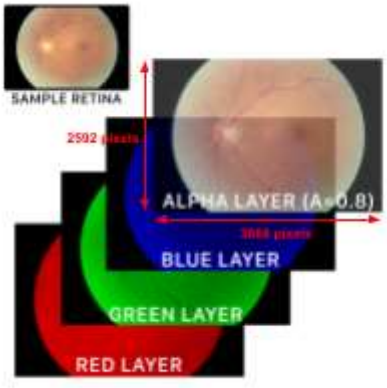
Nicholas Harty and Aum Dhruv
Fort Myers High School
Regional Science & Engineering Fair
28 December 2021

INTRO:

Millions of people across the globe suffer vision complications as a result of diabetic retinopathy, especially due to the lack of appropriate testing as to the severity of the condition. Diabetic retinopathy is a complication caused by an existing history of diabetes/diabetic symptoms that can slowly deteriorate a person's vision and can even lead to partial blindness. Often this diabetic complication can be treated by timely management of the condition and/or modern laser eye treatments/surgery. However, this treatment of diabetic retinopathy cannot completely cure the disease and, when left untreated, its effects seemingly take on worse results. These severe cases are often seen in developing nations where general access to medical testing is bleak with most patients unsure of their severity until physical implications become apparent. In fact, 79% of the adults with diabetes reside within low-to-middle income nations and serve a major risk of joining the estimated 93 million people worldwide suffering from diabetic retinopathy. Without significant medical testing, which can often prove expensive, and local ophthalmologists, this population is the most at risk of developing blindness and suffering the consequences of vision imparity. In this sense, the recent development of low-cost neural networks as a means of detecting early stages of diabetic retinopathy has become a much-needed solution to centuries of unknown suffering as a result of this complication. However, such solutions must be affordable/accessible as well as accurate in their results. This is the basis for the researcher's study and the reason for such urgency when it comes to verifying solutions. The focus of the study relies upon the two major post-processing algorithms, specifically within the realm of image classification, that have proven suitable for low-cost, efficient medical testing: the convolutional neural network algorithm and the k-nearest neighbors algorithm. Research into the function of these algorithms, as well as their applications, is important in reducing the need for physician training and increasing the mobility of worldwide medical testing.

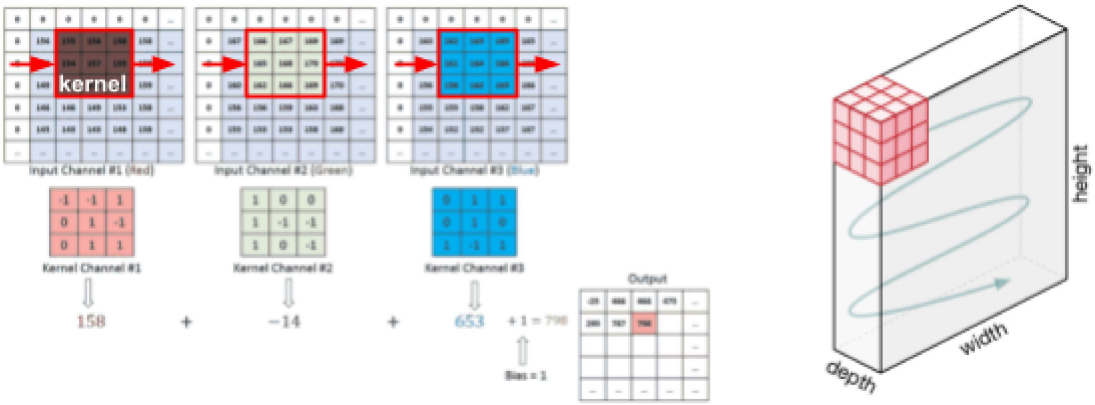
CNN:

The first of these recognition methods, the convolutional neural network algorithm, works by pooling various elements of a training image, assigning importance/weight to various aspects of an image, and then undergoing a convolution process that exponentially decreases the size of the image into basic constraints. Its ability to break down the various prominent features of media allows it to function as an essential component in the classification of bitmaps, moving graphics, and audible



samples. In this study, the description of this algorithm will focus on image classification as it relates to the entirety of the neural network's functionality. This entire process mimics the human brain's (or rather a human's Visual Cortex) ability to discern the components of an image and contrast the features of two corresponding images when dealing with necessary classification. This system of assigning importance to various features of an image is described as the CNN. Initially, this system breaks down sample training images into their essential channels: the Red Layer, the Green Layer, the Blue Layer, and, in some cases, the Alpha/Intensity Layer. This breakdown is a result of each pixel, in a standard training image, being composed of three RGB values between 0-255 in intensity and an alpha value between 0-1 in opacity. From this point of divergence, the algorithm begins dissecting the image for features through a series of four scientific steps defined by most computer analysts as essential "layers". These layers are composed of the Convolution Layer, the Pooling Layer, the Rectified Linear Unit (ReLU) Layer, and the Fully Connected Layer.

