Optimization of Neural Networks

Word Count: 3975

Introduction

The invention of neural networks in the mid 20th century was found to be far ahead for its time to attract any attention. The recent upturn of machine learning and deep learning is due to the vastly improved, powerful hardware available today for computers. Google Translate, self-driving cars, cameras, and even stock market predictors all use these networks. Neural networks are a subset of the machine learning topic, which is further part of the diverse topic of artificial intelligence. These networks consist of very complex algorithms which form layers in the network. These layers are connected to each other in different ways, and each network has a specific architecture of layers that can affect the way the network runs. There are three parts to making one of these neural networks: making the architecture, training the network, and testing the network. The architecture is designed with a specific purpose in mind, and there needs to be a dataset to train and test the network with. Figure 1 is part of the MNIST dataset, which can be used to train a neural network to recognize handwritten digits.

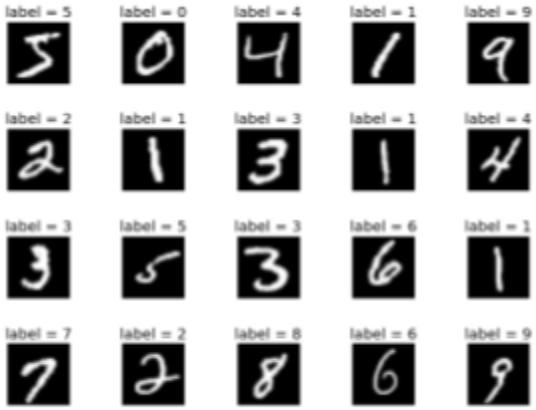
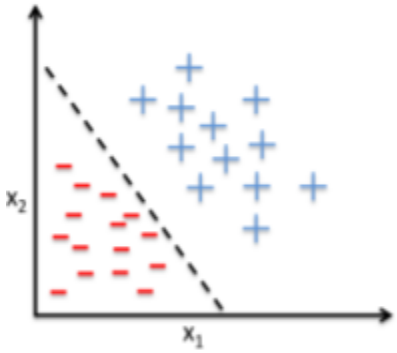


Figure 1

The most important part of a neural network is optimizing the architecture so that it has the highest success rate possible, and conversely the lowest error rate possible. Different types of neural networks are used depending on the situation. Convolutional neural networks (CNNs) are used to classify images, recurrent neural networks (RNNs) are used to look at sequential data, and generative adversarial networks (GANs) are used to generate new data, just to name a few

types of networks. In order to determine what the most optimal architecture is, many different types of networks with a vast array of architectures need to be analyzed.



Example of a linear decision boundary for binary classification.

Figure 2

Neural networks still continue to surprise researchers with the different problems that they are able to solve. All the secrets to the solution lie in how these algorithms run. To give an example, suppose we had two groups of data which, if plotted on a graph, can be separated by a line. A neural network would be used in this case to find the equation of the line that can separate these two groups, as seen in Figure 2. The input to the neural network would be the x and y coordinates of the different points. The neural network then initializes random numbers known as weights and multiplies them to the corresponding inputs and adds them up. A nonlinear function, such as sigmoid, is applied to this number and the neural network outputs a prediction - a 1 or a 0 - depending on if it thinks the point is above or below the line. It then checks to see if it predicted the output correctly by using a statistical error formula such as mean squared error. It uses the error to increase or decrease the weights in order to correctly predict the output, this is known as backpropagation. Figure 3 accurately depicts the formula for backpropagation. The change made to each weight is determined by how the error changes with respect to the loss, how the loss changes with respect to the nonlinear function s(x), and how the function changes with respect to the weights. It is essentially a chain of how everything relates to each other. This is one of the

$$\frac{\delta E}{w} = \frac{\delta E}{Loss} \cdot \frac{\delta Loss}{s(x)} \cdot \frac{\delta s(x)}{weights}$$

Figure 3

most simple problems to solve, but in order to solve complex problems, one would need to scale the network to match the type of data being worked with and change the architecture accordingly.

