Neural Layer Bypassing Network

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

13-10-2021 / 11-01-2022

CITATION

Palasamudram, Amogh (2021): Neural Layer Bypassing Network. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.16806928.v2>

DOI

[10.36227/techrxiv.16806928.v2](https://doi.org/10.36227/techrxiv.16806928.v2)

Neural Layer Bypassing Network

A Novel Architecture to Increase the Speed of Forward Propagation without Compromising on Accuracy, Network Structure, or CPU Load

By: Amogh Palasamudram

Mentor: Dr. Ronjon Nag

Abstract

This research introduces and evaluates the Neural Layer Bypassing Network (NLBN), a new neural network architecture to improve the speed and effectiveness of forward propagation in deep learning. This architecture utilizes 1 additional (fully connected) neural network layer after every layer in the main network. This new layer determines whether finishing the rest of the forward propagation is required to predict the output of the given input. To test the effectiveness of the NLBN, I programmed coding examples for this architecture with 3 different image classification models trained on 3 different datasets: MNIST Handwritten Digits Dataset, Horses or Humans Dataset, and Colorectal Histology Dataset. After training 1 standard convolutional neural network (CNN) and 1 NLBN per dataset (both of equivalent architectures), I performed 5 trials per dataset to analyze the performance of these two architectures. For the NLBN, I also collected data regarding the accuracy, time period, and speed of the network with respect to the percentage of the model the inputs are passed through. It was found that this architecture increases the speed of forward propagation by 6% - 25% while the accuracy tended to decrease by 0% - 4%; the results vary based on the dataset and structure of the model, but the increase in speed was normally at least twice the decrease in accuracy. In addition to the NLBN's performance during predictions, it takes roughly 40% longer to train and requires more memory due to its complexity. However, the architecture can be made more efficient if integrated into TensorFlow libraries. Overall, by being able to autonomously skip neural network layers, this architecture can potentially be a foundation for neural networks to teach themselves to become more efficient for applications that require fast, accurate, and less computationally intensive predictions.

1. Introduction

Neural networks have significantly improved and dominated deep learning in the past decades. They can reach high levels of precision and recall and in some cases, outperform humans as well. One main reason why neural networks do well is because of their large and complex architectures. Though these architectures increase the accuracy of neural networks, they tend to take more time for forward propagation, so they output results slower than other deep learning architectures. This raises issues in applications of neural networks that require speed and accuracy. For instance, this issue is present in self-driving cars, where a minuscule delay can create a life or death situation for passengers.

Some have attempted to reduce the forward propagation time for deep neural networks. However, the majority of people solve this issue by reducing the number of layers or number of neurons per layer, which can improve prediction time, but this significantly decreases the accuracy and capability of neural networks due to potential underfitting. There are also certain architectures that help speed up deep learning applications without sacrificing the accuracy of the algorithm in various ways, but these solutions generally make use of different architectures, such as the 2017 Illustrated Transformer.¹

In this research, I will be proposing a new architecture that uses the fundamental building blocks of standard neural networks, such as convolutional and pooling layers in the case of CNNs, to forward propagate and perform tasks. However, I intend to have checkpoints after each layer to determine whether the forward propagation should continue or not. This decision depends on the probability or confidence of the model that a certain output should be produced after passing inputs through a variable number of layers in the network.

¹ Attention is All You Need - 07/2021: <https://arxiv.org/pdf/1706.03762.pdf>

2. Related Work

2.1 Transformers

Transformers are architectures that use self-attention to learn sequences across elements of inputs, allowing long-term dependencies and scalability.² Initially they were used in Natural Language Processing but have been adapted to work in Computer Vision through data manipulation as well.³ Transformers provide attention and processing power to specific parts of the inputs that are important when obtaining the final output. This makes them more efficient as the network focuses on particular portions of the inputs rather than the entire input.

2.2 Pipelining

Machine Learning Pipelining is a technique to optimize the input data for a neural network to reduce the processing time and memory required for a model to perform forward propagation.⁴ This includes preprocessing and compressing data for neural networks to use. Some examples are Processor Pipelining and Instruction Pipelining.

