```
# Importing required libraries and modules
import numpy as np
import pandas as pd
import keras
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.optimizers import Adam
from keras.utils import to_categorical


(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

num_classes = 10

# Flatten the data to 4 dimensions: samples, width, height, features
train_images = train_images.reshape(train_images.shape[0], 32, 32, 3)
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape(test_images.shape[0], 32, 32, 3)
test_images = test_images.astype('float32') / 255

# One hot encode the labels
train_labels = to_categorical(train_labels, num_classes)
test_labels = to_categorical(test_labels, num_classes)


model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(32, 32, 3)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))


model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=Adam(learning_rate=0.001),
              metrics=['accuracy'])


batch_size = 128

model.fit(train_images, train_labels,
          batch_size=batch_size,
          epochs=20,
          verbose=1,
          validation_data=(test_images, test_labels))
```

```
Epoch 1/20
391/391 [==============================] - 14s 23ms/step - loss: 1.6819 - accuracy:
Epoch 2/20
391/391 [==============================] - 7s 18ms/step - loss: 1.3280 - accuracy: 0
Epoch 3/20
391/391 [==============================] - 6s 16ms/step - loss: 1.1890 - accuracy: 0
Epoch 4/20
391/391 [==============================] - 5s 12ms/step - loss: 1.1015 - accuracy: 0
Epoch 5/20
391/391 [==============================] - 5s 13ms/step - loss: 1.0445 - accuracy: 0
Epoch 6/20
391/391 [==============================] - 5s 12ms/step - loss: 0.9930 - accuracy: 0
Epoch 7/20
391/391 [==============================] - 5s 12ms/step - loss: 0.9479 - accuracy: 0
Epoch 8/20
391/391 [==============================] - 5s 12ms/step - loss: 0.9203 - accuracy: 0
Epoch 9/20
391/391 [==============================] - 4s 11ms/step - loss: 0.8763 - accuracy: 0
Epoch 10/20
391/391 [==============================] - 5s 12ms/step - loss: 0.8555 - accuracy: 0
Epoch 11/20
391/391 [==============================] - 5s 12ms/step - loss: 0.8190 - accuracy: 0
Epoch 12/20
391/391 [==============================] - 4s 11ms/step - loss: 0.7959 - accuracy: 0
Epoch 13/20
391/391 [==============================] - 5s 12ms/step - loss: 0.7715 - accuracy: 0
Epoch 14/20
391/391 [==============================] - 4s 11ms/step - loss: 0.7482 - accuracy: 0
Epoch 15/20
391/391 [==============================] - 5s 12ms/step - loss: 0.7290 - accuracy: 0
Epoch 16/20
391/391 [==============================] - 5s 12ms/step - loss: 0.7105 - accuracy: 0
Epoch 17/20
391/391 [==============================] - 5s 12ms/step - loss: 0.6895 - accuracy: 0
Epoch 18/20
391/391 [==============================] - 5s 13ms/step - loss: 0.6742 - accuracy: 0
Epoch 19/20
391/391 [==============================] - 5s 12ms/step - loss: 0.6621 - accuracy: 0
Epoch 20/20
391/391 [==============================] - 5s 12ms/step - loss: 0.6445 - accuracy: 0
<keras.src.callbacks.History at 0x7ede467da110>
```

```python
# Assuming that test_images contains the data you want to use for prediction
predictions = model.predict(test_images)
Results = model.evaluate(test_images, test_labels, batch_size=batch_size)
print("Testing Results:", Results)
```

```
313/313 [==============================] - 1s 2ms/step
79/79 [==============================] - 0s 5ms/step - loss: 0.8844 - accuracy: 0.70
Testing Results: [0.8844407200813293, 0.7067999839782715]
```

```python
predictions_train = model.predict(train_images)
Results = model.evaluate(train_images, train_labels)
print("Training Results:", Results)
```

```
1563/1563 [==============================] - 4s 2ms/step
1563/1563 [==============================] - 5s 3ms/step - loss: 0.3344 - accuracy:
Training Results: [0.33444735407829285, 0.9061200022697449]
```

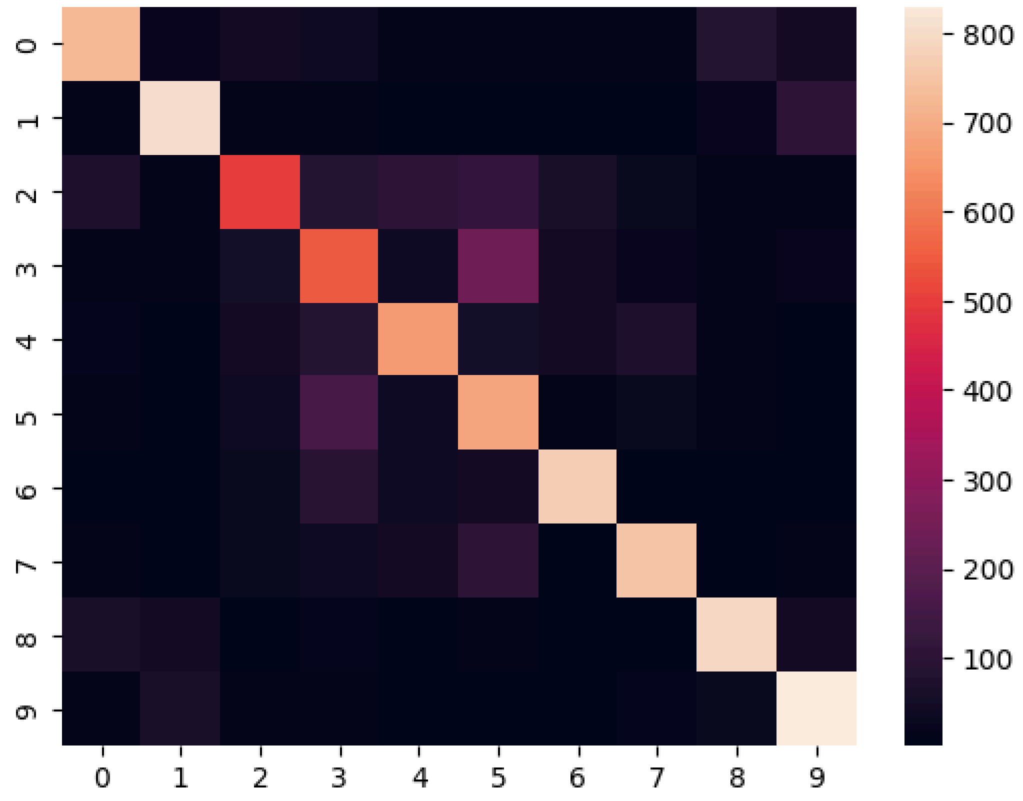
```
predictions[1]
```

```
array([1.8990700e-03, 1.7876151e-01, 1.1856915e-09, 2.8110855e-10,
       1.0656533e-11, 2.2617233e-14, 2.7371031e-14, 3.3971313e-12,
       8.1933844e-01, 1.0085898e-06], dtype=float32)
```

Test Confusion Matrix :

```
import seaborn as sns
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(test_labels.argmax(axis=1), predictions.argma
sns.heatmap(confusion_matrix)
```

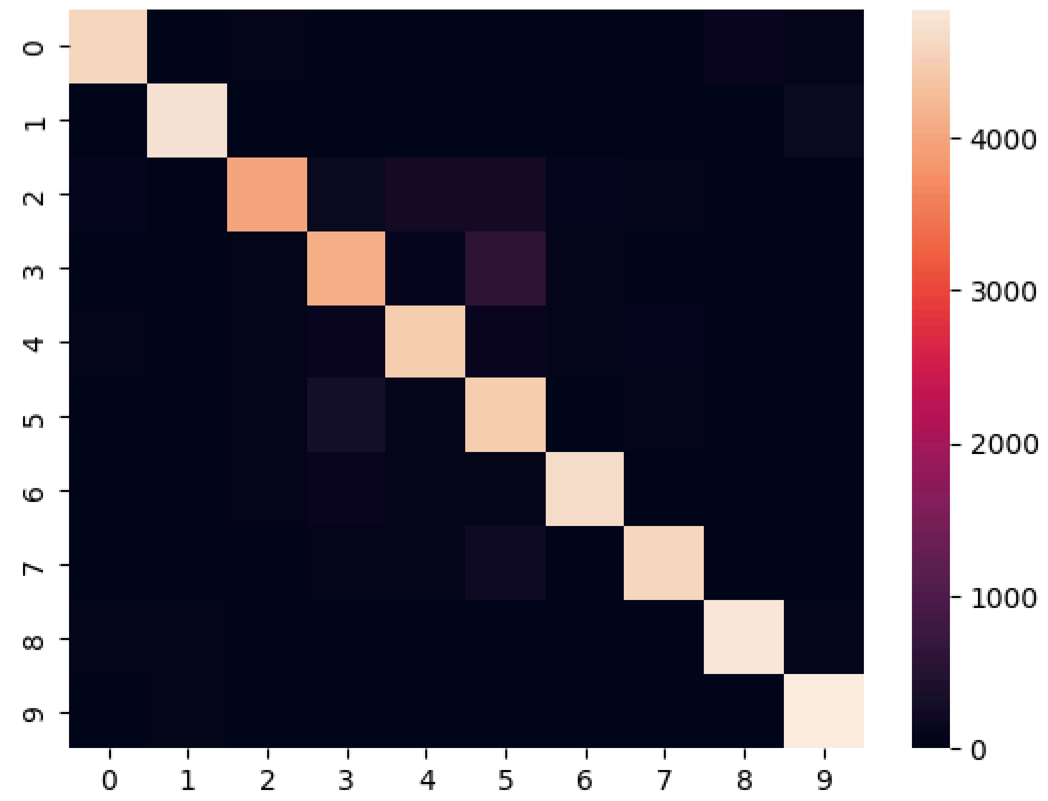
```
<Axes: >
```



Train Confusion Matrix

```
confusion_matrix = metrics.confusion_matrix(train_labels.argmax(axis=1), predictions_tra
sns.heatmap(confusion_matrix)
```

```
<Axes: >
```



```
Start coding or generate with AI.
```