Literature Review

Varior, Haloi, and Wang solved the problem of human re-identification using a gated siamese convolutional neural network (S-CNN). One of the practical applications of their problem includes "matching pedestrians across multiple camera views," which they subsequently solved (Varior, Haloi, & Wang, 2016). A S-CNN is mainly used in problems that require matching two similar pairs of pictures. A "distance", known as the margin, between the two (or more) pictures is computed and then checked by the network to see if they are matching. They proposed a baseline S-CNN architecture to try and match two sets of pictures of the same person from different angles, but failed to do so as seen in Figure 4. The neural network incorrectly

Figure 4

ranks the pictures in the order that it thought were similar to the original picture. Convolutional neural networks (CNNs), networks that deal with images, extract very simple patterns in the early layers, but pick out abstract features such as faces in the deeper layers. To help improve the

network, Varior et al. decided to implement a “gating function” in order to compare the deeper and more relevant abstract features. Their architecture consists of 2 pairs of 4 layers of CNNs stacked in a parallel manner, connected by the aforementioned gating function. The highest multi-query accuracy that they managed to achieve was 68.1% of correct answers were contained within Rank 1, 88.1% of correct answers were contained within Rank 5, and 94.6% of correct answers were contained within Rank 10.

Sak, Senior, and Beaufays used a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) in order to model acoustics or learn the patterns inherent within speech. A conventional RNN is used to model sequential data such as music or speech. It contains layers that take in input from a previous time step, as well as from the data in order to keep some sort of “memory” of the data. The researchers state that they experimented with Deep LSTMs, which include multiple LSTM blocks; LSTM with projection layers (LSTMP), which reduce the numbers of parameters learned by the network, making the training stage computationally cheaper; and Deep LSTMPs, which include multiple LSTMPs stacked together in order to get the advantage of having multiple LSTM blocks along with the benefit of having a smaller parameter number. The dataset they tested their neural networks on was from the Google Voice Search task. This requires the neural network to convert natural language to text, a task we see in everyday life: iPhone’s Siri, Google’s voice command, and even in the latest cars. The results in the paper are self explanatory - conventional shallow LSTM RNNs do not perform as well as deep LSTM RNNs. Furthermore, the performance still improves with a deeper model with 5 layer LSTMP RNN. Increasing the number of layers increases the cost of computation exponentially. Also increasing the number of parameters does not necessarily increase the

accuracy of the network. The best accuracy with the most computationally efficient structure is seen in the model with a 2 layer LSTMP RNN with 13 million parameters. The researchers conclude that adding a projection layer to the LSTM RNN and increasing the number of LSTM blocks in the network increases the accuracy substantially to match state-of-the-art performance.

