Project work deep learning: classificazione del pianto neonatale mediante reti neurali

Giorgia Gammone e Beatrice Laureti

Università degli studi di Ferrara

16 ottobre 2025

Descrizione del progetto

Obiettivo

- Creare una rete neurale capace di riconoscere il motivo del pianto di un neonato a partire da segnali audio.
- **Motivazioni:** Il pianto è il principale mezzo di comunicazione nei neonati. Comprendere il motivo può supportare genitori e caregiver.

Dataset

Descrizione Dataset

- **donateacry-corpus:** Un corpus audio di pianti di neonati creato attraverso la campagna Donate-a-Cry.
- Il dataset contiene campioni audio caricati dagli utenti, nel loro formato originale, non modificato e non verificato. I campioni audio sono stati caricati tramite l'applicazione Donate-a-Cry per Android e iOS (nelle cartelle donateacry-android-upload-bucket, donateacry-ios-upload-bucket).
- I file audio contengono campioni di pianto di neonati, con le informazioni di etichettatura codificate nel nome del file. I campioni sono stati etichettati direttamente dagli utenti.

Pulizia del dataset

- Il dataset di riferimento è donateacry_corpus_cleaned_and_updated_data.
- Tutti i dati sono stati convertiti in formato WAV, con bitrate uniforme di 128 kbps e frequenza di campionamento di 8 kHz.
- Vengono utilizzate come riferimento le categorie DBL (Dunstan Baby Language):
  - I dati etichettati come lonely, scared e unknown sono stati rimossi.
  - I dati etichettati come cold/hot sono stati uniti nella categoria "discomfort".
- Tutti i campioni che non rappresentano veri pianti sono stati rimossi manualmente dopo ascolto diretto.

Statistiche sulla durata dei campioni audio

Dataset	Count	Mean	Std	Min	25%	50%	75%	Max
android	712	6.862	0.250	2.340	6.78	6.90	6.96	7.28
ios	416	6.870	0.885	0.004	7.00	7.00	7.00	7.00
cleaned	457	6.918	0.114	6.520	6.88	6.96	7.00	7.06

Tabella: Statistiche descrittive della durata (in secondi).

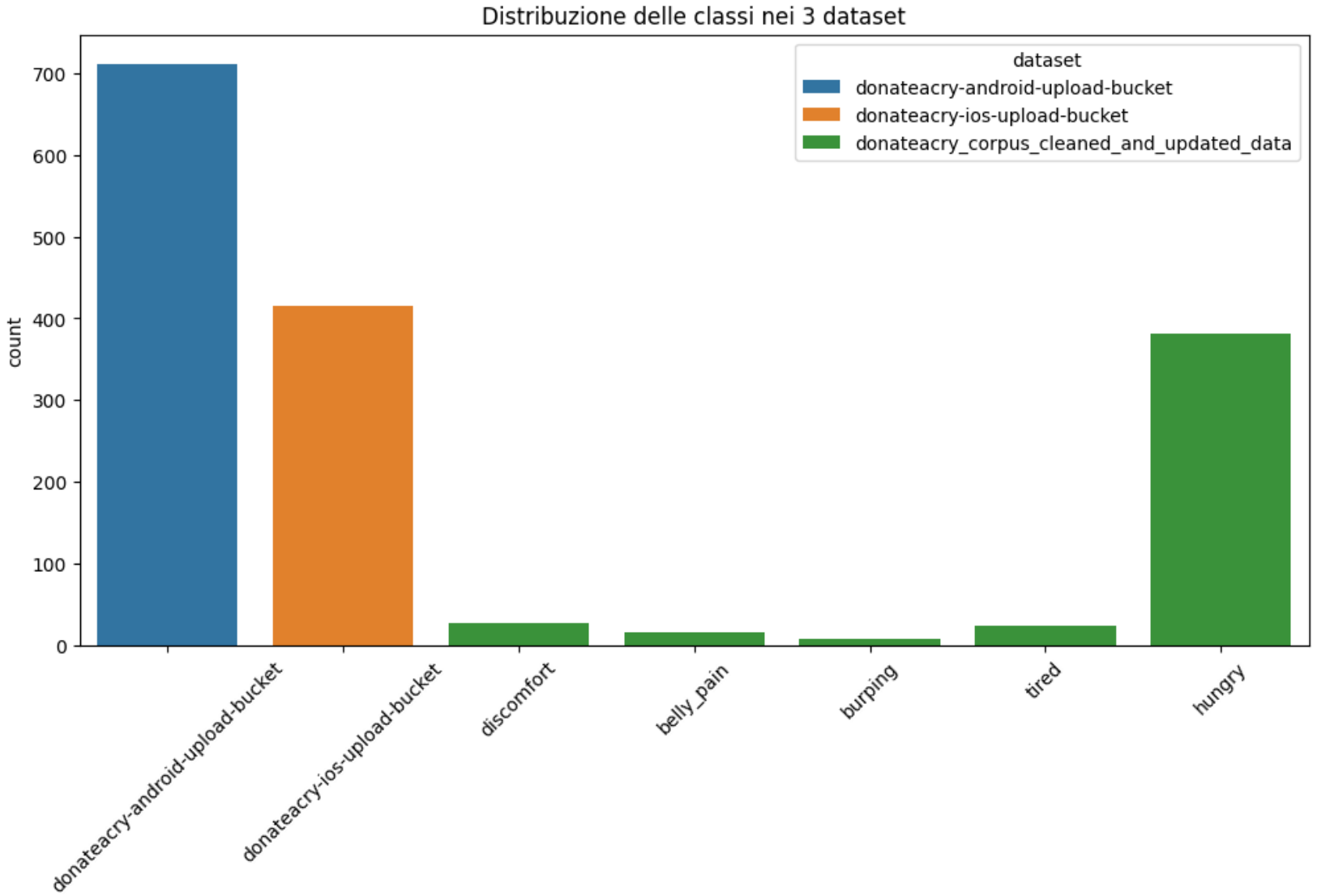
Numero di file per classe nei dataset

Dataset	Numero di file
donateacry-android-upload-bucket	712
donateacry-ios-upload-bucket	416
donateacry_corpus_cleaned_and_updated_data	457

- Dataset utilizzato: donateacry_corpus_cleaned_and_updated_data

Classe	Numero di file
belly_pain	16
burping	8
discomfort	27
hungry	382
tired	24

Distribuzione classi nei dataset



Distribuzione classi: caso binario

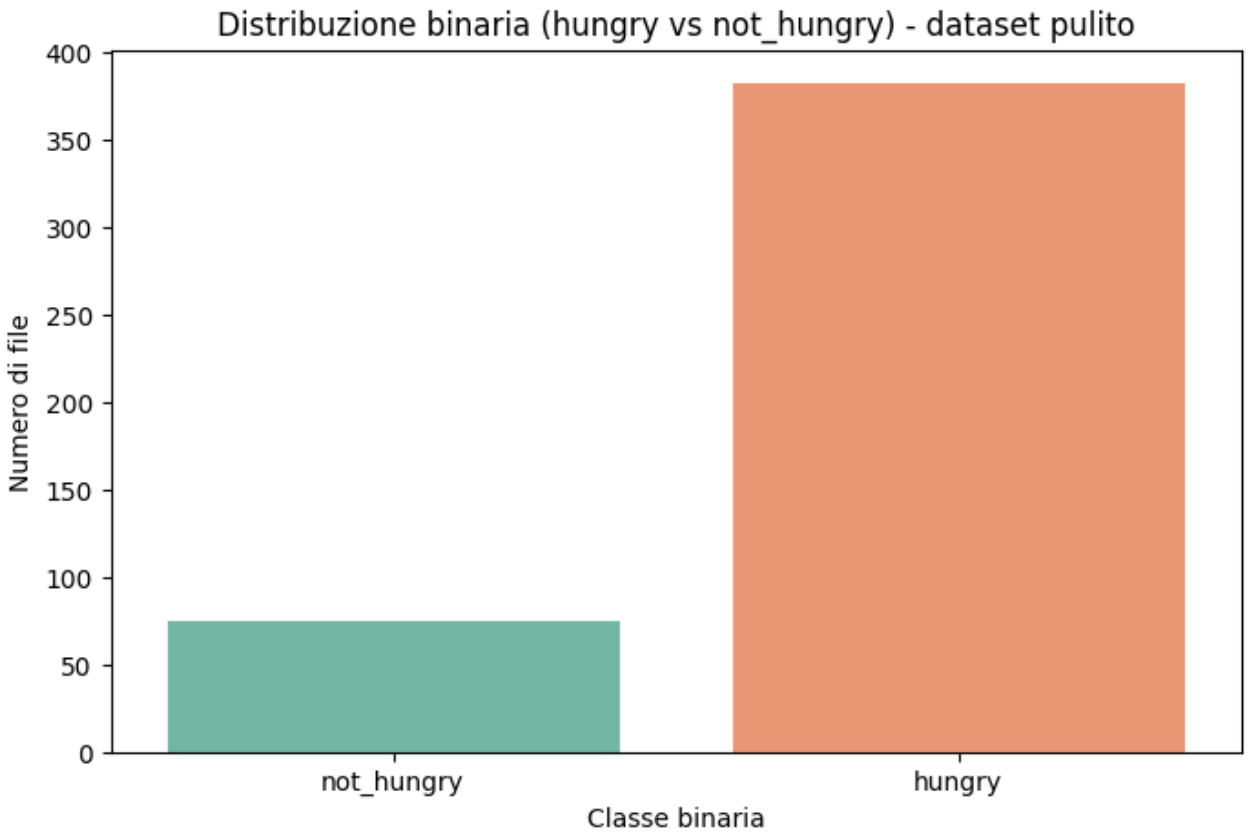


Figura: Not hungry: 75 (16,41%) - Hungry: 382 (83,59%)

Considerazioni sul dataset

- Dataset fortemente sbilanciato tra le diverse classi.
- Numero complessivo di campioni molto limitato.
- Etichette fornite dai genitori, quindi potenzialmente imprecise.

- Questi fattori incidono in modo significativo sulle prestazioni della rete neurale.
- È necessario applicare tecniche specifiche di data augmentation e transfer learning per gli squilibri e il numero limitato di esempi.

PCA 2D

- Indica quanta parte della varianza totale dei dati è spiegata da ciascuna componente.

