Training (Multi Class) 2D

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 import layers
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, y = X.astype(np.float32), y.astype(np.float32)
```

[8]:
```
X = np.expand_dims(X, axis = -1)
```

[9]:
```
X.shape, y.shape
```

[9]:
```
((109416, 250, 2, 1), (109416,))
```

[10]:
```
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)
```

[11]:
```
X_train.shape, X_val.shape, X_test.shape
```

[11]:
```
((70025, 250, 2, 1), (17507, 250, 2, 1), (21884, 250, 2, 1))
```

[12]:
```
# Numberof classes
np.unique(y_train).shape
```

[12]:
```
(14,)
```

[13]:
```
X_train.dtype
```

[13]:
```
dtype('float32')
```

2.2 Model Definition, Compilation and Declaration

[25]:
```
def build_ecg_cnn_model(input_shape = (250, 2, 1), num_classes=14):
    model = Sequential([
        # First convolution layer
        layers.Conv2D(32, (3, 3), activation='relu', padding = "same",␣
 ↪input_shape=(250, 2, 1)),
        layers.MaxPooling2D((2, 1)),

        # Second convolution layer
        layers.Conv2D(64, (3, 3), activation='relu', padding = "same"),
        layers.MaxPooling2D((2, 1)),

        # Third convolution layer
        layers.Conv2D(128, (3, 3), activation='relu', padding = "same"),
        layers.MaxPooling2D((2, 1)),

        # Fourth convolution layer
        layers.Conv2D(256, (3, 3), activation='relu', padding = "same"),
        layers.MaxPooling2D((2, 1)),

        # Flattening and fully connected layers
        layers.Flatten(),

        layers.Dense(128, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(32, activation='relu'),
        layers.Dense(num_classes, activation='softmax')  # For binary␣
 ↪classification (normal/abnormal)
    ])

    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

```
    return model
```

[27]:
```
model = build_ecg_cnn_model()
```

[28]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 250, 2, 32)        320

 max_pooling2d (MaxPooling2D  (None, 125, 2, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 125, 2, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 62, 2, 64)        0
 2D)

 conv2d_2 (Conv2D)           (None, 62, 2, 128)        73856

 max_pooling2d_2 (MaxPooling  (None, 31, 2, 128)       0
 2D)

 conv2d_3 (Conv2D)           (None, 31, 2, 256)        295168

 max_pooling2d_3 (MaxPooling  (None, 15, 2, 256)       0
 2D)

 flatten (Flatten)           (None, 7680)              0

 dense (Dense)               (None, 128)               983168

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 14)                462

=================================================================
Total params: 1,381,806
Trainable params: 1,381,806
Non-trainable params: 0
_________________________________________________________________
```

2.3 Fitting The Model

[29]:
```
early_stop = EarlyStopping(monitor = "val_loss", patience = 3,␣
↪restore_best_weights = True)
```

[31]:
```
history = model.fit(X_train, y_train, validation_data = (X_val, y_val), epochs␣
↪= 50, batch_size = 64)
```

```
Epoch 1/50
1095/1095 [==============================] - 17s 11ms/step - loss: 0.2430 -
accuracy: 0.9413 - val_loss: 0.1123 - val_accuracy: 0.9718
Epoch 2/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0855 -
accuracy: 0.9791 - val_loss: 0.0703 - val_accuracy: 0.9818
Epoch 3/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0628 -
accuracy: 0.9838 - val_loss: 0.0622 - val_accuracy: 0.9853
Epoch 4/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0511 -
accuracy: 0.9861 - val_loss: 0.0539 - val_accuracy: 0.9866
Epoch 5/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0445 -
accuracy: 0.9880 - val_loss: 0.0487 - val_accuracy: 0.9874
Epoch 6/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0370 -
accuracy: 0.9896 - val_loss: 0.0483 - val_accuracy: 0.9870
Epoch 7/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0353 -
accuracy: 0.9900 - val_loss: 0.0442 - val_accuracy: 0.9883
Epoch 8/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0291 -
accuracy: 0.9918 - val_loss: 0.0425 - val_accuracy: 0.9893
Epoch 9/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0279 -
accuracy: 0.9917 - val_loss: 0.0450 - val_accuracy: 0.9887
Epoch 10/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0257 -
accuracy: 0.9922 - val_loss: 0.0528 - val_accuracy: 0.9864
Epoch 11/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0238 -
accuracy: 0.9929 - val_loss: 0.0420 - val_accuracy: 0.9894
Epoch 12/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0204 -
accuracy: 0.9937 - val_loss: 0.0552 - val_accuracy: 0.9876
Epoch 13/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0211 -
accuracy: 0.9931 - val_loss: 0.0519 - val_accuracy: 0.9879
Epoch 14/50
```

