MIT-BIH Arrhythmia Database: Confronto tra CNN e Autoencoder nella classificazione basata su immagini di battiti ECG

Zacchilli Alessandro[1]

[1]a.zacchilli1@campus.uniurb.it

16/7/2025

Riassunto

Questo studio affronta il problema della classificazione automatica dei battiti cardiaci utilizzando il rinomato dataset "MIT-BIH Arrhythmia Database". La motivazione principale di questa ricerca risiede nella crescente necessità di sviluppare strumenti automatici affidabili capaci di supportare la diagnosi precoce e il monitoraggio continuo delle patologie cardiache, contribuendo significativamente alla pratica clinica.
Per rispondere a questa sfida, sono stati esplorati e confrontati due architetture di modelli principali: le reti neurali convoluzionali (CNN) e i Convolutional Autoencoder (CAE). Le immagini di input, sono state generate attraverso una fase di pre-processing dalle serie temporali originali dei battiti cardiaci. Sono state adottate tre diverse strategie sperimentali per ottimizzare il riconoscimento delle immagini: l'utilizzo autonomo della CNN, l'impiego del CAE per l'estrazione di feature combinato con una rete densa per la classificazione, e l'integrazione del CAE per la ricostruzione delle immagini prima della loro classificazione da parte della CNN.
I risultati ottenuti dagli esperimenti hanno chiaramente dimostrato che l'approccio basato sull'utilizzo esclusivo della CNN si è rivelato il più efficace nel riconoscimento e nella classificazione delle immagini dei battiti cardiaci, superando le altre due strategie che incorporavano il Convolutional Autoencoder.

1 Introduzione

Le malattie cardiovascolari rappresentano una delle principali cause di morbilità e mortalità a livello globale, rendendo la diagnosi precoce e il monitoraggio efficace delle patizioni cardiache un'esigenza clinica di primaria importanza. In questo contesto, lo sviluppo di strumenti automatici affidabili per la classificazione dei battiti cardiaci emerge come un campo di ricerca fondamentale, capace di offrire un supporto significativo ai medici nell'identificazione tempestiva di aritmie e anomalie. Il presente lavoro affronta il problema della classificazione automatica dei battiti cardiaci, sfruttando "MIT-BIH Arrhythmia Database". L'input al nostro sistema di classificazione è rappresentato da immagini, generate a partire dalle serie temporali dei segnali dei battiti cardiaci attraverso una fase di pre-elaborazione. L'output atteso è la classificazione automatica del tipo di battito cardiaco, permettendo di categorizzare le diverse morfologie e potenziali anomalie. Per affrontare questa sfida di classificazione, abbiamo esplorato e confrontato le prestazioni di due architetture di modelli di deep learning ampiamente utilizzate in compiti di riconoscimento di pattern e visione artificiale: le reti neurali convoluzionali (CNN) e i Convolutional Autoencoder (CAE)[1]. Sono state investigate tre distinte strategie sperimentali per ottimizzare il processo di riconoscimento delle immagini:

- L'utilizzo autonomo di una rete CNN per la classificazione diretta delle immagini dei battiti.
- L'impiego di un CAE per l'estrazione di feature, successivamente alimentate alla stessa rete densa della prima strategia per la classificazione.
- L'utilizzo di un CAE per la ricostruzione delle immagini dei battiti, che vengono poi classificate dalla stessa architettura CNN utilizzata nella prima strategia.

Questa ricerca mira a determinare l'approccio più efficace tra quelli proposti per la classificazione automatica dei battiti cardiaci, contribuendo a gettare le basi per lo sviluppo di sistemi diagnostici più robusti e precisi. Per la comprensione dei concetti fondamentali della classificazione e delle metodologie di deep learning, il presente studio ha attinto al materiale didattico del corso fornito dall'Università di Urbino [2]. Un approfondimento specifico sul funzionamento del framework PyTorch è stato condotto consultando il libro [3], mentre il sito web [4] ha fornito utili esempi e supporto per l'implementazione pratica del codice.

