

Time series unsupervised anomaly detection for server data

2023-2024



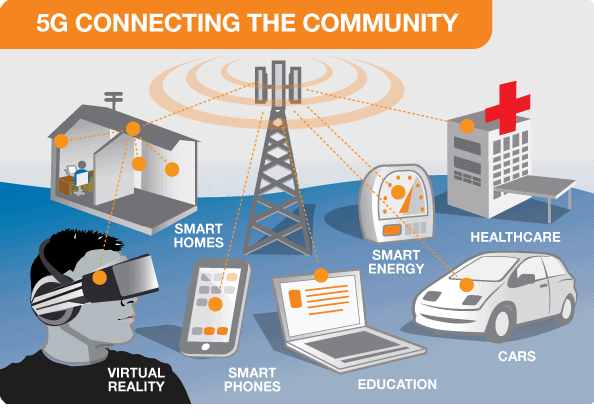
ΚΥΡΙΑΚΗ 9 ΙΟΥΝΙΟΥ 2024

ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ 5G, ΤΕΧΝΟΛΟΓΙΕΣ, ΕΦΑΡΜΟΓΕΣ ΚΑΙ ΒΑΣΙΚΟΙ ΔΕΙΚΤΕΣ ΑΠΟΔΟΣΗΣ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ΟΝΟΜΑ** | **ΕΠΩΝΥΜΟ** | **ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ** | **ΕΤΟΣ ΦΟΙΤΗΣΗ** | **EMAIL** |
| ΔΗΜΗΤΡΙΟΣ | ΤΣΑΜΠΡΑΣ | 1072467 | 5Ο | up1072467@ac.upatras.gr |
| ΑΘΑΝΑΣΙΟΣ-ΑΝΔΡΕΑΣ | ΤΣΟΥΛΗΣ | 1084591 | 4Ο | up1084591@ac.upatras.gr |

# ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Το δίκτυο 5G είναι το αμέσως επόμενο από το 4G LTE. Είναι σχεδιασμένο να παρέχει ακόμα μεγαλύτερες ταχύτητες και να ανταποκρίνεται στις δισεκατομμύρια συσκευές, οι οποίες θα μπορούν να συνδέονται και μεταξύ τους.



Εικόνα 1: Οι δυνατότητες του 5G

Όπως γίνεται αντιληπτό, το δίκτυο 5G θα φέρει σημαντικές αλλαγές στην κοινωνία. Επειδή ακριβώς αυτές οι αλλαγές είναι υψίστης σημασίας, τόσο σημαντική είναι και η ασφάλεια του δικτύου. Η ύπαρξη ανωμαλιών στο δίκτυο μπορεί να φέρει σημαντικές συνέπειες, όχι μόνο για το άτομο, αλλά και για όλο τον κόσμο.

Γενικότερα, με τον όρο **ανωμαλία (anomaly) σε ένα δίκτυο**, αναφερόμαστε σε τιμές ή γεγονότα που αποκλίνουν σημαντικά από την κανονική τάση των δεδομένων. Οι ανωμαλίες μπορεί να είναι δύο τύπων:

* **Aκριβείς (punctual)**: Οι ανωμαλίες σημείων (point anomalies) είναι μεμονωμένες τιμές που διαφέρουν πολύ από άλλες τιμές της χρονοσειράς.
* **Συλλογικές (collective)**: Οι συλλογικές ανωμαλίες είναι ομάδες τιμών που διαφέρουν από τις υπόλοιπες χρονοσειρές.

Επίσης, οι λόγοι που συμβάλλουν στην εμφάνιση ανωμαλιών μπορεί να είναι:

* Λάθος υπολογισμοί.
* Αλλαγές στις υποδομές.
* Κυβερνοεπιθέσεις (Ο πιο επικίνδυνος λόγος).

**Οι χρονοσειρές** είναι δεδομένα που καταγράφουν την τιμή μιας ή περισσότερων μεταβλητών σε διαφορετικά χρονικά σημεία. Οι χρονοσειρές εξαρτώνται από το χρόνο, δηλαδή η σειρά και το εύρος των δεδομένων είναι σχετικά και δεν μπορούν να αγνοηθούν ή να αλλάξουν. Συνήθως, τα δεδομένα αυτά είναι χωρισμένα σε ίσα χρονικά διαστήματα (είτε ανά ώρα, είτε ανά λεπτό κοκ). Η φύση αυτών των δεδομένων, τα καθιστά πολύ χρήσιμα για την μελέτη καταστάσεων που αλλάζουν συνέχεια με τον χρόνο, όπως η κυκλοφοριακή κίνηση, ο καιρός κοκ.

Μια χρονοσειρά διαθέτει τρία βασικά συστατικά:

1. Την ημερομηνία: Συμβολίζει το πότε μετρήθηκε η συγκεκριμένη μεταβλητή.
2. Την ώρα: Ομοίως με την ημερομηνία.
3. Τα χαρακτηριστικά: Οι μεταβλητές που θέλουμε να αναλύσουμε.

Ο εντοπισμός ανωμαλιών είναι δύσκολος. Ο εντοπισμός ανωμαλιών σε χρονοσειρές (time series data) είναι ακόμη πιο δύσκολος, καθώς προσθέτει ένα επιπλέον επίπεδο πολυπλοκότητας, λόγω της δομής των δεδομένων.

