Sprint-2

Model Building

Date	01 Nov 2022
Team ID	PNT2022TMID09656
Project Name	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation

Task

l. Model Building

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

Import The Libraries:

Import the Libraries:

O from tensorflow. keras .models import sequential from tensorflow. layers import Dense from tensorflow.keras.layers import Convolution2D from tensorflow.keras. layers import Maxp001ing2D 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 downsample the input. The flatten layer flattens the input.

Adding CNN Layers:

```
model = Sequential()

[ ]model.add(Conv01ution2D(32, (3, 3) , input shape   (64, 64, 3) , activation    " relu"))

model. id(MaxPooling2D(pool_size - (2,2)))

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

id(MaxPooling2D(pool_size=(2,2)))

[ ]model.add(F1atten()) # 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          "randomuniform",activation ="relu"))

model. add(Dense(units = 128, kernel initializer - "random niform",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:

I J model. dd(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

O model. summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896

max_p001ing2d (maxp001ing2D (None, 31, 31, 32) 0conv2d 1

(conv2D) (None, 29, 29, 32) 9248

max_p001ing2d 1 (Maxpooling (None, 14, 14, 32)0

20)

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 a

(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 proceses

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.

-r vuuc

' accuracy']

)

model. compile(optimizer- 'adam' floss= i categorical crossentropy' j

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:

```
O model. steps_per_epoch                    len(x train),    Oalidation data=x test, validation steps          len(x test))
/usr/10ca1/1ib/python3.7/dist-packages/ipykerne1 launcher .py:l: UserWarning: - Model . fit_generator- is deprecated and will be removed in a future version. Please use -tᵛlodel. fit"Entry point for launching an IPython kernel. Epoch 1/9 480/480 [ - 41s 66ms/step loss: 1. 3631 accuracy: 0.5007 val loss: 1.6149 val accuracy: 0.4544
Epoch  2/9   ==============================] -   480/480 [ - 31s 65ms/step loss: 0.7976 accuracy: 0.6908 val loss: e. 9267 val accuracy:
0.6988
Epoch  3/9   ==============================] -   480/480 [us 71ms/step loss: e. 3399 accuracy : e. 8819 val loss: 0.6958 val accuracy:
0.7965
Epoch  a/9   ==============================] -   480/480 [ - 30s 63ms/step loss: e. 2286 accuracy: 0.9223 val loss : 0.5724 val accuracy
: .8095
Epoch  5/9   ==============================] -   480/480 [ - 30s 63ms/step loss e.1798 accuracy: 0.9439 val loss : 0.4829 val accuracy:
0.8488
Epoch  6/9   ==============================] -   480/480 [30s 63ms/step loss •. 0.1416 accuracy: 0.9555 val loss: 0.5124 val accuracy:
0.8549       ==============================] -
Epoch 7/9
             ==============================] -
480/480 [ -                      2ms/step   loss: 0.1068   accuracy: 0.9662   val loss:          val accuracy: 0.8585
Epoch  8/9   ==============================] -   480/480 [ - 30s 63ms/step loss: 0.0917 accuracy: 0.9710 val loss: 0.4615 val accuracy:
0.8714       ==============================] -
Epoch 9/9

480/480 [ -                   30s 62ms/step   loss: €.0796   accuracy: 0.9750   val loss: o. 7387   val accuracy: 0.8535
<keras .callbacks .History at øx7f85e0Øf641Ø>
```

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 i)
```

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:

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

```
O model—load model( ' ECG.h5' )
```

```
v/ [30] img=image. load img("/content/fig_44.png",target_size=(
```

v [31] x=image. img_to array(img)

✓ [32] img

v [33] import numpy as np

v' [34] x=np. expand_`dims(x,`

✓ [35] pred = model
.predict(x)