RELAZIONE FINALE

Rilevamento di anomalie nel battito cardiaco

mediante Image Classification

Gruppo:

Di Biasi Yuri	0001043300
Marrone Christian	0001059243
Salzano Giuseppe	0001059232

Sommario

1. Abstract

Nell'attuale era digitale, l'ambito sanitario è sempre più influenzato dalle tecnologie emergenti, che promettono di migliorare la diagnosi e la gestione delle condizioni mediche.

In questo contesto, il presente progetto universitario si colloca come un esempio di come l'apprendimento automatico e l'elaborazione delle immagini possano essere applicati all'analisi del ritmo cardiaco. L' obiettivo principale dell'applicazione è fornire agli utenti una valutazione rapida e accurata dello stato del loro sistema cardiaco in base a un elettrocardiogramma (ECG) acquisito. La classificazione avviene in quattro categorie principali:

- battito cardiaco normale
- battito cardiaco anormale
- previsione infarto
- infarto pregresso

Il progetto è stato sviluppato utilizzando una combinazione di tecnologie, tra cui l'addestramento di una rete neurale convoluzionale (CNN) su un dataset ECG preesistente, l'esportazione del modello in formato TFLite e l'integrazione dell'applicazione su Android Studio per l'uso su dispositivi mobili.

Il dataset utilizzato per l'addestramento del modello è stato raccolto da fonti affidabili sul web e la sua diversità e la sua dimensione hanno consentito di ottenere un modello robusto e accurato.

L'architettura del modello è stata progettata specificamente per affrontare il problema della classificazione dell'ECG. Il modello risultante è stato convertito in formato TFLite per garantire l'efficienza computazionale e la portabilità su dispositivi mobili.

L'integrazione dell'applicazione su Android Studio consente di acquisire un ECG attraverso un dispositivo Android e ottenere una valutazione istantanea dello stato cardiaco. L'interfaccia utente è stata progettata per essere intuitiva e facile da usare.

2. Dataset

2.1 Origine del Dataset

Il dataset utilizzato in questo progetto è stato recuperato dalla piattaforma Kaggle, una rinomata comunità online per la condivisione di dataset e competizioni legate all'apprendimento automatico.

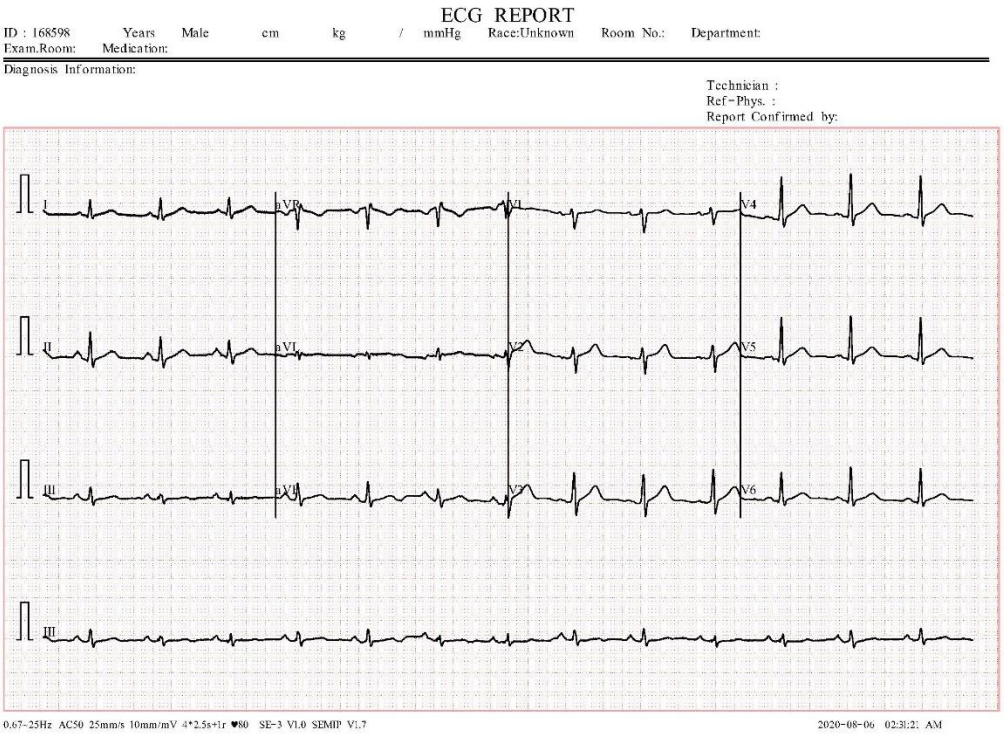
2.2 Descrizione del Dataset

Il dataset è composto da un totale di **1376 registrazioni ECG** in formato digitale. Ogni registrazione rappresenta un'onda cardiaca registrata e salvata come immagine in scala di grigi con una risoluzione di **512x512 pixel**. Le immagini sono state fornite in formato PNG.

