Convolution Neural Network Approach to Diagnosing Diabetic Retinopathy

José Solomon *

Abstract

This is the first draft of the project report of the diabetic retinopathy diagnosis using convolution neural networks (ConvNN). It is derived from the Kaggle Diabetic Retinopathy [1] aimed at creating a robust algorithm to automate the process of classifying the level of diabetic retinopathy found in retinal scans. the purpose of this initial draft is primarily to describe how the ConvNN work, and the inter-dependencies of its various functional elements. In the final draft, details of the implementation will be given and various resulting performance metrics will be reviewed.

Contents

1 Introduction

There are two key facets to the project: first is the concept of the diabetic retinopathy and it's diagnosis; the second is ConvNN and it general functional principles as applied to digital imaging processing and categorization. We begin with a cursory description of the former, and then focus the bulk of the theoretical discussion on the latter.

*jose.e.solomon@gmail.com

2 Diabetic Retinopathy

Diabetic retinopathy (DR) is a disease that generally afflicts those who have dealt with diabetes for a period of 5 years or more [2]. It is defined specifically as the deterioration of blood vessels that feed the retina of the human eye, as illustrated in Figure 1 below.

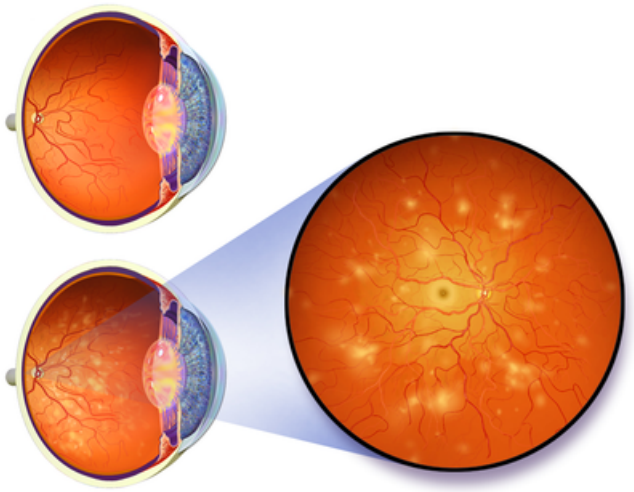
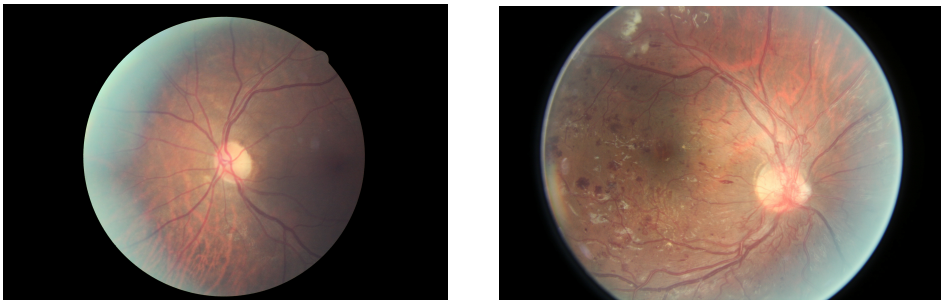


Figure 1: Diabetic retinopathy in comparison to a normal eye [3]

The primary cause of DR is that as a byproduct of the condition of diabetes, thinner blood vessels in the human body tend to form abnormal branching structures and thinning walls. This leads to either deteriorated blood supply to the effected areas or actual hemorrhaging. In terms of the retina, this leads to blind spots forming in the field of vision of the afflicted individual. To diagnose the condition, a retinal scan of the eye is taken and a classification is assigned depending on the number of abnormal vein branching, wall thickness of the blood vessels and blood stains observed. DR is categorized on a scale from 0 to 4, and examples of level 0 and 4 DR are given below. It should be noted that this method of categorizing DR is not fundamentally rigorous in nature and is more of a qualitative scale.



(a) Level 0 DR (b) Level 4 DR

Figure 2: Examples of level 0 and level 4 DR

2.1 The Kaggle Grand Challenge

Kaggle has presented the automated DR diagnosis as open grand challenge. In doing so, Kaggle provided 35 gigabytes of retinal scan data, over 35,000 images, thus each image is roughly 1.5 megabytes in size. To facilitate download and processing of the data set, Kaggle divided the image set into a training group, and a testing group, and the subsequently divided each group into smaller subsets to facilitate file transfer. All the results presented here and in subsequent discussions are based on the first training set of images.

