Machine Learning Engineer Nanodegree

Capstone Project Report

Marie Karam
June 1st, 2018

Identify Monkey

1. Definition

Project Overview

Monkey classification is to classify major types of monkeys based on biometric cues. Usually facial images are used to extract features and then a classifier is applied to the extracted features to learn a type of monkey recognizer. That's in Computer Vision and Biometrics fields. The monkey classification result is a type of monkey. I used two pre-trained model to compare between them and to find the best accuracy.
My data-set from kaggle: https://www.kaggle.com/slothkong/10-monkey-species

Problem Statement

The goal is to create a monkey classification based on monkeys' image and find the best features and accuracy to be more accurate when recognize monkey type. to help students or researchers research on zoology interested in monkey. Can use this project in educational applications in school.
- First download dataset then load train and test datasets
- Second use difference pre-trained CNN model to extract features from the test images and to compare between them.
- Third cheack which pre-trained model is the best and make predictions based on those test image features.

Metrics

I used accuracy metric on test set to evaluate the classifier (InceptionV3) and (VGG-16) that a good metric to be sure the data is balanced.
This the formula used in keras for categorical accuracy:
K.mean(K.equal(K.argmax(y_true, axis=-1), K.argmax(y_pred, axis=-1)))

II. Analysis

Data Exploration

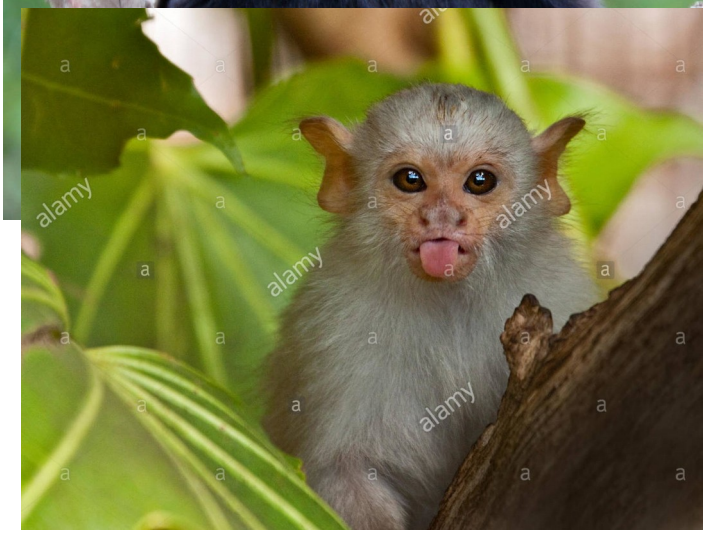
- The dataset provided as two files:
    * Train Images:

1097 images every image 400x300 px or larger and JPEG format (almost 1400 images).

* Validation Images:

272 images every image 400x300 px or larger and JPEG format (almost 1400 images).

- Each folder contains 10 subforders labeled as n0~n9, each corresponding a species form Wikipedia's monkey cladogram.
- Images are 400x300 px (almost 1400 images). Images were downloaded with help of the googliser open source code.



Exploratory Visualization

Label mapping (Latin Name):

n0, alouatta_palliata
n1, erythrocebus_patas
n2, cacajao_calvus
n3, macaca_fuscata
n4, cebuella_pygmea
n5, cebus_capucinus
n6, mico_argentatus
n7, saimiri_sciureus
n8, aotus_nigriceps
n9, trachypithecus_johnii

Monkey Type :

n0 , mantled_howler
n1 , patas_monkey
n2 , bald_uakari
n3 , japanese_macaque
n4 , pygmy_marmoset
n5 , white_headed_capuchin
n6 , silvery_marmoset
n7 , common_squirrel_monkey
n8 , black_headed_night_monkey
n9 , nilgiri_langur
- For more information on the monkey species and number of images per class make sure to check monkey_labels.txt file.