3. Architecture

I am proposing a new architecture (the Neural Layer Bypassing Network or NLBN) that revolves around the standard neural network structure but adds a new layer (the rejection layer or rejection model) to each layer of the main model to determine the probability of the semi-processed input resulting in a certain output. If this probability is above a trainable or fixed threshold, the forward propagation will be stopped and the next input will be taken. Else, the forward propagation will continue, and this process will occur for the next layers as well.

² Transformers in Vision: A Survey - 10/2021: <https://arxiv.org/pdf/2101.01169.pdf>

³ Transformers in CV - 10/2021: <https://towardsdatascience.com/transformer-in-cv-bbdb58bf335e>

⁴ Entropy-Aware I/O Pipelining for Large-Scale Deep Learning on HPC Systems - 10/2021
<https://ieeexplore.ieee.org/abstract/document/8526881>

Figure 1 - Illustration that depicts how this architecture works in a standard convolutional neural network.

In Figure 1, the Legend is as follows:

1. **CONV Layer** - Convolutional Layer for Feature Extraction.
2. **Pooling Layer** - Average or Max Pooling Layer for Feature Extraction.
3. **FC Layer** - Fully Connected Layer with standard weights for classification.
4. **R** - Rejection Layers to reject input if it is unlikely to have an expected result.

In this manner, the NLBN is not expected to forward propagate each input through the entire model. If there are certain inputs that can be classified with fewer layers, this is identified through the rejection models (or rejection layers) and only the required number of layers are used. This will make predictions faster because inputs are only partially passed through the NLBN, resulting in fewer computations.

However, a possible drawback is an increase in training time and memory required due to the extra rejection layers. Additionally, the recall or precision of the network may slightly decrease depending on the number of layers and labels, since inputs can be rejected at any stage of the network, potentially increasing the false negatives or increasing the false positives from predictions.

This approach to speed up neural networks is mostly applicable for networks used in real-time classification problems or utilities that require less power consumption and faster predictions.

This includes public chatbots with millions of simultaneous users and smart security cameras that alert owners when they detect an unidentified individual on their property.

4. Procedure

In my experiment, I will be training a standard CNN and my NLBN architecture for 3 datasets: MNIST Handwritten Digits Dataset, Horses or Humans Dataset, and Colorectal histology dataset.

I will also be collecting and analyzing specific data regarding the performance of the NLBN architecture. This will be the primary focus of this procedure section

4.1 Variables

Independent Variable:

1. Percentage (%) of the model that the inputs are passed through

I chose this as the independent variable because the NLBN is hypothesized to work by reducing the percentage of the model that the inputs are passed through. By changing this variable, I will be able to accurately determine the effectiveness of this architecture.

Dependent Variables:

1. Speed of forward propagation
2. Accuracy
3. Percentage (%) of inputs passed

I chose the speed of forward propagation and the accuracy as two of the independent variables because these are the two primary metrics that the performance of the NLBN will be analyzed on. I chose the percentage (%) of inputs passed through the model as my other dependent variable because it directly influences the time taken for model predictions, according to my theory.

4.2 Datasets

MNIST Handwritten Digits Dataset:⁵ Contains 60,000 images with resolution of 28x28. The dataset contains images of handwritten digits from 0 to 9 with an even distribution.

Horses or Humans Dataset:⁶ Contains 1000+ images with resolution of 300x300 in RGB. The dataset contains images of horses and humans with a 50:50 distribution.

Colorectal Histology Dataset:⁷ Contains 5,000 images with resolution of 150x150 in RGB. The dataset contains images of colorectal cancer cells with an even distribution.

4.3 Experiment

To determine the effectiveness of the NLBN architecture I have proposed, I will be using Google COLAB to analyze the performance of my NLBN and CNN models. These networks were created using the Keras⁸ and TensorFlow⁹ open source libraries.