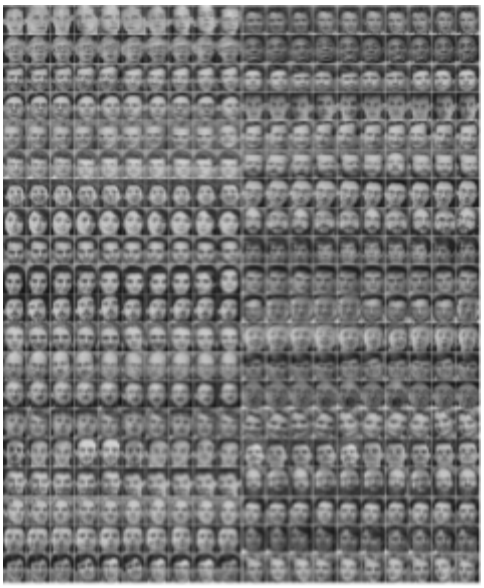


Figure 5

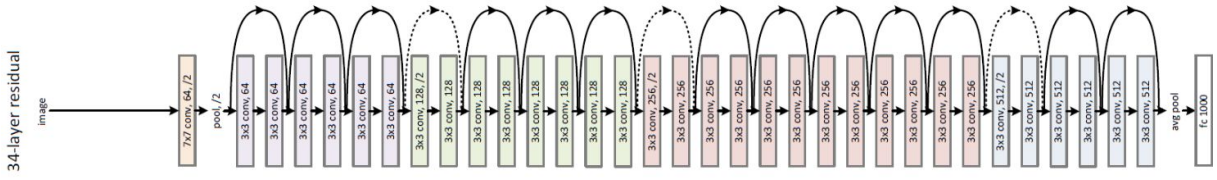
Lawrence, Giles, Tsoi, and Back examine the most efficient network to recognize faces from a multiple class database. The dataset they use contains "10 different images of 40 distinct subject (Lawrence et al., 1997)." An image of the dataset is seen in Figure 5. A high level overview of the architecture shows that the neural network they used is very conventional in today's standards. The image is fed into the neural network which subsamples from the image, reduces the dimensions, extracts the most important features, and finally classifies the image. Lawrence et al. vary the number of classes, the dimensionality, quantization level, image sample extraction algorithm, and the dimensionality reduction method. The best results occur when the number of nodes in the self organizing map of the network is 8 or 9. The network shows an error rate of 3.83% in both cases, which is viable for commercial use if it was to be used.

Methods

The method in which the research was conducted was sought to find a solution to the proposed question: It is possible to find the most optimal architecture for a neural network, and if

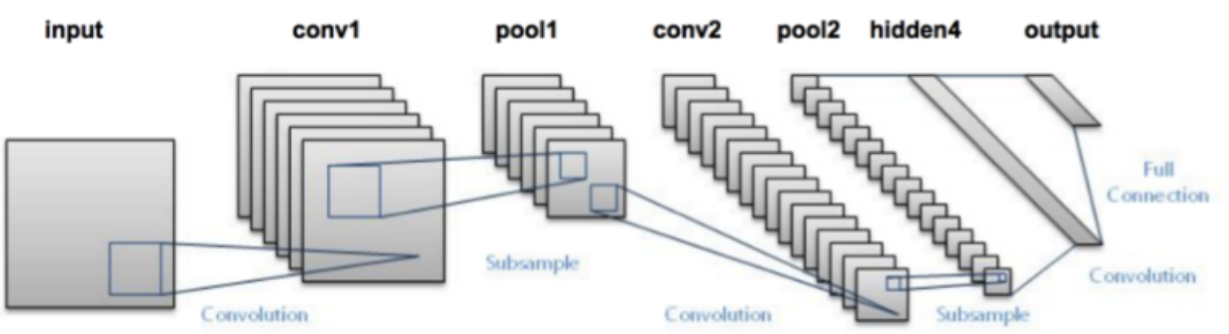
it is, how does one do so? The questions assumes that there *is* an optimal architecture, a one-fits-all solution that would be the most efficient and would produce the best results for any single neural network. Keeping this in mind, I looked through many different sources to find a desirable answer to the question.

My initial proposal to conduct research was to look at the many different architectures present in the purpose-dependant networks and then analyze which produced the best results and if they were viable for purposes other than what they were made for. I looked through many different research papers, textbooks, videos, and especially famous networks' architectures looking specifically for their design and the corresponding results. I made the incorrect assumption that there is a single architecture that would solve the problem of brute force testing different networks to come to a conclusion of efficiency. For example, the competition winning ResNet (2015) has a 3.57% error rate on the dataset, which is surprisingly better than the results that humans have showed. It can be concluded that ResNet has the best architecture for convolutional neural networks as it achieves the lowest error rate on the dataset. Furthermore, it has specific modifications built in in order to lower its error rate even more. However, this network is only specific to tasks that require image recognition and cannot be used for any other problem. Tasks such as natural language processing and even simpler image recognition tasks would not be able to use architectures such as this one.



I then analyzed the architecture of another convolutional neural network, in order to keep the underlying purpose the same - classifying pictures. The change was introduced in what type of pictures the network was classifying. ResNet was used for classifying all types of objects: lemons, pizzas, and sheep just to name a few. The next network that I analyzed was LeNet. This neural network was used to classify handwritten digits from 0 to 9 as shown below.

