Machine Learning Engineer Nanodegree

Capstone Proposal

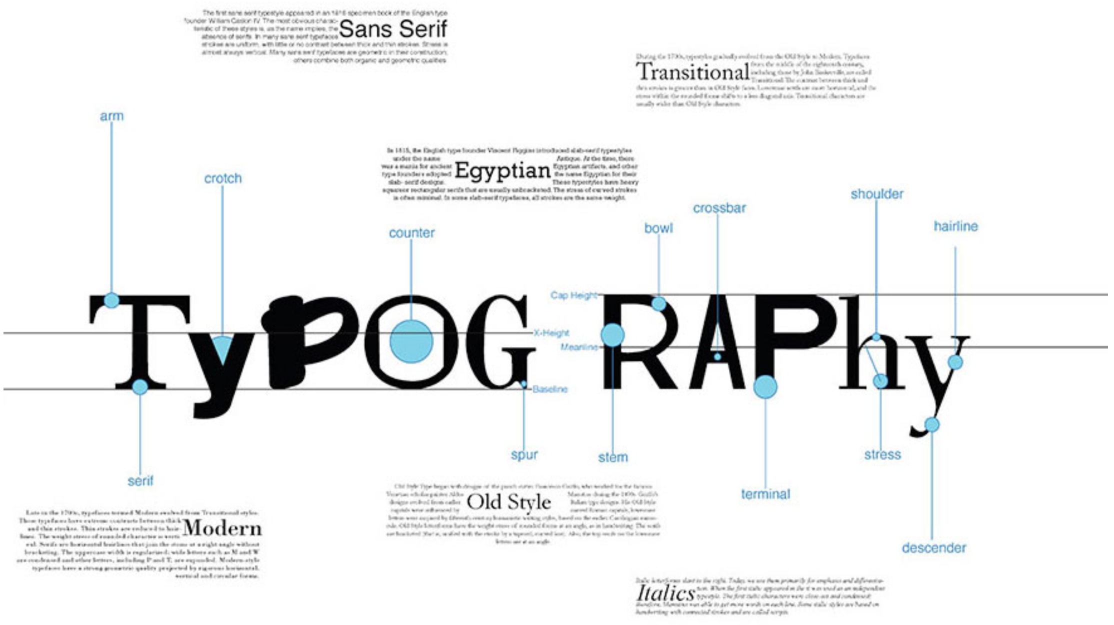
Alan Sun

June 27th, 2017

Proposal

I am planning to build a model to detect the font from an image using CNN networks and some other tricks. CNN networks work so good on number detecting like they did on the MNIST dataset. They can precisely recognize those very little details of the numbers. I was thinking, why not use this tool on another interesting problem, font detecting.

As you know, texts with same font have the same shape and details. There' s a lot things to make a font different from each other. Many experts who know well on Typography will detect a font by a look. Mostly because they have already seen so many fonts before, so they can detect those fonts they used or seen before easily.

