Machine Learning Engineer Nanodegree:
Image Localization for Emotion Detection

Capstone Proposal

Bartlomiej Chrzaszcz
June 25th, 2017

Proposal

Over the last couple of years, image classification has been getting a lot of traction in different industries. Tesla and Google have been using this technology for self-driving cars and Facebook has been doing it for facial recognition for example. They are also very accessible through the different APIs out there and these powerful algorithms can even be used on your smartphone today. These algorithms use deep learning algorithms, specifically convolutional neural networks, which is a subset of machine learning, to learn what something is out of a list of possibilities. For example, if you train an algorithm on a set of images containing 10 different items, that algorithm would be good at identifying those 10 items in new sets of images you give it. As seen in Figure 1, the algorithm uses techniques such as convolutional layers, RELU activations, and pooling layers, to end up figuring out what's the probability of the item in the image being a car, truck, airplane, ship, horse, or something else using a regular fully connected neural network at the end of the convolutional layers. CNNs are still a very new field, with them being explored first in 1998 by researchers at the AT&T Shannon Lab where they discovered how affected CNNs are at recognizing simple objects, such as the familiar MNIST dataset [1]. There have been fairly revolutionary CNNs that have created great classification accuracies such as Fractional Max-Pooling [2] and The All Convolutional Net [3] that have created a lot of buzz in the area about CNNs potential.

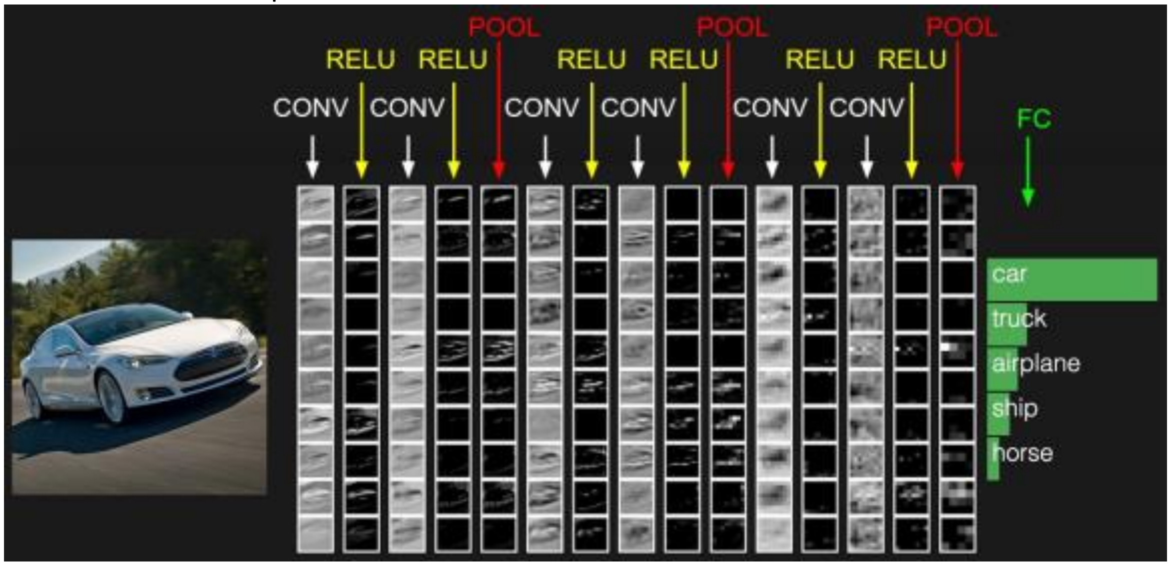


Figure 1

Problem Statement

In this study, we will mainly look into using CNNs to recognize emotions of humans using pictures of their faces and validate its accuracy in recognizing emotions of a single face. As aa secondary goal, we will determine their success in object localization so they can be fed an image of multiple individuals and determine each person's emotion. To test this, we will use the same images we use to test the original CNN by stitching them together into a 4 image square. Then, after doing some further research to see if it is achievable, try feeding the algorithm images of multiple individuals in larger, more complex images like on a typical individuals Facebook profile (will use my own images from Facebook where I have gotten consent from individuals and also stock/shutterstock images which are allowed to be distributed).

