο 1 αφού εξαρτάται μόνο από την επιτυχημένη ομαδοποίηση των θετικών, δηλαδή των φυσιολογικών εικόνων. Το precision από την άλλη, αν και κυμαινόταν σε χαμηλά επίπεδα και στις προηγούμενες περιπτώσεις,

πέφτει ακόμα περισσότερο λόγω της αποτυχίας του αλγορίθμου να αναθέσει στην κατάλληλη κλάση τις μισές μη φυσιολογικές εικόνες. Λαμβάνοντας τα παραπάνω υπόψιν και εφόσον βασική προτεραιότητα της παρούσας εργασίας είναι ο εντοπισμός των εικόνων που περιλαμβάνουν ρωγμές, ο αλγόριθμος K-means δεν αποτελεί τον βέλτιστο αλγόριθμο για τη συγκεκριμένη εφαρμογή, ιδιαίτερα στην περίπτωση ύπαρξης θορύβου.



Εικόνα 62: Διαγραμματική παρουσίαση των μέτρων απόδοσης του K-means με θόρυβο

ε) Σύγκριση αποτελεσμάτων για τον αλγόριθμο K-means



Εικόνα 63: Συγκεντρωτικό διάγραμμα μέτρων απόδοσης για τον αλγόριθμο K-means

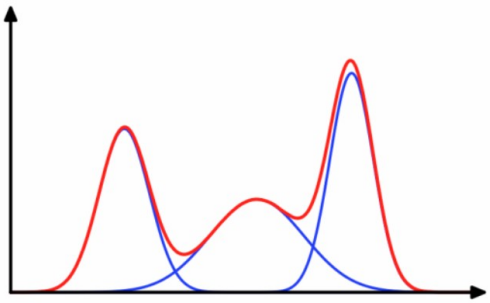
Στο παραπάνω διάγραμμα στον οριζόντιο άξονα φαίνεται το είδος των δεδομένων (κανονικά, με εξομάλυνση ή με θόρυβο) και στον κατακόρυφο άξονα φαίνονται οι τιμές των accuracy, precision, recall και F1 score. Παρατηρώντας τις τεθλασμένες γραμμές φαίνεται ότι η απόδοση του αλγορίθμου παρουσίασε σχετική σταθερότητα. Πρώτα απ' όλα, η τιμή του recall παραμένει σταθερή στη μέγιστη δυνατή τιμή, δηλαδή στο 1. Επιπλέον, το accuracy και το F1 score διαγράφουν σχεδόν παράλληλη πορεία με πτωτική τάση και λαμβάνουν τις ελάχιστες τιμές τους στην περίπτωση των εικόνων με θόρυβο, χωρίς όμως η πτώση αυτή να είναι μεγάλη. Το σημαντικότερο πρόβλημα του αλγορίθμου όμως είναι εμφανές από τις τιμές του precision, το οποίο επίσης παραμένει σχεδόν σταθερό σε χαμηλές τιμές με μία μικρή πτώση στην περίπτωση του θορύβου. Οι τιμές αυτές οφείλονται στο μεγάλο πλήθος ψευδώς θετικών στοιχείων, το οποίο κάνει τον αλγόριθμο ακατάλληλο για εντοπισμό ρωγμών με βάση το συγκεκριμένο σύνολο δεδομένων.

5.4 Mixture of Gaussians (GMM)

α) Γενικά

Το Gaussian Mixture Model (GMM) είναι ένα πιθανολογικό μοντέλο που θεωρεί ότι όλα τα σημεία δεδομένων παράγονται από ένα μείγμα πεπερασμένου αριθμού κατανομών Gauss με άγνωστες παραμέτρους. Το GMM αποτελεί γενίκευση του K-means και ενσωματώνει πληροφορίες σχετικά με τη συνδιακύμανση των δεδομένων [49].

Μορφές του GMM χρησιμοποιούνται συχνά για την αναγνώριση ομιλίας λόγω της ικανότητάς τους να αναπαριστούν μεγάλη ποικιλία κατανομών. Ένα από τα σημαντικότερα χαρακτηριστικά του GMM είναι η ικανότητά του να σχηματίζει ομαλές προσεγγίσεις για αυθαίρετα διαμορφωμένες πυκνότητες. Όπως αναφέρθηκε και παραπάνω, χρησιμοποιεί ένα σύνολο γκαουσιανών συναρτήσεων, με διακριτές μέσες τιμές και πίνακες συνδιακύμανσης, επιτρέποντας έτσι καλύτερη μοντελοποίηση των δεδομένων. [18]



Εικόνα 64: Παράδειγμα GMM όπου με μπλε φαίνονται οι επιμέρους καμπύλες Gauss και με κόκκινο ο συνδυασμός τους

Πηγή: *https://dirichletprocess.weebly.com/clustering.html*

Σε συμφωνία με τα προηγούμενα, τα δεδομένα ελέγχου που χρησιμοποιήθηκαν στον αυτόματο κωδικοποιητή (στην κανονική τους μορφή, με εξομάλυνση και με θόρυβο), θα ζητηθεί να διαχωριστούν σε δύο κλάσεις από το GMM και στη συνέχεια θα συγκριθούν τα αποτελέσματα.

β) Εφαρμογή GMM στα κανονικά δεδομένα

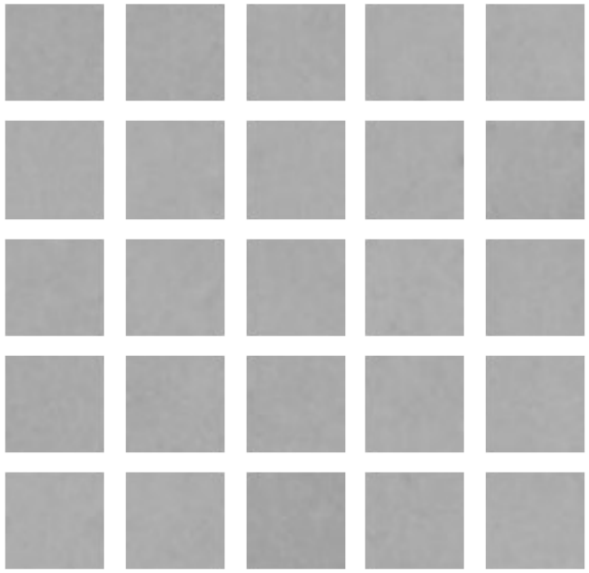
Η εφαρμογή του GMM παρουσιάζει πολλές ομοιότητες με την εφαρμογή του K-means. Πρώτα απ' όλα, αφού δεν απαιτούνται δεδομένα εκπαίδευσης, το μοντέλο εφαρμόζεται μόνο στα δεδομένα ελέγχου. Τα δεδομένα διαβάστηκαν από τους αντίστοιχους φακέλους και δεν πραγματοποιήθηκε κανονικοποίηση. Παράλληλα, δημιουργήθηκε μία λίστα με τις πραγματικές ετικέτες τους για χρήση στην αξιολόγηση της απόδοσης του μοντέλου.

Τα δεδομένα οργανώθηκαν στη λίστα "data" η οποία μετατράπηκε σε numpy array με διαστάσεις (3732, 45, 45). Στη συνέχεια, το array αναδιαμορφώθηκε ώστε οι διαστάσεις του να είναι (3732, 2025), όπως στην περίπτωση του αλγορίθμου K-means. Για την εφαρμογή του GMM

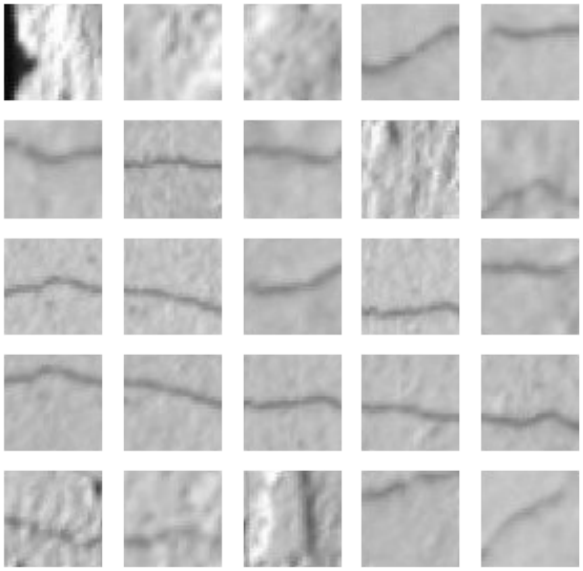
χρησιμοποιήθηκε η συνάρτηση “GaussianMixture” που ανήκει στη βιβλιοθήκη “Scikit-learn” με την παράμετρο “n_components” να ισούται με 2, αφού το επιθυμητό πλήθος κλάσεων είναι επίσης 2.

```
gmm = GaussianMixture(n_components=2)
gmm.fit(data)
Z = gmm.predict(data)
```

Με αυτό τον τρόπο δημιουργείται η λίστα “Z” η οποία περιλαμβάνει την ετικέτα της κλάσης που ανατέθηκε στην κάθε εικόνα. Οι ετικέτες 0 και 1 των κλάσεων που προκύπτουν μετά την εφαρμογή του GMM δεν ταυτίζονται απαραίτητα με τις αληθείς ετικέτες. Δηλαδή, η κλάση 0 που προκύπτει από το GMM δεν είναι απαραίτητα η κλάση που περιλαμβάνει τις φυσιολογικές εικόνες, όπως θα περίμενε κανείς σύμφωνα με τις αληθείς ετικέτες. Παρατηρώντας τα αποτελέσματα φαίνεται ότι αυτή τη φορά η κλάση 0 αντιστοιχεί στην πραγματικότητα στην κλάση των φυσιολογικών εικόνων και η κλάση 1 στην κλάση των μη φυσιολογικών εικόνων. Στις παρακάτω εικόνες φαίνονται ενδεικτικά μερικές από τις εικόνες της κάθε κλάσης και παρατηρείται ότι σε αυτό το μικρό υποσύνολο των δεδομένων ελέγχου φαίνεται να έχουν διαχωριστεί σωστά οι φυσιολογικές και μη φυσιολογικές εικόνες.



Εικόνα 65: Κλάση 0 (φυσιολογικές εικόνες) GMM με κανονικά δεδομένα



Εικόνα 66: Κλάση 1 (μη φυσιολογικές εικόνες) GMM με κανονικά δεδομένα

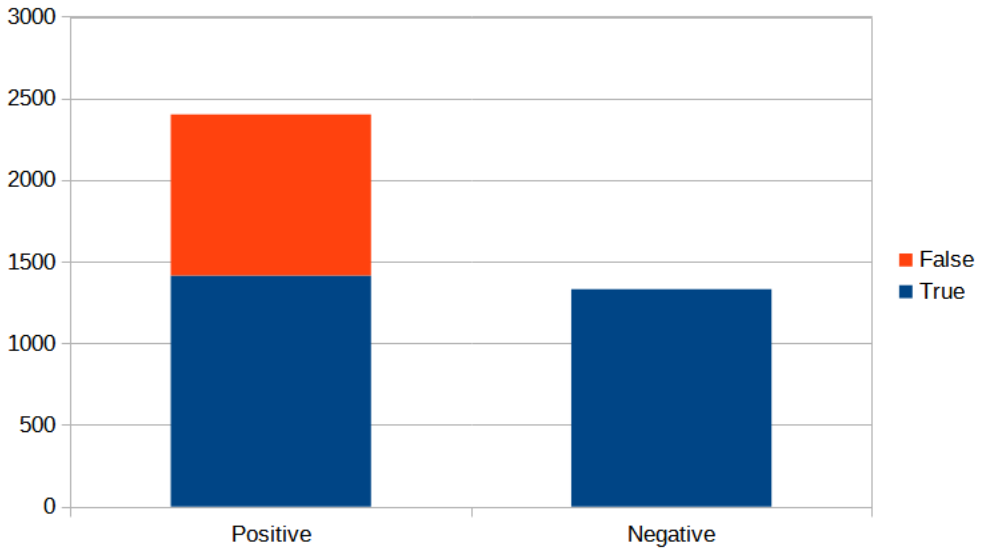
Αφού διαχωρίστηκαν οι εικόνες, χρησιμοποιείται η λίστα “Z” και οι αληθείς ετικέτες για να υπολογιστεί το πλήθος των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών στοιχείων, τα οποία ορίζονται όπως και στις προηγούμενες ενότητες και υπολογίζονται συγκρίνοντας τις αληθείς ετικέτες με τις προβλεπόμενες.

```
TP = 0
TN = 0
```

```
FP = 0
FN = 0
for i in range(len(y_test)):
  if y_test[i] == 0 and Z[i] == 0:
    TP = TP + 1
  elif y_test[i] == 1 and Z[i] == 1:
    TN = TN + 1
  elif y_test[i] == 0 and Z[i] == 1:
    FN = FN + 1
  else:
    FP = FP + 1
```

True Positive (TP)	1415
False Positive (FP)	1331
True Negative (TN)	986
False Negative (FN)	0

Πίνακας 15: Πίνακας TP, TN, FP, FN για εφαρμογή GMM σε κανονικά δεδομένα



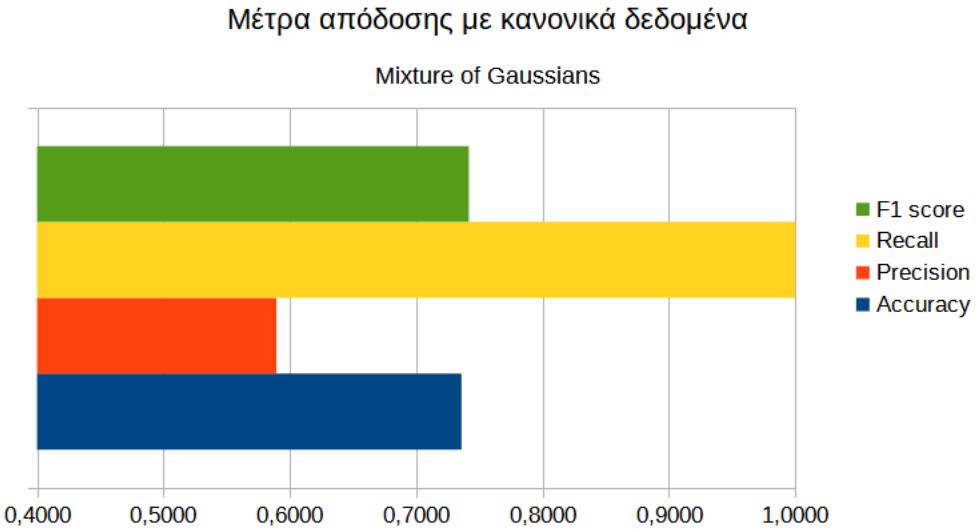
Εικόνα 67: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή GMM με κανονικά δεδομένα

Αν και λόγω της μεγαλύτερης ευελιξίας του GMM θα ήταν αναμενόμενο να προκύψουν καλύτερα αποτελέσματα συγκριτικά με τον αλγόριθμο K-means, στην πραγματικότητα οι τιμές των TP, TN, FP και FN δείχνουν ότι οι διαφορές των φυσιολογικών και μη φυσιολογικών εικόνων δεν έγιναν απόλυτα αντιληπτές από το μοντέλο. Συγκεκριμένα φαίνεται και πάλι ότι οι φυσιολογικές εικόνες στο

σύνολο τους έχουν αναγνωριστεί ως όμοιες και έχουν ανατεθεί στην ίδια κλάση. Από την άλλη πλευρά, στην κλάση που περιλαμβάνει τις φυσιολογικές εικόνες φαίνεται να κατέληξε και ένα μεγάλο μέρος των μη φυσιολογικών, πράγμα που αποτελεί πρόβλημα. Τα συμπεράσματα αυτά απεικονίζονται και στις τιμές των accuracy, precision, recall και F1 score που φαίνονται στον παρακάτω πίνακα.

