Alexnet

March 6, 2023

1 *AlexNet (2012)*

AlexNet is a convolutional neural network architecture that was developed by **Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012**. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in the same year and marked a breakthrough in the field of computer vision.

The architecture consists of eight layers, including five convolutional layers and three fully connected layers. It was the first architecture to use rectified linear units (ReLU) as the activation function, which **helped overcome the vanishing gradient problem** commonly experienced in deep neural networks.

2 *Architectural Flow*

2.1 *Import Necessary Libraries*

[1]:
```
import tensorflow as tf
from tensorflow import keras
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D,
 MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
```

[2]:
```
!pip install tflearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: tflearn in /usr/local/lib/python3.8/dist-packages
(0.5.0)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages
(from tflearn) (1.15.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages
(from tflearn) (1.22.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.8/dist-packages
(from tflearn) (8.4.0)
```

2.2 *Alexnet using oxflower17 dataset*

[3]:
```
# Get Data
import tflearn.datasets.oxflower17 as oxflower17
x, y = oxflower17.load_data(one_hot=True)
```

WARNING:tensorflow:From /usr/local/lib/python3.8/dist-packages/tensorflow/python/compat/v2_compat.py:107: disable_resource_variables (from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term

[4]:
```
# Split data into training and test sets

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=42)

# Print the shapes of the resulting arrays
print("x_train shape:", x_train.shape)
print("y_train shape:", y_train.shape)
print("x_test shape:", x_test.shape)
print("y_test shape:", y_test.shape)
```

x_train shape: (1088, 224, 224, 3)
y_train shape: (1088, 17)
x_test shape: (272, 224, 224, 3)
y_test shape: (272, 17)

