RELAZIONE PROGETTO MPRAI

PHM Data Challenge America 2023

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

CIVITARESE ANDREA
DI RENZO MICHELE
GISSI CARLO
PASQUINI GIOELE

Sommario

Indice delle figure

1 – Introduzione alla challenge

Tutto il nostro lavoro si basa su “PHM North America 2023”, una challenge ancora aperta indetta da PHM Society, la cui documentazione è consultabile sul seguente link.

Sono stati condotti esperimenti di formazione di pitting sugli ingranaggi su un cambio a una fase installato in un banco di prova per trasmissioni elettronicamente controllato. Il banco di prova include due motori servo Siemens da 45 kW. Uno dei motori può agire come motore di guida mentre l'altro può essere configurato come motore di carico. Il cambio testato è a una fase con ingranaggi a denti dritti e un rapporto di riduzione di velocità di 1.8:1. Un accelerometro triassiale è stato fissato sulla scatola del cambio vicino alla cassa del cuscinetto sul lato di uscita. Le coordinate X, Y, Z corrispondono rispettivamente a orizzontale, assiale e verticale.

Il compito consiste nello sviluppare una stima della gravità dei difetti utilizzando i dati forniti. I dati di addestramento includono misurazioni in condizioni operative variabili da uno stato sano e da sei livelli di difetti noti. I dati di test e di validazione contengono dati da undici livelli di salute diversi, nei quali la degradazione dei denti aumenta con l’aumentare del livello. Alcuni livelli di difetti e condizioni operative sono esclusi dai dati di addestramento per riflettere le condizioni del mondo reale in cui la raccolta dati potrebbe essere disponibile solo da un sottoinsieme della gamma completa di operazioni. I dati di addestramento sono raccolti da una serie di diverse condizioni operative sotto 15 diverse velocità di rotazione e 6 diversi livelli di coppia. Le condizioni operative dei dati di test e di validazione spaziano su 18 diverse velocità di rotazione e 6 diversi livelli di coppia. Un modello dovrebbe generalizzare su condizioni operative e livelli di difetti precedentemente non visti.

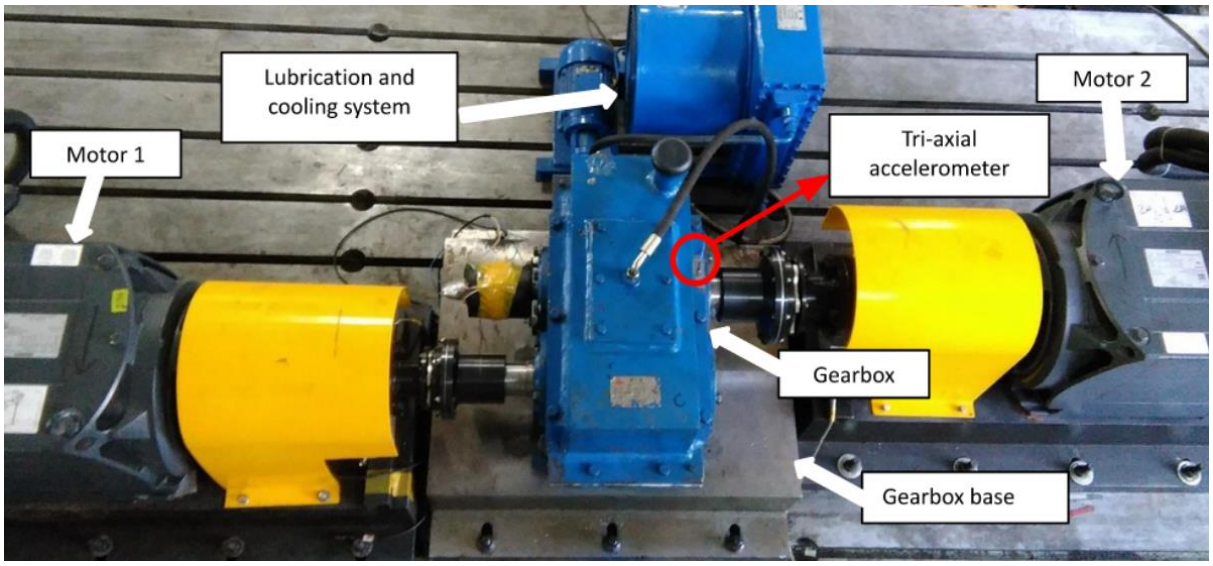


Figura 1: Rappresentazione del sistema di riferimento

2 – Preprocessing dei dati

I dati forniti dalla sfida sono organizzati in una cartella di addestramento che contiene diverse sottocartelle corrispondenti ai diversi livelli di pitting. All'interno di ciascuna sottocartella, sono presenti file "txt" che rappresentano gli esperimenti condotti sotto diverse condizioni operative per quel particolare stato di salute, dove il nome del file segue il formato "[velocità]V_[coppia]N.txt". Per ogni condizione operativa, sono state effettuate più repliche dell'esperimento e quindi sono disponibili più file.

I dati degli esperimenti sono stati raccolti con una frequenza di campionamento di 20.480 Hz, e la quantità di dati disponibili varia a seconda della modalità operativa:

- Per le basse velocità, per ogni conduzione dell'esperimento, sono disponibili finestre di 12 secondi;
- Per le medie velocità, per ogni conduzione dell'esperimento, sono disponibili finestre di 6 secondi;
- Per le alte velocità, per ogni conduzione dell'esperimento, sono disponibili finestre di 3 secondi.

Per ottenere dati utilizzabili per gli esperimenti, abbiamo creato inizialmente una nuova cartella di addestramento con tutti i livelli di pitting, dove i file sono stati tagliati in file da 60.000 righe; questo significa che per le alte velocità abbiamo considerato, all'incirca, tutti e 3 i secondi di campionamento presenti nei file, mentre per le basse e medie velocità abbiamo considerato solo i 3 secondi di campionamento centrali, escludendo il periodo iniziale.

La soluzione finale, invece, consiste nella creazione di una nuova cartella di addestramento con tutti i livelli di pitting, ma questa volta abbiamo preso l'intero file e lo abbiamo diviso in più file da 20.000 righe (circa 1 secondo di rilevamento). Questo ci ha consentito di ottenere un maggior numero di file per addestrare i nostri modelli e allo stesso tempo di estrarre più file da utilizzare nel nostro set di test, riducendo lo sbilanciamento tra i due. Risulta importante considerare che i dati utilizzati negli esperimenti sono relativi all'asse X dell'accelerometro in quanto, dopo una visualizzazione dei risultati dei modelli riportata nella figura 2, abbiamo notato che quest'asse era quello su cui discriminavano meglio le classi di guasto.

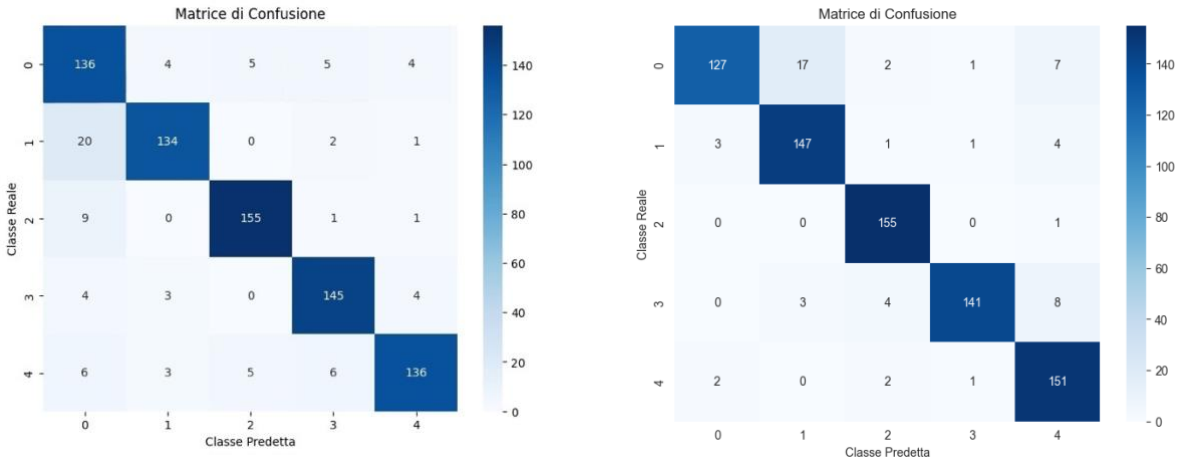


Figura 2: Risultati dei modelli CNN con asse dati degli assi Z (sinistra) e X (destra)

Dato che i dataset di test e validazione forniti all'interno della challenge non sono etichettati, per validare i nostri modelli con delle metriche, abbiamo lavorato con il solo training set, effettuandone

un hold-out per estrarne un training e un test set a nostra volta. Nello specifico abbiamo estratto da ogni condizione operativa e da ogni livello di pitting due file da inserire nel test set, selezionati casualmente, mentre il resto dei file è stato utilizzato ai fini del training.

Per quanto riguarda le classi, per simulare le condizioni della challenge, abbiamo nascosto ai nostri modelli i dati di alcune classi di train, anche se conosciute.

Sostanzialmente, il dataset utilizzato per addestrare e validare i nostri modelli è composto un dataset di test contenente i dati relativi a 7 classi di guasto (0, 1, 2, 3, 4, 6 e 8) e da un dataset di train contenente i dati relativi a 5 classi di guasto (0, 1, 2, 4 e 6). Queste 7 classi disponibili sono state da noi rietichettate da 0 a 6.

Nelle cartelle di test e validazione fornite dalla challenge, invece, i file non sono suddivisi per livelli di pitting e, quindi, non sono etichettati, abbiamo allora estratto da ogni file circa un secondo di rilevamento, corrispondente a 20.000 righe, per effettuarne la classificazione tramite il modello sviluppato esteso a tutte le classi della challenge.

3 – Primi test e tecnologie utilizzate

Di seguito illustreremo brevemente tutti gli approcci pensati e sviluppati, che ci hanno permesso di arrivare alla soluzione definitiva del nostro problema. Per sviluppare le nostre idee sono state utilizzate le seguenti tecnologie:

- **CNN (Convolutional Neural Network)**: Una rete neurale convoluzionale è un tipo di rete neurale artificiale, progettata principalmente per l'analisi di dati grid-like, come immagini. Utilizza operazioni di convoluzione per rilevare pattern locali nei dati di input e strati di pooling per ridurre la dimensione spaziale dell'input, mantenendo intatte le caratteristiche salienti. Le CNN sono ampiamente utilizzate in applicazioni di visione artificiale, riconoscimento di pattern e analisi dei dati spaziali.
- **SAE (Sparse Autoencoder)**: Un SAE è un tipo di rete neurale artificiale utilizzata per l'apprendimento non supervisionato di rappresentazioni efficienti dei dati. L'obiettivo principale di un autoencoder sparso è quello di apprendere una rappresentazione compatta dei dati di input, imponendo contemporaneamente una sparsità sulla rappresentazione stessa. Ciò significa che solo poche unità nascoste sono attive in ogni dato di input, rendendo la rappresentazione più dispersa e interpretabile introducendo un parametro aggiuntivo alla funzione di loss che permette di organizzare al meglio lo spazio latente di rappresentazione dei dati durante l'addestramento.
- **FFT (Fast Fourier Transform):** La FFT è un algoritmo utilizzato per calcolare la trasformata di Fourier discreta di un segnale o di una serie temporale. Trasformando un segnale dal dominio del tempo al dominio delle frequenze, l'FFT consente di analizzare le componenti di frequenza presenti nel segnale. È ampiamente utilizzato in applicazioni di elaborazione del segnale, analisi spettrale e compressione dei dati.
- **K-means**: Il K-means è un algoritmo di clustering utilizzato per raggruppare un insieme di dati in un numero predeterminato di cluster. L'algoritmo funziona iterativamente per assegnare ciascun dato al cluster più vicino, aggiornare i centroidi dei cluster e ripetere il processo fino a quando i centroidi convergono o fino a quando viene raggiunto un numero massimo di iterazioni. K-means è ampiamente utilizzato in applicazioni di analisi dei dati, segmentazione delle immagini e apprendimento non supervisionato.

3.1 – Solo CNN

Nel corso del nostro primo esperimento, abbiamo adottato come punto di riferimento uno degli articoli forniti dalla challenge [1], concentrandoci sulla costruzione di una Convolutional Neural Network (CNN) per il riconoscimento delle classi di guasto. In particolare, abbiamo lavorato con la prima suddivisione dei file, limitandoci a una porzione dei segnali registrati dall'accelerometro. La nostra architettura di rete è stata progettata per accettare in input file suddivisi in vettori con 60.000 valori e con un layer finale dimensionato per consentire anche la classificazione dei dati delle classi non presenti nel training set.

Durante la fase di addestramento, la rete ha mostrato prestazioni promettenti, evidenziando un'accuratezza di circa il 97%. Tuttavia, al momento del test utilizzando i dati del set di test estratto da noi, abbiamo riscontrato risultati deludenti. La rete ha dimostrato difficoltà nel classificare correttamente le classi di salute mai incontrate in precedenza. Le metriche ottenute durante il test hanno riportato una loss elevata e un'accuratezza approssimativamente del 50%.

Questi risultati indicano una mancanza di generalizzazione della rete, che non è in grado di estendere efficacemente la sua capacità di classificazione a nuove classi non viste durante l'addestramento. Ciò suggerisce la necessità di adottare ulteriori approcci non supervisionati che riescano a generalizzare meglio.

3.2 – FFT, Autoencoder e Cluster

In seguito, consultando un altro dei paper messia disposizione dalla challenge [2], abbiamo esplorato l'impiego della Trasformata di Fourier Veloce (FFT) per trasformare i dati nel dominio delle frequenze e l'utilizzo di uno sparse autoencoder per ridurre le dimensionalità dei dati. Dopo aver applicato la FFT considerando un secondo di campionamento, ossia 20.000 campioni facendo lo 0-padding per portare il numero dei campioni alla potenza di 2 più vicina (2^{15}, 32.768), abbiamo esaminato due approcci distinti:

- Estrazione dei 1000 valori di picco più grandi.
- Estrazione delle frequenze associate ai 1000 valori di picco calcolati in precedenza.

L'uso dell'autoencoder permette di ridurre drasticamente le dimensioni dei dati passando da una dimensione di 1000 a una dimensione di 50, rendendoli adatti all'applicazione del clustering mediante l'algoritmo K-means.

Il processo è stato sviluppato secondo i seguenti passaggi:

1. Raccolta dei dati per ogni secondo di tempo nel dataset originale, generando così file composti da 20.000 valori ciascuno.
2. Applicazione della FFT ai dati e selezione delle 1000 frequenze o ampiezze più significative.
3. Normalizzazione del vettore delle frequenze o ampiezze, sottraendo ad ogni componente la media dei componenti del vettore e dividendo tale differenza per la varianza del vettore.
4. Riduzione della dimensionalità tramite utilizzo del SAE.
5. Utilizzo dei vettori dello spazio latente del SAE come input all’algoritmo di clustering K-means.

Tuttavia, abbiamo riscontrato una criticità in questo approccio. I dati estratti nel dominio delle frequenze mostravano una somiglianza eccessiva tra loro, nonostante le diverse classi di guasto. Ciò ha complicato il processo di riduzione della dimensionalità dei dati da parte dell'autoencoder. Di conseguenza, il K-means ha mostrato difficoltà nel creare un numero di cluster adeguato rispetto

alle diverse classi di guasto, raggruppando la maggior parte dei dati in un unico cluster e considerando gli altri come outlier.

Avendo il dubbio che con la selezione di sole 1000 frequenze o ampiezze ci fosse troppa perdita di informazioni, abbiamo provato a tralasciare questa selezione dando in input ad un SAE, di dimensione più grande rispetto al precedente, il vettore di tutte le ampiezze risultanti dalla FFT dei segnali. Questo non ha portato i risultati sperati dato che abbiamo incontrato le stesse problematiche descritte nell'esperimento precedente, portandoci ad escludere l'utilizzo di uno sparse autoencoder per effettuare la riduzione della dimensionalità.