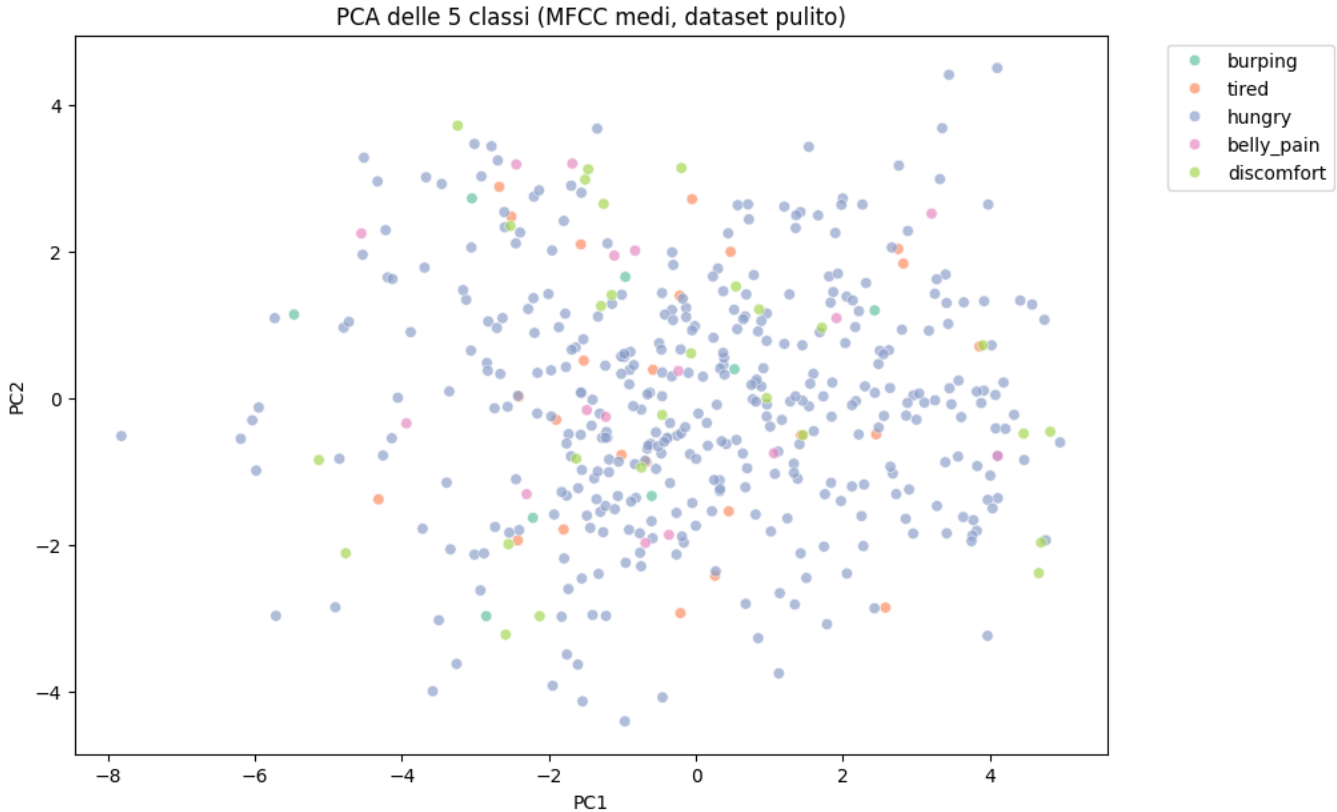


Figura: Explained variance ratio: [0.2915737 0.13891621] - 43%

PCA 3D

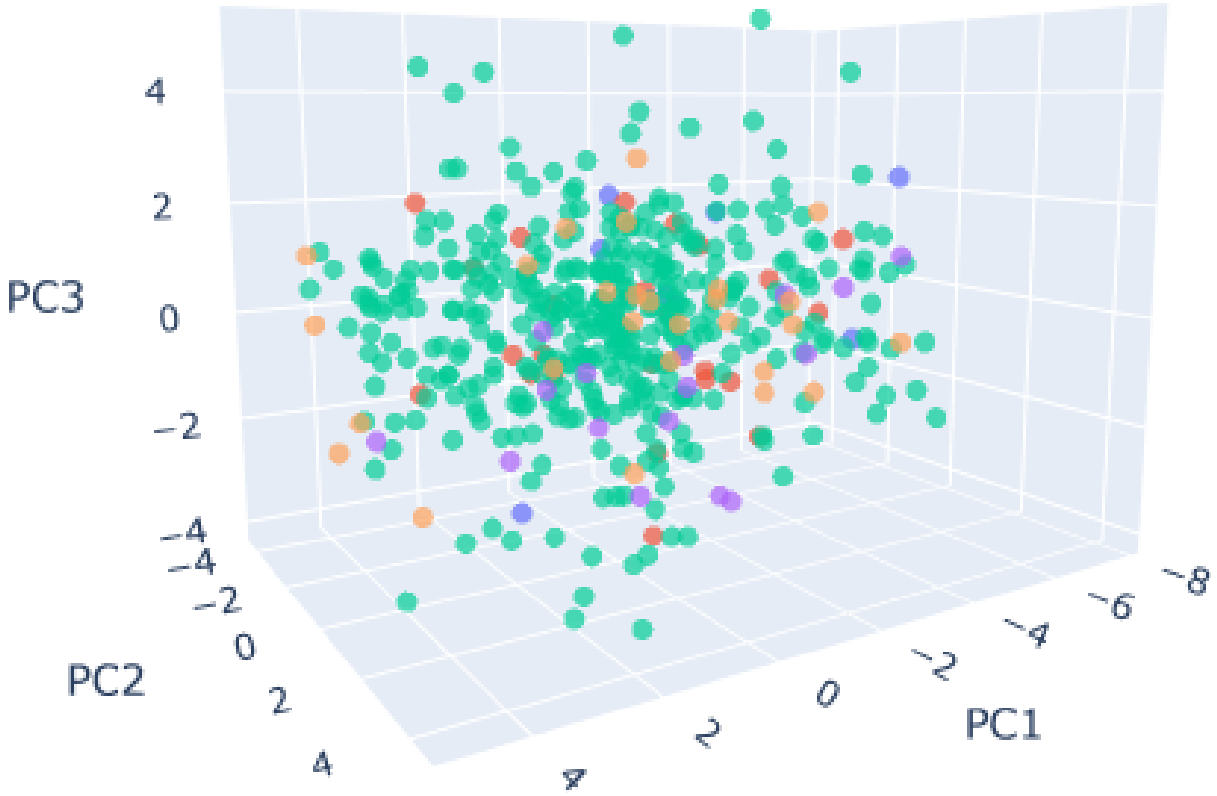
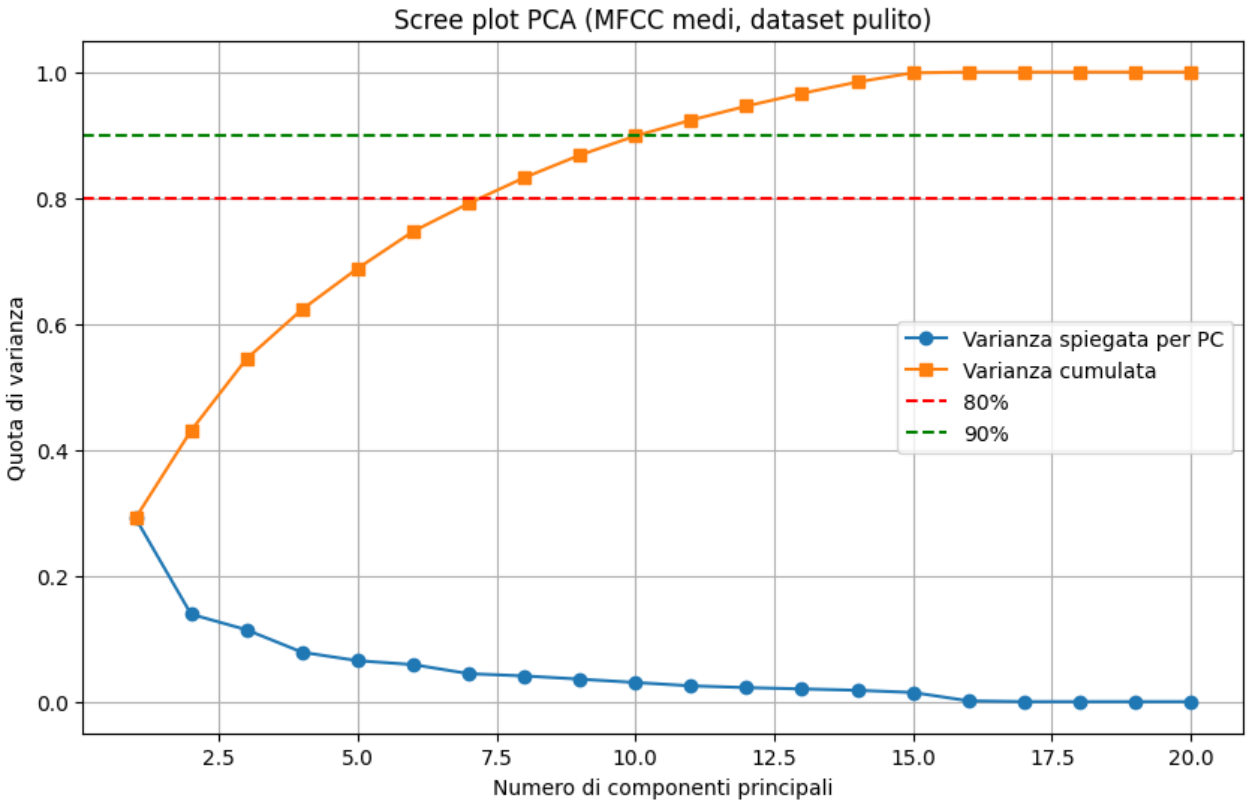


