ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη συστήματος συλλογής – οπτικοποίησης δεδομένων από πολλαπλές πηγές για λήψη αποφάσεων και προγνωστική συντήρηση συσκευών υποστήριξης ατόμων με αναπηρία

ΠΜΣ «Τεχνολογίες και Υπηρεσίες Ευφυών Συστημάτων Πληροφορικής και Επικοινωνιών»
Διπλωματική Εργασία

του

ΝΑΝΟΥ ΜΙΧΑΗΛ

Επιβλέπων: Νικόλαος Βώρος

Πάτρα, Ακαδημαϊκό έτος 2022-2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη συστήματος συλλογής – οπτικοποίησης δεδομένων από πολλαπλές πηγές για λήψη αποφάσεων και προγνωστική συντήρηση συσκευών υποστήριξης ατόμων με αναπηρία

ΠΜΣ «Τεχνολογίες και Υπηρεσίες Ευφυών Συστημάτων Πληροφορικής και Επικοινωνιών»
Διπλωματική Εργασία

του

ΝΑΝΟΥ ΜΙΧΑΗΛ

Επιβλέπων: Νικόλαος Βώρος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Ημερομηνία Εξέτασης.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
……	……	……
Νικόλαος Βώρος	Νικόλαος Πετρέλλης	Χρήστος Αντωνόπουλος
Καθηγητής	Αν.Καθηγητής, ΠΑΠΕΛ	Επ.Καθηγητής, ΠΑΠΕΛ

Πάτρα, Ακαδημαϊκό έτος 2022-2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Υπεύθυνη Δήλωση
Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος μεταπτυχιακών σπουδών του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών (ΗΜΜΥ) του Πανεπιστημίου Πελοποννήσου.

(Υπογραφή)

……..
Μιχαήλ Νάνος

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη των δεδομένων λειτουργίας ενός μηχανολογικού συστήματος για την εκπαίδευση ενός μοντέλου μηχανικής μάθησης, το οποίο θα χρησιμοποιηθεί σε ένα προτεινόμενο σύστημα τεχνητής νοημοσύνης που θα χρησιμοποιηθεί για την προγνωστική συντήρηση του συστήματος. Με το προτεινόμενο σύστημα τεχνητής νοημοσύνης για την προγνωστική συντήρηση, θα μπορεί ο χρήστης του να ελέγχει την ορθή λειτουργία ενός μηχανολογικού συστήματος.

Τα δεδομένα λειτουργίας, θα συλλέγονται από αισθητήρες για παράδειγμα ήχου σε μια βάση δεδομένων και ακολούθως θα κατηγοριοποιούνται και να εξάγονται προγνωστικά μοντέλα, μέσω μηχανικής μάθησης.

Στόχοι:

- Καταγραφή σημάτων λειτουργίας π.χ. ήχου σε βάση δεδομένων.
- Κατηγοριοποίηση σημάτων λειτουργίας ήχου σε διαφορετικές κλάσεις (δημιουργία συνόλου δεδομένων) εκπαίδευση, αξιολόγηση και εξαγωγή μοντέλου Μηχανικής Μάθησης.
- Αναγνώριση ενός σήματος λειτουργίας ήχου από το εκπαιδευμένο μοντέλο ΜΜ μεμονωμένα ή σε πραγματικό χρόνο και κατάταξή του σε μια από τις υπάρχουσες κλάσεις.
- Πρόταση για την δημιουργία συστήματος Τεχνητής Νοημοσύνης που να ενσωματώνει τα παραπάνω με μια εφαρμογή επαυξημένης πραγματικότητας για την απεικόνιση αυτών σε πραγματικό χρόνο.

Θεωρητική μελέτη, ανάπτυξη και ανάλυση αλγορίθμων ΜΜ σε γλώσσες Python, SQL, Τεχνολογίες Ultra Sound, Unity, OpenCV.

Χρήση αισθητήρων όπως: Ακουστικών, οπτικών, υπερήχων για συλλογή - κατηγοριοποίηση δεδομένων μέσω Μηχανικής Μάθησης.

Υλοποίηση σε γλώσσα python της επεξεργασίας των δεδομένων ήχου των αισθητήρων για την δημιουργία dataset για την εκπαίδευση μοντέλου ΜΜ, που θα χρησιμοποιηθεί για την ανίχνευση της προγνωστικής συντήρησης με χρήση τεχνητής νοημοσύνης ως εξής:

- Με είσοδο ήχου ή εικόνας τα δεδομένα από τους αισθητήρες δημιουργείτε ένα φασματογραφικό σύνολο δεδομένων που θα χρησιμοποιηθεί για εκπαίδευση, έλεγχο, δοκιμή και αξιολόγηση του μοντέλου μηχανικής μάθησης.
- Από το παραπάνω dataset εκπαιδεύεται το μοντέλο μηχανικής μάθησης με χρήση Συνελικτικών Νευρωνικών Δικτύων με PyTorch, Time-Frequency Analysis-Synthesis Toolbox
- Δοκιμάζεται το εκπαιδευμένο μοντέλο αν μπορεί να αναγνωρίσει καταστάσεις λειτουργίας (διαφορετικές κλάσεις) του μηχανολογικού εξοπλισμού. Ώστε με αυτό τον τρόπο να

λάβουμε μια απόφαση για προγνωστική συντήρηση για παράδειγμα δημιουργία "συναγερμού" συντήρησης.

Abstract

The subject of this master thesis is the study of operating data a machine learning system for training a machine learning model, which will be used in an artificial intelligence system for forecasting system maintenance. The artificial intelligence system proposed for predictive maintenance enables the user to check the proper functioning of a mechanical system. The data will be gathered from sensors such as audio in a database, then to categorize and extract predictive models, through machine learning.

Python implementation of sensor audio data processing to create a dataset for ML model training, which will be used for detection of predictive maintenance using artificial intelligence as follows:

- By inputing audio or video data from the sensors you create a spectrograph dataset to be used to train/validate/test the model machine learning.
- From the above dataset, the machine learning model is trained using Convolutional Neural Networks with PyTorch, Time-Frequency Analysis-Synthesis Toolbox
- The trained machine learning model is tested if it can recognize operating states (different classes) of mechanical equipment. So with this this way to make a decision about predictive maintenance.

στους Γκέλυ, Μαρία, Νίκο, Βαγγέλη

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Νικόλαο Βώρο για την επίβλεψη αυτής της διπλωματικής εργασίας και τον διδακτορικό φοιτητή του κ. Αλέξανδρο Σπουρνιά ο οποίος με βοήθησε με την υλοποίηση των πειραμάτων που εκτελέστηκαν καθώς μου παρείχε τους απαραίτητους υπολογιστικούς πόρους . Τέλος θα ήθελα να ευχαριστήσω φίλους μου και ιδιαίτερα την οικογένειά μου για την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος σχημάτων

Συντομογραφίες - Αρκτικόλεξα - - Ακρωνύμια

FFT	Fast Fourier Transform
BW	Bandwidth
CNN	Convolutional Neural Network
FE	Front End
NN	Neural Network
ΣΝΔ	Συνελικτικό Νευρωνικό Δίκτυο
Leaky ReL	Διαρρέουσα Διορθωμένη Γραμμική Μονάδα

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
Frame operator	Τελεστής Frame
Constant-Q Transform	Μετασχηματισμός με σταθερό Q
Frame	Πλαίσιο
smooth	Ομαλό
Conditioning	προετοιμάζω
transient	Μεταβατικός
painless case	ανώδυνη περίπτωση
multiplication operator	πολλαπλασιαστικός τελεστής
translation operator	τελεστής ολίσθησης
interferences	παρεμβολές
CrossEntropyLoss	Απώλεια Διασταυρωμένης Εντροπίας
Mini Branch	Μικρή Πατρίδα
LeakyReLU	Διαρρέουσας Διορθωμένης Γραμμικής Μονάδας

Κεφάλαιο 1

Εισαγωγή

Είναι γεγονός ότι το Διαδίκτυο των Πραγμάτων (IoT) έχει αναπτυχθεί τα τελευταία χρόνια, και επιτρέπει στις μηχανές να επικοινωνούν μεταξύ τους μέσω του διαδικτύου σε πραγματικό χρόνο. Ένας από τους τομείς εφαρμογής του IoT στον βιομηχανικό τομέα (Industry 4.0.) δηλαδή του Industrial Internet of Things (IIoT) είναι η τεχνολογία στην οποία ο βιομηχανικός εξοπλισμός χρησιμοποιεί αισθητήρες που συλλέγουν δεδομένα και μπορούν αυτά τα δεδομένα να διοχετευτούν στο cloud για επεξεργασία και εξαγωγή συμπερασμάτων. Δηλαδή τα δεδομένα που συλλέγονται από τους αισθητήρες μπορούν να αποτελέσουν ένα σύνολο δεδομένων όπου με χρήση αλγορίθμων μηχανικής μάθησης να έχουμε την προβλέψιμη συντήρηση τους μηχανολογικού εξοπλισμού.

Η παραδοσιακή προληπτική συντήρηση του μηχανολογικού εξοπλισμού οποιουδήποτε είδους δεν μπορεί να μας εξασφαλίσει ότι δεν θα έχουμε βλάβη εκτός της προγραμματισμένης συντήρησης π.χ. λόγω ξαφνική καταπόνησης του μηχανήματος. Η ξαφνική διακοπή λειτουργίας είτε λόγω ανεπαρκής συντήρησης είτε λόγω μιας απροσδόκητης βλάβης σε κάθε περίπτωση μειώνει τη συνολική παραγωγικότητα του εξοπλισμού, λόγω του χρόνου διακοπής λειτουργίας του μηχανήματος μέχρι να επισκευαστεί.

Η προγνωστική συντήρηση είναι μια στρατηγική που χρησιμοποιείται για τη βελτιστοποίηση της συντήρησης μηχανολογικού εξοπλισμού συστημάτων προβλέποντας πότε είναι πιθανό να αποτύχουν ή να απαιτούν συντήρηση. Αυτή η προσέγγιση μπορεί να βοηθήσει στη μείωση του χρόνου διακοπής λειτουργίας και στη βελτίωση της συνολικής αποτελεσματικότητας ενός συστήματος. Ιδιαίτερα σε συστήματα όπου η περίπτωση απρόβλεπτης αποτυχίας το κόστος είναι μεγαλύτερο από το κόστος πρόβλεψης της.

Η προγνωστική συντήρηση εξαρτάται σε μεγάλο βαθμό από την ανίχνευση ανωμαλιών στην λειτουργία του μηχανολογικού εξοπλισμού μέσω δεδομένων λειτουργίας του που παίρνουμε σε πραγματικό χρόνο λειτουργίας. Θα πρέπει να μπορούμε να έχουμε την ανίχνευση ανωμαλίας ή ακραία τιμή, η ανίχνευση είναι ένα σύνολο σημαντικών τεχνικών με στόχο να να εντοπίσουν μοτίβα ανωμαλιών που αποκλίνουν από τη φυσιολογική συμπεριφορά. Οι αισθητήρες παράγουν μεγάλο όγκο δεδομένων που χρειάζεται παρακολουθούνται σε πραγματικό χρόνο για τη έγκαιρη ειδοποίηση της ανωμαλίας.

Στην παρούσα διπλωματική θα προτείνουμε μια τεχνική που βασίζεται στη μηχανική μάθηση με χρήση βαθιάς μάθησης με συνελικτικά νευρωνικά δίκτυα για την αναγνώριση διαφορετικών κλάσεων ήχων λειτουργίας ενός μηχανολογικού συστήματος. Το οποίο μπορεί να χρησιμοποιηθεί για την ανίχνευση ανωμαλιών στην λειτουργία του και να οδηγήσει στην προληπτική συντήρηση πριν την παύση λειτουργίας του.

Η υπόλοιπη διπλωματική είναι δομημένη με τον ακόλουθο τρόπο:

- Το κεφάλαιο 2 περιγράφει την εξέλιξη και την ανάγκη για Προγνωστική Συντήρηση, περιγράφει την τρέχουσα μεθοδολογία αιχμής για την ανίχνευση ανωμαλίας ως Προγνωστική Συντήρηση. Επίσης περιγράφει μερικές από τις τρέχοντες ερευνητικές εργασίες στον τομέα της ανίχνευσης ανωμαλιών λειτουργίας μηχανικών συστημάτων μέσω χρονοσειρών δεδομένων λειτουργίας τους.

- Το κεφάλαιο 3 περιγράφει την πρόταση για την δημιουργία ενός framework AI για την προγνωστική συντήρηση με χρήση του PyTorch.

- Το κεφάλαιο 4 παρουσιάζονται τα πειράματα της υλοποίησης του μοντέλου TN που προτείνεται με χρήση του PyTorch. Την εκπαίδευση μοντέλου MM με χρήση βαθείας μάθησης και συνελικτικού ΝΔ. Ώστε να γίνεται η ανίχνευση ανωμαλιών με τη δύναμη της Τεχνητής Νοημοσύνης από τα διαθέσιμα σύνολα δεδομένων που λαμβάνουμε με την χρήση πολλαπλών αισθητήρων του μηχανήματος.

- Το κεφάλαιο 5 τέλος παρουσιάζονται περιληπτικά τα συμπεράσματα σχετικά με την ανάγκη χρήσης TN για την προγνωστική συντήρηση και σημεία μελλοντικής έρευνας.

1.1 Επιβλεπόμενη μάθηση

Στην επιβλεπόμενη μάθηση τα δεδομένα και οι ετικέτες τους χρησιμοποιούνται ως υλικό εκπαίδευσης στο μοντέλο το οποίο μέσω της συνάρτησής του προσπαθεί να τα ταξινομήσει όσο το δυνατόν πιο κοντά με το σημείο αναφοράς (target). Τελικός σκοπός είναι να μπορεί να χρησιμοποιήσει αυτή τη γνώση και σε άγνωστα δεδομένα ώστε να τα αναγνωρίσει.

Υπάρχουν δύο κύριες κατηγορίες επιβλεπόμενης μάθησης:

- η ταξινόμηση (classification): στην οποία οι ετικέτες έχουν κατηγορική μορφή, δηλαδή το κάθε δεδομένο κατατάσσεται σε συγκεκριμένη κατηγορία.

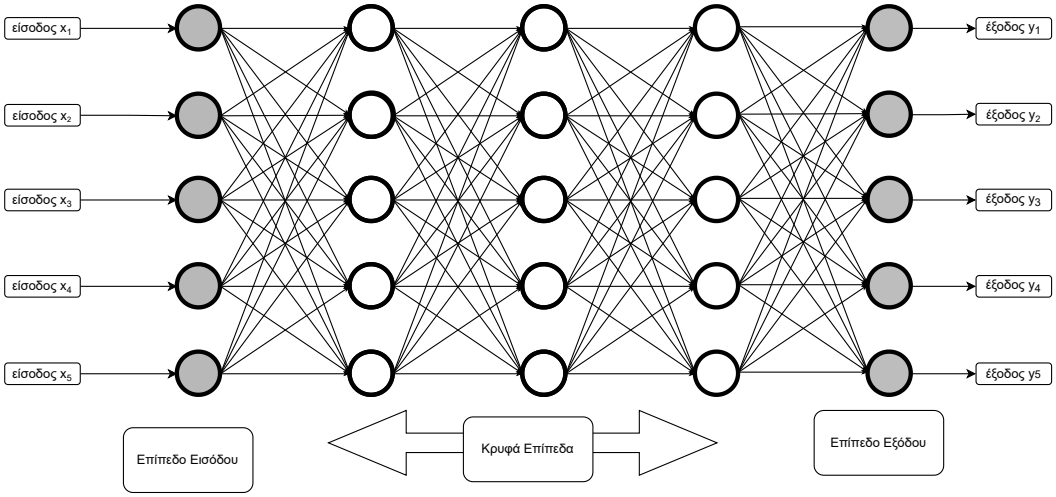
- η παλινδρόμηση (regression): όπου η έξοδος είναι μια συνεχής μεταβλητή.

1.2 Τεχνητά Νευρωνικά Δίκτυα

Τεχνητό νευρωνικό δίκτυο είναι ένας όρος που χρησιμοποιείται στην τεχνητή νοημοσύνη, προσομοιάζουν την λειτουργία του ανθρώπινου εγκεφάλου.

Στην ουσία ένα τέτοιο δίκτυο είναι ένα σύστημα επεξεργασίας δεδομένων, το οποίο αποτελείται από δομικές μονάδες, τους τεχνητούς νευρώνες. Ορίζεται σε διάφορα επίπεδα, στο επίπεδο εισόδου όπου μπαίνουν τα δεδομένα, στο επίπεδο εξόδου όπου βγαίνει το αποτέλεσμα και ενδιάμεσα τους στα κρυμμένα επίπεδα όπου γίνεται η επεξεργασία των δεδομένων.

Το βασικό χαρακτηριστικό των τεχνητών νευρωνικών δικτύων είναι ότι μπορούν να εκπαιδευτούν μέσα από μια διαδικασία μηχανικής μάθησης.