Accuracy	0,7358
Precision	0,5893
Recall	1,000
F1 Score	0,7416

Πίνακας 16: Μέτρα απόδοσης GMM με κανονικά δεδομένα



Εικόνα 68: Διαγραμματική παρουσίαση των μέτρων απόδοσης του GMM με κανονικά δεδομένα

Με μία πρώτη ματιά αποκλειστικά στις τιμές του accuracy και του F1 score υποθέτεται ότι το GMM παρουσιάζει μέτρια απόδοση στον διαχωρισμό των φυσιολογικών δεδομένων, ο οποίος έχει επιτευχθεί με ποσοστό επιτυχίας άνω του 70%. Η τιμή του recall, η οποία στα πρότυπα του K-means είναι και πάλι 1, δείχνει ότι το μοντέλο κατάφερε εύκολα να εντοπίσει τις ομοιότητες μεταξύ των φυσιολογικών εικόνων. Αντίθετα, η τιμή του precision δείχνει εντελώς διαφορετική εικόνα. Συγκεκριμένα φαίνεται ότι το μοντέλο αγνόησε την ύπαρξη ρωγμών ή άλλων ανωμαλιών σε 4 εικόνες σε κάθε δεκάδα, πράγμα που δείχνει ότι το μοντέλο αδυνατεί να εντοπίσει ρωγμές εξίσου καλά με άλλες προσεγγίσεις όπως ο αυτόματος κωδικοποιητής.

γ) Εφαρμογή GMM σε δεδομένα με εξομάλυνση

Όπως και στην περίπτωση του αυτόματου κωδικοποιητή και του K-means, έτσι και στην περίπτωση του GMM, χρειάζεται να δοκιμαστούν και δεδομένα που έχουν υποστεί εξομάλυνση.

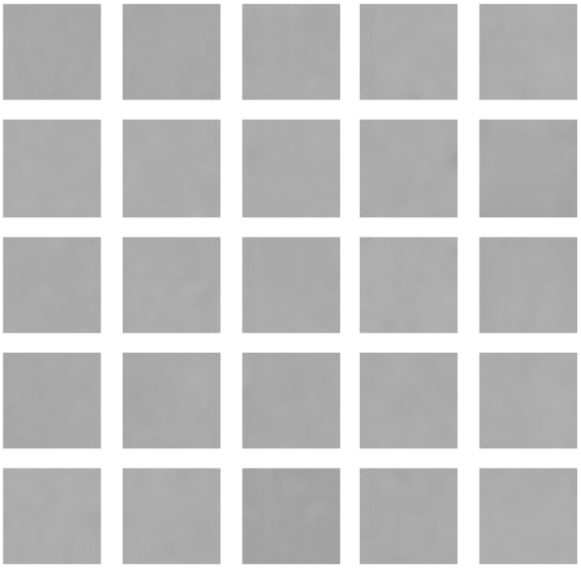
Φυσικά, η εξομάλυνση των δεδομένων ελέγχου πραγματοποιήθηκε όπως και στις προηγούμενες ενότητες, δηλαδή με χρήση φίλτρου Gauss.

Αρχικά, διαβάστηκαν οι εικόνες από τους αντίστοιχους φακέλους και οργανώθηκαν στη λίστα "b_data" η οποία μετατράπηκε σε numpy array με διαστάσεις (3732, 45, 45). Έπειτα, αναδιαμορφώθηκε και οι νέες διαστάσεις του array είναι (3732, 2025).

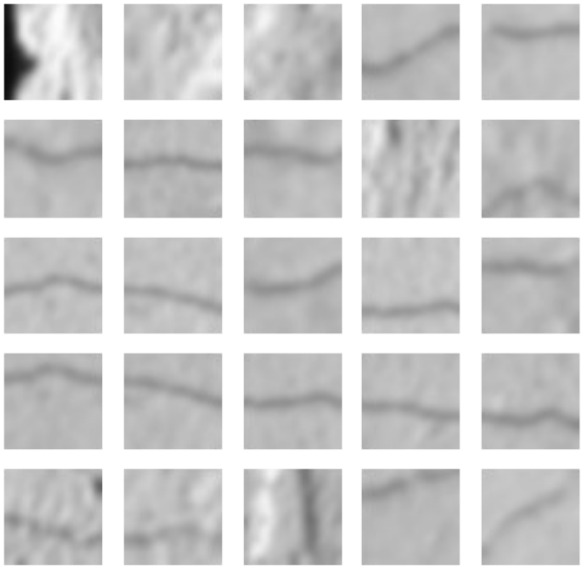
Αφού προετοιμάστηκαν τα δεδομένα, έγινε εισαγωγή τους στη συνάρτηση "GaussianMixture" και παράχθηκαν οι προβλεπόμενες ετικέτες στην λίστα "b_Z".

```
gmm = GaussianMixture(n_components=2)
gmm.fit(b_data)
b_Z = gmm.predict(b_data)
```

Παρατηρήθηκε ότι στη συγκεκριμένη περίπτωση η προβλεπόμενη ετικέτα 1 αντιστοιχεί στις φυσιολογικές εικόνες και η ετικέτα 0 στις μη φυσιολογικές, σε αντίθεση με τις αληθείς ετικέτες. Παρακάτω φαίνονται ενδεικτικά μερικές εικόνες από κάθε κλάση.



Εικόνα 69: Κλάση 1 (φυσιολογικές εικόνες) GMM με εξομάλυνση

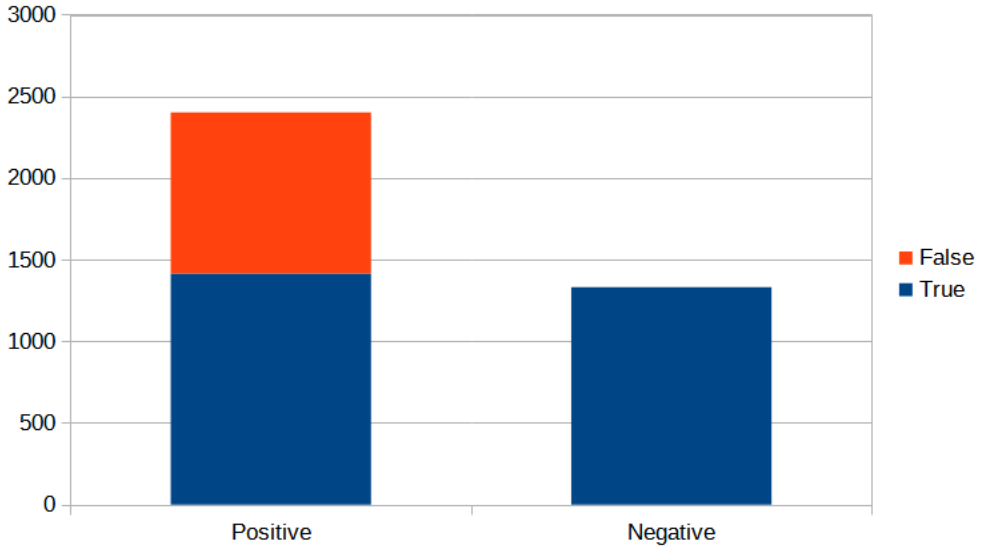


Εικόνα 70: Κλάση 0 (μη φυσιολογικές εικόνες) GMM με εξομάλυνση

Αφού ολοκληρώθηκε η διαδικασία ομαδοποίησης, η επίδοση του μοντέλου αξιολογήθηκε μέσω του πλήθους των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών στοιχείων και στη συνέχεια μέσω των μέτρων απόδοσης accuracy, precision, recall και F1 score.

True Positive (TP)	1415
False Positive (FP)	1330
True Negative (TN)	987
False Negative (FN)	0

Πίνακας 17: Πίνακας TP, TN, FP, FN για εφαρμογή GMM με εξομάλυνση



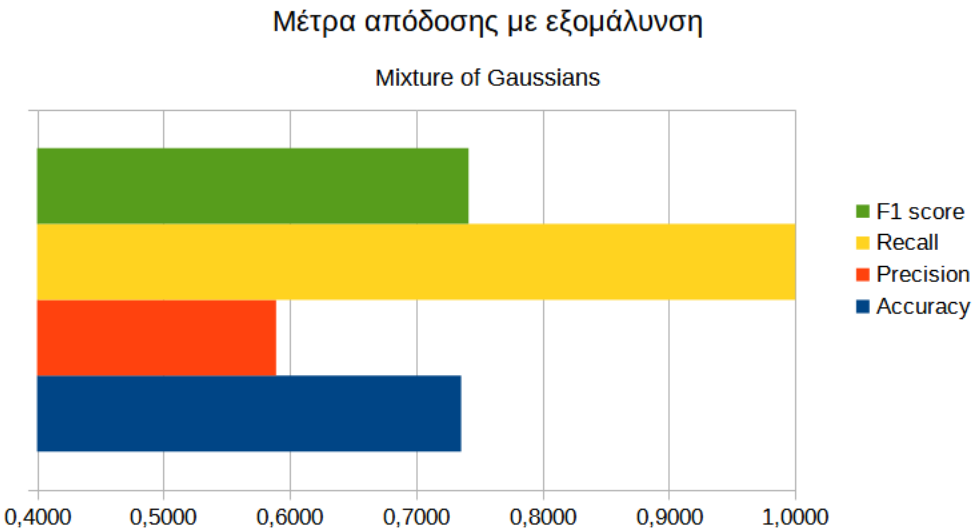
Εικόνα 71: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή GMM με εξομάλυνση

Όπως φαίνεται οι τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών με εξομάλυνση σχεδόν ταυτίζονται με τις αντίστοιχες τιμές με κανονικά δεδομένα. Με αυτό τον τρόπο φαίνεται ότι η εξομάλυνση που εφαρμόστηκε δεν επηρέασε την απόδοση του αλγορίθμου. Από την άλλη πλευρά, αν και ομαδοποιήθηκε σωστά το σύνολο των φυσιολογικών εικόνων, όπως και στην προηγούμενη περίπτωση, σχεδόν 1000 μη φυσιολογικές εικόνες θεωρήθηκαν φυσιολογικές. Το γεγονός αυτό υποδεικνύει ότι ο αλγόριθμος αυτός δεν είναι κατάλληλος για εντοπισμό ρωγμών με τα συγκεκριμένα δεδομένα και θα πρέπει να χρησιμοποιηθεί κάποια άλλη προσέγγιση.

Τα παραπάνω απεικονίζονται και στις τιμές των accuracy, precision, recall και F1 score. Συγκεκριμένα, οι τιμές τους, οι οποίες παρουσιάζονται παρακάτω, βρίσκονται πολύ κοντά στις αντίστοιχες τιμές με χρήση των κανονικών δεδομένων και εξάγονται οι ίδιες παρατηρήσεις.

Accuracy	0,7355
Precision	0,5891
Recall	1,000
F1 Score	0,7414

Πίνακας 18: Μέτρα απόδοσης GMM με εξομάλυνση

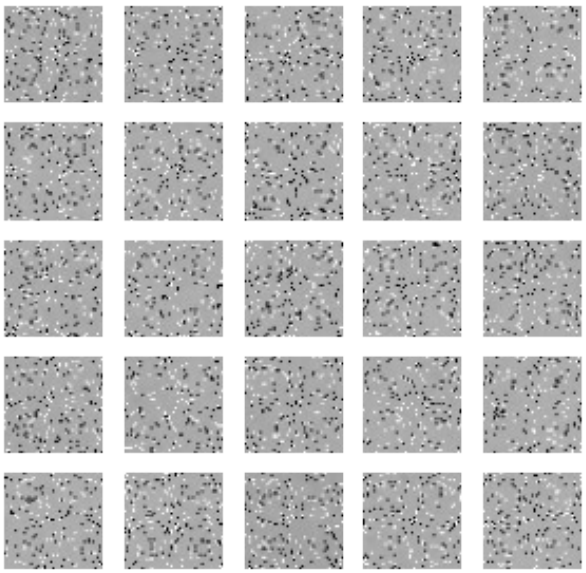


Εικόνα 72: Διαγραμματική παρουσίαση των μέτρων απόδοσης του GMM με εξομάλυνση

δ) Εφαρμογή GMM σε δεδομένα με θόρυβο

Ολοκληρώνοντας τη δοκιμή του μοντέλου GMM, θα πρέπει να ελεγχθεί η απόδοσή του σε δεδομένα με θόρυβο. Ο θόρυβος, όπως και στις προηγούμενες ενότητες είναι της μορφής αλάτι – πιπέρι και εισάγεται στα δεδομένα ελέγχου, αφού διαβαστούν από τους αντίστοιχους φακέλους, μέσω της συνάρτησης "sp_noise", η οποία αναλύθηκε σε προηγούμενη ενότητα.

Τα δεδομένα, δηλαδή οι φυσιολογικές και μη φυσιολογικές εικόνες, οργανώνονται στη λίστα "n_data", η οποία μετατρέπεται σε numpy array και αναδιαμορφώνεται ώστε οι διαστάσεις της να είναι (3732, 2025). Το μοντέλο δημιουργείται και διαχωρίζονται τα δεδομένα μέσω της συνάρτησης "GaussianMixture" όπως φαίνεται παρακάτω.



