Project Proposal

26.07.2019

REVIEW LINK:
https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=fefcc71b-df5c-4766-822f-e7c22746e989&bsft_uid=48b2fb8a-3b2b-4845-b2ef-20c65f374720&bsft_mid=6a976d8b-368e-420f-9703-55eb5d662b16&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=5a3ef338-e5ec-4c71-81d9-78c3c90db4e4#!/reviews/1920677

Kolja Maier

Definition

Project Overview & Domain background

Neural Networks are inspired by biology - actual neurons did not only influence the name of 'neural networks' but also the theoretical foundations of deep learning in terms of neurons activation principles. Also advanced neural-network architectures, like CNNs (Convolutional Neural Networks), have resemblance to the human brain. Experiments even showed that activation patterns in the human cortex correlate with areas of a CNN when solving image classification tasks [https://arxiv.org/ftp/arxiv/papers/1608/1608.03425.pdf https://mindmodeling.org/cogsci2016/papers/0435/paper0435.pdf].

The roots of CNNs, as we know them today, are dated back to around 1990 when the famous scientist Yann LeCun did his pioneer work in that field. The foundations of Deep Learning are even older, reaching back to the 1950s. Hence the theory behind neural networks, deep learning and CNN is well understood.

But can we really be so certain about how CNNs learn?

In recent years criticism raised for using Neural Networks, because they rather behave as a black box in giving predictions for new data. More recently, several methods (for instance LIME https://homes.cs.washington.edu/~marcotcr/blog/lime/) have been developed to bring more transparency into the world of deep learning algorithms. For these reasons it is very important for every Deep Learning application to get a better understanding of their functionality and their usage.

In this project we will focus on some aspects in that domain:

a. how shape and texture of objects (like dogs) influence a CNNs training/learning and inference process
b. how & if it is possible to perform better from these insights (a.)

This project is based around the paper "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." [https://openreview.net/forum?id=Bygh9j09KX], which claims that there is a general bias in CNNs towards learning the texture of objects, instead of the shape.

Particularly I will focus on the domain of dog-pictures, since the fur of dogs seems to hold lots of texture information (long, short, straight, curly,...), which seems to be promising for this use case. This projects domain can also be seen in Data augmentation, since the manipulation of source images and style-transferring their textures leaves us with more input data that is augmented.

Problem statement

[https://openreview.net/forum?id=Bygh9j09KX] showed, that humans naturally recognize objects primarily by their shape and not their texture. [https://openreview.net/forum?id=Bygh9j09KX] also showed, that CNNs actually have a bias towards learning the texture representation instead of the shape representation of an object. This can be problematic since lots of information sits inside the actual shape of an object. Particularly for object recognition (with the special case of image classification) shape is an very important feature. In this project we focus on the problem of image classification with CNNs (of dog pictures and their races) with the focus on data augmentation regarding shape and texture information. The assumption is, that I can increase the accuracy of the CNN model by providing more information on the shape of an object instead of its texture.

Especially in the domain of dog pictures this looks promising, since dogs from different breeds can have a similar fur-texture, but a very distinguished shape (see Figure 1).

