Training (Multi Class) 1D

May 3, 2025

1 Importing Necessary Libraries

[2]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten,␣
 ↪Dropout, BatchNormalization
from sklearn.model_selection import train_test_split
```

2 Training

2.1 Loading and Splitting the Data

[5]:
```
X = np.load("../Features.npy")
y = np.load("../Labels (Mutli Class).npy")
```

[6]:
```
X.shape, y.shape
```

[6]:
```
((109416, 250, 2), (109416,))
```

[7]:
```
X_temp, X_test, y_temp, y_test = train_test_split(X, y, shuffle = True,␣
 ↪random_state = 42, test_size = 0.2)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, shuffle =␣
 ↪True, random_state = 42, test_size = 0.2)
```

[8]:
```
X_train.shape, X_val.shape, X_test.shape
```

[8]:
```
((70025, 250, 2), (17507, 250, 2), (21884, 250, 2))
```

[9]:
```
# Numberof classes
np.unique(y_train).shape
```

[9]:
```
(14,)
```

2.2 Model Definition, Compilation and Declaration

[12]:
```python
def build_ecg_cnn_model(input_shape=(250, 2), num_classes=14):
    model = Sequential([
        Conv1D(filters=32, kernel_size=5, activation='relu',␣
 ↪input_shape=input_shape),
        BatchNormalization(),
        MaxPooling1D(pool_size=2),
        Dropout(0.2),

        Conv1D(filters=64, kernel_size=5, activation='relu'),
        BatchNormalization(),
        MaxPooling1D(pool_size=2),
        Dropout(0.2),

        Conv1D(filters=128, kernel_size=3, activation='relu'),
        BatchNormalization(),
        MaxPooling1D(pool_size=2),
        Dropout(0.2),

        Conv1D(filters=256, kernel_size=3, activation='relu'),
        BatchNormalization(),
        MaxPooling1D(pool_size=2),
        Dropout(0.2),

        Flatten(),

        Dense(256, activation='relu', kernel_regularizer=l2(0.001)),
        Dropout(0.5),

        Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
        Dropout(0.3),

        Dense(num_classes, activation='softmax')
    ])

    model.compile(
        optimizer = tf.keras.optimizers.Adam(learning_rate = 0.0001),
        loss = 'sparse_categorical_crossentropy',
        metrics = ['accuracy']
    )

    return model
```

[13]:
```python
model = build_ecg_cnn_model()
```

[14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 246, 32)           352

 batch_normalization (BatchN  (None, 246, 32)          128
 ormalization)

 max_pooling1d (MaxPooling1D  (None, 123, 32)          0
 )

 dropout (Dropout)           (None, 123, 32)           0

 conv1d_1 (Conv1D)           (None, 119, 64)           10304

 batch_normalization_1 (Batc  (None, 119, 64)          256
 hNormalization)

 max_pooling1d_1 (MaxPooling  (None, 59, 64)           0
 1D)

 dropout_1 (Dropout)         (None, 59, 64)            0

 conv1d_2 (Conv1D)           (None, 57, 128)           24704

 batch_normalization_2 (Batc  (None, 57, 128)          512
 hNormalization)

 max_pooling1d_2 (MaxPooling  (None, 28, 128)          0
 1D)

 dropout_2 (Dropout)         (None, 28, 128)           0

 conv1d_3 (Conv1D)           (None, 26, 256)           98560

 batch_normalization_3 (Batc  (None, 26, 256)          1024
 hNormalization)

 max_pooling1d_3 (MaxPooling  (None, 13, 256)          0
 1D)

 dropout_3 (Dropout)         (None, 13, 256)           0

 flatten (Flatten)           (None, 3328)              0
 dense (Dense)               (None, 256)               852224

 dropout_4 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_5 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 14)                1806

=================================================================
Total params: 1,022,766
Trainable params: 1,021,806
Non-trainable params: 960
_________________________________________________________________
```