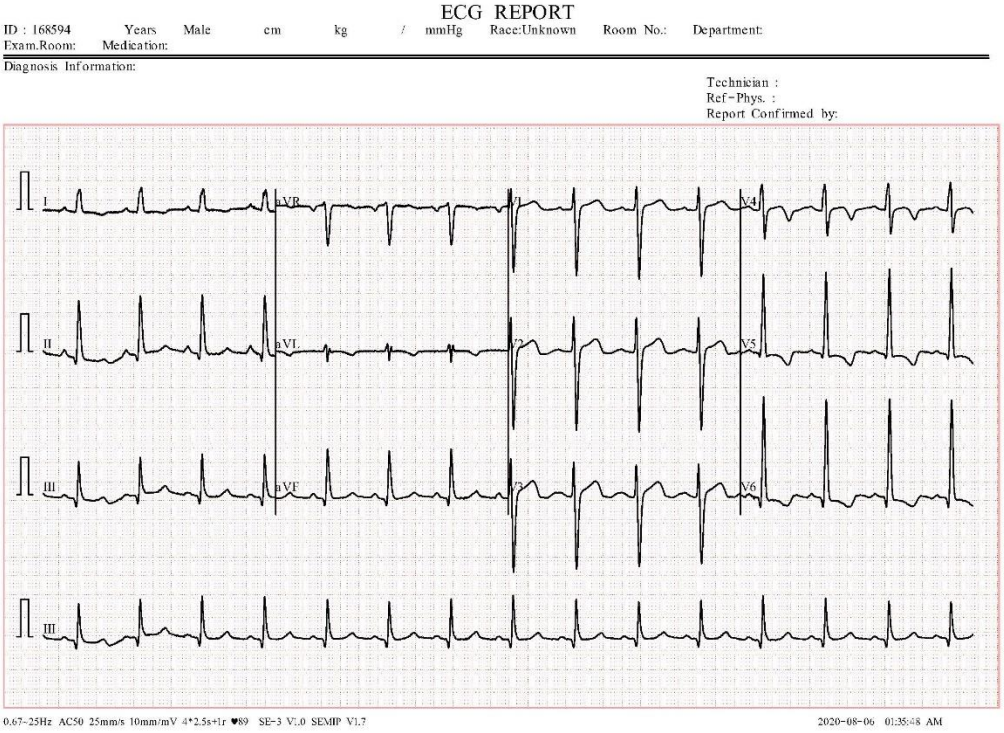
2.3 Etichettatura delle Immagini

Le immagini ECG nel dataset sono state etichettate manualmente da un team di esperti medici. Ogni registrazione è stata assegnata a una delle quattro classi principali:

