Deep Learning Final Project: Human Recognition

Lingfeng Li

April 30, 2019

Abstract

The goal of this project is to demonstrate basic understandings of Deep Learning and the ability to apply skills learnt in Vanderbilt Deep Learning class to applications and research. The report focuses on some basic concepts as well as some basic design and model tuning methods that are helpful for building a basic human detecting application for photographs.

1 Introduction

Along with the massive advancement of computer power and the development of software infrastructures such as TensorFlow and Keras in recent years, computer vision has become a buzzword and a popular sub-trend of artificial intelligence. Nowadays, advanced computer vision applications are able to classify different objects over different categories with high accuracy; some are even able to do classification in real time with high reliability that make self-driving and unmanned vehicles commercially applicable.

This project focuses on the methods of building a basic human detection application with limited resources and computing power. More specifically, a window of fix-sized will be trained to do linear regression by studying sample data with and without upright human(s) in the picture. Then the window will be shifted horizontally and vertically and summarize the result. If a human is detected, a red box will be placed on the area where the regression shows positive outcome.

Because the fix-sized window will be shifted all across the image, usually more than a dozen detections will be run for a single photograph. So for an image with no humans, any wrong regression result (a positive signal) will generate a false detection, thus a model averaging of size 3 is employed in the project in order to increase accuracy for images with no humans.

2 Related Concepts

The following concepts are essential to Deep Learning and are employed in the creation of this project.

2.1 Neural Network

Fully Connected Layer

Fully Connected Layer is widely used in deep learning. It assigns a parameter to each input and tries to summarize features from all inputs. It is very powerful; however, with a large input size, i.e. a high resolution picture, the number of parameters to learn will increase drastically.

Convolutional Layer

Convolutional Layer is widely used for spatial feature generalization. Convolutional Layer generalizes spatial features of input across the size of a filter kernel. With different filter kernels, different features are generalized, based on the assumption that important features may appear multiple times on the key object to be learned. It is different from Fully Connected Layer in that Convolutional Layer disregards the size of the input and each neuron only learns the parameters of the filter, thus creating sparse connections across different inputs.

2.2 Techniques Useful in Neural Networks

2.2.1 Feature Scaling[1]

Deep Learning Algorithm takes into account the magnitude of input data. Thus, data with different scales are treated with implied importance based on their range. To minimize such discrimination and possibly speed up training, various techniques can be employed to bring different features to similar range.

Minibatch Gradient Descent

Minibatch Gradient Descent is the combination of Batch Gradient Descent and Stochastic Gradient Decent. It is used to avoid local minima and let the algorithm to converge to the global minimum faster.

Dropout[2]

Dropout technique is implemented with dropout rate and computation of dropout mask in order to combat overfit of models. A mask is computed based on the dropout rate and generates a matrix of the same size as each layer with each element equal to either zero or one. It shuts down a neuron if the corresponding element in the layer equals zero, thus at each iteration, only a subset of the model is trained and updated.

Exponentially Weighted Moving Average[2]

EWMA is a technique to compute the average of a variable over its past values with much less memory requirement than traditional methods. It is widely employed in Stochastic Gradient Descent in order to expedite the convergence.

Adaptive Learning Rate[2]

Some parameters are sensitive to changes while others aren't. Adaptive Learning Rate

is employed so that learning rate is scaled inversely with respect to the square root of EWMA of historical squared values. It helps achieve greater progress in the more gently sloped directions and reduces the noise at the end of the learning.

Adam Algorithm

Adam Algorithm is a state-of-the-art backward propagation algorithm that uses exponentially weighted moving average for both adaptive learning rate and momentum with bias correction for EWMA.

3 Design Choices for Human Detection

The training and testing data set used to create and test the models for this project is the INRIA Person Dataset [3]. It contains 1218 and 452 various sized pictures without human (negative pictures), and 2416 and 1132 pictures with upright humans of 64 pixel by 128 pixel (positive pictures) for training and testing respectively. To fully utilize negative pictures, each picture is cropped into 64 by 128 pixel subsets with no overlap.

Since the task is to discern pictures with and without people, the error metric is selected to be accuracy. More specifically, number of correct classification over total number of testing data. Since the model will run multiple times on a single high resolution picture to generate a result, accuracy for individual regression is crucial to this project. Thus, the target value for minimum accuracy is 95%.

4 Overview of Detection Methodology

4.1 Models Creation and Testing

To ensure that minimum error would occur while running the application, three slightly different CNN models are used and model averaging is used. Each model draws inspiration from VGG16[4], and a similar layout is used. Each model uses multiple layers of Convolutional Layers and max pooling of size 2 by 2 until the size comes down to 8 by 16. Then fully connected layers with dropout are used to draw conclusion. Parameter normalization is applied to each layer, and all output units are sigmoid and all activation functions are leaky ReLU.