Ειδικότερα σε ένα 5G δίκτυο, είναι επιτακτική ανάγκη η ανίχνευση ανωμαλιών (fault detection) και ακόμα περισσότερο σε time series δεδομένα (Time-Series Anomaly Detection). Το 5G δίκτυο μπορεί να είναι πιο εξελιγμένο σε σχέση με τους προκατόχους του, αλλά τόσο εξελιγμένοι είναι και οι κίνδυνοι που το διέπουν. Μερικά παραδείγματα εξ αυτών είναι:

* Οι αυξημένοι κίνδυνοι στον κυβερνοχώρο λόγω της ύπαρξης του Διαδικτύου των Πραγμάτων (Internet of Things).
* Η αυξημένη χρήση εργαλείων εικονικής πραγματικότητας.

**Η ανίχνευση ανωμαλιών** αναφέρεται στον εντοπισμό οποιασδήποτε ανωμαλίας ή προβλήματος που επηρεάζει τη λειτουργία του δικτύου, ενώ **η ανίχνευση ανωμαλιών σε χρονοσειρές** εστιάζει στην ανίχνευση προβλημάτων σε χρονοσειρές. Στοχεύει στον εντοπισμό αποκλίσεων από την αναμενόμενη συμπεριφορά του δικτύου.

Για να γίνει πιο κατανοητός ο ορισμός της ανίχνευσης ανωμαλιών στο δίκτυο, παρατίθεται το εξής παράδειγμα:

Στην Εικόνα 2, καταγράφεται ο αριθμός επισκεπτών σε ένα website. Οι κόκκινες τελίτσες είναι οι ακριβείς ανωμαλίες, ενώ οι μαύρες είναι οι συλλογικές που έχουν ανιχνευθεί.

Εικόνα που περιέχει γραφικός χαρακτήρας, γραμματοσειρά, γραμμή, καλλιγραφία

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 2: Προσδιορισμός ανωμαλιών

Η ανίχνευση ανωμαλιών επιτυγχάνεται με την υλοποίηση μοντέλων μηχανικής μάθησης. Ουσιαστικά, το μοντέλο “εκπαιδεύεται” για να εντοπίζει ανωμαλίες.

Η μηχανική μάθηση χρησιμοποιεί ένα σύνολο δεδομένων για να κάνει πιθανοτικές προβλέψεις. Μια ομάδα δειγμάτων δεδομένων, που αναφέρεται ως σύνολο εκπαίδευσης, χρησιμοποιείται για την εκπαίδευση του αλγόριθμου. Στη συνέχεια, γίνονται προβλέψεις για μια διαφορετική ομάδα δειγμάτων που αναφέρεται ως σύνολο δοκιμών. Σε ορισμένες εφαρμογές το σετ εκπαίδευσης φέρει **ετικέτα (label).**

Οι κατηγορίες που χωρίζεται η εκπαίδευση του μοντέλου είναι οι εξής:

1. **Supervised learning**: Χρησιμοποιεί datasets τα οποία φέρουν ετικέτακαι εκπαιδεύει το μοντέλο, ώστε να κατηγοριοποιεί τα δεδομένα και να προβλέπει τα αποτελέσματα.
2. **Unsupervised learning:**  Χρησιμοποιεί datasets τα οποία  δεν φέρουν ετικέτακαι εκπαιδεύει το μοντέλο, ώστε να ομαδοποιήσει datasets, κάνοντας μείωση κατά διαστήματα και ανακαλύπτοντας “hidden patterns” χωρίς την ανθρώπινη παρέμβαση. Έτσι, εντοπίζει ανωμαλίες σε ένα dataset χρησιμοποιώντας αυτά τα hidden patterns.
3. **Semi-Supervised learning:** Συνδυάζει μια μικρή ποσότητα από labeled datasets με μια μεγάλη ποσότητα από unlabeled κατά τη διάρκεια της εκπαίδευσης του μοντέλου. Στοχεύει στην αντιμετώπιση του προβλήματος περιορισμού του συνόλου δεδομένων που φέρουν ετικέτα προς εκπαίδευση.
4. **Self-Supervised learning**: Το μοντέλο εκπαιδεύεται μόνο του για να μάθει ένα κομμάτι των δεδομένων εισόδου από ένα άλλο κομμάτι εισόδου. Το unsupervised πρόβλημα μετατρέπεται σε supervised, παράγοντας αυτόματα τις ετικέτες.