Figura: [0.2915737 0.13891621 0.11410655] - 54%

Scree Plot PCA



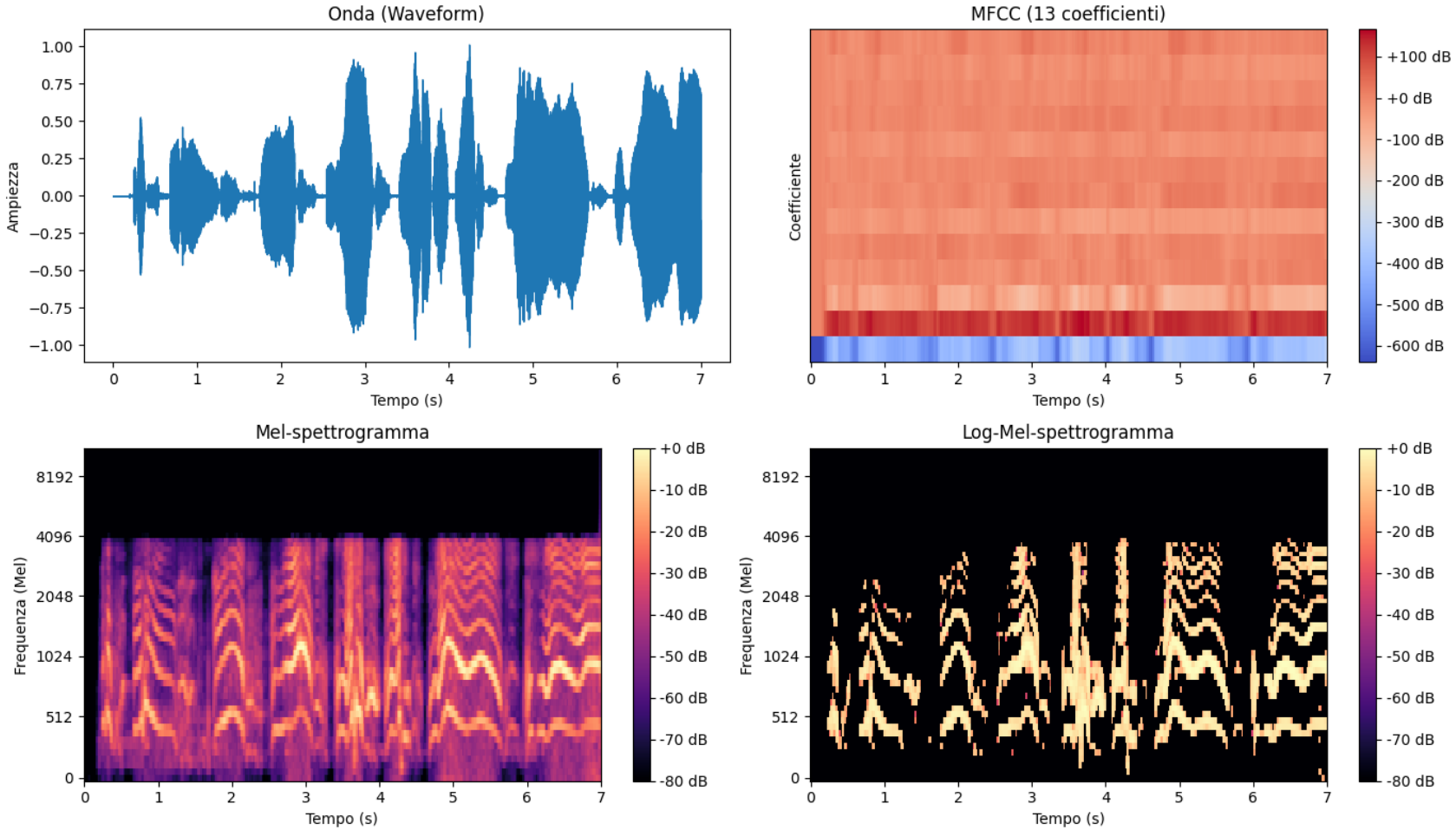
Rappresentazione degli audio

Rappresentazioni degli audio per le reti neurali

- **MFCC (Mel-Frequency Cepstral Coefficients)**
  - Rappresentazione compatta basata sulla percezione uditiva umana.
  - Ottima per il parlato, ma perde dettagli spettrali fini.
- **Spettrogramma**
  - Rappresentazione grafica dell'intensità di un suono in funzione del tempo (asse x) e della frequenza (asse y).
  - Mantiene più informazioni rispetto agli MFCC.
  - Ideale per CNN (può essere trattato come un'immagine 2D).
- **Audio grezzo (Waveform 1D)**
  - Usa direttamente il segnale nel dominio del tempo.
  - Richiede molti dati e risorse.

Mel-Spettrogramma e Log-Mel Spettrogramma

- Il **mel-spettrogramma** è la variante percettiva dello spettrogramma: le frequenze vengono proiettate sulla **scala Mel**, che imita la percezione uditiva umana.
- Riduce la dimensionalità mantenendo l'informazione rilevante per l'orecchio umano.
- Il (**log-mel-spettrogramma**) è il più simile alla percezione umana, riduce la varianza del dataset e aiuta la rete a concentrarsi sulla struttura timbrica del pianto invece che sull'intensità.


Onda (Waveform)
Ampiezza
Tempo (s)
MFCC (13 coefficienti)
Coefficiente
Tempo (s)
+100 dB
+0 dB
-100 dB
-200 dB
-300 dB
-400 dB
-500 dB
-600 dB
Mel-spettrogramma
Frequenza (Mel)
Tempo (s)
Log-Mel-spettrogramma
Frequenza (Mel)
Tempo (s)
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

Tecniche per migliorare l'apprendimento

Data Augmentation: motivazioni

- **Riduce lo sbilanciamento** tra le classi, generando nuovi esempi per quelle più rare, aumentando la robustezza del modello.
- **Limita l'overfitting**, creando varietà nei campioni e rendendo il modello più generalizzabile.
- È importante applicarla solo al training set, per evitare di alterare i dati di validazione e test.

Data augmentation: rischi e buone pratiche

- Evitare trasformazioni troppo forti che rendono il suono irriconoscibile.
- Applicare augmentation anche alle classi maggioritarie, per evitare che il modello impari a riconoscere l'effetto dell'augmentation.
- Combinare più trasformazioni casuali e mantenere sempre anche i campioni originali.

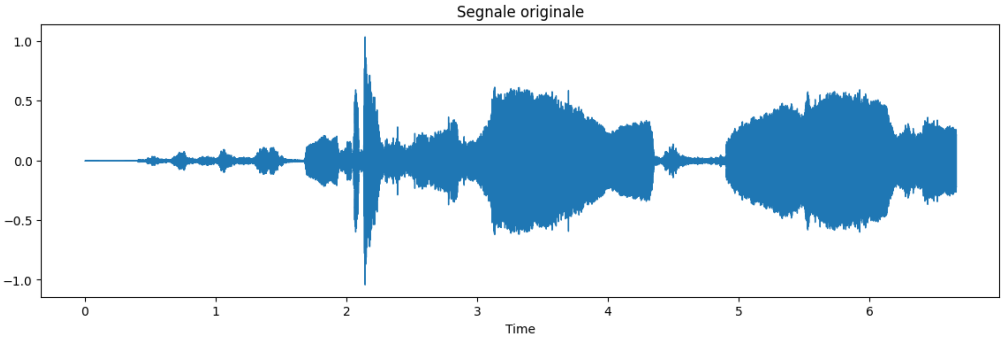
Data augmentation: modalità di applicazione

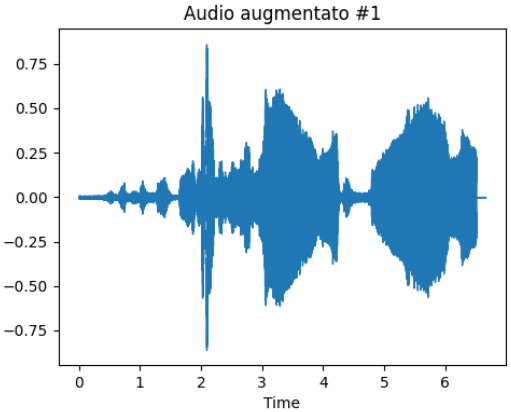
- **Offline** augmentation: i file vengono trasformati e salvati prima del training.
  - Vantaggi: più veloce, riproducibile, adatto a dataset piccoli.
  - Svantaggi: esempi statici, minore variabilità.

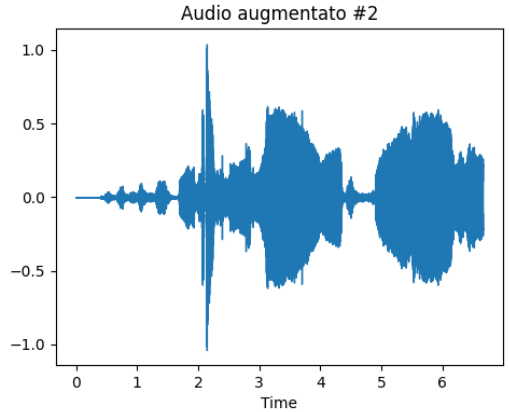
- **On-the-fly** augmentation: le trasformazioni vengono applicate in tempo reale nel DataLoader.
  - Vantaggi: variabilità continua, nessun file duplicato.
  - Svantaggi: più lento e meno riproducibile.

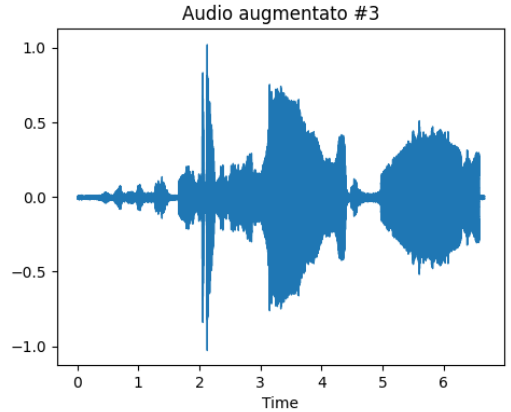
Data Augmentation: metodi utilizzati

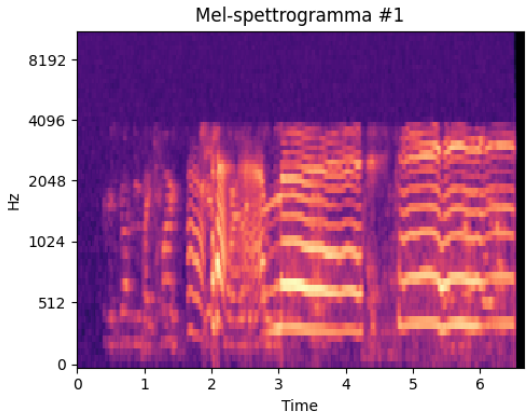
- **AddGaussianNoise**: aggiunge rumore bianco.
- **TimeStretch**: modifica leggermente la velocità del segnale.
- **PitchShift**: sposta la tonalità del pianto di pochi semitoni.
- **Shift**: trasla il segnale nel tempo.
- **Gain**: varia il volume in decibel.
- **ClippingDistortion**: introduce una leggera distorsione.
- **PolarityInversion**: inverte la fase del segnale.

