# Relazione progetto Network Intelligence

Claudio Musmeci 1000056290

## Obiettivo

L’obiettivo del modello proposto è quello di predire la tipologia di guasto hardware in base ai segnali di allarme registrati durante la finestra di osservazione di 15 minuti. Tale modello potrebbe essere utilizzato per prevenire dei guasti nelle reti a microonde migliorandone l’affidabilità. L’intero progetto è stato svolto utilizzando Python. Il progetto consiste poi in una breve analisi di come varia il comportamento del modello sulla base del cambiamento degli iperparametri.

## Dataset

Il dataset utilizzato è stato reperito dalla seguente repository github: <https://github.com/bonsai-lab-polimi/tnsm2024-data-centric/> .

Il dataset è composto da 1669 osservazioni, ognuna delle quali rappresenta una finestra di osservazione di 15 minuti durante la quale si è verificato un guasto hardware nelle reti a microonde. Le etichette di guasto sono codificate come segue:

* 0: Guasto all'unità interna (IDU), 515 osservazioni
* 1: Guasto all'unità esterna (ODU), 611 osservazioni
* 2: Guasto ai cavi, 207 osservazioni
* 3: Guasto all'alimentazione, 336 osservazioni

Ciascuna riga corrisponde ad una finestra di osservazione.

All’interno del progetto, il dataset è stato scaricato attraverso l’utilizzo della funzione read\_csv() della libreria pandas. Una breve descrizione è fornita dalla funzione describe().

A screenshot of a computer

Description automatically generated

Successivamente il dataset è stato separato nelle sue componenti feature X e target Y.

Affinché le feature abbiano tutte lo stesso peso e la stessa rilevanza, è necessario eseguire un’operazione di standardizzazione, in maniera da ricondurle alla stessa “scala” in modo che il modello non attribuisca erroneamente un peso maggiore ad un valore grande.

Il framework utilizzato è pytorch, che opera attraverso l’utilizzo di tensori. Il dataset è poi stato adattato al framework mediante la creazione di un dataset di tensori. L’ultima operazione eseguita è stata la separazione in train test, validation set e test set per valutare il modello proposto. La suddivisione è 70%, 15%, 15%.

A screenshot of a computer program

Description automatically generated

## Modello A

Il modello proposto è caratterizzato dalla seguente struttura:

A screenshot of a computer program

Description automatically generated

Esso è composto da 5 layer lineari in cui si hanno 512 neuroni che si dimezzano progressivamente ad ogni layer fino all’ultimo layer in cui vi è un neurone per classe. All’interno del modello sono stati utilizzati per ogni layer (escluso l’ultimo) dei batch normalization per accelerare l’addestramento e la qualità/stabilità del modello e dei dropout layer che hanno il compito di “spegnere” alcuni neuroni per far sforzare il modello ad apprendere ed evitare dei fenomeni di overfitting. La funzione forward definisce invece come devono essere trattati i dati che entrano all’interno del modello. I dati sono caricati nel modello attraverso dei DataLoader.

## Iperparametri

Gli iperparametri permettono di definire le caratteristiche e le qualità di un modello. Tra questi rientrano oltre ai vari neuroni presenti all’interno di ogni layer, anche il numero di epoche e il learning rate. Generalmente avere un grande numero di epoche implica che il modello analizzerà i dati forniti tante volte, quindi potremmo ricadere in fenomeni di overfitting. Il learning rate è un parametro che ha a che fare con la backpropagation ed il calcolo del della loss function, esso quindi indica la velocità con cui il nostro modello apprende ed estrae le informazioni dai dati forniti, valori alti implicano forti oscillazioni.

Nel modello sono stati proposti i seguenti iperparametri:

A math equations on a white background

Description automatically generated

La loss function utilizzata è la cross-entropy, mentre l’ottimizzatore scelto è Adam. Nel codice è inclusa la funzione di addestramento del modello che oltre ad aggiornare i parametri del modello, inserisce anche dei valori all’interno di quattro vettori (train\_loss\_history, train\_acc\_history, val\_loss\_history, val\_acc\_history) utilizzati per eseguire dei plot e fare delle considerazioni sulla qualità del modello.