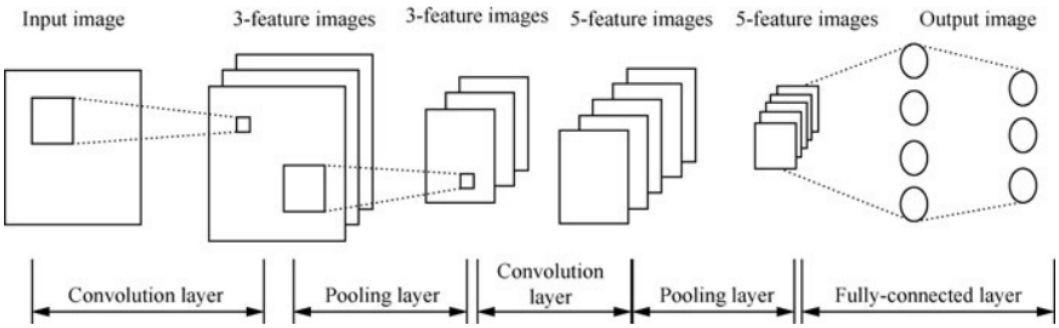


Σχήμα 1.1: Σχηματική αναπαράσταση της δομής ενός τεχνητού νευρωνικού δικτύου

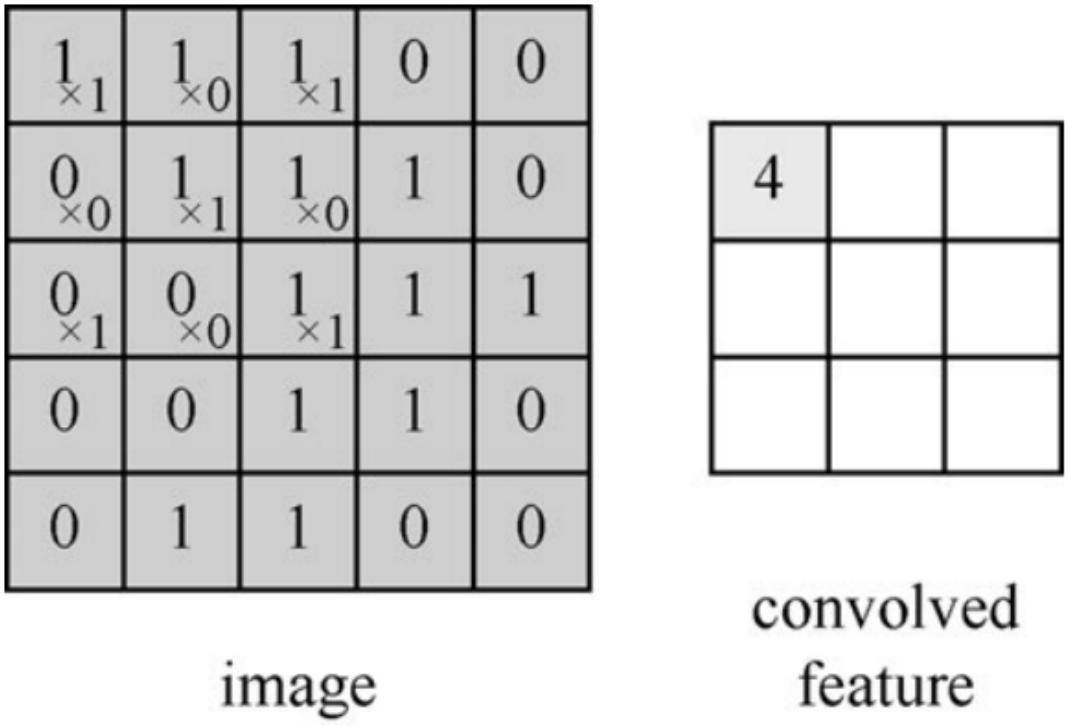
1.3 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network)

Τα Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ) είναι ένα είδος νευρωνικού δικτύου τροφοδοσίας, χρησιμοποιούν έναν αλγόριθμο βαθιάς μηχανικής μάθησης που μπορεί να λάβει μια εικόνα εισόδου, να προσθέσει βάρη και πολώσεις σε διάφορα μέρη της εικόνας ώστε να μπορεί να διαφοροποιεί το ένα από το άλλο. Η ονομασία των συνελικτικών νευρωνικών δικτύων προέρχεται από την ομώνυμη μαθηματική πράξη της συνέλιξης, σημαντικό χαρακτηριστικό τους είναι ότι η προ-επεξεργασία που απαιτείται σε ένα ΣΝΔ είναι πολύ χαμηλότερη σε σύγκριση με άλλα ΝΔ που υλοποιούν αλγορίθμους κατηγοριοποίησης.

Για αυτό τον λόγο η συγκεκριμένη κατηγορία των νευρωνικών δικτύων είναι μια από τις κύριες κατηγορίες για την αναγνώριση εικόνων, κατηγοριοποιήσεις εικόνων, αναγνώριση προσώπων και την ανίχνευση αντικειμένων.



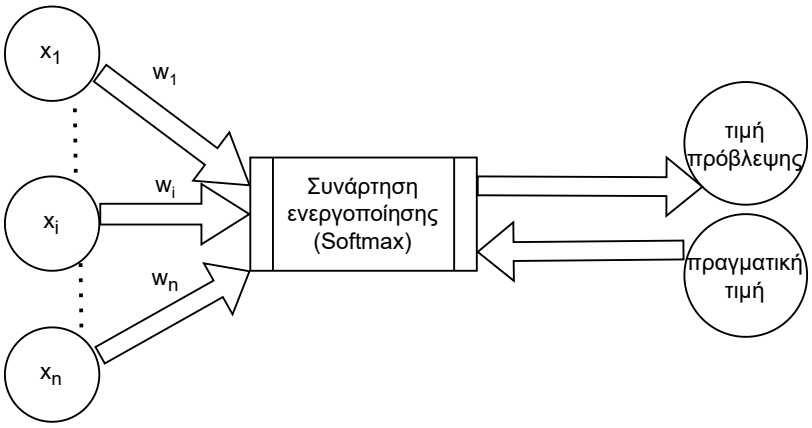
Σχήμα 1.2: Σχηματική αναπαράσταση της δομής ενός Συνελικτικού Νευρωνικού Δικτύου



Σχήμα 1.3: Παράδειγμα μιας πράξης συνέλιξης

1.4 Βαθιά Μάθηση (Deep learning) με Νευρωνικά Δίκτυα

Η βαθιά μάθηση (Deep learning) χρησιμοποιεί κάποιο αλγόριθμο οπισθοδιάδοσης (backpropagation) έτσι ώστε να αλλάζει τις εσωτερικές παραμέτρους που ονομάζονται βάρη στο νευρωνικό δίκτυο που χρησιμοποιείται για την μηχανική μάθηση. Οι ενημερώσεις των βαρών βασίζονται σε μια συνάρτηση απώλειας (ή σφάλματος) η οποία υπολογίζει τις αποκλίσεις. Χρησιμοποιώντας αυτές τις αποκλίσεις, τα βάρη κάθε νευρώνα ενημερώνονται μετά από κάθε επανάληψη, έτσι ώστε η έξοδος του ΝΔ βαθιάς μάθησης να είναι πιο κοντά στην ιδανική έξοδο ή δηλαδή στον στόχο (target).



Σχήμα 1.4: Παράδειγμα εκπαίδευσης

Με την χρήση GPU της NVIDIA γίνεται πιο γρήγορη στην εκπαίδευση ενός ΝΔ με βαθιά μάθηση από ότι αν εκτελούνταν οι πράξεις στην CPU και παρέχει μια σχετικά φθηνή εναλλακτική λύση σε σχέση με την χρήση εξειδικευμένου υλικού. Επίσης η αντικατάσταση του σφάλματος μέσου τετραγώνου με συναρτήσεις βασισμένες σε διασταυρωμένη εντροπία και αντικατάσταση της σιγμοειδούς συνάρτησης ενεργοποίησης με γενίκευση τους softamax βοηθούν στην ταχύτητα.

Παράδειγμα χρήσης της βαθίας μάθησης είναι η ανίχνευση προτύπων δηλαδή ανίχνευση χαρακτηριστικών σε εικόνες, ήχους κτλ. Το ΝΔ έχει πολλά επίπεδα με τα χαμηλότερα επίπεδα του να μαθαίνουν να αναγνωρίζουν τα βασικά χαρακτηριστικά του προτύπου που θέλουμε τα οποία στη συνέχεια τροφοδοτούνται στα υψηλότερα επίπεδα του δικτύου ώστε να γίνει η ταξινόμηση ως αποτελέσμα στην έξοδο του ΝΔ.

1.5 Συναρτήσεις Ενεργοποίησης

Οι νευρώνες ενός ΤΝΔ εκτελούν έναν γραμμικό μετασχηματισμό (y) χρησιμοποιώντας τις εισόδους x_i, τα βάρη w_i και την πόλωση (bias) b:

$$y = \sum x_i w_i + b$$

Αυτός ο γραμμικός μετασχηματισμός ονομάζεται και συνάρτηση σταθμισμένου αθροίσματος (weighted sum function). Η συνάρτηση ενεργοποίησης τοποθετείται στην έξοδο και μετασχηματίζει τον νευρώνα αποφασίζοντας αν ο νευρώνας θα ενεργοποιηθεί ή όχι κανονικοποιώντας την τιμή εξόδου μεταξύ $[0, 1]$ ανάλογα της συνάρτησης ενεργοποίησης που έχει χρησιμοποιηθεί.

Έτσι, το δίκτυο μπορεί να προσεγγίσει συναρτήσεις που δεν είναι γραμμικές δηλαδή μπορεί να προβλέψει μια κλάση που διαχωρίζεται από μη-γραμμικά όρια. Οι συναρτήσεις ενεργοποίησης είναι μονότονες για να βρούμε την ελάχιστη κλίση και διαφορίσιμες για να μπορούμε να ανανεώνουμε τα βάρη και τις πολώσεις των νευρώνων κατά την οπισθοδιάδοση του σφάλματος ώστε να το μειώσουμε.

Στην παρούσα εργασία θα χρησιμοποιήσουμε σαν συνάρτηση ενεργοποίησης την softmax, η οποία είναι μια γενικευμένη μορφή της σιγμοειδούς συνάρτησης και χρησιμοποιείται στο επίπεδο εξόδου ενός νευρωνικού δικτύου που επιλύει πρόβλημα ταξινόμησης πολλών κλάσεων. Επίσης για τα κρυφά επίπεδα του νευρωνικού δικτύου θα χρησιμοποιήσουμε την συνάρτηση

ενεργοποίησης διαρρέουσας διορθωμένης γραμμικής μονάδας Leaky ReLU η οποία έχει πάρα πολύ μικρό υπολογιστικό κόστος εξαιτίας των απλών υπολογισμών της.

1.6 Το Framework Βαθείας Μάθησης PyTorch

[2] Το PyTorch είναι ένα Framework ανάπτυξης βαθιάς μάθησης που ξεκίνησε από το Facebook. Είναι ένα επιστημονικό Framework μηχανικής μάθησης που προέκυψε από το Torch. Το Torch είναι ένα επιστημονικό υπολογιστικό Framework που υποστηρίζεται από μεγάλο αριθμό αλγορίθμων μηχανικής μάθησης και μια βιβλιοθήκη λειτουργιών τανυστών παρόμοια με το NumPy.

Το PyTorch έχει τα ακόλουθα χαρακτηριστικά.

- Το PyTorch συνδέεται καλύτερα με C++ και υποστηρίζει την πρόσβαση σε Python γρήγορα όπως χρησιμοποιείται το NumPy ή το SciPy, το οποίο όχι μόνο βοηθά τους χρήστες να κατανοήσουν την Python πολύ πιο εύκολα, αλλά εγγυάται επίσης ότι ο κώδικας είναι βασικά σε συνοχή με τον εγγενή κώδικα Python.

- Υποστηρίζει Δυναμικά Νευρωνικά Δίκτυα Πολλά mainstream πλαίσια σήμερα δεν υποστηρίζουν δυναμικά νευρωνικά δίκτυα, όπως το TensorFlow 1.x. Η εκτέλεση του TensorFlow 1.x απαιτεί την κατασκευή στατικών υπολογιστικών γραφημάτων εκ των προτέρων και στη συνέχεια την επανειλημμένη εκτέλεση τους. Αλλά η εκτέλεση του PyTorch είναι πολύ λιγότερο περίπλοκη. Τα προγράμματα PyTorch μπορούν να κατασκευάσουν και να προσαρμόσουν δυναμικά υπολογιστικά γραφήματα κατά το χρόνο εκτέλεσης.

- Εύκολος εντοπισμός σφαλμάτων Το PyTorch μπορεί να δημιουργήσει δυναμικά γραφήματα ενώ εκτελείται. Επομένως, οι προγραμματιστές μπορούν να τερματίσουν τον interpreter στη εκτέλεση εντοπισμού σφαλμάτων και να ελέγξουν την έξοδο σε έναν συγκεκριμένο κόμβο.

- Υποστήριξη cuda GPU εντολών, οι τανυστές που παρέχει η PyTorch για την υποστήριξη της CPU και της GPU μπορούν επίσης να επιταχύνουν σημαντικά τον υπολογισμό.

Κεφάλαιο 2

Ανάγκη για προγνωστικές τεχνικές ανίχνευσης για συντήρηση

Ένα από τα μεγαλύτερα προβλήματα που αντιμετωπίζει η συντήρηση των μηχανών είναι ότι οι παραδοσιακές διαδικασίες συντήρησης δεν μπορούν να αναγνωρίσουν όλες τις ανωμαλίες εκ των προτέρων. Αυτό μπορεί να σημαίνει μια αναπάντεχη αστοχία ενός εξαρτήματος την ξαφνική συνολική μη λειτουργία του μηχανήματος. Παράδειγμα: Ένα φθαρμένο ρουλεμάν που οδηγεί σε συνολική βλάβη του μηχανήματος.

Η ανίχνευση ανωμαλιών, είναι η διαδικασία αναγνώρισης μεταβλητών/στοιχείων που δεν ανήκουν σε ένα αναμενόμενο μοτίβο, το οποίο αφορά το ίδιο σύνολο δεδομένων από άλλες παρατηρήσεις λειτουργίας του ίδιου εξαρτήματος και είναι συνήθως μη παρατηρήσιμο στο ανθρώπινο μάτι. Τέτοιες ανωμαλίες που μπορούμε να τις χαρακτηρίσουμε ως πρώιμα σημάδιας που συνήθως οδηγεί σε ολική/μερική διακοπή ή σε λάθη λειτουργίας του εξοπλισμού.

Η ανίχνευση ανωμαλίας κατά την λειτουργία ενός εξαρτήματος είναι δύσκολη γιατί:

- Δεν έχουμε την γνώση για να μπορούμε να ξεχωρίσουμε την τρέχουσα λειτουργία της μηχανής σε μια κατάσταση που θα μπορούσε να φέρει μη λειτουργία.
- Δεν υπάρχουν εκ των προτέρων διαθέσιμα δείγματα στα δεδομένα λειτουργίας για περιγράψουμε την ανωμαλία με ακρίβεια.

2.1 Η εξέλιξη στην σειρά της Προγνωστικής Συντήρησης

- **Συντήρηση μετά την βλάβη**

  Σημαίνει ότι θα επισκευάσουμε την μηχανή μετά τη βλάβη που θα συμβεί.

- **Προληπτική συντήρηση**

  Περιλαμβάνει διεξαγωγή τακτικών εργασιών συντήρησης για να αποφύγουμε τις αποτυχίες. Ωστόσο, πολλές φορές γίνεται χωρίς την πραγματική ανάγκη να γίνει. Για παράδειγμα, αντικατάσταση εξαρτήματος εξοπλισμού μετά από 6 μήνες όταν θα μπορούσε να έχει τρέξει με επιτυχία για τουλάχιστον 3 μήνες ακόμη.

- **Προγνωστική συντήρηση βάσει κανόνων**

  Η συντήρηση πραγματοποιείται βάσει κωδικοποιημένων κανόνων κατωφλίου ώστε σε περίπτωση μια μέτρηση ξεπερνά τα όρια το εξάρτημα αντικαθίσταται.

- **Πρόβλεψη για συντήρηση με βάση τη Μηχανική Μάθηση**

  Χρησιμοποιεί προηγμένες τεχνικές ανάλυσης χρονοσειρών δεδομένων λειτουργίας και μηχανικής μάθησης για να προβλέψουμε πότε θα συμβεί η επόμενη αποτυχία και αναλόγως πραγματοποιούμε την συντήρηση πριν συμβεί η βλάβη.

2.2 Ανίχνευση ανωμαλιών για την πρόβλεψη και συντήρηση

Η ανίχνευση ανωμαλιών χρονοσειρών με βάση τη βαθιά μάθηση έχει μελετηθεί σε δύο κύριες κατηγορίες: [1]

1. Μεθόδους που βασίζονται στην πρόβλεψη.
2. Μεθόδους που βασίζονται σε επανασύνθεση.

2.2.1 Μέθοδοι που βασίζονται στην πρόβλεψη

Οι μέθοδοι που βασίζονται σε προβλέψεις, μαθαίνουν πρώτα από τα κανονικά δεδομένα τα χαρακτηριστικά των χρονοσειρών, προβλέποντας τις τιμές του μέλλοντος χρονικά βήματα με βάση τα δεδομένα προηγούμενων χρονοσειρών.