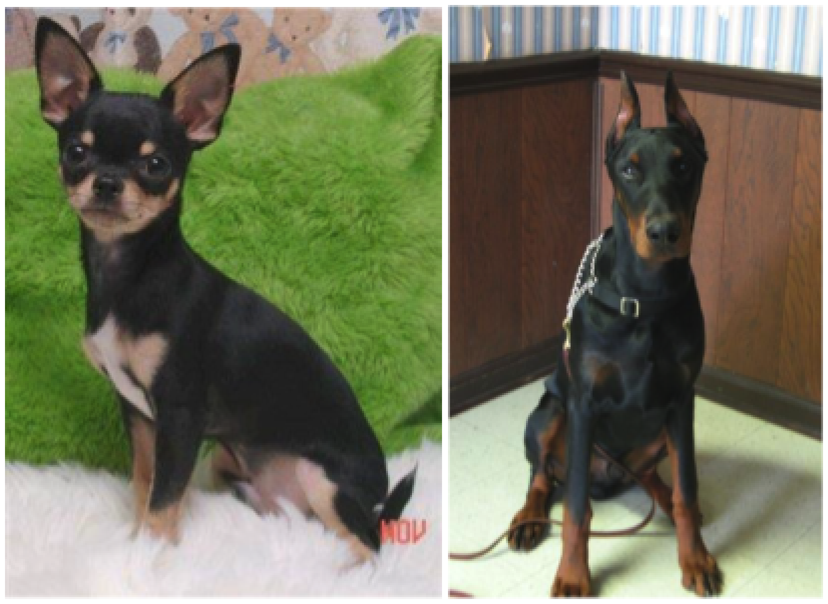

(a) (b)

Figure 1. Images of (a) a Chihuahua and (b) a Doberman. Although their fur has very similar texture (short, straight and slightly shiny hair), their body shape is very different

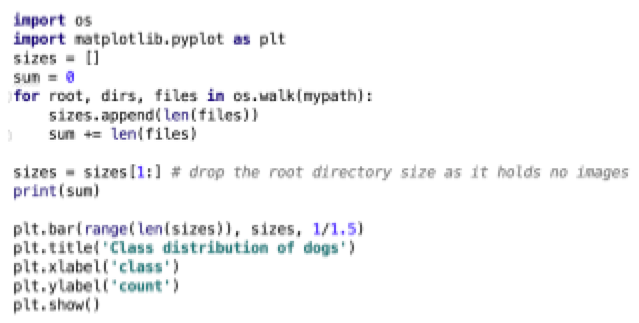
-

Hence the concrete problem to solve is to write a CNN image classification application (for dog breeds) that reduces the inherent texture bias of CNNs and instead focuses on learning shape representation with shape information.

Datasets and Inputs

The dataset I use will be the dogs images set, that we know from our Notebook exercise. Since it is quite large I will provide the link to it instead of a whole copy. This dataset holds various pictures of dogs and their corresponding race-labels - in total there are 133 different breeds of dogs. It holds image data that suits our problem definition from above.

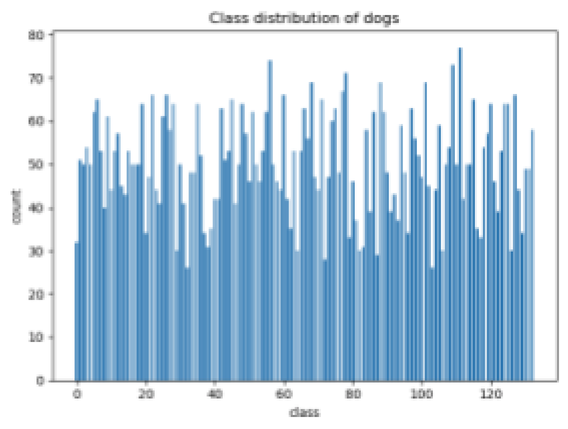
One very important aspect is the class balance in a dataset, which also influences the choice of our Metric (see below). Hence we take a quick look at the class distributions of our training data:



```
import os
import matplotlib.pyplot as plt
sizes = []
sum = 0
for root, dirs, files in os.walk(mypath):
    sizes.append(len(files))
    sum += len(files)

sizes = sizes[1:] # drop the root directory size as it holds no images
print(sum)

plt.bar(range(len(sizes)), sizes, 1/1.5)
plt.title('Class distribution of dogs')
plt.xlabel('class')
plt.ylabel('count')
plt.show()
```



We see that our class distribution is quite balanced. Some classes are more present than others, but in general the distribution is uniformly shaped.

In total there are 6681 pictures of dogs in the training set across the various breed classes. For the test and validation set there are 837 and 836 pictures respectively.

Furthermore, one crucial input component are the textures that will be used for style transfering the images. These are various hand picked textures. You can find the texture images in this submission under the folder **texture_images**.