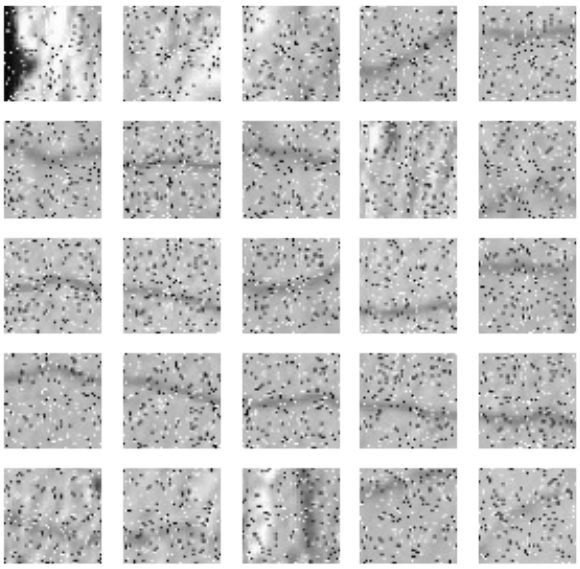
Εικόνα 73: Κλάση 1 (φυσιολογικές εικόνες) GMM με θόρυβο

```
gmm = GaussianMixture(n_components=2)
gmm.fit(n_data)
n_Z = gmm.predict(n_data)
```

Η λίστα "n_Z" περιλαμβάνει τις προβλεπόμενες ετικέτες για τα δεδομένα. Αυτή τη φορά, η προβλεπόμενη ετικέτα 1 αντιστοιχεί στα φυσιολογικά δεδομένα και η προβλεπόμενη ετικέτα 0 στα μη φυσιολογικά δεδομένα. Στις Εικόνες 73 και 74 φαίνεται ένα υποσύνολο των εικόνων που

ταξινομήθηκαν σε κάθε κλάση. Σύμφωνα με το μικρό αυτό υποσύνολο, ο διαχωρισμός φαίνεται να έχει γίνει επιτυχημένα.

Αφού ολοκληρώθηκε ο διαχωρισμός των εικόνων υπολογίστηκε το πλήθος των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών, αναμένοντας οι τιμές που αντιστοιχούν στα ψευδώς θετικά και ψευδώς αρνητικά να είναι υψηλότερες σε σχέση με τις προηγούμενες περιπτώσεις λόγω της επίδρασης του θορύβου.



Εικόνα 74: Κλάση 0 (μη φυσιολογικές εικόνες) GMM με θόρυβο

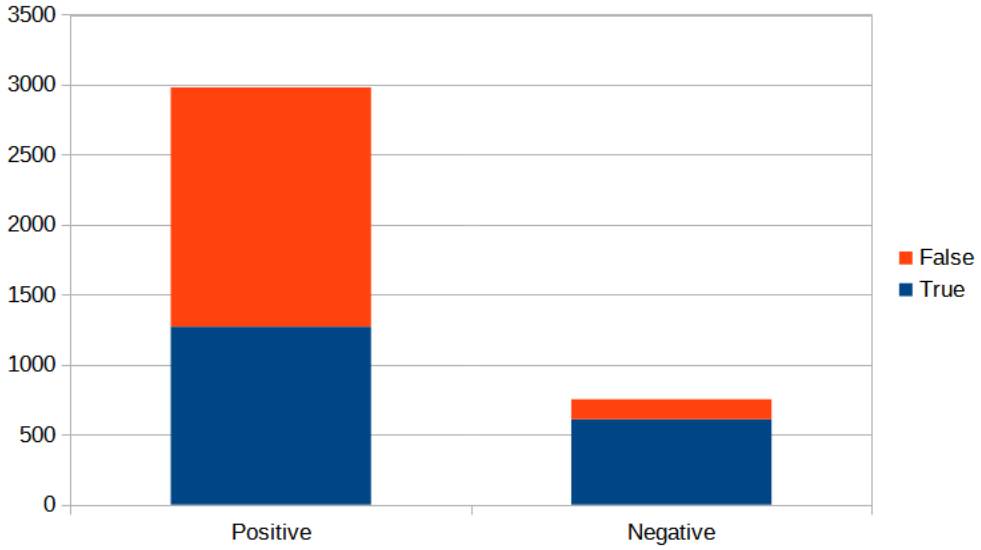
Οι τιμές των TP, TN, FP και FN που υπολογίστηκαν εμφανίζονται στον παρακάτω πίνακα και στο παρακάτω διάγραμμα. Είναι εμφανές ότι ο θόρυβος, σε αντίθεση με την εξομάλυνση, επηρέασε πάρα πολύ την απόδοση του μοντέλου. Συγκεκριμένα, σε αντίθεση με τις προηγούμενες εφαρμογές του GMM στις προηγούμενες ενότητες, ο θόρυβος προκάλεσε αδυναμίες και στον εντοπισμό των φυσιολογικών εικόνων, αφού παρατηρήθηκαν 143 ψευδώς αρνητικά στοιχεία. Το πλήθος των ψευδώς αρνητικών όμως είναι αμελητέο συγκρινόμενο με το πλήθος των ψευδώς θετικών. Όπως φαίνεται, το μοντέλο απέτυχε πλήρως στον διαχωρισμό των μη φυσιολογικών εικόνων από τις φυσιολογικές, με το πλήθος των ψευδώς θετικών να ξεπερνά το πλήθος των αληθινά αρνητικών.

True Positive (TP)	1272
False Positive (FP)	1707
True Negative (TN)	610
False Negative (FN)	143

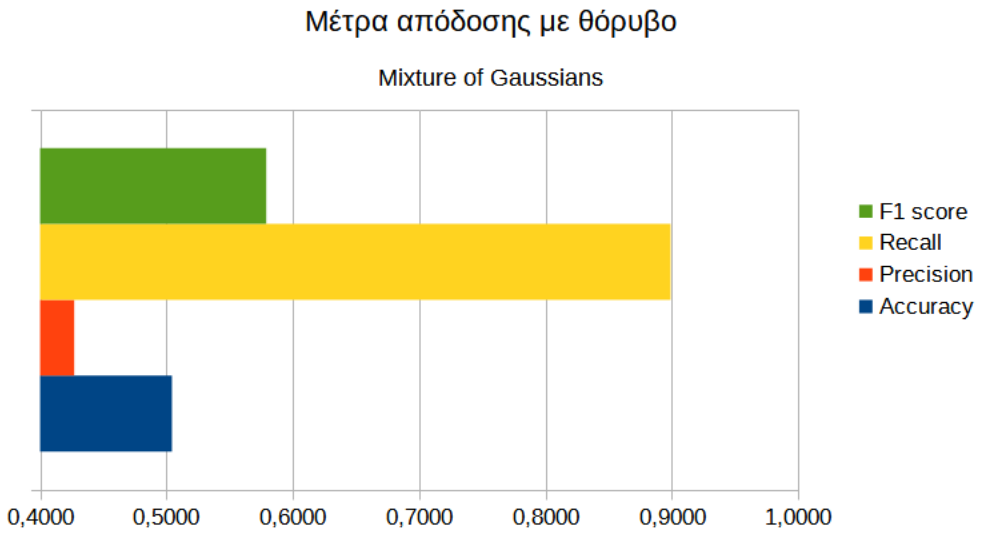
Πίνακας 19: Πίνακας TP, TN, FP, FN για εφαρμογή GMM με θόρυβο

Accuracy	0,5043
Precision	0,4270
Recall	0,8989
F1 Score	0,5790

Πίνακας 20: Μέτρα απόδοσης GMM με θόρυβο



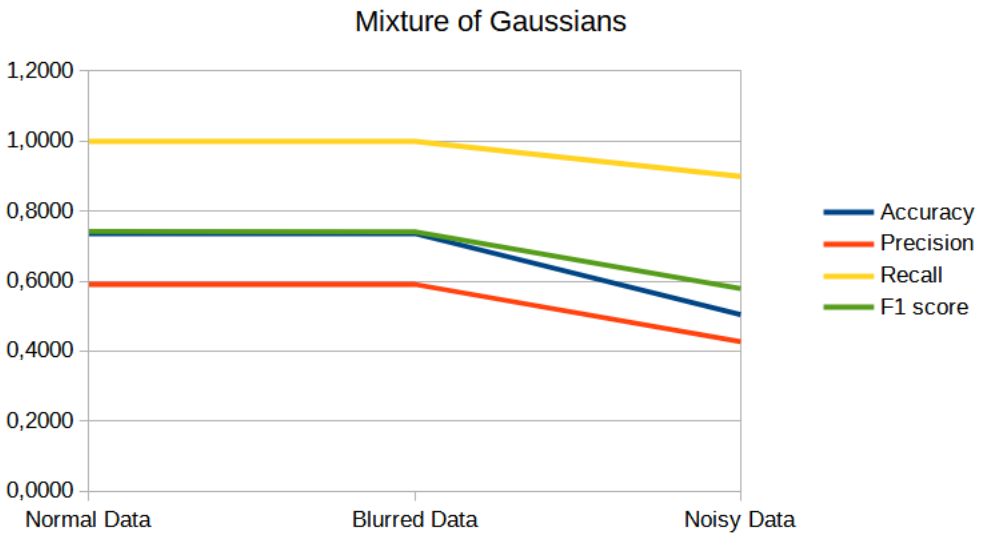
Εικόνα 75: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή GMM με θόρυβο



Εικόνα 76: Διαγραμματική παρουσίαση των μέτρων απόδοσης του GMM με θόρυβο

Οι τιμές των μέτρων απόδοσης στην περίπτωση του θορύβου φαίνεται να κυμαίνονται κοντά στο 50%, με το precision να σημειώνει ακόμα χαμηλότερη τιμή (43%) και το recall να είναι το μοναδικό που παίρνει τιμή κοντά στο 90%, αφού αφορά τα φυσιολογικά δεδομένα. Από τα παραπάνω φαίνεται ότι το μοντέλο δεν πέτυχε τον διαχωρισμό των φυσιολογικών και μη φυσιολογικών εικόνων, ιδιαίτερα στην περίπτωση ύπαρξης θορύβου.

ε) Σύγκριση αποτελεσμάτων για το GMM



Εικόνα 77: Συγκεντρωτικό διάγραμμα μέτρων απόδοσης για το GMM

Στον οριζόντιο άξονα του παραπάνω διαγράμματος φαίνεται το είδος των δεδομένων και στον κατακόρυφο οι τιμές των μέτρων απόδοσης. Με αυτό τον τρόπο είναι εύκολο να συγκριθεί η απόδοση του GMM για κάθε είδος δεδομένων.

Το πρώτο μισό του διαγράμματος αφορά την μεταβολή από κανονικά δεδομένα σε δεδομένα με εξομάλυνση. Η προσθήκη φίλτρου Gauss δεν φάνηκε να επηρεάζει το μοντέλο, το οποίο κατέληξε σχεδόν στα ίδια αποτελέσματα. Έτσι, όπως είναι λογικό, τα μέτρα απόδοσης δεν μεταβλήθηκαν με την προσθήκη του φίλτρου. Συγκεκριμένα, η παράμετρος recall παρέμεινε σταθερή στην μέγιστη δυνατή τιμή της, αφού το μοντέλο πέτυχε να αναθέσει στην ίδια κλάση όλες τις φυσιολογικές εικόνες. Από την άλλη πλευρά, η παράμετρος precision σταθεροποιήθηκε σε χαμηλά επίπεδα (δηλαδή κάτω από το 60%) λόγω του μεγάλου πλήθους μη φυσιολογικών εικόνων που ομαδοποιήθηκαν στην κλάση των φυσιολογικών. Το γεγονός αυτό δείχνει ότι το μοντέλο δεν κατάφερε να αναγνωρίσει τη διαφορετικότητα των μη φυσιολογικών εικόνων, πράγμα που μπορεί να αποτελέσει πρόβλημα σε εφαρμογή ανίχνευσης ρωγμών. Παράλληλα, οι παράμετροι accuracy και F1 score που λαμβάνουν υπόψιν την απόδοση του μοντέλου στις φυσιολογικές και στις μη φυσιολογικές εικόνες κατέληξαν κοντά στο 70%.

Στο δεύτερο μισό του διαγράμματος φαίνεται η μεταβολή των τιμών των μέτρων απόδοσης από τα δεδομένα με εξομάλυνση στα δεδομένα με θόρυβο. Ο θόρυβος φαίνεται να έπαιξε καθοριστικό ρόλο και να αποπροσανατόλισε περαιτέρω το μοντέλο. Συγκεκριμένα αυτή τη φορά το recall έπεσε κοντά στο 90% λόγω της ύπαρξης κάποιων ψευδώς αρνητικών στοιχείων, διατηρώντας όμως υψηλή τιμή. Ομοίως και το precision σημείωσε πτωτική τάση, λαμβάνοντας τιμές κοντά στο 40%, λόγω του μεγάλου πλήθους ψευδώς θετικών στοιχείων (τα οποία ξεπέρασαν τα αληθινά αρνητικά). Το γεγονός αυτό δείχνει πλήρη αποτυχία του μοντέλου στον ορθό διαχωρισμό των δεδομένων στην περίπτωση του θορύβου. Ταυτόχρονα, οι παράμετροι accuracy και F1 score έλαβαν τιμές κοντά στο 50%, δείχνοντας ότι το μοντέλο κατάφερε να αναγνωρίσει επιτυχημένα μόνο τις μισές από τις εικόνες που αντιμετώπισε.

5.5 Επιβλεπόμενο νευρωνικό δίκτυο

α) Γενικά

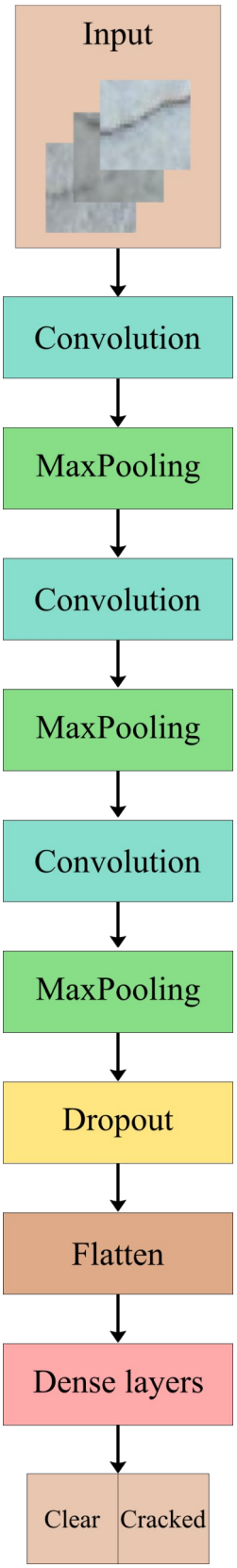
Οι αλγόριθμοι που παρουσιάστηκαν μέχρι το σημείο αυτό στα πλαίσια της παρούσας εργασίας αφορούν μη επιβλεπόμενη μάθηση. Στην ενότητα αυτή θα ελεγχθεί η απόδοση ενός επιβλεπόμενου νευρωνικού δικτύου στην ταξινόμηση των κανονικών δεδομένων, των δεδομένων με εξομάλυνση και των δεδομένων με θόρυβο.

Αφού το νευρωνικό δίκτυο περιλαμβάνει επίπεδα που πραγματοποιούν συνέλιξη στη συνέχεια θα αναφέρεται ως CNN (Convolutional Neural Network – Συνελικτικό Νευρωνικό Δίκτυο) για λόγους συντομίας. Σημειώνεται βέβαια ότι και ο αυτόματος κωδικοποιητής που παρουσιάστηκε στο προηγούμενο κεφάλαιο αποτελεί ένα συνελικτικό νευρωνικό δίκτυο.

Σε αντίθεση με τον αυτόματο κωδικοποιητή ο οποίος χρειάζεται μόνο φυσιολογικά δεδομένα για να εκπαιδευτεί, στο CNN τα δεδομένα εκπαίδευσης θα πρέπει να περιλαμβάνουν φυσιολογικές και μη φυσιολογικές εικόνες, καθώς και τις αντίστοιχες αληθείς ετικέτες τους. Έτσι, δημιουργήθηκε ένα νέο σύνολο δεδομένων εκπαίδευσης που αποτελείται από 500 φυσιολογικές εικόνες με ετικέτα 0 και 500 μη φυσιολογικές με ετικέτα 1. Οι 500 φυσιολογικές εικόνες είναι υποσύνολο των δεδομένων εκπαίδευσης του αυτόματου κωδικοποιητή. Οι 500 μη φυσιολογικές όμως αποτελούν υποσύνολο των δεδομένων ελέγχου, λόγω έλλειψης άλλων δεδομένων, και έτσι σημειώνεται ότι η απόδοση του CNN θα αξιολογηθεί με μικρότερο αριθμό δεδομένων ελέγχου. Τελικά, τα δεδομένα ελέγχου αποτελούνται από 1415 φυσιολογικές εικόνες και 1817 μη φυσιολογικές εικόνες.

Η μορφή του CNN που χρησιμοποιείται στην ενότητα αυτή παραμένει ίδια ανεξαρτήτως των δεδομένων που εισάγονται σε αυτό οπότε επιλέχθηκε να παρουσιαστεί σε αυτό το σημείο του κεφαλαίου.

```
def classifier():
  model = Sequential()
  model.add(Conv2D(32,3,padding="same", activation="relu",
input_shape=(45,45,1)))
  model.add(MaxPool2D())

  model.add(Conv2D(32, 3, padding="same", activation="relu"))
  model.add(MaxPool2D())

  model.add(Conv2D(64, 3, padding="same", activation="relu"))
  model.add(MaxPool2D())
  model.add(Dropout(0.4))

  model.add(Flatten())
```



Εικόνα 78: Σχηματική απεικόνιση CNN

```
  model.add(Dense(128,activation="relu"))
  model.add(Dense(2, activation="softmax"))


  model.compile(optimizer = 'adam' , loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True) , metrics =
['accuracy'])


  return model
```

Όπως φαίνεται το δίκτυο αποτελείται από τρία επίπεδα που πραγματοποιούν συνέλιξη και ακολουθούνται από επίπεδα που πραγματοποιούν max pooling. Τα επίπεδα max pooling μειώνουν τις διαστάσεις της εισόδου τους στο μισό επιλέγοντας τη μέγιστη τιμή των εικονοστοιχείων στο εσωτερικό ενός παραθύρου, το οποίο μετατοπίζεται κατά μήκος της κάθε διάστασης. [52] Εκτός των παραπάνω, στο δίκτυο περιλαμβάνεται και ένα επίπεδο dropout το οποίο προλαμβάνει το φαινόμενο της υπερπροσαρμογής (overfitting). Αυτό επιτυγχάνεται αποβάλλοντας προσωρινά τυχαία επιλεγμένες μονάδες (κρυφές και ορατές) από το δίκτυο σύμφωνα με μία συγκεκριμένη πιθανότητα σε κάθε βήμα της εκπαίδευσης. [19] Στη συγκεκριμένη περίπτωση το ποσοστό των μονάδων που αποβάλλονται αντιστοιχεί στο 0,4. Το επίπεδο flatten μετατρέπει τα δεδομένα σε πίνακα μίας διάστασης ώστε να είναι δυνατό να εισαχθούν στο επόμενο επίπεδο, δηλαδή στα πλήρως συνδεδεμένα στρώματα (dense) μέσω των οποίων γίνεται η ταξινόμηση σε φυσιολογικά και μη φυσιολογικά. [50]

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 45, 45, 32)        320
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 22, 22, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 22, 22, 32)        9248
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 11, 11, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               204928
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 258
=================================================================
Total params: 233,250
Trainable params: 233,250
Non-trainable params: 0
_________________________________________________________________
```

Εικόνα 79: Περίληψη CNN

Η συνάρτηση ενεργοποίησης που χρησιμοποιήθηκε στα συνελικτικά στρώματα και στο πρώτο πλήρως συνδεδεμένο είναι η ReLU (Rectified Linear Unit) η οποία αποδίδει την είσοδο που δέχεται χωρίς να τη μεταβάλει αν αυτή είναι θετική, αλλιώς η έξοδός της είναι 0. Στο τελευταίο επίπεδο χρησιμοποιήθηκε η συνάρτηση ενεργοποίησης softmax, η οποία μετατρέπει τα στοιχεία του διανύσματος που δέχεται σαν είσοδο, έτσι ώστε το άθροισμα τους να είναι 1 και έτσι τα αποτελέσματα να μπορούν να ερμηνευτούν σαν πιθανότητες. [51] Η μορφή του μοντέλου και οι διαστάσεις της εξόδου του κάθε επιπέδου φαίνονται στην παραπάνω περίληψη.

Τα δεδομένα εκπαίδευσης και ελέγχου σε κάθε μία από τις επόμενες περιπτώσεις διαβάζονται από τους αντίστοιχους συμπιεσμένους φακέλους και δεν πραγματοποιείται κανονικοποίηση. Παράλληλα, δημιουργούνται αληθείς ετικέτες με το 0 να αντιστοιχεί στις φυσιολογικές και το 1 στις μη φυσιολογικές εικόνες.

β) Εφαρμογή CNN σε κανονικά δεδομένα

Αφού διαβάστηκαν τα δεδομένα από τους αντίστοιχους φακέλους, οργανώθηκαν σε λίστες και στη συνέχεια αναδιαμορφώθηκαν ώστε να έχουν τις σωστές διαστάσεις. Τα δεδομένα εκπαίδευσης περιλαμβάνονται στη λίστα "x_train" και οι ετικέτες τους στη λίστα "y_train", ενώ τα δεδομένα ελέγχου περιλαμβάνονται στη λίστα "x_test" και οι ετικέτες τους στη λίστα "y_test".

```
x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
x_train = np.reshape(x_train, (-1, 45, 45, 1))
x_test = np.reshape(x_test, (-1, 45, 45, 1))
```

Χρησιμοποιώντας τη συνάρτηση "classifier" δημιουργείται το μοντέλο "model2" και εκπαιδεύεται για 10 εποχές.

```
model2 = classifier()
model2.fit(x_train, y_train, epochs = 10)
```

```
Epoch 1/10
32/32 [==============================] - 3s 84ms/step - loss: 3.9376 - accuracy: 0.5320
Epoch 2/10
32/32 [==============================] - 3s 84ms/step - loss: 0.4704 - accuracy: 0.7700
Epoch 3/10
32/32 [==============================] - 3s 82ms/step - loss: 0.3860 - accuracy: 0.8170
Epoch 4/10
32/32 [==============================] - 3s 82ms/step - loss: 0.3345 - accuracy: 0.8560
Epoch 5/10
32/32 [==============================] - 3s 83ms/step - loss: 0.1453 - accuracy: 0.9510
Epoch 6/10
32/32 [==============================] - 3s 83ms/step - loss: 0.2010 - accuracy: 0.9250
Epoch 7/10
32/32 [==============================] - 3s 82ms/step - loss: 0.1406 - accuracy: 0.9460
Epoch 8/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0426 - accuracy: 0.9880
Epoch 9/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0258 - accuracy: 0.9960
Epoch 10/10
32/32 [==============================] - 3s 82ms/step - loss: 0.0262 - accuracy: 0.9930
<keras.callbacks.History at 0x7f89de4dead0>
```

Εικόνα 80: Διαδικασία εκπαίδευσης CNN με κανονικά δεδομένα

Αφού εκπαιδευτεί το μοντέλο, εισάγονται σε αυτό τα δεδομένα ελέγχου και ζητείται να πραγματοποιηθούν προβλέψεις. Για κάθε στοιχείο των δεδομένων ελέγχου προκύπτει ένα διάνυσμα δύο στοιχείων με τις πιθανότητες να ανήκει το στοιχείο στην κάθε κλάση. Αν το διάνυσμα εμφανίζει μεγαλύτερη τιμή στη θέση 0, τότε η προβλεπόμενη ετικέτα είναι 0. Ομοίως αν εμφανίζει μεγαλύτερη τιμή στη θέση 1, τότε η προβλεπόμενη ετικέτα είναι 1.

```
predictions = model2.predict(x_test)
# predicted labels
y_predicted = []
for i in range(len(predictions)):
  if predictions[i][0] > predictions[i][1]:
    y_predicted.append(0)
  else:
    y_predicted.append(1)
```

Αφού δημιουργήθηκαν οι προβλεπόμενες ετικέτες συγκρίνονται με τις αληθείς και υπολογίζονται οι τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών, καθώς και οι τιμές των accuracy, precision, recall και F1 score.

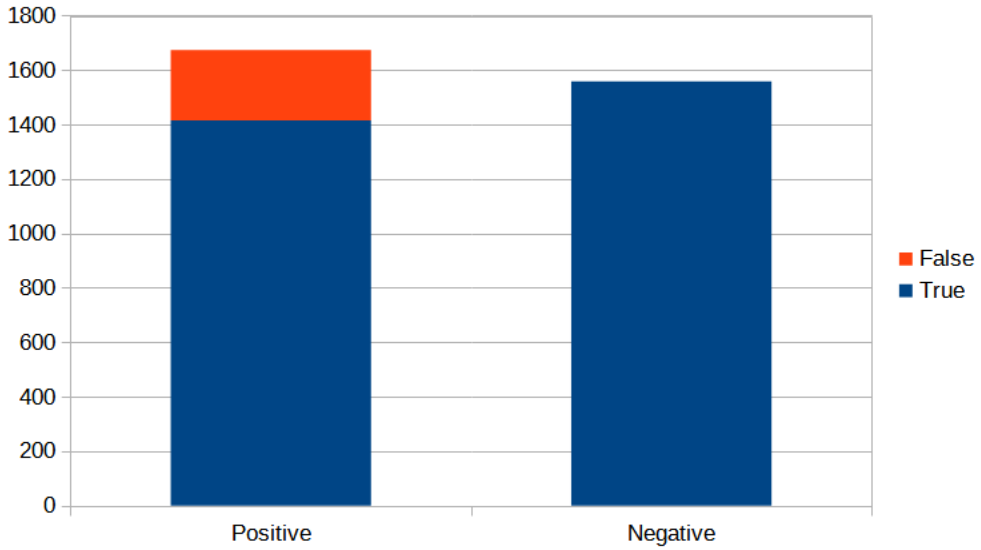
```
TP = 0
TN = 0
FP = 0
FN = 0

for i in range(len(y_test)):
  if y_test[i] == 0 and y_predicted[i] == 0:
    TP = TP + 1
  elif y_test[i] == 1 and y_predicted[i] == 1:
    TN = TN + 1
  elif y_test[i] == 0 and y_predicted[i] == 1:
    FN = FN + 1
  else:
    FP = FP + 1