[5]:
```
x
```

[5]:
```
array([[[[0.3647059 , 0.6901961 , 0.84705883],
         [0.36078432, 0.6862745 , 0.84313726],
         [0.36078432, 0.6862745 , 0.84313726],
         ...,
         [0.32941177, 0.65882355, 0.83137256],
         [0.3254902 , 0.654902  , 0.827451  ],
         [0.3254902 , 0.654902  , 0.827451  ]],

        [[0.36862746, 0.69411767, 0.8509804 ],
         [0.3647059 , 0.6901961 , 0.84705883],
         [0.3647059 , 0.6901961 , 0.84705883],
         ...,
         [0.32941177, 0.65882355, 0.83137256],
         [0.3254902 , 0.654902  , 0.827451  ],
         [0.3254902 , 0.654902  , 0.827451  ]],
```

```
 [[0.37254903, 0.69803923, 0.85490197],
  [0.36862746, 0.69411767, 0.8509804 ],
  [0.3647059 , 0.6901961 , 0.84705883],
  ...,
  [0.32941177, 0.65882355, 0.83137256],
  [0.3254902 , 0.654902 , 0.827451 ],
  [0.3254902 , 0.654902 , 0.827451 ]],

 ...,

 [[0.15294118, 0.22745098, 0.01176471],
  [0.24705882, 0.28235295, 0.02352941],
  [0.49019608, 0.4745098 , 0.14117648],
  ...,
  [0.14509805, 0.20784314, 0.1254902 ],
  [0.13333334, 0.1764706 , 0.10980392],
  [0.15294118, 0.19607843, 0.12941177]],

 [[0.1254902 , 0.17254902, 0.01176471],
  [0.21960784, 0.22745098, 0.01176471],
  [0.46666667, 0.43529412, 0.11764706],
  ...,
  [0.12941177, 0.19607843, 0.10980392],
  [0.12941177, 0.16862746, 0.09803922],
  [0.14509805, 0.18039216, 0.11372549]],

 [[0.11372549, 0.13725491, 0.01960784],
  [0.20784314, 0.2       , 0.01568628],
  [0.4509804 , 0.40784314, 0.10196079],
  ...,
  [0.10980392, 0.1764706 , 0.09019608],
  [0.11764706, 0.15294118, 0.08235294],
  [0.13333334, 0.16078432, 0.09803922]]],


[[[0.60784316, 0.78039217, 0.6666667 ],
  [0.5686275 , 0.74509805, 0.5921569 ],
  [0.48235294, 0.6666667 , 0.4509804 ],
  ...,
  [0.69803923, 0.75686276, 0.4862745 ],
  [0.5882353 , 0.6901961 , 0.36862746],
  [0.6       , 0.7137255 , 0.3764706 ]],

 [[0.5803922 , 0.75686276, 0.63529414],
  [0.5921569 , 0.76862746, 0.6117647 ],
  [0.53333336, 0.7137255 , 0.5058824 ],
  ...,
```

```
 [0.72156864, 0.7882353 , 0.5176471 ],
  [0.5647059 , 0.67058825, 0.34901962],
  [0.5803922 , 0.69803923, 0.3647059 ]],

 [[0.5411765 , 0.72156864, 0.5803922 ],
 [0.58431375, 0.7607843 , 0.59607846],
  [0.5803922 , 0.7607843 , 0.5568628 ],
  ...,
  [0.6745098 , 0.75686276, 0.49411765],
 [0.59607846, 0.70980394, 0.39607844],
  [0.6117647 , 0.73333335, 0.40784314]],

 ...,

 [[0.47058824, 0.5058824 , 0.16470589],
  [0.7294118 , 0.76862746, 0.44313726],
  [0.7607843 , 0.8039216 , 0.50980395],
  ...,
  [0.61960787, 0.78039217, 0.5411765 ],
  [0.5529412 , 0.73333335, 0.45882353],
 [0.45490196, 0.6431373 , 0.35686275]],

 [[0.4627451 , 0.50980395, 0.15686275],
  [0.6431373 , 0.6901961 , 0.35686275],
  [0.77254903, 0.8156863 , 0.5254902 ],
  ...,
  [0.654902 , 0.8392157 , 0.5372549 ],
  [0.6313726 , 0.81960785, 0.5137255 ],
 [0.60784316, 0.8       , 0.49019608]],

 [[0.5568628 , 0.6117647 , 0.2509804 ],
 [0.60784316, 0.6627451 , 0.3254902 ],
  [0.7176471 , 0.7647059 , 0.4745098 ],
  ...,
 [0.58431375, 0.78431374, 0.42745098],
  [0.6509804 , 0.84313726, 0.5176471 ],
  [0.6745098 , 0.85882354, 0.5411765 ]]],


[[[0.10980392, 0.16470589, 0.0627451 ],
 [0.10980392, 0.16470589, 0.06666667],
  [0.11372549, 0.16078432, 0.07058824],
  ...,
  [0.14117648, 0.21176471, 0.08235294],
 [0.13725491, 0.19607843, 0.07450981],
 [0.13725491, 0.1882353 , 0.07843138]],
```

```
 [[0.10980392, 0.16470589, 0.0627451 ],
  [0.10980392, 0.16078432, 0.0627451 ],
  [0.10980392, 0.15686275, 0.06666667],
   ...,
   [0.13725491, 0.20784314, 0.08235294],
  [0.13333334, 0.19607843, 0.07450981],
  [0.13333334, 0.18431373, 0.07450981]],

 [[0.10980392, 0.16470589, 0.0627451 ],
  [0.10588235, 0.16078432, 0.0627451 ],
  [0.10196079, 0.14901961, 0.05882353],
   ...,
   [0.13333334, 0.20392157, 0.07843138],
  [0.12941177, 0.19215687, 0.07058824],
  [0.12941177, 0.18039216, 0.07058824]],

 ...,

  [[0.3647059 , 0.5058824 , 0.15686275],
  [0.34509805, 0.4745098 , 0.16470589],
  [0.28235295, 0.39215687, 0.14509805],
   ...,
   [0.11764706, 0.15686275, 0.04705882],
  [0.10588235, 0.14117648, 0.05098039],
  [0.09411765, 0.12941177, 0.07058824]],

 [[0.33333334, 0.47058824, 0.12156863],
  [0.30588236, 0.42745098, 0.12941177],
  [0.22745098, 0.32941177, 0.10588235],
   ...,
   [0.10980392, 0.13725491, 0.05882353],
  [0.09803922, 0.11764706, 0.0627451 ],
  [0.08235294, 0.10588235, 0.06666667]],

  [[0.3019608 , 0.43529412, 0.09019608],
   [0.2784314 , 0.3882353 , 0.09803922],
  [0.19607843, 0.29411766, 0.07450981],
   ...,
   [0.10196079, 0.12156863, 0.06666667],
  [0.09019608, 0.10196079, 0.07058824],
  [0.07450981, 0.09411765, 0.07450981]]],


...,


[[[0.00392157, 0.02745098, 0.01176471],
```

```
 [0.00784314, 0.03137255, 0.01568628],
  [0.        , 0.02352941, 0.00784314],
  ...,
  [0.01960784, 0.10980392, 0.        ],
 [0.03137255, 0.10980392, 0.00392157],
 [0.03529412, 0.10980392, 0.00392157]],

[[0.        , 0.01960784, 0.00392157],
 [0.00392157, 0.02352941, 0.00784314],
  [0.        , 0.01568628, 0.        ],
  ...,
  [0.02745098, 0.10980392, 0.        ],
 [0.05098039, 0.1254902 , 0.01176471],
 [0.05490196, 0.12941177, 0.01568628]],

[[0.01176471, 0.01960784, 0.00784314],
 [0.01176471, 0.02352941, 0.01176471],
  [0.00784314, 0.01568628, 0.00392157],
  ...,
  [0.05490196, 0.12941177, 0.01960784],
 [0.05882353, 0.13333334, 0.01176471],
 [0.05882353, 0.13333334, 0.01176471]],


...,

[[0.20784314, 0.26666668, 0.00784314],
 [0.20784314, 0.26666668, 0.01176471],
  [0.20784314, 0.26666668, 0.01568628],
  ...,
  [0.5411765 , 0.54901963, 0.34901962],
 [0.21960784, 0.22745098, 0.02745098],
 [0.22352941, 0.23529412, 0.03529412]],

[[0.22352941, 0.28627452, 0.00392157],
 [0.22352941, 0.28627452, 0.00392157],
  [0.22352941, 0.28627452, 0.00784314],
  ...,
  [0.29803923, 0.30980393, 0.08627451],
 [0.23529412, 0.24705882, 0.03137255],
 [0.23137255, 0.24313726, 0.03529412]],

[[0.23921569, 0.30588236, 0.        ],
 [0.23921569, 0.30588236, 0.00392157],
  [0.23921569, 0.3019608 , 0.00784314],
  ...,
  [0.22745098, 0.24705882, 0.00392157],
 [0.23529412, 0.24705882, 0.03137255],
```

```
  [0.20392157, 0.21568628,  0.00392157]]],


[[[0.1882353 , 0.16078432,  0.18039216],
  [0.4       , 0.40392157,  0.3254902 ],
  [0.5372549 , 0.54509807,  0.49803922],
  ...,
  [0.54509807, 0.54509807,  0.54509807],
  [0.5764706 , 0.5764706 ,  0.5764706 ],
  [0.39215687, 0.39215687,  0.39215687]],

 [[0.44313726, 0.42745098,  0.43137255],
  [0.61960787, 0.627451  ,  0.5294118 ],
  [0.92156863, 0.92941177,  0.87058824],
  ...,
  [1.        , 1.        ,  1.        ],
  [1.        , 1.        ,  1.        ],
  [0.7529412 , 0.7529412 ,  0.7529412 ]],

 [[0.49019608, 0.49019608,  0.4745098 ],
  [0.7137255 , 0.72156864,  0.64705884],
  [0.5921569 , 0.6039216 ,  0.54509807],
  ...,
  [0.9529412 , 0.9529412 ,  0.9529412 ],
  [0.9843137 , 0.9843137 ,  0.9843137 ],
  [0.69803923, 0.69803923,  0.69803923]],


 ...,

 [[0.31764707, 0.38039216,  0.23137255],
  [0.46666667, 0.52156866,  0.25882354],
  [0.4745098 , 0.5411765 ,  0.18039216],
  ...,
  [0.19215687, 0.2784314 ,  0.08235294],
  [0.19215687, 0.2627451 ,  0.10196079],
  [0.13725491, 0.2       ,  0.10196079]],

 [[0.3372549 , 0.39607844,  0.22352941],
  [0.47843137, 0.5372549 ,  0.2901961 ],
  [0.49019608, 0.5568628 ,  0.27058825],
  ...,
  [0.2627451 , 0.3137255 ,  0.19215687],
  [0.24705882, 0.29411766,  0.19215687],
  [0.2       , 0.23921569,  0.16470589]],

 [[0.19215687, 0.23921569,  0.11372549],
  [0.27450982, 0.32941177,  0.16078432],
```

```
   [0.2627451 , 0.32156864, 0.15686275],
   ...,
   [0.1764706 , 0.21176471, 0.13333334],
   [0.15294118, 0.1882353 , 0.11764706],
   [0.12156863, 0.15686275, 0.09411765]]],


[[[0.3372549 , 0.4627451 , 0.27058825],
  [0.34117648, 0.47843137, 0.28235295],
   [0.32156864, 0.47058824, 0.27058825],
   ...,
   [0.16470589, 0.34117648, 0.08627451],
  [0.16862746, 0.34117648, 0.07843138],
   [0.1764706 , 0.34509805, 0.06666667]],

 [[0.32941177, 0.4627451 , 0.27058825],
  [0.33333334, 0.4745098 , 0.27450982],
  [0.30588236, 0.4627451 , 0.2627451 ],
   ...,
   [0.14509805, 0.3254902 , 0.07450981],
  [0.15294118, 0.3254902 , 0.07058824],
  [0.16078432, 0.32941177, 0.05882353]],

 [[0.32156864, 0.46666667, 0.26666668],
  [0.32156864, 0.47058824, 0.27058825],
   [0.29411766, 0.45490196, 0.2509804 ],
   ...,
   [0.12941177, 0.30588236, 0.05490196],
  [0.13725491, 0.30588236, 0.05882353],
  [0.14509805, 0.30980393, 0.05098039]],

 ...,

 [[0.37254903, 0.53333336, 0.07058824],
  [0.38039216, 0.54509807, 0.08235294],
   [0.39607844, 0.5686275 , 0.09411765],
   ...,
   [0.12156863, 0.14901961, 0.11764706],
  [0.11764706, 0.14509805, 0.11372549],
  [0.12156863, 0.14901961, 0.11764706]],

 [[0.19607843, 0.3254902 , 0.01176471],
  [0.21568628, 0.34509805, 0.02745098],
   [0.23137255, 0.36862746, 0.02745098],
   ...,
   [0.12156863, 0.14901961, 0.11764706],
  [0.11764706, 0.14509805, 0.11372549],
```

```
        [0.1254902 , 0.15294118, 0.12156863]],

       [[0.11372549, 0.20392157, 0.03921569],
        [0.12941177, 0.22745098, 0.05490196],
        [0.14117648, 0.24313726, 0.04313726],
        ...,
        [0.12941177, 0.15686275, 0.1254902 ],
        [0.13333334, 0.16078432, 0.12941177],
        [0.13725491, 0.16470589, 0.13333334]]], dtype=float32)
```