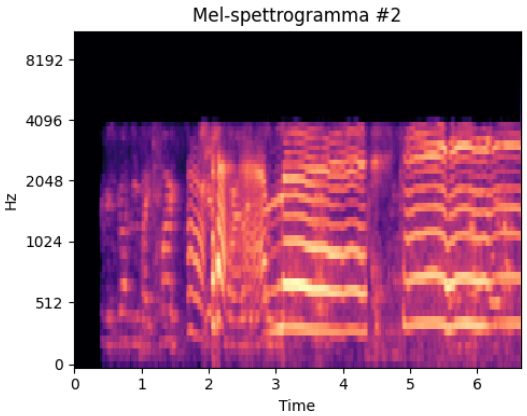

Segnale originale
Time

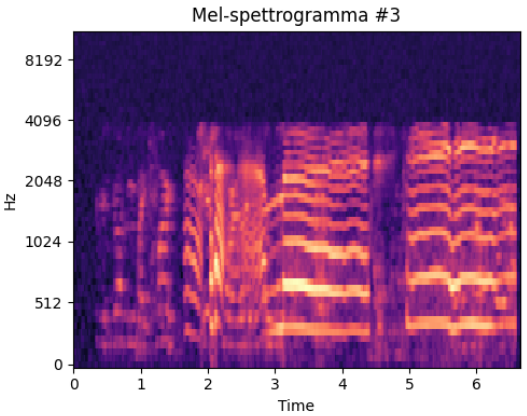

Audio augmentato #1
Time


Audio augmentato #2
Time


Audio augmentato #3
Time


Mel-spettrogramma #1
Hz
Time


Mel-spettrogramma #2
Hz
Time


Mel-spettrogramma #3
Hz
Time

Tecniche da combinare all'augmentation

- **Oversampling**: aumentare i campioni delle classi meno rappresentate per bilanciare il dataset.
- **Weighted Loss**: assegnare pesi diversi alle classi durante l'addestramento per ridurre il bias verso le classi maggioritarie.

Transfer learning

- Tecnica di apprendimento automatico in cui un modello addestrato su un compito viene riutilizzato per un compito simile. Dunque, sfrutta conoscenze già acquisite da modelli pre-addestrati.
- Principali approcci:
  - **Feature Extraction:** si usa il modello come estrattore di caratteristiche, addestrando solo il classificatore finale (è più utile con pochi esempi).
  - **Fine-Tuning:** si riaddestrano parzialmente o totalmente i pesi del modello sui nuovi dati (più efficace con molti dati).
- Nel **transfer learning audio**, modelli pre-addestrati (come PANNs) vengono utilizzati come estrattori di caratteristiche: sono già stati addestrati su grandi dataset di suoni e hanno imparato rappresentazioni generiche dei segnali acustici.

Transformer

- Modello neurale basato sull'attenzione che elabora intere sequenze in parallelo, imparando relazioni complesse tra gli elementi.
- Possono elaborare sequenze audio senza vincoli di passo fisso.
- Utilizzati nel riconoscimento vocale, classificazione e analisi di suoni, sintesi vocale e generazione di audio.
- Vantaggi:
  - Migliore gestione di sequenze lunghe.
  - Possibilità di usare embedding di alto livello, come log-Mel spettrogrammi, come input.

Implementazione reti neurali

Reti neurali provate

- Sono stati testati diversi tipi di reti neurali, ma nessuna di queste ha raggiunto prestazioni soddisfacenti (alcune leggermente migliori di altre).
- La principale limitazione è il dataset: i pochi esempi e il test set ridotto rendono difficile valutare le reali capacità della rete.
- Metodi sperimentati:
  - CNN multiclasse
  - CNN binaria
  - CNN + BiLSTM (RNN)
  - Transfer learning
  - Transformer
- Tra queste, ci siamo concentrate su quelle che sembravano più promettenti.

Classificazione multiclasse - binaria

- I risultati in multiclasse sono scarsi: per le classi non hungry ci sono pochissimi esempi (1–2 nel test), quindi è difficile valutare la recall e capire se il modello generalizza.
  D'altra parte, se togliamo troppi esempi per il test, ne restano pochi per il train e la validation, e il modello non riesce comunque a generalizzare.
- Abbiamo valutato quindi di trattare il problema come binario (hungry vs non-hungry): il dataset resta sbilanciato ma in questo modo la classe minoritaria ha più esempi rispetto al caso multiclasse.
- Tuttavia, la classe non hungry (tired, burping, belly pain, discomfort) è molto eterogenea; per questo il problema è stato inizialmente trattato in multilcasse e solo in seguito è stato effettuato un collasso binario finale.

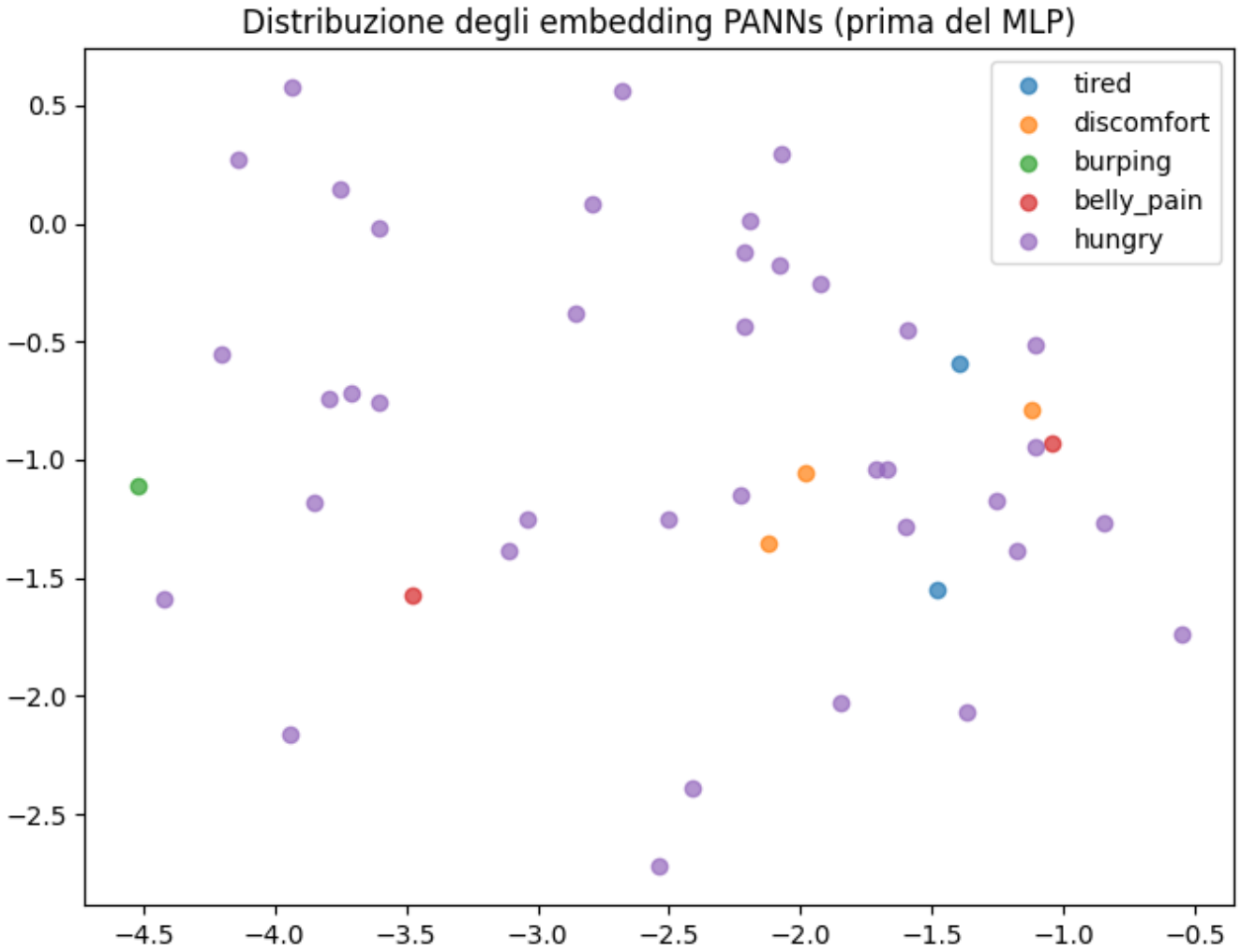
1° rete neurale: Transfer learning con PANNs + classificatore MLP

Descrizione rete neurale

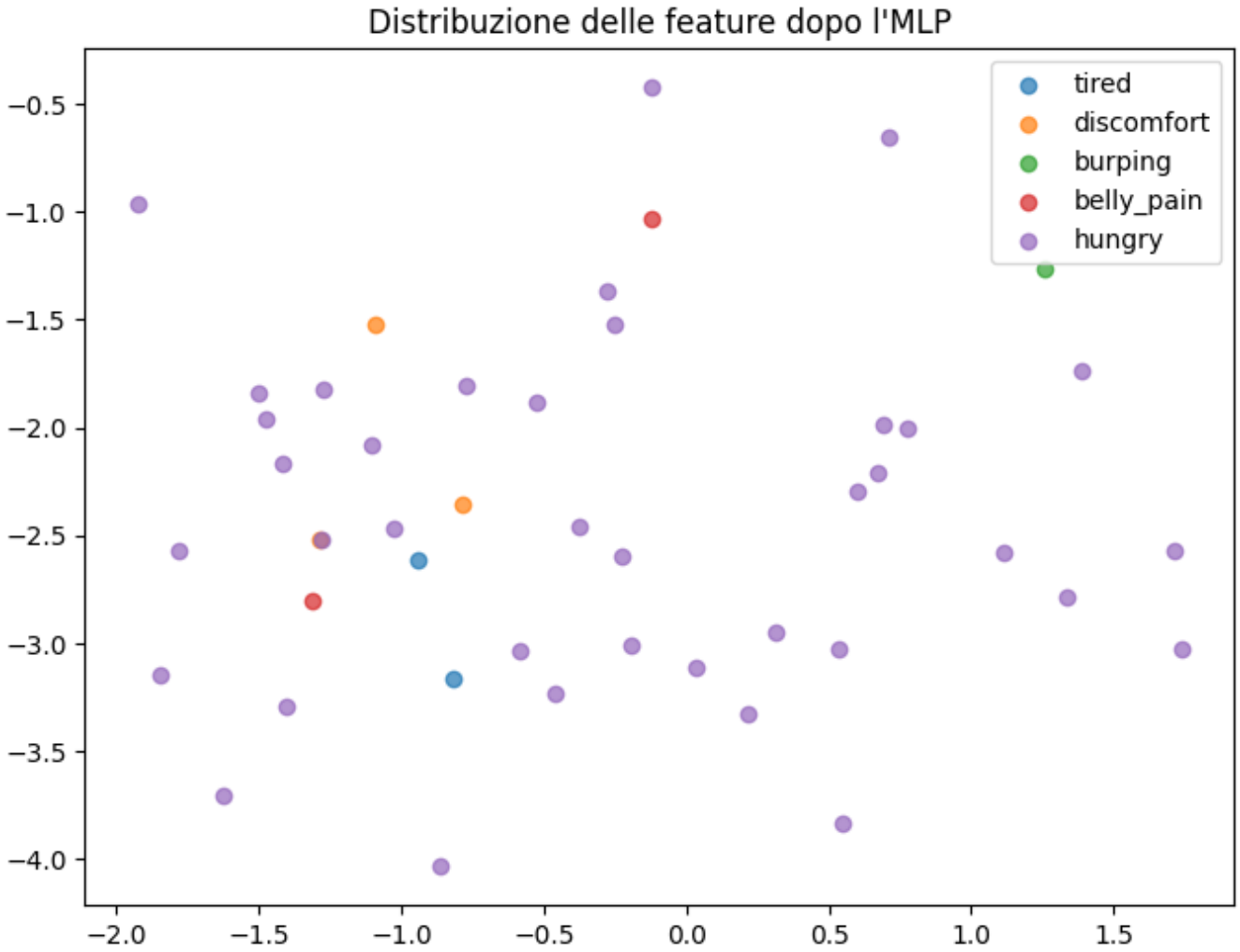
- Split dataset: 75% - 15% - 10%.
- Data augmentation on-the-fly.
- **Transfer learning**: estrattore di embeddings con PANNs (CNN pre-addestrata addestrata su AudioSet).
- Classificatore **MLP**: piccola rete feedforward fully connected
  - Input: embedding 2048D.
  - Hidden layer: 256 neuroni (BatchNorm + ReLU + Dropout).
  - Output layer: n_classes (5) neuroni.

