```
!unzip "/content/drive/My Drive/LeNet/LeNet.zip" -d "/content/drive/My Drive/LeNet"
```

```
Archive:  /content/drive/My Drive/LeNet/LeNet.zip
replace /content/drive/My Drive/LeNet/content/.config/configurations/config_default? [y]
```

```
cd drive/MyDrive/LeNet/content/
```

```
/content/drive/MyDrive/LeNet/content
```

```
import tensorflow as tf
import tensorflow_datasets as tfds
import cifar_utils
import matplotlib.pyplot as plt
from  tensorflow.keras import datasets,layers,models,losses
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import tensorflow_hub as hub
import collections
import functools
import math
import datetime,os
import functools
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-8062ea8029f2> in <module>()
      1 import tensorflow as tf
      2 import tensorflow_datasets as tfds
----> 3 import cifar_utils
      4 import matplotlib.pyplot as plt
      5 from  tensorflow.keras import datasets,layers,models,losses

ModuleNotFoundError: No module named 'cifar_utils'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

SEARCH STACK OVERFLOW

```
BATCH_SIZE = 32
EPOCHS =300
Inception_input_shape = [299,299,3]
random_seed = 42
```

```
%load_ext tensorboard


cifar_info = cifar_utils.get_info()
print(cifar_info)
#train_data , test_data = datasets['train'],datasets['test']

     tfds.core.DatasetInfo(
         name='cifar10',
         version=3.0.2,
         description='The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classe
         homepage='https://www.cs.toronto.edu/~kriz/cifar.html',
         features=FeaturesDict({
             'id': Text(shape=(), dtype=tf.string),
             'image': Image(shape=(32, 32, 3), dtype=tf.uint8),
             'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10),
         }),
         total_num_examples=60000,
         splits={
             'test': 10000,
             'train': 50000,
         },
         supervised_keys=('image', 'label'),
         citation="""@TECHREPORT{Krizhevsky09learningmultiple,
             author = {Alex Krizhevsky},
             title = {Learning multiple layers of features from tiny images},
             institution = {},
             year = {2009}
         }""",
         redistribution_info=,
     )




cifar_info.features['label'].names

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


test_data,test_info = tfds.load('cifar10',split='test',with_info=True)
val_data ,val_info= tfds.load('cifar10',split='train[:10%]',with_info=True)
train_data ,train_info= tfds.load('cifar10',split='train[10%:100%]',with_info=True)


num_train_examples = 0
```

```
for _ in train_data:
  num_train_examples+=1
print(num_train_examples)
```

```
45000
```

```
num_val_examples=0
for _ in val_data:
  num_val_examples +=1
print(num_val_examples)
```

```
5000
```

```
tfds.as_dataframe(train_data.take(10),train_info)
```

	id	image	label
0	train_38425		7 (horse)
1	train_27616		9 (truck)
2	train_15323		8 (ship)
3	train_44111		7 (horse)
4	train_20390		3 (cat)
5	train_32655		4 (deer)
6	train_06033		5 (dog)
7	train_32287		7 (horse)
8	train_48868		9 (truck)
9	train_46296		9 (truck)

```
tfds.as_dataframe(val_data.take(10),val_info)
```

	id	image	label
0	train_16399		7 (horse)
1	train_01680		8 (ship)
2	train_47917		4 (deer)
3	train_17307		4 (deer)
4	train_27051		6 (frog)

```
def normalizer(features,input_shape = [299,299,3]):
  input_shape = tf.convert_to_tensor(input_shape)
  image = features['image']
  image = tf.image.convert_image_dtype(image,tf.float32)
  image = tf.image.resize(image,input_shape[:2])
  features['image'] = image
  return image
```

```
train_data = train_data.map(normalizer)


val_data = val_data.map(normalizer)


train_data = train_data.batch(BATCH_SIZE)
val_data = val_data.batch(BATCH_SIZE)
train_data = train_data.prefetch(1)
val_data = val_data.prefetch(1)
print(val_data)
```

```
<PrefetchDataset shapes: (None, 299, 299, 3), types: tf.float32>
```

```
LeNet = tf.keras.models.load_model("My_LeNet")


LeNet.summary()
```

```
Model: "LeNet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 2048)              192
_________________________________________________________________
logits_pred (Dense)          (None, 10)                20490
=================================================================
Total params: 20,682
Trainable params: 20,490
Non-trainable params: 192
_________________________________________________________________
```

```
num_train_imgs = train_info.splits['train'].num_examples
num_val_imgs = val_info.splits['test'].num_examples
train_steps_per_epoch = math.ceil(num_train_imgs/BATCH_SIZE)
val_steps_per_epoch = math.ceil(num_val_imgs/BATCH_SIZE)


import glob
import numpy as np
from classification_utils import load_image, process_predictions, display_predictions
inception_expected_input_shape = [299, 299, 3]
test_filenames = glob.glob(os.path.join('/content/drive/My Drive/LeNet/res', '*'))
test_images = np.asarray([load_image(file, size=inception_expected_input_shape[:2])
                          for file in test_filenames])
print('Test Images: {}'.format(test_images.shape))

image_batch = test_images[:16]

# Our model was trained on CIFAR images, which originally are 32x32px. We scaled them up
# to 224x224px to train our model on, but this means the resulting images had important
# artifacts/low quality.
# To test on images of the same quality, we first resize them to 32x32px, then to the
#expected input size (i.e., 224x224px):
cifar_original_image_size = cifar_info.features['image'].shape[:2]
class_readable_labels = cifar_info.features["label"].names
```

```
Test Images: (16, 299, 299, 3)
```

```
image_batch_low_quality = tf.image.resize(image_batch, cifar_original_image_size)
image_batch_low_quality = tf.image.resize(image_batch_low_quality, inception_expected_input_s

predictions = LeNet.predict_on_batch(image_batch_low_quality)
top5_labels, top5_probabilities = process_predictions(predictions, class_readable_labels)

print("Inception Predictions:")
display_predictions(image_batch, top5_labels, top5_probabilities)
```

Inception Predictions: