**Filter Size Analysis for Facial Recognition with CNN’s**

John Robinson

Mentored by Dr Girard

Shippensburg University

jr8864@ship.edu

# **ABSTRACT**

This particular research is an exploration of artificial intelligence and deep neural networks in relation to image recognition. The foundation and basis for starting this research is the wide number of applications as well as the potential of this field. Before diving into understanding the facial/image recognition element there must be understanding of the building blocks that support it.

**1. Introduction**

An introductory delve into the world of long term research related to computer science. This research paper is a collection of acquired knowledge related to convolutional neural networks and the construction of such a network and the results of that experiment. No previous work of any note, and the contributions are from the author and the mentors

**2. Neural Networks**

The baseline to take into account is Artificial Neural Networks. “ANNs are processing devices (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the mammalian cerebral cortex but on much smaller scales”(shipyard). Neural networks are usually organized into layers which are made up of interconnected nodes which contain an activation function. Patterns are presented by an input layer which communicates to hidden layers which process the input. This is connected to an output layer. ANN’s are trained by feeding large quantities of data. Training at a basic level is providing input to ANN and then telling it what the input should be.

To break it down a neural network is a number of artificial neurons called units separated by a series of layers. The units are separated into different specific types. The first being input units, input units are for receiving some sort of information that the neural network will process. There are also output units which are what the neural network does with the information it processes. What's in between these two layers are the previously mentioned layers which are also units, called but they are hidden units. Another concept to consider is that most neural networks are fully connected which means every hidden unit and output unit is connected to every unit on either side of it.

(Woodford, 2020)(Kailash, 2017)

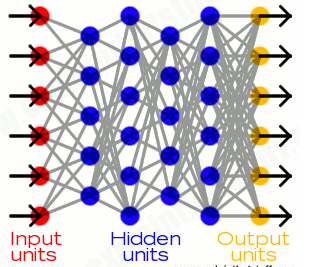


Figure 1 (Woodford, 2020)

The connections between units are related to what's called weight, which is a number that is assigned randomly and used to adjust the output of a unit before it reaches the next unit. The larger the weight the more influence the units have on each other. This can either be negative or positive, whether it is one or the other determines how it influences the units. The process of actual learning is normally done by feeding an input which is processed by the hidden units and then arrives at the output layer, this is all a process called forward propagation. The way it is processed is “Each unit receives inputs from the units to its left, and the inputs are multiplied by the weights of the connections they travel along. Every unit adds up all the inputs it receives in this way and (in the simplest type of network) if the sum is more than a certain threshold value, the unit "fires" and triggers the units it's connected to (those on its right).”(Chris Woodford). This is illustrated by Figure 3 on the next page.

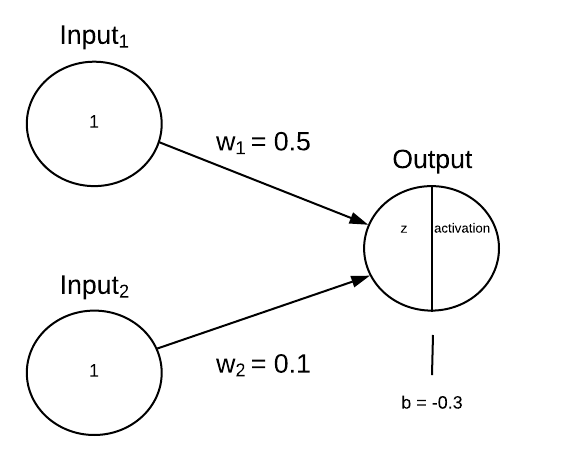


Figure 2 Tóth, B. (2018, August 13)

The simplest way to explain the math behind neural networks is to explain how the Perceptrons neural network are calculated.“Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks.“Sharma, S. (2019, October 11)To start the input layer will input a number of values. Based on the number of inputs, there is the same number of weights for each respective input. The weight is then multiplied by the input values and then the sum is added with a bias value which is usually always one as it is mostly there to ensure neuron activation if the inputs are all zero. The bias is usually randomly assigned much like the weight.(Woodford, 2020)(Kailash, 2017), Sharma, S. (2019, October 11)

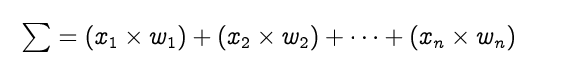




Figure 3 (Saravanan, 2020)

In some cases activation functions are performed to introduce non linearity to the neural network. In this case the sigmoid activation function is used, this can be seen in figure 4 the top left side. Others can be used, but this is the one being used in the example. The result of the previous formula for z is entered into the activation function and the result is the predicted value and end result for forward propagation when it comes to a Perceptron.