The first layer of CNN, the Convolutional Layer, processes the training image through a variety of kernels, also referenced as filters, that allow edges, corners, lines, shadows, and other elements of the image to become apparent. For each of these kernels, there is an output of a 2D map/matrix which outlines the intensity of each individual pixel within the filtered photo. Each channel (RGBA), as mentioned previously, is analyzed by the kernel. The kernel functions are a system that moves across an image and analyzes a specific area for features. For example, a 5x5 pixel image may have a 3x3 pixel kernel that moves across the image horizontally, starting from the left side, one pixel at a time analyzing each 9-pixel area for new features. Once finished with its horizontal motion, the kernel moves down one pixel and begins, once again, from the left side of the image. All throughout this process, these intensities/weights of these features within individual kernels are recorded in a stored model on the available RAM. Once the entire image has been outlined by kernels, the kernel either moves to another channel or, if all channels are covered, finishes its analysis by computing any additional biases provided by the investigators. Following this layer of computation, CNN shifts to the Pooling Layer.



Source: Saha, S. (2018, December 17). A comprehensive guide to convolutional neural networks-the eli5 way. Medium. Retrieved December 29, 2021, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

The second layer of CNN, the Pooling Layer, functions with the purpose of reducing the size of the image in order to decrease the amount of computing power necessary for convolution

and to extract further features at a compressed level of kernelization. Similar to the Convolutional Layer, the Pooling Layer uses its own kernel, with the exact constraints of the convolutional kernel, to simplify the training image. Using the previous example of a 3x3 pixel kernel analyzing a 5x5 pixel training image, each of these 9-pixel area kernels would represent 1-pixel on the resulting image. This revelation means that this 5x5 pixel training image would be reduced to a 3x3 pixel resultant image after undergoing the pooling process. CNN is able to reduce these larger kernels into single pixels by either utilizing the processes of "Average Pooling" or "Max Pooling". Average Pooling takes the numerical average of all of the individual RGBA values within a kernel and uses that average to compose the RGBA of a new corresponding pixel. On the other hand, Max Pooling provides the dimensional compression provided by Average Pooling while also applying a layer of noise reduction to the resulting pixel. This process of Max Pooling helps the reduced training image maintain aspects of the original image and generally performs with greater accuracy when compared to Average Pooling CNN models. After the completion of the Pooling Layer, the CNN continues to cycle through the Convolution-Pooling layers until the necessary features are extracted and the original training image and been broken down into its essential pixels. The continuous nature of this cycle is determined by the number of cycles that are predefined by the investigators. In general, the accuracy of the model, when tested against its own training dataset, increases as it maintains further cycles. After this defined period of cycling between the Convolution and Pooling layers has ended, the CNN progresses into the ReLU and Fully-Connected Layers.

The Rectified Linear Unit (ReLU) Layer works in conjunction with the previously assessed data in order to "flatten"/familiarize the data before passing it on to the final layer. In the scope of image recognition, the ReLU Layer serves as a final simplification of pixels in a set linear matrix (1 pixel per training image that resulted from Convolution-Pooling Layers). From this layer, this linear pixel matrix is processed through the Fully-Connected Layer.

The final layer of CNN, the Fully-Connected Layer, processes this new linear matrix as well as the recorded weights from kernelization by cycling through a series of epochs, defined by the investigators, that helps increase the classification accuracy of the model. These epochs train the model on its existing training images which helps it distinguish dominant features from each individual classification group. This prolonged training of CNN's ability to discern between defining features is an asset used within image classification and the proficiency of trained data/images can often correlate directly to the assumptions of the algorithm when given new data/images. This later classification is refined using the "SoftMax Technique". This technique helps when identifying low-level features and refining the necessary recorded weights/biases from the Convolution Layer. Resulting from the classification of new images through "SoftMax" classification, the investigators should receive corresponding confidence intervals for each of their predefined classification groups.

```
CNN Results (Ex: Diabetic Retinopathy Severity):
      (0): 0.02       (1): 0.945      (2): 0.005
      (4): 0.015      (5): 0.015
```

For example, in the resulting dataset listed above, the investigators could assume, with 94.5% accuracy, that the testing retina image had mild symptoms of diabetic retinopathy based upon the training CNN.

KNN:

The second of these algorithms, the k-nearest neighbors algorithm, is trained on various groups of images in order to classify a given unknown image. Its simple implementation allows it to function as an essential component in image recognition, video recognition, and speech recognition. KNN is used for classification and regression. In this study, the description of this algorithm will focus on image classification as it relates to the entirety of the neural network's functionality. This process uses a predetermined dataset, such as "MobileNet", to depose images into a series of 1000 logits (vectors of raw predictions that a model generates). This can also be done by developing a rudimentary algorithm to break down images via kernaling or random pixel sampling, into the necessary logits. To elaborate, for each group of images, a matrix is created of shape [n, 1000], in which "n" is the number of samples per group. MobileNet deposes new images into logits, which are then normalized to unit length. This means that the logit vectors are scaled to unit vectors using the formula below.

