Animal Image Classifier using Transfer Learning

ME781 Course Project
Autumn ’22

Shubham Kumar	200100148
Tathagat Fulzele	22M1370
Moin Ali Syed	22M1352
Shubham Sen	190040115
Akash Kulthe	190100008

Problem Objective and Definition

- Our project is an animal image classifier which we have built to classify cats and dogs images but it can be used as a prototype to build multi class classifier model.

- The objective is to target B2B market segment in which we will be selling our product to online pet adoption, buying and selling websites.

- For example:-
  There is a website only4pets connecting buyers and sellers for pets adoption, buying and selling. There are instances when the sellers don't mention what he/she is selling and randomly uploads pics of his\her pet. In that case it can be used to classify the images uploaded by the seller and then show the results accordingly for smooth functioning of the website.

- Our product will automatically categorise pet images and will make the website user interface friendly and convenient for users.

Technology Landscape

- We have employed MobileNetV2 model trained on ImageNet as our pre-trained model as it is a light weight model for classification using transfer learning.
- The official paper of MobileNetV2 can be found at the following URL : https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf
- MobileNetV2 trained model on ImageNet can be found at the following URL: https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4
- We obtained our dataset from Kaggle which can be found at the following URL: https://www.kaggle.com/competitions/dogs-vs-cats/data
- The libraries that we used are Kaggle, TensorFlow, Numpy, PIL, Matplotlib, Scikit-Learn, OpenCV
- Other modules and packages that we used are ZipFile, os, glob.

Pre-Processing and Model Architecture

- Google Colab URL of our project - https://colab.research.google.com/drive/11Leg9k35H7qaN0-SafmkyoESdNo7HJWG?usp=share_link
- Binary classification task with cat being 0 and dog being 1.
- The dataset is image data with 25000 images which is resized to 224x224 pixels and scaled.
- All the resized images were converted to numpy arrays.
- Dataset was split into train and test with a test size of 0.2.
- MobileNet V2 model was used as pre-trained model for transfer learning.
- We used accuracy as evaluation metric and Adam Optimizer for optimization.
- Loss function used was Sparse Categorical Cross Entropy.
- Obtained a training accuracy of 99.37% and test accuracy of 97.75% .
- Our model is memory and computation efficient as it built on a pre-trained model.

Model Building, Training and Outcome

```
[ ] mobilenet_model = 'https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4'

    pretrained_model = hub.KerasLayer(mobilenet_model, input_shape=(224,224,3), trainable=False)
```

```
[ ] num_of_classes = 2

    model = tf.keras.Sequential([

        pretrained_model,
        tf.keras.layers.Dense(num_of_classes)

    ])

    model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 1280)              2257984

 dense (Dense)               (None, 2)                 2562

=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

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

```
[ ] model.fit(X_train_scaled, Y_train, epochs=5)
```

```
Epoch 1/5
50/50 [==============================] - 47s 861ms/step - loss: 0.2163 - acc: 0.9162
Epoch 2/5
50/50 [==============================] - 42s 838ms/step - loss: 0.0746 - acc: 0.9756
Epoch 3/5
50/50 [==============================] - 44s 872ms/step - loss: 0.0552 - acc: 0.9825
Epoch 4/5
50/50 [==============================] - 41s 824ms/step - loss: 0.0417 - acc: 0.9894
Epoch 5/5
50/50 [==============================] - 41s 825ms/step - loss: 0.0345 - acc: 0.9937
<keras.callbacks.History at 0x7faedc598090>
```

```
[ ] score, acc = model.evaluate(X_test_scaled, Y_test)
    print('Test Loss =', score)
    print('Test Accuracy =', acc)
```

```
13/13 [==============================] - 12s 866ms/step - loss: 0.0812 - acc: 0.9775
Test Loss = 0.0812455490231514
Test Accuracy = 0.9775000214576721
```

Project Outcome