It is noted that the aim of the current work is to understand ConvNN from a fundamental perspective, and not to actually create a competitive solution for the grand challenge. Due to the size of the reference data set, and the limitations imposed by Python in terms of loading elements into shared memory, it would be very difficult to create a code base that could compete directly with other implementations in CuDa, (a GPU-level system language), which is defacto favorite language for image processing.

3 Convolution Neural Networks

ConvNN are a powerful machine learning technique that fall under the of general premise of deep learning. There are number of flavors of ConvNN, but the specific implementation presented here is one of the first implementations of the technique, the LeNet-5 [4], and is especially adept at processing digital imaging.

The LeNet-5 consists of 3 primary component layers: the convolution layer, the pooling layer, the conventional fully connected layer [5].

3.1 The Convolution Layer

The convolution layer is the work horse of the ConvNN, and it is what makes it such an indispensable tool for image processing. Mathematically, the concept of convolution is somewhat straightforward, and plays a key role in spectral methods and Fourier space treatments. In terms of digital imaging, convolution can be expressed as [5] the product of the pixel intensity of a given image with that of a kernel, and is stated as

$$H[m,n] * G[m,n] = \sum_{u=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u,v]G[m-u,n-v] \tag{1}$$

This may seem somewhat complex at first, but consider that an image is an array of pixels, each with an intensity that ranges from 0 up to a given bit depth, (e.g. 255 for 8-bit images), than we can see that an image can be translated as matrix, where each element is a given pixel's image intensity. The concept is illustrated in Figure 3.

With this concept in place, a kernel, is itself a matrix of a prescribed dimension, smaller than that of the original image, an example of which would be

$$H^{L}_{3\times 3} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2}$$

136	66	145	160	142
134	131	185	141	14
178	111	116	127	127
58	55	56	72	88
50	48	49	63	76

(a) original image (b) Region of red rectangle (c) Gray-scale value

Figure 3: Digital image as a matrix [6]

This matrix is applied, via convolution, across the matrix of the image to create a filtered image which, depending on the *stride*, (i.e. number of times the kernel is applied across the original image), downsizes the original image into smaller matrix which is representative of the dot product between the original image and kernel. The full convolution concept is illustrated in Figure 6. At this point, the ConvNN is ready to pass this resulting matrix to a pooling layer

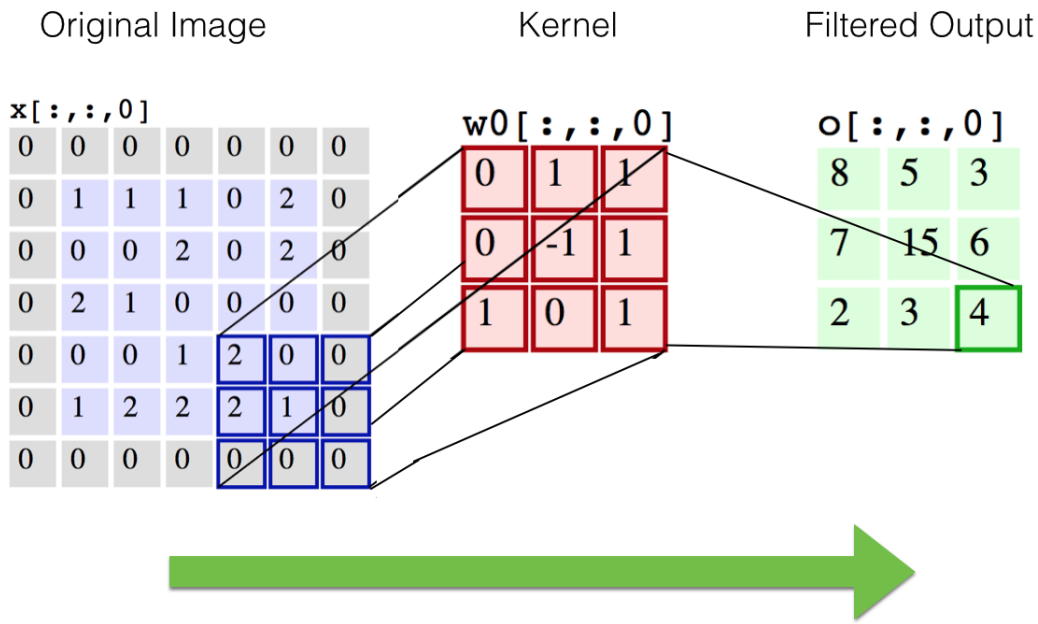


Figure 4: Convolution concept [7]

3.2 The Pooling Layer

In order to reduce the computational expenditure of the ConvNN algorithm, a pooling module is often used to reduce the required number of weights for a subsequent fully connected neural network layer.