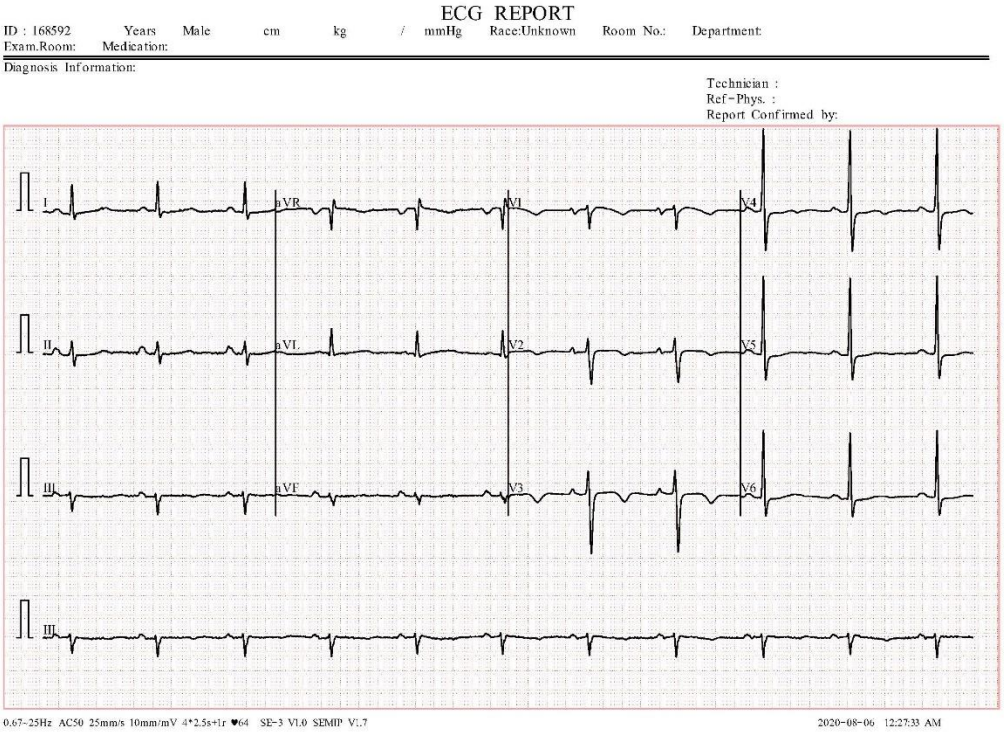
- **Battito Normale**: Questa classe rappresenta gli ECG con ritmo cardiaco normale.



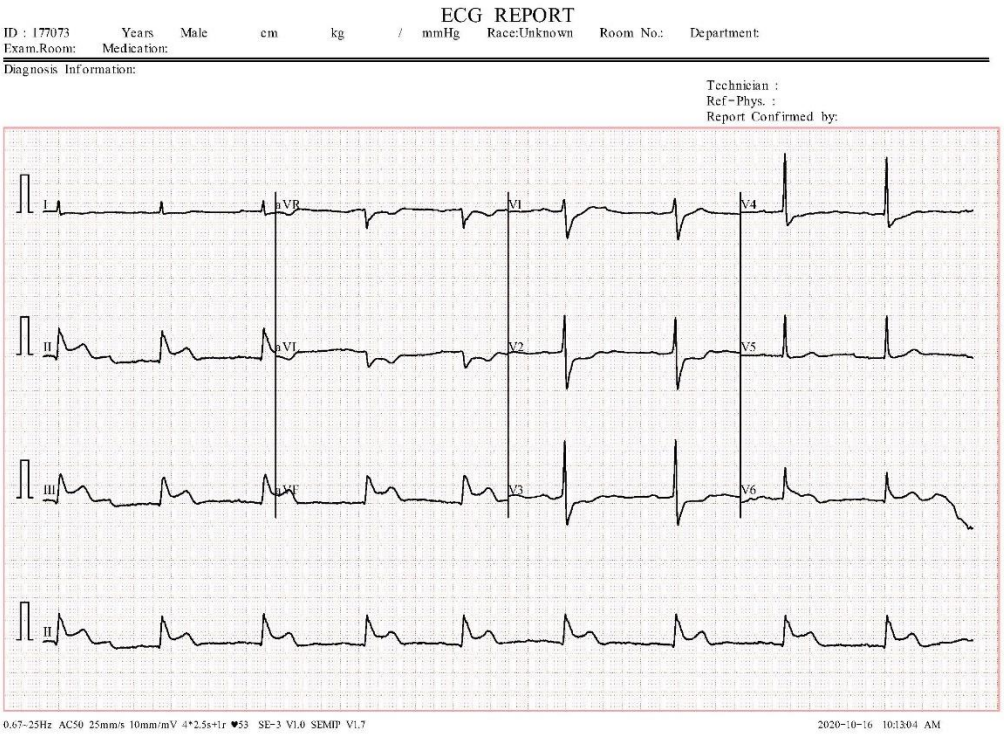
- **Anormale**: Include ECG con segni di anomalie cardiache diverse da un infarto.



- **Infarto pregresso**: Questa classe rappresenta ECG di pazienti che hanno avuto un infarto miocardico in passato.



- **Infarto in corso**: Questa classe identifica ECG di pazienti attualmente affetti da un infarto miocardico.



