Agustín Ávila Marín
November 25, 2021

CNN Project: Dog Breed Classifier

Domain background

I have decided to do the proposed Udacity CNN project due to familiarity about convolutional neural networks for object detection and segmentation in my current job position, where I have done extensive research to deploy CNN models in production, using primarily Tensorflow deployed in C++ for industrial applications, where the need for robust algorithms that can perform well enough at high speeds demands for the use of CNN architectures with the best latency-accuracy trade-off.

Ever since early days of computing, neural network architectures have been proposed for simple tasks, and their performances were often as good as clever designed algorithms such as Support Vector Machines, random forest, dictionary extraction, etc. For image processing though, neural networks promised high accuracy results, but it was easier to design algorithms using some sort of signal processing techniques such as Wavelet and Fourier transforms, Haar cascade filters or other types of custom designed filters depending on the problem to be solved. CNN methods were often overlooked because it was cumbersome to come up with an architecture that performed well at validation, and at the cost of spending valuable amount of time training and fine tuning it due the need of huge amounts of data and access to expensive hardware.

Since 2006 the CNN discussion started to gain momentum again thanks to the Internet, which facilitated collaboration, access to data, monetary investment and a gradual improvement in hardware for the Gaming industry. In 2012 AlexNet results on ImageNet challenge detonated the use of CNN for image classification tasks due to its impressive 10.8 percentage points reduction of the top-5 error versus the best performing algorithm at the time.

There have been a lot of new proposed CNN models for object detection and classification lately and their speeds and accuracies vary depending on the platform they are intended to be implemented.

This project offers me the opportunity to study and implement a MobileDet network architecture from scratch in Pytorch, since MobileDet comes from a Mobile architecture family with the main goal is to have models with state-of-the-art results at high speeds that can be implemented in real-time applications.

My preferred research and deployment platform is Tensorflow but will use Pytorch as suggested in the notebook.

At the end of this project, I will have the necessary understanding to implement a MobileDet architecture of my own design in real world industrial applications where results must be delivered almost instantaneously when dealing with high-speed production assembly lines.

Problem statement

In ML, image classification is a well-known field of research where there have been multiple approaches proposed. But it is often proved to be a difficult domain because there are few models achieving good results in general datasets. Take a dog breed classification task as an example; even humans will have a hard time distinguishing the breed of a dog, because breeds might share same characteristics. Take Fig. 1 and Fig. 2 as examples, they are different dogs but look alike. That's why for some applications it pays off to specialize a CNN model.

Agustín Ávila Marín

Fig. 2 Welsh springer spaniel

Fig. 2 Brittany

Dataset and Inputs

There are two datasets proposed in this notebook:

1. *Human faces dataset:* There is a total of 13233 human images; distributed in 5750 folders separated by name. Images have the same 250x250 size but vary in background, quality, pixel resolution, illumination, pose and are imbalanced. Dataset available for download at https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip
2. *Dog images dataset*: There is a total of 8351 dog images; X images for training, Y for validation and Z for testing, distributed in 133 dog breed classes. Images vary in size, background, image quality, resolution, illumination, pose, some of them have humans in there, dogs often are cropped, and classes are imbalanced. Dataset available for download at https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

Solution statement

In this project there are two detectors proposed, 1) a face detector based on OpenCV Haar Cascade filters to detect humans and 2) a dog detector to classify dogs based on CNN trained on ImageNet dataset. The aim of using two detectors is to identify first if the input image has a human or a dog in it and only then, when there is a dog detected in the image, we can apply a CNN classifier to predict its breed. If there is a human detected in the image, we can use the classifier to see how the CNN was activated to determine to which of the 133 breeds looks more like.

Benchmark model

1. The custom CNN model is expected to have a test accuracy of at least 10% after the first training.
2. When using transfer learning to improve test accuracy, the CNN model is expected to have a test accuracy of at least 60%.
3. Further research will be needed to test the model against pretrained ResNet, Xception, MobileNet, EfficientNet, etc., but for this time will suffice an accuracy better than a random guess of 1/133, since it will require more time to pre-train the CNN from scratch on the ImageNet dataset.

Agustín Ávila Marín

Evaluation metrics

A cross entropy loss will be used to evaluate the model. Due to the imbalance nature of the training dataset and the lack of images, using the accuracy won't be a suitable performance indicator.

Project design

Following the proposed pipeline in the notebook the steps are as follows:

1. Implement a human detector using OpenCV's Haar Cascades classifier
2. Implement a dog detector using a pretrained VGG/MobileNetV3 (small) classifier
3. Write custom data loaders for training, validation and test data.
4. Perform data augmentation in the training set to account for the imbalance and the size of the dataset.
5. Build the proposed CNN architecture from scratch based on the IBN-Fused-Tucker convolution families proposed in [4].
6. Train and test the model from scratch achieving at least 10% of accuracy in the test set.
7. Train and test the model using transfer learning to achieve at least 60% of accuracy in the test set.
8. Write a dog breed classifier application using the human/dog detectors and the CNN model trained to predict the resemblance of an image to a dog breed.

References

1. A. G. Howard, W. Wang et al., *"MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications"*, https://arxiv.org/pdf/1704.04861.pdf
2. M. Sandler, A. Howard et al., *"MobileNetV2: Inverted Residual and Linear Bottlenecks"*, https://arxiv.org/pdf/1801.04381.pdf
3. A. Howard, M. Sandler et al., *"Searching for MobileNetV3"*, https://arxiv.org/pdf/1905.02244.pdf
4. Y. Xiong, H. Liu et al., *"MobileDets: Searching for Object Detection Architectures for Mobile Accelerators"*, https://arxiv.org/pdf/2004.14525.pdf
5. *Udacity's dog breed classifier repository*, https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification
6. *Pytorch documentation*, https://pytorch.org/docs/stable/index.html
7. *Tensorflow models research repository*, https://github.com/tensorflow/models