Datasets and Inputs

The dataset can be attained here: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

The dataset that will be used has been made available by Pierre-Luc Carrier and Aaron Courville and is available on Kaggle under the name "Challenges in Representation Learning: Facial Expression Recognition Challenge". The dataset is also included within this zip file. It consists of 35,887 48x48-pixel grayscale images of faces that have already been classified into seven categories:

- Angry (4953 entries)
- Disgust (547 entries)
- Fear (5121 entries)
- Happy (8989 entries)
- Sad (6077 entries)
- Surprise (4002 entries)
- Neutral (6198 entries

More specifically, the training set consists of 28,709 examples while the testing set is split up into 2 3,589-image groups: the private tests used for the Kaggle leaderboards and public tests for validating one's algorithm. In this study, I'll use one of the testing sets as validation of my accuracy and the other one to test its localization capabilities by stitching the images together like I discussed earlier. Given that I want to train my CNN to detect emotions, this dataset is perfect for doing so as all the images are labelled. Also, as you can see, there are very few pictures of Disgust, so my CNN may not become so great at identifying that emotion in the real world. I may consider removing all images disgust later down the road after further exploration. I'm guessing this will have negative effect on my accuracy and other metrics.

One thing to note is that the .csv file that comes with the dataset has each image as a list of pixels in one long row. So, I will have to do some data processing/transformation to turn that into actual images. I wanted to explore the data before deciding I wanted to use this dataset so I already turned those rows of pixels into images as shown below (using csv2Image.py as located in the *.zip file). As you can see, this dataset will be difficult to train on due to the watermarks in some images and invalid images in general. I will leave the watermarked images but I'll do my best to skim through the dataset and remove some of those invalid images like the one with the 3 dots in the center of Figure 2

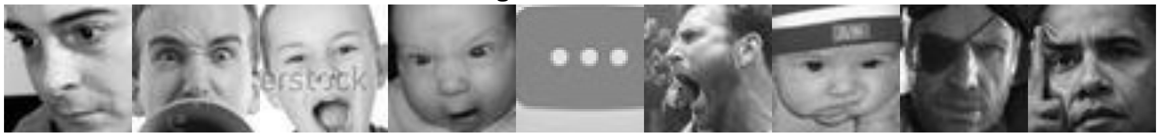


Figure 2

Solution Statement

After preprocessing the data, which includes turning data entries into images, creating a validation set from the training set provided, turn the integer pixel values into 16 bit floats for tensor flow, possibly feature scale the pixel values by something like a min/max rescaler and randomizing the order of images to make sure they really are random, I will start designing my CNN (note that a lot of the preprocessing has been done for me such as gray scaling and dividing into training and testing sets). The CNN that I train should be able to successfully classify the data in the private testing set with at least a 60% accuracy. I based this off the leaderboards for the Kaggle competition where the top 10 people have at least a 60% accuracy, best being 71.2%. The CNN should also be able to recognize/localize faces in the stitched image dataset I create and classify them correctly. I will validate this by picking a couple examples and seeing whether the CNN classified them correctly.

Benchmark Model

I plan to compare my accuracy results of the CNN against the accuracy of those who completed this project on the Kaggle leaderboards. In addition, I will train my CNN I made for the image classification Nanodegree project (with some slight modifications to account for a different sized image) and test it with the same metrics I'll measure my new CNN with. For my localization task, I will just explore how my CNN classifies the faces but I will not explore how accurate it is over all the examples. So, I will just go over some examples looking to see if it correctly detects a face in the combined image first, and then see if it classified its emotions correctly.

Evaluation Metrics

Since my CNN deals with a classification problem, I will use F1-score to find how well my CNN performs since my data is skewed in that there aren't many examples of disgust images (thus I would get a high accuracy percentage which wouldn't be reflective of my data). However, I will attempt training my CNN without the disgust images and in this instance I will try using precision, recall, and accuracy in addition to F1-score. Also, I will want to determine how quickly my CNN classifies images as a CNN that takes minutes to classify isn't very good in some application such as a self driving; in this case, my CNN wouldn't need to be super-fast but it would be an important metric to know. Again, for my image stitching, that will be more for my observation having me pick random examples and see if it can detect a face and correctly classify that face.

Project Design

The very first thing I'll need to do is clean the data. As discussed above, the data consists of one long line of numbers in a .csv file cell. I will need to go through all the data, convert those lines of numbers to 2-d arrays of size 48x48, and add their corresponding label to a large panda data frame. After that, I will need to convert all those entries to images and see what images are "outliers" in that they aren't even pictures of faces. Additionally, I'll need to create the stitched images for once I test out how well my CNN does at localization.