[6]:
```
y
```

[6]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

[7]:
```
x.shape
```

[7]:
```
(1360, 224, 224, 3)
```

[8]:
```
y.shape
```

[8]:
```
(1360, 17)
```

2.3 *Model Architecture*

[9]:
```
# Create a sequential model
model = Sequential()

# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(224,224,3), kernel_size=(11,11),
 strides=(4,4), padding='valid'))
model.add(Activation('relu'))

# Pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))
# Batch Normalisation before passing it to the next layer
model.add(BatchNormalization())

# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), padding='same'))
model.add(Activation('relu'))
```

```
# Pooling
model.add(MaxPooling2D(pool_size=(3,3),     strides=(2,2),     padding='valid'))
# Batch Normalisation
model.add(BatchNormalization())



# 3rd Convolutional Layer
model.add(Conv2D(filters=384,     kernel_size=(3,3),     strides=(1,1),␣
 ↪padding='valid'))
model.add(Activation('relu'))
# Batch Normalisation
model.add(BatchNormalization())

# 4th Convolutional Layer
model.add(Conv2D(filters=384,     kernel_size=(3,3),     strides=(1,1),␣
 ↪padding='valid'))
model.add(Activation('relu'))
# Batch Normalisation
model.add(BatchNormalization())


# 5th Convolutional Layer
model.add(Conv2D(filters=256,     kernel_size=(3,3),     strides=(1,1),␣
 ↪padding='valid'))
model.add(Activation('relu'))


# Pooling
model.add(MaxPooling2D(pool_size=(3,3),     strides=(2,2),     padding='valid'))
# Batch Normalisation
model.add(BatchNormalization())


# Passing it to a dense layer
model.add(Flatten())

# 1st Dense Layer
model.add(Dense(4096, input_shape=(224*224*3,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))
# Batch Normalisation
model.add(BatchNormalization())

# 2nd Dense Layer
model.add(Dense(4096))
```

```
model.add(Activation("relu"))
# Add Dropout
model.add(Dropout(0.4))
# Batch Normalisation
model.add(BatchNormalization())

# Output Layer
model.add(Dense(17))
model.add(Activation("softmax"))

model.summary()
```