# accuracy
accuracy = (TP + TN)/(TP + FP + FN + TN)
# precision
precision = TP/(TP + FP)
# recall
```

```
recall = TP/(TP + FN)
# f1 score
f1_score = 2*(recall*precision)/(recall + precision)
```

Οι τιμές που προέκυψαν φαίνονται στους παρακάτω πίνακες και διαγράμματα.

True Positive (TP)	1415
False Positive (FP)	259
True Negative (TN)	1558
False Negative (FN)	0

Πίνακας 21: Πίνακας TP, TN, FP, FN για εφαρμογή CNN με κανονικά δεδομένα



Εικόνα 81: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή CNN με κανονικά δεδομένα

Όπως φαίνεται παραπάνω, το μοντέλο εντόπισε με πολύ μεγάλη επιτυχία τις φυσιολογικές εικόνες και τους απέδωσε την ετικέτα 0 χωρίς να κάνει λάθος, πράγμα που φαίνεται από την απουσία ψευδώς αρνητικών στοιχείων. Από την άλλη πλευρά, υπάρχουν 259 εικόνες με ρωγμές, οι οποίες θεωρήθηκε από το μοντέλο ότι είναι φυσιολογικές. Το πλήθος αυτό, αν και δεν είναι εξίσου μεγάλο με το πλήθος των ψευδώς θετικών στις περιπτώσεις του K-means και του GMM, είναι μεγαλύτερο από το αντίστοιχο πλήθος στην περίπτωση του αυτόματου κωδικοποιητή, ο οποίος σημείωσε 82 ψευδώς θετικά στοιχεία στην περίπτωση των κανονικών δεδομένων.

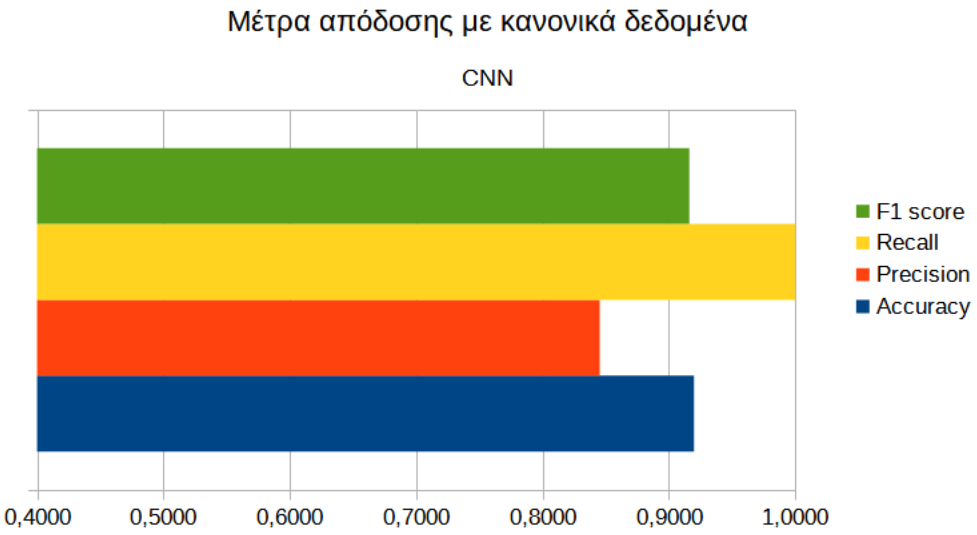
Λαμβάνοντας τα παραπάνω υπόψιν, υπολογίστηκαν τα μέτρα απόδοσης accuracy, precision, recall και F1 score, από τα οποία προκύπτει μία αντικειμενική εκτίμηση της ποιότητας της ταξινόμησης. Οι τιμές τους φαίνονται στον παρακάτω πίνακα και στο διάγραμμα που ακολουθεί.

Accuracy	0,9199
Precision	0,8453
Recall	1,0000
F1 Score	0,9162

Πίνακας 22: Μέτρα απόδοσης CNN με κανονικά δεδομένα

Όπως αναφέρθηκε και παραπάνω, η απόδοση του CNN στην ταξινόμηση των δεδομένων φαίνεται να είναι ικανοποιητική. Συγκεκριμένα, οι τιμές των μέτρων απόδοσης πλησιάζουν πολύ τις αντίστοιχες τιμές του αυτόματου κωδικοποιητή, παρουσιάζοντας μεγαλύτερη επιτυχία στον προσδιορισμό των φυσιολογικών εικόνων, όπως φαίνεται από την παράμετρο recall, ενώ υστερεί ελαφρώς στην ταξινόμηση των μη φυσιολογικών εικόνων, όπως δείχνει η τιμή του precision.

Αν και η επιτυχία στις μη φυσιολογικές εικόνες είναι σημαντικότερη από την επιτυχία στις φυσιολογικές, το CNN πέτυχε τον διαχωρισμό των εικόνων καλύτερα από το K-means και το GMM, οπότε φαίνεται να είναι ο επικρατέστερος αλγόριθμος στα πλαίσια του κεφαλαίου αυτού με εξαίρεση των αυτόματο κωδικοποιητή. Παρακάτω φαίνονται τα μέτρα απόδοσης σε μορφή διαγράμματος.



Εικόνα 82: Διαγραμματική παρουσίαση των μέτρων απόδοσης του CNN με κανονικά δεδομένα

γ) Εφαρμογή CNN σε δεδομένα με εξομάλυνση

Στην ενότητα αυτή θα εφαρμοστούν όσα περιγράφηκαν παραπάνω για την περίπτωση των κανονικών δεδομένων, όμως αυτή τη φορά στα δεδομένα εκπαίδευσης και ελέγχου θα εφαρμοστεί εξομάλυνση μέσω του φίλτρου Gauss.

Τα δεδομένα εκπαίδευσης και ελέγχου αρχικά διαβάζονται από τους αντίστοιχους φακέλους και εφαρμόζεται το φίλτρο Gauss ώστε να επιτευχθεί η εξομάλυνση. Τα δεδομένα εκπαίδευσης

οργανώνονται στη λίστα “b_x_train” με τις αληθείς ετικέτες τους στη λίστα “b_y_train”. Παράλληλα, τα δεδομένα ελέγχου οργανώνονται στη λίστα “b_x_test” και “b_y_test”. Οι λίστες αυτές μετατρέπονται σε numpy array και αναδιαμορφώνονται ώστε οι διαστάσεις του πίνακα της κάθε εικόνας να είναι (45, 45, 1).

```
b_x_train = np.array(b_x_train)
b_x_test = np.array(b_x_test)
b_y_train = np.array(b_y_train)
b_y_test = np.array(b_y_test)

b_x_train = np.reshape(b_x_train, (-1, 45, 45, 1))
b_x_test = np.reshape(b_x_test, (-1, 45, 45, 1))
```

Έπειτα καλείται η συνάρτηση “classifier” και δημιουργείται το μοντέλο, το οποίο εκπαιδεύεται και πάλι για 10 εποχές.

```
b_model2 = classifier()
b_model2.fit(b_x_train, b_y_train, epochs = 10, shuffle=True)
```

```
Epoch 1/10
32/32 [==============================] - 3s 83ms/step - loss: 2.6163 - accuracy: 0.5590
Epoch 2/10
32/32 [==============================] - 3s 81ms/step - loss: 0.5377 - accuracy: 0.7110
Epoch 3/10
32/32 [==============================] - 3s 81ms/step - loss: 0.6498 - accuracy: 0.7260
Epoch 4/10
32/32 [==============================] - 3s 81ms/step - loss: 0.5170 - accuracy: 0.7760
Epoch 5/10
32/32 [==============================] - 3s 81ms/step - loss: 0.1474 - accuracy: 0.9550
Epoch 6/10
32/32 [==============================] - 3s 80ms/step - loss: 0.2395 - accuracy: 0.9090
Epoch 7/10
32/32 [==============================] - 3s 82ms/step - loss: 0.0735 - accuracy: 0.9820
Epoch 8/10
32/32 [==============================] - 3s 82ms/step - loss: 0.0171 - accuracy: 0.9960
Epoch 9/10
32/32 [==============================] - 3s 81ms/step - loss: 0.0104 - accuracy: 0.9970
Epoch 10/10
32/32 [==============================] - 3s 79ms/step - loss: 0.0072 - accuracy: 0.9970
<keras.callbacks.History at 0x7f89e4c4be10>
```

Εικόνα 83: Διαδικασία εκπαίδευσης μοντέλου CNN με δεδομένα με εξομάλυνση

Αφού εκπαιδεύτηκε το μοντέλο, χρησιμοποιούνται τα δεδομένα ελέγχου, τα οποία εισάγονται στο μοντέλο και ζητείται να πραγματοποιήσει προβλέψεις. Οι προβλέψεις του μοντέλου έχουν τη μορφή πίνακα, με την κάθε γραμμή να αντιστοιχεί σε μία εικόνα από τα δεδομένα ελέγχου. Η κάθε γραμμή αποτελείται από δύο στοιχεία τα οποία δείχνουν την πιθανότητα να ανήκει η εκάστοτε εικόνα στην κάθε κλάση. Αν η πιθανότητα στη θέση 0 είναι μεγαλύτερη από την πιθανότητα στη θέση 1, τότε

η εικόνα ανήκει στις φυσιολογικές και το αντίστροφο. Με αυτό τον τρόπο δημιουργείται μια λίστα με τις προβλεπόμενες ετικέτες οι οποίες θα συγκριθούν με τις αληθείς για να αξιολογηθεί η απόδοση του μοντέλου.

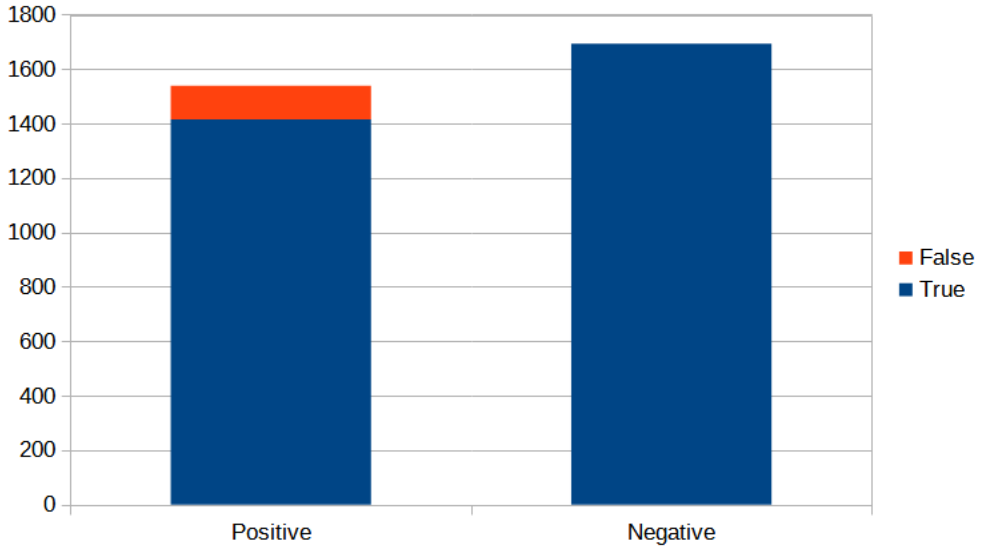
```
b_predictions = b_model2.predict(b_x_test)
# predicted labels
b_y_predicted = []
for i in range(len(b_predictions)):
  if b_predictions[i][0] > b_predictions[i][1]:
    b_y_predicted.append(0)
  else:
    b_y_predicted.append(1)
```

Έχοντας τις αληθείς και τις προβλεπόμενες ετικέτες για κάθε εικόνα, υπολογίζεται το πλήθος των αληθινά θετικών, των αληθινά αρνητικών, των ψευδώς θετικών και των ψευδώς αρνητικών. Δηλαδή, υπολογίζεται το πλήθος των εικόνων που το μοντέλο ταξινόμησε σωστά και λανθασμένα.

```
b_TP = 0
b_TN = 0
b_FP = 0
b_FN = 0

for i in range(len(b_y_test)):
  if b_y_test[i] == 0 and b_y_predicted[i] == 0:
    b_TP = b_TP + 1
  elif b_y_test[i] == 1 and b_y_predicted[i] == 1:
    b_TN = b_TN + 1
  elif b_y_test[i] == 0 and b_y_predicted[i] == 1:
    b_FN = b_FN + 1
  else:
    b_FP = b_FP + 1
```

True Positive (TP)	1415
False Positive (FP)	124
True Negative (TN)	1693
False Negative (FN)	0

Πίνακας 23: Πίνακας TP, TN, FP, FN για εφαρμογή CNN με εξομάλυνση

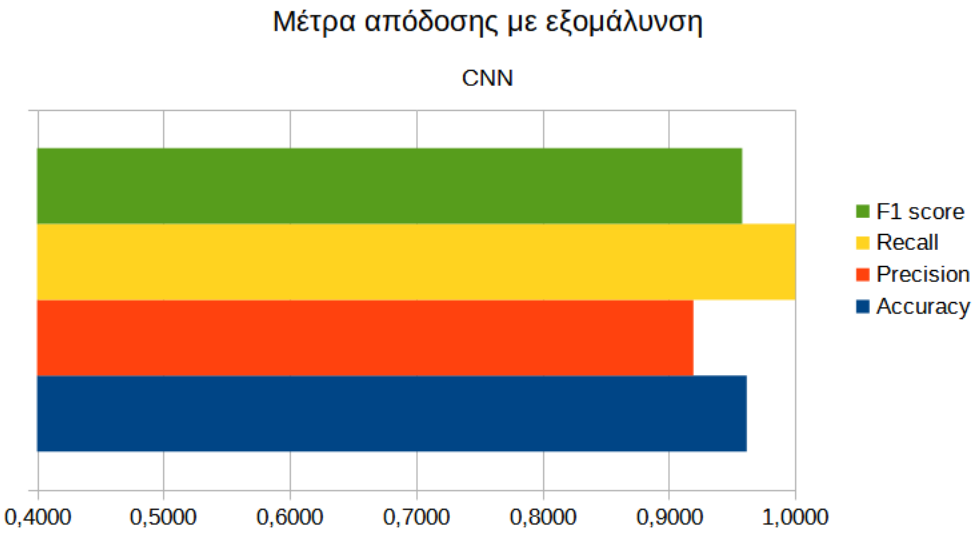


Εικόνα 84: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή CNN με εξομάλυνση

Με την εξομάλυνση των εικόνων, όπως έχει ήδη αναφερθεί, αναμένεται το έργο ταξινόμησης ή ομαδοποίησης που αναθέτεται στον εκάστοτε αλγόριθμο να δυσχεραίνεται λόγω της μεγαλύτερης ομοιότητας των μη φυσιολογικών εικόνων με τις ομοιόμορφες φυσιολογικές εικόνες. Σε όλους τους αλγορίθμους που έχουν εφαρμοστεί στα πλαίσια της παρούσας εργασίας αυτό φαίνεται να συμβαίνει και οι αλγόριθμοι φαίνεται να επηρεάζονται σε κάποιο βαθμό από την εξομάλυνση παρουσιάζοντας χειρότερη απόδοση. Αντίθετα, το CNN φαίνεται να ταξινόμησε τα δεδομένα με εξομάλυνση με μεγαλύτερη επιτυχία. Συγκεκριμένα, πέτυχε και πάλι να αποδώσει τη σωστή ετικέτα σε όλες τις φυσιολογικές εικόνες, ενώ ακόμα και το πλήθος των ψευδώς θετικών μειώθηκε, πλησιάζοντας περισσότερο τα αποτελέσματα του αυτόματου κωδικοποιητή.

Accuracy	0,9616
Precision	0,9194
Recall	1,0000
F1 Score	0,9580

Πίνακας 24: Μέτρα απόδοσης CNN με εξομάλυνση



Εικόνα 85: Διαγραμματική παρουσίαση των μέτρων απόδοσης του CNN με εξομάλυνση

Όπως φαίνεται παραπάνω, τα ποσοστά των accuracy, precision, recall και F1 score λαμβάνουν πολύ υψηλές τιμές σε αυτή την περίπτωση, ξεπερνώντας και την απόδοση του αυτόματου κωδικοποιητή. Σημειώνεται ότι το CNN αξιολογείται με λιγότερα δεδομένα ελέγχου από τον αυτόματο κωδικοποιητή αφού μέρος των μη φυσιολογικών εικόνων χρησιμοποιήθηκε σαν δεδομένα εκπαίδευσης. Βέβαια, τα ποσοστά αυτά υπολογίστηκαν λαμβάνοντας υπόψιν και την απόδοση του μοντέλου στον προσδιορισμό φυσιολογικών εικόνων. Στον προσδιορισμό μη φυσιολογικών εικόνων, πράγμα που έχει ιδιαίτερη σημασία, το CNN φαίνεται να παρουσιάζει όμοια απόδοση με τον αυτόματο κωδικοποιητή, λαμβάνοντας υπόψιν ότι ελέγχεται με λιγότερες μη φυσιολογικές εικόνες στο σύνολο δεδομένων ελέγχου του.

δ) Εφαρμογή CNN σε δεδομένα με θόρυβο

Στην περίπτωση αυτή ακολουθείται όμοια διαδικασία με τα προηγούμενα, προσθέτοντας όμως στα δεδομένα εκπαίδευσης και ελέγχου θόρυβο τύπου αλάτι – πιπέρι με τη βοήθεια της συνάρτησης “sp_noise”, η οποία εφαρμόζεται στα δεδομένα αφού διαβαστούν από τους αντίστοιχους φακέλους. Τα δεδομένα εκπαίδευσης οργανώνονται στη λίστα “n_x_train” και οι ετικέτες τους στη λίστα “n_y_train”, ενώ τα δεδομένα ελέγχου στη λίστα “n_x_test” και οι ετικέτες τους στη λίστα “n_y_test”. Οι λίστες στη συνέχεια μετατρέπονται σε numpy arrays και αναδιαμορφώνονται ώστε να έχουν τις επιθυμητές διαστάσεις.