(Woodford, 2020)(Kailash, 2017), Sharma, S. (2019, October 11)

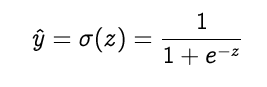


Figure 4 (Saravanan, 2020)

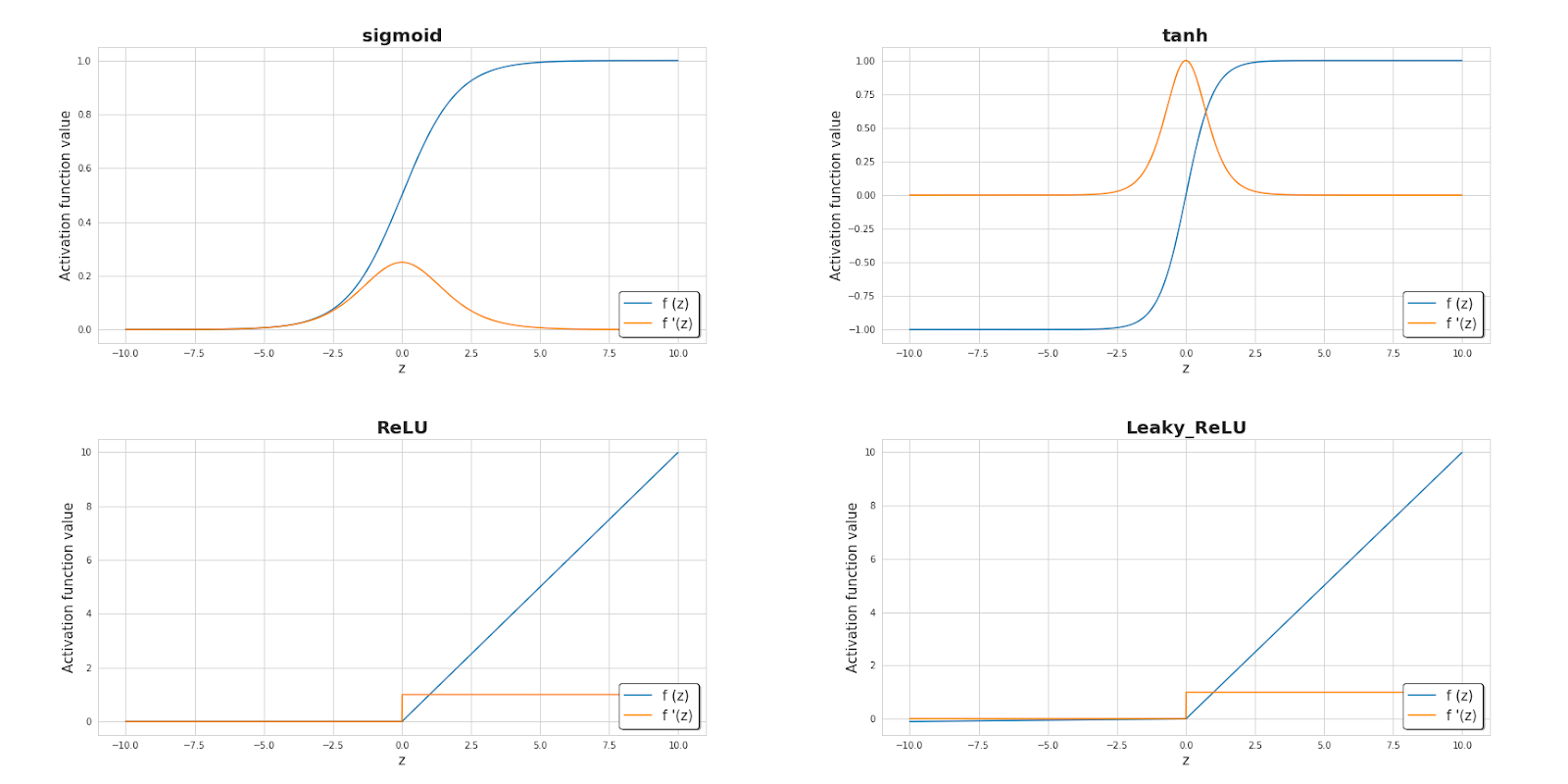


Figure 5 (Skalski, 2018)

 Doing Forward Propagation has a predicted value, calculating the error is done by comparing the predicted output with the actual output. Accounting for errors in that method is called backward propagation. Using the differences found to change the weights between the units which results in the neural networks output to become closer to the expected output, thus in a way learning. The next major part is the part that leads to error prediction. Calculating the loss function can be done in a number of ways. The gradients are how the weights are updated. The several major loss functions are mean squared error, binary cross entropy,  Categorical Cross Entropy, and Sparse Categorical Cross Entropy. The most commonly used of these is the mean squared error, this is simply the mean squared difference between predicted vs expected values which can be observed in figure 6.



Figure 6 (Deeplizard)

The total sum simply being the squared difference of every node summed. To calculate the gradient we must find the derivative of the loss in respect to the weights. So the formula would be the derivative of the loss over the derivative of the weight. This is for a single node so the weight would be for a specific node. Using the chain rule the derivation will look like figure 7.

