AIM

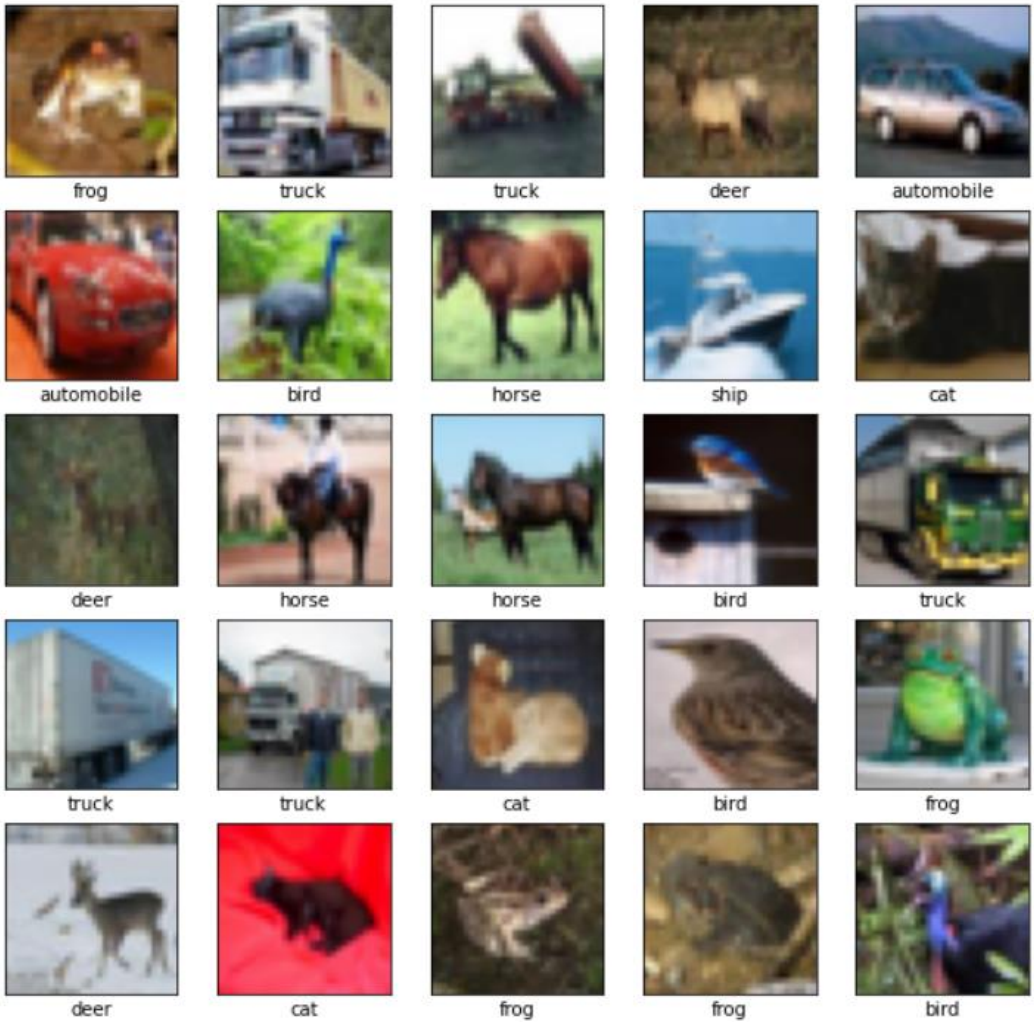
Programs on convolutional neural network to classify images from any standard dataset in the public domain.

Dataset used: cifar10

Programming code:

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

#The CIFAR10 dataset contains 60,000 color images in 10 classes, wi
th 6,000 images in each class
(train_images, train_labels), (test_images, test_labels) = datasets
.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255
.0

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

OUTPUT:



Programming code:

```
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.summary()
```

OUTPUT:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

Programming code:

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

OUTPUT:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Programming code:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(fr
om_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

OUTPUT:

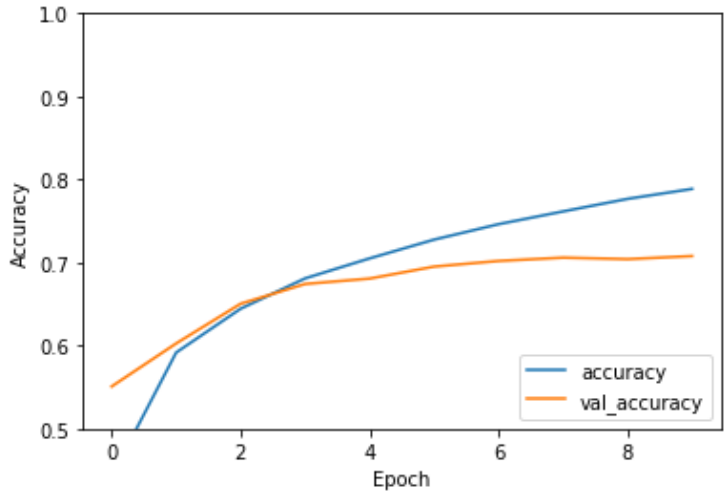
```
Epoch 1/10
1563/1563 [==============================] - 93s 59ms/step - loss: 1.5275 - accuracy: 0.4426 - val_loss: 1.2727 - val_accuracy: 0.5508
Epoch 2/10
1563/1563 [==============================] - 74s 47ms/step - loss: 1.1513 - accuracy: 0.5916 - val_loss: 1.1043 - val_accuracy: 0.6026
Epoch 3/10
1563/1563 [==============================] - 72s 46ms/step - loss: 1.0104 - accuracy: 0.6444 - val_loss: 1.0100 - val_accuracy: 0.6504
Epoch 4/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.9112 - accuracy: 0.6809 - val_loss: 0.9435 - val_accuracy: 0.6742
Epoch 5/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.8387 - accuracy: 0.7050 - val_loss: 0.9262 - val_accuracy: 0.6807
Epoch 6/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.7794 - accuracy: 0.7276 - val_loss: 0.8774 - val_accuracy: 0.6951
Epoch 7/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.7285 - accuracy: 0.7462 - val_loss: 0.8627 - val_accuracy: 0.7019
Epoch 8/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.6793 - accuracy: 0.7616 - val_loss: 0.8734 - val_accuracy: 0.7059
Epoch 9/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.6346 - accuracy: 0.7767 - val_loss: 0.8788 - val_accuracy: 0.7041
Epoch 10/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.6008 - accuracy: 0.7887 - val_loss: 0.8842 - val_accuracy: 0.7078
```

Programming code:

```
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')

test_loss, test_acc = model.evaluate(test_images,  test_labels, ver
bose=2)
```

OUTPUT:

```
313/313 - 4s - loss: 0.8842 - accuracy: 0.7078 - 4s/epoch - 12ms/step
```



Programming code:

```
print(test_acc)
```

OUTPUT:

```
0.7077999711036682
```