I have coded COLAB Jupyter Notebooks with the models for the three different datasets:

1. MNIST Handwritten Digits Dataset¹⁰
2. Horses or Humans Dataset¹¹
3. Colorectal Histology Dataset¹²

I chose these datasets from Tensorflow,¹³ because they have a wide variety of features, including but not limited to image type, image size, and difficulty of classifying images. This diversity of inputs will holistically present the effectiveness of the NLBN.

⁵ MNIST Handwritten Digits Dataset - 07/2021:
https://web.stanford.edu/~hastie/CASI_files/DATA/MNIST.html

⁶ Horses or Humans Dataset - 07/2021: https://www.tensorflow.org/datasets/catalog/horses_or_humans

⁷ Colorectal Histology Dataset - 07/2021: https://www.tensorflow.org/datasets/catalog/colorectal_histology

⁸ Keras API - 07/2021: <https://keras.io/api/>

⁹ TensorFlow Libraries - 07/2021: <https://www.tensorflow.org/resources/libraries-extensions>

¹⁰ Neural Layer Bypassing Network (NLBN) - MNIST Digits Dataset:
https://colab.research.google.com/drive/1KiFhmRnmuasj_oTUNWmncqWiuGkOOsS0?usp=sharing

¹¹ Neural Layer Bypassing Network (NLBN) - Horses or Humans Dataset:
https://colab.research.google.com/drive/1SEvW_YxPNvFXx163PdvPADS4L7ioqcxN?usp=sharing

¹² Neural Layer Bypassing Network (NLBN) - Colorectal Histology Dataset:
<https://colab.research.google.com/drive/17ml4Xkn01lle5XoSt0oAdCVJ2O1DiX2y?usp=sharing>

¹³ TensorFlow Datasets - 07/2021: <https://www.tensorflow.org/datasets/catalog/>

Steps to create and use NLBN architecture:

1. Import the necessary packages and libraries.
2. Upload and preprocess the training and testing data (in this case images).
3. Create a list of TensorFlow / Keras layers called 'layers' that would succeed one another in a model.
4. Create a list of TensorFlow / Keras models (Broken Models - 'bModels') that input the layers from the previous index and output the layers in the same index.
 - a. The first index (0) will input the images from the training data.
5. Train the last model in the 'bModels' list; this will automatically train the preceding models.
6. Create a list of rejection layers (Rejection Models - 'rModels') that input the layers in the same index of the 'bModels' and output the probability of an image being a certain class.
7. Train each element of 'rModels'.
8. Create the final list of models that are used for predictions (Structured Models - sModels). Use the layers in the 'layers' list created earlier and load the weights of the respective layers in the 'bModels' list into these layers.
9. Create a list of confidence levels to determine when to exit forward propagation. (These confidence levels could be trained as well, but this is out of the scope of this research.)
10. Forward propagate through the Structured Models:
 - a. Forward propagate through the first 'sModel'.
 - b. Forward propagate through the first 'rModel'.
 - c. Compare the maximum value of the prediction to the assigned confidence level of the specific layer. If the value is greater, exit forward propagation with this label. Else, resume forward propagation in the next layer.
 - d. Repeat the previous step until the NLBN model exits forward propagation or passes through the entire model.

5. Data

From my experiment in the Google COLAB Jupyter Notebooks, I collected the following data.

Table 1: Speed of NLBN and CNN (standard model) for all three datasets. Average is based on 5 trials.

Dataset	Average Speed of NLBN - Inputs/Second	Average Speed of CNN - Inputs/Second
MNIST Handwritten Digits	8.1	7.6
Horses or Humans	8.8	7.2
Colorectal Histology	7.9	7.4

Table 2: Accuracy of NLBN and CNN (standard model) for all three datasets. Average is based on 5 trials.

Dataset	Average Accuracy of NLBN - (%)	Average Accuracy of CNN - (%)
MNIST Handwritten Digits	95.8	99.2
Horses or Humans	99.5	99.7
Colorectal Histology	94.8	97.1

The following tables are based on data collected from the NLBNs in all three datasets. Each table contains data regarding the accuracy, speed, and number of inputs passed into the models with respect to how far the inputs travel through the model layers. Because the number of layers is different for each model, the tables cannot be standardized across datasets. Hence, the following tables are separated by the dataset used.