```
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0182 -
accuracy: 0.9944 - val_loss: 0.0502 - val_accuracy: 0.9889
Epoch 15/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0164 -
accuracy: 0.9948 - val_loss: 0.0608 - val_accuracy: 0.9876
Epoch 16/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0166 -
accuracy: 0.9948 - val_loss: 0.0452 - val_accuracy: 0.9889
Epoch 17/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0157 -
accuracy: 0.9950 - val_loss: 0.0486 - val_accuracy: 0.9890
Epoch 18/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0136 -
accuracy: 0.9958 - val_loss: 0.0556 - val_accuracy: 0.9898
Epoch 19/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0164 -
accuracy: 0.9950 - val_loss: 0.0458 - val_accuracy: 0.9898
Epoch 20/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0117 -
accuracy: 0.9961 - val_loss: 0.0529 - val_accuracy: 0.9898
Epoch 21/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0131 -
accuracy: 0.9960 - val_loss: 0.0479 - val_accuracy: 0.9902
Epoch 22/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0109 -
accuracy: 0.9967 - val_loss: 0.0525 - val_accuracy: 0.9891
Epoch 23/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0122 -
accuracy: 0.9962 - val_loss: 0.0525 - val_accuracy: 0.9899
Epoch 24/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0107 -
accuracy: 0.9967 - val_loss: 0.0445 - val_accuracy: 0.9900
Epoch 25/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0119 -
accuracy: 0.9966 - val_loss: 0.0522 - val_accuracy: 0.9888
Epoch 26/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0106 -
accuracy: 0.9968 - val_loss: 0.0476 - val_accuracy: 0.9892
Epoch 27/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0088 -
accuracy: 0.9972 - val_loss: 0.0551 - val_accuracy: 0.9899
Epoch 28/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0106 -
accuracy: 0.9968 - val_loss: 0.0534 - val_accuracy: 0.9906
Epoch 29/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0095 -
accuracy: 0.9971 - val_loss: 0.0534 - val_accuracy: 0.9892
Epoch 30/50
```

```
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0105 -
accuracy: 0.9967 - val_loss: 0.0621 - val_accuracy: 0.9896
Epoch 31/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0081 -
accuracy: 0.9976 - val_loss: 0.0595 - val_accuracy: 0.9898
Epoch 32/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0093 -
accuracy: 0.9973 - val_loss: 0.0486 - val_accuracy: 0.9903
Epoch 33/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0076 -
accuracy: 0.9976 - val_loss: 0.0534 - val_accuracy: 0.9897
Epoch 34/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0099 -
accuracy: 0.9973 - val_loss: 0.0530 - val_accuracy: 0.9911
Epoch 35/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0085 -
accuracy: 0.9976 - val_loss: 0.0671 - val_accuracy: 0.9868
Epoch 36/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0080 -
accuracy: 0.9977 - val_loss: 0.0621 - val_accuracy: 0.9898
Epoch 37/50
1095/1095 [==============================] - 12s 11ms/step - loss: 0.0068 -
accuracy: 0.9981 - val_loss: 0.0596 - val_accuracy: 0.9885
Epoch 38/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0094 -
accuracy: 0.9973 - val_loss: 0.0666 - val_accuracy: 0.9872
Epoch 39/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0104 -
accuracy: 0.9969 - val_loss: 0.0578 - val_accuracy: 0.9909
Epoch 40/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0043 -
accuracy: 0.9985 - val_loss: 0.0569 - val_accuracy: 0.9909
Epoch 41/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0048 -
accuracy: 0.9986 - val_loss: 0.0699 - val_accuracy: 0.9885
Epoch 42/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0088 -
accuracy: 0.9975 - val_loss: 0.0597 - val_accuracy: 0.9907
Epoch 43/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0079 -
accuracy: 0.9979 - val_loss: 0.0649 - val_accuracy: 0.9891
Epoch 44/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0063 -
accuracy: 0.9981 - val_loss: 0.0713 - val_accuracy: 0.9905
Epoch 45/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0057 -
accuracy: 0.9982 - val_loss: 0.0749 - val_accuracy: 0.9899
Epoch 46/50
```

```
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0092 -
accuracy: 0.9976 - val_loss: 0.0565 - val_accuracy: 0.9908
Epoch 47/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0082 -
accuracy: 0.9975 - val_loss: 0.0784 - val_accuracy: 0.9898
Epoch 48/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0059 -
accuracy: 0.9983 - val_loss: 0.0822 - val_accuracy: 0.9901
Epoch 49/50
1095/1095 [==============================] - 13s 12ms/step - loss: 0.0058 -
accuracy: 0.9983 - val_loss: 0.0698 - val_accuracy: 0.9894
Epoch 50/50
1095/1095 [==============================] - 13s 11ms/step - loss: 0.0075 -
accuracy: 0.9978 - val_loss: 0.0673 - val_accuracy: 0.9898
```

2.4 Saving The Model

[46]:
```
# Saving the model in .h5 format
model.save("../Models/Model 2D.h5")
```

[48]:
```
# Saving the model in .keras format
model.save("../Models/Model 2D.keras")
```

[50]:
```
# Saving the model in tf format
model.save("../Model 2D", 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 2D\assets

INFO:tensorflow:Assets written to: ../Model 2D\assets
```

3 Saving the Model Training History

[53]:
```
hist_df = pd.DataFrame(history.history)
```

[55]:
```
hist_df.to_csv("../History 2D.csv")
```