DATE : ___/___/______

201900 Muhammad Hassan Mukhtar

Artifical Neural Network

Assignment 01

formulas :

1. Convolution layer

No of param = [(fw x fh x depth) + 1] x No of filter

2. Fully Connected (Dense)

No of param = (Input + 1) x output size

Note :: for Input Polling layer their is no Learning parameters
but the activation is Length x width x height.

3. Activation Size

AS = height ~~x~~ width x channels

Solution :

for conv 1. filter size (5,5) stride (1)

so, No of param = [(5 x 5 x 3) + 1] x 8 = 608

3 for No of channels of Input Image

8 for No of filter to be applied to Image.

DATE : __/__/____

For Conv 2 filter (5,5) stride (1)

So, No of param = $[(5\times5\times8)+1]\times16$ = 3216

8 for previous layer depth

16 for filter applied onto them.

for Conv 2

Activation size = $10\times10\times16$ = 1600

for Conv 1

Activation size = $28\times28\times8$ = 6272

for Pooling 1

Activation size = $14\times14\times8$ = 1568

for FC

Activation size = 120×1 = 120

for Fc

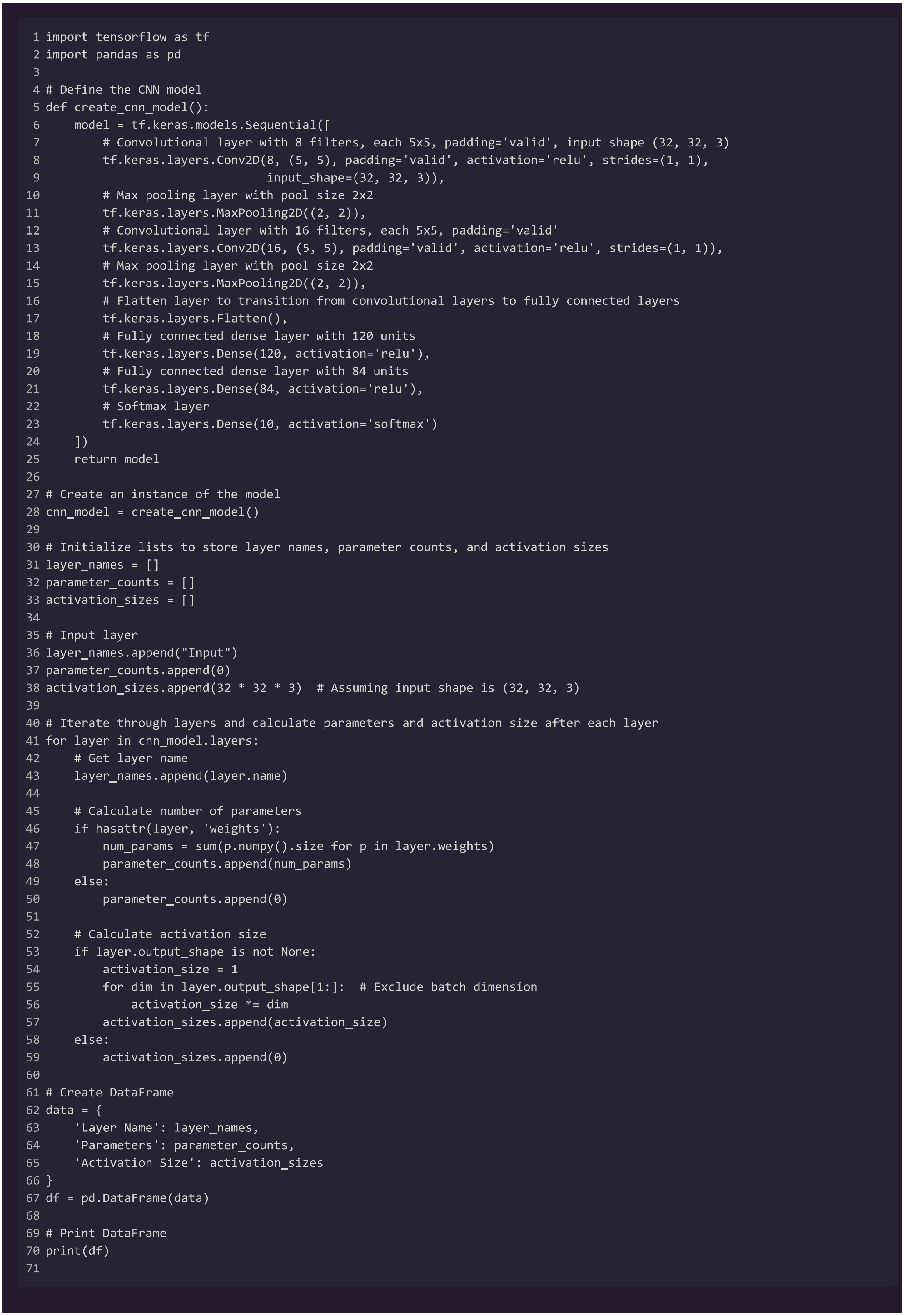
No of param = (Input + 1) x output

= $(400+1)\times120$ = 48120

For output (softmax)

No of param = $(84+1)\times10$

= 850.



```python
import tensorflow as tf
import pandas as pd

# Define the CNN model
def create_cnn_model():
    model = tf.keras.models.Sequential([
        # Convolutional layer with 8 filters, each 5x5, padding='valid', input shape (32, 32, 3)
        tf.keras.layers.Conv2D(8, (5, 5), padding='valid', activation='relu', strides=(1, 1),
                               input_shape=(32, 32, 3)),
        # Max pooling layer with pool size 2x2
        tf.keras.layers.MaxPooling2D((2, 2)),
        # Convolutional layer with 16 filters, each 5x5, padding='valid'
        tf.keras.layers.Conv2D(16, (5, 5), padding='valid', activation='relu', strides=(1, 1)),
        # Max pooling layer with pool size 2x2
        tf.keras.layers.MaxPooling2D((2, 2)),
        # Flatten layer to transition from convolutional layers to fully connected layers
        tf.keras.layers.Flatten(),
        # Fully connected dense layer with 120 units
        tf.keras.layers.Dense(120, activation='relu'),
        # Fully connected dense layer with 84 units
        tf.keras.layers.Dense(84, activation='relu'),
        # Softmax layer
        tf.keras.layers.Dense(10, activation='softmax')
    ])
    return model

# Create an instance of the model
cnn_model = create_cnn_model()

# Initialize lists to store layer names, parameter counts, and activation sizes
layer_names = []
parameter_counts = []
activation_sizes = []

# Input layer
layer_names.append("Input")
parameter_counts.append(0)
activation_sizes.append(32 * 32 * 3)  # Assuming input shape is (32, 32, 3)

# Iterate through layers and calculate parameters and activation size after each layer
for layer in cnn_model.layers:
    # Get layer name
    layer_names.append(layer.name)

    # Calculate number of parameters
    if hasattr(layer, 'weights'):
        num_params = sum(p.numpy().size for p in layer.weights)
        parameter_counts.append(num_params)
    else:
        parameter_counts.append(0)

    # Calculate activation size
    if layer.output_shape is not None:
        activation_size = 1
        for dim in layer.output_shape[1:]:  # Exclude batch dimension
            activation_size *= dim
        activation_sizes.append(activation_size)
    else:
        activation_sizes.append(0)

# Create DataFrame
data = {
    'Layer Name': layer_names,
    'Parameters': parameter_counts,
    'Activation Size': activation_sizes
}
df = pd.DataFrame(data)

# Print DataFrame
print(df)

```

```
       Layer Name  Parameters  Activation Size
0           Input           0             3072
1         conv2d_2         608             6272
2  max_pooling2d_2           0             1568
3         conv2d_3        3216             1600
4  max_pooling2d_3           0              400
5        flatten_1           0              400
6          dense_3       48120              120
7          dense_4       10164               84
8          dense_5         850               10
```

1. Number of Parameters:

   The number of parameters in a layer depends on the type of layer. For convolutional layers and fully connected (dense) layers, the number of parameters can be calculated using the following formulas:

   - For Convolutional layers: Number of Parameters = [(filter_width * filter_height * input_depth) + 1] * num_filters Where:
     - filter_width and filter_height are the width and height of the filter/kernel respectively.
     - input_depth is the number of channels in the input.
     - num_filters is the number of filters in the layer.
     - +1 is added for the bias term associated with each filter.
   - For Fully Connected (Dense) layers: Number of Parameters = (input_size + 1) * output_size Where:
     - input_size is the number of neurons in the previous layer.
     - output_size is the number of neurons in the current layer.
     - +1 is added for the bias term associated with each neuron.

     Note:: No parameters to learn in Input / Pooling Layer.

2. Activation Size:

   The activation size after each layer can be computed by multiplying the dimensions of the output shape of the layer (excluding the batch dimension). If the output shape is (batch_size, height, width, channels), then the activation size is given by: Activation Size = height * width * channels