2 Metodi

Questa sezione ha lo scopo di descrivere in maniera esaustiva le metodologie adottate nel presente studio, illustrando i modelli e gli algoritmi di apprendimento impiegati, il dataset utilizzato e le fasi di pre-elaborazione dei dati.

2.1 Dataset Utilizzato: MIT-BIH Arrhythmia Database

Il presente studio ha utilizzato il "MIT-BIH Arrhythmia Database", un dataset ampiamente riconosciuto e utilizzato nella ricerca sull'analisi dei segnali elettrocardiografici (ECG).

2.1.1 Dimensione e composizione

Il dataset originale è composto da 48 registrazioni della durata di circa 30 minuti ciascuna, acquisite tramite Holter. Per motivi di praticità, è stata utilizzata una versione del dataset già convertita in un formato più semplice da gestire [5](disponibile sulla piattaforma Kaggle). Questo dataset è strutturato in 48 file CSV, che rappresentano le serie temporali dei battiti, e altrettanti 48 file TXT contenenti le annotazioni dei battiti. Per questo studio, è stata considerata solamente la derivazione MLII, ritenuta più significativa. Sebbene il dataset originale contenga circa 17 tipi diversi di battiti cardiaci, per questo progetto sono state considerate solo 3 annotazioni specifiche: "N" (Normal beat), "V" (Premature ventricular contraction) e "A" (Atrial premature beat).

2.1.2 Selezione e distribuzione dei dati

Per motivi di limiti computazionali, sono state selezionate solo 2 registrazioni: la numero 200 e la numero 209.

- La registrazione 200 include un totale di 2601 battiti, di cui 1743 normali (N), 30 atriali prematuri (A) e 826 ventricolari prematuri (V).
- La registrazione 209 include 3005 battiti, di cui 2621 normali (N), 383 atriali prematuri (A) e 1 ventricolare prematuro (V).
- Del totale complessivo dei battiti dalle due registrazioni, per gli stessi limiti computazionali, ne sono stati considerati solo 2435, distribuiti come segue: 1368 N, 413 A, 648 V.

Per vedere il bilanciamento delle varie classi è utile l'istogramma in figura 1

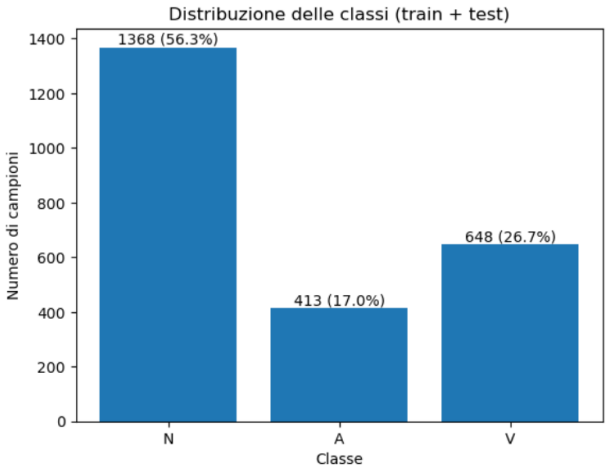

Figura 1: Distribuzione delle classi delle immagini scelte

2.1.3 Suddivisione dei dati

I battiti sono stati suddivisi in modo casuale in set di training e test, mantenendo un rapporto di 0,2, ovvero 80% per il training e 20% per il test. Le immagini utilizzate per l'addestramento e il test sono state caricate da directory specifiche ('Images/train' e 'Images/test'), mappando le etichette 'N', 'A', 'V' ai valori numerici 0, 1, 2 rispettivamente. Le immagini sono state poi caricate e convertite in bianco e nero, con l'applicazione di trasformazioni come il flip orizzontale casuale, la rotazione casuale di 10 gradi e la conversione in tensore.

2.2 Pre-elaborazione dei dati

La fase di pre-elaborazione è stata fondamentale per trasformare le serie temporali originali dei battiti in un formato adatto all'input dei modelli di deep learning.