WARNING:tensorflow:From /usr/local/lib/python3.8/dist-packages/keras/layers/normalization/batch_normalization.py:561: _colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 54, 96)	34944
activation (Activation)	(None, 54, 54, 96)	0
max_pooling2d (MaxPooling2D)	(None, 26, 26, 96)	0
batch_normalization (BatchNormalization)	(None, 26, 26, 96)	384
conv2d_1 (Conv2D)	(None, 26, 26, 256)	614656
activation_1 (Activation)	(None, 26, 26, 256)	0
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 256)	0
batch_normalization_1 (BatchNormalization)	(None, 12, 12, 256)	1024
conv2d_2 (Conv2D)	(None, 10, 10, 384)	885120
activation_2 (Activation)	(None, 10, 10, 384)	0
batch_normalization_2 (Batc	(None, 10, 10, 384)	1536

hNormalization)		
conv2d_3 (Conv2D)	(None, 8, 8, 384)	1327488
activation_3 (Activation)	(None, 8, 8, 384)	0
batch_normalization_3 (Batc hNormalization)	(None, 8, 8, 384)	1536
conv2d_4 (Conv2D)	(None, 6, 6, 256)	884992
activation_4 (Activation)	(None, 6, 6, 256)	0
max_pooling2d_2 (MaxPooling 2D)	(None, 2, 2, 256)	0
batch_normalization_4 (Batc hNormalization)	(None, 2, 2, 256)	1024
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 4096)	4198400
activation_5 (Activation)	(None, 4096)	0
dropout (Dropout)	(None, 4096)	0
batch_normalization_5 (Batc hNormalization)	(None, 4096)	16384
dense_1 (Dense)	(None, 4096)	16781312
activation_6 (Activation)	(None, 4096)	0
dropout_1 (Dropout)	(None, 4096)	0
batch_normalization_6 (Batc hNormalization)	(None, 4096)	16384
dense_2 (Dense)	(None, 17)	69649
activation_7 (Activation)	(None, 17)	0

==========

Total params: 24,834,833
Trainable params: 24,815,697
Non-trainable params: 19,136

We have used 224 x 224 x 3 as input shap, consider the above image for the same, as the methodology of this CNN Architecture remains the same.

[10]:
```
!pip install tensorflow-addons==0.16.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow-addons==0.16.1 in /usr/local/lib/python3.8/dist-packages (0.16.1)
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.8/dist-packages (from tensorflow-addons==0.16.1) (2.7.1)
```

[11]:
```
# Compile the model
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.
  optimizers.Adadelta(learning_rate=0.01),        metrics=["accuracy"])
```

[12]:
```
# Train
history_alexnet_batchnorm = model.fit(x_train, y_train, batch_size=128,
  epochs=12, verbose=1, validation_data=(x_test, y_test))
```

```
Train on 1088 samples, validate on 272 samples
Epoch 1/12
1088/1088 [==============================] - ETA: 0s - loss: 3.4831 - acc: 0.0754

/usr/local/lib/python3.8/dist-packages/keras/engine/training_v1.py:2333: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `updates` are applied automatically.
  updates = self.state_updates
1088/1088 [==============================] - 7s 7ms/sample - loss: 3.4831 - acc: 0.0754 - val_loss: 2.8329 - val_acc: 0.0478
Epoch 2/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 2.7744 - acc: 0.1673 - val_loss: 2.8327 - val_acc: 0.0625
Epoch 3/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 2.3161 - acc: 0.3079 - val_loss: 2.8350 - val_acc: 0.0588
Epoch 4/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 2.0319 - acc: 0.3695 - val_loss: 2.8389 - val_acc: 0.0588
Epoch 5/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 1.8058 - acc: 0.4283 - val_loss: 2.8449 - val_acc: 0.0588
Epoch 6/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 1.6174 - acc:
```

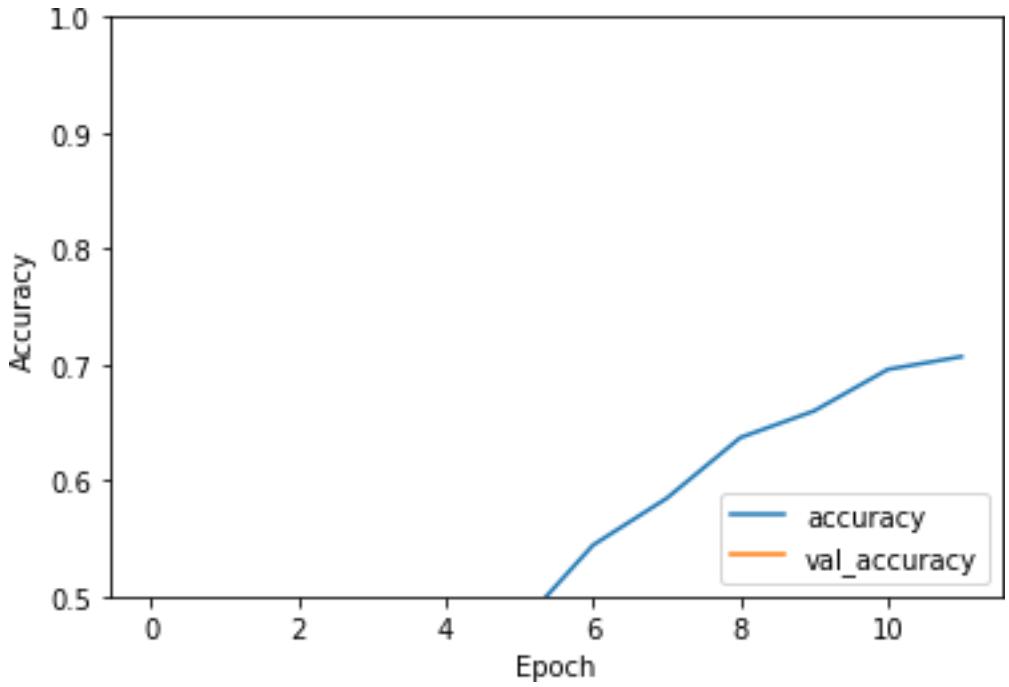
```
0.4743 - val_loss: 2.8529 - val_acc: 0.0588
Epoch 7/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 1.4455 - acc:
0.5441 - val_loss: 2.8680 - val_acc: 0.0588
Epoch 8/12
1088/1088 [==============================] - 3s 3ms/sample - loss: 1.2881 - acc:
0.5846 - val_loss: 2.8859 - val_acc: 0.0588
Epoch 9/12
1088/1088 [==============================] - 3s 2ms/sample - loss: 1.1605 - acc:
0.6369 - val_loss: 2.9080 - val_acc: 0.0588
Epoch 10/12
1088/1088 [==============================] - 2s 2ms/sample - loss: 1.0783 - acc:
0.6599 - val_loss: 2.9318 - val_acc: 0.0588
Epoch 11/12
1088/1088 [==============================] - 3s 2ms/sample - loss: 0.9744 - acc:
0.6958 - val_loss: 2.9608 - val_acc: 0.0735
Epoch 12/12
1088/1088 [==============================] - 3s 2ms/sample - loss: 0.9527 - acc:
0.7068 - val_loss: 2.9972 - val_acc: 0.0993
```