## Risultati modello A

A graph of a graph

Description automatically generatedA graph showing a line of blue and orange

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Il primo modello è stato addestrato per 50 epoche con un learning rate moderato pari a 1x10^-4. I risultati ottenuti sono molto buoni con una accuracy in fase di training che diventa massima nella 50esima epoca, mentre in fase di validation nella 42 esima epoca. Dai grafici è possibile notare che il modello sta imparando correttamente senza ricadere in marcati fenomeni di underfitting ed overfitting.

A screenshot of a computer screen

Description automatically generated

Utilizzando il modello in evaluation mode e dandogli in ingresso il test set (mai visto dal modello), il modello si comporta in maniera positiva, ottenendo dei risultati tra i vari indici che si attestano intorno al 93%.

## Modello B

Il modello B consiste in un semplice cambiamento del learning rate applicato al modello precedente.

Il learning rate è impostato ad un valore di 1x10^-8, quindi molto basso.

A graph showing a graph of a train and vallation loss

Description automatically generated with medium confidenceA graph of a graph with blue and orange lines

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Dai grafici della Loss e Accuracy è possibile notare come il modello non riesca a performare in maniera positiva. Dal primo grafico si può osservare che la train loss è molto alta ed è più o meno stabile intorno a 1.60, mentre la validation loss è molto più bassa, questo potrebbe essere dovuto ad un fenomeno di overfitting, cosa confermata anche dal secondo grafico, in cui si nota che il modello opera meglio sul validation set, con prestazioni che non sono ottime in quanto si ha un picco intorno alla decima epoca con un valore pari a 0.3880. Ulteriori conferme arrivano dall’analisi delle altre metriche e dalla matrice di confusione che mostrano come il modello tenda a sbagliare.

A screenshot of a computer screen

Description automatically generatedA graph with numbers and a number in blue squares

Description automatically generated with medium confidence

Nel caso di un learning rate molto alto come 0.1 per poter apprezzare i cambianti l’allenamento è stato realizzato per 100 epoche, si ha che:

A graph of a graph with blue and orange lines

Description automatically generatedA graph of a graph

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Sebbene i dati indichino che le prestazioni siano buone, i grafici mettono in risalto il comportamento oscillatorio del modello stesso. Entrambi i grafici nelle prime epoche hanno dei comportamenti accettabili, ma dalla 21esima epoca in poi le curve iniziano ad oscillare in maniera molto marcata.

Da queste due brevi analisi, quindi, è possibile concludere che il learning rate consigliato che permette di avere delle ottime prestazioni è quello moderato di 1x10^-4.

## Modello C

L’ultimo modello proposto consiste in un’architettura più complessa del modello, in cui, rispetto al precedente, ogni layer ha raddoppiato i neuroni.

Un modello più complesso è in grado di estrarre dalle informazioni dei pattern più complessi e nascosti. In particolare, aumentando la profondità della rete ed il numero di neuroni per layer le prestazioni dovrebbero migliorare.

A graph with blue and orange lines

Description automatically generatedA graph with blue and orange lines

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Dai grafici è possibile notare come il comportamento sia simile a quello del modello A, mentre i valori indicano un leggero miglioramento nelle prestazioni che può anche essere notato mediante l’analisi delle varie metriche e della matrice di confusione.

A screenshot of a graph

Description automatically generatedA graph with numbers and squares

Description automatically generated

Dai grafici è possibile notare che il modello riesce a generalizzare meglio i dati ed ottenere quindi una maggiore qualità, inoltre il modello commette anche un numero di errori limitato.

## Conclusione

Queste brevi analisi hanno mostrato che un modello è fortemente dipende dagli iperparametri scelti. In particolare, andrebbe realizzato un modello capace di generalizzare le informazioni in maniera corretta ed efficace. Proprio per questo motivo tra un modello estremamente complesso ed un modello più semplice, nel caso di prestazioni più o meno simili come nel caso in esame, dovrebbe essere privilegiato il modello semplice per diversi motivi tra cui la semplicità di allenamento e la complessità di realizzazione. Mentre per quanto riguarda il learning rate, andrebbe scelto un learning rate moderato che non faccia ricadere il modello né in overfitting né in underfitting o che abbia problemi di oscillazione. Un learning rate moderato consigliato è 1x10^-4.