2.2.1 Generazione delle Immagini del Battito

Per generare le immagini, è stato preso come riferimento il picco del battito (quello a cui si riferiscono le annotazioni). La finestra temporale di ogni battito è stata definita come l'intervallo tra il picco del battito precedente e il picco del battito successivo.

2.2.2 Normalizzazione della Lunghezza del Battito

Dato che ogni battito ha una lunghezza variabile, è stato preso il battito più lungo come riferimento, che misurava precisamente 724 campioni. Per uniformare la lunghezza, è stato eseguito un padding con zeri (0) a tutti i battiti più corti, aggiungendo gli zeri dopo la finestra temporale del battito.

2.2.3 Formato dell'Immagine Risultante

L'immagine finale risultante da questa pre-elaborazione è di 198x120 pixel, e viene rappresentata in bianco e nero. Un esempio dell'immagine generata può essere visionato nella figura 2.

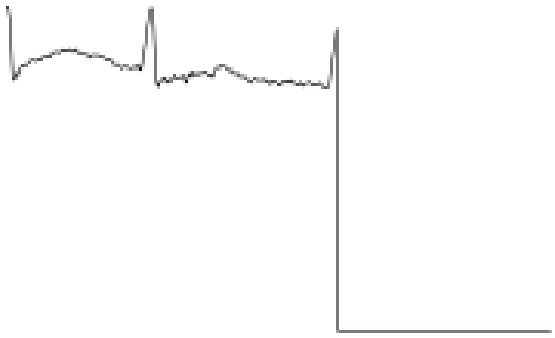

Figura 2: Esempio di battito appartenente alla classe A

2.3 Descrizione Generale dei Modelli e degli Algoritmi di Apprendimento

In questa sezione vengono descritti i modelli di apprendimento utilizzati e le strategie sperimentali adottate. L'obiettivo è analizzare le performance dei diversi approcci nella classificazione dei battiti cardiaci. Gli iperparametri comuni a tutti i modelli sono il numero di epoche (5), il learning rate (0.001), l'ottimizzatore (Adam) e la dimensione del batch (64).

2.3.1 Rete Densa

La rete densa utilizzata è una semplice rete neurale con un solo strato nascosto. L'architettura prevede un primo strato lineare che mappa l'input a 160 neuroni, seguito da una funzione di attivazione ReLU, e infine un secondo strato lineare che mappa i 160 neuroni al numero di classi di output.

2.3.2 Rete Neurale Convoluzionale (CNN - ConvNet)

L'architettura della CNN è composta da due blocchi convoluzionali sequenziali seguiti da un livello fully connected.

- Il primo strato convoluzionale ha 1 canale di input e 16 canali di output, la dimensione del kernel di 5, seguito da una funzione di attivazione ReLU e un layer di pooling con dimnesione del kernel di 3.
- Il secondo strato convoluzionale è simile, con 16 canali di input e 16 canali di output, dimensione del kernel di 5, seguito da ReLU e un layer di pooling con dimensione del kernel di 3.
- Dopo questi strati convoluzionali, l'output viene appiattito e passato a un livello fully connected, che è un'istanza della rete densa definita precedentemente, con un input dimensionale di '20 * 11 * 16' (risultante dall'appiattimento degli output convoluzionali) e un output pari al numero di classi.

2.3.3 Convolutional Autoencoder (CAE)

L'autoencoder è un modello neurale utilizzato per apprendere una rappresentazione efficiente dei dati. Si compone di due parti principali: un encoder e un decoder.

L'autoencoder viene addestrato con nn.MSELoss come funzione di costo.

- L'encoder è una sequenza di strati convoluzionali che comprimono l'input in una rappresentazione latente a dimensione ridotta. Il primo strato convoluzionale dell'encoder ha 1 canale di input e 16 canali di output, con la dimensione del kernel di 3, stride di 1, padding di 1, seguito da ReLU e un layer di pooling con dimensione del kernel a 3 e stride di 2. Il secondo strato convoluzionale dell'encoder ha 16 canali di input e 8 canali di output, con dimensione del kernel di 3, stride di 1, padding di 1, seguito da ReLU e un layer di pooling con la dimensione del kernel a 2 e stride di 2.