[13]:
```
# Evaluate the model on the test data
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 2.997237570145551
Test accuracy: 0.0992647
```

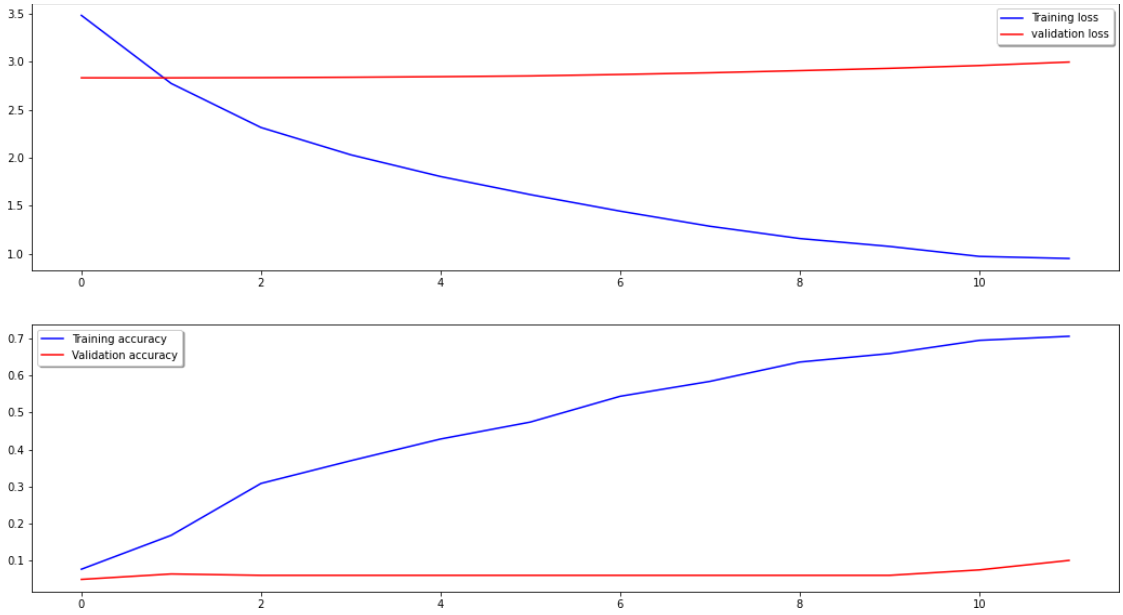
2.4 *Model Loss*

[14]:
```
import matplotlib.pyplot as plt

plt.plot(history_alexnet_batchnorm.history['acc'], label='accuracy')
plt.plot(history_alexnet_batchnorm.history['val_acc'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
```



2.5 *Model Accuracy*

[15]:
```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1, figsize=(18, 10))
ax[0].plot(history_alexnet_batchnorm.history['loss'], color='b',
  label="Training loss")
ax[0].plot(history_alexnet_batchnorm.history['val_loss'], color='r',
  label="validation loss",axes=ax[0])
legend = ax[0].legend(loc='best', shadow=True)
ax[1].plot(history_alexnet_batchnorm.history['acc'], color='b', label="Training
  accuracy")
ax[1].plot(history_alexnet_batchnorm.history['val_acc'],
  color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```



2.6 *Alexnet using MNIST dataset*

[16]:
```
from keras.datasets import mnist
# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

2.6.1 *Reshaping the dataset*

[17]:
```
# Preprocess the data
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")
x_train /= 255
x_test /= 255
y_train = keras.utils.np_utils.to_categorical(y_train, 10)
y_test = keras.utils.np_utils.to_categorical(y_test, 10)
```

