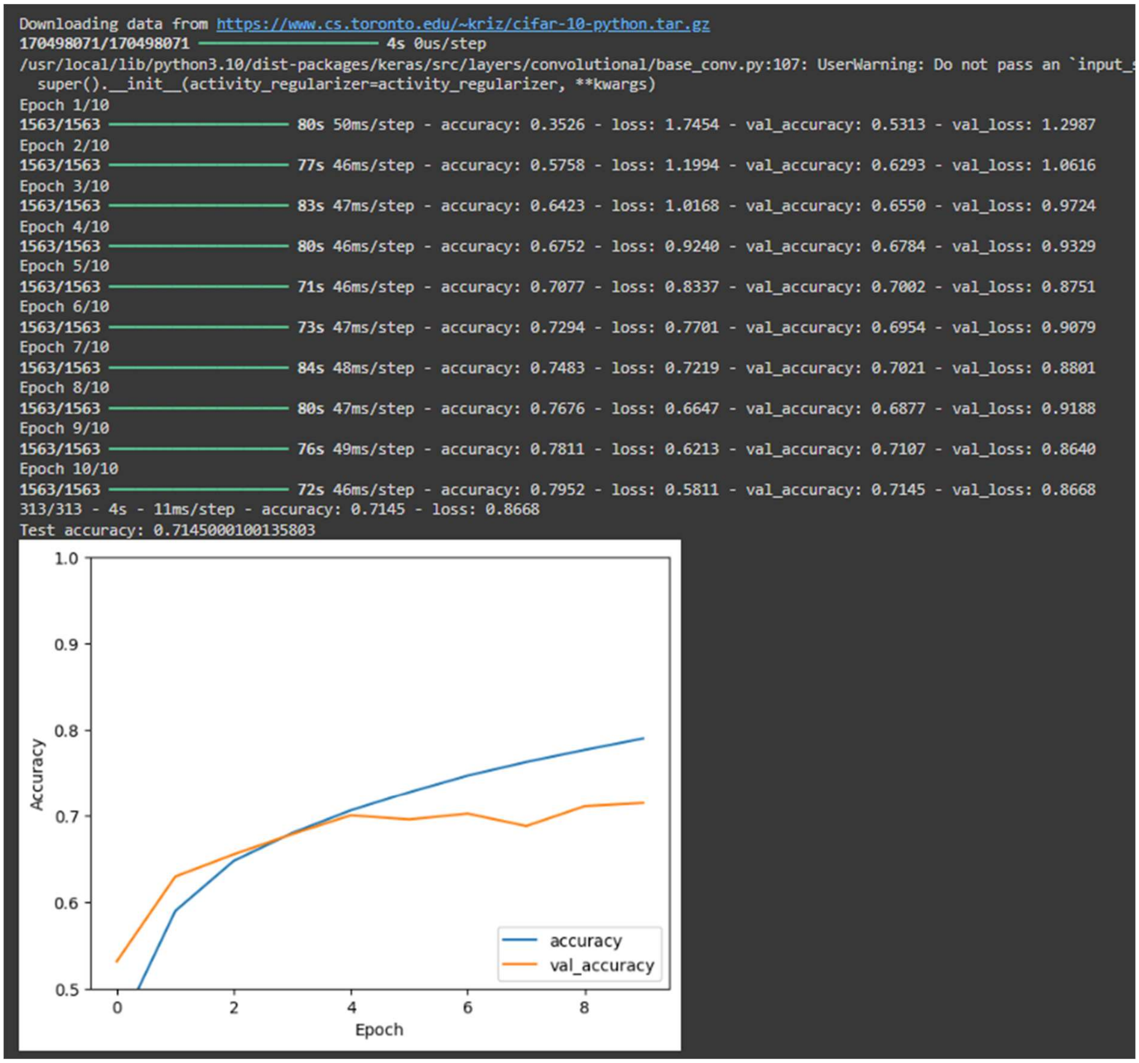
Practical 11

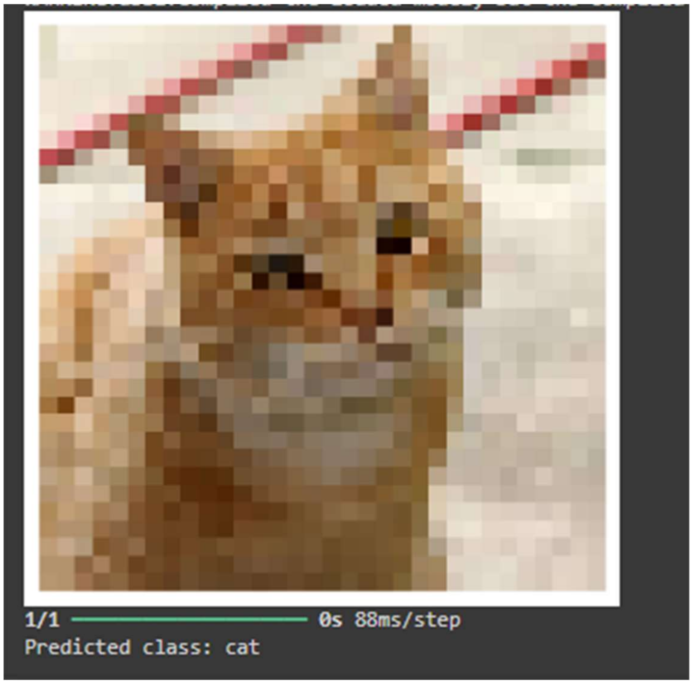
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
# Load and preprocess the CIFAR10 dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0

# Define the CNN model
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")

# Plot training history (optional)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
plt.show()
```



```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_s
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 80s 50ms/step - accuracy: 0.3526 - loss: 1.7454 - val_accuracy: 0.5313 - val_loss: 1.2987
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 77s 46ms/step - accuracy: 0.5758 - loss: 1.1994 - val_accuracy: 0.6293 - val_loss: 1.0616
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 83s 47ms/step - accuracy: 0.6423 - loss: 1.0168 - val_accuracy: 0.6550 - val_loss: 0.9724
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 80s 46ms/step - accuracy: 0.6752 - loss: 0.9240 - val_accuracy: 0.6784 - val_loss: 0.9329
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 71s 46ms/step - accuracy: 0.7077 - loss: 0.8337 - val_accuracy: 0.7002 - val_loss: 0.8751
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 73s 47ms/step - accuracy: 0.7294 - loss: 0.7701 - val_accuracy: 0.6954 - val_loss: 0.9079
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 84s 48ms/step - accuracy: 0.7483 - loss: 0.7219 - val_accuracy: 0.7021 - val_loss: 0.8801