Στα δίκτυα 5G, το fault detection και το Time-Series Anomaly Detection μπορεί να περιλαμβάνουν την ανίχνευση προβλημάτων στον εξοπλισμό, στις συνδέσεις, στο λογισμικό ή σε άλλες κρίσιμες υποδομές. Μεγάλη είναι η σημασία τους για την:

* Αξιοπιστία Υπηρεσιών: Η ύπαρξη ανωμαλιών στο δίκτυο καθιστά πολλές υπηρεσίες που το χρησιμοποιούν αναξιόπιστες και, σε χειρότερη περίπτωση, επικίνδυνες. Χαρακτηριστικό παράδειγμα αποτελούν οι ιατρικές συσκευές που λειτουργούν με 5G δίκτυο και είναι απαραίτητες για την θεραπεία του ασθενούς.
* Μείωση Κόστους: Για να αντιμετωπιστεί οποιαδήποτε ανωμαλία, εκτός από πόρους, χρειάζεται και χρήμα το οποίο αρκετές φορές είναι αρκετό.
* Ασφάλεια Δικτύου: Είναι πασιφανές ότι το δίκτυο δεν είναι ασφαλές όταν υπάρχουν ανωμαλίες σε αυτό και ελλοχεύει ένας από τους μεγαλύτερους κινδύνους, που είναι η διαρροή των προσωπικών δεδομένων.
* Βελτιστοποίηση Απόδοσης: Η ανίχνευση των ανωμαλιών σε ένα δίκτυο πολλές φορές αποκαλύπτει ευπάθειες του ίδιου του δικτύου, με αποτέλεσμα να είναι εφικτή η βελτιστοποίηση του.

# ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ

Τα δεδομένα που είχαμε να επεξεργαστούμε αποτελούταν από multivariate time series δεδομένα. Κάθε γραμμή αποτελούταν από ένα **δείγμα (sample)** και κάθε στήλη αντιπροσωπεύει μια μεταβλητή (38 στο σύνολο) για κάθε χρονική στιγμή.

**Multivariate** σημαίνει πρωτίστως ότι τα δεδομένα αντιστοιχούν σε περισσότερες από μια μεταβλητές. Επίσης, κάθε μεταβλητή δεν εξαρτάται μόνο από την παρελθοντική της τιμή, αλλά και από κάποια άλλη μεταβλητή.

Κάθε σύνολο δεδομένων περιείχε τα εξής στοιχεία:

* Train data: Το σύνολο των δεδομένων που εισάγουμε ώστε να εκπαιδευτεί το μοντέλο και δεν περιέχει ανωμαλίες.
* Test data: Το σύνολο των δεδομένων που περιέχει ανωμαλίες.
* Test labels: Περιέχει τις τιμές **μηδέν (False)** και **ένα (True)** για κάθε δείγμα. Η τιμή μηδέν δείχνει ότι το δείγμα δεν περιέχει κάποια ανωμαλία, ενώ η τιμή ένα δείχνει ότι υπάρχει κάπου ανωμαλία στο δείγμα. Με αυτά ελέγχουμε αν βρίσκει το μοντέλο σωστά τις ανωμαλίες από τα test data.

Για το συγκεκριμένο μοντέλο, χρησιμοποιήθηκε το σύνολο machine-1-1 από τον σύνδεσμο <https://www.kaggle.com/datasets/mgusat/smd-onmiad/data>

# ΕΚΠΑΙΔΕΥΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ

Το μοντέλο που χρησιμοποιήθηκε για την κωδικοποίηση είναι ένα **Συνελικτικό Νευρωνικό Δίκτυο Αυτοκωδικοποιητή** (CNN Deep Autoencoder). Ας δούμε τι σημαίνει ξεχωριστά κάθε όρος:

**Νευρωνικό δίκτυο:**

Είδος αλγορίθμου μηχανικής μάθησης που διδάσκει στους υπολογιστές να επεξεργάζονται δεδομένα, όπως κάνει ο ανθρώπινος εγκέφαλος. Αποτελούνται από **επίπεδα (layers)** διασυνδεδεμένων κόμβων, ή τεχνητών νευρώνων, που επεξεργάζονται διάφορες πληροφορίες και εξάγουν ένα αποτέλεσμα με βάση τα δεδομένα εκπαίδευσης. Τα επίπεδα που περιέχει ένα νευρωνικό δίκτυο είναι:

* **Επίπεδο εισόδου (input layer)**: Πρόκειται για το πρώτο επίπεδο σε ένα νευρωνικό δίκτυο που λαμβάνει τα αρχικά δεδομένα εισόδου. Οι κόμβοι εισόδου επεξεργάζονται τα δεδομένα αυτά, τα αναλύουν και τα μεταβιβάζουν στο επόμενο επίπεδο.
* **Κρυμμένο επίπεδο (hidden layer)**: Τα κρυμμένα στρώματα αποτελούν τα ενδιάμεσα επίπεδα ανάμεσα στο επίπεδο εισόδου και εξόδου και επιτελούν το μεγαλύτερο μέρος του υπολογισμού. Ενδέχεται να υπάρχουν πολλά κρυμμένα επίπεδα σε ένα νευρωνικό δίκτυο.
* **Επίπεδο εξόδου (output layer)**: Το επίπεδο εξόδου είναι το τελικό επίπεδο σε ένα νευρωνικό δίκτυο και οδηγεί στην έξοδο του δικτύου. Ο αριθμός των νευρώνων στο επίπεδο αυτό εξαρτάται από το εκάστοτε πρόβλημα που λύνεται.

Εικόνα που περιέχει γραμμή, διάγραμμα, κύκλος

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 3: Τα επίπεδα ενός νευρωνικoύ δικτύου

**Συνελικτικό:**

Χρησιμοποιεί την πράξη συνέλιξης, η οποία είναι μια μαθηματική πράξη που ενώνει δύο σετ πληροφοριών. Η πιο ευρεία χρήση τους είναι η αναγνώριση και ταξινόμηση εικόνων.