Disclaimer: While these should cover most of the textures I will use in this project, it is possible that I will still add new ones to improve my implementation (since I did not write my implementation yet, it is hard to judge, if all of these texture pictures will perform good).

Further augmentations will be specified with filter-operations (for example edge filters and greyscale that the CNN is forced to focus on the shape while learning). But this is no actual data but some preprocessing step on the existing data.

Right now, I have already decided on 27 textures, which will augment the raw image data. Hence, after preprocessing with textures I will have an augmented data set of 27*6681= 180387 training examples.

Solution Statement

As described above, CNNs have a bias in learning the texture of objects instead of its shape (which is the primary 'feature' for humans to identify objects [https://openreview.net/forum?id=Bygh9j09KX]). The scope of this projects solution for this problem is to 'force' the CNN to focus on learning object shapes. This will be done in augmenting and manipulating the input images. To visualize this, have a look at Figure 2. A lot of different textures are applied to the same source image. This results in multiple input pictures for the same object. Since there are various textures for the same object, the CNN is forced to learn the shape of this object, since this is now the most distinct description feature. You can also see how the curly hair texture of the original source image is washed away by applying different textures.

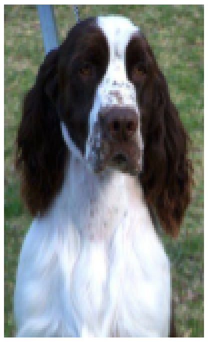
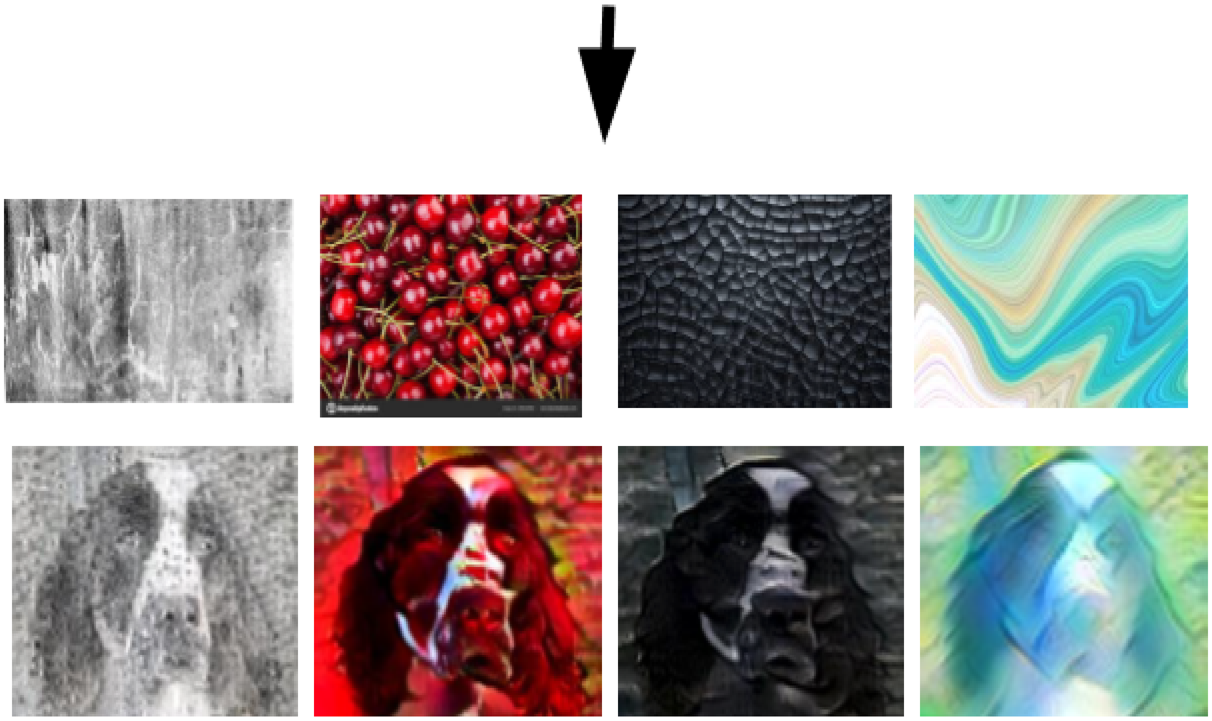