Table 3: MNIST Handwritten Digits Dataset - Accuracy, Speed of Predictions (as a % of the greatest speed), and % of Inputs Passed vs. the % of the Model the Inputs are Passed through.

% of Model	Accuracy (%)	Speed	% of Inputs
10	92.23	100.00	100.00
20	94.27	96.82	17.90
30	96.94	96.06	2.78
40	97.26	93.84	0.55
50	97.72	100.00	0.07
60	98.55	98.39	0.02
70	98.68	92.42	0.00
80	98.65	91.04	0.00
90	98.59	91.73	0.00
100	98.71	89.70	0.00

Table 4: Horses or Humans Dataset - Accuracy, Speed of Predictions (as a % of the greatest speed), and % of Inputs Passed vs. the % of the Model the Inputs are Passed through.

% of Model	Accuracy (%)	Speed	% of Inputs
5	96.67	100.00	100.00
10	99.23	98.31	0.10
15	100.00	94.31	0.00
20	100.00	95.08	0.00
25	100.00	93.55	0.00
30	100.00	95.87	0.00
35	100.00	92.06	0.00
40	100.00	96.67	0.00
45	100.00	95.87	0.00
50	100.00	88.55	0.00
55	100.00	87.22	0.00
60	100.00	86.57	0.00
65	100.00	87.22	0.00
70	100.00	86.57	0.00
75	100.00	87.22	0.00
80	100.00	85.93	0.00
85	100.00	85.29	0.00
90	100.00	84.06	0.00
95	100.00	83.45	0.00
100	100.00	82.86	0.00

Table 5: Colorectal Histology Dataset - Accuracy, Speed of Predictions (as a % of the greatest speed), and % of Inputs Passed vs. the % of the Model the Inputs are Passed through.

% of Model	Accuracy (%)	Speed	% of Inputs
7	88.14	100.00	100.00
13	93.47	99.60	69.20
20	95.23	99.00	45.73
27	95.76	98.22	19.65
33	96.21	98.02	12.23
40	96.24	97.83	1.48
46	96.58	97.64	1.13
53	96.69	96.69	0.80
60	97.08	97.25	0.45
67	97.75	96.31	0.18
73	98.06	95.94	0.10
80	98.52	95.02	0.05
87	98.78	94.84	0.03
93	98.41	94.66	0.00
100	98.73	94.30	0.00

6. Analysis

From the data above, I will be plotting three different graphs with the different dependent variables (accuracy, speed, and percentage of inputs passed) against the independent variable (percentage of model that the inputs pass through). Each graph uses data from Tables 3 - 5. Hence, each graph is not made for each dataset as done for the tables.

Graph 1: Accuracy (%) vs. % of Model that Inputs are Passed through

In Graph 1, there is a strong, positive (but not directly proportional) correlation between the accuracy and the percentage of the model that inputs pass through. There are no major outliers in the data.

For the MNIST Handwritten Digits dataset, the accuracy increases quickly in the beginning (by roughly 5%, from 92% to 97% after the model percentage increases from 10% to 30%).

However, when the percentage of the model that inputs pass through increases from 30% to 100%, the accuracy only increases by 2%, from 97% to 99%. The accuracy appears to start stabilizing at 99%. The maximum accuracy is reached after inputs pass through 60% of the

model. From this, it can be inferred that the model only requires the beginning 60% to be as accurate as possible.

In the case of the Horses or Humans dataset, the accuracy quickly rises by roughly 3% (from 97% to 100%) after the model percentage increases from 5% to 15%. The accuracy then stays constant at 100% for the rest of the model. This suggests that only the first 15% of the model is required to make accurate predictions.