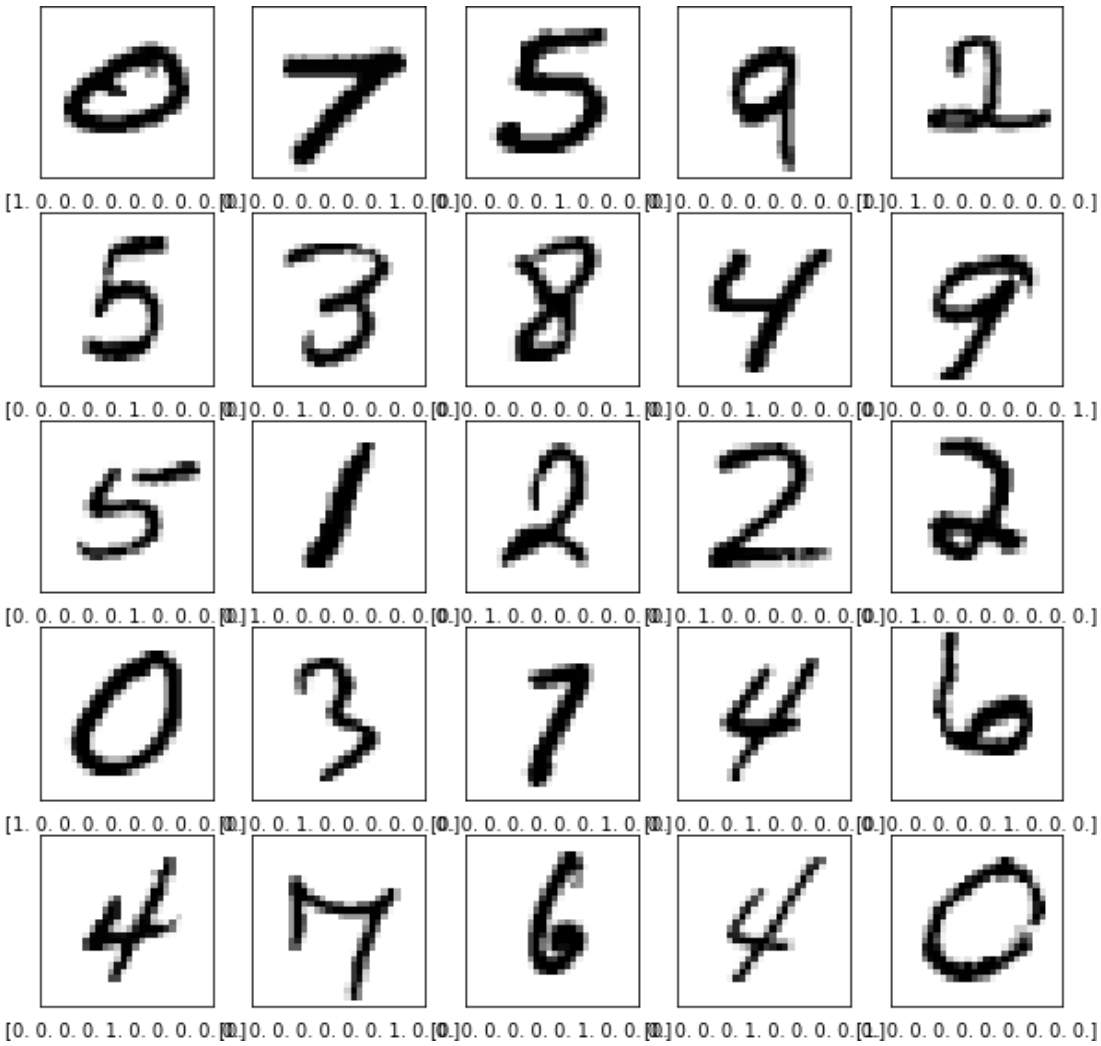
[18]:
```
import numpy as np

def plot_images_sample(X, Y):
    # Draw plot for images sample
    plt.figure(figsize=(10,10))
    rand_indicies = np.random.randint(len(X), size=25)
    for i in range(25):
        plt.subplot(5,5,i+1)
```

```
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        index = rand_indicies[i]
        plt.imshow(np.squeeze(X[index]), cmap=plt.cm.binary)
        plt.xlabel(Y[index])
    plt.show()
```

2.7 *Train Dataset*

[19]:
```
# Draw plot for images sample
plot_images_sample(x_train,y_train)
```

/usr/local/lib/python3.8/dist-packages/matplotlib/text.py:1223: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  if s != self._text:



3 *Model Architecture*

[20]:
```
# Define the AlexNet model

model_alex = Sequential()
model_alex.add(Conv2D(32, kernel_size=(3, 3), activation='relu',␣
 ↪input_shape=(28, 28, 1)))
model_alex.add(MaxPooling2D(pool_size=(2, 2)))
model_alex.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model_alex.add(MaxPooling2D(pool_size=(2, 2)))

model_alex.add(Flatten())
model_alex.add(Dense(128, activation='relu'))
model_alex.add(Dropout(0.5))
model_alex.add(Dense(10, activation='softmax'))
```