$$u = \frac{v}{\|v\|}$$

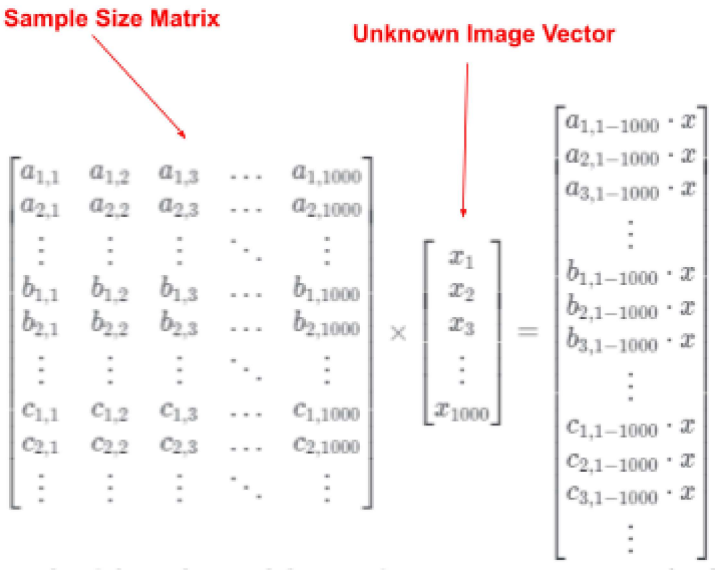
Following this, the logits are concatenated to the end of the matrix, forming an additional row.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,1000} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,1000} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{1,1} & b_{1,2} & b_{1,3} & \dots & b_{1,1000} \\ b_{2,1} & b_{2,2} & b_{2,3} & \dots & b_{2,1000} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{1,1} & c_{1,2} & c_{1,3} & \dots & c_{1,1000} \\ c_{2,1} & c_{2,2} & c_{2,3} & \dots & c_{2,1000} \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

Source: Thorat, N. (2018, June 20). *How to build a teachable machine with Tensorflow.js*. Observable. Retrieved December 29, 2021, from https://observablehq.com/@nsthorat/how-to-build-a-teachable-machine-with-tensorflow-js)

The matrix ends up looking similar to the image above.

When given an image with no existing classification, MobileNet deposes it into logits, then normalizes to a vector of shape [1000]. After this, the unknown image's vector is multiplied to the matrix.



Source: Thorat, N. (2018, June 20). *How to build a teachable machine with Tensorflow.js*. Observable. Retrieved December 29, 2021, from https://observablehq.com/@nsthorat/how-to-build-a-teachable-machine-with-tensorflow-js)

The equation ends up looking similar to the image above. The resulting vectors are sorted by distance from the unknown image in increasing order. The closest vectors are called the "nearest neighbors", and the K value determines the extent to which "neighbors" are taken into account when classifying the unknown image. Hence, the algorithm's name is the k-nearest neighbors algorithm.

Concluding Comparison Between Neural Networks:

The main difference between the convolutional neural network algorithm and the k-nearest neighbors algorithm exists in their ability to discern between distinct features of an image. The convolutional neural network algorithm processes feature regardless of spatial orientation because of its utilization of kernels that process training images while the k-nearest neighbors algorithm identifies features based upon color/intensity and takes spatial orientation into account when classifying testing images. Another difference exists in their individual training processes. The k-nearest neighbors algorithm trains upon its sample dataset by conforming the data into logit-based matrices which requires little computational power as it simply is building a record/ledger for later classification. On the other hand, the convolutional neural network algorithm uses higher computational power in its training process as it cycles through multiple Convolutional Layers as well as the refinement of its feature identification across a series of predefined epochs. Although these two algorithms are mathematically distinct in almost every component of their individual function, their output, in terms of correlation, are the same and

they share similar efficiency when tasked with the training/classification of bitmapped medical datasets.

Additional Sources Used:

Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A., & Tan, R. S. (2017, August 24). A deep convolutional neural network model to classify heartbeats. Computers in Biology and Medicine. Retrieved December 29, 2021, from https://www.sciencedirect.com/science/article/pii/S0010482517302810?casa_token=ufM7dWVQvzIAAAAA%3A6Q7840z1UyqkEDSc5HRGebPpdSi5el09lxikoaeQFj8SVlkSGdhFVaoG7bjfvYfMl7LgaxmaBj0

Peterson, L. E. (n.d.). *K-Nearest Neighbor*. Scholarpedia. Retrieved December 29, 2021, from http://www.scholarpedia.org/article/K-nearest_neighbor

Saha, S. (2018, December 17). A comprehensive guide to convolutional neural networks - the eli5 way. Medium. Retrieved December 29, 2021, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Thorat, N. (2018, June 20). *How to build a teachable machine with Tensorflow.js*. Observable. Retrieved December 29, 2021, from https://observablehq.com/@nsthorat/how-to-build-a-teachable-machine-with-tensorflow-js)