```
n_x_train = np.array(n_x_train)
n_x_test = np.array(n_x_test)
n_y_train = np.array(n_y_train)
n_y_test = np.array(n_y_test)

n_x_train = np.reshape(n_x_train, (-1, 45, 45, 1))
n_x_test = np.reshape(n_x_test, (-1, 45, 45, 1))
```

Καλώντας τη συνάρτηση “classifier” δημιουργείται το μοντέλο και στη συνέχεια εκπαιδεύεται για 10 εποχές.

```
n_model2 = n_classifier()
n_model2.fit(n_x_train, n_y_train, epochs = 10)
```

```
Epoch 1/10
32/32 [==============================] - 4s 86ms/step - loss: 3.4356 - accuracy: 0.5320
Epoch 2/10
32/32 [==============================] - 3s 84ms/step - loss: 0.5261 - accuracy: 0.7210
Epoch 3/10
32/32 [==============================] - 3s 83ms/step - loss: 0.3398 - accuracy: 0.8560
Epoch 4/10
32/32 [==============================] - 3s 85ms/step - loss: 0.3018 - accuracy: 0.8780
Epoch 5/10
32/32 [==============================] - 3s 84ms/step - loss: 0.3549 - accuracy: 0.8520
Epoch 6/10
32/32 [==============================] - 3s 85ms/step - loss: 0.2323 - accuracy: 0.9050
Epoch 7/10
32/32 [==============================] - 3s 84ms/step - loss: 0.1489 - accuracy: 0.9420
Epoch 8/10
32/32 [==============================] - 3s 84ms/step - loss: 0.1565 - accuracy: 0.9480
Epoch 9/10
32/32 [==============================] - 3s 84ms/step - loss: 0.1067 - accuracy: 0.9640
Epoch 10/10
32/32 [==============================] - 3s 83ms/step - loss: 0.1467 - accuracy: 0.9440
<keras.callbacks.History at 0x7fd50b0faf10>
```

Εικόνα 86: Διαδικασία εκπαίδευσης του CNN με δεδομένα με θόρυβο

Έχοντας το εκπαιδευμένο μοντέλο, είναι δυνατό να χρησιμοποιηθούν τα δεδομένα ελέγχου για την πραγματοποίηση προβλέψεων και τη σύγκριση των προβλεπόμενων ετικετών με τις αληθείς.

```
n_predictions = n_model2.predict(n_x_test)
# predicted labels
n_y_predicted = []
for i in range(len(n_predictions)):
  if n_predictions[i][0] > n_predictions[i][1]:
    n_y_predicted.append(0)
  else:
    n_y_predicted.append(1)
```

Συγκρίνοντας τις προβλεπόμενες ετικέτες με τις αληθείς προκύπτουν οι τιμές των αληθινά θετικών, αληθινά αρνητικών, ψευδώς θετικών και ψευδώς αρνητικών που φαίνονται παρακάτω.