- Il decoder è una sequenza di strati di convoluzione trasposta che decomprimono la rappresentazione latente per ricostruire l'immagine originale. Il primo strato del decoder ha 8 canali di input e 16 canali di output, con dimensione del kernel di 3, stride di 2, padding di 1, l'output_padding di 1, seguito da ReLU. Il secondo strato del decoder ha 16 canali di input e 1 canale di output, con dimensione del kernel di 3, stride di 2, padding di 1, output_padding di 1, seguito da una funzione di attivazione Sigmoid. Infine, viene applicata un'interpolazione per raggiungere le dimensioni di output desiderate (120x198 pixel).

2.4 Strategie Sperimentali Adottate

Sono state esplorate tre diverse strategie per la classificazione dei battiti cardiaci:

2.4.1 Strategia 1: CNN Autonoma

In questa strategia, la CNN descritta precedentemente è stata utilizzata direttamente per classificare le immagini dei battiti pre-elaborate. La funzione di costo utilizzata è la Cross Entropy Loss.

2.4.2 Strategia 2: CAE per Feature Extraction + Rete Densa

Questa strategia ha sfruttato il Convolutional Autoencoder (CAE) per l'estrazione di feature significative dalle immagini dei battiti. In particolare, la rappresentazione latente è stata ottenuta dal collo di bottiglia dell'encoder dell'autoencoder addestrato. Le feature sono state estratte utilizzando solo la parte 'encoder' dell' 'Autoencoder' addestrato e appiattite in un vettore 1D per ogni immagine, risultando in una dimensione di 11760. Successivamente, queste feature estratte sono state alimentate a una rete densa per la classificazione. La rete densa utilizzata è la stessa di quella descritta in precedenza e utilizzata nella prima strategia. La funzione di costo utilizzata è la Cross Entropy Loss.

2.4.3 Strategia 3: CAE per Ricostruzione Immagini + CNN

In questa strategia, il Convolutional Autoencoder (CAE) è stato utilizzato per ricostruire le immagini dei battiti. Le immagini originali sono state passate attraverso l'autoencoder per ottenere le loro versioni ricostruite. Le immagini ricostruite sono state poi passate alla stessa CNN utilizzata nella prima strategia per la classificazione. La CNN è stata addestrata con la funzione di costo Cross Entropy Loss.

3 Risultati sperimentali

In questa sezione riportiamo i risultati degli esperimenti condotti, mostrando le relative metriche.

3.1 Metriche di Valutazione

Per valutare le prestazioni dei modelli, abbiamo utilizzato le seguenti metriche: accuratezza, precisione, recall e F1-score. Queste metriche sono fondamentali per comprendere come i modelli si comportano in termini di classificazione e sono particolarmente utili in contesti in cui le classi sono sbilanciate.

3.1.1 Accuratezza

L'accuratezza è definita come la proporzione di previsioni corrette rispetto al numero totale di campioni. È calcolata come:

$$\text{Accuratezza} = \frac{\text{Veri Positivi} + \text{Veri Negativi}}{\text{Totale Campioni}}$$

E' utile perché ci dice quante previsioni corrette ha effettuato il modello, tuttavia, l'accuratezza può essere fuorviante in situazioni in cui le classi sono sbilanciate. Ad esempio, nel nostro caso il 56% delle istanze appartiene a una classe, se il modello prevedesse sempre quella classe, avrebbe un'accuratezza del 56%, ma chiaramente non è un buon modello perché non è utile per identificare le istanze delle classi minoritarie.

3.1.2 Precisione

La precisione misura la proporzione di veri positivi rispetto al totale delle istanze classificate come positive. È calcolata come:

$$\text{Precisione} = \frac{\text{Veri Positivi}}{\text{Veri Positivi} + \text{Falsi Positivi}}$$

Questa metrica è particolarmente importante in contesti in cui i costi di un falso positivo sono elevati. Ad esempio, in un'applicazione di screening medico, una bassa precisione potrebbe significare che molti pazienti sani vengono erroneamente identificati come malati, portando a stress e costi inutili.

