

# Sprint-2

## Model Building

### Task:

#### 1. Model Building:

We are ready with the augmented and pre-processed imagedata, we will begin our build our model by following the below steps.

### Import The Libraries:

#### ▾ Import the Libraries:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
```

### Initializing The Model:

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model.

In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method. Now, will initialize our model.

### Adding CNN Layer:

We are adding a convolution layer with an activation function as “relu” and with a small filter size (3,3) and a number of filters as (32) followed by a max-pooling layer.

The Max pool layer is used to down sample the input. The flatten layer flattens the input.

#### ▾ Adding CNN Layers:

```
model = Sequential()

[ ] model.add(Convolution2D(32,(3,3),input_shape = (64,64,3),activation = "relu"))

[ ] model.add(MaxPooling2D(pool_size = (2,2)))

[ ] model.add(Convolution2D(32,(3,3),activation='relu'))

[ ] model.add(MaxPooling2D(pool_size=(2,2)))

[ ] model.add(Flatten()) # ANN Input...
```

## Adding Dense Layer:

Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

### Adding Dense Layers:

```
[ ] model.add(Dense(units = 128, kernel_initializer = "random_uniform", activation = "relu"))  
[ ] model.add(Dense(units = 128, kernel_initializer = "random_uniform", activation = "relu"))  
[ ] model.add(Dense(units = 128, kernel_initializer = "random_uniform", activation = "relu"))  
[ ] model.add(Dense(units = 128, kernel_initializer = "random_uniform", activation = "relu"))  
[ ] model.add(Dense(units = 128, kernel_initializer = "random_uniform", activation = "relu"))
```

## Adding Output Layer:

### Adding Output Layer:

```
[ ] model.add(Dense(units = 6, kernel_initializer = "random_uniform", activation = "softmax"))
```

Understanding the model is very important phase to properly use it for training and prediction purposes. keras provides a simple method, summary to get the full information about the model and its layers

```
model.summary()
```

Model: "sequential"

| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d (Conv2D)                | (None, 62, 62, 32) | 896     |
| max_pooling2d (MaxPooling2D)   | (None, 31, 31, 32) | 0       |
| conv2d_1 (Conv2D)              | (None, 29, 29, 32) | 9248    |
| max_pooling2d_1 (MaxPooling2D) | (None, 14, 14, 32) | 0       |
| flatten (Flatten)              | (None, 6272)       | 0       |
| dense (Dense)                  | (None, 128)        | 802944  |
| dense_1 (Dense)                | (None, 128)        | 16512   |
| dense_2 (Dense)                | (None, 128)        | 16512   |
| dense_3 (Dense)                | (None, 128)        | 16512   |
| dense_4 (Dense)                | (None, 128)        | 16512   |

## Configure The Learning Process:

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation processes.

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer.

Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in the training process.

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

## Train The Model:

We will train our model with our image dataset. fit generator functions used to train a deep learning neural network.

### ▼ Train the model:

```
model.fit_generator(generator=x_train,steps_per_epoch = len(x_train), epochs=9, validation_data=x_test,validation_steps = len(x_test))
```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: UserWarning: 'Model.fit\_generator' is deprecated and will be removed in a future version. Please use 'Model.fit',  
"""Entry point for launching an IPython kernel.

| Epoch     | 480/480 | Time          | loss   | accuracy | val_loss | val_accuracy |
|-----------|---------|---------------|--------|----------|----------|--------------|
| Epoch 1/9 | [=====] | 41s 66ms/step | 1.3631 | 0.5007   | 1.6149   | 0.4544       |
| Epoch 2/9 | [=====] | 31s 65ms/step | 0.7976 | 0.6908   | 0.9267   | 0.6988       |
| Epoch 3/9 | [=====] | 34s 71ms/step | 0.3399 | 0.8819   | 0.6958   | 0.7965       |
| Epoch 4/9 | [=====] | 30s 63ms/step | 0.2286 | 0.9223   | 0.5724   | 0.8095       |
| Epoch 5/9 | [=====] | 30s 63ms/step | 0.1798 | 0.9439   | 0.4829   | 0.8488       |
| Epoch 6/9 | [=====] | 30s 63ms/step | 0.1416 | 0.9555   | 0.5124   | 0.8549       |
| Epoch 7/9 | [=====] | 30s 62ms/step | 0.1068 | 0.9662   | 0.5708   | 0.8585       |
| Epoch 8/9 | [=====] | 30s 63ms/step | 0.0917 | 0.9710   | 0.4615   | 0.8714       |
| Epoch 9/9 | [=====] | 30s 62ms/step | 0.0796 | 0.9750   | 0.7387   | 0.8535       |

<keras.callbacks.History at 0x7f85e0f6410>

## Save The Model:

The model is saved with .h5 extension as follows.

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

### ▼ Save the model:

```
[ ] #Saving Model.  
    model.save('ECG.h5')
```

## Test The Model:

Load necessary libraries and load the saved model using load\_model Taking an image as input and checking the results.

The target size should for the image that is should be the same as the target size that you have used for training.

### Testing the model:

```
▶ model=load_model('ECG.h5')  
[32] img
```