**Αυτοκωδικοποιητής (Autoencoder):**

Χρησιμοποιείται για την εκμάθηση αποτελεσματικών κωδικοποιήσεων με Unsupervised learning σε νευρωνικά δίκτυα. Αποτελείται από έναν κωδικοποιητή (encoder) και έναν αποκωδικοποιητή (decoder).

Εικόνα που περιέχει γραμμή, διάγραμμα, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 4: Η δομή ενός Autoencoder

Η υλοποίηση του Autoencoder έγινε με 4 <<συμμετρικά>> επίπεδα (8 στο σύνολο), όπως φαίνεται στην Εικόνα 5 παρακάτω. Στόχος ήταν το μοντέλο να μάθει πως είναι η κανονική χρήση του server. Όποια ενέργεια δεν αντιστοιχούσε σε κανονική χρήση, την εντόπιζε ως ανωμαλία. Επίσης, θέλουμε η έξοδος του autoencoder να είναι , ιδανικά, ίδια με την είσοδο.

A diagram of a number of rectangular objects

Description automatically generated

Εικόνα 5: Η αρχιτεκτονική του CNN autoencoder

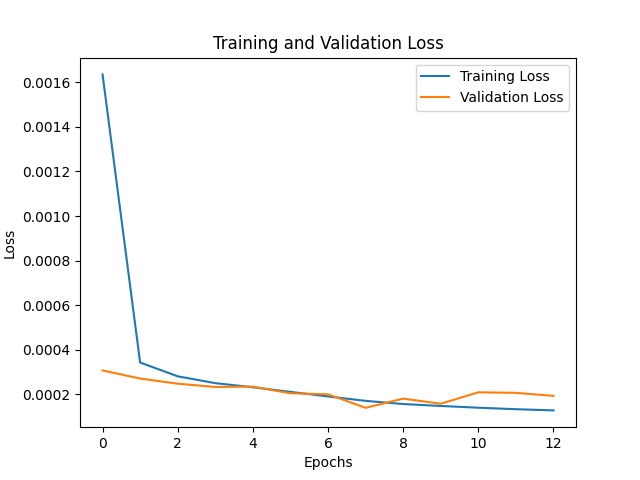
**Επίπεδο Α**: Σε αυτό το επίπεδο, το κομμάτι του κωδικοποιητή δέχεται το input, που στη δική μας περίπτωση είναι το sample εύρους 38 μεταβλητών.

**Επίπεδο Β:** Είναι το 1ο επίπεδο συνέλιξης. Η συνέλιξη κάνει το εξής: Από το Input παίρνει κάθε φορά ένα-ένα τα κομμάτια του ανά γραμμή και ψάχνει να βρει **πρότυπα (patterns)** και **χαρακτηριστικά (features)**. Επειδή είναι το 1ο επίπεδο, τα πρότυπα ακολουθούν μια σχετικά απλή μορφή **χαμηλού επιπέδου (low feature).**

**Επίπεδο C:** Είναι το 1ο επίπεδο **dropout**. Αυτό χρησιμοποιείται για να γίνει corruptτο input. Corrupt σημαίνει ότι κάποιοι νευρώνες του δικτύου αγνοούνται τυχαία κατά τη διάρκεια της εκπαίδευσης (they are dropped out). Έτσι, οι υπόλοιποι νευρώνες πρέπει να αναλάβουν την δουλειά των χαμένων και αυτό είναι πιθανόν να οδηγήσει στην αποφυγή του **overfitting**, δηλαδή στην συμπεριφορά κατά την οποία το μοντέλο δίνει ακριβείς προβλέψεις για ένα συγκεκριμένο σύνολο δεδομένων, αλλά όχι για καινούρια δεδομένα.

**Επίπεδο D:** Είναι το 2ο επίπεδο συνέλιξης. Σε αυτό το επίπεδο, τα πρότυπα που εντόπισε το Επίπεδο Α, επεξεργάζονται περισσότερο και εντοπίζονται πιο σύνθετα χαρακτηριστικά.

**Τα επίπεδα ΑΤ, , ΒΤ , CΤ και DΤ** αποτελούν τις ανάστροφες διαδικασίες των επιπέδων αντίστοιχα όπως αναφέρονται τα ονόματα τους.