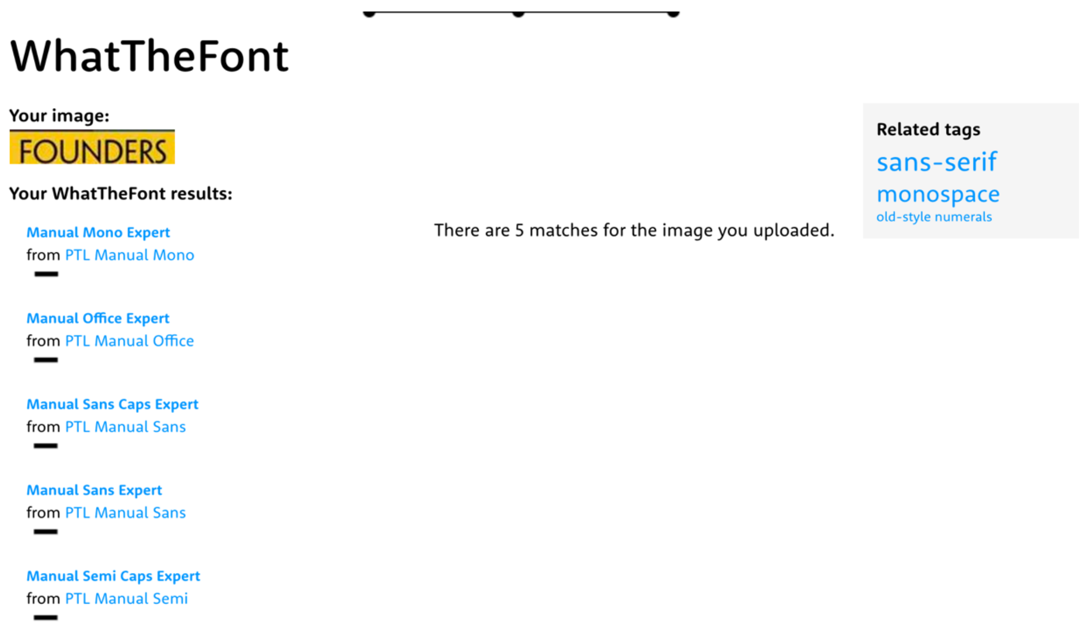


Intuitively, the Machine Learning can do as well as the human experts, since the problem here looks like another classification problem. Since this problem should be an image classification problem, I think we should try the CNNs to solve the problem.

Domain Background

Background information:

Designers are the group of people who struggle to find what is the font of the text in a picture. Currently, there are only a few chooses. The most recommended tool is the website of MyFonts.com (https://www.myfonts.com). That website can relatively detect fonts in some very regularized pictures. But it can barely get a correct answer through some real world photographs or screen shots, like the following ones.



WhatTheFont

Your image:

FOUNDERS

Your WhatTheFont results:

There are 5 matches for the image you uploaded.

Manual Mono Expert
from PTL Manual Mono

Manual Office Expert
from PTL Manual Office

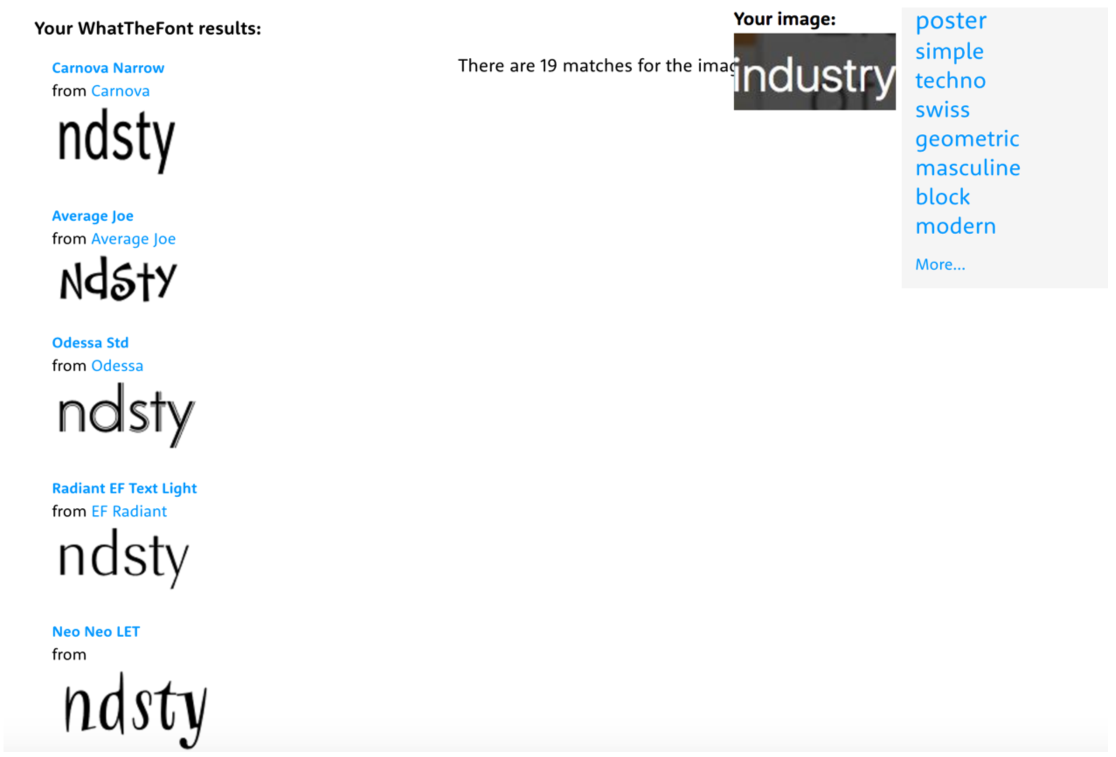
Manual Sans Caps Expert
from PTL Manual Sans