(Saravanan, 2020) (Skalski, 2018) (Deeplizard), K., H. M. (2019, November 03), Ahirwar, K. (2017, November)

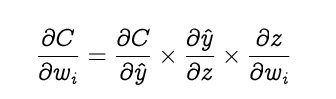


Figure 7 (Saravanan, 2020)

The derivative of the loss function over the weight of the node i equals the derivative of the loss function in respect to the activation function times, the derivative of the activation function in respect to the input for node i times the derivative of the input to node i in respect to the weight. The sum of the three derivations is how backpropagation calculates the gradients of the loss in respect to the weights of the network. Next the process of gradient descent occurs which “is an iterative optimization algorithm for finding the minimum of a function; in our case we want to minimize the error function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point”K., H. M. (2019, November 03), this is also illustrated in figure 8. So the old weight is subtracted from the arbitrary learning rate, a which is determined beforehand much like the bias, multiplied times the result of the chain rule calculation with the loss function. The result is the new weight. This is done until the local mininia is found.

(Saravanan, 2020) (Skalski, 2018) (Deeplizard), K., H. M. (2019, November 03), Ahirwar, K. (2017, November)

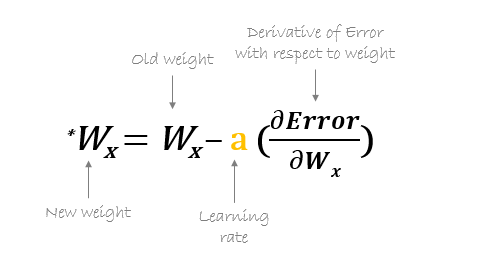


Figure 8

**3. Convolutional Neural Networks**

There are many types of neural networks, the major one to be covered is the CNN. The main reason this was chosen above others was due to its popularity for image recognition. Going in depth with Convolutional Neural Networks starts with “When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers.”(Adit Deshpande). The three at the end refers to the three channels red, green, blue. For grayscale images there is only one channel. So for example a jpg is used for an input its size is 480x480 pixels. The array will be 480x480x3. Each array position will have a value between 0 and 255. So for color images there are 3 array’s filled with values of 0-255 while greyscale will have just the one array of 0-255.

The general idea is then based on combinations of the array filled with numbers that will provide a probability that it is a certain object. ”ConvNets derive their name from the “convolution” operator. The primary purpose of Convolution in the case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data”(ujjwalkarn). The actual Convolutional layer is an analysis of the previously mentioned array’s. What occurs is a much smaller array called a filter, neuron or kernel which slides across the larger array. The section being checked against the other array is called the receptive field. As the filter slides the array values in the receptive field are being multiplied by the elements in the filter. The multiplications are then summed and put into a new result array called the feature map.(Adit Deshpande)(Ujjwalkarn, 2017)

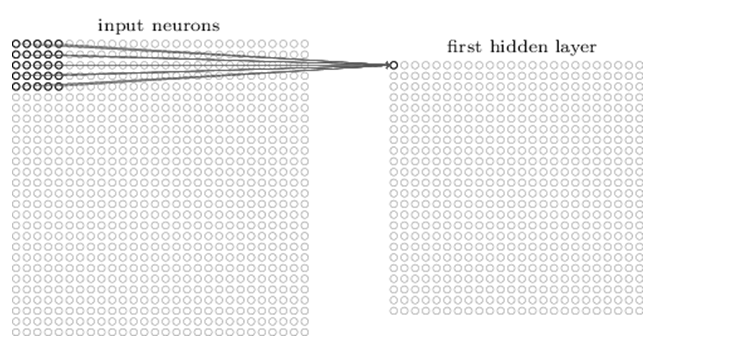


Figure 9 (Adit Deshpande)

The filter will move one position right until it hits the far right side then start again on the left one position lower than where it last checked. The process of sliding and multiplication is called Convolved Feature. The size of the aforementioned convolved feature is based on 3 parameters. The depth which determines the number of filters in use. Stride is the number of pixels which are slid over in the input matrix. Lastly, is zero padding, which is when you pad the input matrix with zeros around the border so the filter can be applied to bordering elements. Zero padding is also used to control the size of the feature map. Wide vs narrow convolution is just having vs not having zero padding. CNN’s have other features. Determining the dimensions after one pass through a convolutional layer can be determined by a simple formula. (AISHWARYA SINGH, 2020)

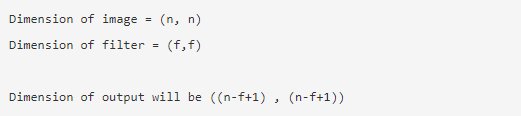


Figure 10 (AISHWARYA SINGH, 2020)

Another small feature which may be used is the rectified linear units activation function which simply is just when a negative number is swapped with a zero. “This helps the CNN stay mathematically healthy by keeping learned values from getting stuck near 0 or blowing up toward infinity”(e2eML school).

The last feature to cover is pooling which essentially is taking large images and shrinking them down while keeping the important information. The process is just a small window that goes through the input array and makes a new array with the largest value in that window. It moves just like the convolutional layer. Based on the previously mentioned mechanics a classic CNN would go through input, convolution, ReLU, convolution,ReLU,pooling,convolution, ReLU,pooling, to being fully connected. (AISHWARYA SINGH, 2020)

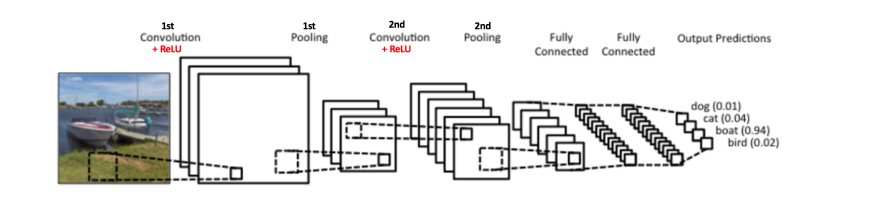
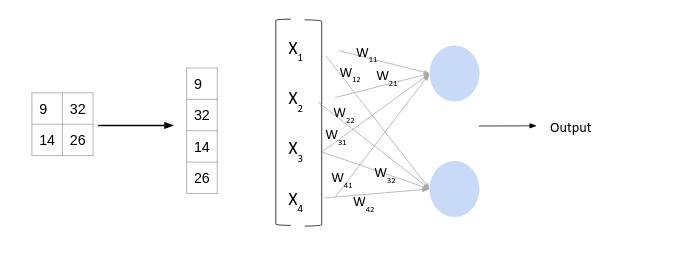
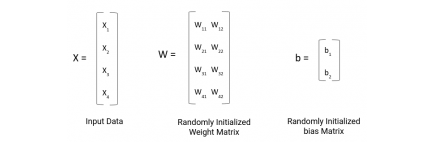


Figure 11 (Ujjwalkarn, 2017)

The fully connected layer is the last few layers of the network. “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”(Adit Deshpande). In the context of a CNN the fully connected layer takes what the previous convolution layer output, which would be an array of a certain size, that is converted into a one dimensional array. The new one dimensional array is what is sent to the fully connected layer where linear and non linear transformations occur. Linear transformation is done by the formula Z = W^T.X + b, X being the input, W is the weight and b being the bias. The weight was discussed in the first section; it changed during the backpropagation when the neural network is trying to find a specific value. The bias is a constant that is mostly there to guarantee that even when all inputs from layers are zero there is still neuron activation. So as shown below the input array, weight array and bias are all condensed into an equation for the linear transformation which is the last step in a cNN’s forward propagation process. In some cases activation functions are performed during the process, the ReLU is one of these functions but there are others depending on what is trying to be achieved. (AISHWARYA SINGH, 2020)(Ujjwalkarn, 2017)





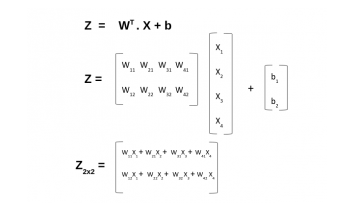


Figure 12 Aishwarya, S. (2020, May 08)

**4. Additional notes diving into CNN’s**

Application for Neural Networks for facial recognition has vast options at the current moment. CNN’s seem to be the best overall approach but it has its downsides. One big one being “they lack a very important property of incorporating any prior information [ 6 ]. When applied to expression analysis, it seems insufficient to well describe expressional images as variations such as identities add redundant noises to these features.” Xie, S., & Hu, H. (2017, January 24). This can be made up for with the concept of TI-pooling. Ti pooling is the process of making a CNN transformation invariant. At a basic level based on transformations such as rotations, scale changes, shifts or illumination changes the output will not depend on whether or not the input was transformed. This is essentially done by transforming the original image to a set of transformations. Every transformation is a parallel instance of a partial siamese network. Each instance only having convolutional and subsampling layers.

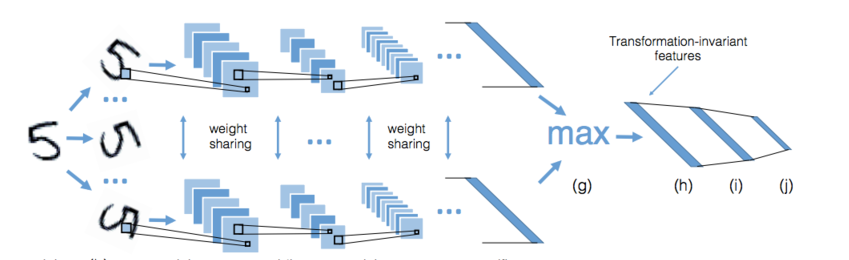


Figure 13 (Dlaptev)

The instances are passed until reaching the fully connected layer similar to normal CNN. The other important to solving the aforementioned issue is that most CNN’s use one channel. A solution discussed in Siyue Xie’s article was the use of siamese networks which uses multiple channels to enhance the power of discriminative features. These channels also use the above-mentioned Ti-pooling. To summarize, “images from the same transformation set are set as the inputs of identical parallel channels. Then, each image passes through the convolutional module and yields a concatenate feature vector. Weight sharing is implemented among all the channels. TI-pooling is implemented across all these channels” Xie, S., & Hu, H. (2017, January 24)(Dlaptev)Laptev, D., Savinov, N., Buhmann, J., & Pollefeys, M. (2016, September 22)