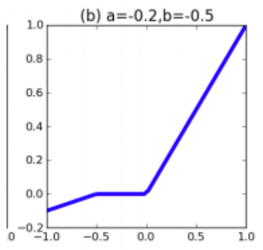
After that, I will need to figure out the architecture of my CNN. Specifically, I want to try implementing Fractional Max-Pooling [2] and a learned activation function for my CNN [3][4]. All 3 papers' CNNs were tested on the CIFAR-10 dataset and achieved stellar results, both classifying 32x32 color images with 10 possible labels with 92-97% accuracy (they were tested on some other datasets too, not just this one). Since my research will focus on similarly sized images (48x48) with similar number of labels (seven), I feel these approaches will still be highly affective for my dataset even though the CIFAR-10 dataset consists of RGB images and my dataset is gray scaled.

Firstly, Fractional Max-Pooling , or FMP for short, is a recent new development in CNN research. For a while, an MP2 layer which is a 2x2 max-pooling layer has been used primarily in CNNs because they are fast, reduce the size of the layers efficiently, and encode common patterns in images. However, since the size decreases very quickly, you cannot have a convolutional layer followed by a max pooling layer several times so a deep layer of many CNNs is needed which may introduce overfitting in its own right. FMPs on the other hand, reduce the size of an image by some factor a where $a \in (1,2)$. Because of this, they can be more arbitrarily inserted into a layer of CNNs to maintain the invariance of the CNN without losing too much information by reducing the size before introducing more convolutional layers. This fraction is usually calculated by dividing the current size of the matrix by the new matrix/image you want (e.g. I have a 100x200px image reduced to 75*150 so I'll have a 4/3 max-pooling layer). However, they can still be effective when choosing alphas randomly and also in random locations in the CNN. One issue with random FMP is that they tend to underfit when combined with dropout.

For a learned activation function, I will first try to implement an Adaptive Piecewise Linear (APL) activation unit [4] which has the following formula:

$$h_i(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$$

where S is a hyper-parameter set in advanced which says how many "hinges" or rectifier activation functions there are. The other variables, a_i^s and b_i^s for i $\in (1,S)$. The , a_i^s variables control the slopes of the linear segments while the b_i^s variables control the locations of the rectifiers. Some interesting things can happen such as non convex activation functions like below and for a large enough S, can create complex shaped activation functions:



If I fail to successfully implement this activation function (not in TensorFlow yet), I'll most likely use a PReLU activation function [5] which also learns some value alpha when trained as seen in the function below on the right:

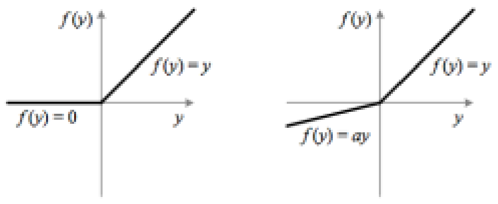


Figure 1. ReLU vs. PReLU. For PReLU, the coefficient of the negative part is not constant and is adaptively learned.

After training my model and verifying that it accurately classifies images, I'll go ahead and test out its ability to localize and correctly classify faces in the stitched images. And finally, I will try and figure out whether I can modify my CNN so that it can take in pictures of arbitrary size so I can test it against general Facebook images.

References

[1] LeCun, Yann, Patrick Haffner, Leon Bottou, and Yoshua Bengio. (1998). *AT&T Shannon Lab*. Retrieved June 26, 2017, from http://yann.lecun.com/exdb/publis/pdf/lecun-99.pdf

[2] Graham, B. (2015). Fractional Max-Pooling. *Dept of Statistics, University of Warwick*. Retrieved June 23, 2017, from https://arxiv.org/abs/1412.6071.

[3] Springenberg, J., Dosovitskiy, A., Brox,, T., & Riedmiller, M. (2015). *Department of Computer Science, University of Freiburg*. Retrieved June 23, 2017, from https://arxiv.org/abs/1412.6806.

[4] Agostinelli, F., Hoffman, M., Sadowski,, P., & Baldi, P. (2015). Learning Activation Functions To Improve Deep Neural Networks. *ICLR 2015*. Retrieved June 27, 2017, from https://arxiv.org/pdf/1412.6830.pdf.

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Microsoft Research*. Retrieved June 27, 2017, from https://arxiv.org/pdf/1502.01852.pdf.