Finally, for the Colorectal Histology dataset, the accuracy significantly increases when the model percentage increases from 7% to 20% (the increase in accuracy is 7% from 88% to 95%). From 20% to 80% of the model, the accuracy steadily increases by 3% (from 95% to 98%). After 80% of the model, the accuracy seems to be roughly constant at 98.5%. This suggests that it would require 80% of the model to make predictions as accurate as the full model. However, it would only require as much as 43% to make accurate predictions.

From this analysis and Graph 1, it can be seen that as the percentage of the model that inputs pass through increases, the accuracy of the model increases until it cannot be further increased. For this reason the accuracy tends to increase much more in the early stages of the model, in comparison to the later stages. The curves are limited or asymptote-like to the maximum accuracy. This signifies that the NLBN architecture will not need to pass through 100% of the model to make accurate predictions.

The results of the percentage of inputs passed through a certain percentage of the model is given in Graph 2.

Graph 2: % of Inputs Passed vs. % of Model that the Inputs Passed through in the NLBN Architecture.

In Graph 2, there is a strong correlation between the percentage of inputs and the percentage of the model these inputs pass through in the NLBN. There are no major outliers, and there is a constant downwards trend of the percentage of inputs, as expected. This is because the layers of the model can only use a percentage of the input less than or equal to that of the previous layer.

For the MNIST Handwritten Digits dataset, the percentage of inputs decreases by over 80% in the first 10% of the model. Next, the percentage on inputs reduces by 15% (from 18% to 3%). The percentage of inputs then continuously decreases in an exponential-decay fashion until it reaches 0.00%. This shows that nearly all of the inputs only pass through the first 30% of the model, as predicted by the analysis of Graph 1. Moreover, the greatest decrease in the percentage of inputs is within the first 10% of the model. This highlights how the first layer of the model was accurate enough to predict the majority of the inputs, portraying the NLBN's effectiveness.

With the Horses or Humans dataset, there is a drastic decrease in the percentage of inputs after the first 5% of the model. The percentage of inputs decreases from 100% to 0.10%. 0.00% of

the inputs are passed through 10% to 100% of the model. This aligns with the analysis of the dataset for Graph 1, as the accuracy with only 5% and 10% of the model was extremely high (close to 100%). Hence, it can be determined that the NLBN architecture improved performance by significantly reducing the number of layers used to still make accurate predictions.

Lastly, for the Colorectal Histology dataset, there is a constant downwards trend for the percentage of inputs passed through the given percentage of the model, until the percentage of inputs reaches 0.00%. In the first 20% of the model, the inputs steadily reduce from 100% to 20%. In the next 13% of the model, the percentage of inputs reduces from 20% to 1% and eventually decreases to 0.00% after passing through 93% of the model. From this, it can be inferred that the NLBN is consistently reducing the number of inputs passing through the layers of the model. This is done slower, compared to the NLBNs in the other datasets but is justified by the analysis in Graph 1, which showed that the accuracy of the model slowly but steadily increased in the middle layers. Hence, the percentage of inputs passed through the model slowly and steadily reduces in the beginning and middle of the model as well.

Overall, this analysis of Graph 2 for the three datasets presents how the NLBN architecture does not pass all the inputs through the layers but stops forward propagation when necessary. As the percentage of the model increases, the percentage of inputs passed through reduces. Moreover, the decrease in percentage of inputs tends to be the highest in the early stages of the network. This is because the accuracy of the model while using only the initial layers is still high enough to accurately classify most inputs. As a result there is an exponential-decay-like trend in the data.

To analyze the changes in speed as the percentage of the model increases, Graph 3 will be used.

Graph 3: Speed (as % of maximum speed -- speed of 1 layer of model) vs. % of Model that Inputs Passed through.

In Graph 3, there is a visible, negative correlation between the speed of forward propagation and the percentage of the model that the inputs are passed through. However, there are certain outliers in the data that do not suggest a negative trend.