2.4 Distribuzione delle Classi

Di seguito è riportata una tabella che illustra la distribuzione delle classi nel dataset:

Classe	**Numero di ECG**
Battito Normale	351 (239, 112)
Anormale	345 (233, 112)
Infarto in Passato	284 (172, 112)
Infarto in Corso	396 (284, 112)

2.5 Divisione del Dataset

La parte di “train” del dataset è stata suddivisa in modo conveniente per l'addestramento e la valutazione del modello:

- **Training Set**: 90% delle registrazioni ECG per l'addestramento del modello.
- **Validation Set**: 10% delle registrazioni ECG per la validazione durante l'addestramento.

2.6 Riferimenti

Il dataset ECG è stato acquisito dal seguente riferimento su Kaggle:
https://www.kaggle.com/datasets/jayaprakashpondy/ecgimages?select=train

3. Architettura e addestramento del modello

Considerando la mole di dati a nostra disposizione abbiamo deciso di sviluppare e addestrare una rete neurale convoluzionale (CNN) da zero, facendo uso dell'API Keras integrata in TensorFlow. Il nostro modello segue una struttura tradizionale ampiamente riconosciuta nell'ambito della classificazione di immagini.

3.1 Partizionamento del Dataset

Per la creazione dei dataset, abbiamo utilizzato la libreria TensorFlow (tf) e la funzione **image_dataset_from_directory**. In particolare, abbiamo creato due dataset distinti: **train_ds** e **val_ds** per l'addestramento e la convalida del modello, e **test_ds** per il test finale.

Il dataset di addestramento (**train_ds**) è stato creato a partire dalla directory "train" e include immagini con dimensioni di 512x512 pixel. Inoltre, abbiamo specificato un **batch_size** di 16, che significa che durante l'addestramento verranno processate 16 immagini alla volta. Abbiamo anche impostato una divisione di convalida del 10% (validation_split = 0.1), il che significa che il 10% del dataset di addestramento verrà utilizzato per la cross-validatio. Inoltre, abbiamo specificato che sia le immagini di addestramento che di convalida verranno trattate come immagini in scala di grigi (color_mode = "grayscale") e abbiamo utilizzato un seed (variabile **seed**) per la riproducibilità dei risultati.

```
img_height, img_width = 512, 512
batch_size = 16


train_ds, val_ds = tf.keras.utils.image_dataset_from_directory(directory = "train",
                                                               image_size = (img_height, img_width),
                                                               batch_size = batch_size,
                                                               validation_split = 0.1,
                                                               subset = "both",
                                                               color_mode = "grayscale",
                                                               seed = seed
                                                               )
```

Il dataset di test (**test_ds**) è stato creato a partire dalla directory "test" e ha lo stesso **batch_size** e modalità di colore in scala di grigi.

```
test_ds = tf.keras.utils.image_dataset_from_directory(directory = "test",
                                                      image_size = (img_height, img_width),
                                                      batch_size = batch_size,
                                                      color_mode = "grayscale")
```

Questi dataset saranno utilizzati come input per il nostro modello di machine learning, consentendoci di addestrare il modello sui dati di addestramento, convalidarlo sui dati di convalida e, infine, testarlo sui dati di test per valutarne le prestazioni. La divisione dei dati in dataset di addestramento, convalida e test è una pratica comune nella costruzione di modelli di machine learning per valutare l'efficacia del modello su dati precedentemente non visti.

3.2 Architettura della rete

Iniziamo con un Input layer, che prende in ingresso immagini di dimensioni fisse pari a 512x512 pixel. Considerando la natura del dataset, non è stato possibile lavorare con immagini di dimensione inferiore poiché ne avrebbero risentito le prestazioni del modello. Successivamente, applichiamo uno strato di normalizzazione, al fine di riscalare i valori dei pixel in un intervallo compreso tra 0 e 1. Tale procedura contribuisce a garantire una maggiore stabilità durante l'addestramento.

```
model = Sequential(
    [
        # Input layer.
        Input(shape = (img_height, img_width, 1)),
        Rescaling(1./255),
```

Proseguiamo con un Conv2D layer che applica 16 filtri di dimensione 3x3 e come activation function ReLU. Questo layer è seguito da un max pooling 2x2, che consente di ridurre la dimensione dell'immagine. Questo schema di strati Conv2D e max pooling si ripete per un totale di quattro iterazioni, e il numero di filtri aumenta progressivamente, raggiungendo 32 per il secondo Conv2D layer e 64 per gli ultimi due.

```
# Convolutions with subsequent pooling.
Conv2D(filters = 16, kernel_size = (3, 3), activation = "relu", padding = "same"),
MaxPooling2D((2, 2)),
Conv2D(filters = 32, kernel_size = (3, 3), activation = "relu", padding = "same"),
MaxPooling2D((2, 2)),
Conv2D(filters = 64, kernel_size = (3, 3), activation = "relu", padding = "same"),
MaxPooling2D((2, 2)),
Conv2D(filters = 64, kernel_size = (3, 3), activation = "relu", padding = "same"),
MaxPooling2D((2, 2)),
```