3.3 – CNN e Cluster con suddivisione tra le diverse modalità operative

Come indicato nel titolo del paragrafo, abbiamo ulteriormente raffinato la fase di preprocessing suddividendo il set di addestramento in due categorie: velocità alte (> 1000 rpm) e velocità basse (≤ 1000 rpm), senza considerare i valori di coppia. Questo è stato fatto con l'intenzione di facilitare la classificazione da parte della rete neurale convoluzionale (CNN) e successivamente dell'algoritmo di clustering.

L'approccio che abbiamo seguito consisteva nell'addestrare due CNN, una per i dati relativi ad alte velocità ed una per le basse velocità, utilizzando dati di input composti da 20.000 valori, al fine di classificare i dati in una delle 5 classi possibili. Abbiamo ottenuto un'accuratezza molto buona durante il training come anche per i precedenti addestramenti, circa il 95%.

Per il dataset di test, comprendente 7 classi possibili, abbiamo applicato un algoritmo di clustering K-means in coda alla CNN per ogni etichetta predetta.

L'algoritmo K-means riceveva in ingresso i vettori di 20.000 componenti che erano stati classificati dalla CNN con la stessa etichetta. Questo passaggio aveva lo scopo di separare i dati che all'interno di una stessa classe predetta risultassero essere significativamente distanti, al fine di individuare la possibile esistenza di ulteriori classi. Tuttavia, nonostante la logica di questo approccio fosse ragionevole, i risultati ottenuti sono stati di scarsa qualità. Questo è stato principalmente dovuto alla difficoltà dell'algoritmo di clustering nel separare i dati, causata dalla grande dimensionalità degli stessi (20.000 valori per ogni dato) e dalla loro prossimità in termini di distanza euclidea, nonostante le differenze nelle etichette di classe.

3.4 – CNN con soglie sul valore della confidenza

In un altro approccio, abbiamo tentato di utilizzare le valutazioni di confidenza fornite in output dalla CNN come criterio decisionale, questo poiché abbiamo ipotizzato che i dati appartenenti a classi non viste nel training set venissero classificati con valori di confidenza più bassi rispetto ai dati effettivamente appartenenti a quelle classi.

La rete neurale è stata sempre addestrata con dati di input composti da 20.000 valori e produce in output etichette per 5 classi, raggiungendo un'accuratezza elevata come nei tentativi precedenti. Per il dataset di test, abbiamo introdotto soglie minime di confidenza scelte in maniera euristica.

Si va a verificare che la confidenza massima di appartenenza ad una classe fosse maggiore di 0.7. In caso questa situazione non si verificasse, si controlla se le due confidenze maggiori restituite dalla CNN fossero riferite a due classi consecutive e differissero in valore assoluto di meno di un valore prefissato a 0.2. Se ciò si fosse verificato allora si sarebbe ipotizzato la presenza di una classe intermedia tra le due in questione.

Tuttavia, questo metodo è stato abbandonato poiché si è osservato che, esclusi circa 20 dati su 1600 con una confidenza inferiore a 0.35, la totalità dei restanti presentava una confidenza superiore a 0.99, rendendo impossibile l'individuazione di soglie ragionevoli e, di conseguenza, anche l'applicazione efficace di tale approccio.

4 – Descrizione del modello finale

A seguito dei test effettuati che non hanno portato a risultati soddisfacenti, il modello finale, che è anche quello che ha dato i risultati più promettenti, è il seguente, composto da un modello CNN con in coda diversi modelli Isolation Forest, come mostrato nella figura 4.

L'Isolation Forest, in particolare, è un algoritmo di machine learning per il rilevamento delle anomalie. L'algoritmo costruisce alberi di isolamento (isolation trees) in modo casuale, selezionando casualmente attributi e valori di soglia per separare i dati in ciascun nodo dell'albero.

I dati vengono suddivisi ricorsivamente fino a quando ogni punto è isolato in un singolo nodo foglia o fino a quando viene raggiunto un numero massimo di livelli dell'albero. Durante la costruzione degli alberi, vengono calcolati i percorsi di anomalia per ciascun dato.

I dati che richiedono meno divisioni per essere isolati sono considerati più anomali. Gli alberi di isolamento costruiti vengono quindi utilizzati per calcolare un punteggio di anomalia per ciascun dato, che può essere utilizzato per identificare le anomalie nel dataset.

Questa idea di modello deriva da una prova effettuata nella quale, tramite Isolation Forest, siamo andati a controllare quante anomalie questo rilevasse per ogni gruppo di dati classificati dalla CNN con una stessa etichetta. Abbiamo addestrato 5 modelli di Isolation Forest, ciascuno con i dati relativi ad una specifica classe (0, 1, 2, 4 e 5). A ciascun modello, sono stati fatti analizzare i dati di test (relativi alle classi 0, 1, 2, 3, 4, 5 e 6) la cui etichetta assegnata dalla CNN corrispondeva con quella dei dati di addestramento.

Come si può notare in figura 3, il maggior numero di anomalie viene rilevato nei dati etichettati come classe 2 e 5, che sono proprio le classi precedenti alle classi 3 e 6, i quali dati non sono presenti nel training set.

Figura 3: Anomalie rilevate per dati classificati con stessa etichetta (1: non anomalia, -1: anomalia)
Come si può notare, per i dati etichettati come classe 0,1 e 4 (a sinistra) che non sono subito precedenti a classi nascoste,

sono state rilevate molte meno anomalie rispetto ai dati etichettati come classe 2 e 5 (a destra), che sono subito precedenti alle classi 3 e 6, ossia quelle non presenti nel training set.

Da questa osservazione deriva l'dea di assegnare ai dati rilevati come anomalie l'etichetta della classe successiva a quella identificata dalla CNN; questo solo per le classi subito precedenti a classi mancanti nel training set.