Manual Sans Expert
from PTL Manual Sans

Manual Semi Caps Expert
from PTL Manual Semi

Related tags
sans-serif
monospace
old-style numerals

MyFont.com failed on texts on colored backgrounds



Your WhatTheFont results:

There are 19 matches for the imag

Your image:

industry

Carnova Narrow
from Carnova
ndsty

Average Joe
from Average Joe
NdStY

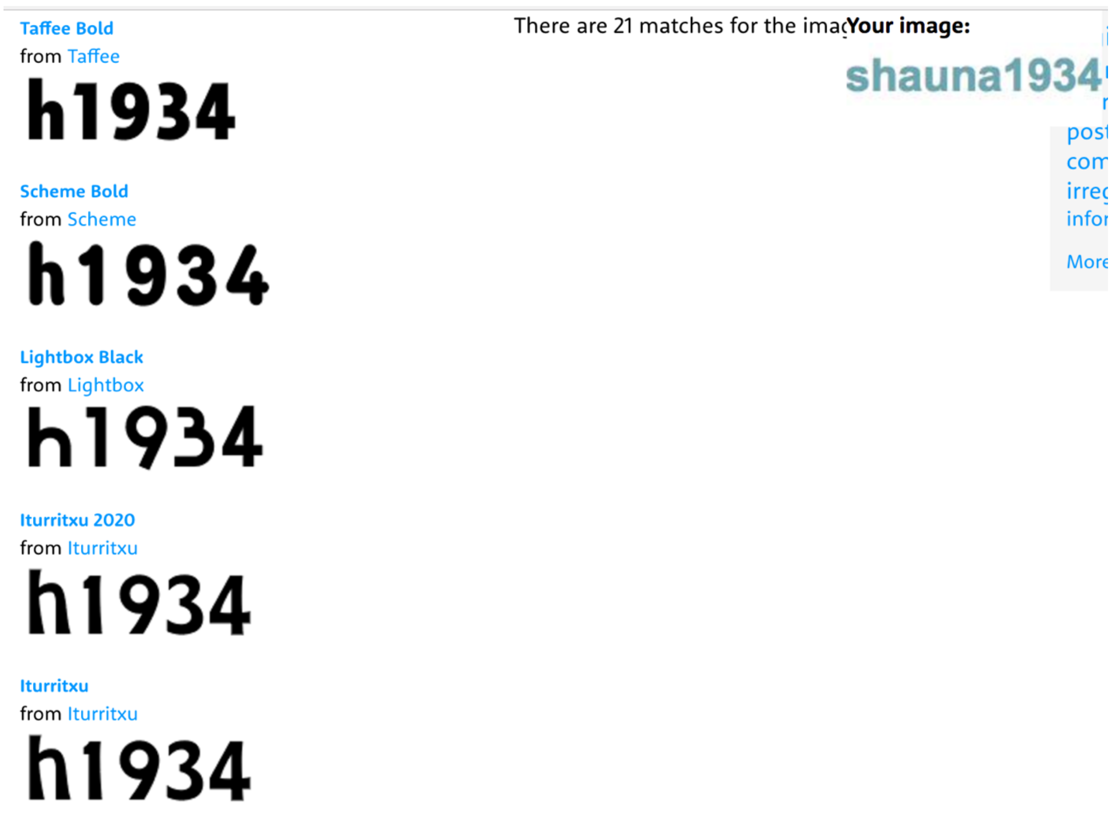
Odessa Std
from Odessa
ndsty

Radiant EF Text Light
from EF Radiant
ndsty

Neo Neo LET
from
ndsty

poster
simple
techno
swiss
geometric
masculine
block
modern
More...

