```
import tensorflow as tf
import os
import numpy as np

from matplotlib import pyplot as plt
%matplotlib inline

if not os.path.isdir('models'):
    os.mkdir('models')

print('TensorFlow version:', tf.__version__)
print('Is using GPU?', tf.test.is_gpu_available())
```

```
def get_three_classes(x, y):
    def indices_of(class_id):
        indices, _ = np.where(y == float(class_id))
        return indices

    indices = np.concatenate([indices_of(0), indices_of(1),
indices_of(2)], axis=0)

    x = x[indices]
    y = y[indices]

    count = x.shape[0]
    indices = np.random.choice(range(count), count, replace=False)

    x = x[indices]
    y = y[indices]

    y = tf.keras.utils.to_categorical(y)

    return x, y
```

```
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.cifar10.load_data()

x_train, y_train = get_three_classes(x_train, y_train)
x_test, y_test = get_three_classes(x_test, y_test)

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
class_names = ['aeroplane', 'car', 'bird']

def show_random_examples(x, y, p):
    indices = np.random.choice(range(x.shape[0]), 10, replace=False)
    x = x[indices]
    y = y[indices]
    p = p[indices]

    plt.figure(figsize=(10, 5))
    for i in range(10):
        plt.subplot(2, 5, i + 1)
        plt.imshow(x[i])
        plt.xticks([])
        plt.yticks([])
        col = 'green' if np.argmax(y[i]) == np.argmax(p[i]) else 'red'
        plt.xlabel(class_names[np.argmax(p[i])], color=col)
    plt.show()

show_random_examples(x_train, y_train, y_train)
```

```
show_random_examples(x_test, y_test, y_test)
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
BatchNormalization
from tensorflow.keras.layers import Dropout, Flatten, Input, Dense

def create_model():

    def add_conv_block(model, num_filters):

        model.add(Conv2D(num_filters, 3, activation='relu',
padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(num_filters, 3, activation='relu',
padding='valid'))
        model.add(MaxPooling2D(pool_size=2))
        model.add(Dropout(0.2))

        return model

    model = tf.keras.models.Sequential()
    model.add(Input(shape=(32, 32, 3)))

    model = add_conv_block(model, 32)
    model = add_conv_block(model, 64)
    model = add_conv_block(model, 128)

    model.add(Flatten())
    model.add(Dense(3, activation='softmax'))

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

```
    return model

model = create_model()
model.summary()

%%time

h = model.fit(
    x_train/255., y_train,
    validation_data=(x_test/255., y_test),
    epochs=20, batch_size=256,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',
patience=2),

tf.keras.callbacks.ModelCheckpoint('models/model_{val_accuracy:.3f}.h5'
, save_best_only=True,
                                          save_weights_only=False,
monitor='val_accuracy')
    ]
)
```

```
losses = h.history['loss']
accs = h.history['accuracy']
val_losses = h.history['val_loss']
val_accs = h.history['val_accuracy']
epochs = len(losses)

plt.figure(figsize=(12, 4))
for i, metrics in enumerate(zip([losses, accs], [val_losses, val_accs],
['Loss', 'Accuracy'])):
    plt.subplot(1, 2, i + 1)
    plt.plot(range(epochs), metrics[0], label='Training
{}'.format(metrics[2]))
    plt.plot(range(epochs), metrics[1], label='Validation
{}'.format(metrics[2]))
    plt.legend()
plt.show()
```

```
model = tf.keras.models.load_model('models/model_0.913.h5')
preds = model.predict(x_test/255.)
```