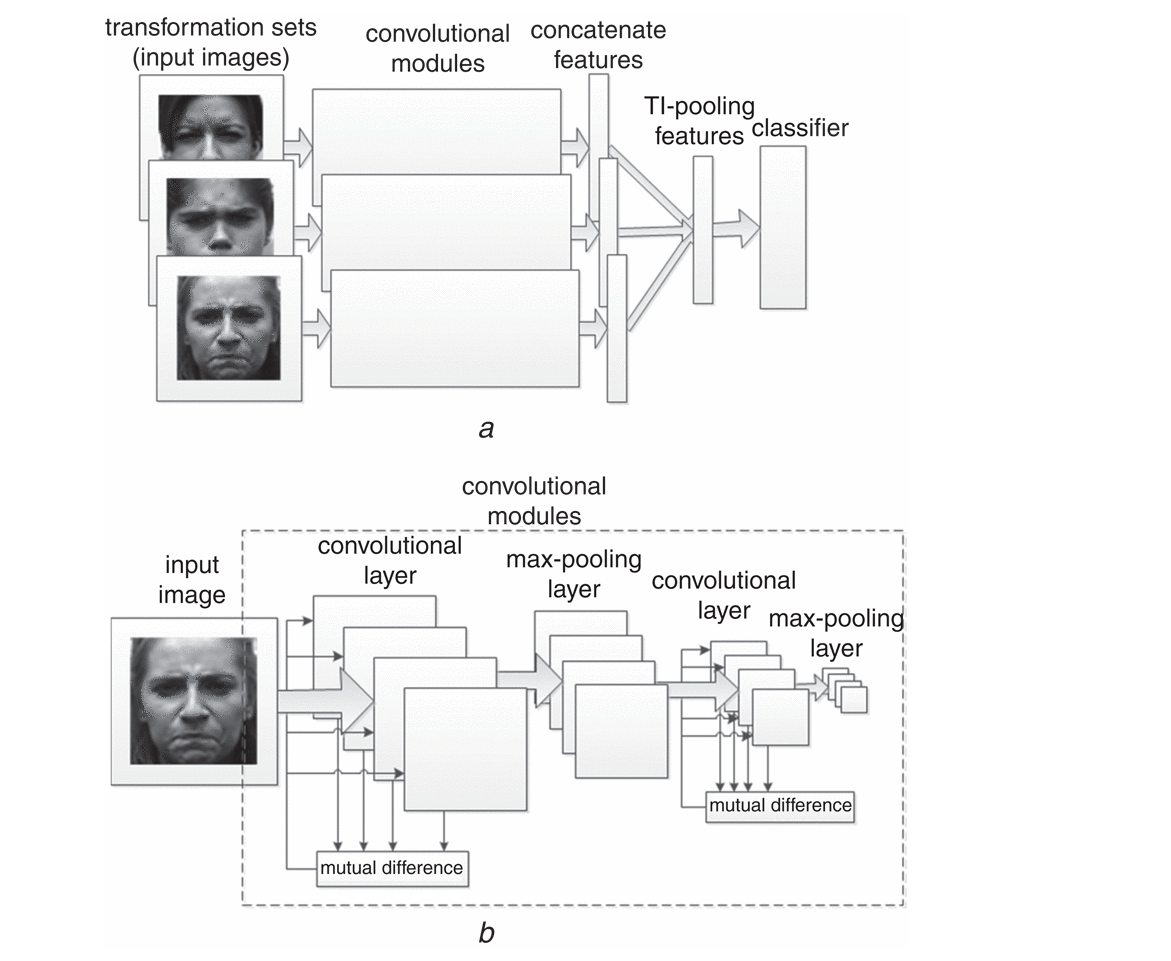


Figure 14 Xie, S., & Hu, H. (2017, January 24)

Another concept is the geometric feature based networks. This is another type of CNN that in relation to facial recognition “captures the movements of the landmarks of emotion. The feature of the partial elements obtained by detecting the movement of the landmark is added to the overall features so that more robust features can be extracted”(Ji-Hae Kim, 2019). The main part of gCNN is the process of capturing dynamic changes in facial expressions and extracting landmarks in the face. An example would be the eyebrows, nose ,eyes etc… Xie, S., & Hu, H. (2017, January 24)

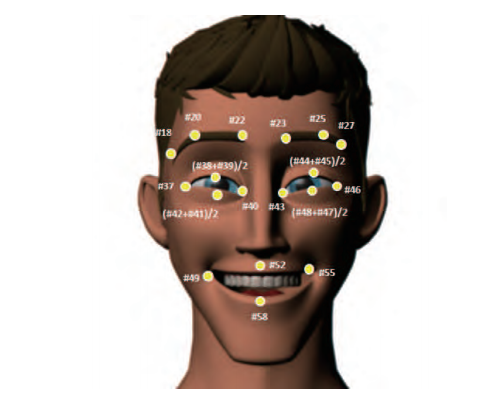


Figure 15 Kim, J. (2017)

gCNN are similar to standard CNN’s with the exception that they are involved with surface data. gCNN architecture is made up of input data layer, mesh convolutional layers with data reshaping, batch normalization layers, ReLU layers, mesh pooling layers and lastly fully connected layer. This goes deeper than an experiment will likely go, but it is a possible avenue. Kim, J. (2017) Xie, S., & Hu, H. (2017, January 24)