Algorithms and Techniques

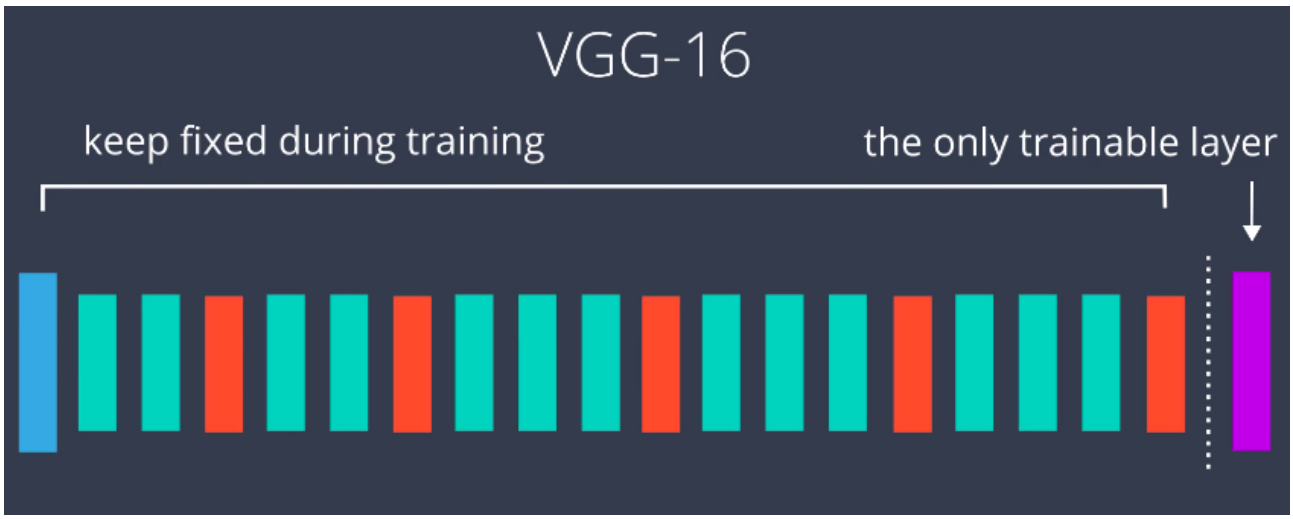
I am tried to compare between two techniques to get the best technique extract the features and get high accuracy.

First: I used a pre-trained CNN model for image classification(InceptionV3)

I used inception_v3 model pretrained on the ImageNet data set. ImageNet is a large dataset used for image classification. The goal of the inception module is to act as a “multi-level feature extractor” by computing 1x1, 3x3, and 5x5 convolutions inside identical module of the network. The output of those filters are then stacked on the channel dimension and before being fed into following layer within the network. Then I build a neural network that give me the image extracted of features from InceptionV3 can classify the monkey in the image which type

Second: I used a pre-trained CNN model for image classification(VGG-16):

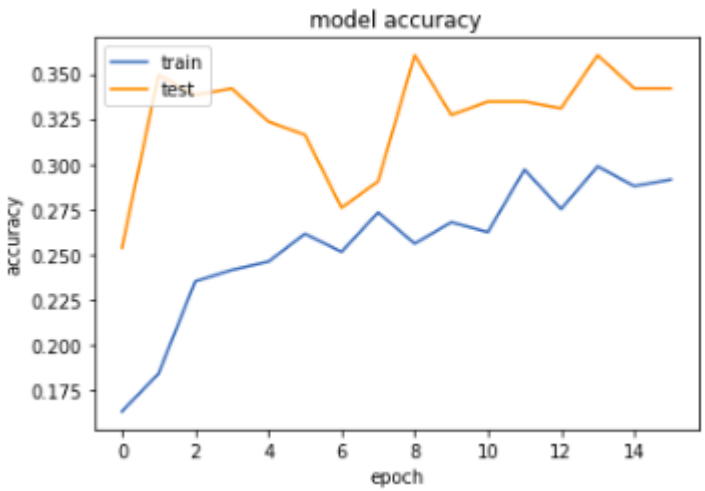
I used VGG-16 model pretrained on the ImageNet data set. The monkey data set is comparatively little and it’s important overlap with a set of the ImageNet classes. I slice off the top of the network and add a replacement classification layer with 133 nodes. Then train only the weights in this layer, freezing all the weights in the other layers.
This model as a fixed feature extractor, where the last convolutional output of VGG-16 is fed as input to our model. We tend to add a worldwide average pooling layer and a completely connected layer, wherever the latter contains one node for every monkey class and is supplied with a softmax.Used pre-traind model better than use extract features from images instead of training a new CNN that will need more of time and data to be trained well.