3.1.3 Recall

Il recall, noto anche come sensibilità o tasso di veri positivi, misura la proporzione di veri positivi rispetto al totale delle istanze realmente positive. È calcolato come:

$$\text{Recall} = \frac{\text{Veri Positivi}}{\text{Veri Positivi} + \text{Falsi Negativi}}$$

Il recall è cruciale in situazioni in cui è importante identificare tutte le istanze positive. Ad esempio, in un sistema di rilevamento di frodi, un basso recall potrebbe significare che molte frodi non vengono rilevate, con conseguenti perdite finanziarie significative.

3.1.4 F1-score

L'F1-score è la media armonica tra precisione e recall e fornisce un bilanciamento tra le due metriche. È particolarmente utile quando si desidera un compromesso tra precisione e recall. È calcolato come:

$$\text{F1-score} = 2 \cdot \frac{\text{Precisione} \cdot \text{Recall}}{\text{Precisione} + \text{Recall}}$$

L'F1-score è una metrica utile in scenari in cui le classi sono sbilanciate e si desidera evitare di ottimizzare solo per l'accuratezza. Un alto F1-score indica che il modello ha sia una buona precisione che un buon recall.

3.2 Risultati delle Strategie di Apprendimento

Le tre strategie di apprendimento hanno restituito i risultati mostrati nella tabella 1. Ogni strategia ha le seguenti voci:

- CNN, usata per la prima strategia, quindi unicamente la CNN.
- CAE, per la seconda strategia, quindi sfruttando le feature nello spazio latente per addestrare la rete densa.
- CAECNN, per la terza, ovvero per CNN dopo la ricostruzione delle immagini tramite Autoencoder.

Ogni voce principale ha la classe a cui appartiene la rispettiva metrica di valutazione, quindi per esempio CNN-N, si riferisce alla prima strategia e alla classe N. L'accuratezza è uguale per tutte le classi della stessa strategia perché è una metrica che tiene conto di tutte le classi.

Tabella 1: Risultati delle Strategie di Apprendimento

	Accuracy	Precision	Recall	F1
CNN-N	0.85	0.84	0.95	0.89
CNN-A	0.85	0.71	0.67	0.69
CNN-V	0.85	0.99	0.73	0.84
CAE-N	0.57	0.57	1.00	0.73
CAE-A	0.57	0.00	0.00	0.00
CAE-V	0.57	0.00	0.00	0.00
CAECNN-N	0.57	0.57	1.00	0.73
CAECNN-A	0.57	0.00	0.00	0.00
CAECNN-V	0.57	0.00	0.00	0.00

3.2.1 Strategia 1: CNN Autonoma

In questa strategia, la Rete Neurale Convoluzionale (CNN) è stata addestrata direttamente sulle immagini pre-elaborate dei battiti cardiaci. I risultati dei test sono visibili nella tabella 1, come si può notare l'accuratezza è buona (85%) ma si nota un calo delle prestazioni per la classe A, questo è dato probabilmente dal fatto che è la classe con meno sample a disposizione, quindi soffre dello sbilanciamento delle classi. Per analizzare meglio i risultati possiamo prendere come esempio la matrice di confusione in figura 3, notiamo che quasi metà dei campioni della classe A sono stati etichettati come appartenenti alla classe N, questo indica il fatto che il modello tende a etichettare di più la classe maggioritaria, un problema tipico dei dataset sbilanciati.

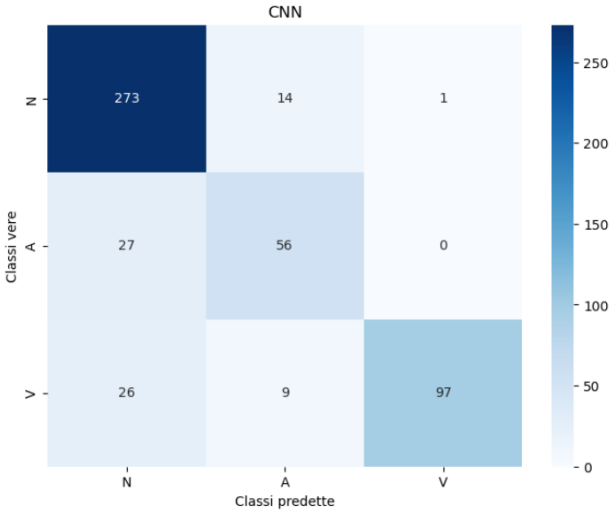

Figura 3: Matrice di confusione della prima strategia (CNN)