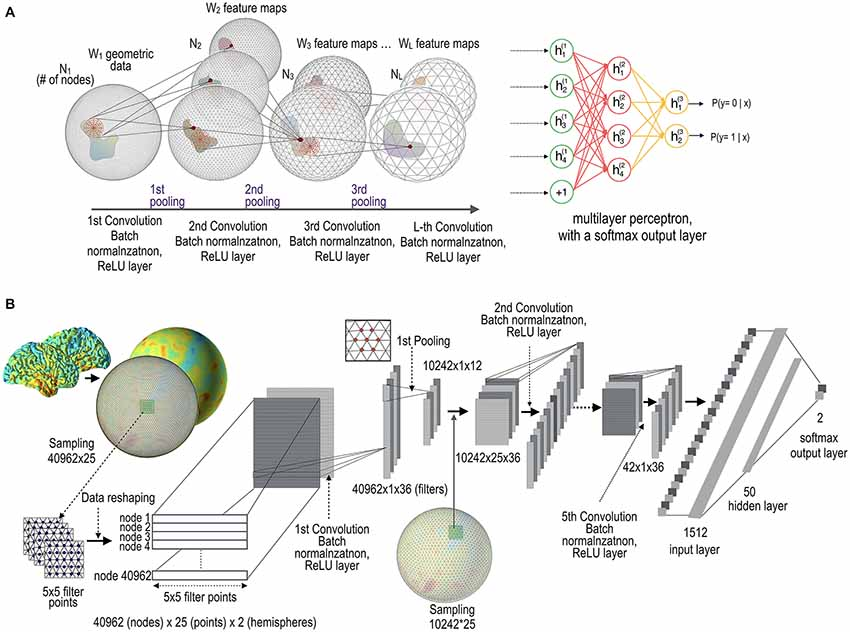


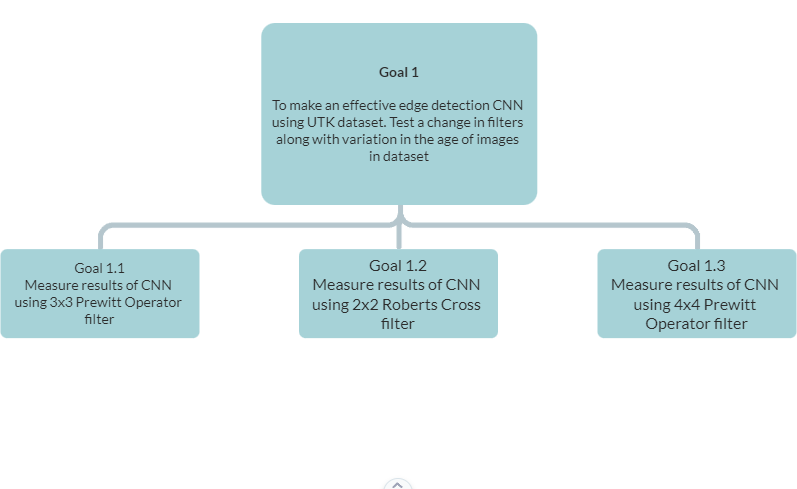
Figure 16 Seong, S. (2018, June 12)

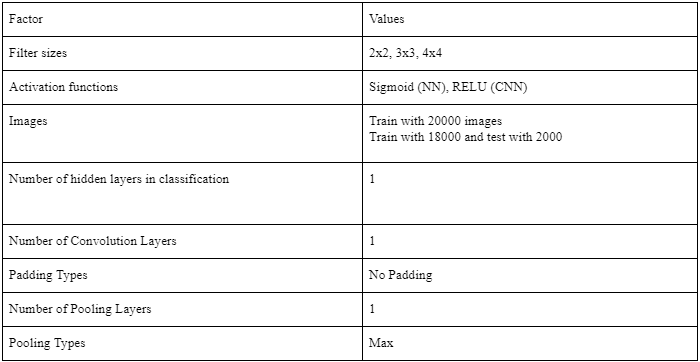
Coming down to actually training a convolutional neural network is commonly done within a python framework. Usually something such as Google colab or jupyter notebook. A large dataset is also required for the learning process, the larger the better. Operating a cNN does not require the large dataset but the neural network will have a larger accuracy when it comes to facial recognition the larger the dataset. Seong, S. (2018, June 12)

In conclusion, A brief description of what an ANN was covered then a deep dive into what CNN’s are and how they work. Lastly an overview of flaws of CNN’s and ways experts have circumvented issues along with possible alternative routes for facial recognition. Despite the flaws CNN’s appear to be the best candidate for developing a facial recognition system ANN. There are many routes to take for an efficient and diverse network. Best of all there is much supporting work behind CNN’s and they are constantly improving due to experts innovating in their field.