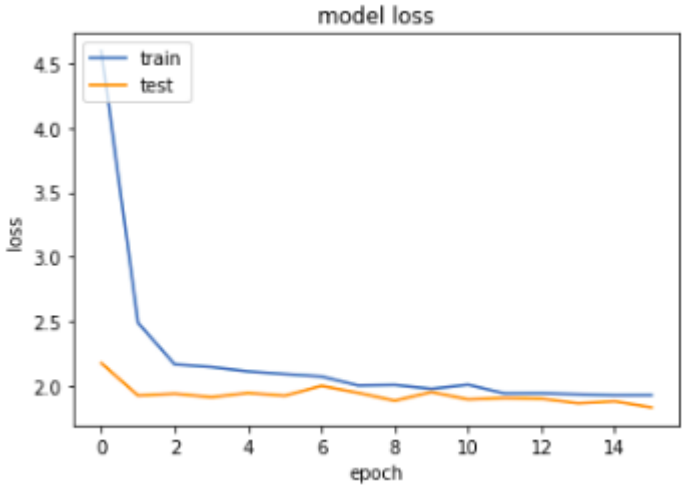


Benchmark Model

I used a network with one convolution layer with 128 filters then a global averaging layer then a dense layer with 1024 neurons and a dropout of 0.7 then a softmax layer with 10 neurons to make the classification.
Applying this network on the images with preprocessing done by keras built-in function I managed to get 29% accuracy on the training data and 34% accuracy on the validation data.
So this seems to be very low accuracy considering this only a 10 class problem, so on this project I will try to get better accuracy using different methods.

The results of benchmark





III. Methodology

Data Preprocessing

I find the data clear and not need more processes to be clear. And I see the distribution of label and I find it clear and distribution very well. And I used function from keras (preprocess_input) its designed for using in pre-trained model vgg16 and inception-v3

Implementation

After I found the data clear and not need more processes. So I will try two pre-trained model trained on imagenet data-set for image classification. I will use it to extracted the features. After that I will use the extracted features to train dense layer to classify the type and try add dropout layer to avert over-fit.

My work steps :-

- creating a benchmark model for testing my wok against
- choosing model for extracted features from keras then load the model from keras
- remove the predicted layer
- load data-set
- build a dense and dropout layer
- train the layers of a dense and dropout to predict type of monkey.
- use another pre-trained model
- try use different numbers of a dense layer, dropout layer and neurons to find the best numbers of a dense layer, dropout layer and neurons get high validation accuracy
- test the best numbers of a dense layer, dropout layer and neurons find it on test data.
- by using the model make predict for monkeys.

Our benchmark model only achieved 34% accuracy, so I needed to make a big improvement, and since this an images related problem, it’s almost always good to use a pretrianed model to extract features from the images, instead of training a new one.

So I tried using different pretrianed models and choosing the best one, that was the hardest challenge I faced on this problem.

Then I tried different number of networks with different layers so I tried a network of a dense layer of 128, 512 ,1024 neurons and for dropout layer rate ranging from 0.4 to 0.7 and the best numbers for dense layer 1024 and for dropout layer 0.7 in inception-v3 and VGG-16

Refinement

In this project I worked on

- The First pre-traind model i used VGG16 to get features extraction and used a Dense layer = 1024 neurons, the model good to over-fitted the trainind data very fast. Used Acctivation = ‘relu’ to increase channels. And add a dropout layer with 0.4 ...0.7 and I tried different rate but the best rate I find fully connected layer dropout = 0.7 , 16 epoch , 1024 neurons and final layer 10 neurons and softmax to get output the validation accuracy = 0.9890

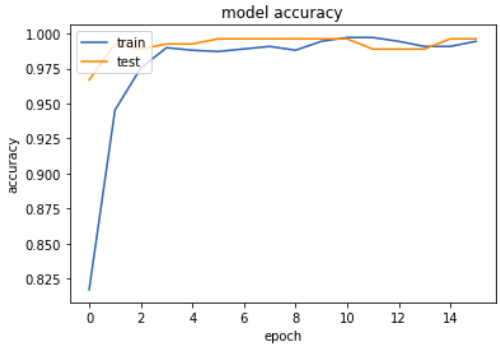
-The Second pre-traind model i used inception-v3 to get features extraction and used a Dense layer = 1024 neurons, the model good to over-fitted the trainind data very fast. Used Acctivation = ‘relu’ to increase channels. And add a dropout layer with 0.4 ...0.7 and I tried different rate but the best rate I find fully connected layer dropout = 0.7 , 16 epoch , 1024 neurons and final layer 10 neurons and softmax to get output the validation accuracy = 0.9963