Usually the form of the pooling is *max Pooling*, where a specific number of elements of the convoluted image are defined as common-pool members, (as illustrated in Figure **??**), and only the pool member with the maximum value is carried forward in the network. The size of a pool is usually, but not required to be, the same size as the kernel applied during convolution. By focusing on the maximum value of a convolution, the network is becoming more sensitive to those image features which are more dominant of the total feature set.

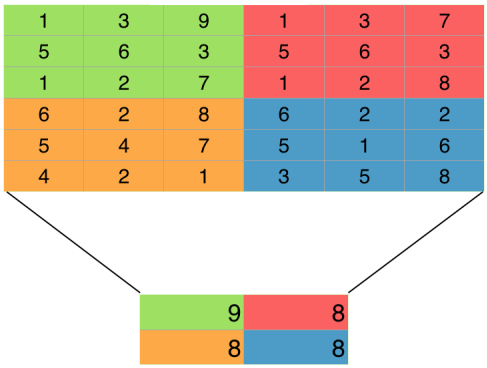


Figure 5: Max-pooling: the largest value of each sub-cell is fed forward

It is noted that a ConvNN often uses a series of convolution and pooling layers to continuously reduce the computational load of the final layer, or layers, of fully connected NN, which is normally the most computational expensive facet of the ConvNN pipeline.

3.3 The Fully Connected NN Layer

The fully connected NN layer is actually what a *conventional* neural network, which is comprised of a series of nodes that are interconnected via a series of weights and bias units. Let's begin with the illustration shown in Figure **??**

As can bee seen from the figure, each node is connected to the series of nodes in the layer directly downstream of it. The node downstream is in turn connected to each node in the layer upstream of it, as well as each node in the layer downstream.

Let's say that for each input connection of a given node, there is an associated weight w_i. Given that there is N number of nodes in the upstream layer, the input to node can be seen as

$$f(\vec{w}^T\vec{x}) = f(w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_N \cdot x_N) \tag{3}$$

We note that the dot product shown above is the input to a function, known as the activation function, that in turn is the input value of the downstream node. There are a

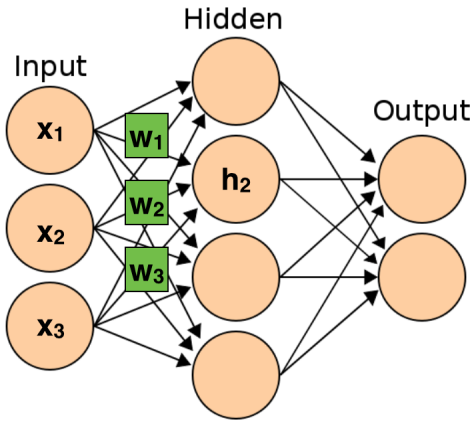


Figure 6: A fully connected neural network [8]

variety of activation functions that can and have been used in NN. One of the most popular kind, and the one which is used here, is the *tanh* activation function, which is defined as

$$f_{\tanh}(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{4}$$

which has an activation profile shown in Figure **??** As can bee seen, the function has asymptotes at 1 and -1 and exhibits positive values in y for positive values of x. Another common activation function is the sigmoid activation function, which has a similar profile as *tanh* and has the functional form

$$f_{sigmoid}(a) = \frac{1}{1 - e^{-a}} \tag{5}$$

The sigmoid activation function has asymptotes at 1 and 0, and is centered at the origin in x and at 0.5 in y.

These two functions are very similar in profile, but the *tanh* function is more commonly used due to its favorable back-propagation properties, (i.e. the learning process of the NN). Back-propagation is a key concept to NN and ConvNN, but its description is beyond the scope of this first draft and will most likely be discussed in the final draft of the current document.

3.4 Logistic Regression

4 Current Project Status

References

[1] Kaggle. https://www.kaggle.com/.

[2] NIH National Eye Institute. https://nei.nih.gov/health/diabetic/retinopathy.

[3] Diabetic Retinopathy Wiki Entry. http://en.wikipedia.org/wiki/Diabetic_retinopathy.

[4] Bottou L. Bengio Y. LeCun, Y. and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 11(86):22782324, 1998.

[5] Theano Convolution Network. http://deeplearning.net/tutorial/lenet.html.

[6] IEEE Computer Society. http://www.computer.org/csdl/trans/tc/2013/04/ttc2013040631-abs.html.

[7] Stanford Computer Science Dept: Class 231 -Convolution Neural Networks. https://cs231n.github.io/.

[8] wikiBooks: Artificial Neural Networks. http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Print_Version.