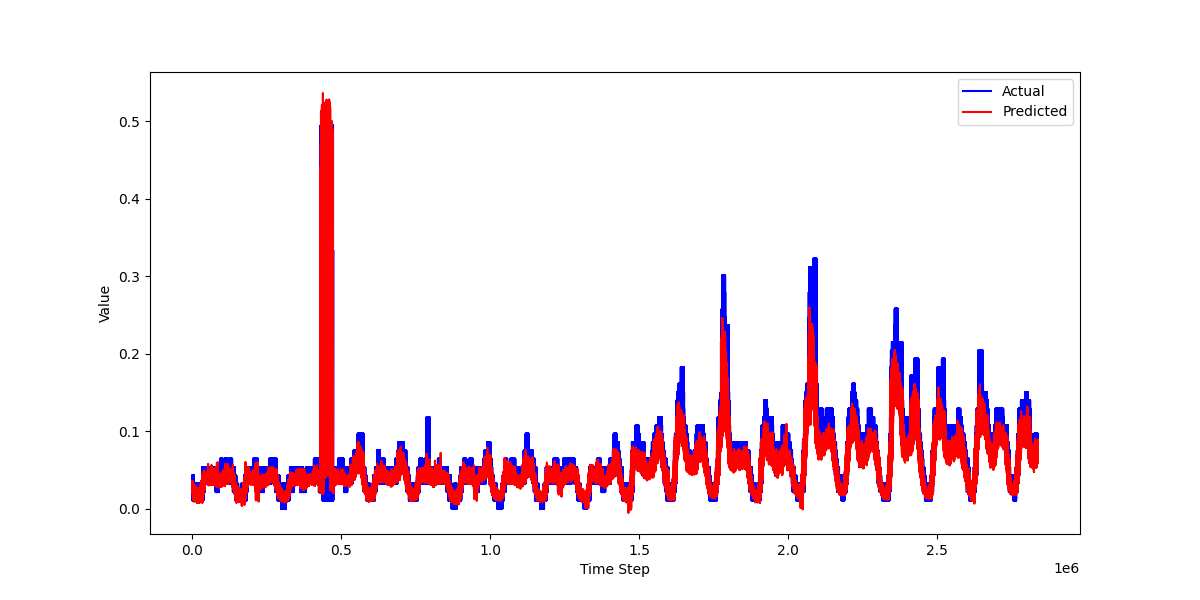
Εικόνα 6: Απώλειες

Παρατηρούμε ότι με την εκπαίδευση τα αντίστοιχα Training και Validation Losses, «πέφτουν» αισθητά, ενώ για το μεγαλύτερο κομμάτι το Validation Loss είναι κάτω (ή πολύ κοντά στο Training). Τα σφάλματα φαίνεται να είναι πολύ μικρά γιατί χρησιμοποιούμε το **τετράγωνο** το μέσου σφάλματος (MSE). Το εκπαιδευμένο μοντέλο με τα εξής χαρακτηριστικά είναι το anomaly\_complete\_v0\_3.keras.

# PREDICTIONS

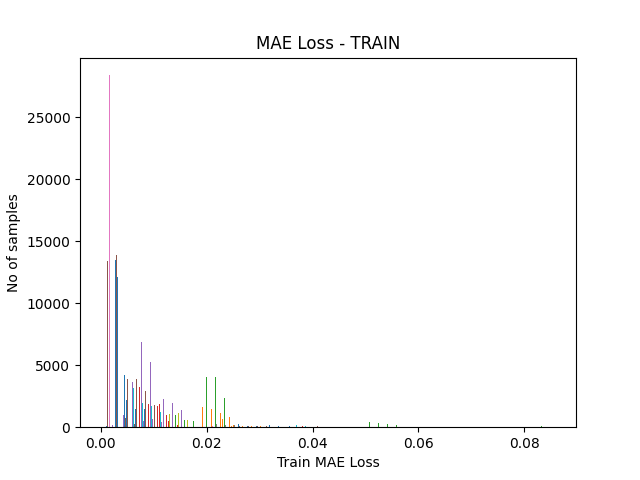
**TRAIN PREDICTIONS:**

Με τον όρο «predictions» αναφερόμαστε στην έξοδο του ήδη εκπαιδευμένου μοντέλου, όταν στην είσοδο του δεχθεί κάποια δεδομένα. Αφού έχουμε υλοποιήσει ένα autoencoder, θα θέλαμε η έξοδος του να είναι όσο το δυνατό, ίδια με την είσοδο. Η διαδικασία όμως που ακολουθεί το μοντέλο έχει ως αποτέλεσμα ένα κομμάτι των δεδομένων να χάνεται κατά την συμπίεση (και αποσυμπίεση). Στο παρακάτω γραφικό φαίνεται το prediction του 1ου feature με κόκκινο. Παρατηρήστε ότι διαφέρει ελάχιστα από το αρχικό μπλε.



Εικόνα 7: Prediction του 1ου feature

Αυτό το σφάλμα (MAE) που οφείλεται στην διαδικασία που αναφέραμε το βλέπουμε στο «train mae loss» και δεν αντιστοιχεί σε ανωμαλία (το train θεωρητικά δεν περιέχει ανωμαλίες). Με άλλα λόγια είναι το σφάλμα το οποίο θα είχε οποιοδήποτε φυσιολογικό δεδομένο βάζαμε στην είσοδο. Παρουσιάζουμε τα Losses για το Train:



Εικόνα 8: Μέσο τετραγωνικό σφάλμα

**TEST PREDICTIONS:**

Το test set περιέχει μεταξύ φυσιολογικών δεδομένων και ανωμαλίες. Για να μπορέσουμε να εντοπίσουμε τα σφάλματα τα οποία προέρχονται από τις ανωμαλίες, θεωρούμε ότι θα παρεκκλίνουν από τα «φυσιολογικά» σφάλματα. Για να κάνουμε αυτή την κατηγοριοποίηση, θέτουμε ένα threshold στην τιμή των σφαλμάτων.