Αυτές οι μέθοδοι υποθέτουν ότι ένα μοντέλο πρόβλεψης δεν μπορεί να εκτιμήσει με ακρίβεια τα μη φυσιολογικά πρότυπα όταν εκπαιδεύεται μόνο σε κανονικά δεδομένα. Κατά τη διάρκεια της εξαγωγής συμπερασμάτων, προσδιορίζουν συγκεκριμένα χρονικά τμήματα ως μη φυσιολογικά εάν η διαφορά μεταξύ οι πραγματικές και οι προβλεπόμενες τιμές για το τμήμα είναι πάνω από ένα προκαθορισμένο όριο τότε προβλέπουν ανωμαλία στην λειτουργία.

2.2.2 Μέθοδοι που βασίζονται σε επανασύνθεση

Οι μέθοδοι που βασίζονται στην ανασυγκρότηση μαθαίνουν την κανονική λειτουργία κωδικοποιώντας δεδομένα κανονικών χρονοσειρών σε συμπιεσμένα διανύσματα και στη συνέχεια ανακατασκευάζοντας (αποκωδικοποιώντας) την είσοδο από τα συμπιεσμένα διανύσματα.

Αυτές οι προσεγγίσεις υποθέτουν ότι οι ανωμαλίες δεν μπορούν να ανακατασκευαστούν καλά από το μοντέλο που έχει εκπαιδευτεί μόνο με κανονικά μοτίβα, επειδή η χαρτογράφηση άγνωστων ανωμαλιών στον συμπιεσμένο χώρο θα είχε ως αποτέλεσμα μεγάλη απώλεια ανακατασκευής.

Οι περισσότερες μέθοδοι που βασίζονται στην ανακατασκευή βασίζονται σε έναν αυτόματο κωδικοποιητή, ο οποίος αποτελείται από έναν κωδικοποιητή για την καταγραφή των χρονικών χαρακτηριστικών των δεδομένων χρονοσειράς εισόδου και έναν αποκωδικοποιητή για την ανακατασκευή της εισόδου από το λανθάνον κωδικοποιημένο διάνυσμα.

Σε αυτές τις μεθόδους, τα χρονικά τμήματα ανιχνεύονται ως ανωμαλίες εάν η διαφορά μεταξύ της εισόδου και των ανακατασκευασμένων εξόδων είναι μεγάλη. Το κύριο μειονέκτημα των μεθόδων αυτών είναι ότι είναι πολύ δύσκολα στην εκπαίδευσή ενός μοντέλου ΜΜ. Ανάλογα με τον τύπο τους, άλλο μοντέλο απλώς καταρρέει κατά τη διάρκεια της εκπαίδευσης, άλλο δε συγκλίνει ποτέ ή συμβαίνουν διάφορα άλλα προβλήματα.

2.3 IIoT- Industrial Internet of Things:

Ο κύριος στόχος της προγνωστικής συντήρησης είναι ο εντοπισμός ανωμαλιών ή προτύπων αστοχίας για την παροχή έγκαιρης ειδοποίησης για την πιθανή αστοχία τους μηχανολογικού εξοπλισμού. Στο [3] προτείνεται η χρήση βαθιάς μάθησης που είναι προληπτική υπό τους όρους

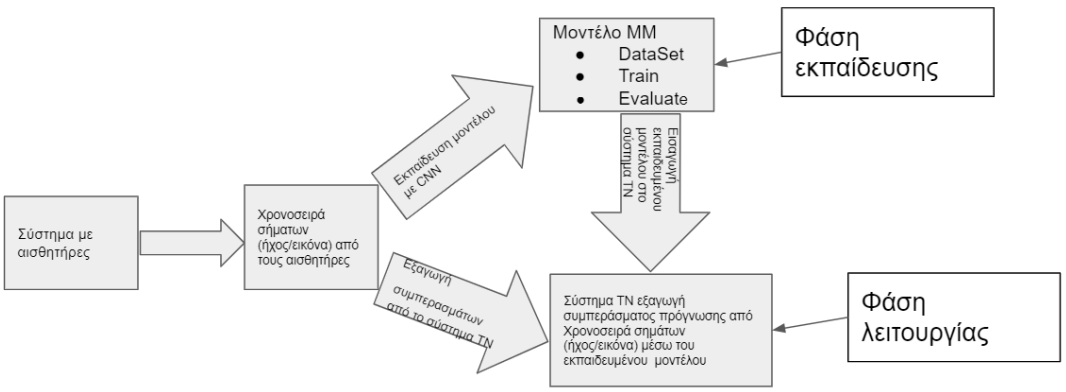
ότι θα «μάθουν» και θα μελετήσουν εκ των προτέρων μοτίβα που οδηγούν σε ανωμαλίες και ελαττώματα στα μηχανήματα ώστε να προειδοποιούν προληπτικά για πιθανή αστοχία πολύ νωρίτερα.

Αυτό θα ωφελήσει ώστε οι "προειδοποιήσεις" και οι "συναγερμοί" μπορούν να επιτρέψουν την αποτελεσματική συντήρηση των ελαττωματικών στοιχείων, επίσης οι πόροι που απαιτούνται για τη συντήρηση μπορούν να χρησιμοποιηθούν με τον βέλτιστο τρόπο.

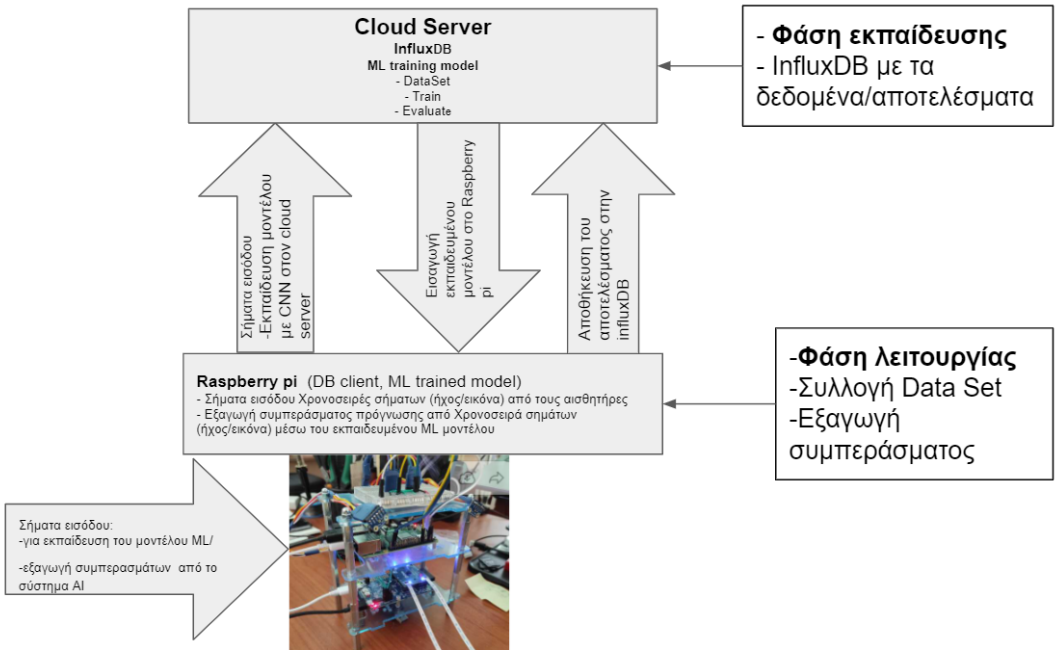
Κεφάλαιο 3

Προτεινόμενο σύστημα Τεχνητής Νοημοσύνης για την ανίχνευση βλαβών σε μηχανολογικό σύστημα

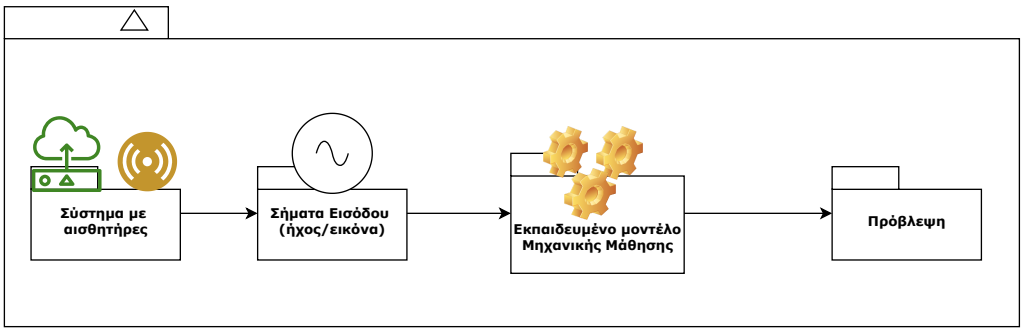
3.1 Το προτεινόμενο σύστημα Τεχνητής Νοημοσύνης



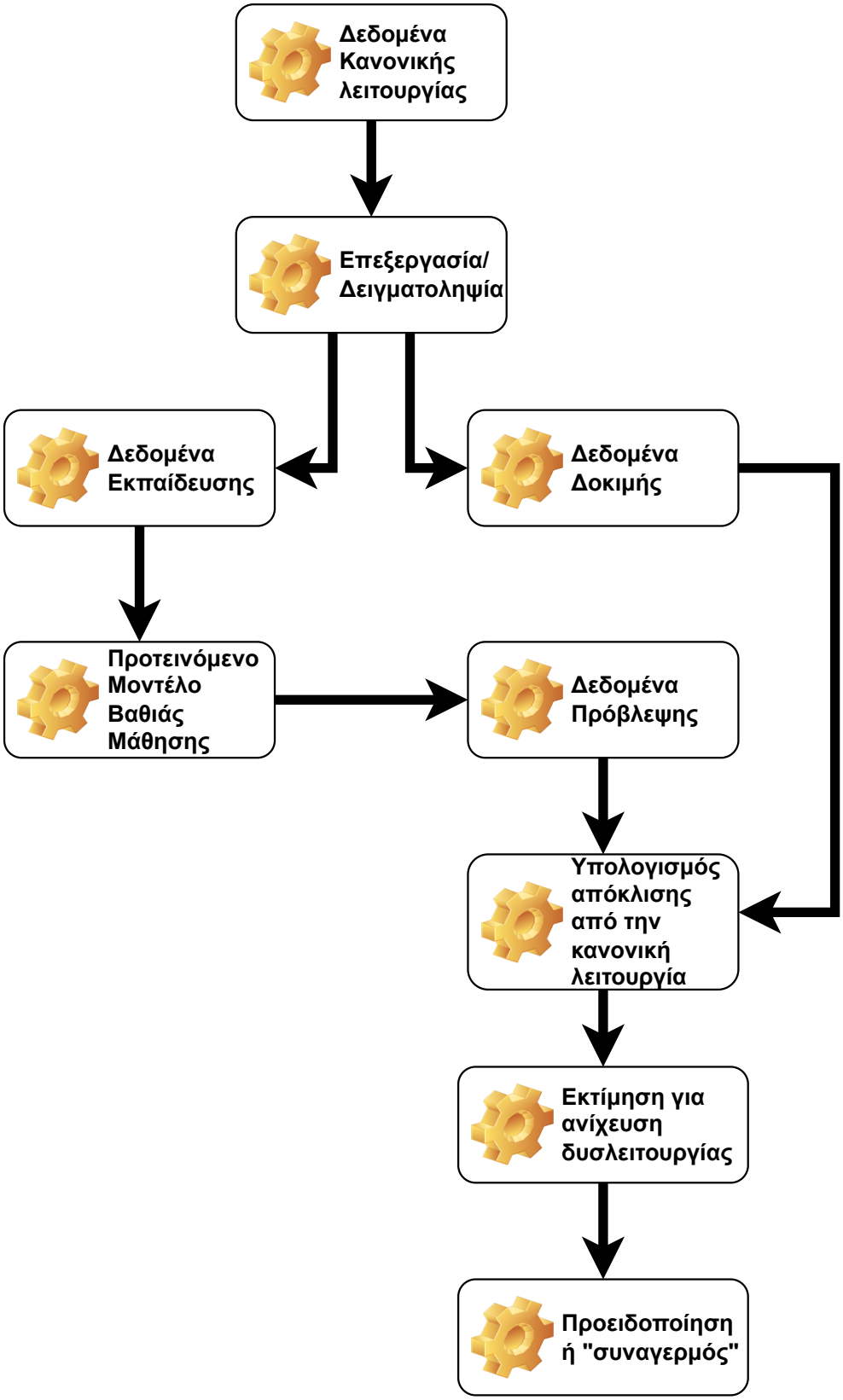
Σχήμα 3.1: Σχηματική απεικόνιση του συστήματος Τεχνητής Νοημοσύνης που προτείνεται



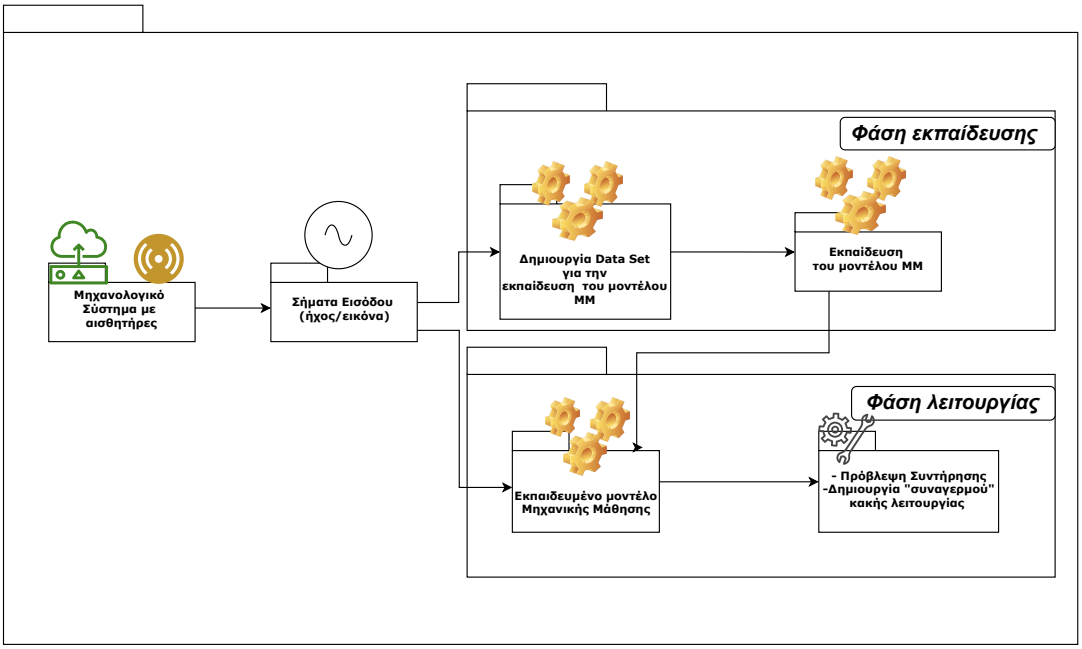
Σχήμα 3.2: Αναπαράσταση των φάσεων λειτουργίας του συστήματος Τεχνητής Νοημοσύνης που προτείνεται



Σχήμα 3.3: Σχηματική απεικόνιση της πρόβλεψης



Σχήμα 3.4: Βήματα για την δημιουργία ειδοποίησης δυσλειτουργίας μέσω του συστήματος ΤΝ με βαθεία μάθηση



Σχήμα 3.5: Σχηματικά οι δύο Φάσεις του συστήματος ΤΝ Εκπαίδευσης - Λειτουργίας

1. Τα αρχεία του δεδομένων κανονικής λειτουργίας (dataset) θα δημιουργούνται από το σύστημα με τους αισθητήρες (για παράδειγμα ένα raspberry με μικρόφωνο) και θα στέλνονται στον cloud server που έχει influDB. Στον cloud server θα υπάρχει ένα linux systemd service που θα τρέχει τα python scripts που φτιάχνουν το data set (create_-dataset.py) που χρησιμοποιούμε για να εκπαιδεύσουμε και να αποθηκεύσουμε το μοντέλο model.pth (με το script train.py).

2. Το εκπαιδευμένο μοντέλο (model.pth) θα στέλνεται πίσω στο raspberry και θα κάνει εξαγωγή συμπεράσματος, σχετικά με την προγνωστική συντήρηση της μηχανής, μέσω της ταξινόμησης μιας/περισσότερων καινούργιων εισόδων από τους αισθητήρες (με το script inference.py).

3. Το αποτέλεσμα του συμπεράσματος και της ταξινόμησης θα στέλνεται πίσω στον cloud server για αποθήκευση στην infuxDB.

4. το βήμα 1-2 μπορούν να γίνονται ξανά με την χρήση του model.chpnt που παράγεται από script train.py για κάνουμε ξανά εκπαίδευση του μοντέλου.

