In [ ]:

```
#General imports
from __future__ import print_function
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import progressbar
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA

# Keras imports
import keras
from tensorflow.keras import layers
import tensorflow.keras

from keras.preprocessing.image import load_img
from keras.models import Sequential, Model
from tensorflow.keras.optimizers import *
from keras.utils.np_utils import to_categorical
import keras.backend as K

# application (model) imports
from tensorflow.keras import applications
#from keras.applications.inception_v3 import preprocess_input
from keras.layers import Dense

import tensorflow as tf
import tensorflow_datasets as tfds
```

Load data

In [ ]:

```
#load dataset
tfds.disable_progress_bar()

(train, test), info = tfds.load(
    'oxford_iiit_pet',
    split=['train', 'test'],
    with_info = True,
    shuffle_files=True)
```

```
Downloading and preparing dataset oxford_iiit_pet/3.2.0 (download: 773.52 MiB, generated:
774.69 MiB, total: 1.51 GiB) to /root/tensorflow_datasets/oxford_iiit_pet/3.2.0...
Shuffling and writing examples to /root/tensorflow_datasets/oxford_iiit_pet/3.2.0.incompl
ete2FCWGU/oxford_iiit_pet-train.tfrecord
Shuffling and writing examples to /root/tensorflow_datasets/oxford_iiit_pet/3.2.0.incompl
ete2FCWGU/oxford_iiit_pet-test.tfrecord
Dataset oxford_iiit_pet downloaded and prepared to /root/tensorflow_datasets/oxford_iiit_
pet/3.2.0. Subsequent calls will reuse this data.
```

Data processing

In [ ]:

```
# transform training data into numpy array
train_data = []
train_labels = []

for example in train:
  image = tf.image.resize(example["image"], [224, 224])
  train_data.append(image)
  train_labels.append(example["label"])
```

```
train_data = np.array(train_data)
train_labels = np.array(train_labels)

print(train_data.shape, train_labels.shape)
```

```
(3680, 224, 224, 3) (3680,)
```

In [ ]:

```
# transform test data into numpy array
test_data = []
test_labels = []

for example in test:
  image = tf.image.resize(example["image"], [224, 224])
  test_data.append(image)
  test_labels.append(example["label"])

test_data = np.array(test_data)
test_labels = np.array(test_labels)

print(test_data .shape, test_labels.shape)
```

```
(3669, 224, 224, 3) (3669,)
```

In [ ]:

```
from keras.applications.inception_v3 import preprocess_input
train_data = preprocess_input(train_data)
test_data = preprocess_input(test_data)
```

In [ ]:

```
#tidy up memory
import gc
gc.collect()
```

Out[ ]:

```
100
```

Transfer learning

In [ ]:

```
#set up base model
input_shape = (224,224,3)
num_classes=120

base_model = applications.inception_v3.InceptionV3(input_shape=input_shape,
                                                   include_top=False, weights='imagenet
', pooling='avg')

extraction_model = base_model


train_features = extraction_model.predict(train_data)
test_features = extraction_model.predict(test_data)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 2s 0us/step
87924736/87910968 [==============================] - 2s 0us/step
```

In [ ]:

```
print(train_features.shape)
print(test_features.shape)
```

```
print(train_features[0])
```

```
(3680, 2048)
(3669, 2048)
[0.43303147 0.01800284 0.05010956 ... 0.4942393  0.25515327 2.0667357 ]
```

In [ ]:

```
normalize = False
pca_reduce = False
n_components = 2


if normalize:
  train_features = normalize(train_features)
  test_features = normalize(test_features)

if pca_reduce:
  pca_transform = PCA(n_components= n_components, whiten=True)
  print("here")
  train_features = pca_transform.fit_transform(train_features)
  test_features = pca_transform.transform(test_features)

print(train_features.shape)
print(test_features.shape)
```

```
(3680, 2048)
(3669, 2048)
```

In [ ]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
```

In [ ]:

```
#clf = SVC(kernel='poly', gamma='auto')
#clf.fit(train_features, np.squeeze(train_labels))
#acc = clf.score(test_features, np.squeeze(test_labels))
#print(acc)
```

In [ ]:

```
from sklearn.metrics import classification_report
svc = SVC(kernel='poly', gamma='auto')
svc.fit(train_features, train_labels)

Y_preds = svc.predict(test_features)

print(classification_report(test_labels, Y_preds))
```

```
              precision    recall  f1-score   support

           0       0.77      0.71      0.74        98
           1       0.81      0.78      0.80       100
           2       0.65      0.68      0.66       100
           3       0.94      0.92      0.93       100
           4       0.91      0.96      0.94       100
           5       0.57      0.85      0.69       100
           6       0.66      0.54      0.59       100
           7       0.91      0.94      0.93        88
           8       0.85      0.91      0.88        99
           9       0.82      0.75      0.78       100
          10       0.86      0.95      0.90       100
          11       0.88      0.65      0.75        97
          12       0.94      0.96      0.95       100
          13       0.97      0.97      0.97       100
          14       0.95      0.94      0.94       100
          15       0.97      0.96      0.96       100
```

16	0.88	0.98	0.92	100
17	0.98	1.00	0.99	100
18	0.98	0.96	0.97	99
19	0.99	0.96	0.97	100
20	0.84	0.78	0.81	100
21	0.99	0.91	0.95	100
22	0.98	0.98	0.98	100
23	0.93	0.88	0.90	100
24	0.94	0.98	0.96	100
25	0.99	0.97	0.98	100
26	0.68	0.62	0.65	100
27	0.79	0.76	0.78	100
28	0.98	0.96	0.97	100
29	0.98	0.96	0.97	100
30	0.95	0.98	0.97	99
31	0.91	1.00	0.95	100
32	0.69	0.90	0.78	100
33	0.93	0.79	0.85	100
34	0.70	0.66	0.68	89
35	0.98	0.91	0.94	100
36	1.00	0.94	0.97	100
accuracy			0.88	3669
macro avg	0.88	0.87	0.87	3669
weighted avg	0.88	0.88	0.87	3669

In [ ]:

```
acc = svc.score(test_features,test_labels)
print(acc)
```

```
0.8751703461433633
```