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 80s 47ms/step - accuracy: 0.7676 - loss: 0.6647 - val_accuracy: 0.6877 - val_loss: 0.9188
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 76s 49ms/step - accuracy: 0.7811 - loss: 0.6213 - val_accuracy: 0.7107 - val_loss: 0.8640
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 72s 46ms/step - accuracy: 0.7952 - loss: 0.5811 - val_accuracy: 0.7145 - val_loss: 0.8668
313/313 - 4s - 11ms/step - accuracy: 0.7145 - loss: 0.8668
Test accuracy: 0.7145000100135803
```

Object detection using CNN

```
import tensorflow as tf

import numpy as np

import cv2

from tensorflow.keras.losses import mse


# Load the saved model

model = tf.keras.models.load_model('my_cifar10_model.h5', custom_objects={'mse': mse})


# Define class names for CIFAR-10

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',

               'dog', 'frog', 'horse', 'ship', 'truck']


# Display the uploaded image

plt.imshow(img)

plt.axis('off')
plt.show()



# Load and preprocess an image (replace with your own image)
img_path = '/content/download.jpg'  # Replace with the path to your image
img = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (32, 32))  # Resize to CIFAR-10 input size
img = img / 255.0  # Normalize


# Make a prediction
img_array = np.expand_dims(img, axis=0)
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions[0])

# Display the result
print(f"Predicted class: {class_names[predicted_class]}")
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
Predicted class: cat
```