Per una migliore comprensione, nella figura 4 è riportata graficamente la struttura del nostro modello.

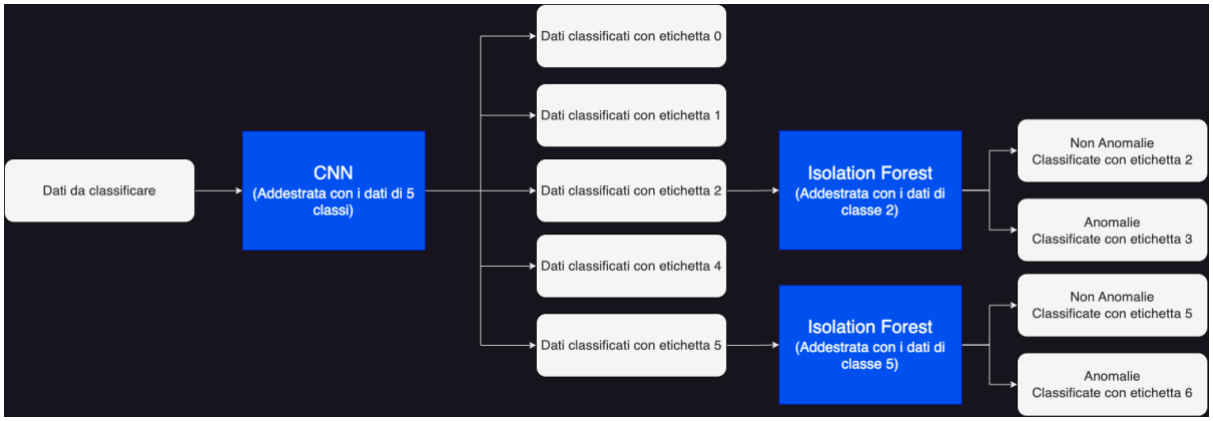


Figura 4: Workflow del nostro modello

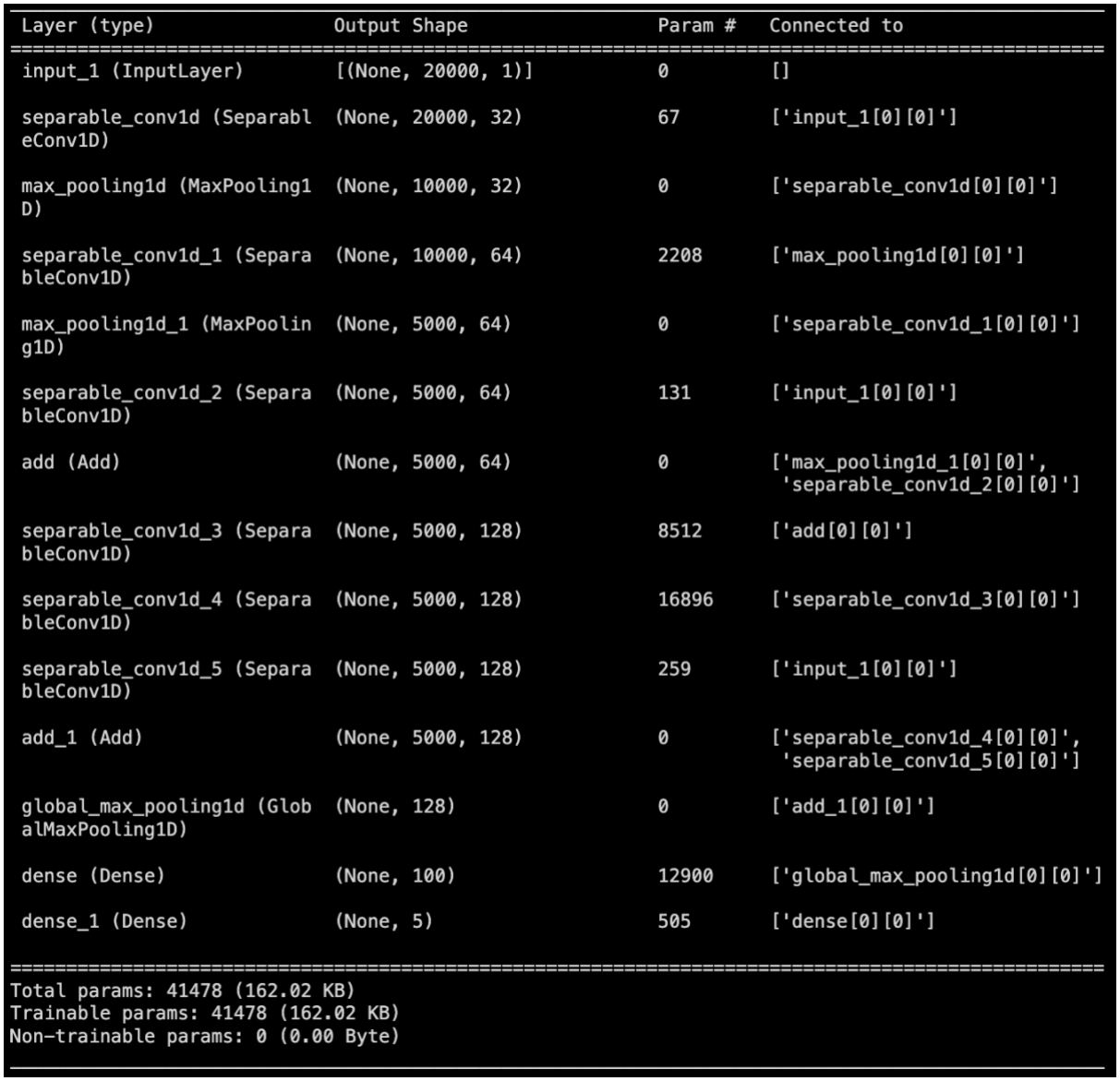
4.1 – Composizione del modello

Il modello è composto, nella prima parte, da una rete neurale convoluzionale alla quale vengono passati i dati delle vibrazioni da classificare. La CNN restituisce un vettore di probabilità di appartenenza alle classi di dimensione pari al numero di classi presenti nel training set. La struttura della CNN utilizzata è mostrata nella figura 5.