- Ottimizzatori e regolarizzazione:
  - Loss function: Cross Entropy
  - Ottimizzatore: Adam
  - Scheduler: ReduceLROnPlateau (riduce il learning rate se val_loss non migliora).
  - Early stopping: ferma l'addestramento se non migliora più (previene l'overfitting).
  - Dropout + BatchNorm: migliorano generalizzazione e stabilità dell'apprendimento.

Distribuzione embedding PANNs prima di MLP



Distribuzione embedding PANNs dopo MLP



Risultati ottenuti

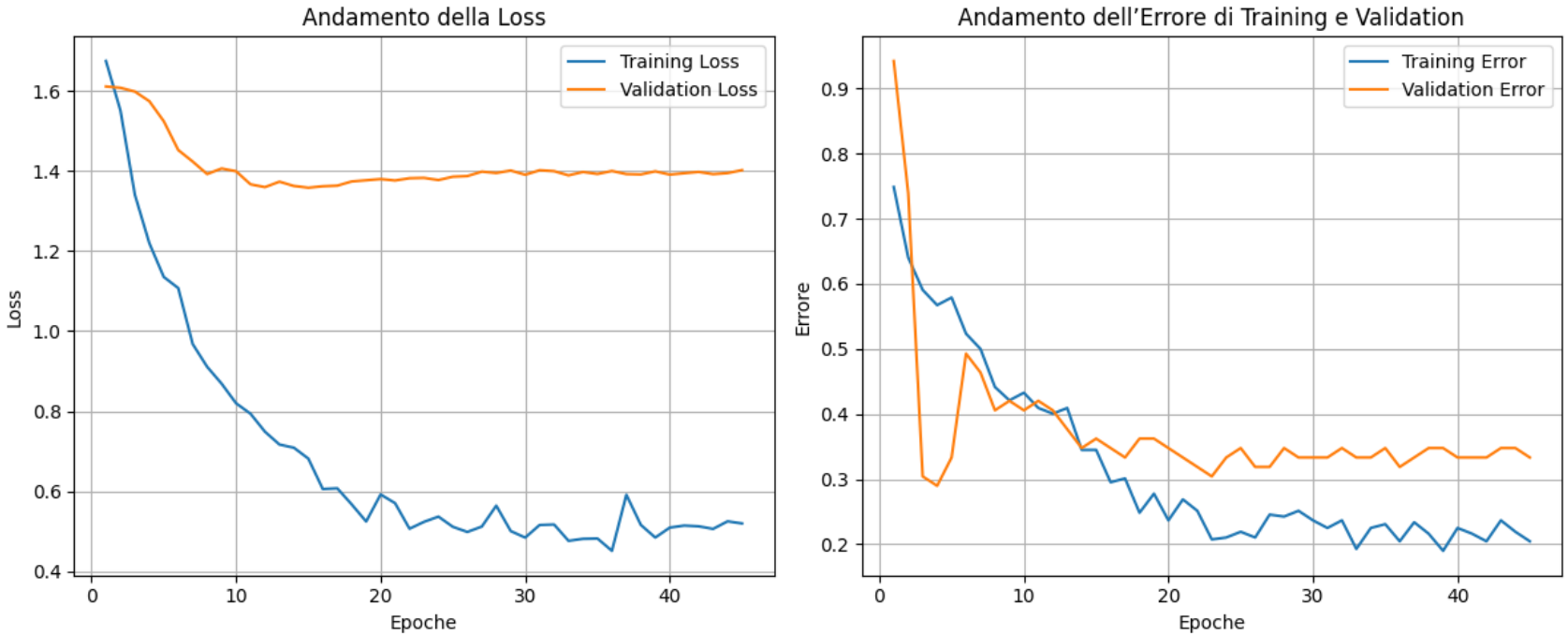


Figura: Andamento della loss (sinistra) e dell'errore (destra)

Epoch	Train Loss	Val Loss	Train Acc	Train Recall	Val Acc	Val Recall
45/200	0.5197	1.4017	0.7953	0.9406	0.6667	0.3948

Matrici di confusione

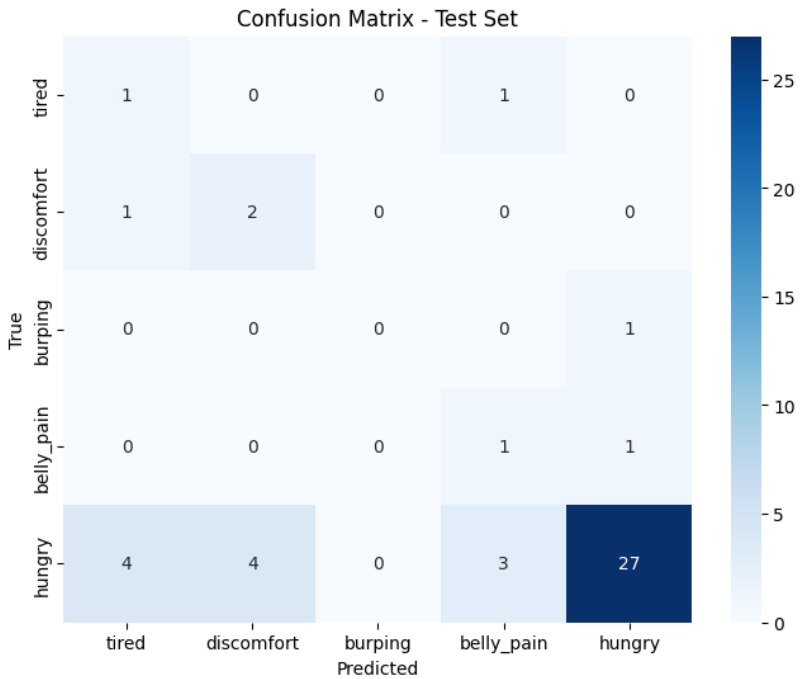


Figura: Classificazione multiclasse.

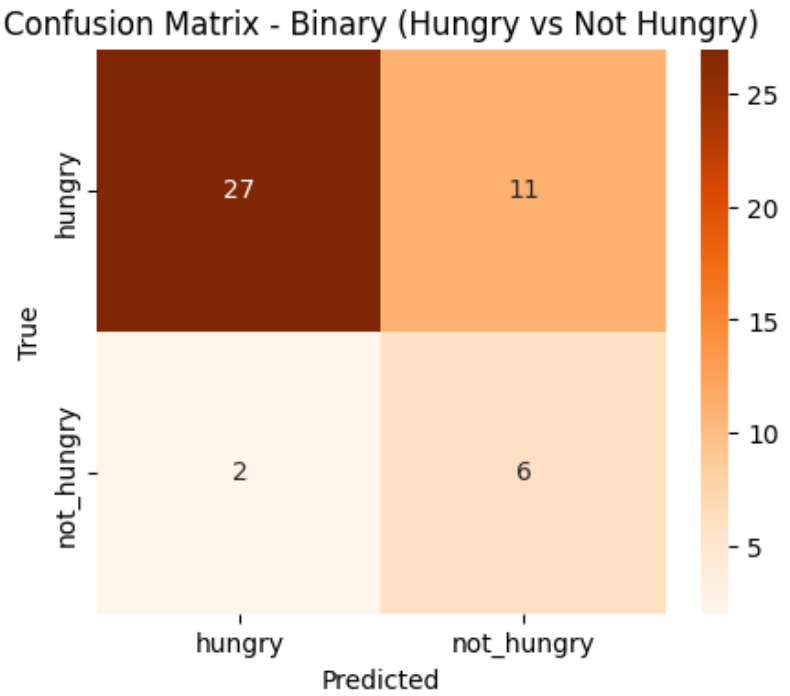


Figura: Classificazione binaria.

Metriche di performance (caso multiclasse)

Classe	**Precision**	**Recall**	**F1-score**	**Support**
tired	0.17	0.50	0.25	2
discomfort	0.33	0.67	0.44	3
burping	0.00	0.00	0.00	1
belly_pain	0.20	0.50	0.29	2
hungry	0.93	0.71	0.81	38
Accuracy			0.67	
Macro Avg	0.33	0.48	0.36	46
Weighted Avg	0.81	0.67	0.72	46

Tabella: Classificazione multiclasse.

Metriche di performance (caso binario)

Classe	Precision	Recall	F1-score	Support
hungry	0.93	0.71	0.81	38
not_hungry	0.35	0.75	0.48	8
Accuracy			0.72	
Macro Avg	0.64	0.73	0.64	46
Weighted Avg	0.83	0.72	0.75	46

Tabella: Classificazione binaria.