[21]:
```
model_alex.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_3 (MaxPooling 2D)	(None, 13, 13, 32)	0
conv2d_6 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_4 (MaxPooling 2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_3 (Dense)	(None, 128)	204928
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 10)	1290

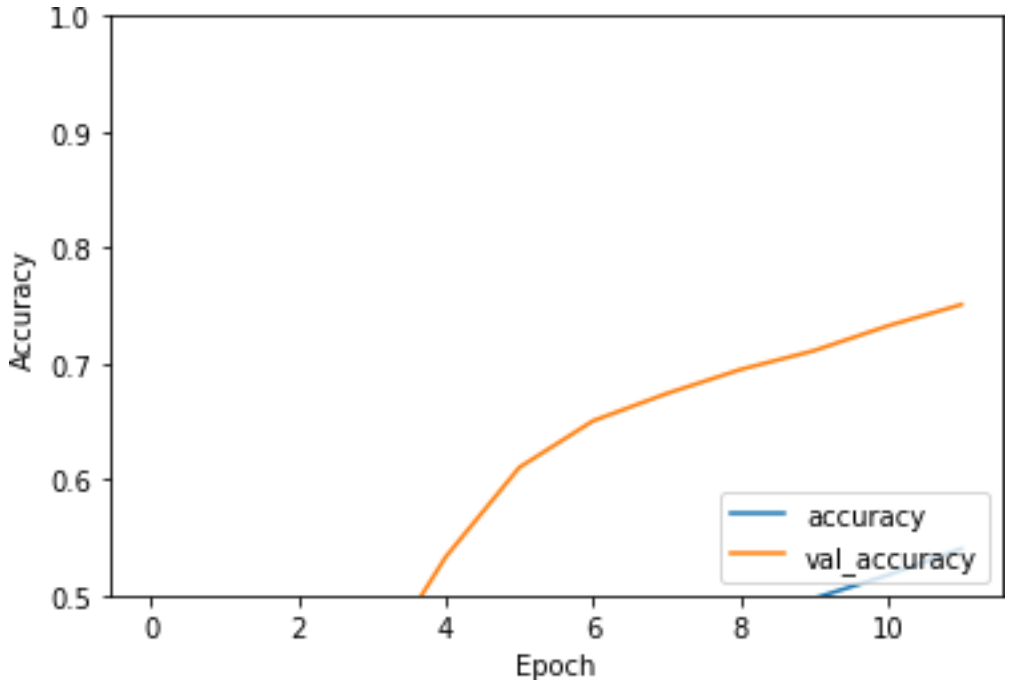
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0

[22]:
```
# Compile the model
model_alex.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.
    optimizers.Adadelta(), metrics=["accuracy"])

# Train the model
history_alexnet = model_alex.fit(x_train, y_train, batch_size=128, epochs=12,
    verbose=1, validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 3s 46us/sample - loss: 2.3166 -
acc: 0.1258 - val_loss: 2.2846 - val_acc: 0.2235
Epoch 2/12
60000/60000 [==============================] - 2s 36us/sample - loss: 2.2731 -
acc: 0.1628 - val_loss: 2.2443 - val_acc: 0.2562
Epoch 3/12
60000/60000 [==============================] - 2s 38us/sample - loss: 2.2360 -
acc: 0.2039 - val_loss: 2.2072 - val_acc: 0.3285
Epoch 4/12
60000/60000 [==============================] - 3s 53us/sample - loss: 2.1986 -
acc: 0.2527 - val_loss: 2.1689 - val_acc: 0.4315
Epoch 5/12
60000/60000 [==============================] - 2s 41us/sample - loss: 2.1625 -
acc: 0.3001 - val_loss: 2.1275 - val_acc: 0.5330
Epoch 6/12
60000/60000 [==============================] - 3s 55us/sample - loss: 2.1219 -
acc: 0.3475 - val_loss: 2.0818 - val_acc: 0.6103
Epoch 7/12
60000/60000 [==============================] - 4s 60us/sample - loss: 2.0767 -
acc: 0.3951 - val_loss: 2.0305 - val_acc: 0.6505
Epoch 8/12
60000/60000 [==============================] - 5s 90us/sample - loss: 2.0268 -
acc: 0.4321 - val_loss: 1.9731 - val_acc: 0.6741
Epoch 9/12
60000/60000 [==============================] - 4s 73us/sample - loss: 1.9714 -
acc: 0.4653 - val_loss: 1.9091 - val_acc: 0.6948
Epoch 10/12
60000/60000 [==============================] - 5s 81us/sample - loss: 1.9101 -
acc: 0.4972 - val_loss: 1.8394 - val_acc: 0.7111
Epoch 11/12
60000/60000 [==============================] - 5s 86us/sample - loss: 1.8474 -
acc: 0.5173 - val_loss: 1.7642 - val_acc: 0.7325
Epoch 12/12
60000/60000 [==============================] - 4s 73us/sample - loss: 1.7755 -
acc: 0.5402 - val_loss: 1.6838 - val_acc: 0.7509
```

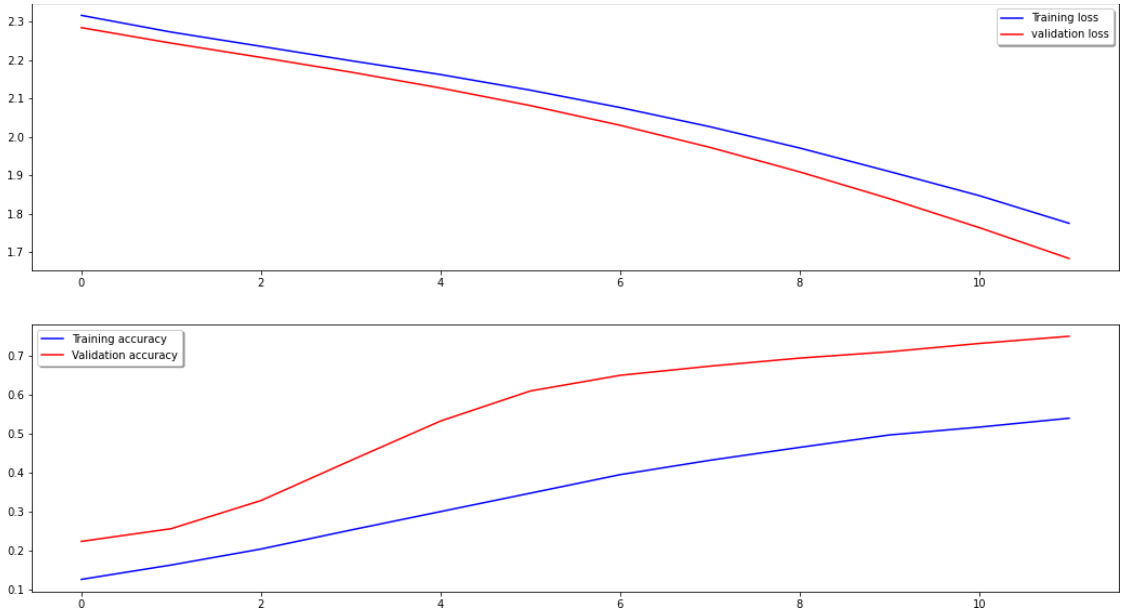
3.1 *Model Loss*

[23]:
```
plt.plot(history_alexnet.history['acc'], label='accuracy')
plt.plot(history_alexnet.history['val_acc'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
test_loss, test_acc = model_alex.evaluate(x_test, y_test, verbose=2)
```



3.2 *Model Accuracy*

[24]:
```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1, figsize=(18, 10))
ax[0].plot(history_alexnet.history['loss'], color='b', label="Training loss")
ax[0].plot(history_alexnet.history['val_loss'], color='r', label="validation
 loss",axes=ax[0])
legend = ax[0].legend(loc='best', shadow=True)
ax[1].plot(history_alexnet.history['acc'], color='b', label="Training accuracy")
ax[1].plot(history_alexnet.history['val_acc'], color='r',label="Validation
 accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```