2.3 Fitting The Model

[24]:
```
early_stop = EarlyStopping(monitor = "val_loss", patience = 3,␣
 ↪restore_best_weights = True)
```

[26]:
```
history = model.fit(X_train, y_train, validation_data = (X_val, y_val), epochs␣
 ↪= 50, batch_size = 64)
```

```
Epoch 1/50
1095/1095 [==============================] - 12s 9ms/step - loss: 1.2469 -
accuracy: 0.8396 - val_loss: 0.8744 - val_accuracy: 0.9269
Epoch 2/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.8445 -
accuracy: 0.9315 - val_loss: 0.7022 - val_accuracy: 0.9563
Epoch 3/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.7022 -
accuracy: 0.9505 - val_loss: 0.5937 - val_accuracy: 0.9670
Epoch 4/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.5946 -
accuracy: 0.9597 - val_loss: 0.5135 - val_accuracy: 0.9737
Epoch 5/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.5041 -
accuracy: 0.9657 - val_loss: 0.4248 - val_accuracy: 0.9769
Epoch 6/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.4241 -
accuracy: 0.9709 - val_loss: 0.3600 - val_accuracy: 0.9790
Epoch 7/50
1095/1095 [==============================] - 10s 10ms/step - loss: 0.3611 -
accuracy: 0.9745 - val_loss: 0.3065 - val_accuracy: 0.9817
Epoch 8/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.3075 -
accuracy: 0.9759 - val_loss: 0.2620 - val_accuracy: 0.9839
```

```
Epoch 9/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.2657 -
accuracy: 0.9790 - val_loss: 0.2257 - val_accuracy: 0.9853
Epoch 10/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.2322 -
accuracy: 0.9798 - val_loss: 0.1964 - val_accuracy: 0.9859
Epoch 11/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.2032 -
accuracy: 0.9820 - val_loss: 0.1710 - val_accuracy: 0.9874
Epoch 12/50
1095/1095 [==============================] - 11s 10ms/step - loss: 0.1800 -
accuracy: 0.9830 - val_loss: 0.1590 - val_accuracy: 0.9871
Epoch 13/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1632 -
accuracy: 0.9837 - val_loss: 0.1380 - val_accuracy: 0.9885
Epoch 14/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1471 -
accuracy: 0.9849 - val_loss: 0.1259 - val_accuracy: 0.9894
Epoch 15/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1347 -
accuracy: 0.9858 - val_loss: 0.1177 - val_accuracy: 0.9888
Epoch 16/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1247 -
accuracy: 0.9862 - val_loss: 0.1091 - val_accuracy: 0.9896
Epoch 17/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1164 -
accuracy: 0.9867 - val_loss: 0.1005 - val_accuracy: 0.9901
Epoch 18/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1086 -
accuracy: 0.9877 - val_loss: 0.0952 - val_accuracy: 0.9898
Epoch 19/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.1021 -
accuracy: 0.9879 - val_loss: 0.0910 - val_accuracy: 0.9899
Epoch 20/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0972 -
accuracy: 0.9880 - val_loss: 0.0899 - val_accuracy: 0.9897
Epoch 21/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0934 -
accuracy: 0.9885 - val_loss: 0.0837 - val_accuracy: 0.9911
Epoch 22/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0881 -
accuracy: 0.9893 - val_loss: 0.0789 - val_accuracy: 0.9919
Epoch 23/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0863 -
accuracy: 0.9889 - val_loss: 0.0840 - val_accuracy: 0.9908
Epoch 24/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0823 -
accuracy: 0.9895 - val_loss: 0.0767 - val_accuracy: 0.9910
```

```
Epoch 25/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0816 -
accuracy: 0.9894 - val_loss: 0.0763 - val_accuracy: 0.9914
Epoch 26/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0784 -
accuracy: 0.9899 - val_loss: 0.0735 - val_accuracy: 0.9916
Epoch 27/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0764 -