It is evident that ResNet has a relatively large architecture with 34 layers. LeNet is known to be the “Hello World” introduction to neural networks and is one of the most simple problems to solve. As seen below, LeNet does not have a very complex architecture. The reason being that it is not a complex problem.



I tried to train ResNet to recognize handwritten digits on my own computer, but the architecture was so large and computationally intense that my computer could not handle the training. It is evident from this problem itself that one architecture can be completely overkill if it is trying to solve a problem it was not designed to solve. I, however, was able to implement two other neural networks to test my findings from my analysis of the different research papers.

Results

The research papers that were analyzed each had a main problem to which the solution was a neural network. The solution did not necessarily have to be a neural network, but seemingly, it was the best approach to the corresponding problem. Most of the papers had data concerning the statistics of each network and how it performed with that particular architecture. This provided a lot of insight into the various aspects that optimize the network, and how a universal architecture could be brought around that utilized these optimization techniques. It's important to clarify that it is not yet possible for there to be one single architecture that can optimize the performance of all neural networks. Architectures cannot translate between different networks for the equivalent task. However, it is possible to identify trends evident in the architectures that seem to correlate with better accuracies.

In the research paper solving the problem of human re-identification, Varior et al. propose a baseline architecture, which fails to accurately identify the pictures, but move on to create a superior architecture that is on par with state-of-the-art networks of the same caliber. The initial model was not successful due to the incorrect pattern identification by the network. As seen in Figure 4, the model misattributed patterns from the query image to the identified image. For the first query, the model only looks near the same region of the pattern recognized in the query image (in this case the backpack area). It thinks that the dark blue backpack was closer attributed the black suit in the picture ranked 1 and to the stripes of the clothing on the person that ranked 2. In the second query, the pictures look much more similar so it can be understood why the images were misclassified. However, it is still apparent that the neural network looks at the same region on the classified images as the patterns recognized on the query image. The beanie and shoulder areas contained unique patterns identified by the network, but it failed to rank the correct picture as rank 1. The pictures ranked 1 and 3 by the network instead are the same image and rank 2 was the correct counterpart. In order to fix this misattribution, the researchers came up with a matching gate function that allows the network to learn patterns from the mid layers. These patterns are then compared and evaluated with a loss function that informs the network how close the two patterns in the two images that it matched are. The function is then optimized to find the least distance, or loss, between the images which subsequently forces the network to learn what patterns to look for in matching pairs of images. The architecture itself is pretty common in the world of convolutional neural networks, but “the proposed work is the first of its nature to introduce differentiable gating functions in siamese architecture for human re-identification” (Varior, Haloi, & Wang, 5). On multi-query testing, their network achieved an

accuracy of 76.04% on the Market-1501 dataset, about 3% better than the baseline model. This model outperformed all of its counterpart, proving its efficiency for commercial use.

The models used in *Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling* include information that will allow new correlations to develop for optimization of architectures. Many different types of RNNs are proposed and tested in this research paper. As mentioned in the Literature Review, the models range from shallow LSTMs to deep LSTMs to deep LSTMPs. It is clear that as the number of layer increases from 1 to 7, the word error rate percentage (WER%) decreases, then increases again after the 5 layer threshold. There is most likely a correlation between the number of layers and the accuracy of the network. As the number of layers increase, the accuracy increases, but then decreases again after hitting a certain threshold. This threshold can be determined by testing the network out multiple times. Furthermore, adding a projection layer reduced the accuracy even more compared to the LSTM without a projection layer. A projection layer does not necessarily increase accuracy; this can only be determined after testing the accuracy and if given the problem that the network is trying to solve. With the LSTMP models in the paper, after increasing the number of layer from 2 to 3 and with a variable number of projection units, the WER% increased .2% from 350 units to 512 units with 2 layers, then stayed the same after increasing the number of layers. It can be concluded that increasing the number of parameters in this case will only increase the training time, but will not necessarily increase the accuracy.

