Ex.No.3

BUILD A CONVOLUTIONAL NEURAL NETWORK

AIM:

To build a simple convolutional neural network using Keras/TensorFlow

PROCEDURE:

1. Create a dataset using rand() and randint() for the independent variable and dependent variable respectively.
2. Split the dataset into training data and test data.
3. Build a convolutional neural network model using Keras/TensorFlow.
4. Compile and fit the model on the training data and use the test data as the validation data.
5. Perform prediction with the test data.
6. Calculate performance metrics.
7. Plot the graph of training accuracy and validation accuracy, and training loss and validation loss.

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from sklearn.model_selection import train_test_split

np.random.seed(42)
```

```
X = np.random.rand(1000, 28, 28, 3)
y = np.random.randint(0, 10, 1000)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

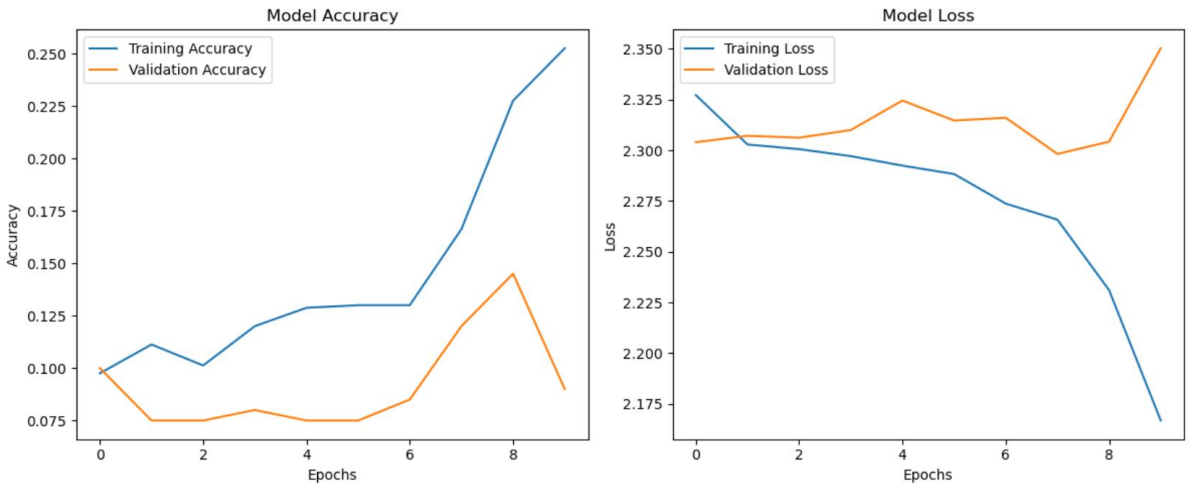
```
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()
```

OUTPUT:

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.0984 - loss: 2.3462 - val_accuracy: 0.1000 - val_loss: 2.3040
Epoch 2/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.1082 - loss: 2.3006 - val_accuracy: 0.0750 - val_loss: 2.3072
Epoch 3/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.1212 - loss: 2.2979 - val_accuracy: 0.0750 - val_loss: 2.3063
Epoch 4/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.1128 - loss: 2.2969 - val_accuracy: 0.0800 - val_loss: 2.3100
Epoch 5/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.1324 - loss: 2.2918 - val_accuracy: 0.0750 - val_loss: 2.3246
Epoch 6/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.1296 - loss: 2.2895 - val_accuracy: 0.0750 - val_loss: 2.3147
Epoch 7/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.1375 - loss: 2.2745 - val_accuracy: 0.0850 - val_loss: 2.3161
Epoch 8/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1935 - loss: 2.2542 - val_accuracy: 0.1200 - val_loss: 2.2982
Epoch 9/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.2921 - loss: 2.2319 - val_accuracy: 0.1450 - val_loss: 2.3043
Epoch 10/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.2290 - loss: 2.1821 - val_accuracy: 0.0900 - val_loss: 2.3503
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0791 - loss: 2.3551
Test Loss: 2.3503401279449463
Test Accuracy: 0.09000000357627869
```



RESULT:

Thus, a convolutional neural network using Keras/TensorFlow was built successfully.