3.1.1 Φάση εκπαίδευσης: Εκπαίδευση του ταξινομητή σημάτων εισόδου (ήχου,εικόνων κτλ)

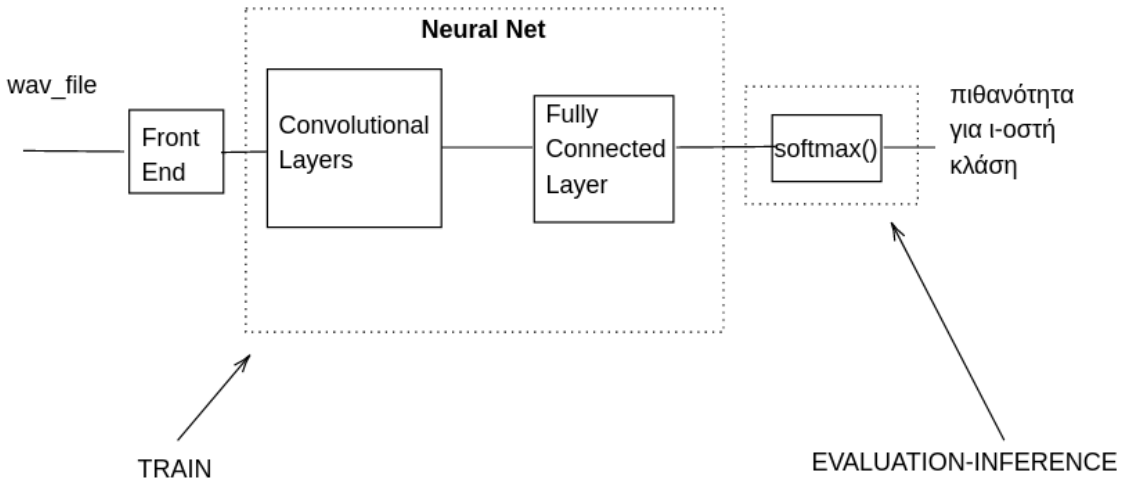
Τι κάνουμε με τα δεδομένα; Γενικά όταν έχουμε να χειριστούμε δεδομένα από εικόνες, κείμενο, ήχο ή βίντεο, μπορούμε να χρησιμοποιήσουμε τα κανονικά πακέτα της python που φορτώνουν τα δεδομένα σε έναν πίνακα numpy. Στην συνέχεια μετατρέπουμε αυτό τον πίνακα σε τανυστή torch.*Tensor.

- Για εικόνες τα πακέτα Pillow, OpenCV
- Για ήχους τα πακέτα scipy, librosa
- Για κείμενο raw Python ή Cython based loading, ή NLTK και SpaCy

Για να εκπαιδεύσουμε τον ταξινομητή Θα κάνουμε τα παρακάτω βήματα:

- Φορτώνουμε και κανονικοποιούμε τα σύνολα δεδομένων εκπαίδευσης και δοκιμής με χρήση λογισμικού που επεξεργάζεται αρχεία εικόνας και ήχου π.χ. audacity.
- Ορίζουμε ένα συνελικτικό νευρωνικό δίκτυο
- Ορίζουμε μια συνάρτηση απώλειας
- Εκπαιδεύουμε το δίκτυο στα δεδομένα εκπαίδευσης
- Δοκιμάζουμε το δίκτυο στα δεδομένα δοκιμής

3.1.2 Ορισμός/Αρχιτεκτονική του Συνελικτικού Νευρωνικού Δικτύου



Σχήμα 3.6: Σχηματική αναπαράσταση ενός επιπέδου του ΣΝΔ που προτείνεται

Το σύστημα[1] του συνελικτικού νευρωνικού δικτύου που προτείνεται φαίνεται στο σχήμα 3.6

Για να ορίσουμε το δικό μας νευρωνικό δίκτυο, θα πρέπει να κατανοήσουμε τις αναμενόμενες εισόδους και εξόδους. Σε ένα πρόβλημα δυαδικής ταξινόμησης (binary-classification problem), η έξοδος μας μπορεί να είναι ένας μεμονωμένος νευρώνας. Στη συνέχεια θα πρέπει να αποφασίσουμε για την αρχιτεκτονική που θέλουμε. Δηλαδή πόσα και τι είδουςεπίπεδα (layers) πρέπει να έχουμε και πόσους νευρώνες σε κάθε στρώμα.

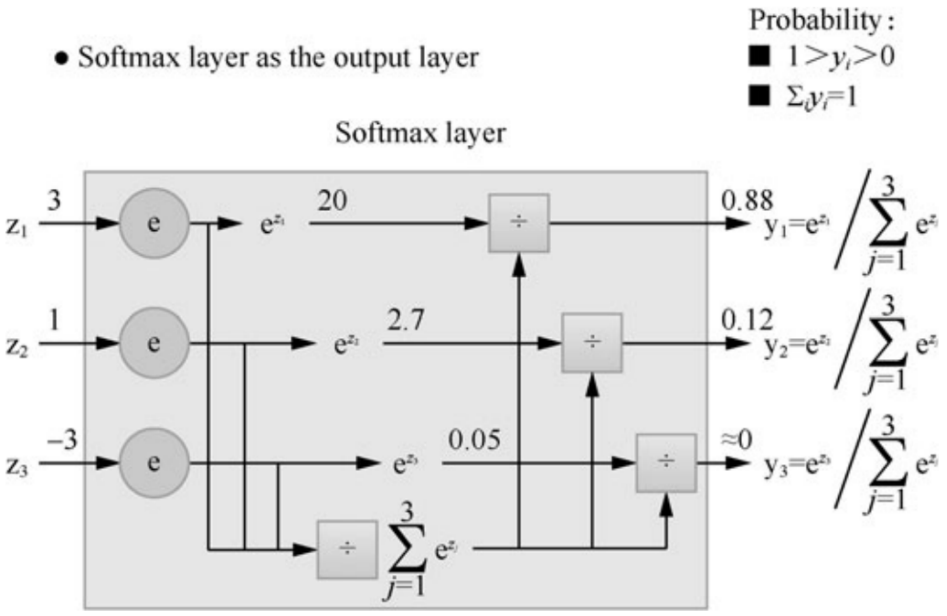
Για το πρόβλημα της προγνωστικής συντήρησης έχουμε ένα διαφορετικό πρόβλημα ταξινόμησης, δηλαδή αντί για ένα πρόβλημα δυαδικής ταξινόμησης, θα πρέπει να εργαστούμε σε ένα πρόβλημα γενικής ταξινόμησης, όπου μια τιμή εισόδου θα ταξινομηθεί σε μία από τις πολλές κατηγορίες (ή κλάσεις) [0,C) όπου C ο αριθμός των διαφορετικών κλάσεων.

[1]Για την ακρίβεια η είσοδος στο σύστημα που φαίνεται είναι ένα block του ολόκληρου wav αρχείου

Για συνάρτηση ενεργοποιήσης (activation function) θα χρησιμοποιήσουμε την softmax η οποία είναι μια γενικευμένη μορφή της σιγμοειδούς συνάρτησης (sigmoid function) και χρησιμοποιείται στο επίπεδο εξόδου ενός νευρωνικού δικτύου που επιλύει πρόβλημα ταξινόμησης πολλών κλάσεων. Οι τιμές που μπορεί να πάρει είναι μεταξύ [0, 1] με το άθροισμα των πιθανοτήτων να ισούται με 1.

Η κανονικοποιημένη εκθετική συνάρτηση (softmax) εκφράζεται από τον τύπο:

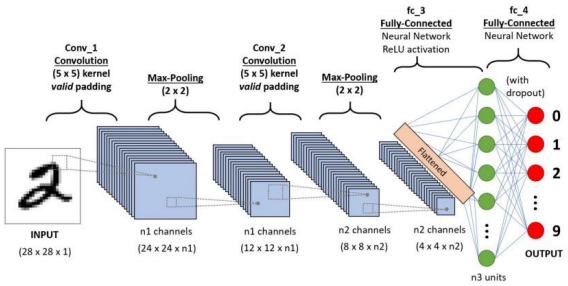
$$\sigma(\mathbf{x_j}) = \frac{\mathbf{e}^{\mathbf{x_j}}}{\sum_{\mathbf{i}} \mathbf{e}^{\mathbf{x_j}}}$$



Σχήμα 3.7: Παράδειγμα: Σχηματική αναπαράσταση του υπολογισμού της συνάρτησης softmax προέλευση εικόνας

Θα μελετήσουμε πειραματικά το πρόβλημα ταξινόμησης ήχων, εικόνας κτλ που παίρνουμε σαν είσοδο από τους αισθητήρες και το ΝΔ θα πρέπει να κατηγοριοποιεί την είσοδο σε μια από τις C κλάσεις ταξινόμησης που του έχουμε ορίσει, ώστε να εκπαιδεύει το μοντέλο. Το εκπαιδευμένο μοντέλο θα χρησιμοποιηθεί στην συνέχεια για να αναγνωρίζει την είσοδο σε ποια κατηγορία ανήκει κατά την διάρκεια λειτουργίας της μηχανής. Η διαδικασία θα επαναλαμβάνεται για περαιτέρω εκπαίδευση του μοντέλου.

Χρησιμοποιούμε Συνελικτικό νευρωνικό δίκτυο (ΣΝΔ) (Convolutional Neural Network (CNN)).



Σχήμα 3.8: Παράδειγμα Αρχιτεκτονικής CNN Νευρωνικού Δικτύου
προέλευση εικόνας

Το σχήμα 3.8 δείχνει ένα παράδειγμα ΣΝΔ που παίρνει μια εικόνα ψηφίου του 2 και μας δίνει το αποτέλεσμα του ψηφίου που εμφανίστηκε στην εικόνα ως αριθμός.

Αυτό το είδος ΝΔ χρησιμοποιείται κυρίως για την επεξεργασία εικόνας ως ένας απλός ταξινομητής όπου εκπαιδεύει ένα μοντέλο ΜΜ να ξεχωρίζει εικόνες. Για να πετύχουν την ταξινόμηση αυτού του είδους τα ΝΔ έχουν την προσθήκη περισσότερων επιπέδων στο δίκτυο και τη χρήση της ανάστροφης διάδοσης για την εκμάθηση των βαρών. Αλλά αυτό δημιουργεί ένα πρόβλημα ο αριθμός των βαρών αυξάνεται υπερβολικά και, κατά συνέπεια, ο όγκος των δεδομένων εκπαίδευσης που απαιτείται για την επίτευξη ικανοποιητικής ακρίβειας μπορεί να καταστεί υπερβολικά μεγάλος και, ως εκ τούτου, μη ρεαλιστικός. Η λύση στο πρόβλημα του υπερβολικά μεγάλου αριθμού βαρών είναι η εισαγωγή ενός ειδικού είδους επιπέδου που μπορεί να συμπεριληφθεί στο νευρωνικό δίκτυο.

Το ειδικό αυτό είδος επιπέδου ονομάζεται συνελικτικό επίπεδο. Τα δίκτυα που περιλαμβάνουν συνελικτικά επίπεδα ονομάζονται συνελικτικά νευρωνικά δίκτυα (ΣΝΔ). Η βασική ιδιότητά τους είναι ότι μπορούν να εντοπίζουν χαρακτηριστικά εικόνων όπως φωτεινά ή σκοτεινά σημεία (ή συγκεκριμένο χρώμα), τις ακμές σε διάφορους προσανατολισμούς, πρότυπα, και ούτω καθεξής. Τα ΣΝΔ μπορούν να αναγνωρίζουν αντικείμενα σε μια εικόνα οπουδήποτε, ανεξάρτητα από το πού έχει παρατηρηθεί στις εικόνες εκπαίδευσης.

Αυτό μπορεί να εφαρμοστεί και σε ήχους αν κάνουμε μετασχηματισμό του ήχου στο φασματόγραμμα ώστε να έχουμε μια φασματογραφική εικόνα του ήχου. Με την χρήση του φασματογραφήματος του ήχου έχουμε έναν τρόπο οπτικής απεικόνισης ήχων που παρέχει πληροφορίες σχετικά με τις συχνότητες που συνθέτουν τον ήχο. Για παράδειγμα ο κατακόρυφος άξονας εμφανίζει τη συχνότητα όσο πιο έντονο είναι το χρώμα τόσο πιο μεγάλη η ένταση του ήχου.

3.1.3 Ορισμός της συνάρτησης σφάλματος διασταυρωμένης εντροπίας

Η συνάρτηση σφάλματος ή απώλειας (loss function) είναι ένας τρόπος μέτρησης που υπολογίζει το σφάλμα δηλαδή υπολογίζει το πόσο κοντά ή όχι είναι η πρόβλεψη (prediction) που έχει κάνει το ΝΔ κατά την εκπαίδευσή του ΜΜ σε σχέση με την πραγματικότητα. Η επιλογή της κατάλληλης συνάρτησης κόστους είναι σημαντικός παράγοντας που επηρεάζει την ακρίβεια του ΝΔ. Η τιμή του σφάλματος θα χρησιμοποιηθεί στη συνέχεια από κάποια συνάρτηση βελτιστοποίησης η οποία μέσω ρυθμίσεως κάποιων παραμέτρων της (optimization) θα οδηγήσει στη βελτίωση της ακρίβειας του μοντέλου.

Οι συναρτήσεις σφάλματος μπορεί να είναι είτε παλιδρόμισης π.χ. Συνάρτηση μέσου τετραγωνικού σφάλματος είτε ταξινόμησης π.χ. Συνάρτηση σφάλματος πολλών κλάσεων διασταυρωμένης εντροπίας (Categorical Cross Entropy Loss).

Θα χρησιμοποιήσουμε συνάρτηση σφάλματος (ή απώλειας) διασταυρωμένης εντροπίας (Cross Entropy Loss) από PyTorch που είναι η εξής:

Listing 3.1: Python Cross Entropy Loss

```
CLASS torch.nn.CrossEntropyLoss(weight=None, size_average=None,
ignore_index=-100, reduce=None, reduction='mean',
label_smoothing=0.0)
```

Παράμετροι:

- **weight (Tensor, optional)** – a manual rescaling weight given to each class. If given, has to be a Tensor of size C
- **size_average (bool, optional)** – Deprecated (see reduction). By default, the losses are averaged over each loss element in the batch. Note that for some losses, there are multiple elements per sample. If the field size_average is set to False, the losses are instead summed for each minibatch. Ignored when reduce is False. Default: True
- **ignore_index (int, optional)** – Specifies a target value that is ignored and does not contribute to the input gradient. When size_average is True, the loss is averaged over non-ignored targets. Note that ignore_index is only applicable when the target contains class indices.
- **reduce (bool, optional)** – Deprecated (see reduction). By default, the losses are averaged or summed over observations for each minibatch depending on size_average. When reduce is False, returns a loss per batch element instead and ignores size_average. Default: True
- **reduction (str, optional)** – Specifies the reduction to apply to the output: 'none' | 'mean' | 'sum'. 'none': no reduction will be applied, 'mean': the weighted mean of the output is taken, 'sum': the output will be summed. Note: size_average and reduce are in the process of being deprecated, and in the meantime, specifying either of those two args will override reduction. Default: 'mean'
- **label_smoothing (float, optional)** – A float in [0.0, 1.0]. Specifies the amount of smoothing when computing the loss, where 0.0 means no smoothing. The targets become a mixture of the original ground truth and a uniform distribution as described in Rethinking the Inception Architecture for Computer Vision. Default: 0.00.0.

Η συνάρτηση σφάλματος της Cross-Entropy είναι πολύ διαδεδομένη σε προβλήματα ταξινόμησης (classification) καθώς ποσοτικοποιεί τη διαφορά μεταξύ των εισόδων (logits) και του στόχου (target). Είναι χρήσιμο όταν εκπαιδεύουμε ένα μοντέλο ΜΜ σε ένα πρόβλημα ταξινόμησης με C κλάσεις να έχουμε ένα προαιρετικό βάρος ορίσματος μέσω ενός τανυστή (tensor) 1D που εκχωρεί βάρος σε κάθε μία από τις κλάσεις.

Αυτό είναι ιδιαίτερα χρήσιμο όταν δεν έχουμε ένα ισορροπημένο σύνολο δεδομένων εκπαίδευσης όπως συμβαίνει στην συνήθως με πραγματικά σύνολα δεδομένων.

Οι είσοδοι (input logits) αναμένεται να είναι μη κανονικοποιημένες για κάθε κλάση ούτε να είναι θετικές ή το άθροισμα τους να είναι 1. Η είσοδος πρέπει να είναι ένας τανυστής (Tensor) μεγέθους C για είσοδο όχι χωρισμένο σε παρτίδες (unbatched) ή χωρισμένο σε παρτίδες (minbatch, C), (minbatch, $C, d_1, d_2, ..., d_K$) με K≥1 για την περίπτωση K-διάστασης. Η τελευταία περίπτωση είναι χρήσιμη για εισόδους υψηλότερων διαστάσεων, όπως υπολογισμό απώλειας διασταυρούμενης εντροπίας ανά pixel για εικόνες 2D.

Ο στόχος που αναμένει αυτό το κριτήριο θα πρέπει να περιέχει :

- Τον δείκτη των κλάσεων στην περιοχή $[0, C)$ όπου C είναι ο αριθμός των κλάσεων. εάν οριστεί $ignore\_index$, αυτή η απώλεια δέχεται επίσης αυτόν τον επιπλέον δείκτη κλάσης που αυτός ο δείκτης δεν είναι απαραίτητα στην περιοχή των κλάσεων.