To feed data to the model, the concept of minibatch gradient descent is employed. Data contains both positive images and negative images, with all data scaled by 1/255. For each iteration, the sample is shuffled randomly and then divided into minibatches. Each minibatch is then fed to the model until no more minibatches are left. At the end of each iteration, parameters will be applied on the model over a validation set and validation accuracy will be calculated and displayed. Parameters are then stored to the parameter list for further use.

Ideally, the stored parameters will be put to test in the testing set and the one set with highest accuracy will be stored on the disk. The order of which set of parameters should be fed with testing data is based on the vali-

dation accuracy in the training session.

4.2 Application of models

Due to the fact that the model created can only detect if there is an upright human in the fixed 64 by 128 window, images need to be processed in order to make detection functional. Each image need to be resized and analyzed multiple times. During the process, if more than half of the models vote that there is a human appearing in any of the sub images, the area identified with a human will be highlighted with a red box. A more detailed illustration of model is as follow:

Algorithm for upright human detection:
|Load Models
|Initiate imageRatio to 1
|Set window dimension: 64x128 pixel
|Align window with top left of image
| #Resize image to appropriate size
|Do until imageWidth $<$ 64 pixels OR
|imageHeight $<$ 128 pixels:
| Multiply imageRatio by 1.5
| Shrink original image by imageRatio
| #Analyze image by shifting window
|Initiate loopCount to 0
|Do until loopCount equals 2:
| Divide imageRatio by 1.5
| Shrink original image by imageRatio
| Do until window hit bottom border:
| Shift window 32 pixels right
| Do until window hit right border:
| Shift window 64 pixel down
| Record position, analyze window
| Increment loopCount by 1

5 Experiments and Results

Models can achieve on average around 95% testing accuracy after early stopping. When combined, the model can achieve good results when recognizing photos. Most pictures with upright human figures, even in very small scale, are able to be identified. However, when the detector encounters images with strong vertical features, the model will tend to make mistakes.

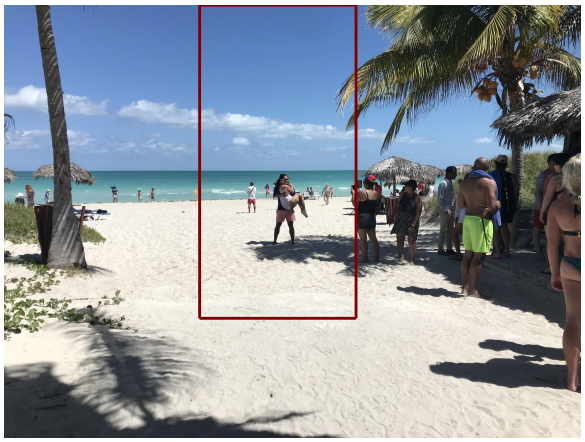

Figure 1: Model can detect human figures in small size

Figure 2: Model classifies a monument as a human

6 Analysis and Conclusion

The application can usually correctly identify human figures if they exist in images, while tending to make wrong decisions when there are strong vertical linear features in the images. This is due to the fact that the fix-sized window will be shifted all across the image, so usually more than a dozen detections will be run for a single photograph. So for an image with no humans, any wrong regression result (a positive signal) will generate a false detection, thus the accuracy expectation for images with no human features suffers.

However, there are a few measurements to be done in order to improve the performance of the detector. After some tweaking to minibatch size to each model, very interesting results can be found. The discussed detector (loose detector) uses a batch size of 512 samples. If the batch size is decreased, models can achieve much higher testing accuracy using the same testing set, around 98%. However, the models will detect humans with stricter terms. For example, much fewer segments are classified as human. Even though some of the false human detection can be avoided, some human figures will not be detected as well. In the detector created with minibatch size of 64 (strict detector), the incorrect detection of the monument as a human mentioned in previous section can be avoided; however, no human on the beach is detected either.

Some other measurements can be applied to the project to improve accuracy as well. No data augmentation methods are applied in this project so far. Some common methods such as adding tiny amounts of white noise to all data sets, shifting hue, and affine distortion could more or less make improvement upon the results. In addition, the dataset doesn't contains many non-human vertical features, such as trees, doors or buildings. In order to counter the tendency of classifying vertical objects as humans, adding more non-human vertical features can invariably help the generalization.

7 Reference

1. Asaithambi, Sudharsan. "Why, How and When to Scale Your Features." Medium, GreyAtom, 4 Dec. 2017, medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e.

2. Koutsoukos, Xenofon. "Deep Learning" Vanderbilt University. 29 April 2019. Lecture

3. Dalal, Navneet. "INRIA Person Dataset." INRIA Person Dataset, 2005, pascal.inrialpes.fr/data/human/.

4. Hassan, Muneeb. "VGG16 - Convolutional Network for Classification and Detection." VGG16 - Convolutional Network for Classification and Detection, 21 Nov. 2018, neurohive.io/en/popular-networks/vgg16/.

8 Link to Project

The dataset, all source code and pre-trained models can be found HERE.