**5. Initial Experiment Design**

For the experiment to be conducted a facial recognition CNN will be developed and trained in a python environment with filter sizes of 2x2,3x3,4x4. The kernels for each being the Roberts cross for the 2x2,  Prewitt operator for the 3x3 and Prewitt operator for the 4x4, these are also referenced in the figure below. Comparing resulting accuracy to find best results. The secondary variables will include the use of one convolution layer, sigmoid activation function for the classification, one hidden layer in classification, and the use of max pooling. In addition, due to the nature of convolutional neural networks, the age of the individuals on the images is another variable being taken into account.  The hypothesis of this experiment is that a 3x3 filter will yield a higher accuracy than a smaller or larger filter size. Below is the goal tree and block diagram overviewing the variables involved in the experiment along with the goals required along the way

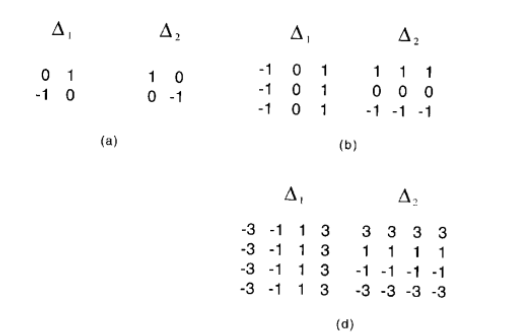




**6. Solution Description**

Going through the actual experiment some things that come through as solutions were found. As stated in the initial experiment expectations python was used as it is the industry standard for coding with artificial intelligence, it was only natural to use it. As it was created from scratch any language could have been used. No packages apart from standard one such as numpy and ones for random number generation and file manipulation were used. The initial design of the tests did not include the age of the individual on the image but that was a misunderstanding of how the neural network actually learns. The neural network has to be shown what is a correct answer and what an incorrect answer is. So therefore the initial idea of just seeing how accurate it is with no context or variable relating to success is impossible. The dataset used was the UTK dataset, a large face dataset with around 20000 faces having various ages, races, and a few other distinguishing features. Each file is denoted with certain numbers spaced out with underscores. The number at the beginning is the age associated with the respective image. Training was done with 80% of the dataset split into 10% subgroups. The testing was then done with the remaining 20%. The accuracy was based on the number of correct guesses the neural network made. The correct answers came from one of four categories. They were age ranges that are considered major age blocks and usually have considerably visual differences. The age ranges are 0-17, 18-30, 31-60, and 61 and above. The thought process was that those younger probably have less lines to read while those in higher ranges may have many wrinkles and be easier for the edge detection to catch.

The single convolution layer was performed with 3 types of edge detection filters. There was an inconsistency that was overlooked that will be discussed in the conclusion in relation to the filters chosen. As mentioned previously there was one 2x2, 3x3 and 4x4 filter. Each are proven and commonly used edge detection filters.

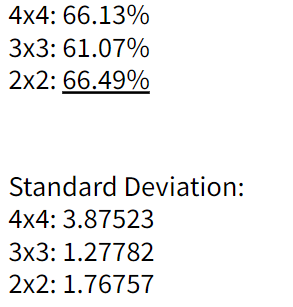


These filters were applied to a 200 by 200 pixel grayscale image. The images in the dataset are actually color but the color is not relevant when it comes to edge detection so all the images were converted to greyscale for simplification of calculation. The process of convolution is explained in detail in the earlier sections so will not be covered in the solution. For the rest of the single convolution layer there is a single max pool layer, an ReLU activation and the flattening layer.

For the neural network part of the code, there is a single hidden layer and output layer. There was no initial node count for the hidden layer, and after some discussion and testing a node count of 50 was chosen. As mentioned earlier there were 4 possible correct answers for the age variable so the output layer has 4 nodes. Both output and hidden layer use the sigmoid activation function. The input layer was just the result of the flattening layer from the convolution section. The weights connecting the input layer to the hidden layer and then connecting the hidden layer to the output layer were initialized as random numbers from -0.01 to 0.01. Tests were performed at varying nital weights from as high as from -1 to 1 and as low as -0.005 to 0.005. The selected range provided what seemed the most stable results. The backpropagation that occurred during training iterated 25 per image as to build up the weight values. A larger amount of training was considered but the run time was unreasonable. The learning rate was set to 0.2. This was also tested at different values and 0.2 seemed the most stable.

**7. Results**

The results of the experiment were somewhat unexpected but were also somewhat undermined by initial choices in some variables. The starting hypothesis stated that the 3x3 filter would provide the highest accuracy when it comes to guessing the age from the images in the dataset. The 3x3 filter performed the worst of the 3 but



did yield a lower standard deviation. It was less accurate but more consistent even if at a lower margin. The overall accuracy was higher than expected but the most unusual anomaly was the accuracy by age category. The accuracy was very low for images of younger individuals and higher the older they got. But there was a very low accuracy for the last category, the very elderly. I would expect them to have the highest  accuracy due to wrinkles. But this is an assumption that may be unfounded. No research has gone into  this hypothesis as part of my tests. It is not what is being tested here so no need to investigate further at the current time. Another factor affecting the results will have been the variation in the size of the datasets by category. This is most likely another factor skewing the data in an unexpected way. Did not have time to readjust the data and ensure each category was equally distributed. Unfortunately another factor overlooked during initial stages of development.