In the case of the MNIST Handwritten Digits dataset, there appears to be a consistent downwards trend in the beginning 40% of the model, where the speed decreases from 100% of the initial speed to 94%. However, in the next 10% of the model (the next layer), the speed increases back to 100% of the initial speed. After this, the speed consistently decreases to 90%. The sudden increase in the speed at the 50% mark of the model is an outlier. However, it could be attributed to the complexity of the network and the efficiency of the CPU/TPU. It is possible that going through certain layers with certain parameters to reduce the size of the input and then reaching a fully connected layer requires fewer computations than passing a large input from the earlier stages into a fully connected layer. Hence, the speed increases. Apart from this, the speed does decrease as the percentage of the model used increases. This is intuitively correct as it generally takes more time to pass inputs through a greater number of layers.

For the Horses or Humans dataset, there is a trend similar to that of the previous dataset. There is an overall negative correlation, which is in accordance with the theory of the NLBN architecture. However, the speed of forward propagation at the 20%, 30%, and 40% mark

increases in comparison to the speed previous to these points. The cause of these outlier points may be similar to that of the MNIST Handwritten Digits dataset. The number and size of the computations required for these specific layers is less, in comparison to other layers. Hence, the speed has increased. Besides these data points, there is a clear trend between the independent and dependent variable. As the percentage of the model that the inputs pass through increases, the speed of the model decreases. In this case, it decreases from 100% the initial speed to 83%.

Finally, there is a stronger trend for the Colorectal Histology dataset. There are no major outliers, and the speed steadily decreases from 100% the initial speed to 94%, which is, again, in agreement with the theory behind the NLBN architecture.

From this analysis of Graph 3, it is shown that there is a negative correlation between the percentage of the model that inputs passed through and the speed of forward propagation. There are certain outliers, but it is possible for them to be due to the specific structure of the NLBN and the resulting computations. All in all, the NLBN architecture can increase the speed of forward propagation by reducing the number of layers (the percentage of the model) that the inputs pass through.

7. Results

From Tables 1 and 2 and the analysis of Graphs 1, 2, and 3, the performance of the NLBN can be analyzed and depicted in Graph 4.

Graph 4: The % Change in Speed and Change in Accuracy of the NLBN compared to that of the CNN.

For all three dataset, it is shown that the speed of the NLBN's predictions is greater than that of the CNN (standard model), but the accuracy of the NLBN is less than that of the CNN.

From Graph 1 and 2, it is evident that the accuracy of the NLBN architecture is less than that of the CNN. Graph 1 and the respective analysis depict how increasing the percentage of the model that inputs pass through causes an increase in the accuracy of the predictions. Moreover, Graph 2 and the respective analysis highlight how the NLBN architecture tends to utilize only the beginning of the models (a small percentage of the model). Through deduction, the NLBN will tend to have a lower accuracy than the CNN as shown in Graph 4. However, because the early stages of the model can make quite accurate predictions, the decrease in accuracy of the NLBN is low (0.2% - 3.4%).

From Graph 2 and 3, it is shown that the speed of the NLBN is greater than the speed of the CNN. Graph 3 establishes the negative correlation between the speed of the model and the percentage of the model used. The percentage of the model used is less in the NLBN as it does not perform forward propagation through the entire network for all inputs the way the CNN does. This is shown in Graph 2 and its analysis. Hence, as seen in Graph 4, the NLBN makes faster predictions in comparison to the CNN (an increase in speed from 6.8% - 22.2%).

8. Conclusion

The NLBN architecture reduces the time taken for forward propagation per image or training example. This results in an increase in the propagation speed as the models tend to exit the forward propagation of inputs at earlier layers, reducing the number of computations.

The accuracy tends to decrease, due to the use of rejection layers to exit forward propagation midway which could possibly reduce recall and precision. This is because labels may be predicted inaccurately in rejection layers with high confidence, thus increasing the number of false positives and false negatives predicted by the NLBN model.

Overall, the NLBN tends to increase the forward propagation time, and will most likely decrease accuracy by a percentage relatively less than the percentage increase in speed. The exact results heavily depend on the application, model structure, and dataset features. For example, the Colorectal Histology dataset had the least decrease in speed, possibly because its images (of cells) are relatively complex and require more layers (percentage) of the NLBN architecture to be used for an accurate prediction. The effectiveness of the NBLN can be determined by the significance of each feature of the model; these features include precision, recall, speed, CPU load, and more.