– Αν το σφάλμα δεν μειώνεται δηλαδή με τη μείωση σε "none" το μη μειωμένο σφάλφα για αυτήν την περίπτωση μπορεί να περιγραφεί ως:

$$l(x,y) = L = \{l_1, ..., l_N\}^T, \quad \text{με}$$
$$l_n = -w_{y_n} log \frac{exp(x_n, y_n)}{\sum_{c=1}^{C} exp(x_n, c)} \cdot 1\{y_n \neq ignore\_index\}$$

όπου x είναι η είσοδος, y η έξοδος (στόχος/target), w είναι το βάρος (weight), C είναι ο αριθμός των κλάσεων, N καλύπτει την διάσταση της παρτίδας (minibatch) 1 ή K-διάστασεων d_1, d_2, ..., d_K.

– Αν το σφάλμα μειώνεται τότε το μειωμένο σφάλφα για αυτήν την περίπτωση μπορεί να περιγραφεί ως:

$$l(x,y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n} \cdot 1\{y_n \neq ignore\_index\}} & , if reduction =' mean' \\ \sum_{n=1}^{N} l_n & , if reduction =' sum' \end{cases}$$

- Η πιθανότητα για κάθε κλάση είναι χρήσιμη όταν απαιτούνται ετικέτες πέραν μιας κατηγορίας ανά στοιχείο παρτίδας (minibatch), όπως για ανάμεικτες ετικέτες και για εξομάλυνση ετικετών.

– Αν το σφάλμα δεν μειώνεται δηλαδή με τη μείωση σε "none" το μη μειωμένο σφάλφα για αυτήν την περίπτωση μπορεί να περιγραφεί ως:

$$l(x,y) = L = \{l_1, ..., l_N\}^T, \quad \text{με}$$
$$l_n = -w_{y_n} log \frac{exp(x_n, y_n)}{\sum_{c=1}^{C} exp(x_n, c)} y_{n,c}$$

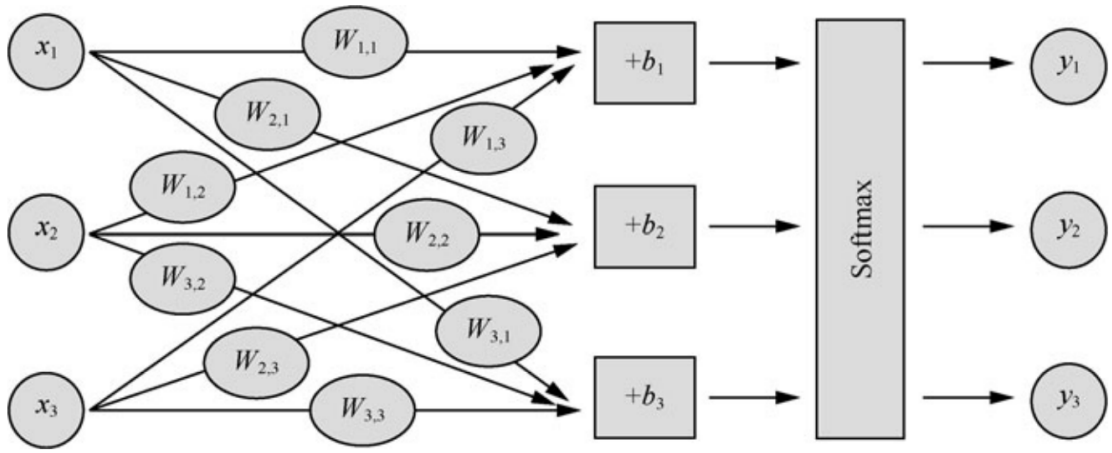
όπου x είναι η είσοδος, y η έξοδος (στόχος/target), w είναι το βάρος (weight), C είναι ο αριθμός των κλάσεων, N καλύπτει την διάσταση της παρτίδας (minibatch) 1 ή K-διάστασεων d_1, d_2, ..., d_K.

– Αν το σφάλμα μειώνεται τότε το μειωμένο σφάλφα για αυτήν την περίπτωση μπορεί να περιγραφεί ως:

$$l(x,y) = \begin{cases} \frac{\sum_{n=1}^{N} l_n}{N} & , if reduction =' mean' \\ \sum_{n=1}^{N} l_n & , if reduction =' sum' \end{cases}$$

3.1.4 Εκπαίδευση του Συνελικτικού Νευρωνικού Δικτύου

Το ΣΝΔ στην φάση της εκπαίδευσης επαναληπτικά θα μειώνει την τιμή της συνάρτησης κόστους προκειμένου η ανανέωση των βαρών των νευρώνων κάθε επιπέδου να οδηγήσει στην εκμάθηση καλύτερων χαρακτηριστικών ώστε να μπορεί να κάνει ταξινόμηση άγνωστων εισόδων.



Σχήμα 3.9: Παράδειγμα: Σχηματική αναπαράσταση της διαδικασίας υπολογισμού του NN προέλευση εικόνας

3.1.5 Ανάλυση δεδομένων ήχου

Υπάρχουν τρία βασικά χαρακτηριστικά που πρέπει να λαμβάνονται υπόψη κατά την ανάλυση δεδομένων ήχου η χρονική περίοδος, το πλάτος και η συχνότητα.

- Η χρονική περίοδος είναι πόσο διαρκεί ένας συγκεκριμένος ήχος ή, με άλλα λόγια, πόσα δευτερόλεπτα χρειάζονται για να ολοκληρωθεί ένας κύκλος δονήσεων.
- Το πλάτος είναι η ένταση του ήχου που μετράται σε ντεσιμπέλ (dB) την οποία αντιλαμβανόμαστε ως ένταση.
- Η συχνότητα που μετράται σε Hertz (Hz) υποδεικνύει πόσες ηχητικές δονήσεις συμβαίνουν ανά δευτερόλεπτο, συνήθως ερμηνεύουμε τη συχνότητα ως χαμηλό ή υψηλό τόνο.

Ενώ η συχνότητα είναι μια αντικειμενική παράμετρος, η ένταση είναι υποκειμενική. Το εύρος της ανθρώπινης ακοής κυμαίνεται μεταξύ 20 και 20.000 Hz. Οι περισσότεροι άνθρωποι αντιλαμβάνονται ως χαμηλού τόνου όλους τους ήχους κάτω από 500 Hz - όπως το βρυχηθμό του κινητήρα του αεροπλάνου. Επίσης, υψηλός τόνος για εμάς είναι οτιδήποτε πάνω από τα 2.000 Hz για παράδειγμα, ένα σφύριγμα.

Μορφές αρχείων ήχου που θα χρησιμοποιήσουμε

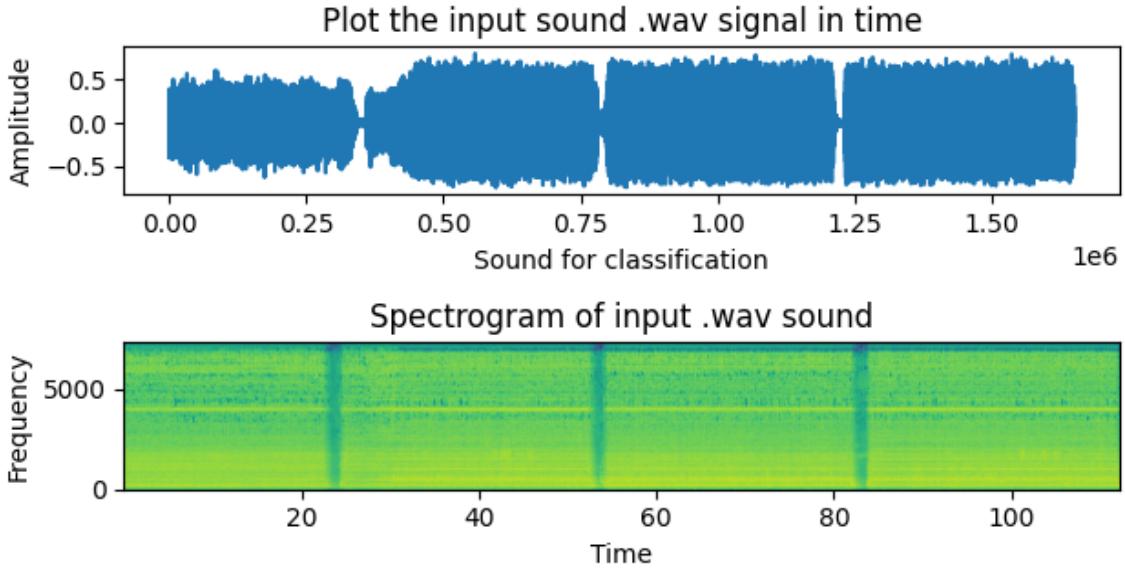
Όπως τα κείμενα και οι εικόνες έτσι και ο ήχος είναι αδόμητα δεδομένα που σημαίνει ότι δεν είναι σε μορφή πινάκων με γραμμές και στήλες. Ο ήχος όταν αποθηκεύεται wave που είναι μορφή lossless ή raw δηλαδή δεν έχει υποστεί συμπίεση η αρχική ηχογράφηση μπορεί να τον επεξεργαστούμε στην συνέχεια για να της ανάγκες της αναγνώρισης και κατηγοριοποίησης του ήχου μέσω μηχανικής μάθησης.

Γραφικές-Οπτικές Αναπαραστάσεις του ηχητικού σήματος

- **Κυματομορφή** είναι μια βασική οπτική αναπαράσταση ενός ηχητικού σήματος που αντικατοπτρίζει τον τρόπο με τον οποίο αλλάζει ένα πλάτος με την πάροδο του χρόνου. Το γράφημα εμφανίζει τον χρόνο στον οριζόντιο άξονα X και το πλάτος στον κάθετο άξονα Y, αλλά δεν μας λέει τι συμβαίνει με τις συχνότητες.

- **Φάσμα ή φασματική γραφική παράσταση** είναι ένα γράφημα όπου ο άξονας Χ δείχνει τη συχνότητα του ηχητικού κύματος ενώ ο άξονας Υ το πλάτος του. Αυτός ο τύπος οπτικοποίησης δεδομένων ήχου μας βοηθά να αναλύσουμε το περιεχόμενο συχνότητας, αλλά χάνει το στοιχείο του χρόνου.

- **Φασματόγραμμα** είναι μια αναπαράσταση ενός σήματος που καλύπτει και τα τρία χαρακτηριστικά του ήχου. Συγκεκριμένα έχουμε τον χρόνο στον άξονα Χ, τις συχνότητες στον άξονα Υ και το πλάτος με το χρώμα. Όσο πιο δυνατή είναι η ένταση του ήχου τόσο πιο φωτεινό είναι το χρώμα. Το να έχουμε και τις τρεις διαστάσεις σε ένα γράφημα είναι πολύ βολικό διότι μας επιτρέπει να παρακολουθήσουμε πώς αλλάζουν οι συχνότητες με την πάροδο του χρόνου. Μπορούμε να μελετήσουμε τον ήχο σε όλη του την πληρότητα και να εντοπίσουμε διάφορες προβληματικές περιοχές (όπως θορύβους) και μοτίβα.



Σχήμα 3.10: Παράδειγμα σήματος εισόδου στον χρόνο και στο φασματογράφημα

Κεφάλαιο 4

Υλοποίηση-Πειράματα

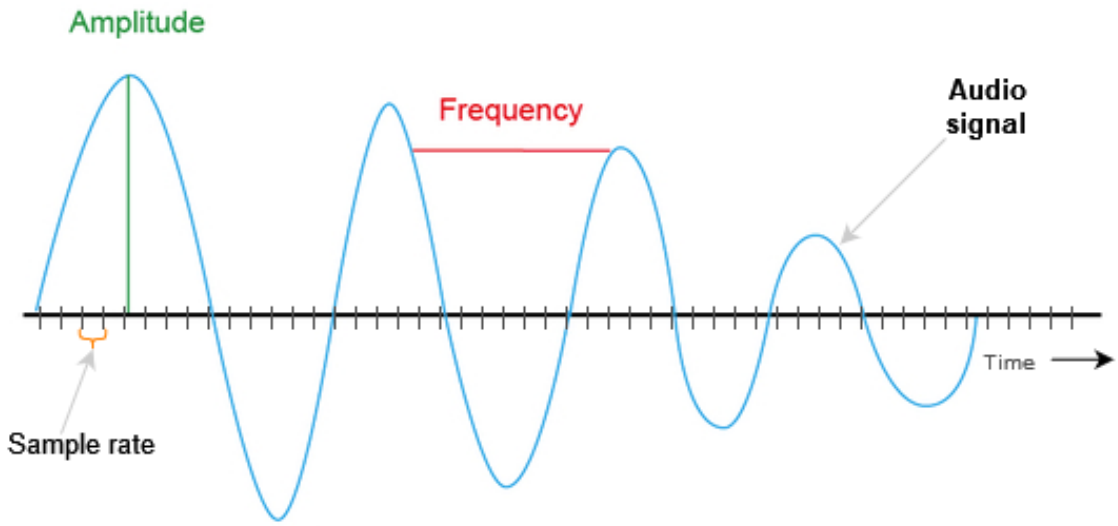
Σε αυτό το κεφάλαιο περιγράφεται λεπτομερειακά η πειραματική διαδικασία για είσοδο ήχου σε μορφή αρχείων wave από τους αισθητήρες, ο κώδικας υπάρχει στο GitHub predictive maintenance

- Δημιουργία του σετ δεδομένων (Data Set) χωρισμού των ήχων για train-validation και αποθήκευση σε φακέλους.
- Εκπαίδευση και αποθήκευση του μοντέλου.
- Επικύρωση της εκπαίδευσης του μοντέλου.
- Εξαγωγή συμπεράσματων.

4.1 Προεπεξεργασία wave αρχείων

Πρέπει πρώτα να κατανοήσουμε καλύτερα τα δεδομένα εισόδου που είναι αρχεία wave, θα εξετάσουμε μερικές βασικές έννοιες και χαρακτηριστικά των δεδομένων ήχου. Ένα ηχητικό σήμα είναι η δόνηση που δημιουργείται όταν ο ήχος διέρχεται από τον αέρα. Για τον ήχο, όταν ο ήχος λαμβάνεται από ένα μικρόφωνο, είναι σε αναλογική μορφή. Ο αναλογικός ήχος μετατρέπεται σε ψηφιακή μορφή ήχου με δειγματοληψία σε σταθερά χρονικά διαστήματα. Ο αριθμός των σημείων δεδομένων ήχου που καταγράφονται κάθε δευτερόλεπτο ονομάζεται ρυθμός δειγματοληψίας. Όσο υψηλότερος είναι ο ρυθμός δειγματοληψίας, τόσο υψηλότερη είναι η ποιότητα του ήχου. Ωστόσο, μετά από ένα ορισμένο σημείο, το ανθρώπινο αυτί δεν μπορεί να εντοπίσει τη διαφορά. Ο μέσος ρυθμός δειγματοληψίας ήχου είναι 48 kilohertz (KHz) ή 48.000 δείγματα ανά δευτερόλεπτο.

Έτσι αν για παράδειγμα στο σύνολο δεδομένων που χρησιμοποιούμε έγινε δειγματοληψία στα 16 KHz, ο ρυθμός δειγματοληψίας μας είναι 16.000.



Σχήμα 4.1: Παράδειγμα αρχείου ήχου

Όταν γίνεται δειγματοληψία του ήχου, η συχνότητα του ήχου είναι ο αριθμός των φορών ανά δευτερόλεπτο που επαναλαμβάνεται ένα ηχητικό κύμα. Το πλάτος είναι πόσο δυνατός είναι ο ήχος. Μπορούμε να πάρουμε το ρυθμό δειγματοληψίας, τη συχνότητά και να αναπαραστήσουμε το σήμα οπτικά (φασματογράφημα). Αυτό το οπτικό σήμα μπορεί να αναπαρασταθεί ως κυματομορφή, η οποία είναι η αναπαράσταση του σήματος με την πάροδο του χρόνου σε μια γραφική μορφή. Επίσης ο ήχος μπορεί να εγγραφεί σε διαφορετικά κανάλια για παράδειγμα, οι στερεοφωνικές εγγραφές έχουν δύο κανάλια, δεξιά και αριστερά.

Εάν έχουμε μεγαλύτερα αρχεία ήχου, μπορεί να θέλουμε να τα χωρίσουμε τμήματα (ή πλαίσια/καρέ) του ήχου που θα ταξινομηθούν ξεχωριστά. Ένα άλλο βήμα επεξεργασίας μπορεί να είναι η μετατόπιση κάποιων αριθμό καρέ από την αρχή του αρχείου wave για να ξεκινήσει η φόρτωση δεδομένων.

4.2 Επεξήγηση των python scripts

4.2.1 Δημιουργία μοντέλου ΣΝΔ

Το ΣΝΔ ταξινομεί και στην συνέχεια προβλέπει την είσοδο που είναι ένα αρχείο wave που χωρίζεται σε blocks διάρκειας seq_dur (5sec) σε ποια κλάση (από τις 4) ανήκει. Η έξοδος του ΣΝΔ είναι ένα μητρώο διάστασης (nb_blocks,nb_classes) και η κάθε του γραμμή είναι οι unormalized αποκρίσεις για τις 4 κλάσεις για ένα block.