MyFont.com failed on texts on pictured backgrounds


There are 21 matches for the imag Your image:

shauna1934

post
com
irreg
infor

More

Taffee Bold
from Taffee
h1934

Scheme Bold
from Scheme
h1934

Lightbox Black
from Lightbox
h1934

Iturritxu 2020
from Iturritxu
h1934

Iturritxu
from Iturritxu
h1934

MyFont.com failed on colored bold texts

The task is quite challenging because there will be a lot of differs onto the original font. Like more weights on the font, Italian type of the font, transferred horizontally or vertically, etc. However, to anyone who are already quite familiar with MNIST challenge or text detection challenges, these problems are just some clichés. Tools like Keras already provide some mature functions to deal with it.

Relative research

The research field is called VFR (Visual Font Recognition). There is a recent research called deep font : identify your font from an image from Zhangyang Wang (https://arxiv.org/abs/1507.03196). And the team finally launched the font finding tool in Adobe called 'Recognize Type' in Photoshop (https://www.youtube.com/watch?v=5eJ3IXYcw3M). Basically, recent research of this aims on using Deep Neural Networks like CNN to solve this problem. And I think recent tech like special transformer (also using CNN net) may help to improve the results.

Personal motivation

As I am working as a Product Manager with a bunch of Visual Designers. They tell me that finding the font from an image is a challenge they are facing every day. They have requests and I have tools, why not have a try.

This project is related with visual recognition, deep learning and CNN networks.

Problem Statement

Let me reclaim the task here.

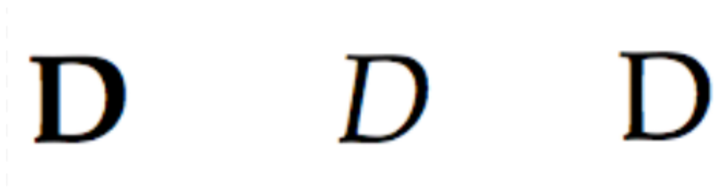
Task: train a network to recognize font in the image

Limits: this project do not contain word splitting and recognizing. All the inputs should be grouped correctly as one image which only contain one letter or one digit and the letter or digit in it.

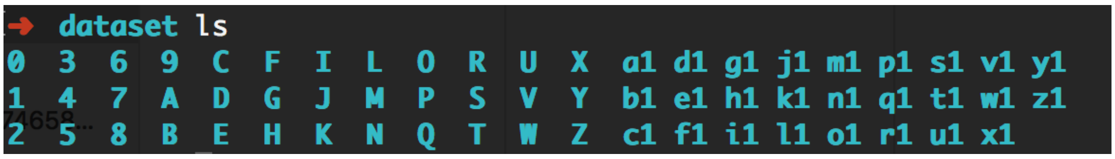
Goal: Get an Top5 Accuracy more than 90%. Top1 Accuracy should be more than 75%.

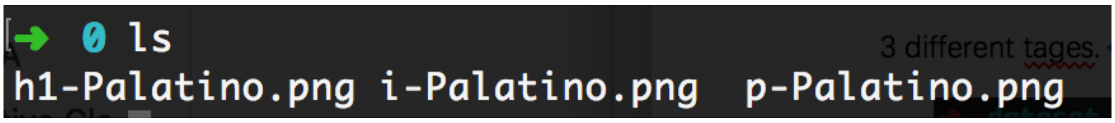
Datasets and Inputs

Training Dataset: images of only one letter (including upper case, lower case and digits in total) of different fonts (at least 150 fonts) and a label file of what the letter or digit is. The size of the Dataset should be bigger than 150*(26+26+10)*3 = 27900. The first 26 is the number of upper case letters and the second 26 is the number of all lower case letters. And finally the last 10 is the number of all digits from 0 to 9. So every group of font samples should have at least 62 images. To improve the robustness of the model. We will capture 3 different cases which are p tag, h1 tag and I tag (html tag). They are relatively quite different like the image below. I have already tried the h2 and h3 tag, they are just quite the same with h1 tag so that they will not be sampled in the dataset.