**8. Conclusion**

Overall, the experiment was successful in creating a functional convolutional neural network that tested the accuracy of certain edge detection filters after training with a decent sized dataset. There were many roadblocks in the way over the course of development, some due to an underestimation of how long certain things would take and the problems that would occur during testing. As referenced several times in the previous sections there were some factors that undermined the accuracy and results of neural networks. Many of them being the rocky foundation of initial assumptions of my experiment and the nature of progressive learning. The process of learning how some things didn't work the way that was assumed in the first place or new information was discovered after the fact. The largest of these being that edge detection filters come in diagonal variations. The 2x2 is a diagonal variant so therefore not a partially good candidate to compare to the 3x3 and 4x4 filters chosen. Nonetheless it was not a catastrophic difference, but it may have polluted the results somewhat. Many of the other changes were corrected over the course of development so aren't seen. Regardless, just like the neural network, a process of learning occurred over a long period. If there was more time to work on the project the statistical analysis could use more work, in addition, new filters would be chosen along with larger datasets for those very young or very old. In addition, the run time could use some work. There is a titanic amount of information but some of it is coded inefficiently as some parts were coded more for functionality rather than speed.

**References:**

* Seong, S. (2018, June 12). *Geometric Convolutional Neural Network for Analyzing Surface-Based Neuroimaging Data*. Frontiers. https://www.frontiersin.org/articles/10.3389/fninf.2018.00042/full.
* Xie, S., & Hu, H. (2017, January 24). Facial expression recognition with FRR-CNN. Retrieved from https://digital-library.theiet.org/content/journals/10.1049/el.2016.4328#C6
* Dlaptev. (n.d.). Dlaptev/ti-pooling. Retrieved May 23, 2021, from https://github.com/dlaptev/TI-pooling#:~:text=TI%2Dpooling%20is%20a%20simple,(CNN)%20transformation%2Dinvariant
* Laptev, D., Savinov, N., Buhmann, J., & Pollefeys, M. (2016, September 22). TI-POOLING: Transformation-invariant pooling for feature learning in convolutional neural networks. Retrieved May 23, 2021, from https://arxiv.org/abs/1604.06318
* Ujjwalkarn. (2017, May 29). An intuitive explanation of convolutional neural networks. Retrieved May 23, 2021, from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
* *Library for end-to-end machine learning. (n.d.). Retrieved May 23, 2021, from https://e2eml.school/how\_convolutional\_neural\_networks\_work.html*
* *A basic introduction to neural networks. (n.d.). Retrieved May 23, 2021, from http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html*
* Deshpande, A. (n.d.). A beginner's guide to Understanding convolutional neural networks. Retrieved May 23, 2021, from https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
* Kim, J. (2017). Efficient facial expression recognition algorithm based on hierarchical deep neural network structure. Retrieved May 23, 2021, from https://ieeexplore.ieee.org/document/8673885
* [10] Woodford, C. (2020, June 17). How neural networks work - a simple introduction. Retrieved May 23, 2021, from http://www.explainthatstuff.com/introduction-to-neural-networks.html
* Ahirwar, K. (2017, November). Everything you need to know about Neural Networks. Retrieved from https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491
* Aishwarya, S. (2020, May 08). Introduction to neural Network: Convolutional neural network. Retrieved May 23, 2021, from http://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network/
* Saravanan, D. (2020, October 30). A gentle introduction to math behind neural networks. Retrieved May 23, 2021, from <https://towardsdatascience.com/introduction-to-math-behind-neural-networks-e8b60dbbdeba>
* Skalski, P. (2019, April 14). Gentle dive into math behind convolutional neural networks. Retrieved May 23, 2021, from https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9
* Skalski, P. (2020, February 16). Deep dive into math behind deep networks. Retrieved May 23, 2021, from <https://towardsdatascience.com/https-medium-com-piotr-skalski92-deep-dive-into-deep-networks-math-17660bc376ba>
* Deeplizard. (n.d.). Loss in a neural Network explained. Retrieved May 24, 2021, from <https://deeplizard.com/learn/video/Skc8nqJirJg>
* Sharma, S. (2019, October 11). What the hell is Perceptron? Retrieved May 24, 2021, from <https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53>
* Tóth, B. (2018, August 13). How do forward and backward propagation work? Retrieved May 24, 2021, from https://tech.trustpilot.com/forward-and-backward-propagation-5dc3c49c9a05
* K., H. M. (2019, November 03). Backpropagation step by step. Retrieved May 24, 2021, from https://hmkcode.com/ai/backpropagation-step-by-step/#:~:text=Backpropagation%2C%20short%20for%20%E2%80%9Cbackward%20propagation,proceeds%20backwards%20through%20the%20network.