Augmented Face Recognition

Marc Minnee - Udacity Machine Learning ND program 2020

Intro

The project started out as part of Udacity's Machine Learning nanodegree to identify a breed of dog from an input image. The notebook provided by Udacity follows a normal machine learning pipeline: importing datasets, creating a CNN classifier from scratch or using a pretrained model, training, validating and testing the model; in this case using Python's deep learning library PyTorch. I would like reaching an accuracy of > 90% but Udacity's (own expectance?) criterium was at least 60%. This is nothing special, lots of articles are already spent on describing workflows for machine learning.

The good, the bad and the ugly

So I decided to give a twist to this classification problem. Image and face recognition is getting a lot of attention in AI and machine learning. Applications can be used for both good and bad purposes. The good: easy and swift face recognition solves the problem of authentication in ever expanding granular digital authentication workflows. No more password hassle. The bad: state surveillance threatens human rights when people are constantly being watched and checked upon. The ugly: predictive policing with image recognition shows too many false positives based on biased (racial) inputs and poor performing models, not recognising subtle differences in facial attributes from human races.

The last one is especially interesting to solve. If we'd like to combat the ugly before getting worse, we need to understand why modelling predictive policing is controversial. First of all, used models are opaque in the sense that authorities don't want to open source these in order to outsmart criminals. This way we cannot objectively determine the performance and evaluate the inputs used to train the model. And if we are to investigate the inputs, we likely find datasets, based on historical data of imprisoned persons which are biased towards Afro-Americans and Latinos. So, the problem has its origin in biased datasets, training models with more images of a certain kind, also described by Cathy Neil. We should be able to enhance models by reducing biased inputs and account for differences in human race. In order to prevent the ugly becoming the bad…

Getting unbiased input

Udacity's Jupyter notebook for dog breed classification gives me a good base for developing a model classifying human race, or at least for some derived classes of human race. We can train a model which is better capable of recognising differences in gender and race, as a means to show more transparency in model use, to account for (un)biased input and to perform better at recognising faces of different races.

Disclaimer: I'm not suggesting we should model a classifier as a means to segment or discriminate people, I would merely like to point out that I'm investigating the use of balanced inputs in order to increase the performance of face recognition as a research objective.

As a base input I used a processed CelebA dataset from Kaggle, with 202,599 number of face images of various celebrities. However we can extract many facial features, the label we are interested in is race or skin type. We need to infer this label from the other features in our dataset, by labeling a sample of the images by hand and then use a data augmentation or a generative modelling technique, a GAN, to get more training data with race or skin type labels. It would have been better to use the Gender Shades dataset, which already accounts for different skin types, unfortunately the organization managing this dataset was not responsive to my data acquisition requests. Back to square 1: the CelebA dataset.

First, I had to assemble subsets from this dataset using a kind of filter app to look for features describing my classes. Fortunately, the Kaggle processed CelebA dataset had the following labels described: `5_o_Clock_Shadow, Arched_Eyebrows, Attractive, Bags_Under_Eyes, Bald, Bangs, Big_Lips, Big_Nose, Black_Hair, Blond_Hair, Blurry, Brown_Hair, Bushy_Eyebrows, Chubby, Double_Chin, Eyeglasses, Goatee, Gray_Hair, Heavy_Makeup, High_Cheekbones, Male, Mouth_Slightly_Open, Mustache, Narrow_Eyes, No_Beard, Oval_Face, Pale_Skin, Pointy_Nose, Receding_Hairline, Rosy_Cheeks, Sideburns, Smiling, Straight_Hair, Wavy_Hair, Wearing_Earrings, Wearing_Hat, Wearing_Lipstick, Wearing_Necklace, Wearing_Necktie, Young.`

Based on the outcome of the Gender shades project, stating it is especially hard to accurately classify black women in face recognition, my aim was to assemble a balanced dataset with 4 x 2 classes (skin type x gender): ***black, caucasian, blond and 'tanned'*** *(everything* ***not*** *black, caucasian or blond)* ***men and women***. Yes, I admit having chosen these classes somewhat arbitrarily or maybe even awkwardly, but the goal here is to generate a balanced dataset, not so much account for segmentations of human race (I'd rather stay away from this sensitive subject).