accuracy: 0.9899 - val_loss: 0.0695 - val_accuracy: 0.9921
Epoch 28/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0747 -
accuracy: 0.9901 - val_loss: 0.0693 - val_accuracy: 0.9918
Epoch 29/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0726 -
accuracy: 0.9911 - val_loss: 0.0707 - val_accuracy: 0.9919
Epoch 30/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0720 -
accuracy: 0.9909 - val_loss: 0.0679 - val_accuracy: 0.9919
Epoch 31/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0698 -
accuracy: 0.9905 - val_loss: 0.0654 - val_accuracy: 0.9924
Epoch 32/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0709 -
accuracy: 0.9908 - val_loss: 0.0677 - val_accuracy: 0.9921
Epoch 33/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0679 -
accuracy: 0.9911 - val_loss: 0.0652 - val_accuracy: 0.9921
Epoch 34/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0674 -
accuracy: 0.9913 - val_loss: 0.0646 - val_accuracy: 0.9925
Epoch 35/50
1095/1095 [==============================] - 10s 10ms/step - loss: 0.0664 -
accuracy: 0.9916 - val_loss: 0.0664 - val_accuracy: 0.9921
Epoch 36/50
1095/1095 [==============================] - 10s 10ms/step - loss: 0.0667 -
accuracy: 0.9913 - val_loss: 0.0646 - val_accuracy: 0.9927
Epoch 37/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0636 -
accuracy: 0.9922 - val_loss: 0.0684 - val_accuracy: 0.9915
Epoch 38/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0647 -
accuracy: 0.9915 - val_loss: 0.0643 - val_accuracy: 0.9922
Epoch 39/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0628 -
accuracy: 0.9918 - val_loss: 0.0646 - val_accuracy: 0.9921
Epoch 40/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0612 -
accuracy: 0.9921 - val_loss: 0.0621 - val_accuracy: 0.9924
```

```
Epoch 41/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0619 -
accuracy: 0.9920 - val_loss: 0.0635 - val_accuracy: 0.9923
Epoch 42/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0598 -
accuracy: 0.9924 - val_loss: 0.0621 - val_accuracy: 0.9923
Epoch 43/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0611 -
accuracy: 0.9922 - val_loss: 0.0615 - val_accuracy: 0.9922
Epoch 44/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0604 -
accuracy: 0.9922 - val_loss: 0.0603 - val_accuracy: 0.9925
Epoch 45/50
1095/1095 [==============================] - 10s 10ms/step - loss: 0.0591 -
accuracy: 0.9927 - val_loss: 0.0627 - val_accuracy: 0.9929
Epoch 46/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0577 -
accuracy: 0.9930 - val_loss: 0.0604 - val_accuracy: 0.9925
Epoch 47/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0584 -
accuracy: 0.9927 - val_loss: 0.0652 - val_accuracy: 0.9918
Epoch 48/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0562 -
accuracy: 0.9930 - val_loss: 0.0619 - val_accuracy: 0.9925
Epoch 49/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0573 -
accuracy: 0.9926 - val_loss: 0.0610 - val_accuracy: 0.9923
Epoch 50/50
1095/1095 [==============================] - 10s 9ms/step - loss: 0.0574 -
accuracy: 0.9926 - val_loss: 0.0613 - val_accuracy: 0.9927
```

2.4 Saving The Model

[28]:
```
# Saving the model in .h5 format
model.save("../Models/Model 1D.h5")
```

[29]:
```
# Saving the model in .keras format
model.save("../Models/Model 1D.keras")
```

[30]:
```
# Saving the model in tf format
model.save("../Model 1D", save_format = "tf")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op while saving (showing 4 of 4). These functions will
not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: ../Model 1D\assets
```

```
INFO:tensorflow:Assets written to: ../Model 1D\assets
```

```
[31]: hist_df = pd.DataFrame(history.history)
```

```
[32]: hist_df.to_csv("../History 1D.csv")
```