MACHINE LEARNING ENGINEER NANODEGREE – CAPSTONE PROJECT

SANGAME KRISHNAMANI
FEB 10, 2018

DEFINITION

PROJECT OVERVIEW

The problem domain for this exercise is computer vision and deep learning. Image classification is one of the reasons why I got interested in machine learning course. Deep learning Dog Breed recognition project made me interested in pursuing further in improving the algorithm. But instead of staying in the realms of the same idea and improving on top of it, I have made it more challenging. The idea is to recognize missing pets and by security camera footage/images posted by people online of homeless pets.

The American Humane Association estimates

- over 10 million dogs and cats are lost or stolen in the U.S. every year. One in three pets will become lost at some point during their life.
- One in three pets will become lost at some point during their life
- Only 22 percent of lost dogs and 2 percent of lost cats that entered the animal shelters were reunited with their families

To make this more focused and achievable the aim of this project is to build a convolutional neural network that classifies whether image contains dogs vs cats.

It is easy for humans to recognize a dog vs cat, but is a challenge for computers because of the variety of breeds, large diversity of photo noise (wide variety of backgrounds, angles, poses, lighting, etc). This makes accurate classification quite difficult to achieve.

In an informal poll conducted many years ago, computer vision experts posited that a classifier with better than 60% accuracy would be difficult without a major advance in the state of the art. For reference, a 60% classifier improves the guessing probability of a 12-image HIP from 1/4096 to 1/459.

Advent in Machine learning has made this problem solvable with decent accuracy and what excites me the most is the fact that ML has opened the door to unlimited possibilities.

This base idea can be expanded to other species and even humans for face recognition, very similar to how Facebook provides suggestions for tagging a face.

Data Source: https://www.kaggle.com/c/dogs-vs-cats/data

Academic Paper:

https://www.robots.ox.ac.uk/~vgg/publications/2012/parkhi12a/parkhi12a.pdf

https://www.mathworks.com/company/newsletters/articles/deep-learning-for-computer-vision-with-matlab.html

PROBLEM STATEMENT

The goal is to build a face recognition system for Dogs and Cats and classify them appropriately.

The dataset images were manually classified and labeled by Asirra. We have a dataset of 25000 images for cats and dogs with labels to accomplish this. Once the computer is trained, we have a test dataset of 12,500 with no labels (cat or dog). Assumption would be that image would contain either a dog or cat but not both. The images might also contain other objects.

The images are not all of the same size and there are other objects that act as noise on the image. This is a supervised learning problem because each image acts as an input and a desired output value (dog or cat) is expected, i.e. a binary classification. My approach here is to use a convolutional neural network (CNN).

The steps used in the project:

1. Get data from Kaggle
2. Preprocess the data
   a) Divide images to training and validation set
   b) Resize images in training folder
3. Define the architecture of the CNN from scratch
4. Train and Optimize for accuracy.
5. Use the classifier to recognize dogs vs cats.

METRICS

For this project we have two datasets:

- Train dataset of 25,000 images with labels
- Test dataset of 12,500 images. **No labels provided**.

I am dividing the train dataset into 3 different sets –

1. Training dataset of 10400 images that will be used to train the model,
2. Validation dataset of 2600 images for testing the performance of the CNN
3. A test set of 5000 images that will be used to do a final evaluation before submission.

The goal of creating this test set is that as the CNN is repeatedly trained and tested on validation set, it will eventually "see" the validation set. This test set will be a more objective evaluation of the accuracy of CNN on an out of sample. I will be using two metrics for measuring the success of the project:

Either Log Loss or Accuracy Metrics:

Log Loss (categorical cross entropy) quantifies the accuracy of a classifier by penalizing false classifications. Minimizing the Log Loss is basically equivalent to maximizing the accuracy of the classifier, but there is a subtle twist which we'll get to in a moment.

In order to calculate Log Loss the classifier must assign a probability to each class rather than simply yielding the most likely class. Mathematically Log Loss is defined as

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}p_{ij}$$

where N is the number of samples or instances, M is the number of possible labels, y_{ij} is a binary indicator of whether or not label j is the correct classification for instance i, and p_{ij} is the model probability of assigning label j to instance i. A perfect classifier would have a Log Loss of precisely zero.

Accuracy is the ratio of correct predictions to total number of events. In this case it would be the percentage labels correctly classified.

$$\frac{\mathrm{TN}+\mathrm{TP}}{\mathrm{TN}+\mathrm{FP}+\mathrm{FN}+\mathrm{TP}}$$

Where, TN – number of True Negative cases, FP – False Positives, FN – False Negatives, TP – True Positives.

ANALYSIS

DATASETS EXPLORATION

A subset of **Asirra dataset** will be used for this project. CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) or HIP (Human Interactive Proof) are challenges that are easy for people to solve but not for computers. HIPs are used for many purposes, such as to reduce email and blog spam and prevent brute-force attacks on web site passwords.

Asirra (Animal Species Image Recognition for Restricting Access) is a HIP that works by asking users to identify photographs of cats and dogs. This task is difficult for computers, but studies have shown that people can accomplish it quickly and accurately. Many even think it's fun! Here is an example of the Asirra interface:

Asirra is unique because of its partnership with Petfinder.com, the world's largest site devoted to finding homes for homeless pets. They've provided Microsoft Research with over three million images of cats and dogs, manually classified by people at thousands of animal shelters across the United States.

Each image will have either a dog or cat in the image and can contain other objects and human. The train folder contains 25,000 images of dogs and cats. Each image in this folder has the label as part of the filename. The test folder contains 12,500 images, named according to a numeric id.

Few challenging images are shown here:

Some are blurry, other have more than one cat/dog, some have humans and other objects, the image sizes are different and the animals are facing in different angles.

Image format: .jpg file

Number of Images:

Training

Cat images: 12500
Dog images: 12500

Each image in this folder has the label as part of the filename

Testing

Cat/Dog images: 12500

EXPLORATORY VISUALIZATION

Size and description of images

All images are different in sizes.

- The images have varying levels of lightening
- there are other objects and humans in the dataset along with cats or dogs
- The face of the animal is angled at different positions and is not always front facing
- The animal is not always present in the center of the frame.

Data Source: https://www.kaggle.com/c/dogs-vs-cats/data

ALGORITHMS AND TECHNIQUES

Deep learning techniques are most effective on image classification and the same approach would be used to solve this problem. Data augmentation and transfer learning will be used to train a Convolutional Neural Network (CNN) to classify the images as dogs vs cats.

Option 1:

The CNN architecture is known for best image classification. As mentioned in the workflow section below, a CNN will be trained from scratch. Using an architecture similar to the one shown below, I would attempt to train a CNN from scratch. The architecture has 3 convolution layers with a ReLu activation function, followed by max-pooling layers. Before the fully-connected layer, because of the large amount of data a dropout layer can considered to reduce overfitting.

OR/AND

Option 2:

Transfer learning or inductive transfer is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. Many such networks pre-trained on imagenet challenge are available today – RESNET, Inception-V3, VGG-16, VGG-19, etc.

The CNN will be trained using transfer layer, one of the pre-trained network will be used for the same.

Comparison (optional)

If the previous option of training the CNN from scratch is attempted Option 1 can be compared with 2.

BENCHMARK

"The security of ASIRRA is based on the presumed difficulty of classifying images of cats and dogs automatically. As reported in [6], evidence from the 2006 PASCAL Visual Object Classes Challenge suggests that cats and dogs are particularly difficult to tell apart algorithmically. A classifier based on color features, described in [6], is only 56.9% accurate."
Ref: https://eprint.iacr.org/2008/126.pdf

The benchmark implementation with no optimization yielded an accuracy around 50% and validation loss around 81%.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
flatten_3 (Flatten)          (None, 197136)            0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 394274
=================================================================
Total params: 394,482
Trainable params: 394,482
Non-trainable params: 0
_________________________________________________________________
Train on 10000 samples, validate on 2500 samples
Epoch 1/4
 9980/10000 [============================>.] - ETA: 0s - loss: 8.0426 - acc: 0.5003Epoch 00001: val_loss improved from inf to 8.12352, saving
model to saved_models/weights.best.benchmark_from_scratch.hdf5
10000/10000 [==============================] - 116s 12ms/step - loss: 8.0443 - acc: 0.5002 - val_loss: 8.1235 - val_acc: 0.4960
Epoch 2/4
 9980/10000 [============================>.] - ETA: 0s - loss: 8.0526 - acc: 0.5004Epoch 00002: val_loss did not improve
10000/10000 [==============================] - 109s 11ms/step - loss: 8.0542 - acc: 0.5003 - val_loss: 8.1235 - val_acc: 0.4960
Epoch 3/4
 9980/10000 [============================>.] - ETA: 0s - loss: 8.0542 - acc: 0.5003Epoch 00003: val_loss did not improve
10000/10000 [==============================] - 102s 10ms/step - loss: 8.0542 - acc: 0.5003 - val_loss: 8.1235 - val_acc: 0.4960
Epoch 4/4
 9980/10000 [============================>.] - ETA: 0s - loss: 8.0526 - acc: 0.5004Epoch 00004: val_loss did not improve
10000/10000 [==============================] - 103s 10ms/step - loss: 8.0542 - acc: 0.5003 - val_loss: 8.1235 - val_acc: 0.4960
```

METHODOLOGY

DATA PREPROCESSING

IMPLEMENTATION

Libraries: Scikit-learn, Keras, Tensorflow, NumPy, Pandas, matplotlib, OpenCv

Workflow:

Potentially these would be the steps I would follow -

- Resize the images, so they are of the same size
- Train a CNN from scratch for comparison with transfer learning models (optional)

- Fine tuning the pre-trained network by choosing different optimizers.

Results for CNN built from scratch, the algorithm used is shown below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 26, 26, 64)        16448
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 13, 13, 4)         2308
_________________________________________________________________
flatten_4 (Flatten)          (None, 676)               0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 1354
=================================================================
Total params: 30,654
Trainable params: 30,654
Non-trainable params: 0
_________________________________________________________________
```

REFINEMENT

Learnings from refinement:

- Image normalization and adding color channels did not have as much positive effect as I thought
- The depth of neural network affects accuracy. As the network gets deeper the ability to learn features improves
- The optimization function had the most impact on improving the accuracy

RESULTS

MODEL EVALUATION AND VALIDATION

I tried all the techniques I possibly could to refine the model. The final output seems reasonable in terms of parameters and layer structure. The incremental benefit of changes to the model was very small after a point. So theoretically refinement can continue but in my opinion the effect would be small and not worth the time

Log loss is lower than the benchmark established in the beginning and the model accuracy score of 88.46% on validation set is reasonable

The Log loss on the test dataset at Kaggle was **0.39.** This is in the ballpark we had with the final model on our validation and test sample. As expected performance is slightly weaker on the sample never seen before however the model seems robust.

```
Train on 10000 samples, validate on 2500 samples
Epoch 1/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.4603 - acc: 0.7826Epoch 00001: val_loss improved from inf to 0.48968, saving
model to saved_models/weights.best.from_new_scratch.hdf5
10000/10000 [==============================] - 243s 24ms/step - loss: 0.4605 - acc: 0.7824 - val_loss: 0.4897 - val_acc: 0.7556
Epoch 2/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.4261 - acc: 0.8064Epoch 00002: val_loss improved from 0.48968 to 0.44223, sa
ving model to saved_models/weights.best.from_new_scratch.hdf5
10000/10000 [==============================] - 244s 24ms/step - loss: 0.4261 - acc: 0.8064 - val_loss: 0.4422 - val_acc: 0.8008
Epoch 3/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.4009 - acc: 0.8174Epoch 00003: val_loss did not improve
10000/10000 [==============================] - 244s 24ms/step - loss: 0.4006 - acc: 0.8176 - val_loss: 0.4513 - val_acc: 0.7992
Epoch 4/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.3739 - acc: 0.8319Epoch 00004: val_loss improved from 0.44223 to 0.40052, sa
ving model to saved_models/weights.best.from_new_scratch.hdf5
10000/10000 [==============================] - 231s 23ms/step - loss: 0.3745 - acc: 0.8315 - val_loss: 0.4005 - val_acc: 0.8224
Epoch 5/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.3525 - acc: 0.8427Epoch 00005: val_loss did not improve
10000/10000 [==============================] - 234s 23ms/step - loss: 0.3521 - acc: 0.8428 - val_loss: 0.4298 - val_acc: 0.7964
Epoch 6/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.3278 - acc: 0.8551Epoch 00006: val_loss did not improve
10000/10000 [==============================] - 236s 24ms/step - loss: 0.3284 - acc: 0.8548 - val_loss: 0.4384 - val_acc: 0.7896
Epoch 7/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.3018 - acc: 0.8716Epoch 00007: val_loss improved from 0.40052 to 0.39073, sa
ving model to saved_models/weights.best.from_new_scratch.hdf5
10000/10000 [==============================] - 219s 22ms/step - loss: 0.3019 - acc: 0.8715 - val_loss: 0.3907 - val_acc: 0.8164
Epoch 8/8
 9980/10000 [============================>.] - ETA: 0s - loss: 0.2803 - acc: 0.8845Epoch 00008: val_loss did not improve
10000/10000 [==============================] - 228s 23ms/step - loss: 0.2800 - acc: 0.8846 - val_loss: 0.3964 - val_acc: 0.8292
```

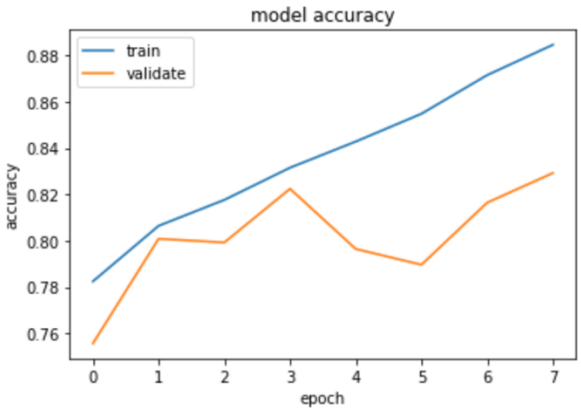
The model has an 88.46% accuracy on the validation set and 92 % on the test set

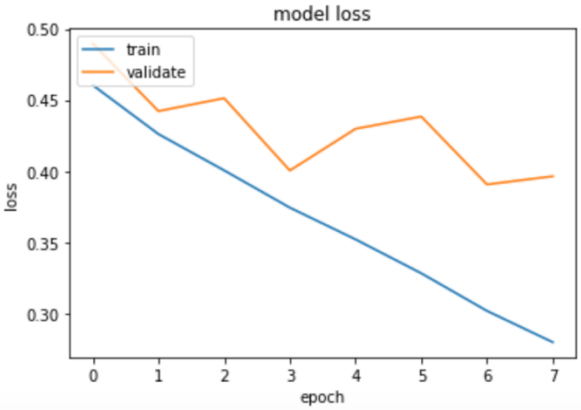
JUSTIFICATION

The final model for this problem is a deep convolution neural network with 5 convolution layers, 4 MaxPool layers, 1 dense layer and 1 flatten layers on colored images to predict probability of cat vs dog. After trying multiple combinations of parameters and optimization function to tune the model to its final setting there is minute incremental benefit of continuing this optimization after this point. The model has an accuracy of 88.46% on validation set in its ability to correctly predict cat vs dog. This is still considerably low when compared with how well humans can categorize the images. I consider this as a representation of limitation on machine learning on images and also the fact that there is more factors to be considered for making this prediction even better.

CONCLUSION

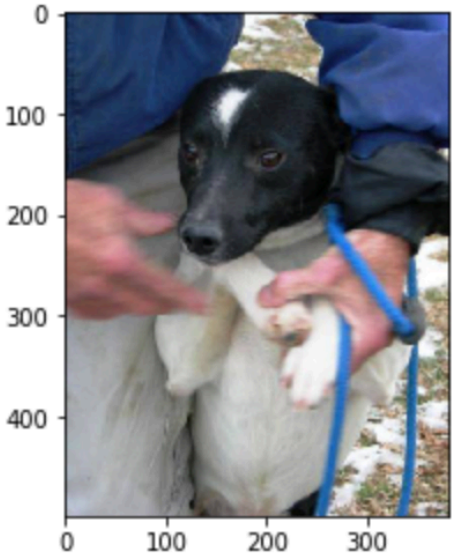
FREE-FORM VISUALIZATION



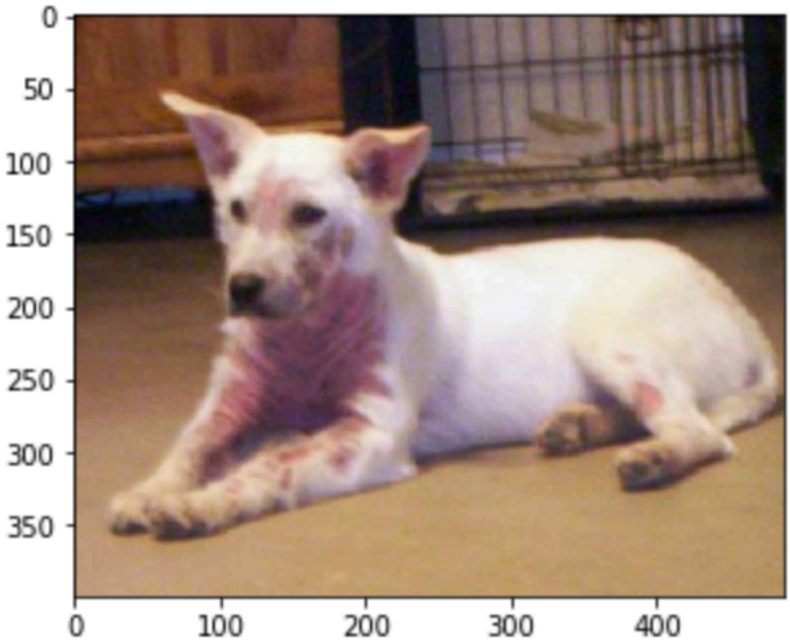


There were 2 images in the test set where predictions were incorrectly made. Looking at the images the ambiguity can be assessed, the dog looks unique or the position in which the dog image was captures leads to this ambiguity. We still got a accuracy of 92% on test images.

```
This image contains cat
```




```
--------------------------------------------------
This image contains cat
```



REFLECTION

After working on these projects these are my learnings and thoughts:

1. I have a better understanding of deep learning and CNN. Along with increased confidence in applying these algorithms to use by using python. I could get to this point by learning these concepts and techniques one at a time and putting them all together.

2. Dataset and image augmentation – Understanding the problem started with looking into the dataset/images and analyzing what type of preprocessing and normalization has to be done on the images before feeding it to a CNN (resizing, flattening, color contrasts, etc were considered).

3. Build a base model for benchmark and Optimization and Refinement – This step helped with getting a better understanding of the architecture and what each step/process meant. It was challenging and at the same time exciting to perform optimization to achieve higher accuracy with results. I also explored using EC2 instance to perform multiple iterations, as the speed of processing was slow on my laptop.

4. Documentation – Step by step documentation and reflection on everything that went into putting this project together (in a structured way), helped with giving me confidence and gaining a deeper understanding.

IMPROVEMENT

On comparing my results with the leaderboard in Kaggle, where I see log loss rate is 0.047, which is way better than what I have achieved here.

- Overfitting – Use techniques to reduce overfitting and thus providing more accurate predictions.
- Preprocessing of image – I still think I could do better with preprocessing of images. Where I could clear out the noise (other objects in the image) and improve the accuracy.

REFERENCES

https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
https://www.kaggle.com/c/dogs-vs-cats
https://www.petfinder.com/dogs/lost-and-found-dogs/why-microchip/
https://en.wikipedia.org/wiki/Transfer_learning
https://eprint.iacr.org/2008/126.pdf