threshold = np.percentile(train\_mae\_loss, 89.5) + 0.085

# ΑΝΙΧΝΕΥΣΗ ΑΝΩΜΑΛΙΩΝ

Το μοντέλο στην συνέχεια φτιάχνει έναν Boolean πίνακα (παρόμοιο με το test\_lables) στο οποίο έχει κατηγοριοποιήσει αν τα αντίστοιχα δείγματα είναι ή όχι ανωμαλίες. Ακολουθεί confusion matrix μετά την ανίχνευση και γράφημα τύπου βαρών:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

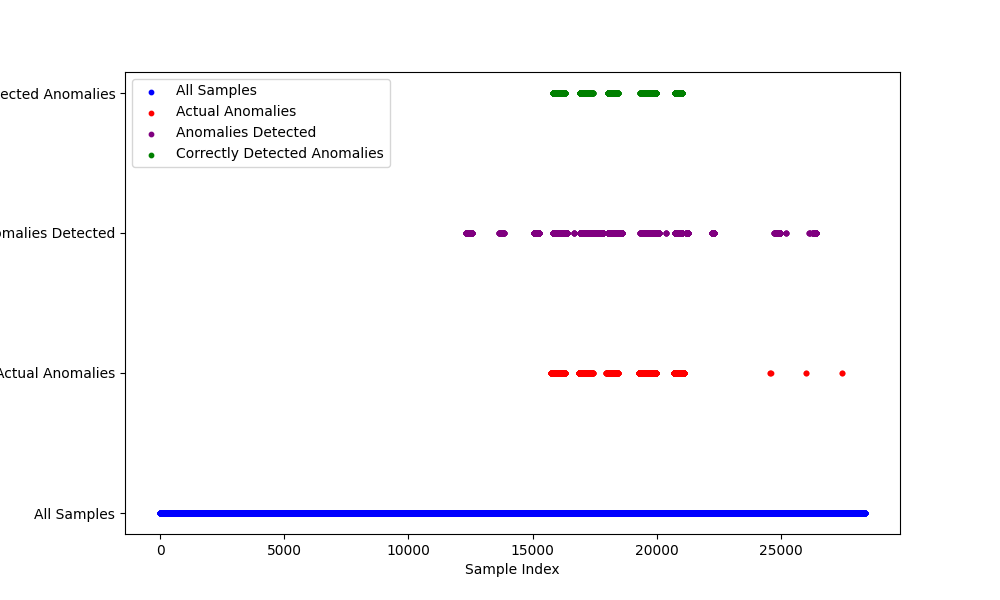
Εικόνα 9: Confusion Matrix

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, ορθογώνιο παραλληλόγραμμο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα 10: Ανίχνευση των ανωμαλιών

Καθώς μιλάμε για ανίχνευση ανωμαλιών σε χρονοσειρές, θεωρούμε ότι έχει σημασία να δούμε και την **κατανομή των ανωμαλιών στον χρόνο** στην οποία διακρίνεται η χρονική τοπικότητα των ανωμαλιών. Ακολουθεί γράφημα στο οποίο παρουσιάζονται τα σφάλματα στον χρόνο.



Εικόνα 11: Κατανομή των ανωμαλιών

**Σχολιασμός:** Όπως μπορούμε να διακρίνουμε, υπάρχουν πυκνές συγκεντρώσεις (συστάδες) ανωμαλιών στο test (πορτοκαλί) και φαίνεται να είναι συγκεντρωμένες όλες εκεί. Οι ανωμαλίες που βρήκε το μοντέλο θα μπορούσαμε να πούμε ότι ακολουθούν αρκετά την **τοπικότητα** των ανωμαλιών, καθώς διακρίνουμε αντίστοιχη πυκνή συγκέντρωση των ανωμαλιών που βρήκε το σύστημα (actual detected anomalies - μωβ) πιο πάνω. Με πράσινο παρουσιάζονται τα true positives, τα δείγματα τα οποία δηλαδή το μοντέλο επιτυχώς θεώρησε ανωμαλίες.

Αν δούμε το σύνολο των false negatives (1640), είναι σχετικά ψηλό. Aυτό επηρεάζει το precision, θεωρούμε όμως εμείς ότι το μοντέλο είναι σχετικά precise, καθώς τα false positives βρίσκονται κοντά στα true positives.

# ΜΕΤΡΙΚΕΣ

Για την αξιολόγηση των μοντέλων μηχανικής μάθησης και ειδικότερα των μοντέλων ταξινόμησης, χρησιμοποιούμε μετρικές όπως το **Precision, Recall, F1 score**.

**PRECISION:** Πόσα από τα δείγματα ανωμαλιών που εντόπισε το μοντέλο είναι πραγματικά ανωμαλίες (true positives).

Ορίζεται από τον λόγο:

Ένας υψηλός δείκτης *ακρίβειας* (precision), αντιστοιχεί σε χαμηλό ποσοστό false positives, γεγονός που μπορεί να είναι σημαντικό σε συστήματα πραγματικού χρόνου οπού η αντιμετώπιση των ανωμαλιών απαιτεί πόρους και γρήγορη αντιμετώπιση. *(Δεν προσπαθούμε να βρούμε όλες τις ανωμαλίες, αλλά θέλουμε να είμαστε σίγουροι για αυτές που θα βρούμε.)* Στα θετικά είναι η ελαχιστοποίηση των False Positives, ενώ στα αρνητικά είναι το ότι δεν δίνει σημασία στα False Negatives.