La rete prende in input un vettore di dimensione 20.000 contenente i dati grezzi dell'accelerometro, è composta da una serie di layer convoluzionali tutti con funzione di attivazione ReLu, intervallati da layer di max pooling. Sono, inoltre, presenti delle skip connections per riportare le informazioni in input anche ai layer più profondi della rete. Sono presenti, infine, due layer densi, di cui l'ultimo con funzione di attivazione SoftMax che restituisce un vettore di probabilità la cui somma delle componenti è 1.

La particolarità dei layer convoluzionali utilizzati è che questi utilizzano convoluzioni separabili spazialmente, ossia il kernel convoluzionale è diviso in filtri che operano lungo la singola feature map e filtri che operano in profondità sulle diverse feature map, permettendo, così, di risparmiare in termini di numero di parametri del modello.

Come si può notare, infatti, si tratta di un modello di rete neurale relativamente leggero e con pochi parametri rispetto alla dimensione del dato che può analizzare, è proprio questo il vantaggio delle CNN. Ciò consente un veloce addestramento e un'altrettanta veloce classificazione.



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 20000, 1)]	0	[]
separable_conv1d (SeparableConv1D)	(None, 20000, 32)	67	['input_1[0][0]']
max_pooling1d (MaxPooling1D)	(None, 10000, 32)	0	['separable_conv1d[0][0]']
separable_conv1d_1 (SeparableConv1D)	(None, 10000, 64)	2208	['max_pooling1d[0][0]']
max_pooling1d_1 (MaxPooling1D)	(None, 5000, 64)	0	['separable_conv1d_1[0][0]']
separable_conv1d_2 (SeparableConv1D)	(None, 5000, 64)	131	['input_1[0][0]']
add (Add)	(None, 5000, 64)	0	['max_pooling1d_1[0][0]', 'separable_conv1d_2[0][0]']
separable_conv1d_3 (SeparableConv1D)	(None, 5000, 128)	8512	['add[0][0]']
separable_conv1d_4 (SeparableConv1D)	(None, 5000, 128)	16896	['separable_conv1d_3[0][0]']
separable_conv1d_5 (SeparableConv1D)	(None, 5000, 128)	259	['input_1[0][0]']
add_1 (Add)	(None, 5000, 128)	0	['separable_conv1d_4[0][0]', 'separable_conv1d_5[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 128)	0	['add_1[0][0]']
dense (Dense)	(None, 100)	12900	['global_max_pooling1d[0][0]']
dense_1 (Dense)	(None, 5)	505	['dense[0][0]']

Total params: 41478 (162.02 KB)
Trainable params: 41478 (162.02 KB)
Non-trainable params: 0 (0.00 Byte)

Figura 5: Struttura della CNN utilizzata

In coda alla CNN sono posti tanti modelli di Isolation Forest (2 modelli addestrati con i dati delle classi 2 e 5) quanto il numero di classi presenti nel training set subito precedenti a classi nascoste (classi 3 e 6), che vanno a identificare anomalie tra i dati predetti con stessa etichetta.

L'algoritmo finale, dunque, risulta essere composto dai seguenti passi:

1. Il dato da classificare viene dato in pasto alla CNN. Al dato di input viene associata l'etichetta della classe predetta con probabilità di appartenenza più alta (per probabilità di appartenenza si intendono le probabilità restituite dalla funzione di attivazione SoftMax dell'ultimo layer della rete);
2. Se la classe predetta non corrisponde ad una delle classi che sono subito precedenti a classi nascoste, allora l'etichetta rimarrà quella predetta dalla CNN senza ulteriori verifiche.
3. Se la classe predetta corrisponde ad una delle classi che sono subito precedenti a classi nascoste, il dato viene dato in input al modello di Isolation Forest corrispondente a quella classe.

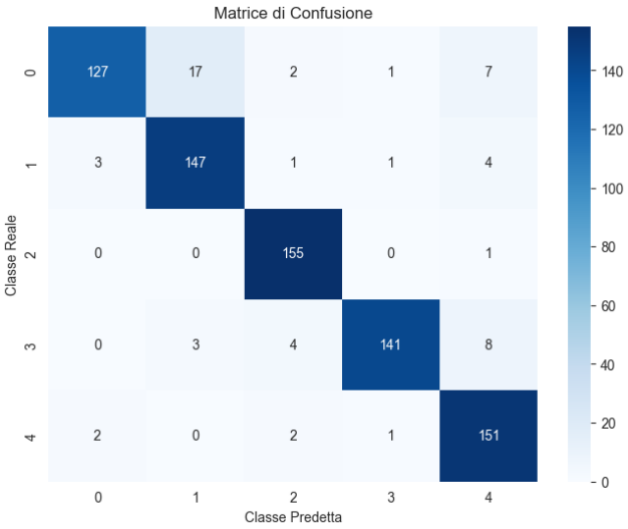
4. Se il dato è rilevato come anomalia, allora viene assegnata come etichetta quella della classe nascosta (ossia la classe successiva), altrimenti si lascia la classe predetta dalla CNN.