9. Further Scope

9.1 Further Improvements

This research delves into the theoretical and preliminary implementation of this newly proposed NLBN. It has been implemented in Google COLAB using TensorFlow and Keras packages.

Because it is new, the TensorFlow and Keras libraries are not highly efficient for the application of this architecture, considering the abundance of if-statements and the lack of matrix multiplications for the coded prediction method.

Furthermore, the confidence levels for the rejection layers were fixed, rather than trained. If they were trained, it is likely that the accuracy and forward propagation speed would increase.

Additionally, the datasets used in this research are considered to be relatively easy to train on while attaining a high accuracy. Hence, for these datasets, the accuracy may not significantly depend on the complexity of the model. Additionally, these datasets have an even distribution of data, which is not always applicable. In real-world scenarios, especially those that require real time applications, there tend to be many more true negatives which the model identifies. For the NLBN, it is expected that true negatives are easier to predict, so the architecture will pass through fewer layers, making it faster in practice.

This NLBN model architecture can be further improved through more investigation and can be deployed in more applicable use cases, such as in smart security cameras or in autonomous vehicles.

9.2 Potential & Future Applications

This research ventures into the beginnings of deep learning architectures that do not follow the traditional method of forward propagating through all the layers in a model. The NLBN lays the ground for other potential neural network architectures that may learn to skip or propagate through layers to improve performance in terms of accuracy, CPU load, and speed. In a more intuitive sense, the next steps are to teach deep learning models to be more effective and efficient using deep learning. Analogous to human perception, this idea is similar to humans thinking and understanding their thought process and learning how to think in the future.

The NLBN architecture requires more memory and more time to train, because it has extra rejection layers. However, the impact of this downside is based on the resources, circumstances, and limitations of developers for the given applications.

10. Acknowledgements & References

I would like to thank Dr. Ronjon Nag for supporting me and guiding me through this research.

- [1] Attention is All You Need - 07/2021: <https://arxiv.org/pdf/1706.03762.pdf>
- [2] Transformers in Vision: A Survey - 10/2021: <https://arxiv.org/pdf/2101.01169.pdf>
- [3] Transformers in CV - 10/2021: <https://towardsdatascience.com/transformer-in-cv-bbdb58bf335e>
- [4] Entropy-Aware I/O Pipelining for Large-Scale Deep Learning on HPC Systems - 10/2021
<https://ieeexplore.ieee.org/abstract/document/8526881>
- [5] Keras API - 07/2021: <https://keras.io/api/>
- [6] TensorFlow Libraries - 07/2021: <https://www.tensorflow.org/resources/libraries-extensions>
- [7] Tensorflow Datasets - 07/2021: <https://www.tensorflow.org/datasets/catalog/>
- [8] MNIST Handwritten Digits Dataset - 07/2021:
https://web.stanford.edu/~hastie/CASI_files/DATA/MNIST.html
- [9] Neural Layer Bypassing Network (NLBN) - MNIST Digits Dataset:
https://colab.research.google.com/drive/1KiFhmRnmuasj_oTUNWmncqWiuGkOOsS0?usp=sharing
- [10] Horses or Humans Dataset - 07/2021:
https://www.tensorflow.org/datasets/catalog/horses_or_humans
- [11] Neural Layer Bypassing Network (NLBN) - Horses or Humans Dataset:
https://colab.research.google.com/drive/1SEvW_YxPNvFXx163PdvPADS4L7ioqcxN?usp=sharing
- [12] Colorectal Histology Dataset - 07/2021:
https://www.tensorflow.org/datasets/catalog/colorectal_histology
- [13] Neural Layer Bypassing Network (NLBN) - Colorectal Histology Dataset:
<https://colab.research.google.com/drive/17ml4Xkn01lle5XoSt0oAdCVJ2O1DiX2y?usp=sharing>