**RECALL:** Ποσά από τα δείγματα που είναι ανωμαλίες κατάφερε να ανιχνεύσει το μοντέλο.

Ορίζεται από τον λόγο:

Ένας υψηλός δείκτης *ανάκλησης* (recall), αντιστοιχεί σε χαμηλό ποσοστό false negatives, μεγάλη βάση δίνεται σε εφαρμογές όπου απαιτείται η εύρεση όλων των ανωμαλιών (true positives) ακόμη και να έχουμε και μερικά false positives. Στα θετικά είναι η ελαχιστοποίηση των False Negatives, ενώ στα αρνητικά είναι το ότι δεν δίνει σημασία στα False Positives.

**F1 SCORE:** Η ισορροπία μεταξύ precision και recall, εκφράζει την γενικότερη αξιολόγηση του μοντέλου απέναντι στην ακρίβεια και την ευαισθησία.

Ορίζεται από τον λόγο: F1 score

Οι μετρικές του μοντέλου βρίσκονται στον παρακάτω πίνακα:

|  |  |
| --- | --- |
| **Precision** | 0.5492028587135789 |
| **Recall** | 0.7416481069042317 |
| **F1 Score** | 0.6310802274162982 |

# ΣΥΝΟΛΙΚΟΣ ΚΩΔΙΚΑΣ

import numpy as np

import pandas as pd

import keras

from keras import layers, Input

from keras.models import load\_model

import tensorflow as tf

# Visualisation:

from matplotlib import pyplot as plt

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

# Define the number of time steps

TIME\_STEPS = 100

# Change file root as required!

file\_train\_root = "train.txt"

file\_test\_root = "test.txt"

# make the dataframes

df\_train = pd.read\_csv(file\_train\_root, sep=',')

df\_test = pd.read\_csv(file\_test\_root, sep=',')

'''

# DEBUG: shape of the data

print(df\_train.shape)

print(df\_test.shape)

'''

# training sequences for use in the model.

def create\_sequences(values, time\_steps=TIME\_STEPS):

output = []

for i in range(len(values) - time\_steps + 1):

output.append(values[i: (i + time\_steps)])

return np.stack(output)

# make the sequences

x\_train = create\_sequences(df\_train.values)

x\_test = create\_sequences(df\_test.values)

'''

# DEBUG: shape of the data

#due to the window approach the data is reduced

print("Training input shape: ", x\_train.shape)

print("Test input shape: ", x\_test.shape)

'''

'''

# ========== Model ==========

model = keras.Sequential(

[

layers.Input(shape=(x\_train.shape[1], x\_train.shape[2])),

layers.Conv1D(

filters=32,

kernel\_size=7,

padding="same",

strides=1, # Adjusted the strides to be 1

activation="relu",

),

layers.Dropout(rate=0.2),

layers.Conv1D(

filters=16,

kernel\_size=7,

padding="same",

strides=1, # Adjusted the strides to be 1

activation="relu",

),

layers.Conv1DTranspose(

filters=16,

kernel\_size=7,

padding="same",

strides=1, # Adjusted the strides to be 1

activation="relu",

),

layers.Dropout(rate=0.2),

layers.Conv1DTranspose(

filters=32,

kernel\_size=7,

padding="same",

strides=1, # Adjusted the strides to be 1

activation="relu",

),

layers.Conv1DTranspose(filters=38, kernel\_size=7, padding="same"),

]

)

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001), loss="mse")

model.summary()

history = model.fit(

x\_train,

x\_train,

epochs=40,

batch\_size=64,

validation\_split=0.1,

callbacks=[

keras.callbacks.EarlyStopping(monitor="val\_loss", patience=5, mode="min")

],

)

# ========== Plots ==========

# Plotting the Training Loss:

plt.plot(history.history['loss'], label='Training Loss')

# Plotting the Validation Loss:

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

'''

# Load the model

model = load\_model("anomaly\_complete\_v0\_3.keras")

# ========== Predictions ==========

# Predict and MAE loss for the training data

train\_predictions = model.predict(x\_train)

train\_mae\_loss = np.mean(np.abs(train\_predictions - x\_train), axis=1)

# Predict and MAE loss for the test data

test\_predictions = model.predict(x\_test)

test\_mae\_loss = np.mean(np.abs(test\_predictions - x\_test), axis=1)

# Plot the MAE loss for the training data

plt.hist(train\_mae\_loss, bins=50)

plt.title('MAE Loss - TRAIN')

plt.xlabel('Train MAE Loss')

plt.ylabel('No of samples')

plt.show()

# ========== Anomaly Detection ==========

# Set the threshold

threshold = np.percentile(train\_mae\_loss, 89.5) + 0.085

# Detect anomalies in the test data (boolean multi-dimensional)

anomalies = test\_mae\_loss > threshold

# (boolean one-dimensional)

anomalous\_samples = np.any(anomalies, axis=1)

# Load the test\_labels.txt file

test\_labels = np.loadtxt('test\_labels.txt')

# Trim the first TIME\_STEPS-1 elements from test\_labels

test\_labels\_trimmed = test\_labels[TIME\_STEPS:]