4.2 – Training del modello

La CNN è stata implementata con Python, più precisamente tramite la libreria Tensorflow, e addestrata con tutti i dati appartenenti alle classi presenti nel training set, l'addestramento è stato effettuato per 65 epoche, con un learning rate di 0,002, batch size 32 e ottimizzatore RMSProp con parametri di default. La lunghezza, in termini di epoche, dell'addestramento è stata scelta visualizzando l'andamento dell'accuratezza durante il training, a 65 epoche le prestazioni erano ampiamente sufficienti per il task da effettuare. Il tempo di addestramento è di circa 22 minuti o, più nello specifico, 20 secondi per epoca ed è stato effettuato con una GPU Nvidia RTX 2060.

Nella figura seguente sono riportati i risultati del test della CNN effettuato con i dati delle sole classi presenti anche nel training set (0, 1, 2, 4 e 5). Notare che le classi 4 e 5, presenti nel training set, per effettuare questa prova, sono state rietichettate rispettivamente come 3 e 4.

Questi risultati dimostrano come la CNN sia uno strumento molto efficace ed efficiente nella classificazione di dati temporali grezzi quando il training set contiene tutte le classi presenti anche nel test set.



	precision	recall	f1-score	support
0	0.96	0.82	0.89	154
1	0.88	0.94	0.91	156
2	0.95	0.99	0.97	156
3	0.98	0.90	0.94	156
4	0.88	0.97	0.92	156
micro avg	0.93	0.93	0.93	778
macro avg	0.93	0.93	0.93	778
weighted avg	0.93	0.93	0.93	778
samples avg	0.93	0.93	0.93	778

Figura 6: Risultati test CNN con dati delle classi presenti sia nel training set che nel test set

Sono stati poi addestrati due modelli di Isolation Forest con i dati di training delle classi 2 e 5; questi modelli serviranno a rilevare anomalie nei dati predetti rispettivamente con etichette 2 e 5 poiché sono le classi antecedenti alle classi 3 e 6 (quelle da noi artificialmente eliminate dal training set). I modelli di Isolation Forest sono stati implementati con la libreria SciKit Learn, anch'essa di Python e il loro addestramento ha richiesto all'incirca 150 millisecondi l'uno. La libreria utilizzata non sfrutta la GPU quindi l'addestramento è stato effettuato su CPU, nello specifico un Intel i9 11900.

4.3 – Modello per test con tutto il dataset della challenge

Come scritto precedentemente, nelle fasi precedenti è stato utilizzato solo il training set fornito dalla challenge, questo poiché è l'unico etichettato, e che abbiamo, a sua volta, diviso in training e test set in modo da poter validare le performance del nostro modello.

Una volta fatto ciò, abbiamo esteso l'approccio descritto sopra all'intero dataset della challenge per visualizzarne il comportamento. Il modello esteso è composto in maniera del tutto simile a quello utilizzato per il solo training set, a parte per alcune differenze.

La prima differenza sta nell'ultimo layer della CNN che è composto da 7 neuroni invece che 5, questo poiché nel training set totale sono presenti 7 classi. Il suo addestramento ha richiesto 33 minuti, a causa del maggior numero di dati di training, ed è stato effettuato con gli stessi iper-parametri descritti in precedenza.

La seconda differenza è nelle classi nascoste, che nel caso dei dataset dell'intera challenge sono le classi 5, 7, 9 e 10; proprio per questo i modelli di Isolation Forest utilizzati sono quelli addestrati con i dati delle classi 4, 6 e 8 che andranno a rilevare le anomalie nei dati etichettati con le rispettive classi di training. L'addestramento dei modelli di anomaly detection di questo approccio ha richiesto lo stesso tempo di quelli del modello utilizzato con il solo training set, questo poiché il numero di dati disponibili per ciascuna classe è molto simile e ogni modello di Isolation Forest è addestrato con i dati di una sola classe; quello che cambia è il numero di questi modelli che, in questo caso, sono 3 invece che 2.