3.3 Strategia 2: CAE per Feature Extraction + Rete Densa

Questa strategia ha sfruttato un Convolutional Autoencoder (CAE) per l'estrazione di feature dalle immagini dei battiti. La rappresentazione latente, ottenuta dal "collo di bottiglia" (bottleneck) dell'encoder del CAE, è stata poi utilizzata come input della stessa rete densa della strategia 1, per la classificazione. Dai risultati della tabella 1 notiamo che questa volta il modello non è in grado di classificare le classi adeguatamente, guardando anche la matrice di confusione 4, si vede chiaramente che tutte le classi predette sono appartenenti alla classe maggioritaria, quindi abbiamo un notevole calo di performance rispetto alla prima strategia.

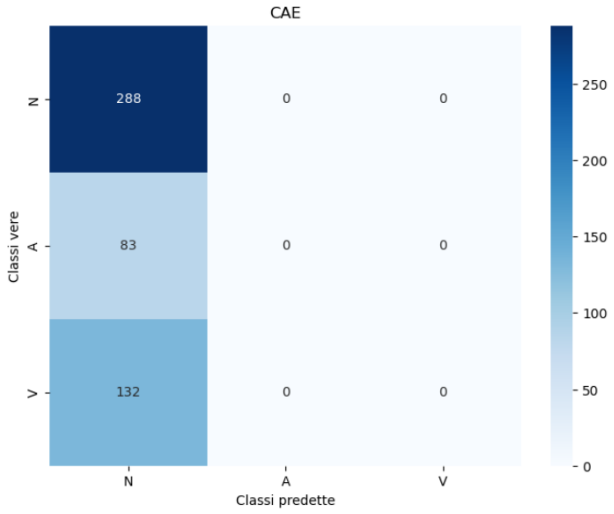

Figura 4: Matrice di confusione della seconda strategia (CAE)

3.4 Strategia 3: CAE per Ricostruzione Immagini + CNN

In questa strategia, il Convolutional Autoencoder (CAE) è stato impiegato per ricostruire le immagini dei battiti. Le immagini così ricostruite sono state poi passate alla stessa CNN utilizzata nella strategia 1 per la classificazione. Anche in questo caso sia dalla tabella 1 che dalla matrice di confusione 5 notiamo risultati identici alla seconda strategia, evidenziando cioé l'inefficacia di questo modello, nella classificazione delle classi minoritarie.

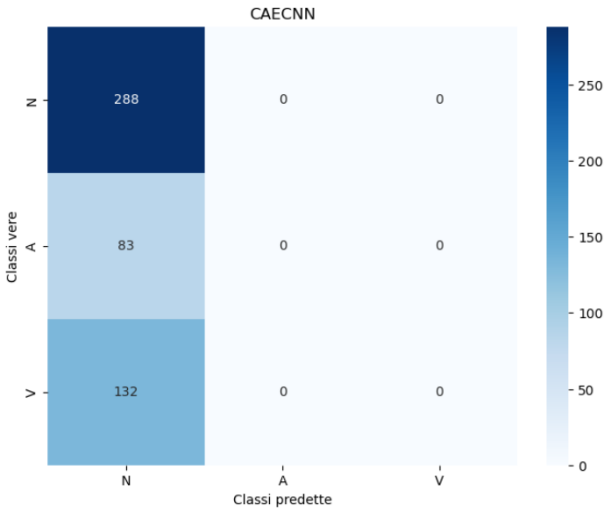

Figura 5: Matrice di confusione della terza strategia (CAECNN)

Inoltre in figura 6 possiamo notare degli esempi delle immagini ricostruite che, visibilmente, hanno perso informazioni importanti che impediscono alla CNN di fare una classificazione adeguata.