The aforementioned high-class convolutional neural network research paper explicitly changes the different variables in order to see if there are any correlations to the changed variable and the affected accuracy. The first variable changed, is the number of classes used to train and

test the network on. When the number of classes increases from 10 to 20 to 40, the error rate increases from 1.33% to 4.33% to 5.75%. Additionally, Lawrence et al. talk about a self organizing map (SOM). An SOM reduces the dimensions of the image to some number and only keeps the most important traits in the data that can be modeled by some latent variables. The variables can then be used to distinguish different features in images, as when the dimensions of the image is reduced, the latent variables will be distinct. Essentially, only the most important features of the image are being extracted, reduced to some numbers known as latent variable, and then are compared with each other. Going back to the paper, when the number of the SOM dimensions increases from 1 to 4 the error rate decreases from 8.25% to 5.83%. The reasoning being that when you reduce an image to 1 dimension, you are extracting only the most important feature in that set of images, and comparing them. When you increase the number for dimensions to 4, there are more features that are important being compared. But there comes a certain threshold for the number of dimensions, that when exceeded, either does not change the error rate or might even increase the error. This is because when there are too many features being compared, a lot of inessential features make their way to the latent variables and make the comparisons inefficient. A good example would be comparing the latent variables to the features of a face. When reducing the number of dimensions to 1, only the most important feature of the face is preserved, say skin color for example. When the number of dimensions is increased to 4, other important features make their way into the variable space. For example, skin color, hair length, nose shape, and head size could be considered allowing for a better differentiation between faces. When the number of dimensions gets too big, less important features, like moles and pimples, might be compared, which presents a very small difference in comparison to the

previously mentioned features. Other variables in the research paper include image sample algorithm: pixel intensity produces an error rate of 5.75% and differences w/base intensity - 7.17%. The other variables are not as influential as the ones mentioned and will be omitted.

It should also be mentioned that it is guaranteed that networks are transferable between similar tasks. I personally tested this out by using a very computationally expensive network to classify the MNIST dataset seen in Figure 1. As said before, this is a very simple task that should be solved by a basic neural network. However, I trained the Ademxapp Model A neural network - a very large network that is able to identify any object inputted. An analogy would be like using a 10,000 watt power source for a light bulb - it is completely overkill. Before even trying to train the network, I immediately ran into an error - my computer would run out of memory and the software crashed, similar to how the light bulb would explode when using too much electricity. Additionally, I experimented with creating an RNN that incorporates multiple LSTMs to model the wording similar to Shakespeare's language and even the American Constitution's language. The network consisted of 2 LSTM layers and had a training accuracy of 86.8% and a testing accuracy 71.6%. During the testing stage, it was found that the network was consistently spewing out the same phrase, with no variation. So a remedy was to take a random sample of the probability of each letter showing up next, and take a letter from one of the higher probabilities. This resulted in dramatic improvement as the network now had no way of repetition and modeled the texts it was trained on accurately. I also testing adding and subtracting LSTM layers from my network and found that increasing the amount of layers did correlate to better accuracies, but it had to be done in moderation. I could get away with adding 1 layer, but any more than that and the network's training time would increase and the accuracy would stagger.

In testing a CNN that I made, I found that increasing the number of compounded layers did make a relatively large change to how the network performed. But again, testing is incredibly important to determine how many layers need to be added or taken away and how to improve the accuracy of a net in practice.

Discussion

In order to understand the results, it is important to state the limitations that might have influenced the study. Many of the research papers analyzed, if not all, had a complex notation of the mathematical operations involved in the structure of the neural network and the optimization of the architecture. This restricted the interpretation of the studied research papers, which subsequently may have resulted in omitting some of the information from the studies. Furthermore, the techniques used in the optimization are specific to the neural network and depend on the dataset the researchers were using. It is not guaranteed that using similar techniques will improve the network in the same manner demonstrated in the papers. None of the techniques were tested in order to verify if they are accurate or not.