left: <h1> middle: <i> right: <p>

All images are grouped by the content they have. So the dataset will have 62 folders. In each folder there would be 150*3 = 450 images for 150 fonts and 3 different tags. This structure will help to train 62 different Neural Network to identify the fonts, since we will tag each train and test images what the content they have. I believe 62 separate Neural Networks will give a better results than 1 single network, since the networks have a more specific task to finish.


```
➜  dataset ls
0  3  6  9  C  F  I  L  O  R  U  X  a1 d1 g1 j1 m1 p1 s1 v1 y1
1  4  7  A  D  G  J  M  P  S  V  Y  b1 e1 h1 k1 n1 q1 t1 w1 z1
2  5  8  B  E  H  K  N  Q  T  W  Z  c1 f1 i1 l1 o1 r1 u1 x1
```
a1 means lower case a, and A means upper case A


```
➜  0 ls
h1-Palatino.png i-Palatino.png  p-Palatino.png
```

in each folder, there are 3 images for each font

Test Dataset: some picture of texts with colored text and image background (at least 300)

Inputs of the net: Images of one letter or digit

Outputs of the net: the most possible 5 fonts and their scores

How to get the train dataset: The work flow is like this. Firstly, I use Javascript to generate a html file which has all 62 contents I want. Then I took a screenshot of this webpage. Finally, I run a python script to split the screenshot to 62*3 (3 tags remember?) images and save them into separate folders. By doing so, I believe I can capture all the data of one font for about less than 10mins. Although the workload for gathering the dataset still sounds a little bit heavy, though doable, I am already trying some possible ways to verify the process to minimize human interferences.

About resolution: The auto-proceeded images are 129*129. There is about half of the image is blank for enough bleeding for any mistakes of the auto-proceeded process. So the resolution for the content is like 52*70. I think the resolution is big enough so that I can identify the characters of the font by eyes. And the Keras.ImageDataGenerator will help to process the datasets further, so I will leave it for now.

Solution Statement

Solution: Using OpenCV and Keras to improve the quality of the input. Then use Spatial Transform Network and batch norm function to normalize the data. Finally, train a VGG like CNN network with dropouts to capture the details to classify the image.

The OpenCV and keras.imagedatagenerator will be used to clean the data, get the layout of contents and keep the model robust with some basic special transformations.

Benchmark Model

We will try the test dataset on both the classical MyFont.com website and the font detector from Adobe. Both of them give a result lists, so we will measure the Top1 Accuracy and Top5 Accuracy with these two model.

Project Design

About the dataset: We will try to use java or python to print all upper case and lower case letters of all 150 fonts with a crazy big font size and make a screenshot. After splitting those images we will get a relatively complete dataset to train.

About the workflow: Firstly, we will try to build the core CNN network. After some tuning, we will add some processes (OpenCV and Keras. ImageDataGenerator) to the input images to keep the inputs clean and grey. Finally, we will try to use LSTM to share the weights all over the same font weights instead separating the weights of each letters of a same font. Hope we will get a better result.

I would try ZCA Whitening (ImageDataGenerator(zca_whitening=True)) to emphasize the layout for the network. And also for potential rotation, I would try Random Rotations (ImageDataGenerator(rotation_range=degree)). And finally, with cautions, I would use Random Shifts

(ImageDataGenerator(width_shift_range=shift, height_shift_range=shift)) to try to detect contents not center localized.

For OpenCV, I will use cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) function to turn colored images to grey format. And use dilate and erode to clean the potential noises. Maybe I will try to use the cv2.adaptivethreshould to clean the grey part once again to make it more clear.

Basically, those are the plans for now. After all these tricks for data processing, I will try to use LSTM to share the weights between different networks (over all the lower case and upper case separately) to see if we can push the result further.

The OpenCV function will be used at the very beginning, following with special transform networks, and then to the keras image data generators, and then CNN networks, and then some batch normalizations, and some fully connected Neural Networks, and finally the output. I have used this structure (but without OpenCV functions) during my first kaggle competition and get a top100 result for the MNIST dataset. I cannot wait to give it this more challenging task.