Successivamente, introduciamo un flatten layer per convertire i dati in un vettore unidimensionale. Infine, applichiamo un dense layer con 128 neuroni, che precede l'ultimo Dense layer composto da quattro parametri. Quest'ultimo ha come activation function softmax, che genera una distribuzione di probabilità per la predizione finale in base alle quattro classi specificate.

```
# Classification head.
Flatten(),
Dense(units = 128, activation = "relu"),
Dense(units = n_classes, activation = "softmax"),
```

Ricordiamo che i primi layer, ovvero dove avvengono le convoluzioni, svolgono un ruolo fondamentale nell'estrazione delle caratteristiche dalle immagini, mentre i Dense layers sono responsabili della fase di classificazione. L'uso di ReLU, come activation function, è necessario per introdurre non linearità, permettendo al modello di apprendere relazioni complesse tra le caratteristiche estratte e le classi target.
Il modello ha dimostrato notevoli prestazioni, raggiungendo un'accuracy superiore al 90% in meno di 20 epoche di addestramento. Questo risultato è stato ottenuto grazie all'implementazione di

una strategia di early stopping, la quale è stata configurata per interrompere l'addestramento qualora l'accuracy non fosse migliorata ulteriormente nelle dieci epoche successive alla miglior performance registrata.

Infine, il modello è stato convertito in TensorFlow Lite (TFLite) al fine di poter proseguire con lo sviluppo dell'applicazione Android.

4. Portabilità su Android Studio

Come primo passo nel processo di portabilità del modello su un dispositivo mobile, abbiamo proceduto alla sua conversione in formato TFLite.

4.1 Conversione del Modello in TFLite

La conversione del modello è stata eseguita utilizzando Tensorflow Lite, una libreria che offre una soluzione efficiente per eseguire modelli di machine learning su dispositivi embedded. Il seguente codice rappresenta il processo mediante il quale il modello addestrato con Keras è stato convertito nel formato .tflite, rendendolo così pronto per essere utilizzato su un dispositivo mobile. Questa conversione è essenziale per garantire che il modello sia ottimizzato per le risorse limitate dei dispositivi mobili, garantendo al contempo prestazioni efficienti e accurate nella classificazione dei dati ECG.

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

4.2 Integrazione del Modello in Android Studio

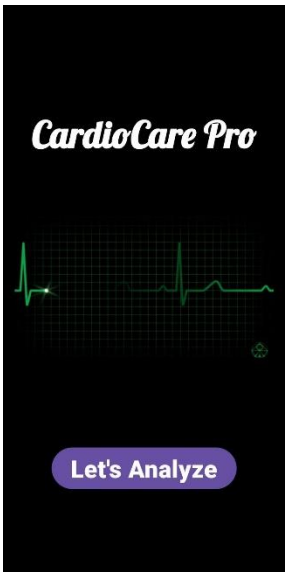
Dopo aver completato la conversione del modello in formato TFLite, il passo successivo ha comportato l'integrazione di questo modello TFLite in un'applicazione Android sviluppata utilizzando Android Studio. Per realizzare l'applicazione finale, abbiamo sfruttato Java 17.0.2. Questo processo di integrazione è stato essenziale per consentire agli utenti di acquisire dati ECG tramite i loro dispositivi mobili e ottenere una valutazione istantanea del loro stato cardiaco.

4.3 Interfaccia Utente User-Friendly

L'interfaccia dell'applicazione è stata progettata con una particolare attenzione alla user-friendliness e all'intuitività, per garantire un'esperienza utente senza soluzione di continuità.

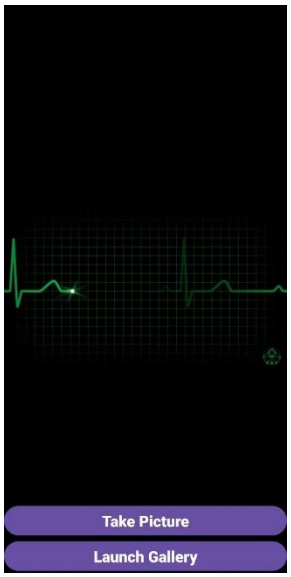
L'interfaccia è divisa in diverse schermate chiave:

1. **Schermata Iniziale:** L'esperienza inizia con una schermata iniziale che fornisce agli utenti un accesso rapido alla funzionalità principale dell'applicazione. Un semplice tocco sul pulsante “Let’s Analyze” posto nella schermata iniziale consente di accedere alla pagina principale dell'applicazione.

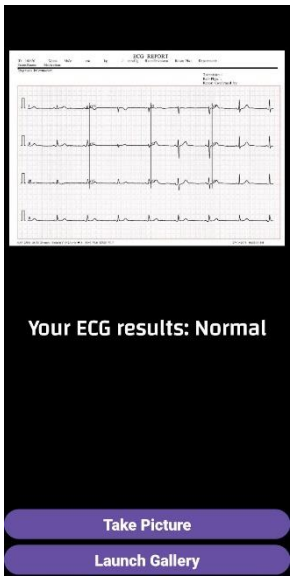


2. **Pagina Principale:** La pagina principale dell'applicazione è il fulcro dell'interazione. Qui, gli utenti si trovano di fronte a due opzioni chiave, rendendo l'interfaccia estremamente intuitiva. Due pulsanti chiaramente etichettati sono messi a disposizione degli utenti per guidarli attraverso il processo:

- **Pulsante "Take Picture":** Il secondo pulsante permette agli utenti di catturare una foto ECG direttamente tramite la fotocamera del dispositivo. Questa funzionalità offre una soluzione pratica e immediata per acquisire i dati ECG in tempo reale.
- **Pulsante "Launch Gallery":** Uno dei pulsanti consente agli utenti di caricare una foto ECG presente nella galleria del loro dispositivo. Questa opzione offre una maggiore flessibilità ai pazienti, consentendo loro di utilizzare dati ECG preesistenti per l'analisi.



3. **Pagina di visualizzazione del risultato:** Questa pagina mostra, successivamente al caricamento di un'immagine, quest'ultima ed il relativo risultato.



L'interfaccia intuitiva è stata progettata con l'obiettivo di rendere l'applicazione accessibile e facilmente utilizzabile da utenti di diverse competenze e conoscenze. Questo approccio favorisce l'uso dell'applicazione per scopi di monitoraggio cardiaco e diagnosi.

5. Conclusioni

Questo progetto dimostra come le tecnologie avanzate di image classification e machine learning possano essere utilizzate per sviluppare un'applicazione pratica e di valore nel campo della salute cardiovascolare. La combinazione di un modello di machine learning accurato con la portabilità su dispositivi mobili offre un'opportunità significativa per il monitoraggio cardiaco personale e la diagnosi precoce di anomalie cardiache.

5.1 Risultati chiave e conoscenze apprese

Durante il corso di questo progetto, abbiamo raggiunto i seguenti risultati:

- **Addestramento efficace**: Grazie all'utilizzo di un dataset di ECG ampio e accuratamente etichettato, siamo stati in grado di addestrare un modello di Deep Learning che ha dimostrato una buona capacità di classificazione.
- **Ampliamento conoscenze accademiche:** Questo progetto universitario ha contribuito alla nostra comprensione delle complesse dinamiche dell'apprendimento automatico applicato alla medicina. Ha offerto l'opportunità di applicare concetti teorici e metodi pratici in un ambiente di studio controllato.

- **Integrazione nell'app android**: Sviluppo di un'app Android che sfrutta il modello addestrato per fornire una soluzione pratica e accessibile per la classificazione degli ECG. L'app consente agli utenti di acquisire o caricare immagini degli ECG e ricevere una diagnosi istantanea.

5.2 Implicazioni e Futuro Sviluppo

Questo progetto presenta implicazioni e suggerimenti per sviluppi futuri:

- **Supporto Clinico**: L'applicazione potrebbe essere ulteriormente sviluppata per offrire supporto clinico in tempo reale, consentendo ai medici di effettuare diagnosi più rapide ed efficaci.
- **Espansione del Dataset**: L'aggiunta di ulteriori dati e informazioni cliniche potrebbe migliorare ulteriormente le prestazioni del modello.