Figure 2. The input image from our dog data images data set will be style transferred by different textures (row one).
The result is the same dog with different texture styles applied (row two). Since now different textures are present, but the shape is still the same and recognizable, we enforce the CNN to learn the shape-feature



Since there is not only one texture for the same object, the CNN is forced to focus on the actual shape of the object. The hypothesis is to reduce the texture bias that usually occurs in training CNNs and afterwards get a more robust and performant (in terms of accuracy) CNN classifier.

This will be done by:

- Reduce CNN texture bias by custom preprocessing/augmentation (with style transfer)

- Transfer learning: A pre-trained model (probably trained on ImageNet) will be used since there are also many animal classes
    - The last layer(s) will be retrained with our custom data
- Measure the importance of the features
- Tuning the model (probably Hyperparameter Tuning with SageMaker)

Furthermore, since this is the capstone project for ML Engineering, I want to focus on proper methodologies & technologies that was taught in the course.

Benchmark model

The benchmark model will be (self implemented) the CNN we were instructed to implement during the dog-classification notebook in the courses material.

This model leverages some advanced techniques like transfer learning and hence is a quite ambitious baseline.

Metrics

Since we are facing a classification problem a valid approach for insights of the quality of a model is the accuracy metric which is defined as:

- $accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions}$

In fact this is also exactly the metric we want to pay attention to in terms of our problem & solution statement, which should answer the questions:

- Is there an inherent texture bias in our benchmark model solution?
- Will the augmentation and new feature-attention (on shape) give better results?

We want to pay particular attention for a proper engineering process that separates the data into

- train set
- test set
- validation set

With this kind of data separation it is guaranteed that the model will only learn the representation from training data and is validated (accuracy) on the exclusively set of validation data.

As we learned in the project about fraud detection, it is very important to check the balance of class proportions in our training data. If it is not balanced enough, accuracy is not a great measure and should be replaced with recall for example. As we saw in "Datasets and Inputs" the class distribution is quite balanced, hence accuracy will be the metric of choice.

Project design

The core of this project is an image classifier based on CNNs. The project design follows the typical ML Workflow presented in the course (or also on the AWS page https://aws.amazon.com/blogs/machine-learning/build-end-to-end-machine-learning-workflows-with-amazon-sagemaker-and-apache-airflow/)

In particular this includes the steps:

- Retrieve data
- Clean & Explore
- Prepare/Transform
- Develop & Train
- Validate / Evaluate Model

As the preprocessing part will be a crucial part of my project I will give a rough overview on how to implement these steps in my workflow:

- cropping pictures programmatically or build the preprocessing methods in a fashion, that they accept different sized images
- style transfer: apply the texture structures on dog images (based on AdaIn algorithm https://arxiv.org/abs/1703.06868)
- manual manipulations: edge detector & greyscale (probably using opencv)

Throughout the course I learned that AWS SageMaker and PyTorch in combination provide everything, to cover up this workflow process in each single step.

Hence the technical stack for the actual implementation will be

- PyTorch
- AWS & SageMaker

The CNN model specification will be defined with PyTorch. Since the authors of [https://openreview.net/forum?id=Bygh9j09KX] used a ResNet-50 models I am planning to do as well. PyTorch luckily already provides a pretrained model for this architecture https://pytorch.org/docs/stable/torchvision/models.html . Of course the last layer(s) will be retrained with my preprocessed images.

The actual training of the model will take place within AWS SageMaker to guarantee efficient training and a solid foundation for next steps in maintaining the application (since in the AWS SageMaker context such an application can be easily deployed to an endpoint, A/B testing is possible and much more).