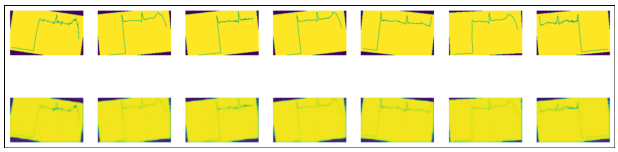

Figura 6: Nella prima riga troviamo delle immagini originali usate per l'addestramento, nella seconda riga le rispettive immagini ricostruite dal CAE

4 Conclusioni

Questo studio ha esplorato l'applicazione di diverse strategie di apprendimento profondo per la classificazione dei battiti cardiaci dal "MIT-BIH Arrhythmia Database", concentrandosi in particolare sui battiti Normali ('N'), Atriali Prematuri ('A') e Ventricolari Prematuri ('V'). L'obiettivo era valutare l'efficacia di una Rete Neurale Convoluzionale (CNN) autonoma e di approcci che integrano un Convolutional Autoencoder (CAE) per l'estrazione di feature o la ricostruzione delle immagini, nella complessa attività di analisi di segnali ECG trasformati in immagini.

I risultati hanno evidenziato una grande differenza fra la prima strategia e le altre due, mostrando che la

CNN autonoma si è dimostrata il modello più robusto e appropriato per il compito di classificazione in questo contesto, nonostante le sfide poste dallo sbilanciamento delle classi. Gli approcci basati sul CAE, pur concettualmente validi per la riduzione della dimensionalità o il denoising, non hanno offerto benefici tangibili per la classificazione e, anzi, hanno portato a un drastico peggioramento delle prestazioni sulle classi minoritarie.

È probabile che il CAE, nel processo di compressione/decompressione o di estrazione dello spazio latente, abbia perso informazioni cruciali per la distinzione delle classi minoritarie, oppure abbia amplificato l'effetto dello sbilanciamento delle classi. Guardando al futuro, per migliorare ulteriormente le prestazioni e affrontare le limitazioni emerse, si potrebbero esplorare diverse direzioni:

- Bilanciamento del Dataset: Implementare tecniche avanzate di bilanciamento del dataset, come il sovra-campionamento (e.g., SMOTE) per le classi minoritarie, il sotto-campionamento della classe maggioritaria, o l'uso di funzioni di costo pesate, per mitigare l'impatto dello sbilanciamento delle classi sulla capacità di generalizzazione del modello.
- Architetture CAE Ottimizzate: Sperimentare con architetture di Autoencoder più complesse o specificamente progettate per mantenere le feature discriminative nelle immagini ricostruite o nello spazio latente, magari variando la dimensione del "collo di bottiglia" o le funzioni di attivazione.
- Confronto con Modelli Temporali: Confrontare le prestazioni della CNN basata su immagini con modelli specifici per serie temporali, come le Reti Neurali Ricorrenti (RNN) o CNN 1D, che potrebbero catturare meglio le dipendenze temporali intrinseche dei segnali ECG.

Riferimenti bibliografici

[1] Yifei Zhang. "A better autoencoder for image: Convolutional autoencoder". In: (2018), p. 34.

[2] Università degli Studi di Urbino Carlo Bo. *Materiale del Corso di Machine Learning*. `https://blended.uniurb.it/`. 2023.

[3] Eli Stevens, Luca Antiga e Thomas Viehmann. *Deep Learning with PyTorch*. Manning Publications Co., 2020.

[4] Geeksforgeeks. *Implement Convolutional Autoencoder in PyTorch with CUDA*. `https://www.geeksforgeeks.org/machine-learning/implement-convolutional-autoencoder-in-pytorch-with-cuda/`. 2025.

[5] TaeJoongYoon. *MIT-BIH Arrhythmia Database*. `https://www.kaggle.com/datasets/taejoongyoon/mitbit-arrhythmia-database`. 2018.