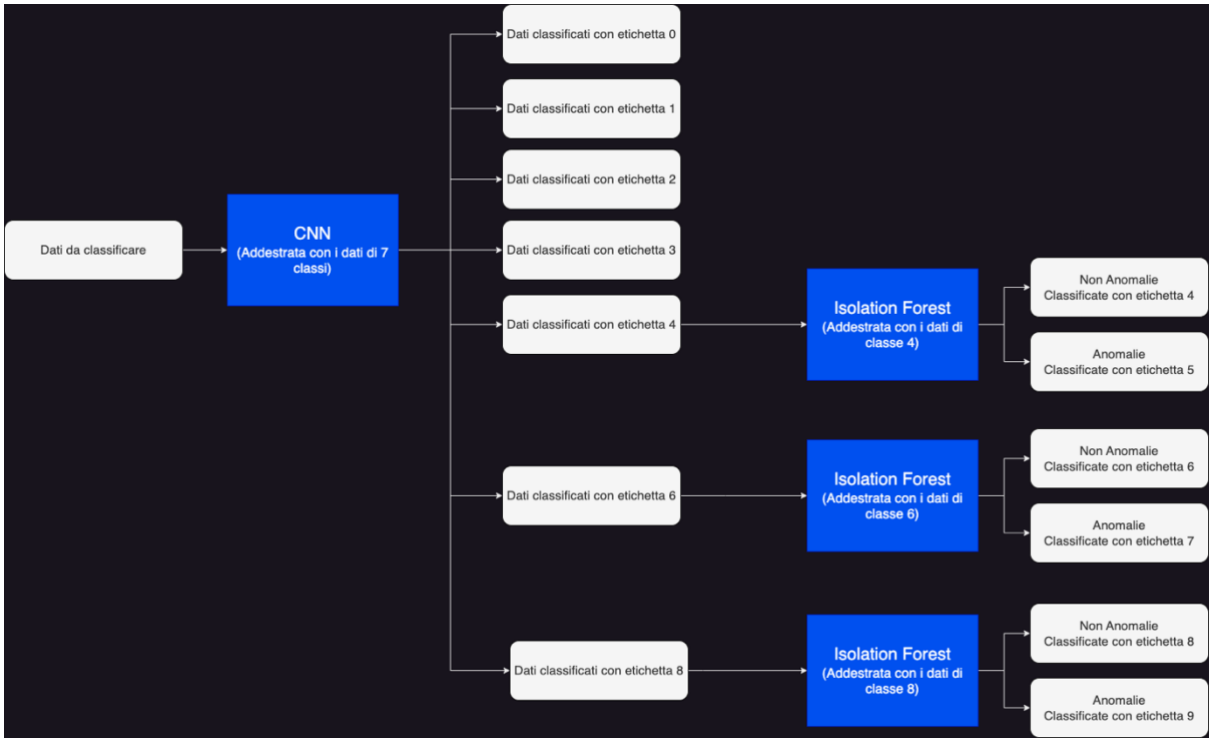


Figura 7: Workflow del nostro modello utilizzato con il dataset completo

Come si può intuire, con l'approccio da noi adottato non è possibile eseguire la differenziazione tra gli elementi delle classi 9 e 10, questo poiché i dati rilevati come anomalie nel nostro approccio vengono etichettati con la classe direttamente successiva a quella con cui l'algoritmo è stato addestrato. Abbiamo quindi accorpato in un'unica etichetta queste due classi di guasto, ossia l'etichetta 9.

5 – Risultati e conclusioni

Alla luce di tutte le considerazioni fatte in precedenza, in questo capitolo verranno riportati i risultati e le conclusioni relative alle diverse prove fatte.

5.1 – Risultati

Per quanto riguarda la prima prova effettuata, abbiamo potuto calcolare le metriche del nostro modello complessivo. Ciò è stato possibile grazie alla divisione fatta del dataset totale di training della challenge tramite hold-out.

Nella seguente immagine sono riportate le metriche del modello complessivo valutato sui dati di test generati artificialmente a partire dai dati originali di train.

	precision	recall	f1-score	support
0	0.93	0.82	0.88	154
1	0.71	0.94	0.81	156
2	0.64	0.87	0.74	156
3	0.20	0.06	0.09	156
4	0.56	0.90	0.69	156
5	0.64	0.85	0.73	156
6	0.23	0.05	0.08	156
micro avg	0.64	0.64	0.64	1090
macro avg	0.56	0.64	0.57	1090
weighted avg	0.56	0.64	0.57	1090
samples avg	0.64	0.64	0.64	1090

Figura 8: Metriche modello complessivo

Per quanto riguarda le metriche del modello addestrato con i dati di tutte le classi di training della challenge (classi 0, 1, 2, 3, 4, 6 e 8) e testato con i dati di tutte le classi (dalla 0 alla 10), riportiamo i risultati da noi ottenuti, senza però poter calcolare metriche oggettive, in quanto i dati di test della challenge non sono etichettati.

I risultati sono riportati nella figura 6: sulla sinistra vi è la classe, sulla destra il numero di elementi del dataset di test etichettati come quella classe. È importante ricordare che le classi 9 e 10 del test sono state accorpate nella classe 9.

```
0: 200
1: 156
2: 311
3: 99
4: 105
5: 56
6: 174
7: 96
8: 284
9: 127
```

Figura 9: Risultati test complessivo con 7 classi in training e 11 in test

5.2 – Conclusioni

A primo impatto, i risultati della prova con il dataset da noi ricavato sembrano essere scadenti per le classi 3 e 6, soprattutto per recall e f1, prossimi allo zero (vedi figura 5). In realtà bisogna contestualizzare: infatti, i risultati portano con loro molta informazione. Nessuna delle numerose prove da noi effettuate riusciva ad individuare in alcun modo classi non viste in train. A seguito di

uno studio approfondito dei dati e a seguito di vari tentativi, siamo riusciti a trovare una soluzione che, nonostante i risultati oggettivamente “non ottimi”, riesca a individuare la presenza di classi mai viste in training.

Ciò risulta essere un buon risultato se consideriamo, oltre alla difficoltà della challenge in sé, il fatto che la challenge sia ancora aperta.

Per quanto riguarda la prova generale, non possiamo dire nulla sul comportamento del modello da noi elaborato, in quanto i dati di test della challenge non sono etichettati. Ci aspettiamo, però, che il comportamento sia, in linea di massima, simile alla prima prova da noi effettuata.

Bibliografia

[1] Xueyi Li, Jialin Li, Chengying Zhao, Yongzhi Qu, David He, Gear pitting fault diagnosis with mixed operating conditions based on adaptive 1D separable convolution with residual connection, *Mechanical Systems and Signal Processing*, Vol. 142, Aug. 2020.

[2] Yongzhi Qu, Yue Zhang, Miao He, David He, Chen Jiao, and Zude Zhou, Gear pitting fault diagnosis using disentangled features from unsupervised deep learning, *Journal of Risk and Reliability*, Vol. 233, No. 5, pp. 719-730, 2019.