The biggest assumption made was that there is a specific architecture that can be optimized in order to prove useful. However, as the study progressed, it was seen that this assumption proved to be false and that each neural network is designed differently, to correspond to the problem. In order to accomodate for the false assumption, the study shifted to identify the general optimization techniques used for neural network architectures. Neural networks are part of supervised learning - a branch of machine learning that is considered a solved problem. It is still in the best interest of researchers to improve on the optimization techniques in order to bring these learning systems to the commercial market. The results are significant because they are

applicable in practice and they show a general increase in the accuracy of the neural network. It should also be noted that optimizing the architecture may or may not be the most important part to the problem. For example, a good dataset or specializing your net could increase the accuracy by a significant portion. In these cases it would be up to the creator whether to optimize the architecture or not.

Conclusion

The evidence brought forward by the research has led up to a concrete conclusion. This has been developed throughout the paper and will be reiterated again: there is no single optimal architecture for the entirety of neural networks. Neural networks come in a different types and the specified problem needs to be indicated in order for them to be useful. The reason this topic has not been looked at before is due to the apparent answer. If neural networks inherently have different architectures, then there is no way to find a single optimal architecture that could be applied to all networks. Therefore, it is key that the type of neural network is identified, then optimized using the techniques described in this paper.

It would be interesting to work on optimizing only a single neural network in the future. It could serve as a precedence for works that take inspiration from it and could be of help to other people who are working on a similar task. Taking a single problem and expanding on it would give larger room for detail and a smaller domain of research. Furthermore, the research would be limited to only the single problem, and translating the optimization to other networks would not be a factor of the paper. If a new way of optimizing a network does show up, it could be used in addition to the many other procedures in order to reduce the error rate to improve the

neural network. These learning systems still have a long way to improve before they take over all areas of our, but an increase in use will not be unheard of.

References

"ARTIFICIAL INTELLIGENCE with JOHN McCARTHY, Ph.D." *The Intuition Network*, www.intuition.org/txt/mccarthy.htm.

Benardos, Panorios & Vosniakos, George-Christopher. (2007). Optimizing feedforward artificial neural network architecture. Engineering Applications of Artificial Intelligence. 20. 365-382. 10.1016/j.engappai.2006.06.005.

Can Yılmaz, Ali & Aci, Cigdem & Aydin, Kadir. (2015). MFFNN and GRNN Models for Prediction of Energy Equivalent Speed Values of Involvements in Traffic Accidents / Trafik Kazalarında tutulumunun Enerji Eşdeğer Hız Değerleri Tahmininde MFFNN ve GRNN Modelleri. International Journal of Automotive Engineering and Technologies. 4. 102. 10.18245/ijaet.78159.

Corochann. "MNIST Dataset Introduction." *CorochannNote*, corochann.com/mnist-dataset-introduction-1138.html.

Das, Siddharth. "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and More ...." *Medium.com*, Medium, 16 Nov. 2017, medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5.

Demuth, Howard B., et al. *Neural network design*. Martin Hagan, 2014.

Gregor, Karol, et al. "Draw: A recurrent neural network for image generation." *arXiv preprint arXiv:1502.04623* (2015).

H. A. Rowley, S. Baluja and T. Kanade, "Neural network-based face detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, Jan. 1998.

Haykin, Simon Neural networks and learning machines / Simon Haykin.—3rd ed

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

Rosebrock, Adrian. “LeNet - Convolutional Neural Network in Python.” *PyImageSearch*, 1 Aug. 2016, www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/.

S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face recognition: a convolutional neural-network approach," in *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98-113, Jan. 1997.

Sak, Haşim / Senior, Andrew / Beaufays, Françoise (2014): "Long short-term memory recurrent neural network architectures for large scale acoustic modeling", In *INTERSPEECH-2014*, 338-342.

“Single-Layer Neural Networks and Gradient Descent.” *Dr. Sebastian Raschka*, 24 Mar. 2015, sebastianraschka.com/Articles/2015_singlelayer_neurons.html.

Varior, R.R., Haloi, M., Wang, G.: Gated siamese convolutional neural network architecture for human re-identification. In: ECCV. (2016)

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

Zhang, Quanshi, Ying Nian Wu, and Song-Chun Zhu. "Interpretable convolutional neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.