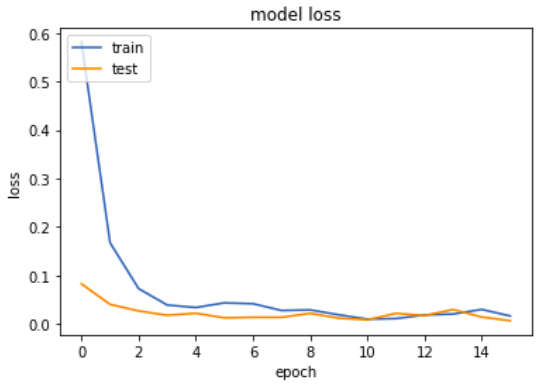
IV. Result

Model Evaluation and Validation

During development a validation set was choosing of .. images to make sure that the model won’t over-fit to the training data and that the model can generalize to new unseen images.

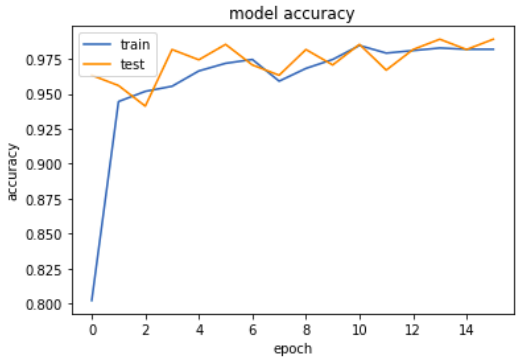
First : I used plot training history to evaluate the classifier(InceptionV3**)**

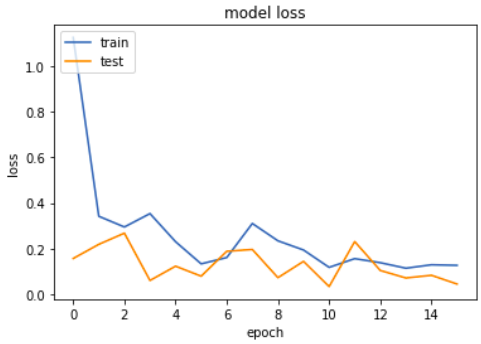




As we can see here with only 16 epochs I managed to get 99.4% accuracy on the training set and 99.6% on the validation set.

Second : I used plot training history to evaluate the classifier(VGG16**)**





Also here using the VGG16 model I managed to get 98.2% accuracy on the training set and 98.9% on the validation set.

here’s a complete description of the approach that got best result:
- using Inception V3 to extract features from the images
- building a network with 1024 neurons and a dropout rate of 0.7 and finally a dense layer of 10 neurons to make the classifcations.
To verify the robustness of the model I tested the final model on images from the validation set and it predicted all of them correct

Justification

Using only a small data-set consists of 1370 image. i managed to get 99% accuracy with already pretrained model which is good accuracy for this problem. And also our benchmark model only got 34% accuracy on the validation set, so we made a lot of improvement from the benchmark model.

V. Conclusion

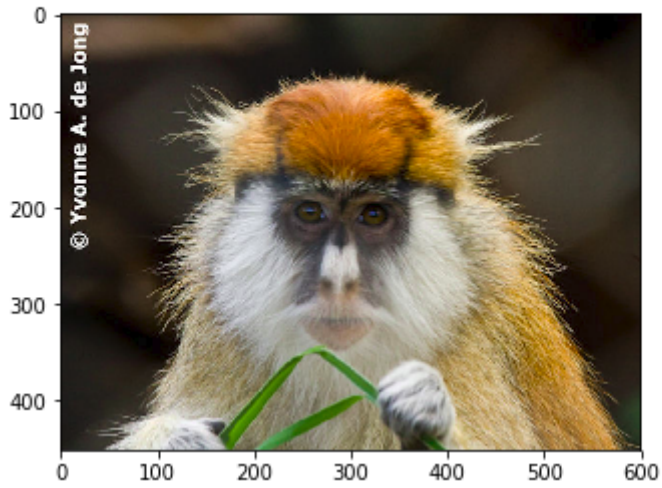
Free-Form Visualization

To show the results this picture of monkeys the model can identify it and its type.

This is the output of pre-trained model VGG-16

```
: predict_class('training/n1/n1018.jpg')
```

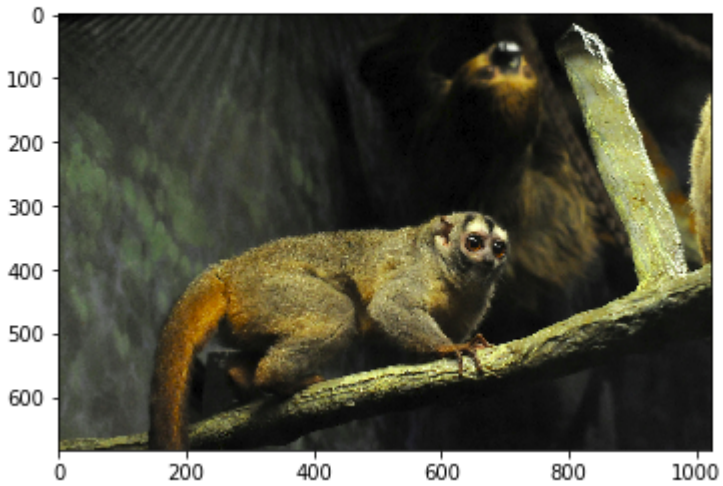
```
1
```



And this is the output of pre-trained model inception-v3

```
: predict_class('validation/n8/n807.jpg')
```

```
8
```



Reflection

Through working on this project :
1- choosing dataset
2- exploaring dataset
3- choosing pret-trained model
4- building a network for classifying extracted features
the best model inception-v3 to get features extraction and used a Dense layer = 1024 neurons, the model good to over-fitted the trainind data very fast. Used Acctivation = ‘relu’ to increase channels. And add a dropout layer with 0.4 ...0.7 and I tried different rate but the best rate I find fully connected layer dropout = 0.7 , 16 epoch , 1024 neurons and final layer 10 neurons and softmax to get output the validation accuracy = 0.9963 .
The difficult stage on this project:
- find a good and clear dataset to work on. its take a lot of time to find it
Our benchmark model only achieved 34% accuracy, so I needed to make a big improvement, and since this an images related problem, it’s almost always good to use a pretrianed model to extract features from the images, instead of training a new one.
So I tried using different pretrianed models and choosing the best one, that was the hardest challenge I faced on this problem.
Then I tried different number of networks with different layers so I tried a network of a dense layer of 128, 512 ,1024 neurons and for dropout layer rate ranging from 0.4 to 0.7 and the best numbers for dense layer 1024 and for dropout layer 0.7 in inception-v3 and VGG-16

Improvement

1- using a different pre-traind model, here I only used not very complex pre-trained model like VGG16 and Inception V3 because I don’t have much of a computational power to work with more complex network like VGG19 or inception resnet which can improve the result
2- using a bigger network for classifying on the extracted features as here I only used one layer because limited computational power.
3- training the model for more epochs, as here I only trained the model for 16 epochs, and also try more dropout rate and see which one works better.
4- re-train the pre-trained model layers or some layers on the new data.
5- try another data-set.
6- use more preprocessing on the dataset, like image masking or image sharping that can make recognizing the features from the images faster and more accuracte.

References:

* https://keras.io/utils/#to_categorical
*https://datascience.stackexchange.com/questions/14415/how-does-keras-calculate-accuracy/14742#14742
* https://keras.io/applications/#inceptionv3
* http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py
* https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
* https://github.com/chasingbob/keras-visuals/blob/master/visual_callbacks.py
* https://keras.io/preprocessing/image/
* https://www.kaggle.com/paultimothymooney/predicting-pathologies-in-x-ray-images?scriptVersionId=1951852
* https://www.kaggle.com/kmader/detect-retina-damage-from-oct-images-hr/versions
* https://www.kaggle.com/paultimothymooney/identify-monkey-species-from-image
* https://www.kaggle.com/slothkong/10-monkey-species
* https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/
* https://github.com/udacity/dog-project/blob/master/dog_app.ipynb
* https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/
* http://junyelee.blogspot.com/2018/01/deep-learning-with-python.html