```
n_TP = 0
n_TN = 0
```

```
n_FP = 0
n_FN = 0

for i in range(len(n_y_test)):
  if n_y_test[i] == 0 and n_y_predicted[i] == 0:
    n_TP = n_TP + 1
  elif n_y_test[i] == 1 and n_y_predicted[i] == 1:
    n_TN = n_TN + 1
  elif n_y_test[i] == 0 and n_y_predicted[i] == 1:
    n_FN = n_FN + 1
  else:
    n_FP = n_FP + 1
```

True Positive (TP)	941
False Positive (FP)	703
True Negative (TN)	1114
False Negative (FN)	474

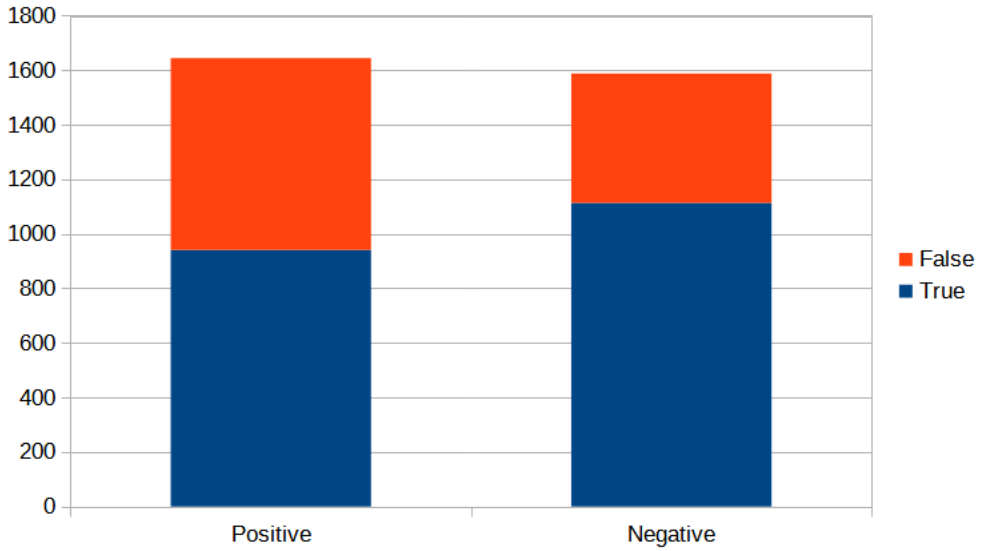
Πίνακας 25: Πίνακας TP, TN, FP, FN για εφαρμογή CNN με θόρυβο

Όπως φαίνεται από τον παραπάνω πίνακα και από το διάγραμμα που ακολουθεί, η προσθήκη θορύβου επηρέασε και πάλι σημαντικά την ποιότητα της ταξινόμησης. Αυτή τη φορά πολλές από τις φυσιολογικές εικόνες θεωρήθηκαν μη φυσιολογικές, όπως φαίνεται από το πλήθος των ψευδώς αρνητικών το οποίο στις προηγούμενες περιπτώσεις ήταν 0. Παράλληλα το πλήθος των ψευδώς θετικών επίσης αυξήθηκε σημαντικά αφού περίπου 700 μη φυσιολογικές εικόνες δεν αναγνωρίστηκαν και θεωρήθηκαν φυσιολογικές. Τα παραπάνω αντικατοπτρίζονται και στα ποσοστά accuracy, precision, recall και F1 score, στα οποία παρατηρείται κατακόρυφη πτώση με την προσθήκη θορύβου.

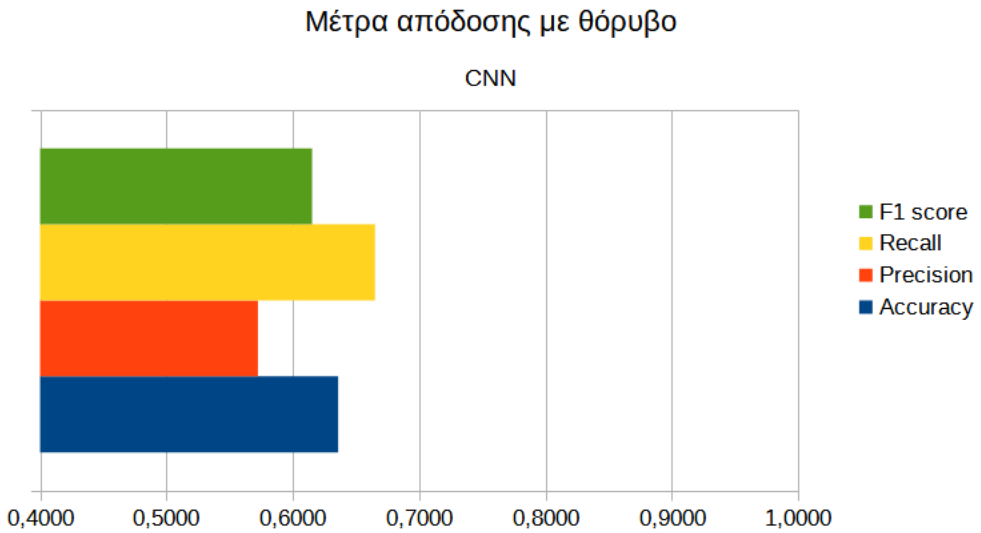
Συνολικά, το CNN αν και ταξινόμησε τα δεδομένα με μεγάλη ακρίβεια στις περιπτώσεις των κανονικών δεδομένων και των δεδομένων με εξομάλυνση, στην περίπτωση της ύπαρξης θορύβου παρουσίασε μεγάλη αστάθεια και η ταξινόμηση πραγματοποιήθηκε με πολύ χαμηλότερο βαθμό επιτυχίας. Παρακάτω φαίνονται σε μορφή διαγράμματος οι τιμές των TP, TN, FP και FN καθώς και οι τιμές των accuracy, precision, recall και F1 score σε μορφή πίνακα και διαγράμματος.

Accuracy	0,6358
Precision	0,5724
Recall	0,6650
F1 Score	0,6152

Πίνακας 26: Μέτρα απόδοσης CNN με θόρυβο

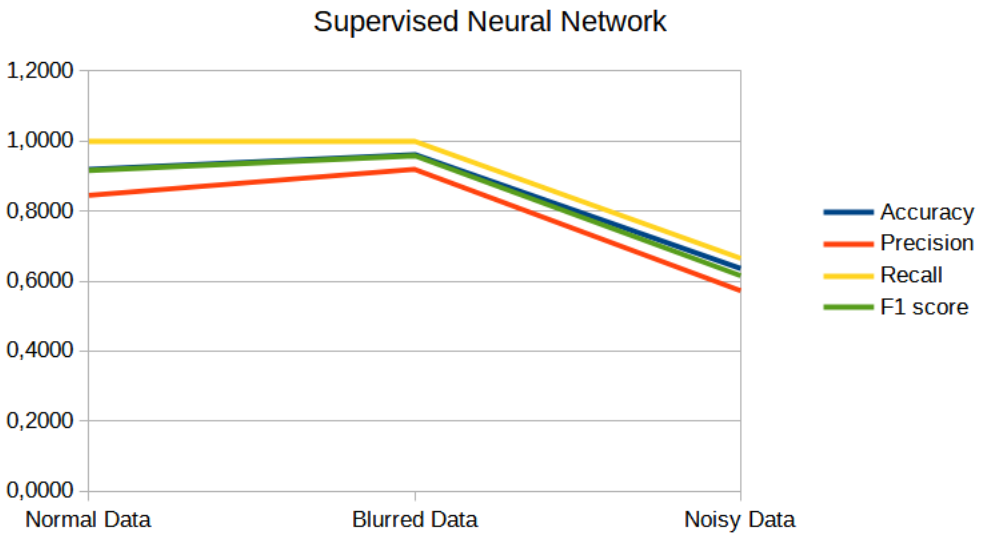


Εικόνα 87: Διαγραμματική απεικόνιση TP, TN, FP, FN για εφαρμογή CNN με θόρυβο



Εικόνα 88: Διαγραμματική παρουσίαση των μέτρων απόδοσης του CNN με θόρυβο

ε) Σύγκριση αποτελεσμάτων για το CNN



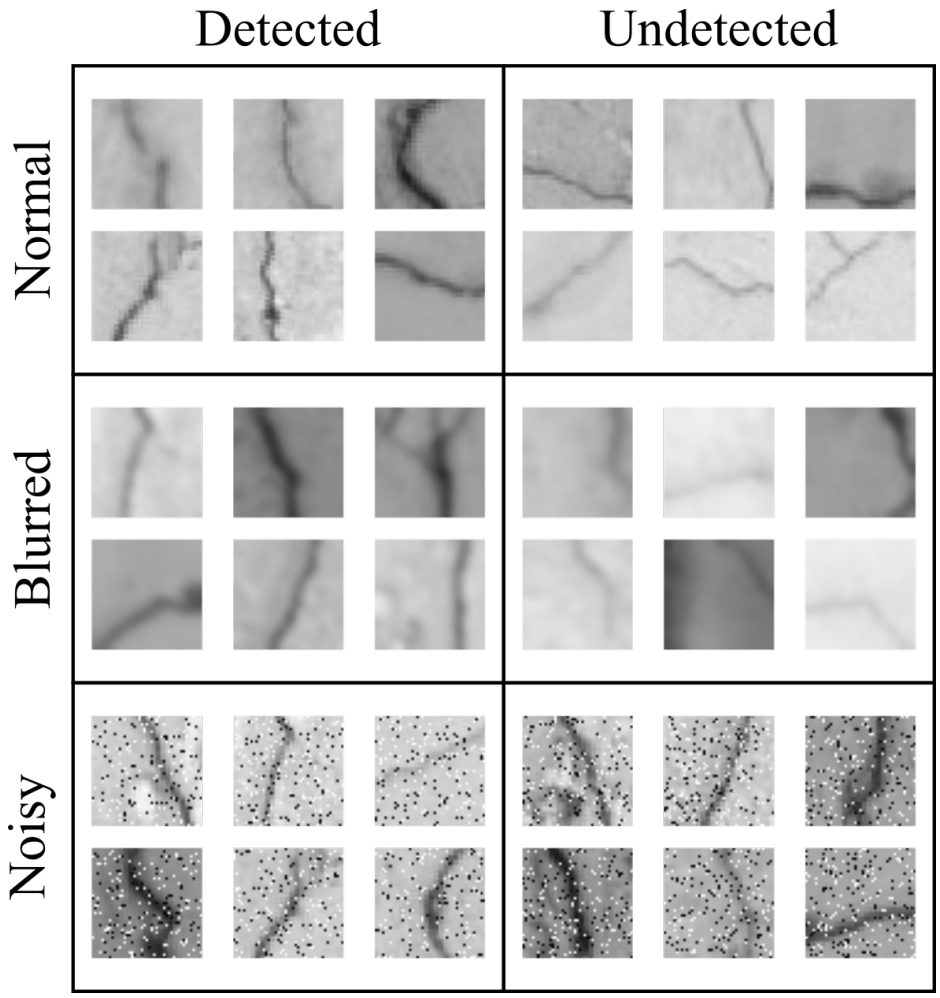
Εικόνα 89: Συγκεντρωτικό διάγραμμα μέτρων απόδοσης για το CNN

Στο παραπάνω διάγραμμα στον οριζόντιο άξονα φαίνεται το είδος των δεδομένων (κανονικά, με εξομάλυνση και με θόρυβο) και στον κατακόρυφο φαίνονται οι τιμές των accuracy, precision, recall και F1 score.

- Accuracy και F1 score: Οι ευθείες των δύο αυτών παραμέτρων φαίνεται σχεδόν να ταυτίζονται στο μεγαλύτερο μέρος του διαγράμματος, αφού και οι δύο λαμβάνουν υπόψιν στον υπολογισμό τους το πλήθος των TP, TN, FP και FN. Συγκεκριμένα, στην περίπτωση των κανονικών δεδομένων φαίνεται και οι δύο παράμετροι να λαμβάνουν μία τιμή κοντά στο 92% πράγμα που δείχνει ότι η ταξινόμηση έγινε επιτυχώς. Στη συνέχεια, ενώ αναμενόταν πτώση στην περίπτωση των δεδομένων με εξομάλυνση, τελικά παρατηρήθηκε αύξηση και οι παράμετροι έλαβαν τιμή κοντά στο 96%, επιβεβαιώνοντας ότι και σε αυτή την περίπτωση η ταξινόμηση πραγματοποιήθηκε επιτυχώς. Στην περίπτωση της ύπαρξης θορύβου όμως παρατηρήθηκε κατακόρυφη πτώση με το accuracy να λαμβάνει τιμή κοντά στο 64% και το F1 score κοντά στο 62%, δείχνοντας έτσι ότι το μοντέλο επηρεάστηκε αισθητά από την προσθήκη θορύβου.
- Precision: Η παράμετρος precision λαμβάνει τις χαμηλότερες τιμές σε σχέση με τα υπόλοιπα τρία μέτρα απόδοσης επειδή για τον υπολογισμό της λαμβάνει υπόψιν το πλήθος των ψευδώς θετικών στοιχείων. Στα κανονικά δεδομένα η τιμή του υπολογίστηκε κοντά στο 85%, πράγμα που δείχνει ότι, αν και η ταξινόμηση ήταν γενικά επιτυχής, παρατηρήθηκαν μερικά ψευδώς θετικά στοιχεία. Στην περίπτωση των δεδομένων με εξομάλυνση παρατηρήθηκε η μέγιστη τιμή του precision, το οποίο υπολογίστηκε κοντά στο 92% και το πλήθος των ψευδώς θετικών στην περίπτωση αυτή μειώθηκε, πλησιάζοντας την αντίστοιχη τιμή με χρήση του αυτόματου κωδικοποιητή. Τέλος, στην περίπτωση του θορύβου, παρατηρήθηκε και πάλι αισθητή μείωση, με την τιμή της παραμέτρου να υπολογίζεται κοντά στο 57%, αφού παρατηρήθηκαν περίπου 700 ψευδώς θετικά στοιχεία.
- Recall: Η παράμετρος recall, όπως αναφέρθηκε και σε προηγούμενες ενότητες, εξαρτάται μόνο από την επιτυχία εντοπισμού των θετικών, δηλαδή των φυσιολογικών εικόνων. Το μοντέλο έδειξε σε γενικές γραμμές να μπορεί με μεγάλη επιτυχία να προσδιορίσει τις φυσιολογικές

εικόνες σε όλες τις περιπτώσεις, με εξαίρεση την περίπτωση του θορύβου. Συγκεκριμένα, όταν έγινε χρήση κανονικών και δεδομένων με εξομάλυνση, η τιμή του recall βρισκόταν σταθερά στο 1, αφού δεν υπήρχαν ψευδώς αρνητικά στοιχεία. Η προσθήκη θορύβου όμως αποπροσανατόλισε το μοντέλο και έτσι παρατηρήθηκαν σχεδόν 500 ψευδώς αρνητικά στοιχεία, ρίχνοντας την τιμή του recall στο 67%.

Στην εικόνα που ακολουθεί φαίνονται ενδεικτικά κάποιες εικόνες με ρωγμές που το μοντέλο εντόπισε επιτυχημένα ή απέτυχε να εντοπίσει. Όπως φαίνεται, στην περίπτωση των φυσιολογικών δεδομένων και των δεδομένων με εξομάλυνση το μοντέλο εντόπισε τις χαρακτηριστικές ρωγμές, ενώ αγνόησε τις δυσδιάκριτες. Στην περίπτωση του θορύβου από την άλλη πλευρά φαίνεται και πάλι ότι το μοντέλο οδηγήθηκε σε λανθασμένα συμπεράσματα, θεωρώντας εικόνες με εμφανείς ρωγμές σαν φυσιολογικές.



Εικόνα 90: Ενδεικτικές εικόνες με ρωγμές που εντόπισε και που δεν εντόπισε το CNN

6 *Συμπεράσματα*

Αφού δοκιμάστηκε ο αυτόματος κωδικοποιητής και οι υπόλοιποι αλγόριθμοι σε κανονικά δεδομένα, δεδομένα με εξομάλυνση και δεδομένα με θόρυβο, στο κεφάλαιο αυτό θα συγκριθούν τα αποτελέσματα και θα εξαχθούν τελικά συμπεράσματα.

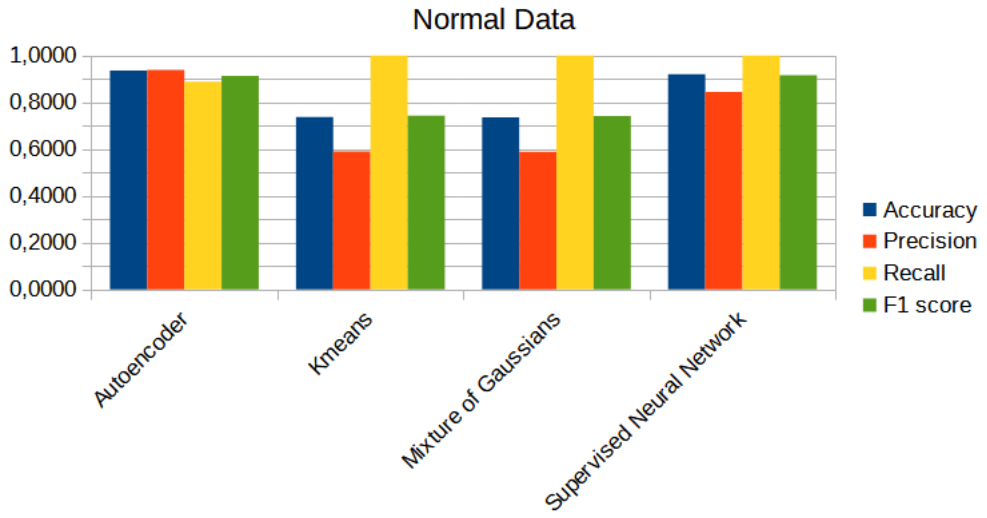
6.1 Συνολική σύγκριση αποτελεσμάτων

α) Κανονικά δεδομένα

Στο προηγούμενο κεφάλαιο, με σκοπό να αξιολογηθεί η απόδοση του αυτόματου κωδικοποιητή, εφαρμόστηκαν στα κανονικά δεδομένα οι αλγόριθμοι K-means, GMM και ένα επιβλεπόμενο νευρωνικό δίκτυο (CNN). Στο σημείο αυτό θα συγκριθούν όλοι οι αλγόριθμοι μεταξύ τους με βάση τις παραμέτρους accuracy, precision, recall και F1 score που προέκυψαν κατά την εφαρμογή τους.

Normal Data				
	Accuracy	Precision	Recall	F1 score
Autoencoder	0,9360	0,9388	0,8890	0,9132
Kmeans	0,7371	0,5906	1,0000	0,7426
Mixture of Gaussians	0,7358	0,5893	1,0000	0,7416
Supervised Neural Network	0,9199	0,8453	1,0000	0,9162

Πίνακας 27: Συγκεντρωτικός πίνακας αποτελεσμάτων για τα κανονικά δεδομένα



Εικόνα 91: Διαγραμματική παρουσίαση αποτελεσμάτων για τα κανονικά δεδομένα

Όπως φαίνεται στον παραπάνω πίνακα και στο παραπάνω διάγραμμα οι αλγόριθμοι K-means και GMM, αν και πέτυχαν τη μέγιστη τιμή του recall, πράγμα που δείχνει επιτυχημένη ομαδοποίηση των φυσιολογικών εικόνων, στις υπόλοιπες παραμέτρους σημειώνουν αρκετά χαμηλές τιμές. Συγκεκριμένα, και οι δύο αλγόριθμοι φαίνεται να υστερούν στον εντοπισμό των ομοιοτήτων μεταξύ των μη φυσιολογικών εικόνων, σημειώνοντας τιμή precision κοντά στο 60%. Παράλληλα, οι τιμές των accuracy και F1 score κυμαίνονται σε μέτρια επίπεδα, δηλαδή κοντά στο 74%. Συνεπώς, φαίνεται ότι οι δύο αυτοί αλγόριθμοι είναι ακατάλληλοι για αναγνώριση ρωγμών με τα κανονικά δεδομένα.

Η απόδοση του CNN, από την άλλη πλευρά, ήταν εμφανώς καλύτερη. Όπως φαίνεται παραπάνω, και στην περίπτωση αυτή το recall σημείωσε τη μέγιστη δυνατή τιμή, επιτυγχάνοντας τέλεια ταξινόμηση των φυσιολογικών εικόνων. Το precision από την άλλη είναι αισθητά χαμηλότερο διατηρώντας όμως την τιμή του σε υψηλά επίπεδα, δηλαδή κοντά στο 85%. Συνολικά, σύμφωνα με το accuracy και το F1 score που έλαβαν τιμές κοντά στο 92%, το CNN είναι μια ικανοποιητική λύση για τη συγκεκριμένη εφαρμογή.

Ο αυτόματος κωδικοποιητής πέτυχε συνολικά εξίσου καλή απόδοση στον διαχωρισμό των φυσιολογικών και μη φυσιολογικών εικόνων, ενώ παρατηρήθηκε ότι και οι τέσσερις παράμετροι

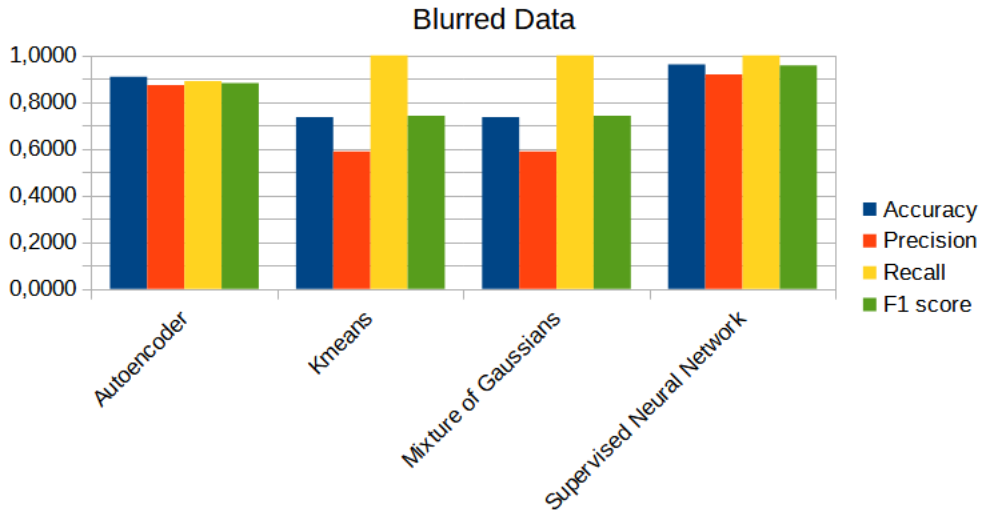
έλαβαν κοντινές τιμές. Αν και δεν επιτεύχθηκε απόλυτα επιτυχής διαχωρισμός των φυσιολογικών εικόνων όπως στις προηγούμενες περιπτώσεις και η τιμή του recall κατέληξε κοντά στο 89%, στις μη φυσιολογικές εικόνες ο αυτόματος κωδικοποιητής υπερτερεί των προηγούμενων αλγορίθμων αφού η τιμή του precision πλησιάζει το 94%.

Τελικά, στην περίπτωση των κανονικών δεδομένων, η βέλτιστη απόδοση θεωρείται ότι σημειώθηκε από τον αυτόματο κωδικοποιητή, με το CNN να ακολουθεί. Αν και οι δύο αλγόριθμοι πέτυχαν συνολικά πολύ καλή απόδοση, το CNN υστερεί στην αναγνώριση των μη φυσιολογικών εικόνων στην οποία δίνεται μεγαλύτερη βαρύτητα.

β) Δεδομένα με εξομάλυνση

Blurred Data				
	Accuracy	Precision	Recall	F1 score
Autoencoder	0,9092	0,8731	0,8898	0,8813
Kmeans	0,7355	0,5891	1,0000	0,7414
Mixture of Gaussians	0,7355	0,5891	1,0000	0,7414
Supervised Neural Network	0,9616	0,9194	1,0000	0,9580

Πίνακας 28: Συγκεντρωτικός πίνακας αποτελεσμάτων για δεδομένα με εξομάλυνση



Εικόνα 92: Διαγραμματική παρουσίαση αποτελεσμάτων για δεδομένα με εξομάλυνση

Η προσθήκη εξομάλυνσης δεν φαίνεται να επηρέασε ιδιαίτερα τους αλγορίθμους αφού οι τιμές των accuracy, precision, recall και F1 score δεν έχουν μεγάλη διαφορά από τις αντίστοιχες τιμές για τα κανονικά δεδομένα. Σύμφωνα με τα παραπάνω, οι τιμές των παραμέτρων που υπολογίστηκαν για τους αλγορίθμους K-means και GMM σχεδόν ταυτίζονται, αφού διαφοροποιούνται μετά το τέταρτο

δεκαδικό ψηφίο. Για τους αλγορίθμους αυτούς παρατηρείται και πάλι συνολικά μέτρια απόδοση, η οποία όμως υστερεί στην περίπτωση των μη φυσιολογικών εικόνων (πράγμα που φαίνεται στην τιμή του precision). Αφού το πλήθος των ψευδώς θετικών στοιχείων, δηλαδή των μη φυσιολογικών εικόνων που θεωρούνται φυσιολογικές, έχει ιδιαίτερη σημασία στη συγκεκριμένη εφαρμογή, οι αλγόριθμοι αυτοί και πάλι δεν είναι οι προτιμώμενοι.

Ο αυτόματος κωδικοποιητής φαίνεται να έχει εμφανώς καλύτερη απόδοση, παρουσιάζοντας την ίδια επιτυχία στα φυσιολογικά και στα μη φυσιολογικά δεδομένα. Η τιμή του recall σχεδόν ταυτίζεται με την αντίστοιχη τιμή για τα κανονικά δεδομένα, δείχνοντας ότι το μοντέλο δεν συνάντησε επιπλέον δυσκολία στον εντοπισμό των κανονικών δεδομένων, όπως αναμενόταν. Από την άλλη πλευρά, το precision σημείωσε την αναμενόμενη πτώση, διατηρώντας όμως την τιμή του κοντά στο 87%. Συνολικά, ο αυτόματος κωδικοποιητής και πάλι διαχώρισε με επιτυχία τις εικόνες, με την τιμή του accuracy να βρίσκεται κοντά στο 91%.

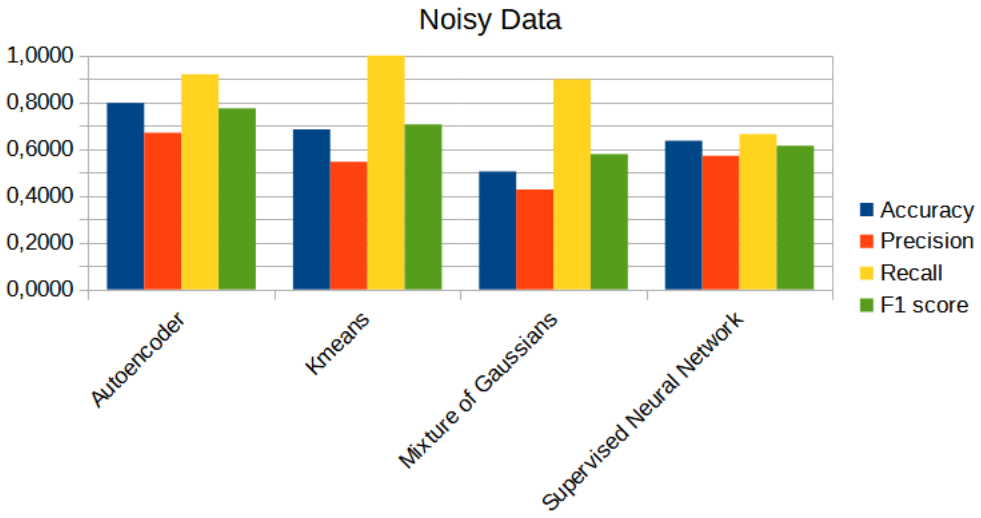
Το CNN από την άλλη πλευρά φαίνεται να διαχώρισε τα δεδομένα με εξομάλυνση με μεγαλύτερη επιτυχία σε σχέση με τα κανονικά δεδομένα, αν και αναμενόταν να παρατηρηθεί μια μικρή πτώση στην απόδοσή του. Το accuracy και το F1 score έλαβαν τιμές κοντά στο 96%, πράγμα που ξεπερνά αισθητά τα αντίστοιχα ποσοστά των άλλων αλγορίθμων. Παράλληλα, το recall και πάλι έλαβε την τιμή του 100%, ενώ το precision πλησίασε το 92%, δείχνοντας έτσι ότι το μοντέλο πραγματοποίησε με μεγάλη επιτυχία την ταξινόμηση στο σύνολό της.