```
show_random_examples(x_test, y_test, preds)
```

Output –

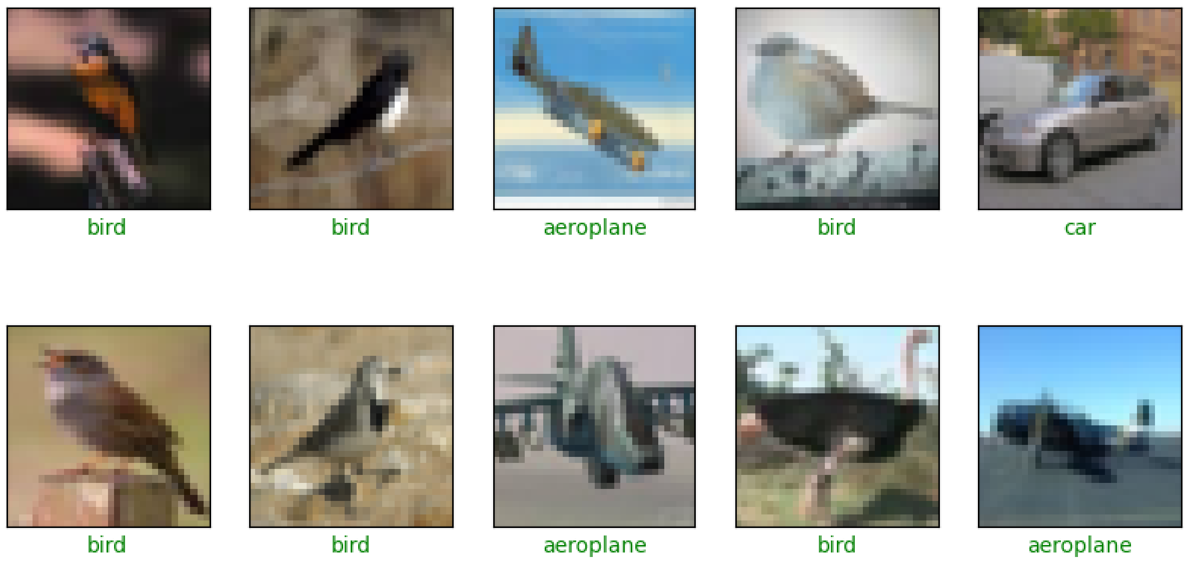
```
WARNING:tensorflow:From <ipython-input-1-20b300c48c1c>:12: is_gpu_available (from tensorflow.python.framework.test_util) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
TensorFlow version: 2.15.0
Is using GPU? False
```

[3]

6s

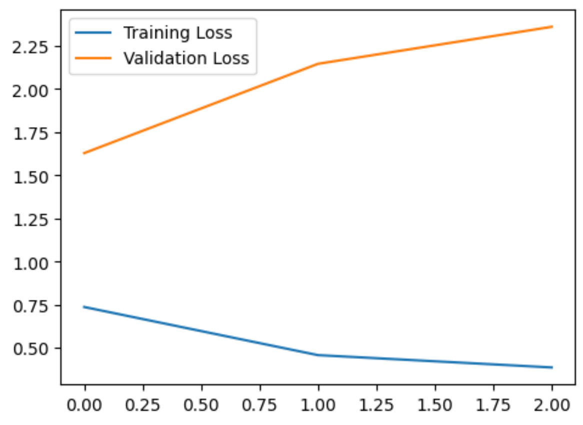
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_d
ata()
x_train, y_train = get_three_classes(x_train, y_train)
x_test, y_test = get_three_classes(x_test, y_test)
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

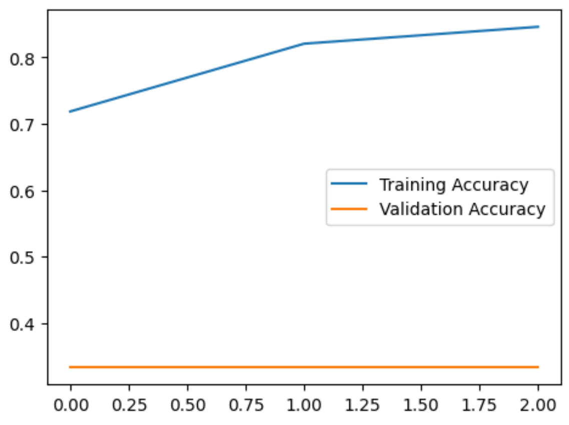
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
(15000, 32, 32, 3) (15000, 3)
(3000, 32, 32, 3) (3000, 3)
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896       
                                                                 
 batch_normalization (Batch  (None, 32, 32, 32)        128       
 Normalization)                                                  
                                                                 
 conv2d_1 (Conv2D)           (None, 30, 30, 32)        9248      
                                                                 
 max_pooling2d (MaxPooling2  (None, 15, 15, 32)        0         
 D)                                                              
 dropout (Dropout)            (None, 15, 15, 32)        0

 conv2d_2 (Conv2D)            (None, 15, 15, 64)        18496

 batch_normalization_1 (Bat   (None, 15, 15, 64)        256
 chNormalization)

 conv2d_3 (Conv2D)            (None, 13, 13, 64)        36928

 max_pooling2d_1 (MaxPoolin   (None, 6, 6, 64)          0
 g2D)

 dropout_1 (Dropout)          (None, 6, 6, 64)          0

 conv2d_4 (Conv2D)            (None, 6, 6, 128)         73856

 batch_normalization_2 (Bat   (None, 6, 6, 128)         512
 chNormalization)

 conv2d_5 (Conv2D)            (None, 4, 4, 128)         147584

 max_pooling2d_2 (MaxPoolin   (None, 2, 2, 128)         0
 g2D)

 dropout_2 (Dropout)          (None, 2, 2, 128)         0

 flatten (Flatten)            (None, 512)               0

 dense (Dense)                (None, 3)                 1539

=================================================================
Total params: 289443 (1.10 MB)
Trainable params: 288995 (1.10 MB)
Non-trainable params: 448 (1.75 KB)
```

```
Epoch 1/20
59/59 [==============================] - 90s 1s/step - loss: 0.7372 - accuracy: 0.7186 - val_loss:
1.6283 - val_accuracy: 0.3333
Epoch 2/20
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are
saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
45/59 [=====================>........] - ETA: 19s - loss: 0.4654 - accuracy: 0.8179
```





```
94/94 [==============================] - 5s 45ms/step
```