Udacity Machine Learning Engineer Nanodegree Capstone Proposal

Eric Seidler

2021-03-03

Domain Background

This Person Does Not Exist is a website released in 2018 that displays photorealistic images of people that do not exist. The images are created using a type of unsupervised learning known as generative adversarial networks [1]. This type of network contains two main components, a generator and a discriminator. In the context of this specific application, the generator starts with random data and through training learns how to create more and more realistic photos of humans while the discriminator attempts to tell the difference between photos of real humans and artificially generated ones [2]. The end result is something that (to a human) appears to be a photo of a real person but is not.

I was fascinated by this concept when I first heard about it. Immediately a question came to mind: "Could a machine learning model be trained to outsmart another one?". In other words, a human would not be able to differentiate between a photo of a real human and one from this website, but could we train an ML model to do so?

Problem Statement

The problem at hand can be described as such. How can we use machine learning, specifically a neural network, to differentiate between images of real people and artificially generated images of people? And once a model is trained, how accurate is that model?

Datasets and Inputs

The images of people that do not exist will be obtained from thispersondoesnotexist.com via a Python script. Images of real people will be obtained from pexels.com and/or kaggle.com. Five hundred images from each category (real and not real) will be collected. This assumes that the aforementioned websites will provide that many photos of people who do not exist. If that assumption does not hold, the size of the input datasets will be reduced.

Solution Statement

Obtain a set of data that contains an equal number of photos of real people and artificially generated photos of people. Train a neural network on this dataset. Then as input to the trained model, accept a real or artificially generated image of a person. The hypothesis is that a properly chosen and trained model should be able to accurately determine whether an input

image is that of a real person or one that is artificially generated. The rationale is that neural networks are great at recognizing patterns in data, even when a human cannot. As a point of reference, we can also record the accuracy of a human being performing the same task.

Benchmark Model

Two simple models will be used as a point of comparison. The first will be a dummy estimator from sklearn [3]. The second will be a binary SVM classifier. Both of these will serve as good reference points for how well the custom neural network will perform and both will be able to be evaluated using the same accuracy metric, mentioned below.

Evaluation Metrics

The metric used in this project will be accuracy. Within the problem domain of binary classification, accuracy is defined as the sum of true positives and true negatives divided by the size of the dataset [4]. Furthermore, the accuracy of the custom neural net will be compared against the accuracy of a dummy classifier and perhaps a binary SVM classifier.

Project Design

Data preprocessing

Data will be collected from the sources mentioned in the Datasets and Inputs section.

After all datasets have been assembled, image data will be standardized across both real and generated photos. Depending on the variability of the images, the following transformations will be applied: resizing to fixed image dimensions, flipping the image (horizontally only since we won't be training or recognizing upside-down faces), and ensuring that all input images are converted to standard RGB values and then normalized. Another common preprocessing step when working with image data is to convert to black and white values (no color). But in this case, RGB values will be used since the models might need that extra color information to be able to differentiate between types of photos.

Splitting the data

Once the data has been prepared, splitting into train and test datasets should be trivial. Given the relatively small size of the input datasets (1000 images total), an 80/20 train and test split will be used.

Training the Model

A pretrained model from PyTorch [5] will be used in conjunction with the custom neural net that will serve as the classifier. A typical model training loop will be used that includes the following steps of feed-forward, calculate loss, back-propagate the loss, and take an optimizer step. Number of epochs and other hyperparameters will be adjust as needed to increase overall model accuracy.

Sources

[1] https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
[2] https://developers.google.com/machine-learning/gan/gan_structure
[3] https://scikit-learn.org/stable/modules/model_evaluation.html#dummy-estimators
[4] https://en.wikipedia.org/wiki/Accuracy_and_precision#In_binary_classification
[5] https://pytorch.org/vision/0.8/models.html