BRAND USAGE ANALYSIS

GAURANG DAVDA

DIVYA EMMANUEL

NIRANJHANI VASUDEVAN

Overview

Social Media services are websites or applications where people share their views about anything using images, videos, comments etc. These websites have more than 100 million users registered worldwide. In America alone, around 71% of teenagers had a social media account in 2015 and the number is increasing exponentially. It is said that user-generated content is the lifeblood of Social Media. User-generated content is in different forms like images, videos, tweets etc. This is bread and butter of Business Analytics on Social Media. Brand usage among different type of people living at different locations can be effectively analysed to derive conclusions on Social Media Marketing etc.

Goals

Images posted online contains key information about behaviour, relationships, hobbies, status etc. Here, we will be scraping images from Instagram, which is very famous for posting images and videos, and,

1. Analyze the brands preferred by the user
2. Derive conclusions on the demographic popularity of the brands, most visible brands, advertisements in which the user might be interested

Data

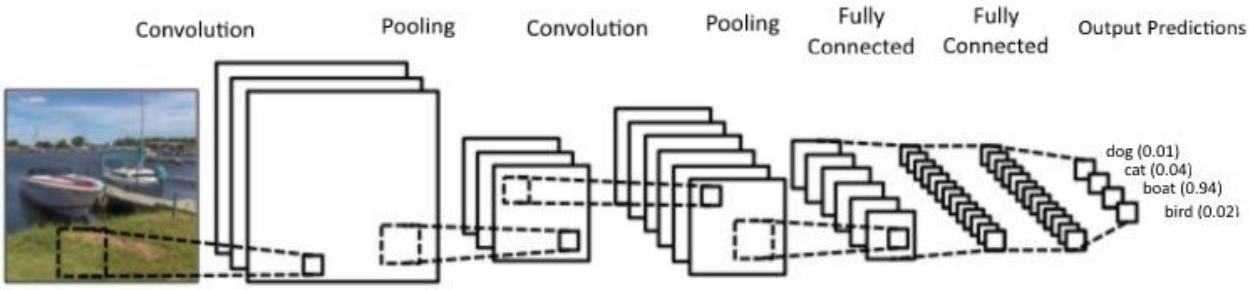
We will be creating a training set of images by using google image api for downloading and save them in hard disk.

Process outline

1. Removing corrupted images
2. Read all the images from the training images
3. Use convolutional neural network to localize the brand logos
4. Crop the localized image and pass it to another cnn to identify the brand name
5. Design a pipeline and a system to implement this approach
6. Deploy the model on aws
7. Build a webapp to demonstrate the different analysis

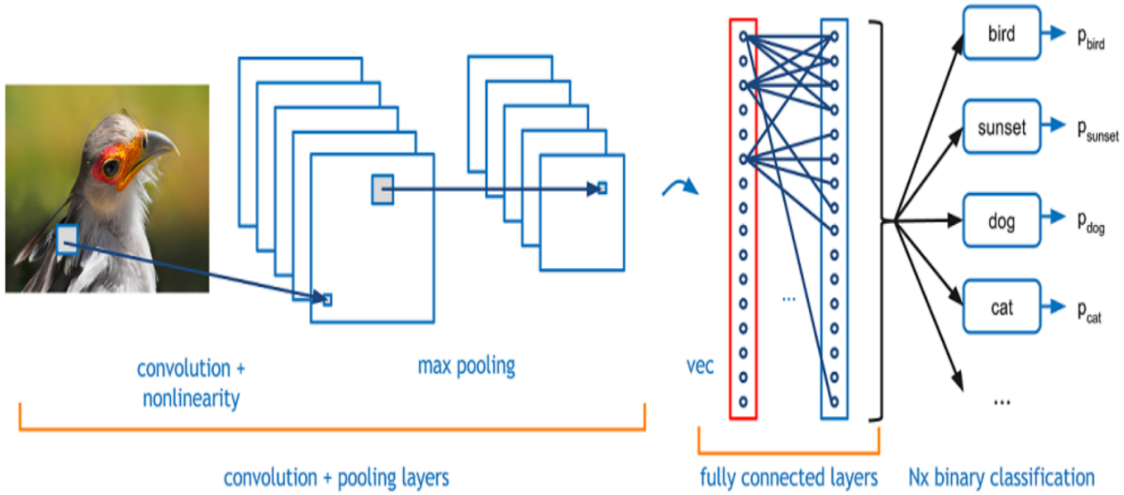
Convolutional neural network

CNN architecture:



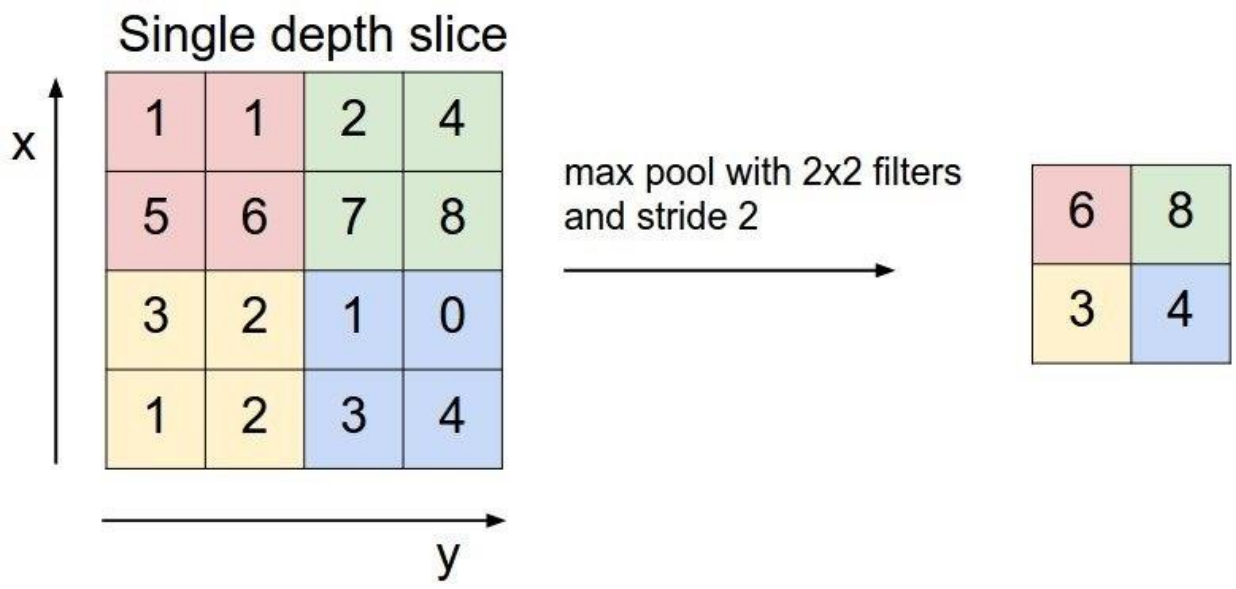
1. Convolutional layer

The first layer in a cnn is always a convolutional layer. Consider input image as 32 x 32 x 3 array of pixel values now, the best way to explain a conv layer is to imagine a flashlight that is shining over the top left of the image. Let's say that the light this flashlight shines covers a 5 x 5 area. And now, let's imagine this flashlight sliding across all the areas of the input image. In machine learning terms, this flashlight is called a filter and the region that it is shining over is called the receptive field. Now this filter is also an array of numbers (the numbers are called weights or parameters). A very important note is that the depth of this filter has to be the same as the depth of the input, so the dimensions of this filter is 5 x 5 x 3. Now, let's take the first position the filter is in for example. It would be the top left corner. As the filter is sliding, or convolving, around the input image, it is multiplying the values in the filter with the original pixel values of the image (aka computing element wise multiplications). These multiplications are all summed up and we get a single number. Now, we repeat this process for every location on the input volume. (next step would be moving the filter to the right by 1 unit, then right again by 1, and so on). Every unique location on the input volume produces a number. After sliding the filter over all the locations, you will find out that what you're left with is a 28 x 28 x 1 array of numbers, which we call an activation map or feature map. The reason you get a 28 x 28 array is that there are 784 different locations that a 5 x 5 filter can fit on a 32 x 32 input image. These 784 numbers are mapped to a 28 x 28 array.



2. Pooling layer:

Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping sub regions of the initial representation. Let's say we have a 4x4 matrix representing our initial input. Let's say, as well, that we have a 2x2 filter that we'll run over our input. We'll have a stride of 2 (meaning the (dx, dy) for stepping over our input will be (2, 2)) and won't overlap regions. For each of the regions represented by the filter, we will take the max of that region and create a new, output matrix where each element is the max of a region in the original input.



3. **Fully connected layer:**

This layer basically takes an input volume (whatever the output is of the conv or relu or pool layer preceding it) and outputs an n dimensional vector where n is the number of classes that the program has to choose from. For example, if you wanted a digit classification program, n would be 10 since there are 10 digits. Each number in this n dimensional vector represents the probability of a certain class. The way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high level features) and determines which features most correlate to a particular class. Basically, a fc layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.

CNN architecture of the project:

	Layers
1	Conv 32 filters of 5x5
2	Pool (max) with stride 2
3	Relu
4	Conv 32 filters of 5x5
5	Relu
6	Pool (average) with stride 2
7	Conv 64 filters of 5x5
8	Relu
9	Pool (average) with stride 2
10	Fully Connected of size 64
11	Fully Connected of size 33
12	Softmax

Object localization:

Algorithm we are following for object localization -

1. Make a window of size much smaller than actual image size. Crop it and pass it to convnet (CNN) and have convnet make the predictions.
2. Keep on sliding the window and pass the cropped images into convnet.
3. After cropping all the portions of image with this window size, repeat all the steps again for a bit bigger window size. Again, pass cropped images into convnet and let it make predictions.

At the end, you will have a set of cropped regions which will have some object, together with class and bounding box of the object.

Milestones:

Timeframe	Delivery
Day 1-2	Image scraping and storing on hard disk
Day 3-8	Model building, training
Day 9	Designing pipeline & deployment of model on cloud
Day 10-13	Performing analysis & building web app

References & sources:

1. Tensorflow – https://www.tensorflow.org/tutorials/layers
2. Deep learning for logo recognition – https://arxiv.org/pdf/1701.02620.pdf
3. Guide to understanding convolutional neural network – https://adeshpande3.github.io/a-beginner%27s-guide-to-understanding-convolutional-neural-networks/
4. Logo detection using yolo - https://medium.com/@akarshzingade/logo-detection-using-yolov2-8cda5a68740e
5. An intuitive explanation of convolutional neural network – https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/