Model Development Phase Template

Date	24 SEPTEMBER 2024
Team ID	SWTID1727151090
Project Title	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

```
# Import necessary libraries for model building
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
```

```
# Initialize the model
model = Sequential()
```

```
# Adding CNN layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))  # F
model.add(MaxPooling2D(pool_size=(2, 2)))  # Max pooling

model.add(Conv2D(64, (3, 3), activation='relu'))  # Second convolutional layer
model.add(MaxPooling2D(pool_size=(2, 2)))  # Max pooling

model.add(Conv2D(128, (3, 3), activation='relu'))  # Third convolutional layer
model.add(MaxPooling2D(pool_size=(2, 2)))  # Max pooling
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Flattening the layers
model.add(Flatten())

# Adding Hidden Layer
model.add(Dense(128, activation='relu'))  # Fully connected hidden layer
model.add(Dropout(0.5))  # Dropout layer to prevent overfitting
```

```
# Adding Output Layer
model.add(Dense(training_set.num_classes, activation='softmax'))
```

```
# Configure the Learning Process
model.compile(optimizer='adam',
              loss='categorical_crossentropy',  # Use categorical_crossentropy f
              metrics=['accuracy'])
```

```
# Using EarlyStopping to avoid overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten (Flatten)	(None, 86528)	0
dense (Dense)	(None, 128)	11,075,712
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 2)	258

Total params: 11,169,218 (42.61 MB)
Trainable params: 11,169,218 (42.61 MB)
Non-trainable params: 0 (0.00 B)

```
# Train model
history = model.fit(
    training_set,
    validation_data=test_set,
    epochs=10,
    steps_per_epoch=steps_per_epoch,
    validation_steps=validation_steps,
    callbacks=[early_stopping]
)
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

ECG-Model

Deep Learning (DL) is a **subset of machine learning** that involves neural networks with multiple layers (deep neural networks). These deep neural networks are capable of learning and representing complex patterns in data.

Types of DL:

- Artificial Neural Networks (ANN)
- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory Networks (LSTM)

Artificial Neural Networks (ANN)

- The **human brain** is a complex system made of **billions of neurons.**
- The **attempts to mimic** the structure and function of the human brain led to a new field of study called **Deep Learning**.

- Intended to mimic the neural networks of the human brain, the structure of artificial neural networks is similar to that of biological neural networks.

Convolutional Neural Network (CNN)

- CNN stands for Convolutional Neural Network. It is a type of neural network architecture designed for processing structured grid data, such as **images and Videos**.
- The major key components of a CNN:
  - Input Layer,
  - Convolutional Layers,
  - Activation Function,
  - Pooling (Subsampling or Downsampling) Layers,
  - Fully Connected Layers,
  - Flattening,
  - Output Layer,
  - Loss Function, Optimization Algorithm,
  - Backpropagation

Recurrent Neural Networks

- Human brain deals with information streams. Most data is obtained, processed, and generated sequentially.
  - E.g., listening: soundwaves □ vocabularies/sentences
  - E.g., action: brain signals/instructions □ sequential muscle movements
- Human thoughts have persistence; humans don't start their thinking from scratch every second.
  - As you read this sentence, you understand each word based on your prior knowledge.
- The applications of standard Artificial Neural Networks (and also Convolutional Networks) are limited due to:
  - They only accepted a fixed-size vector as input (e.g., an image) and produce a fixed-size vector as output (e.g., probabilities of different classes).
  - These models use a fixed amount of computational steps (e.g. the number of layers in the model).
- Recurrent Neural Networks (RNNs) are a family of neural networks introduced to **learn sequential data**.

```
history = model.fit(
    training_set,
    validation_data=test_set,
    epochs=10,
    steps_per_epoch=steps_per_epoch,
    validation_steps=validation_steps,
    callbacks=[early_stopping]
)
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning:
  self._warn_if_super_not_called()
21/21 ━━━━━━━━━━━━━━━━━━━━ 29s 843ms/step - accuracy: 0.5941 - loss: 2.1416 - val_accuracy: 0.6964 - val_loss: 0.
Epoch 2/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 15s 742ms/step - accuracy: 0.7492 - loss: 0.5920 - val_accuracy: 0.6994 - val_loss: 0.
Epoch 3/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 15s 760ms/step - accuracy: 0.6899 - loss: 0.6346 - val_accuracy: 0.6920 - val_loss: 0.
Epoch 4/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 16s 780ms/step - accuracy: 0.7094 - loss: 0.6326 - val_accuracy: 0.6994 - val_loss: 0.
Epoch 5/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 13s 635ms/step - accuracy: 0.7283 - loss: 0.6188 - val_accuracy: 0.6756 - val_loss: 0.
Epoch 6/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 14s 677ms/step - accuracy: 0.7059 - loss: 0.6317 - val_accuracy: 0.7307 - val_loss: 0.
Epoch 7/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 13s 667ms/step - accuracy: 0.6728 - loss: 0.6335 - val_accuracy: 0.6830 - val_loss: 0.
Epoch 8/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 14s 674ms/step - accuracy: 0.6624 - loss: 0.6433 - val_accuracy: 0.7068 - val_loss: 0.
Epoch 9/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 14s 675ms/step - accuracy: 0.7028 - loss: 0.6309 - val_accuracy: 0.7098 - val_loss: 0.
Epoch 10/10
21/21 ━━━━━━━━━━━━━━━━━━━━ 14s 684ms/step - accuracy: 0.6340 - loss: 0.6540 - val_accuracy: 0.7188 - val_loss: 0.
Restoring model weights from the end of the best epoch: 10.
```