2° rete neurale: CNN + Transformer + classificatore MLP

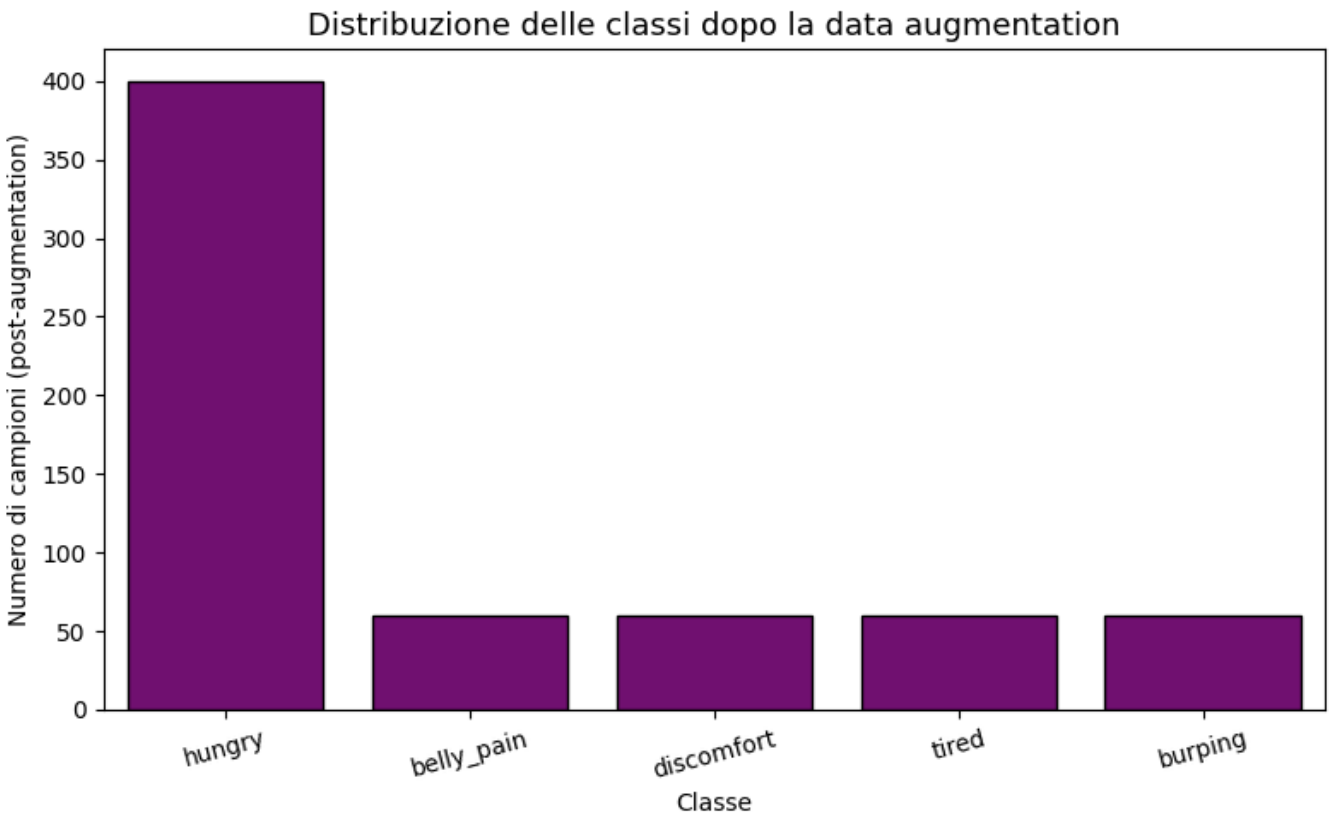
Descrizione rete neurale

- Split dataset: 75% - 15% - 10%.
- Preprocessing: precomputazione dei log-mel-spettrogrammi + augmentation offline.
- Estrattore di feature: CNN + Transformer.
  - **CNN**: 3 blocchi Convolution (BatchNorm, ReLU, MaxPool).
  - Output CNN: ridimensionati in tokens per il Transformer.
  - **Transformer Encoder**: 1 layer, 2 teste di attenzione, modellazione delle dipendenze tra feature.
- Classificatore finale **MLP**:
  - Input: embedding derivato dal Transformer.
  - Hidden layer: dimensione dimezzata con ReLU + Dropout.
  - Output layer: n_classes neuroni (5 classi).

Descrizione rete neurale

- Ottimizzazione e regolarizzazione:
  - Loss function: CrossEntropy.
  - Ottimizzatore: AdamW.
  - Scheduler: ReduceLROnPlateau sul F1 macro.
  - Early stopping: ferma l'addestramento se val_F1 non migliora.
  - Dropout + BatchNorm: migliorano la generalizzazione e stabilità dell'apprendimento.

Distribuzione delle classi dopo data augmentation offline



Risultati ottenuti

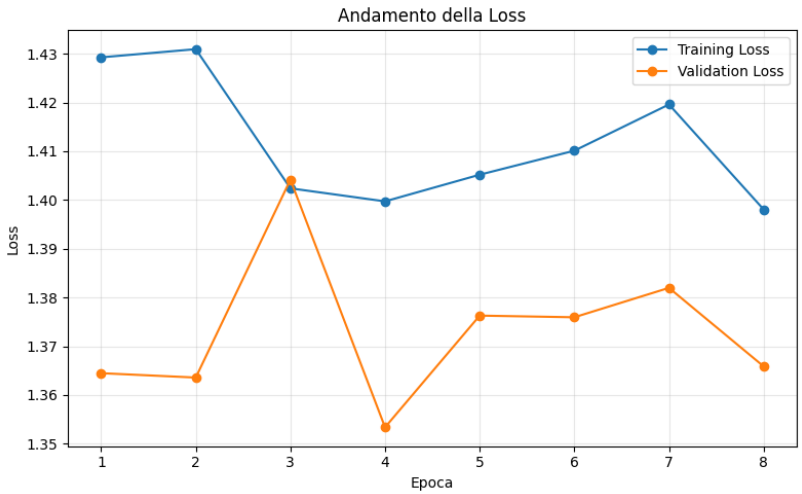


Figura: Andamento della loss

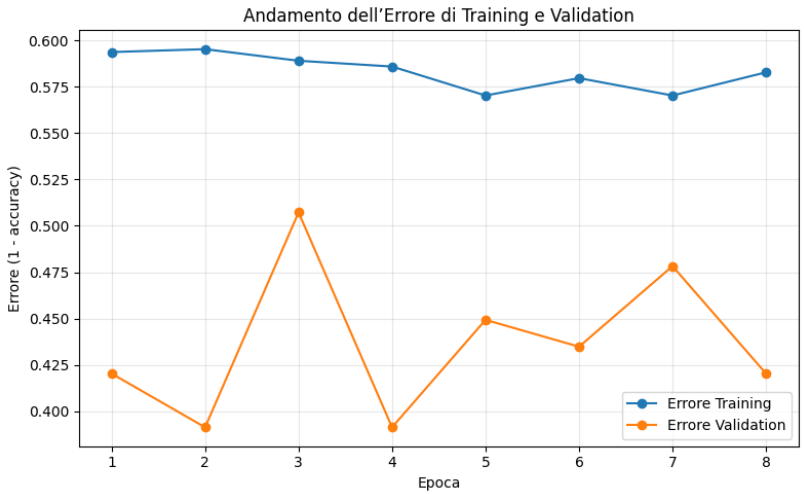


Figura: Andamento dell'errore

Epoch	Train Loss	Val Loss	Train Acc	Val Acc	Val F1
2/30	1.4309	1.3636	0.4047	0.6087	0.2324

Matrici di confusione

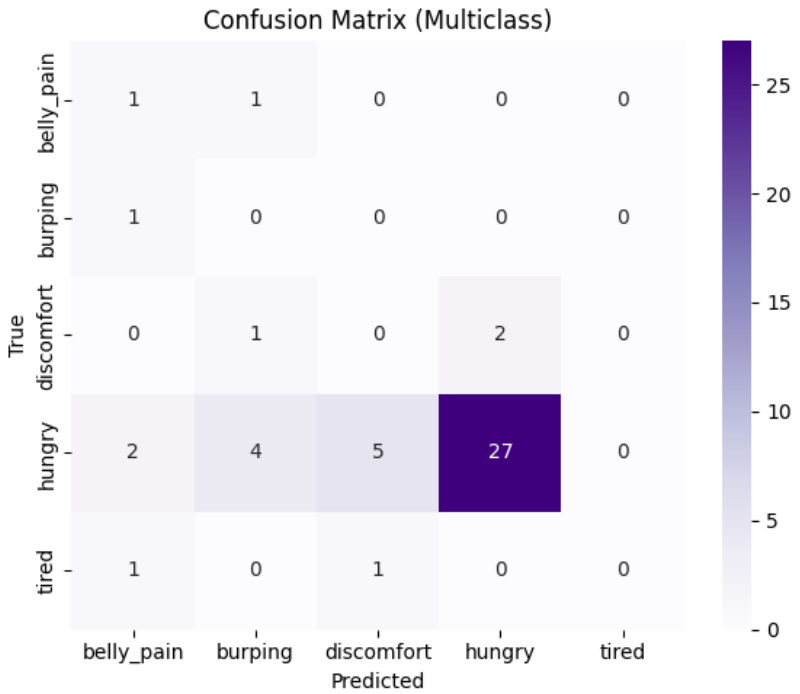


Figura: Classificazione multiclasse.

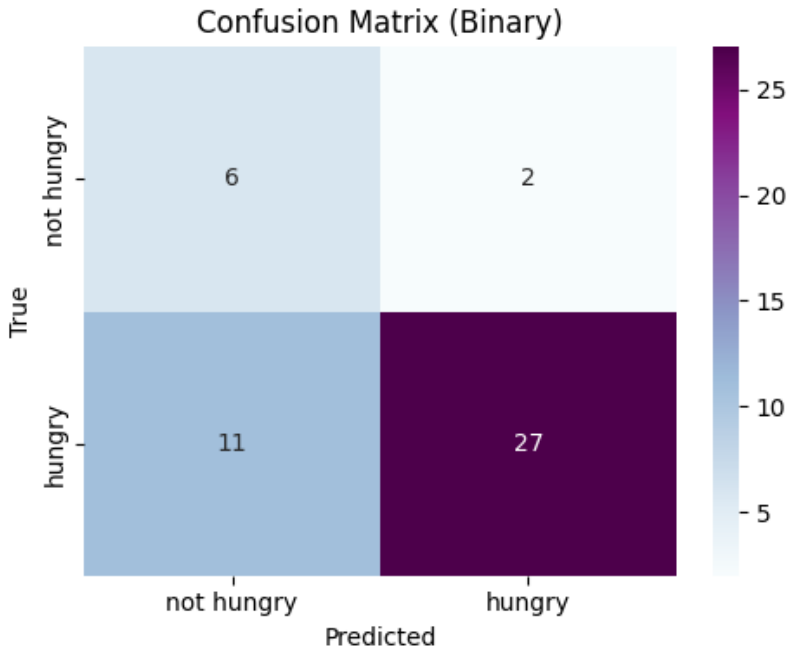


Figura: Classificazione binaria.