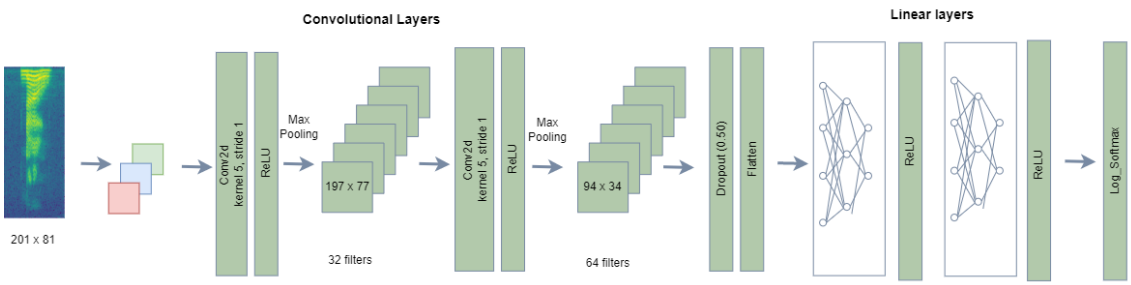
Η αντικειμενική συνάρτηση (objective function) που χρησιμοποιούμε για ελαχιστοποίηση είναι η συνάρτηση διασταυρωμένης εντροπίας που υπάρχει στο pytorch CrossEntropyLoss όπου χρησιμοποιούμε και βάρη για την κάθε κλάση για ισορροπήσουμε είσοδο γιατί το data set συνήθως είναι μη ζυγισμένο (unbalanced).

Python script: model.py

Listing 4.1: Python Παράδειγμα ΣΝΔ model.py

```
class Net(nn.Module):
    def __init__(self,fc_dim,nb_classes):
        super(Net, self).__init__()
        # Define the network components
```

```
self.conv1 = nn.Sequential(
    nn.Conv2d(1, 16, kernel_size = (5, 5), stride=(2, 2), padding=2),
    nn.BatchNorm2d(16),
    nn.LeakyReLU(True)
)
self.conv2 = nn.Sequential(
    nn.Conv2d(16, 32, kernel_size = (5, 5), stride=(2, 2), padding=2)
    nn.BatchNorm2d(32),
    nn.LeakyReLU(True)
)
self.conv3 = nn.Sequential(
    nn.Conv2d(32, 64, kernel_size = (5, 5), stride=(2, 2), padding=2)
    nn.BatchNorm2d(64),
    nn.LeakyReLU(True)
)
self.conv4 = nn.Sequential(
    nn.Conv2d(64, 128, kernel_size = (5, 5), stride=(2, 2), padding=2
    nn.BatchNorm2d(128),
    nn.LeakyReLU(True)
)
self.conv5 = nn.Sequential(
    nn.Conv2d(128, 256, kernel_size = (5, 5), stride=(2, 2), padding=
    nn.BatchNorm2d(256),
    nn.LeakyReLU(True)
)
self.fc = nn.Linear(fc_dim, nb_classes)
```



Σχήμα 4.2: Παράδειγμα ΣΝΔ με PyTorch

4.2.2 Δημιουργία Σετ Δεδομένων (Data Set)

Δημιουργία του dataset σε μορφή φασματογραφημάτων για να χρησιμοποιηθούν για την εκπαίδευση του μοντέλου. **DATASET:** Το Dataset (WAVs) δημιουργήθηκε με τη βοήθεια του λογισμικού Audacity. Ακολουθήκε η εξής διαδικασία:
Οι διαφορετικές κινήσεις μας δίνουν διαφορετικούς ήχους τις οποίους θεωρούμε κλάσεις.

Για παράδειγμα έχουμε 4 κλάσεις ήχου (class 0, 1 , 2, 3) που λαμβάνουμε από τους αισθητήρες κίνησης και να περικόψουμε τα wave αρχεία που είχαμε για την κάθε μια ώστε να τα βάλουμε κατάλληλα στους φακέλους (test,valid) με το αντίστοιχο όνομα.

Python script: Create_Dataset.py
parameters: -dataset-params "{ <Wav_folder>, <Target_folder>, Fs : συχνότητα δηγματοληψίας,

seq_dur : μέγεθος segments, classes_lookup :{ Moter_single : 0 , Motor_gran_no_chain : 1 , seatrak_all_elements : 2 , kinhsh_koble : 3 } , FE_params : { front_end_name : STFT_custom , a : 768 , M : 1024 , support : 1024 } , preproc : None } "

Επεξήγηση παραμέτρων:

1. Wav_folder [string] →Είναι το PATH του φακέλου το οποίο πρέπει να έχει την δομή 4.3 τα αρχεία έχουμε δημιουργήσει με το audacity:

2. Target_folder [string] →Είναι το PATH του φακέλου (το οποίο δημιουργείται αν δεν υπάρχει) και στο οποίο θα αποθηκευτούν τα ακόλουθα:

   - Spec_seg_pair_list_train.pt η μεταβλητή (iterable) η οποία θα περιέχει τα παραδείγματα training ζεύγη (In_spectr,label). η μεταβλητή (iterable) η οποία θα περιέχει τα παραδείγματα validation ζεύγη (In_spectr,label).
   - Spec_seg_pair_list_valid.pt
   - Dataset_Params_log.json το οποίο περιέχει τις παραμέτρους του Dataset.

   τα οποία δημιουργήθηκαν με το προηγούμενο script (Create_Dataset.py).

3. Fs [INT] →Συχνότητα δειγματοληψίας στην οποία γίνονται resample τα wavs **ΧΡΗΣΗ:** σε περίπτωση που έχεις δει ότι οι κυματομορφές δεν περιέχουν ενέργεια πάνω από μια συχνότητα μπορείς να κάνεις resampling σε αυτή για γρηγορότερο processing)

4. seq_dur [INT] →Η διάρκεια της ακολουθίας (σε sec) των παραδειγμάτων εκπαίδευσης τα οποία τροφοδοτούμε στο δίκτυο. **Χρήση:** Λόγω ότι τα αρχεία wavs που έχουμε είναι συνήθως μεγάλης διάρκειας (π.χ. 30mins) σε blocks διάρκειας seq-dur=5 sec ώστε :

   (α΄) Να γίνεται γρηγορότερα το processing

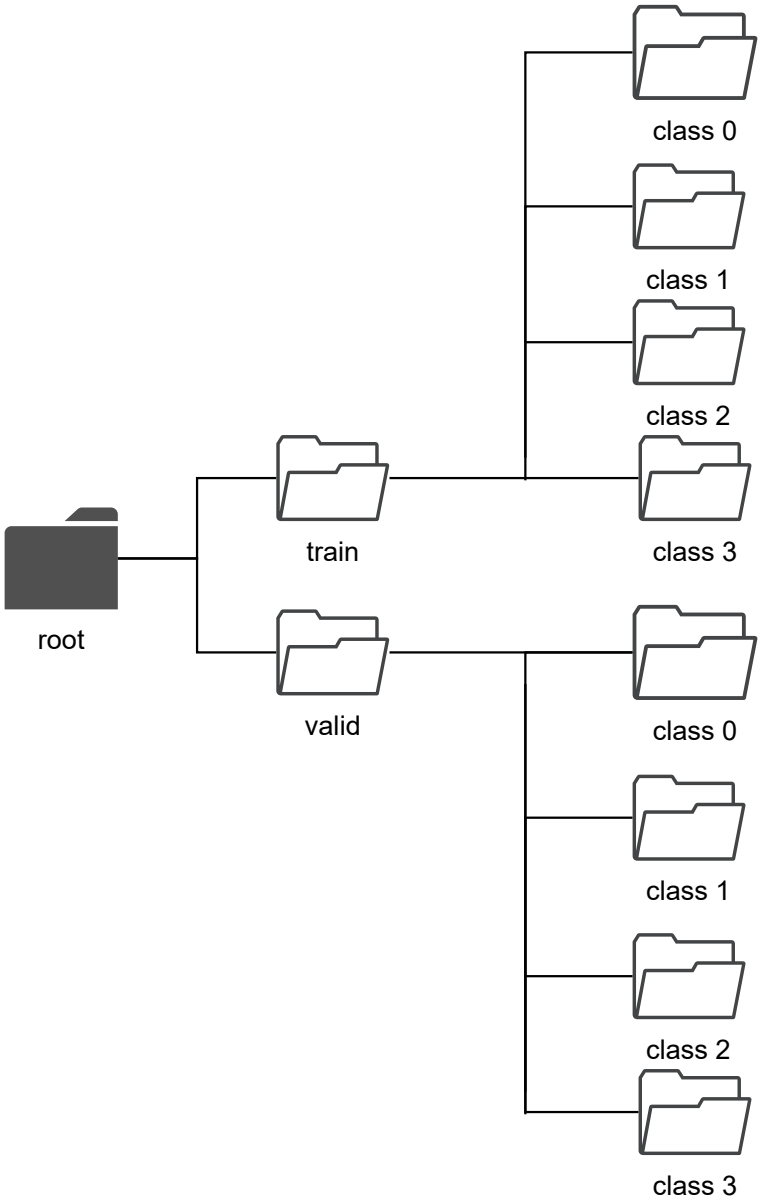
   (β΄) Να έχουμε περισσότερα παραδείγματα εκπαίδευσης

5. FE_params (dictionary) →Είναι οι παράμετροι του FE (front end ή αναπαράσταση εισόδου) με το οποίο θα τροφοδοτούμε το δίκτυο. Επιλέξαμε το FE να είναι φασματογράφημα (spectrogram) για να εκμεταλλευτούμε τα συνελικτικά νευρωνικά δίκτυα καθώς αυτά τα πάνε πολύ καλά με εικόνες και τα spectrograms είναι εικόνες.

6. classes_lookup (dictionary) →Είναι ένα lookup table όπου :

   - key [string] →είναι το όνομα του φακέλου μιας κλάσης.
   - value [int] →είναι το αντίστοιχο label.

Σημαντική Σημείωση: Για λόγους υλοποίησης το key value pair που αντιστοιχεί στην κλάση με label=0 να γράφεται πρώτο στο classes_lookup dictionary.



Σχήμα 4.3: Παράδειγμα φακέλων με το σετ των δεδομένων train-valid

Listing 4.2: Python Παράδειγμα Δημιουργία φακέλων με Data Set CreateData.py

```
c:\ML1\ML_pipeline_code_multiple_classes_v1>python3 Create_Dataset.py -datase
C:\Users\mnano\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qb
  warn("nsgt.fft falling back to numpy.fft")
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/
: : 0it [00:00, ?it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:01<00:00, 1.92s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/
```

```
1.92s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/Mot
: : 1it [00:01,  1.99s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/Mot
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:01<00:00,  1.84s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/Mot
1.84s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/sea
: : 2it [00:03,  1.95s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/sea
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:03<00:00,  3.52s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/sea
3.52s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/kin
: : 3it [00:07,  2.72s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/kin
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:00<00:00,  2.42it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/kin
2.42it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/train/kin
: : 4it [00:07,  1.99s/it]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/Mo
: : 0it [00:00, ?it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/Mo
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:00<00:00,  6.11it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/Mo
6.11it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/Mot
: : 1it [00:00,  5.92it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/Mot
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:00<00:00,  6.11it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/Mot
6.11it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/sea
: : 2it [00:00,  5.91it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/sea
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:00<00:00,  8.23it/s]
```

```
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/
8.23it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/
: : 3it [00:00,  6.66it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/
| 0/1 [00:00<?, ?it/s]

100%|| 1/1 [00:00<00:00,  6.02it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/
6.02it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/valid/
: : 4it [00:00,  6.29it/s]
```

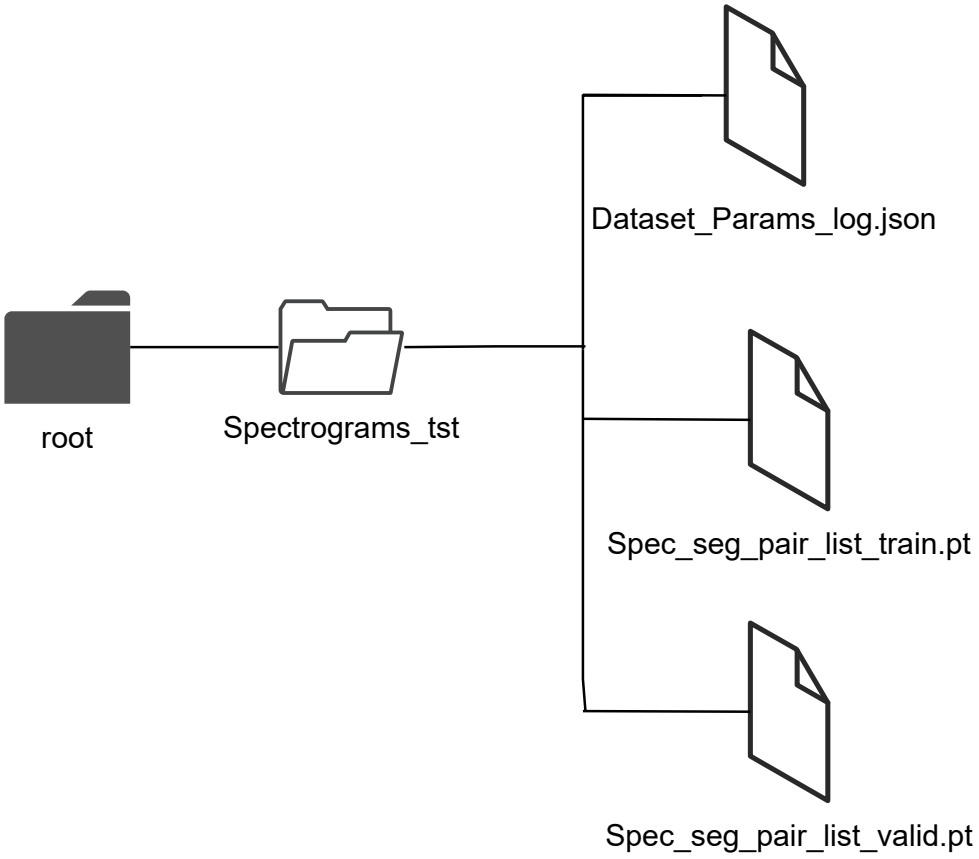
Με το τρέξιμο του script (Create_Dataset.py) δημιουργούνται τα παρακάτω αρχεία στον φάκελο που έχει ως όρισμα Target_folder.

- Spec_seg_pair_list_train.pt

- Spec_seg_pair_list_valid.pt

- Dataset_Params_log.json τα οποία δημιουργήθηκαν με το προηγούμενο script (Create_-Dataset.py).

Κώδικας για την δημιουργία φακέλων με Data Set CreateData.py στο GitHub



Σχήμα 4.4: Παράδειγμα δημιουργία φακέλου και αρχείων σετ δεδομένων από το CreateData.py

4.2.3 Εκπαίδευση του μοντέλου

Η εντολή **python3** train.py --root /home/user/ML_pipeline_code_multiple_classes_v1/Spectrograms_-tst --nb_classes 4 --output /home/user/ML_pipeline_code_multiple_classes_v1/Spectrograms_-tst/pretr_model --epochs 10

Python script: train.py
Επεξήγηση παραμέτρων:

- root (string) →Είναι το path του φακέλου το οποίο περιέχει τα αρχεία:
  - Spec_seg_pair_list_train.pt
  - Spec_seg_pair_list_valid.pt
  - Dataset_Params_log.json τα οποία δημιουργήθηκαν με το προηγούμενο script (Create_-Dataset.py). Ως εκ τούτου το path αυτό θα πρέπει να είναι ίδιο με το Target_folder (παράμετρος του Create_Dataset.py)
- output (string) →Είναι το PATH του φακέλου (το οποίο δημιουργείται αν δεν υπάρχει) και στο οποίο θα αποθηκευτούν τα ακόλουθα αρχεία:

- model.pth →Αρχείο απαραίτητο αν θέλουμε να χρησιμοποιήσουμε το μοντέλο για inference ή για evaluation

- model.json →Αρχείο Log το οποίο περιέχει στοιχεία για την εκπαίδευση δηλαδή training-validation losses, execution time, Dataset parameters, arguments του train.py script.

- model.chkpnt →Αρχείο απαραίτητο αν θέλουμε να συνεχίσουμε την εκπαίδευση ενός ήδη εκπαιδευμένου μοντέλου από εκεί που είχε σταματήσει.

- nb_classes →Είναι το πλήθος των κλάσεων για τις οποίες θα εκπαιδευτεί το ΣΝΔ.

