Ex No: 3	Building Deep Neural Network for image classification

AIM:

To build a deep neural network for image classification.

PROCEDURE:

1. Import the required libraries and packages.
2. Load the MNIST digit image dataset from Keras.
3. Upgrade Keras and install np_utils if necessary.
4. Build the neural network model.
5. Flatten the images to convert them into one-dimensional arrays.
6. Split the data into training and testing sets.
7. Normalize the pixel values of the images to facilitate training.
8. Apply one-hot encoding to the class labels using Keras's utilities.
9. Build a linear stack of layers with the sequential model.
10. Use weight initializers (optional).
11. Use Optimizer (optional).
12. Train and test the model.
13. Perform a live prediction with a sample image.

CODE:

```
from keras.datasets import mnist
# loading the dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# let's print the shape of the dataset
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)
pip install np_utils
!pip install --upgrade keras
# keras imports for the dataset and building our neural network
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D
from keras.utils import to_categorical
# Flattening the images from the 28x28 pixels to 1D 787 pixels
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# normalizing the data to help with the training
X_train /= 255
X_test /= 255
# one-hot encoding using keras' numpy-related utilities
n_classes = 10
print("Shape before one-hot encoding: ", y_train.shape)
Y_train = to_categorical(y_train, n_classes)
Y_test = to_categorical(y_test, n_classes)
print("Shape after one-hot encoding: ", Y_train.shape)
import tensorflow as tf
# building a linear stack of layers with the sequential model
model = Sequential()
# hidden layer
model.add(Dense(100, input_shape=(784,), activation='relu'))
# output layer
model.add(Dense(10, activation='softmax'))
# looking at the model summary
model.summary()
# compiling the sequential model
model.compile(loss='categorical_crossentropy',
metrics=['accuracy'], optimizer='adam')
# training the model for 10 epochs
model.fit(X_train, Y_train, batch_size=128, epochs=10,
validation_data=(X_test, Y_test))
model.save('final_model.keras')
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
```

```
import numpy as np

# Load and prepare the image
def load_image(sample_image):
    # Load the image
    img = load_img(sample_image, color_mode='grayscale',
target_size=(28, 28))
    print("Loaded image shape:", img.size)
    # Convert to array
    img = img_to_array(img)
    print("Image array shape after conversion:", img.shape)
    # Flatten the image array
    img = img.reshape((1, 784))
    # Prepare pixel data
    img = img.astype('float32')
    img = img / 255.0
    return img

# Load an image and predict the class
def run_example(model):  # Pass the model object as an
argument
    # Load the image
    img = load_image('sample_image.jpg')
    predict_value = model.predict(img)
    digit = np.argmax(predict_value)
    print("Predicted digit:", digit)

# Load the model
model = load_model('final_model.keras')
# Entry point, run the example
run_example(model)  # Pass the model object to the
run_example function
```

OUTPUT:

Predicted Output: **7**

RESULT:

Model : "sequential"

No of epochs	Training Accuracy	Testing Accuracy	Training Loss	Testing Loss
1	0.8211	0.9408	0.6509	0.2012
2	0.9465	0.9567	0.1886	0.1471
3	0.9608	0.9647	0.1325	0.1170
4	0.9705	0.9701	0.1024	0.1037
5	0.9749	0.9707	0.0861	0.0968
6	0.9800	0.9735	0.0708	0.0870
7	0.9830	0.9759	0.0601	0.0816
8	0.9856	0.9756	0.0515	0.0793
9	0.9878	0.9746	0.0447	0.0771
10	0.9897	0.9770	0.0378	0.0759

CONCLUSION:

A deep neural network for image classification has been built and trained and tested on MNIST Digit dataset and output has been obtained successfully.