Metriche di performance (caso multiclasse)

Classe	Precision	Recall	F1-score	Support
tired	0.00	0.00	0.00	2
discomfort	0.00	0.00	0.00	3
burping	0.00	0.00	0.00	1
belly_pain	0.20	0.50	0.29	2
hungry	0.93	0.71	0.81	38
Accuracy			0.61	46
Macro Avg	0.23	0.24	0.22	46
Weighted Avg	0.78	0.61	0.68	46

Tabella: Classificazione multiclasse.

Metriche di performance (caso binario)

Classe	Precision	Recall	F1-score	Support
hungry	0.93	0.71	0.81	38
not_hungry	0.35	0.75	0.48	8
Accuracy			0.72	
Macro Avg	0.64	0.73	0.64	46
Weighted Avg	0.83	0.72	0.75	46

Tabella: Classificazione binaria.

3° rete neurale: Transfer learning + Transformer classifier + K-fold validation

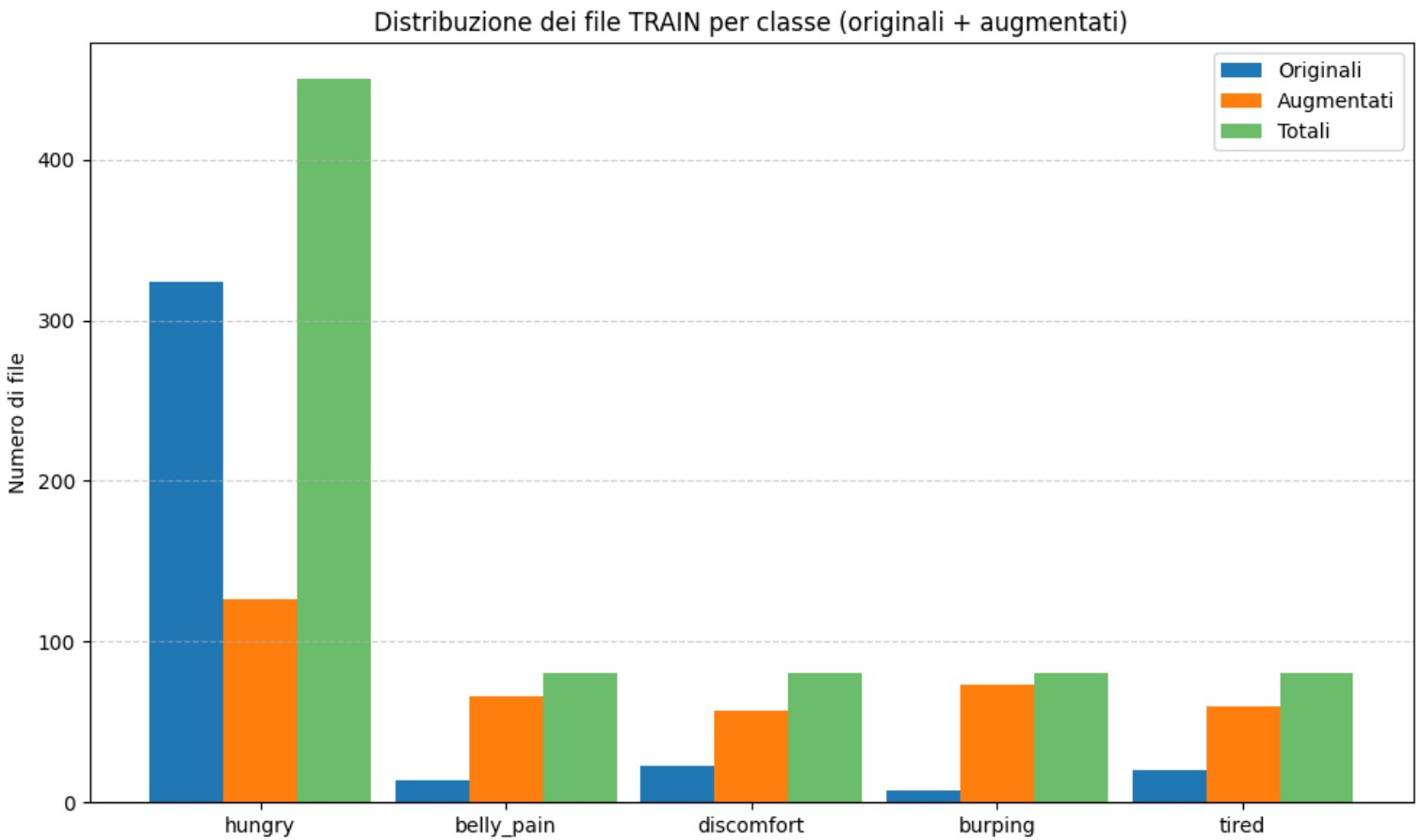
Descrizione rete neurale

- Split dataset: 85% - 15%
- Data augmentation offline.
- **Transfer learning**: estrattore di embeddings con PANNs.
- Transformer Classifier:
  - Input: Layer lineare $2048 \rightarrow 256$.
  - Encoder: 2 layer, 4 head attention, feed-forward, dropout 0.5.
  - Pooling: media sui token (global mean pooling).
  - Output: layer lineare per classificazione.

Addestramento e validazione

- K-fold validation ($k = 7$):
  - Ogni fold usa un sottoinsieme diverso per la validazione.
  - Fornisce una stima più affidabile delle prestazioni medie.
- Ottimizzazione e regolarizzazione:
  - WeightedRandomSampler per bilanciare le classi nel training.
  - Class weights nella CrossEntropyLoss.
  - Ottimizzatore: Adam.
  - Scheduler: ReduceLROnPlateau.
  - Early stopping.

Distribuzione classi prima e dopo data augmentation offline



Risultati ottenuti

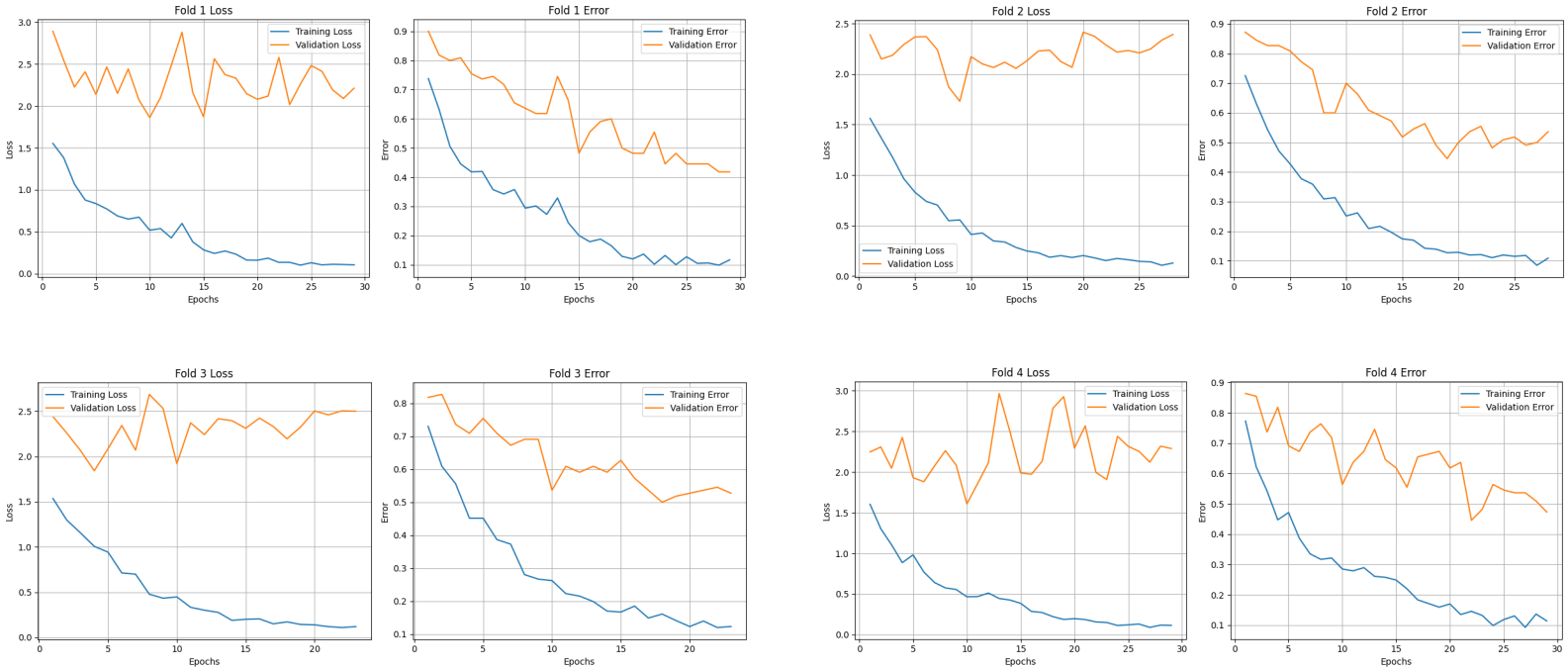
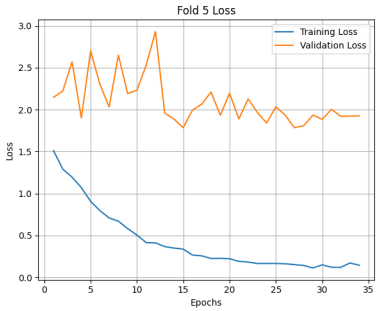
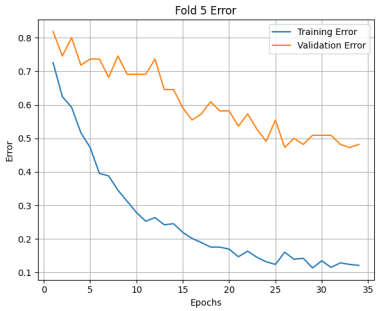
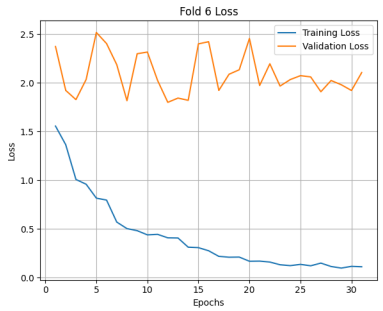
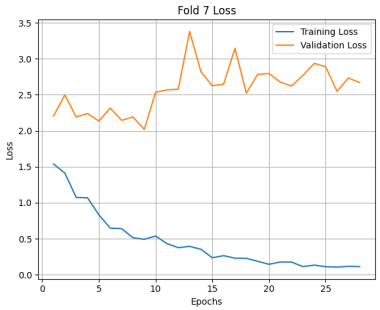


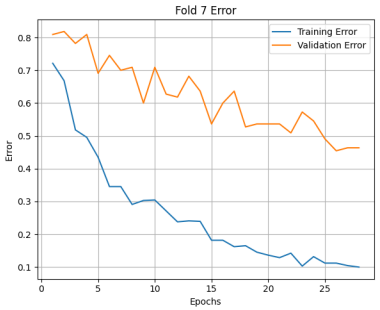
Figura: Andamento della loss e dell'errore per ogni fold.