'''

# DEBUG: shape of the data

# due to the window approach the data is reduced

# test\_labels should be trimmed

print("Shape of test\_labels: ", test\_labels.shape)

print("Shape of test\_labels\_trimmed: ", test\_labels\_trimmed.shape)

print("Shape of anomalous\_samples: ", anomalous\_samples.shape)

'''

#check

assert test\_labels\_trimmed.shape == anomalous\_samples.shape

# Compare

common\_values = np.logical\_and(anomalous\_samples, test\_labels\_trimmed)

# Calculate the sum for each one

anomalies\_detected = np.sum(anomalous\_samples)

actual\_anomalies = np.sum(test\_labels\_trimmed)

correctly\_detected\_anomalies = np.sum(common\_values)

print("Number of anomalies detected: ", anomalies\_detected)

print("Actual number of anomalies: ", actual\_anomalies)

print("Number of correctly detected anomalies: ", correctly\_detected\_anomalies)

# Plot a bar graph with these three values

labels = ['Anomalies Detected', 'Actual Anomalies', 'Correctly Detected Anomalies']

values = [anomalies\_detected, actual\_anomalies, correctly\_detected\_anomalies]

plt.bar(labels, values)

plt.title('Anomaly Detection Results')

plt.xlabel('Categories')

plt.ylabel('Number of Anomalies')

plt.show()

# ========== Confusion Matrix ==========

# Calculate the confusion matrix

cm = confusion\_matrix(test\_labels\_trimmed, anomalous\_samples)

# Display the confusion matrix

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=["Normal", "Anomaly"])

disp.plot(cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.show()

# ========== METRICS ==========

# Calculate the precision, recall, and F1 score

precision = correctly\_detected\_anomalies / anomalies\_detected

recall = correctly\_detected\_anomalies / actual\_anomalies

f1\_score = 2 \* precision \* recall / (precision + recall)

print("Precision: ", precision)

print("Recall: ", recall)

print("F1 Score: ", f1\_score)

# ========== VISUALISATION ==========

#-------------

# Plot the anomalies through time:

# Create an array for all samples

all\_samples = np.arange(len(anomalous\_samples))

# Create arrays for actual anomalies, detected anomalies, and correctly detected anomalies

actual\_anomalies\_indices = np.where(test\_labels\_trimmed == 1)[0]

detected\_anomalies\_indices = np.where(anomalous\_samples == True)[0]

correctly\_detected\_anomalies\_indices = np.where(common\_values == True)[0]

# Create a scatter plot

plt.figure(figsize=(10,6))

# Plot all samples

plt.scatter(all\_samples, [0]\*len(all\_samples), color='blue', s=10, label='All Samples')

# Plot actual anomalies

plt.scatter(actual\_anomalies\_indices, [1]\*len(actual\_anomalies\_indices), color='red', s=10, label='Actual Anomalies')

# Plot detected anomalies

plt.scatter(detected\_anomalies\_indices, [2]\*len(detected\_anomalies\_indices), color='purple', s=10, label='Anomalies Detected')

# Plot correctly detected anomalies

plt.scatter(correctly\_detected\_anomalies\_indices, [3]\*len(correctly\_detected\_anomalies\_indices), color='green', s=10, label='Correctly Detected Anomalies')

# Label the axes

plt.xlabel('Sample Index')

plt.yticks([0, 1, 2, 3], ['All Samples', 'Actual Anomalies', 'Anomalies Detected', 'Correctly Detected Anomalies'])

plt.ylabel('Category')

# Add a legend

plt.legend()

# Show the plot

plt.show()

#--------------

# plot the predicted var1

# Select the first variable from the actual and predicted training data

actual\_data = x\_train[:, :, 0]

predicted\_data = train\_predictions[:, :, 0]

# Flatten the actual\_data and predicted\_data arrays

actual\_data\_flat = actual\_data.flatten()

predicted\_data\_flat = predicted\_data.flatten()

# Plot the actual data

plt.figure(figsize=(12,6))

plt.plot(actual\_data\_flat, 'b', label='Actual')

# Plot the predicted data

plt.plot(predicted\_data\_flat, 'r', label='Predicted')

# Label the axes

plt.xlabel('Time Step')

plt.ylabel('Value')

# Add a legend

plt.legend()

# Show the plot

plt.show()

# ΒΙΒΛΙΟΓΡΑΦΙΑ/ΠΗΓΕΣ

* <https://victoriametrics.com/blog/victoriametrics-anomaly-detection-handbook-chapter-1/>
* <https://medium.com/aimonks/anomaly-detection-for-time-series-analysis-eeecd6282f53>
* <https://aktinovolia.gr/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CF%84%CE%BF-5g-%CF%80%CF%89%CF%82-%CE%BB%CE%B5%CE%B9%CF%84%CE%BF%CF%85%CF%81%CE%B3%CE%B5%CE%AF-%CF%84%CE%BF-5g>
* <https://bigblue.academy/gr/neuronika-diktua>
* <https://medium.com/@polanitzer/building-a-cnn-based-autoencoder-with-denoising-in-python-on-gray-scale-images-of-hand-drawn-digits-61131ec492e4>
* <https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/>