- **Βασικές Υπερπαράμετροι (hyperparameters) εκπαίδευσης:**

  - epochs [INT] →Το πλήθος των εποχών τις οποίες θα εκπαιδευτεί το Νευρωνικό δίκτυο

  - batch-size [INT] →Το μέγεθος της παρτίδας (batch) με το οποίο τροφοδοτούμε το ΣΝΔ, δηλαδή το πλήθος των παραδειγμάτων που του δίνουμε ώστε μετά να κάνει οπισθοδιάδοσης (backpropagation) σε όλα τα επίπεδα του ΣΝΔ ώσπου να μειωθεί το σφάλμα δηλαδή η διαφορά μεταξύ πρόβλεψης και πραγματικής τιμής και να ολοκληρωθεί έτσι η εκπαίδευσή του. Όσο μεγαλύτερο το μέγεθος της παρτίδας με τη διαδικασία εκπαίδευσης θα βρεθεί σίγουρα ένα τοπικό ελάχιστο. Έχουμε επίσης γρηγορότερη επεξεργασία καθώς εκμεταλλευό-μαστε περισσότερο την GPU όμως έχουμε περισσότερες απαιτήσεις σε μνήμη.

Listing 4.3: Python Κώδικας για την χρήση GPU

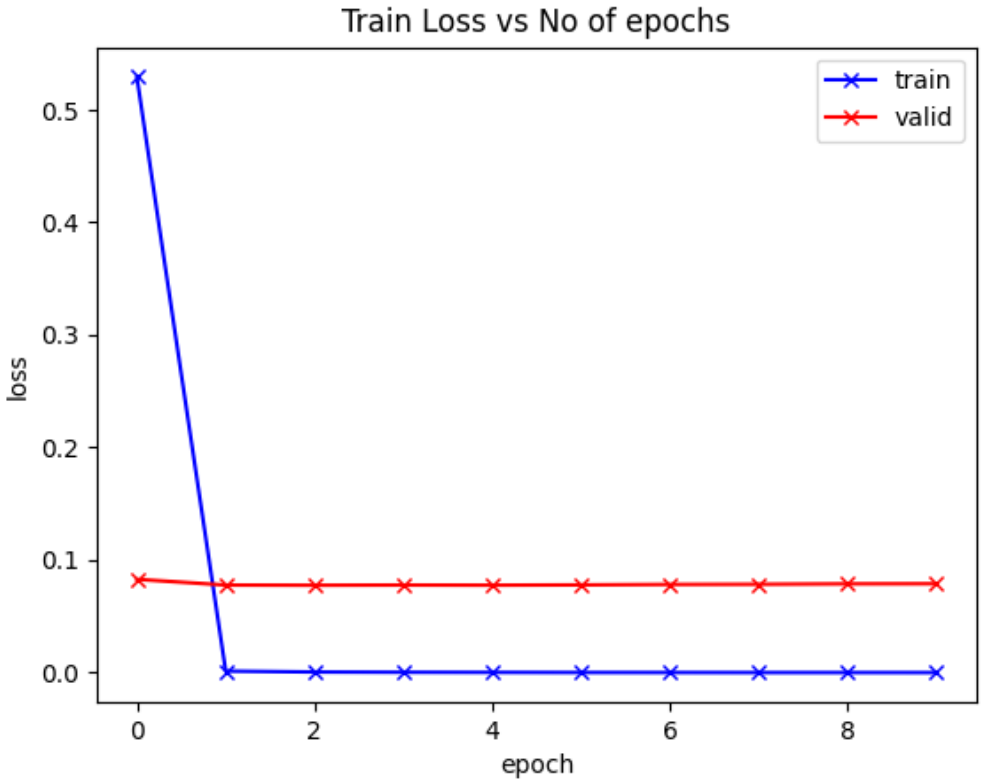
```
use_cuda = not args.no_cuda and torch.cuda.is_available()
```

Αφού τρέξουμε στην GPU το train.py ας δούμε τη συνοπτική ανάλυση της αρχιτεκτονικής του μοντέλου η οποία δείχνει τον αριθμό των φίλτρων που χρησιμοποιούνται για την εξαγωγή χαρακτηριστικών και τη μείωση της εικόνας του φασματογραφήματος από τη συγκέντρωση (pooling) για κάθε συνελικτικό επίπεδο. Στη συνέχεια, εμφανίζει 819456 χαρακτηριστικά εισόδου και τις 4 εξόδους που χρησιμοποιούνται για ταξινόμηση στα γραμμικά επίπεδα.

```
c:\ML1\ML_pipeline_code_multiple_classes_v1>python3 train.py --root c:\ML1\ML_pipeline_code_multiple_classes_v1\Spectrograms_
tst --nb_classes 4 --output c:\ML1\ML_pipeline_code_multiple_classes_v1\Spectrograms_tst\pretr_model --epochs 10
Using GPU: True
the model will be running on cuda device
  0%|                                                                                                    | 0/10 [00:00<?, ?it/s]=
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net                                      [16, 4]                   --
├─Sequential: 1-1                        [16, 16, 257, 145]        --
│    └─Conv2d: 2-1                       [16, 16, 257, 145]        416
│    └─BatchNorm2d: 2-2                  [16, 16, 257, 145]        32
│    └─LeakyReLU: 2-3                    [16, 16, 257, 145]        --
├─Sequential: 1-2                        [16, 32, 129, 73]         --
│    └─Conv2d: 2-4                       [16, 32, 129, 73]         12,832
│    └─BatchNorm2d: 2-5                  [16, 32, 129, 73]         64
│    └─LeakyReLU: 2-6                    [16, 32, 129, 73]         --
├─Sequential: 1-3                        [16, 64, 65, 37]          --
│    └─Conv2d: 2-7                       [16, 64, 65, 37]          51,264
│    └─BatchNorm2d: 2-8                  [16, 64, 65, 37]          128
│    └─LeakyReLU: 2-9                    [16, 64, 65, 37]          --
├─Sequential: 1-4                        [16, 128, 33, 19]         --
│    └─Conv2d: 2-10                      [16, 128, 33, 19]         204,928
│    └─BatchNorm2d: 2-11                 [16, 128, 33, 19]         256
│    └─LeakyReLU: 2-12                   [16, 128, 33, 19]         --
├─Sequential: 1-5                        [16, 256, 17, 10]         --
│    └─Conv2d: 2-13                      [16, 256, 17, 10]         819,456
│    └─BatchNorm2d: 2-14                 [16, 256, 17, 10]         512
│    └─LeakyReLU: 2-15                   [16, 256, 17, 10]         --
├─Linear: 1-6                            [16, 4]                   595,084
==========================================================================================
Total params: 1,684,972
Trainable params: 1,684,972
Non-trainable params: 0
Total mult-adds (G): 8.45
==========================================================================================
Input size (MB): 9.52
Forward/backward pass size (MB): 300.87
Params size (MB): 6.74
Estimated Total Size (MB): 317.13
==========================================================================================
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 52.70it/s, loss=0.529]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.47it/s, loss=0.001]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.34it/s, loss=0.001]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.73it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.21it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.07it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 64.87it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 65.44it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.03it/s, loss=0.000]
Training batch: 100%|████████████████████████████████████████████████████| 49/49 [00:00<00:00, 66.58it/s, loss=0.000]
Training epoch: 100%|██████████████████████████████| 10/10 [00:12<00:00,  1.29s/it, train_loss=0.000143, val_loss=0.0791]

c:\ML1\ML_pipeline_code_multiple_classes_v1>
```

Σχήμα 4.5: Παράδειγμα τρεξίματος εκπαίδευσης του μοντέλου με το ΣΝΔ

Στο παρακάτω σχήμα βλέπουμε το σφάλμα εκπαίδευσης μπλε γραμμή που είναι πολύ μικρό από την 2η εποχή και και το σφάλμα επικύρωσης κόκκινη γραμμή το οποίο είναι χαμηλό από 1η εποχή.



Σχήμα 4.6: Παράδειγμα σφάλμα εκπαίδευσης - σφάλμα επικύρωσης

Listing 4.4: Python Κώδικας για την εκπαίδευση ΣΝΔ train.py

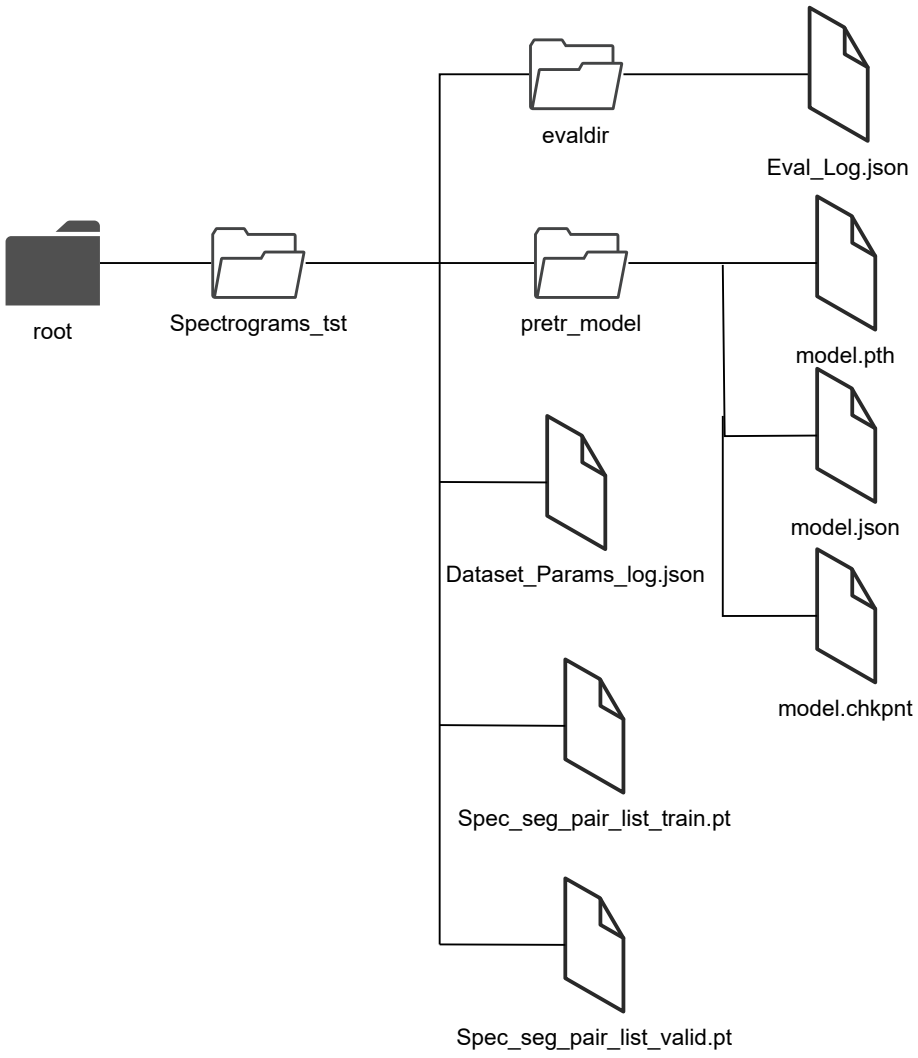
```
for epoch in t:
        t.set_description("Training␣epoch")
        end = time.time()
        train_loss = train(args, unmix , device, train_sampler, optimizer,cri
        valid_loss = valid(args, unmix, device, valid_sampler,critireon)
        scheduler.step(valid_loss)
        train_losses.append(train_loss)
        valid_losses.append(valid_loss)
        t.set_postfix(train_loss=train_loss, val_loss=valid_loss)
        stop = es.step(valid_loss)

        if valid_loss == es.best:
            best_epoch = epoch
        utils.save_checkpoint(
            {
                "epoch": epoch + 1,
                "state_dict": unmix.state_dict(),
                "best_loss": es.best,
                "optimizer": optimizer.state_dict(),
                "scheduler": scheduler.state_dict(),
            },
```

```
            is_best=valid_loss == es.best,
            path=target_path,
            target="model",
        )
        # save params
        with open(args.root+'/Dataset_Params_log.json', 'r') as openfile:
            # Reading from json file
            Dataset_params = json.load(openfile)
        args_json = vars(args)
        args_json["Dataset_params"] = Dataset_params
        args_json["fc_dim"] = fc_dim
        params = {
            "epochs_trained": epoch,
            # "args": vars(args),
            "args": args_json,
            "best_loss": es.best,
            "best_epoch": best_epoch,
            "train_loss_history": train_losses,
            "valid_loss_history": valid_losses,
            "train_time_history": train_times,
            "num_bad_epochs": es.num_bad_epochs,
            #"commit": commit,
        }
        with open(Path(target_path, "model" + ".json"), "w") as outfile:
            outfile.write(json.dumps(params, indent=4, sort_keys=False))
        train_times.append(time.time() - end)
        if stop:
            print("Apply Early Stopping")
            break
```

Listing 4.5: Python παράμετροι εκπαίδευσης του ΣΝΔ train.py

```
# Training Parameters
    parser.add_argument("--epochs", type=int, default=1000)
    parser.add_argument("--batch-size", type=int, default=16)
    parser.add_argument("--lr", type=float, default=0.001, help="learning rate, 
    parser.add_argument(
        "--patience",
        type=int,
        default=300,
        help="maximum number of train epochs (default: 140)",
    )
```

κώδικας train.py στο GitHub



Σχήμα 4.7: Παράδειγμα καταλόγων και αρχείων με το εκπαιδευμένο μοντέλο train.py

4.2.4 Αξιολόγηση του εκπαιδευμένου μοντέλου

Με το script evaluate αξιολογούμε την απόδοση του δικτύου στο πόσο καλά κατηγοριοποιεί τα blocks που του δίνουμε στις διάφορες κλάσεις. Για να αποφανθούμε σε πια κλάση ανήκει ένα block: Εφαρμόζουμε στην unormalized απόκριση του δικτύου (διάνυσμα 4 στοιχείων όσες και οι κλάσεις) τη softmax μη-γραμμικότητα.

Αυτό το βήμα το κάνουμε έτσι ώστε να μοντελοποιήσουμε (μετατρέψουμε) αυτές τις unormalized αποκρίσεις σε 4 πιθανότητες με τιμή από 0 μέχρι το 1 .

Ο δείκτης της μεγαλύτερης πιθανότητας (του διανύσματος) αποτελεί και την κλάση που απαντάει το δίκτυο για το συγκεκριμένη παρτίδα batch.

Python script: evaluate.py κώδικας evaluate.py στο github

Listing 4.6: Python αξιολόγηση evaluate.py

```
def evaluate(y_true, y_pred):
    #y_true, y_pred referes to all the test samples given to the NN
```

```
    from sklearn.metrics import confusion_matrix
    classes_names = list(Dataset_params["classes_lookup"].keys())
    confusion = confusion_matrix(y_true, y_pred)
    print('Confusion Matrix\n')
    print(confusion)
    #importing accuracy_score, precision_score, recall_score, f1_score
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f
    print('\nAccuracy: {:.2f}\n'.format(accuracy_score(y_true, y_pred)))
    print('Micro Precision: {:.2f}'.format(precision_score(y_true, y_pred, avera
    print('Micro Recall: {:.2f}'.format(recall_score(y_true, y_pred, average='mi
    print('Micro F1-score: {:.2f}\n'.format(f1_score(y_true, y_pred, average='mi
    print('Macro Precision: {:.2f}'.format(precision_score(y_true, y_pred, avera
    print('Macro Recall: {:.2f}'.format(recall_score(y_true, y_pred, average='ma
    print('Macro F1-score: {:.2f}\n'.format(f1_score(y_true, y_pred, average='ma
    print('Weighted Precision: {:.2f}'.format(precision_score(y_true, y_pred, av
    print('Weighted Recall: {:.2f}'.format(recall_score(y_true, y_pred, average=
    print('Weighted F1-score: {:.2f}'.format(f1_score(y_true, y_pred, average='w
    from sklearn.metrics import classification_report
    print('\nClassification Report\n')
    print(classification_report(y_true, y_pred, target_names=classes_names ))

def Inference_and_evaluate(val_sampler,model):
    #Inference_AND_EVALUATE------------------------------
    device = "cpu"
    #INFERENCE-------------------------------------------
    model.eval()
    with torch.no_grad():
        for count,(x , y) in enumerate(val_sampler):
            x = x.to(device)
            if count:
                #INFERENCE FOR ONE BATCH-----------------------------------------
                seg_pred_class = inference(x,model)
                y_pred = y_pred + list(seg_pred_class)
                #---------------------------------------------------------------
                y = y.cpu().detach().numpy()
                y_true = y_true + list(y)
            else:
                #INFERENCE FOR ONE BATCH-----------------------------------------
                seg_pred_class = inference(x,model)
                y_pred = list(seg_pred_class)
                #---------------------------------------------------------------
                y = y.cpu().detach().numpy()
                y_true = list(y)
    #-----------------------------------
    #EVALUATE--------------------------------------------------------------------
    evaluate(y_true,y_pred)
```

```
C:\Users\mnano\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\
site-packages\nsgt\fft.py:116: UserWarning: nsgt.fft falling back to numpy.fft
  warn("nsgt.fft falling back to numpy.fft")
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Moter_single class dir of wavs
: : 0it [00:00, ?it/s]                                                                                          p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Moter_single               | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  4.56it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Moter_single class dir of wavs<00:00,  4.57it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Motor_gran_no_chain class dir of wavs
: : 1it [00:00,  4.42it/s]                                                                                      p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Motor_gran_no_chain        | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  3.95it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Motor_gran_no_chain class dir of wavs  3.95it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/seatrak_all_elements class dir of wavs
: : 2it [00:00,  4.06it/s]                                                                                      p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/seatrak_all_elements       | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  5.22it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/seatrak_all_elements class dir of wavs 5.22it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble class dir of wavs
: : 3it [00:00,  4.45it/s]                                                                                      p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble               | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  4.65it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble class dir of wavs<00:00,  4.65it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble class dir of wavs
: : 4it [00:00,  4.39it/s]
Confusion Matrix

[[23  0  0  0]
 [ 0 23  0  0]
 [ 0  0 23  0]
 [ 0  0  0 23]]

Accuracy: 1.00

Micro Precision: 1.00
Micro Recall: 1.00
Micro F1-score: 1.00

Macro Precision: 1.00
Macro Recall: 1.00
Macro F1-score: 1.00

Weighted Precision: 1.00
Weighted Recall: 1.00
Weighted F1-score: 1.00

Classification Report

                     precision    recall  f1-score   support

       Moter_single       1.00      1.00      1.00        23
 Motor_gran_no_chain       1.00      1.00      1.00        23
seatrak_all_elements       1.00      1.00      1.00        23
       kinhsh_koble       1.00      1.00      1.00        23

           accuracy                           1.00        92
          macro avg       1.00      1.00      1.00        92
       weighted avg       1.00      1.00      1.00        92
```