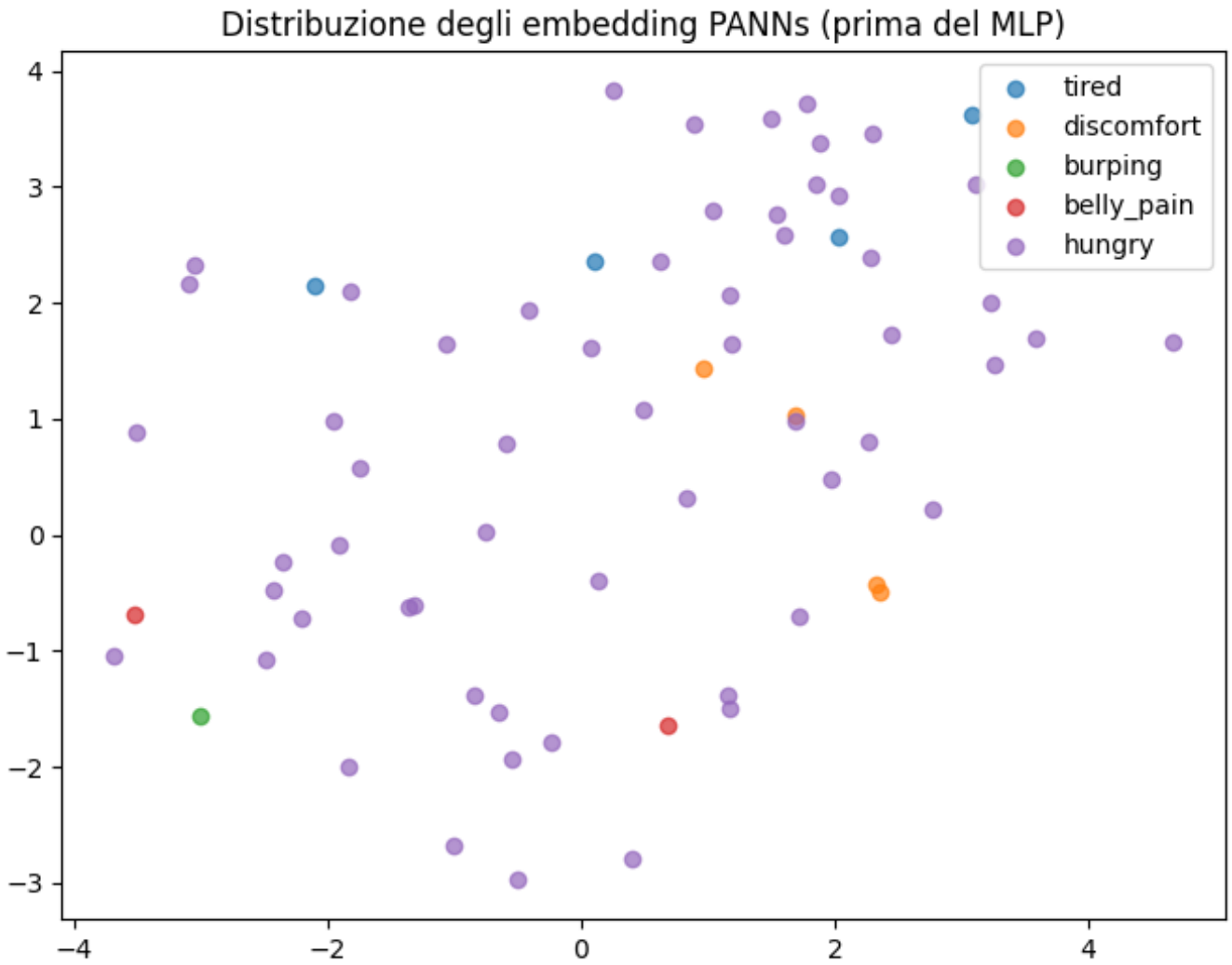




Metrica	**Valore medio**
Accuracy	0.5091
Recall	0.5966

Tabella: Risultati medi della K-Fold Cross-Validation

Distribuzione embedding PANNs



Matrici di confusione

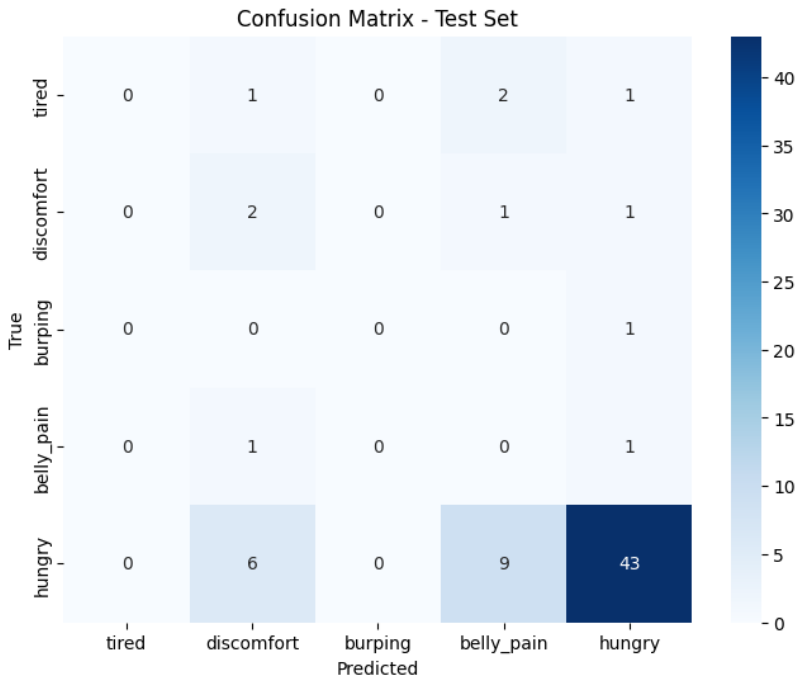


Figura: Classificazione multiclasse.

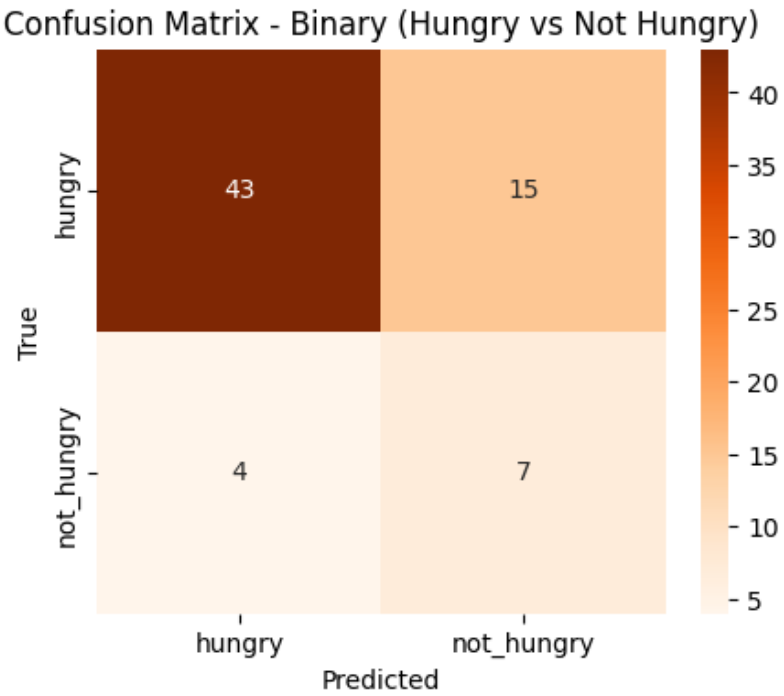


Figura: Classificazione binaria.

Metriche di performance (caso multiclasse)

Classe	**Precision**	**Recall**	**F1-score**	**Support**
tired	0.00	0.00	0.00	4
discomfort	0.20	0.50	0.29	4
burping	0.00	0.00	0.00	1
belly_pain	0.00	0.00	0.00	2
hungry	0.91	0.74	0.82	58
Accuracy			0.65	
Macro Avg	0.22	0.25	0.22	69
Weighted Avg	0.78	0.65	0.71	69

Tabella: Classificazione multiclasse.

Metriche di performance (caso binario)

Classe	Precision	Recall	F1-score	Support
hungry	0.91	0.74	0.82	58
not_hungry	0.32	0.64	0.42	11
Accuracy			0.72	
Macro Avg	0.62	0.69	0.62	69
Weighted Avg	0.82	0.72	0.76	69

Tabella: Classificazione binaria.

Considerazioni generali e possibili miglioramenti

Considerazioni generali sui risultati delle reti

- Prestazione massima raggiunta: discreta solo nel task binario, comunque non soddisfacente.
- Problema principale: dataset piccolo e sbilanciato.
- Overfitting evidente: il modello si adatta troppo ai dati di training.
- La performance sul validation set non migliora.

Possibili miglioramenti

- **Aumentare la quantità di dati**: raccogliere più esempi o svolgere data augmentation avanzata o generativa.
- Ottimizzare l'architettura della rete: testare diverse combinazioni di layer, dimensione degli embedding e dropout.
- Sperimentare con i parametri di training: testare diverse combinazioni di learning rate, batch size e optimizer.