Έτσι, στην περίπτωση αυτή προτιμάται το CNN με τον αυτόματο κωδικοποιητή να ακολουθεί, αν και σημειώνεται ότι ο έλεγχος του CNN πραγματοποιήθηκε με λιγότερες μη φυσιολογικές εικόνες, αφού μέρος αυτών χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου.

γ) Δεδομένα με θόρυβο

Noisy Data				
	Accuracy	Precision	Recall	F1 score
Autoencoder	0,7980	0,6701	0,9201	0,7755
Kmeans	0,6849	0,5461	1,0000	0,7064
Mixture of Gaussians	0,5043	0,4270	0,8989	0,5790
Supervised Neural Network	0,6358	0,5724	0,6650	0,6152

Πίνακας 29: Συγκεντρωτικός πίνακας αποτελεσμάτων για δεδομένα με θόρυβο



Εικόνα 93: Διαγραμματική παρουσίαση αποτελεσμάτων για δεδομένα με θόρυβο

Όπως φαίνεται η προσθήκη θορύβου στις εικόνες επηρέασε αισθητά την απόδοση και των τεσσάρων αλγορίθμων. Το GMM παρουσίασε την χειρότερη απόδοση με το συνολικό accuracy να καταλήγει κοντά στο 50%. Το recall, το οποίο είχε την τιμή 100% στις προηγούμενες περιπτώσεις, κατέληξε κοντά στο 90% δείχνοντας ότι οι φυσιολογικές εικόνες με την προσθήκη θορύβου δεν ομαδοποιήθηκαν εξίσου ξεκάθαρα. Παράλληλα, η τιμή του precision πλησιάζει το 43%, δείχνοντας έτσι ότι ο θόρυβος αποπροσανατόλισε πλήρως το μοντέλο και το μεγαλύτερο μέρος των μη φυσιολογικών εικόνων θεωρήθηκαν φυσιολογικές. Το γεγονός αυτό καθιστά το GMM ακατάλληλο για αναγνώριση ρωγμών με τα συγκεκριμένα δεδομένα.

Το K-means σημείωσε συνολικά καλύτερη απόδοση από το GMM, αν και επηρεάστηκε από την προσθήκη θορύβου, με την τιμή του accuracy να βρίσκεται κοντά στο 68% και την τιμή του F1 score κοντά στο 71%. Ακόμα και στην περίπτωση αυτή, ο αλγόριθμος κατάφερε να ομαδοποιήσει μαζί όλες τις φυσιολογικές εικόνες σημειώνοντας recall ίσο με 100%. Από την άλλη πλευρά, το precision έλαβε τιμή κοντά στο 55%, δείχνοντας με αυτό τον τρόπο ότι μόνο οι μισές περίπου μη φυσιολογικές εικόνες ομαδοποιήθηκαν ξεχωριστά από τις φυσιολογικές, καθιστώντας έτσι και τον αλγόριθμο K-means ακατάλληλο.

Το CNN, σε αντίθεση με τις προηγούμενες περιπτώσεις, σημείωσε χειρότερη απόδοση από το K-means με το accuracy να λαμβάνει τιμή κοντά στο 64% και το F1 score κοντά στο 62%. Παράλληλα, η τιμή 67% στο recall και 57% στο precision δείχνει την ύπαρξη πολλών ψευδώς θετικών και ψευδώς αρνητικών στοιχείων, με αποτέλεσμα να θεωρείται ότι το μοντέλο συνάντησε δυσκολία στην ορθή ταξινόμηση όλων των εικόνων. Συνεπώς, το CNN δεν είναι ο προτιμώμενος αλγόριθμος στη συγκεκριμένη περίπτωση.

Ο αυτόματος κωδικοποιητής, όπως και οι υπόλοιποι αλγόριθμοι, επηρεάστηκε σημαντικά από την ύπαρξη θορύβου. Παρ' όλα αυτά φαίνεται να σημειώνει την καλύτερη απόδοση με την τιμή του accuracy κοντά στο 80% και την τιμή του F1 score κοντά στο 78%. Οι τιμές αυτές είναι αισθητά μειωμένες σε σχέση με τις περιπτώσεις των κανονικών δεδομένων και των δεδομένων με εξομάλυνση, όμως ξεπερνούν με διαφορά τις αντίστοιχες τιμές που παρατηρήθηκαν για τους υπόλοιπους αλγορίθμους με την προσθήκη θορύβου. Παράλληλα, ακόμα και με την ύπαρξη θορύβου, ο αυτόματος κωδικοποιητής φαίνεται να προσδιόρισε με μεγάλη επιτυχία τις φυσιολογικές εικόνες, αφού η τιμή του

recall κατέληξε κοντά στο 92%. Στις μη φυσιολογικές εικόνες όμως δεν παρατηρήθηκε όμοιος βαθμός επιτυχίας αφού το precision έλαβε την τιμή 67%. Οι τιμές αυτές, αν και δείχνουν ότι η ταξινόμηση δεν έγινε με τον βέλτιστο δυνατό τρόπο, ξεπερνούν τις αντίστοιχες τιμές που σημειώθηκαν με εφαρμογή των υπόλοιπων αλγορίθμων και έτσι ο αυτόματος κωδικοποιητής προτιμάται στην περίπτωση ύπαρξης θορύβου.

6.2 Τελικά συμπεράσματα

Λαμβάνοντας υπόψιν όλα τα παραπάνω, οι αλγόριθμοι που σημειώνουν την καλύτερη απόδοση είναι ο αυτόματος κωδικοποιητής και το CNN, δηλαδή τα δύο νευρωνικά δίκτυα. Συγκρίνοντας τις τιμές των μέτρων απόδοσης accuracy, precision, recall και F1 score στις περιπτώσεις των κανονικών δεδομένων, των δεδομένων με εξομάλυνση και των δεδομένων με θόρυβο φαίνεται ότι τελικά ο βέλτιστος αλγόριθμος για τη συγκεκριμένη εφαρμογή είναι ο αυτόματος κωδικοποιητής, αφού παρουσίασε καλή απόδοση και με τα τρία είδη δεδομένων.

Αντίθετα, το CNN παρουσίασε πολύ καλή απόδοση στην περίπτωση των κανονικών δεδομένων και των δεδομένων με εξομάλυνση αλλά στην περίπτωση του θορύβου τα μέτρα απόδοσης σημείωσαν κατακόρυφη πτώση, δείχνοντας έτσι ότι η απόδοση του αλγορίθμου είναι ασταθής. Επίσης, όπως αναφέρθηκε και παραπάνω, ο έλεγχος του CNN πραγματοποιήθηκε με λιγότερα δεδομένα ελέγχου σε σχέση με τους υπόλοιπους αλγορίθμους και έτσι τα αποτελέσματα του αυτόματου κωδικοποιητή θεωρούνται πιο αξιόπιστα.

Συνεπώς, από τους αλγορίθμους που δοκιμάστηκαν στα πλαίσια της παρούσας εργασίας, ο αυτόματος κωδικοποιητής κρίνεται προτιμώμενος για την αναγνώριση ρωγμών.

Βιβλιογραφία

Ξενόγλωσση

[1] Alpaydin, Ethem. *Introduction to machine learning*. MIT press, 2020.

[2] Atienza, Rowel. *Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more*. Packt Publishing Ltd, 2018.

[3] Bhardwaj, Anurag, Wei Di, and Jianing Wei. *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018.

[4] Webb, Geoffrey I., Eamonn Keogh, and Risto Miikkulainen. "Naïve Bayes." *Encyclopedia of machine learning* 15 (2010): 713-714.

[5] Noble, William S. "What is a support vector machine?." *Nature biotechnology* 24.12 (2006): 1565-1567.

[6] Banerjee, Arindam, et al. "Model-based overlapping clustering." *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005.

[7] Kaplanis, Christos, Murray Shanahan, and Claudia Clopath. "Continual reinforcement learning with complex synapses." *International Conference on Machine Learning*. PMLR, 2018.

[8] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *science* 313.5786 (2006): 504-507.

[10] Chen, Jinghui, et al. "Outlier detection with autoencoder ensembles." *Proceedings of the 2017 SIAM international conference on data mining*. Society for Industrial and Applied Mathematics, 2017.

[11] Hawkins, Simon, et al. "Outlier detection using replicator neural networks." *International Conference on Data Warehousing and Knowledge Discovery*. Springer, Berlin, Heidelberg, 2002.

[12] Nikolaos Bakalos, Nikolaos Doulamis, Anastasios Doulamis, Konstantinos Makantasis, “Muli-property tensor-based learning for abnormal event detection”, 2021, IEEE ICIP 2021

[13] Zhao, Yiru, et al. "Spatio-temporal autoencoder for video anomaly detection." *Proceedings of the 25th ACM international conference on Multimedia*. 2017.

[14] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

[15] Lawrence Carin, David Carlson, Timothy Dunn, Kevin Liang, “Introduction to Machine Learning”, Duke University, Coursera, https://www.coursera.org/learn/machine-learning-duke

[16] Balaji, E., et al. "Automatic and non-invasive Parkinson’s disease diagnosis and severity rating using LSTM network." *Applied Soft Computing* 108 (2021): 107463.

[17] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[18] Reynolds, Douglas A. "Gaussian mixture models." *Encyclopedia of biometrics* 741 (2009): 659-663.

[19] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.

[20] Voulodimos, Athanasios, et al. "Deep learning for computer vision: A brief review." *Computational intelligence and neuroscience* 2018 (2018).

[21] Makantasis, Konstantinos, et al. "Deep convolutional neural networks for efficient vision based tunnel inspection." *2015 IEEE international conference on intelligent computer communication and processing (ICCP)*. IEEE, 2015.

[22] Makantasis, Konstantinos, et al. "Tensor-based classification models for hyperspectral data analysis." *IEEE Transactions on Geoscience and Remote Sensing* 56.12 (2018): 6884-6898.

[23] Protopapadakis, Eftychios, et al. "Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing." *Applied intelligence* 49.7 (2019): 2793-2806.

[24] Protopapadakis, Eftychios, et al. "Stacked autoencoders for outlier detection in over-the-horizon radar signals." *Computational intelligence and neuroscience* 2017 (2017).

[25] Maltezos, Evangelos, et al. "Deep convolutional neural networks for building extraction from orthoimages and dense image matching point clouds." *Journal of Applied Remote Sensing* 11.4 (2017): 042620.

[26] Protopapadakis, Eftychios, et al. "Crack identification via user feedback, convolutional neural networks and laser scanners for tunnel infrastructures." *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Vol. 5. SCITEPRESS, 2016.

Ιστοσελίδες

[27] Hosch, William L.. "Machine learning". Encyclopedia Britannica, 2 Jun. 2020, https://www.britannica.com/technology/machine-learning. Accessed 17 June 2021.

[28] https://www.ibm.com/cloud/learn/supervised-learning

[29] https://builtin.com/data-science/random-forest-algorithm

[30] https://www.ibm.com/cloud/learn/unsupervised-learning

[31] https://www.geeksforgeeks.org/apriori-algorithm/

[32] https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

[33] https://developer.ibm.com/technologies/artificial-intelligence/articles/an-introduction-to-deep-learning/

[34] https://www.yash.com/blog/an-introduction-to-reinforcement-learning-part-1/

[35] https://www.unite.ai/what-is-deep-reinforcement-learning/

[36] https://towardsdatascience.com/anomaly-detection-with-autoencoder-b4cdce4866a6

[37] https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/

[38] https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

[39] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose

[40] https://www.journaldev.com/49083/tanh-activation-function

[41] https://www.ibm.com/cloud/learn/convolutional-neural-networks

[42] https://www.ibm.com/cloud/learn/recurrent-neural-networks?mhsrc=ibmsearch_a&mhq=lstm

[43] https://learnpython.com/blog/python-array-vs-list/

[44] https://www.tensorflow.org/api_docs/python/tf/keras/Model

[45] https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

[46] https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[47] https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[48] https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

[49] https://scikit-learn.org/stable/modules/mixture.html

[50] https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

[51] https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[52] https://keras.io/api/layers/pooling_layers/max_pooling2d/