I decided to filter on the features making any sense to my chosen labels and ended up after some probing with these: `Big_Lips, Big_Nose, Black_Hair, Blond_Hair, Brown_Hair, Bushy_Eyebrows, Male, Pale_Skin, Pointy_Nose, Young.` Tweaking the features (+1, -1) and feeding this in a **'Filter' notebook** generates a reasonable subset to work with per class, however I needed to do quite some data cleansing afterwards, like women labeled as men (transgenders, anyone?), red hair typed as blond, this kind of noise. *Indeed, the real deal of machine learning is not the modern art of training your model, but the medieval job of hand picking the dirt from your input data.*

When done I had a nice subset of 50 to 100 samples for each class. Time to get more. First, I tried generating more samples with a GAN, used from the PyTorch tutorial https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html, however results were too poor to make use of them, because of the very small subset of samples I had to feed the GAN with. Plan B: I used a very handy and simple data augmentation python library, `imgaug` in order to rapidly get more training data, using a **'Transform' notebook**. I ended up with about 20k samples for each of the 8 classes.

Training the model

Now we can account for unbiased inputs, a balanced dataset to work with as input for training. I used the exact same steps as in the dog breed classifier, the hard part is in architecting the layers of the CNN. This is truly something of an art: how many convolution layers, input/output ratio's, pooling, dropout? I admit googling for some answers, in the end it's about getting it right by some rule of thumb. So I used 3 (to 5) convolution with corresponding max pooling layers. The filters doubled in each layer, concluding with 1 fully connected linear layer outputting 8 nodes, our 8 classes of skin type x gender.

After quite a while (I could have used some cloud provider GPU instance, but I wanted to see how my turbo MacBook was performing on CPU… a pity CUDA is not Mac's friend any longer), about 1 hour per epoch, 15 epochs in total training, I reached a model that performed 95% on the chopped-off test set, which was very promising. Training with a much smaller set (of about 4k samples in total) reached 66% accuracy, so that's why I generated 20k per class. But, since I used a small base of 50-100 samples per class before augmenting them, I feared overfitting because the 20k samples were generated from only a couple of 'parents' so maybe this 'incestuous' action had been clouding the machine's judgement. To be sure, I generated a new testset, based on samples that were not derived from the original subsets.

Ok, back again to the Filter notebook, this time sampling from the end of the CelebA data set (sample # > 170000) selecting about 50 samples per class and augmenting again until we have some 4k samples per class as a new testset feeding into our shiny model. Alas, as expected, our model does not shine anymore: only 52% accuracy. My fear had become true, this what you get when feeding images that are too much alike. I should have taken more time to build a thoroughly balanced dataset, using more advanced techniques to augment data samples or tuning a GAN to do that. Feel free to grab my stuff and do as you like…maybe I'm suffering from augmentation fatigue..

Conclusion

Starting out with the dog breed classification project, I managed to train a CNN which at first sight did the job on an unbiased, balanced dataset of skintype x gender classes. Unfortunately, I had quite some data cleansing to do in order to get this balance, but in the process I could not help being biased myself in labeling images the correct way. I didn't have the time to traverse 200k samples so I relied on simple data augmentation techniques to build a decent dataset, however prone to overfitting as it turned out.

Well, in the end we got some results and the main goal of this project was to produce an unbiased dataset or at least a method to produce one. As we learned from the dog breed classification project, predicting classes of this kind is a real challenge. Even for our 8 classes it turned out to be difficult, and we didn't have a pretrained model in order to speed things up. Also, it was a pity I could not use the Gender shades dataset, which accounted for better labeling to start with. Nevertheless, this was a good exercise in finding ways to produce unbiased, balanced datasets, as a means to prevent the ugly becoming the bad…