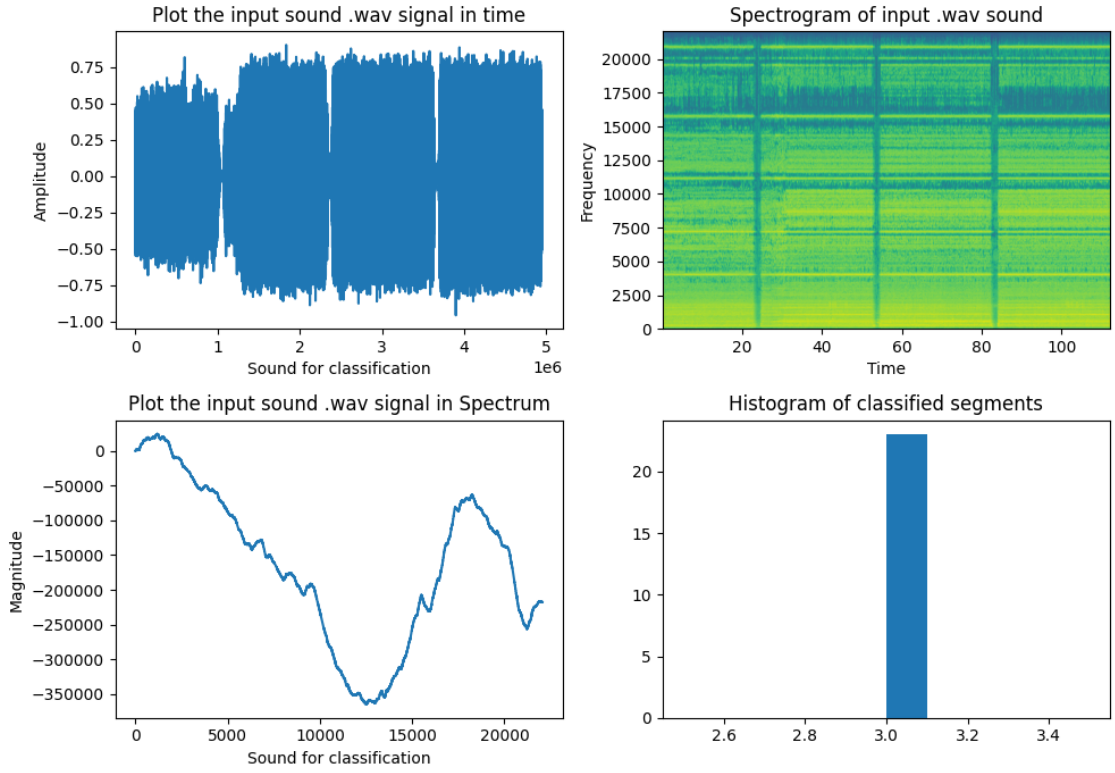
Σχήμα 4.8: Παράμετροι αξιολόγησης εκπαίδευσης μοντέλου

4.2.5 Χρήση του εκπαιδευμένου μοντέλου για αναγνώριση ενός ήχου

Στο inference για να αποφανθούμε για το σε πια κλάση ανήκει το input-wav εκτελούμε το ίδιο βήμα με αυτό στο evaluation στην έξοδο του δικτύου. Εδώ όμως θέλουμε να αποφασίσουμε σε πια κλάση ανήκει όλο το input-wav και όχι απλά τα blocks του. Επομένως αποφασίζουμε υπέρ της κλάσης με τα περισσότερα occurences, δηλαδή σαν να συμβουλευόμαστε ένα ιστόγραμμα. Θα μπορούσαμε να πούμε ότι η κλάση που ανήκει ένα block είναι τυχαία μεταβλητή και μέσω αυτού του ιστογράμματος υπολογίζουμε τη κατανομή αυτής της τυχαίας μεταβλητής.

Python script: inference.py κώδικας στο GitHub

```
c:\ML1\ML_pipeline_code_multiple_classes_v1>python3 inference.py --Model_dir c:\ML1\ML_pipeline_code_multiple_classes_v1\Spec
trograms_tst\pretr_model --input-wav c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset\test\kinhsh_koble\kinisi-komple-
3.wav
C:\Users\mnano\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\
site-packages\nsgt\fft.py:116: UserWarning: nsgt.fft falling back to numpy.fft
  warn("nsgt.fft falling back to numpy.fft")
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
Most frequent value in the above array:
3
kinhsh_koble
```



Σχήμα 4.9: Παράδειγμα αναγνώρισης ήχου από το εκπαιδευμένο μοντέλο

4.2.6 Χρήση του εκπαιδευμένου μοντέλου για αναγνώριση ενός ήχου σε πραγματικό χρόνο

Στο inference_real_time για να αποφανθούμε για το σε πια κλάση ανήκει το input-wav εκτελούμε το ίδιο βήμα με αυτό στο evaluation στην έξοδο του δικτύου. Εδώ όμως θέλουμε να αποφασίσουμε σε πια κλάση ανήκει όλο το input-wav και όχι απλά τα blocks του. Επομένως αποφασίζουμε υπέρ της κλάσης με τα περισσότερα occurences, δηλαδή σαν να συμβουλευόμαστε ένα ιστόγραμμα (Θα μπορούσαμε να πούμε ότι η κλάση που ανήκει ένα block είναι τυχαία μεταβλητή και μέσω αυτού του ιστογράμματος υπολογίζουμε τη κατανομή αυτης της τυχαίας μεταβλητής).

Python script: inference_real_time.py κώδικας στο GitHub

```
c:\ML1\ML_pipeline_code_multiple_classes_v1>python3 inference_real_time.py --Model_dir c:\ML1\ML_pipeline_code_multiple_class
es_v1\Spectrograms_tst\pretr_model
C:\Users\mnano\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\
site-packages\nsgt\fft.py:116: UserWarning: nsgt.fft falling back to numpy.fft
  warn("nsgt.fft falling back to numpy.fft")
seatrak_all_elements
seatrak_all_elements
Moter_single
Moter_single
Moter_single
Moter_single
Moter_single
Moter_single
kinhsh_koble
Moter_single
Moter_single
Moter_single
Moter_single
Moter_single
seatrak_all_elements
seatrak_all_elements
Moter_single
```

Σχήμα 4.10: Παράδειγμα αναγνώρισης ήχου σε πραγματικό από το εκπαιδευμένο μοντέλο

4.3 Εκπαίδευση

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net                                      [16]                      --
├─Sequential: 1-1                        [16, 16, 257, 80]         --
│    └─Conv2d: 2-1                       [16, 16, 257, 80]         416
│    └─BatchNorm2d: 2-2                  [16, 16, 257, 80]         32
│    └─LeakyReLU: 2-3                    [16, 16, 257, 80]         --
├─Sequential: 1-2                        [16, 32, 129, 40]         --
│    └─Conv2d: 2-4                       [16, 32, 129, 40]         12,832
│    └─BatchNorm2d: 2-5                  [16, 32, 129, 40]         64
│    └─LeakyReLU: 2-6                    [16, 32, 129, 40]         --
├─Sequential: 1-3                        [16, 64, 65, 20]          --
│    └─Conv2d: 2-7                       [16, 64, 65, 20]          51,264
│    └─BatchNorm2d: 2-8                  [16, 64, 65, 20]          128
│    └─LeakyReLU: 2-9                    [16, 64, 65, 20]          --
├─Sequential: 1-4                        [16, 128, 33, 10]         --
│    └─Conv2d: 2-10                      [16, 128, 33, 10]         204,928
│    └─BatchNorm2d: 2-11                 [16, 128, 33, 10]         256
│    └─LeakyReLU: 2-12                   [16, 128, 33, 10]         --
├─Sequential: 1-5                        [16, 256, 17, 5]          --
│    └─Conv2d: 2-13                      [16, 256, 17, 5]          819,456
│    └─BatchNorm2d: 2-14                 [16, 256, 17, 5]          512
│    └─LeakyReLU: 2-15                   [16, 256, 17, 5]          --
├─Linear: 1-6                            [16, 1]                   82,081
==========================================================================================
Total params: 1,171,969
Trainable params: 1,171,969
Non-trainable params: 0
Total mult-adds (G): 4.46
==========================================================================================
Input size (MB): 5.25
Forward/backward pass size (MB): 164.17
Params size (MB): 4.69
Estimated Total Size (MB): 174.11
==========================================================================================
```

Σχήμα 4.11: Torchinfo Summary

Loss function:
Κάνει ακριβώς αυτό που κάνουμε στο inference.

1. Εφαρμογή sigmoid
2. Εφαρμογή mean

Περισσότερες πληροφορίες:
`https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`

Πίνακας υπερπαραμέτρων

Υπερπαράμετρος	Τιμή
--epochs	10
--batch-size	32
--patience	10 epochs
--seq-dur	5 sec
--hidden-size	1
--learning-rate	0,001
--lr-decay-patience	80
--lr-decay-gamma	0,3
--weight-decay	0,00001
--nb-workers	6

Καμπύλες εκμάθησης

4.4 Αξιολόγηση

Μετρικές αξιολόγησης

- F1 score
- Recall
- Precision

```
C:\Users\mnano\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\
site-packages\nsgt\fft.py:116: UserWarning: nsgt.fft falling back to numpy.fft
  warn("nsgt.fft falling back to numpy.fft")
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Moter_single class dir of wavs
: : 0it [00:00, ?it/s]                                                                                    p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Moter_single              | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  4.56it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Moter_single class dir of wavs<00:00,  4.57it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Motor_gran_no_chain class dir of wavs
: : 1it [00:00,  4.42it/s]                                                                                p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Motor_gran_no_chain       | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  3.95it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/Motor_gran_no_chain class dir of wavs  3.95it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/seatrak_all_elements class dir of wavs
: : 2it [00:00,  4.06it/s]                                                                                p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/seatrak_all_elements      | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  5.22it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/seatrak_all_elements class dir of wavs 5.22it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble class dir of wavs
: : 3it [00:00,  4.45it/s]                                                                                p
rocessing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble              | 0/1 [00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  4.65it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble class dir of wavs<00:00,  4.65it/s]
Processing the c:\ML1\ML_pipeline_code_multiple_classes_v1\tst_dataset/test/kinhsh_koble class dir of wavs
: : 4it [00:00,  4.39it/s]
Confusion Matrix

[[23  0  0  0]
 [ 0 23  0  0]
 [ 0  0 23  0]
 [ 0  0  0 23]]

Accuracy: 1.00

Micro Precision: 1.00
Micro Recall: 1.00
Micro F1-score: 1.00

Macro Precision: 1.00
Macro Recall: 1.00
Macro F1-score: 1.00

Weighted Precision: 1.00
Weighted Recall: 1.00
Weighted F1-score: 1.00

Classification Report

                     precision    recall  f1-score   support

       Moter_single       1.00      1.00      1.00        23
 Motor_gran_no_chain       1.00      1.00      1.00        23
seatrak_all_elements       1.00      1.00      1.00        23
       kinhsh_koble       1.00      1.00      1.00        23

           accuracy                           1.00        92
          macro avg       1.00      1.00      1.00        92
       weighted avg       1.00      1.00      1.00        92
```

Σχήμα 4.12: Παράμετροι αξιολόγησης εκπαίδευσης μοντέλου

Η πηγή που χρησιμοποιήθηκε βρίσκεται σε αυτό το σύνδεσμο.

4.5 Λεπτομέρειες Συστημάτων που διεξήχθησαν τα πειράματα

Χαρακτηριστικό	
model	KVM
O/S	Ubuntu 20.04.5 LTS
CPU(s):	8
On-line CPU(s) list:	0-7
Thread(s) per core:	1
Core(s) per socket:	8
Socket(s):	1
NUMA node(s):	1
Vendor ID:	GenuineIntel
CPU family:	15
Model:	6
Model name:	Common KVM processor
Stepping:	1
CPU MHz:	2095.076
BogoMIPS:	4190.15
Hypervisor vendor:	KVM
Virtualization type:	full
L1d cache:	256 KiB
L1i cache:	256 KiB
L2 cache:	32 MiB
L3 cache:	16 MiB

Item	Value
OS Name	Microsoft Windows 11 Pro
Version	10.0.22621 Build 22621
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	DELL_9520_MIKE
System Manufacturer	Dell Inc.
System Model	XPS 15 9520
System Type	x64-based PC
System SKU	0B19
Processor	12th Gen Intel(R) Core(TM) i7-12700H, 2300 Mhz, 14 Core(s), 20 Logical Pro...
BIOS Version/Date	Dell Inc. 1.8.1, 10/13/2022
SMBIOS Version	3.4
Embedded Controller Version	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	Dell Inc.
BaseBoard Product	0YD3W1
BaseBoard Version	A00
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "10.0.22621.819"
User Name	dell_9520_mike\mnano
Time Zone	GTB Standard Time
Installed Physical Memory (RAM)	16.0 GB
Total Physical Memory	15.7 GB
Available Physical Memory	4.40 GB
Total Virtual Memory	37.7 GB
Available Virtual Memory	19.0 GB
Page File Space	22.0 GB
Page File	C:\pagefile.sys
Kernel DMA Protection	On
Virtualization-based security	Running

Σχήμα 4.13: windows system

4.6 Προγραμματιστικό περιβάλλον

Όλα τα πειράματα έγιναν με χρήση της γλώσσας προγραμματισμού **Python**

$$Python \quad 3 \quad version : 3.10.9(default, Dec \quad 6 \quad 2022, 20 : 01 : 21)$$

Οι βασικές βιβλιοθήκες[1] που χρειάστηκαν για τη διεξαγωγή των πειραμάτων είναι οι παρακάτω:

- **numpy**
- **argparse**
- **librosa**
- **torch**
- **tqdm**
- **pathlib**

[1] Θα δοθούν σε ένα requirments.txt για την αναπαραγωγή των αποτελεσμάτων

- **Time_Frequency_Analysis**
  `https://github.com/nnanos/Time_Frequency_Analysis`
- **scipy**
- **typing**
- **sklearn**
- **torchaudio**
- **torchinfo**
- **Audio_proc_lib**
  `https://github.com/nnanos/Audio_proc_lib`
- **openunmix**
- **PyTorch NSGT/sliCQ**
  `https://github.com/sevagh/nsgt`

Κεφάλαιο 5

Σημεία Μελλοντικής Έρευνας

Η παραπάνω πειραματική διαδικασία του κεφαλαίου 4 μπορεί:

1. Να επεκταθεί για περισσότερα του ενός WAV ανά κλάση.
2. Να επεκταθεί για είσοδο αρχεία εικόνας (π.χ. υπέρυθρων) και δημιουργία κλάσεων εικόνας.
3. Να επεκταθεί για περισσότερες κλάσεις.

Επίσης μπορούν να υλοποιηθούν πλήρως τα βήματα που περιγράφονται στο σχήμα3.2 για το σύστημα τεχνητής νοημοσύνης για την προγνωστική συντήρηση.

Να ενσωματωθούν τα παραπάνω σε μια συσκευή επαυξημένης πραγματικότητας π.χ. microsoft hololens με την δημιουργία μιας εφαρμογής επαυξημένης πραγματικότητας για την απεικόνιση αυτών σε πραγματικό χρόνο.

Βιβλιογραφία

[1] Heejeong Choi, Subin Kim και Pilsung Kang. *Recurrent Auto-Encoder With Multi-Resolution Ensemble and Predictive Coding for Multivariate Time-Series Anomaly Detection*. 2022. DOI: 10.48550/ARXIV.2202.10001. URL: https://arxiv.org/abs/2202.10001.

[2] Designated by Huawei HiAI certi. *Artificial Intelligence Technology*. 2022. DOI: 10.1007/978-981-19-2879-6. URL: https://doi.org/10.1007/978-981-19-2879-6.

[3] Kamat, Pooja και Sugandhi, Rekha. *Anomaly Detection for Predictive Maintenance in Industry 4.0- A survey*. 2020. DOI: https://doi.org/10.1051/e3sconf/202017002007. URL: https://doi.org/10.1051/e3sconf/202017002007.