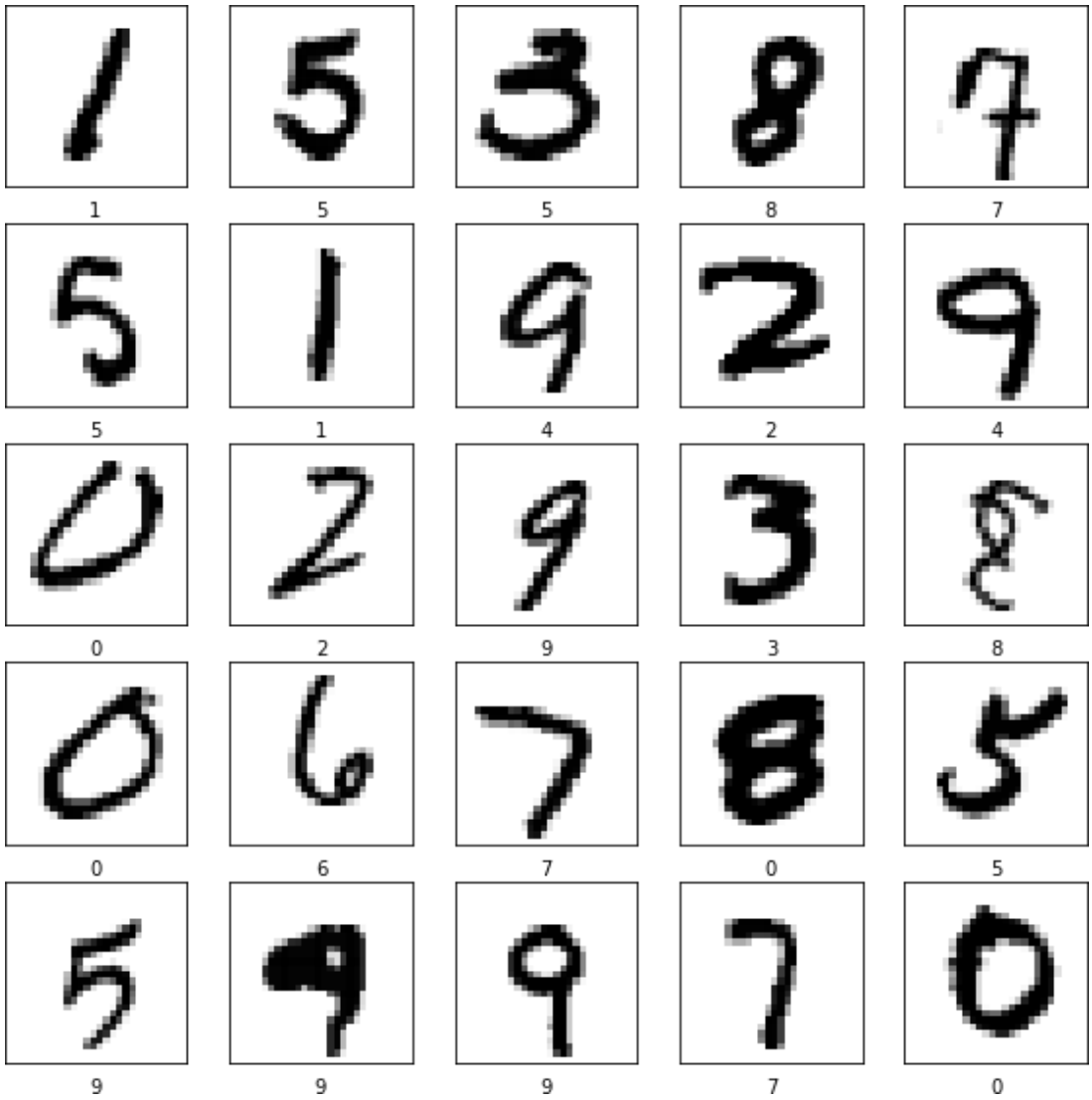


[25]:
```
def get_predictions(X_test):
    # Digits prediction
    predictions_alex = model_alex.predict(X_test)
    predictions_alex = np.argmax(predictions_alex, axis=1)
    return predictions_alex
```

3.3 *Final predictions by the Model*

[26]:
```
# Prediction and display it
predictions_alex = get_predictions(x_test)
plot_images_sample(x_test, predictions_alex)
```

```
/usr/local/lib/python3.8/dist-packages/keras/engine/training_v1.py:2357:
UserWarning: `Model.state_updates` will be removed in a future version. This
property should not be used in TensorFlow 2.0, as `updates` are applied
automatically.
  updates=self.state_updates,
```



[26]:

Author: Shobhandeb Paul Github: https://github.com/herbert0419/ Linkedin: https://www.linkedin.com/in/shobhandeb-paul-b6914a168/

[ ]: