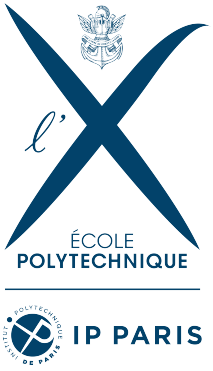


École Polytechnique

BACHELOR THESIS IN COMPUTER SCIENCE

Gauging the Prediction Power of Artificial Neural Networks

Author:

Yassine Turki, École Polytechnique

Advisor:

Leo Liberti, LIX, École Polytechnique

Academic year 2023/2024

Abstract

This thesis is an attempt to prove or disprove a conjecture about the quality of predictions of an Artificial Neural Network (ANN) based on the node values during training of this network. Specifically, we investigate the correlation between the percentage of node activations that fall into an interval computed from the values of these nodes during training and the accuracy of the prediction. We start by looking at a simple interval, then proceed into trimming it using percentiles to make it narrower. Furthermore, we look at the distance from the mean of training activations of the activations for an unseen input. We also implement a clustering approach using Gaussian mixture models, as well as the investigation of the impact of the addition of a noisy class. Finally, we narrow down the recording of training activations by separating them for each class and computing ranges for each specific class. While the first approaches did not show that a correlation exists, this last one indicated that there was a somewhat strong correlation between number of node activations that fall in the range for their class and the accuracy of the ANN. We highlight that our results are purely empirical and are done for a relatively simple feed-forward network for the MNIST dataset. In order to obtain further results, a generalization on other networks and datasets, as well as mathematical proofs, are required.

Contents

1 Introduction and Background

1.1 Introduction

Artificial Neural Networks (ANNs) have emerged as a transformative force in Machine Learning, revolutionizing how computers learn and perform tasks that were previously thought to be limited to human cognition. Neural networks are computational models made up of interconnected nodes, or "neurons," that are organized in layers. They are inspired by the structure and functionality of the human brain. These networks have the ability to detect patterns, make predictions, and derive insights from complex datasets, much like the cognitive processes of the human mind.Neural networks, which were first proposed in the 1940s [10], have improved dramatically over the years, thanks to breakthroughs in computer hardware, algorithmic innovations, and the availability of massive amounts of data. Today, they are used in a variety of areas such as image [7, 8] and audio recognition [3], natural language processing [16], autonomous cars [2], medical diagnostics [15], and financial forecasting [13].

These networks are able to predict outcomes that need to be as close to the ground truth as possible. For example, if we work on a classification problem, the network should be able to correctly predict the class of an image when trained correctly. It is then easy for humans to check the output and confirm that the network is accurate. However, there arises a problem when we do not have access to the ground truth. If we would like to forecast the volume of natural gas available in 2050 using a neural network, then we would not have the actual ground truth to back the network's prediction. This paper aims to investigate a way to characterize the quality of a network's prediction without having access to the ground truth. More importantly, this characterization would focus on the network's parameters, in particular the different node activation values throughout the training of this network, in order to assess the robustness of an Artificial Neural Network.

1.2 Background

1.2.1 Formal definition

An Artificial Neural Network is a Machine Learning algorithm used to predict outputs based on unseen inputs. In other words, it learns a function (ground truth) $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ where n is the dimension of the input vector and m is the dimension of the output vector. Formally, it is defined as a triplet $\mathcal{N} = (G, T, P)$ where

- $G = (V, A)$ is a directed graph, with V the set of nodes and A the set of arcs of the ANN
- T is a suitable dataset for our problem
- $P = (\mathcal{T}, \mathcal{E})$ denotes a pair of optimization problems, with $\mathcal{T}$ denoting the training problem and $\mathcal{E}$ the evaluation problem

In more detail, $\mathcal{T}$ refers to obtaining the parameter values, which in the case of ANNs are the arc weights w and biases b, using the training set T. $\mathcal{E}$ approximates the ground truth f given input values for f. We denote the node activations, also known as node values or node weights by $v \in V$. Let $j \in V$ a node and let $N \setminus \{j\}$ denote the nodes connected to j through arcs $(i, j) \in A$. The activation of j, v_j, is computed as follows:

$$v_j = \phi\left(\sum_{i \in N \setminus \{j\}} w_{ij} v_i + b_j\right) \tag{1}$$

with ϕ denoting an activation function (e.g. sigmoid), w_{ij} is the weight of the arc (i, j), i is the node connected to j through the arc (i, j) and b_j is the bias associated with node j.

1.2.2 Activation functions

To get the value of a node, we apply a function ϕ called the activation function. This function is generally non-linear, which allows the network to learn from complex data and become more powerful than simple regressions.

Sigmoid activation: The sigmoid activation function takes an input in $\mathcal{R}$ and returns a probability ranging from 0 to 1.

$$\phi_{\text{sigmoid}}(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

ReLU activation: The Rectified Linear Unit (ReLU) activation function is one of the most popular activation functions used in modern Neural Networks. It is defined as follows:

$$\phi_{\text{ReLU}}(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Softmax: The softmax function is generally used at the last layer of the ANN to compute the probability distribution of the event over n different events. It is useful when performing multi-class classification.

$$\phi_{\text{softmax}}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}, \text{ where } k \text{ denotes the number of classes} \tag{4}$$

It occurs sometimes that the exponential of an output value can lead to a very large number. Because of the limits of computer floating-point representations, the use of that number for further computations, such as the calculation of the loss, may cause numerical instability.

Log Softmax: To remedy the problem mentioned above, the Log Softmax function has been introduced. It involves computing the logarithm of the softmax function.

$$\phi_{\text{Log\_softmax}}(z) = \log(\phi_{\text{softmax}}(z)) \tag{5}$$

1.2.3 Related work

We are interested in characterizing the robustness of an ANN. However, when speaking about robustness, the literature mostly focuses on the network's ability to correctly predict adversarial samples [17], and this generally does not involve an in-depth study about the impact of node values during training on the prediction power of the ANN. This thesis is the continuation of a previous work done by students last year. They analyzed shallow networks; useful for predicting somewhat simple functions such as $sin(x)$. The results found was the distance from the mean of node values had some impact on the quality prediction. However, it is difficult to generalize a result given the small size of the network and the simplicity of the dataset. In this work, we will first investigate the need for such measure, and try to generalize or disprove it using a larger network and a more complex dataset.

1.2.4 Dataset

In this thesis, we solely focus on the MNIST dataset [9]. The MNIST dataset contains 70,000 images of hand-written digits, as well as their correct classification. It is one of the most popular classification datasets for Deep Learning. The images are 28 by 28 greyscale images, with normalized pixel values from 0 to 1. The MNIST dataset is divided into two parts. The first part contains 60,000 images that will be used as our training set. The second part contains 10,000 images that will be used for testing our network. The digits in the test images were

written by different people than in the training set. This gives us confidence that our network will not overfit our data, as it needs to predict digits that are not identical to what it has been trained on.

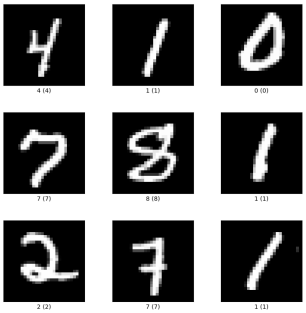


Figure 1: Sample MNIST images

1.2.5 Our models

In this thesis, we have only focused on simple feed-forward architectures, meaning we will not be using convolutional layers, dropout layers, recurrent architectures, etc. We have used three models:

1. Our first model, $model_1$, comprises of 2 hidden layers with size 128 nodes and 64 nodes all using the sigmoid activation function
2. The second model, $model_2$, has the same architecture than the first one, but uses the ReLU activation
3. The third model, $model_3$, is smaller, with only one hidden layer composed of 128 nodes. The activation function used for this model is sigmoid

The three models use Cross-Entropy as a loss metric. For the optimizer, the first two models will use Stochastic Gradient Descent, whereas the third model will use the ADAM optimizer [5]. The first two models will be trained on 15 epochs with a batch size of 64. As for the third model, the number of epochs is 6 with batch size of 128. When trained then evaluated on the testing set, $model_1$ reaches 94% accuracy, $model_2$ 97% accuracy and $model_3$ reaches an accuracy of 97%. When not specified, the model used will be $model_1$. Although its accuracy is the lowest between the three, we will find that sigmoid gives us better results and will prefer a larger network than a small one.

1.2.6 Correlation Coefficients

In this report, we are interested in providing a characterization of the network's robustness. In other words, we would like to see how much the quality of the predictions is correlated with our characterization. To do this, we calculate three well-known correlation coefficients: the Pearson's product moment correlation coefficient, the Spearman's rank correlation coefficient [11], and Point-biserial correlation coefficient [6].

Pearson's Product Moment Correlation Coefficient: For a correlation between variables x and y, Pearson's product moment correlation coefficient r is calculated as follows [14]:

$$r = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{[\sum_i^n (x_i - \bar{x})^2]\,[\sum_i^n (y_i - \bar{y})^2]}} \tag{6}$$

where x_i and y_i are the values of x and y for the i-th sample. Pearson's coefficient is used when the two variables follow a normal distribution. Moreover, it is very sensitive to outliers, which means that extreme values can exaggerate or diminish the strength of the relationship. Moreover, the coefficient indicates a linear relationship between the two variables, meaning that the increase/decrease of one variable when the other increases/decreases is constant.

Spearman's Rank Correlation Coefficient: When the values are not normally distributed or do not follow a linear relationship, a useful correlation coefficient is Spearman's rank correlation coefficient r_s [1]. Instead of using the raw samples, we will rank them between themselves and use those ranked values to calculate the correlation. For variables x and y, it is calculated as follows:

$$r_s = 1 - \frac{6 \sum_i^n d_i^2}{n(n^2 - 1)} \tag{7}$$

where d_i is the difference in ranks for x and y.

Point-biserial Correlation Coefficient: When our target y is binary (for example, when it is 1 when the prediction is correct and 0 otherwise), it is more useful to look at the Point-biserial correlation coefficient r_{pb} defined as follows:

$$r_{pb} = \frac{\bar{x}_2 - \bar{x}_1}{sd_x} \cdot \sqrt{\frac{n_1 \cdot n_2}{n^2}} \tag{8}$$

where

- $\bar{x}_2$ denotes the mean value of the group who failed (misclassified samples)
- $\bar{x}_1$ denotes the mean value of the group who succeeded (accurate predictions)
- sd_x is the standard deviation of x
- n_2 denotes the number of misclassified samples
- n_1 denotes the number of correct predictions
- n is the total number of samples ($n = n_1 + n_2$)

2 First Experiment

2.1 Original conjecture

The first experiment provided the motivation to start this thesis. For each node $i \in V$ in our ANN, we record the value of its activation after each training sample. When the network is fully trained, we will have an array of 60,000 values * number of epochs for each v_i. We then compute the range $R_i = [v_i^L, v_i^H]$ where v_i^L (resp. v_i^H) denotes the minimum (resp. the maximum) activation that the node i took during training. After the range has been computed for each node, we feed an unseen input from the testing set and count the number of node values v_i for this input that fall within their respective range R_i. We run this experiment for all 10,000 testing samples and record the percentage of nodes that fall within their respective range for each sample, as well as a boolean variable that looks at the prediction of the model compared to the ground truth. Our guess was that if the model predicts an unseen input correctly, then the majority of node activations for that input will be within their range. Conversely, if an input is misclassified, then we assumed that the percentage would be low. We first plot the percentages of nodes falling in range for each sample and if it is classified correctly or not (figure 2).

Next, we test the correlation between the accuracy of prediction and percentages. Since our target is binary (0 if the sample is misclassified, 1 otherwise), we decide to use the Point biserial correlation.

(a) Percentages from 0 to 100

(b) Percentages from 88 to 100

Figure 2: Percentages of node values in range for each test sample (Sigmoid)

Figure 2 shows that most of the testing samples have their associated node values being 100% in their range R_i. Moreover, the calculated coefficient is equal to 0.0077 with a p-value of 0.44. These results indicate that there is almost no correlation between accuracy and the node values falling into their range. We can however not conclude anything, given that the p-value is greater than 0.05 and thus, our result is not statistically significant.

We decided to extend this experiment and try it with a different activation function, namely, the ReLU activation, using $model_2$ (figure 3).

(a) Percentages from 0 to 100

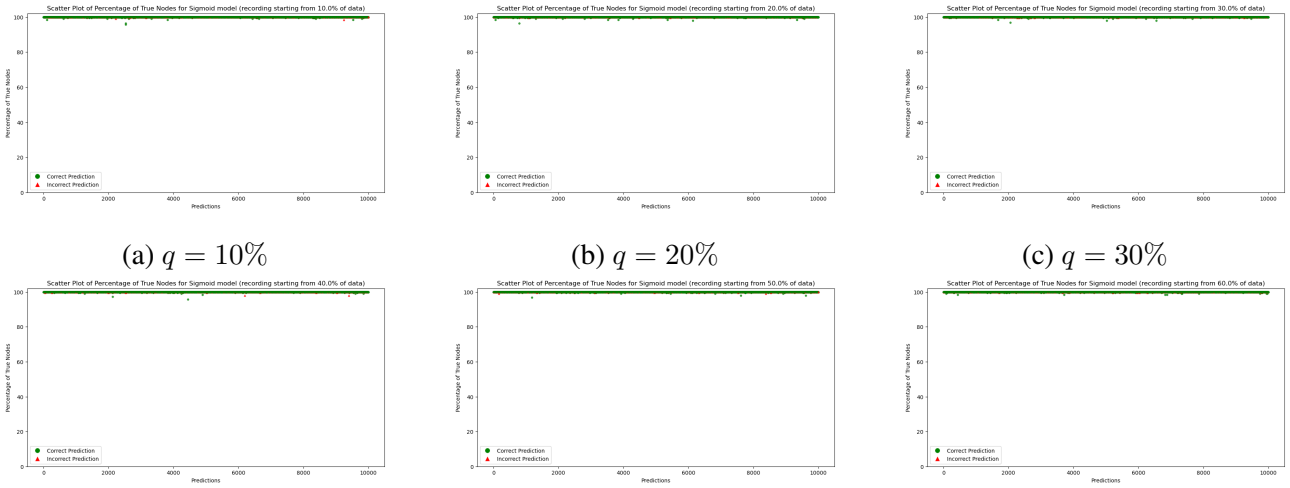
(b) Percentages from 80 to 100

Figure 3: Percentages of node values in range for each test sample (ReLU)

We can see that changing the activation function does not affect the plot much. We still have the same phenomenon: the node values are almost always in the 100% range. Moreover, the correlation coefficient is equal to 0.018 with a p-value of 0.07. As the p-value is greater than 0.05, our result i again statistically insignificant. Since these plots and correlation did not provide much insight to the analysis compared to the plots obtained using the sigmoid function, we will only use the sigmoid function for the rest of this report.

2.2 Postponing the recording of activations

We make the hypothesis that the first node values from training will not be very significant, as they will probably result in a high loss. Since we are using gradient descent, we assume that in the beginning, our gradient will be far from the optimum, and thus, w need a lot of training samples to reach a minimum for our loss. Therefore, we perform again the same experiment, but we start recording the node activations only after we train the network with a percentage q of the training set.



(a) $q = 10\%$ (b) $q = 20\%$ (c) $q = 30\%$

(d) $q = 40\%$ (e) $q = 50\%$ (f) $q = 60\%$

Figure 4: Percentage of nodes in range with activations recorded after q percent of the trainset

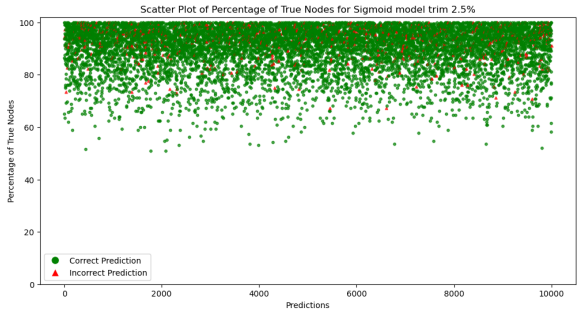
q	Point biserial correlation	p-value
10%	0.00635	0.5256
20%	0.00199	0.8419
30%	0.007089	0.4784
40%	0.03408	0.00065
50%	0.02193	0.0283
60%	0.01043	0.29695

Table 1: Correlation and p-value for each q

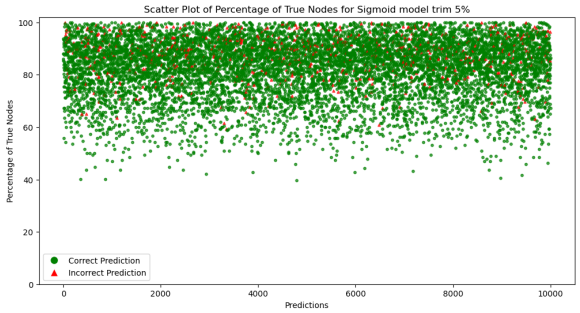
We can see from figure 4 and table 1 that our hypothesis is false. Recording the node values after a certain percentage of training data does not show any correlation between the percentage of activation in the ranges R_i and the quality of prediction. Moreover, we encounter that most of the percentages are in the 100% area.

2.3 Removing highest and lowest activations

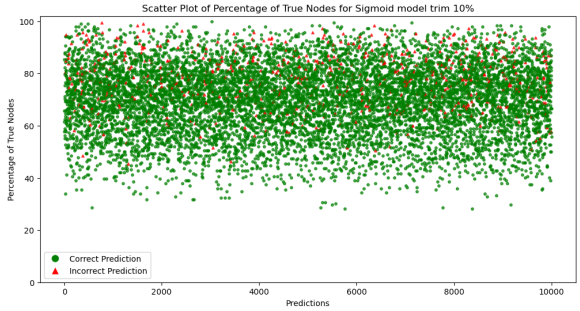
We try to remedy the problem of this clustering around 100% by trimming the node activations. For every node v_i, after computing the activations throughout training, we sort the list of activations and delete the bottom p-percent and the top p-percent from the list, with $p \in \mathcal{R}$. After that, we find R_i and run the experiment with the new ranges, as depicted in figure 5.



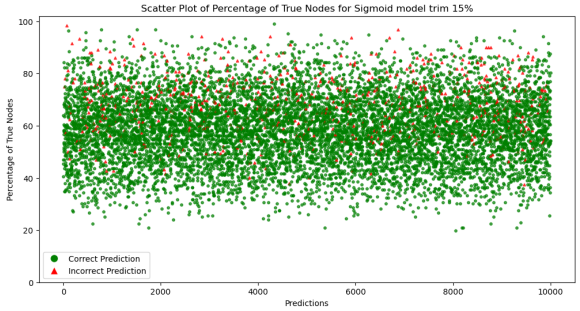
(a) $p = 2.5$



(b) $p = 5$



(c) $p = 10$



(d) $p = 15$

Figure 5: Percentages of node values in range after trim (Sigmoid)

Now we see that the percentages are becoming more sparse as we increase the value of p. However, there is still no clear pattern based on the accuracy of the prediction. Table 2 shows the point biserial correlation coefficient for each p, as well as its associated p-value.

p	Point biserial correlation	p-value
2.5%	-0.0796	1.5×10^{-15}
5%	-0.1103	1.9×10^{-28}
10%	-0.1497	3.1×10^{-51}
15%	-0.1759	2.6×10^{-70}

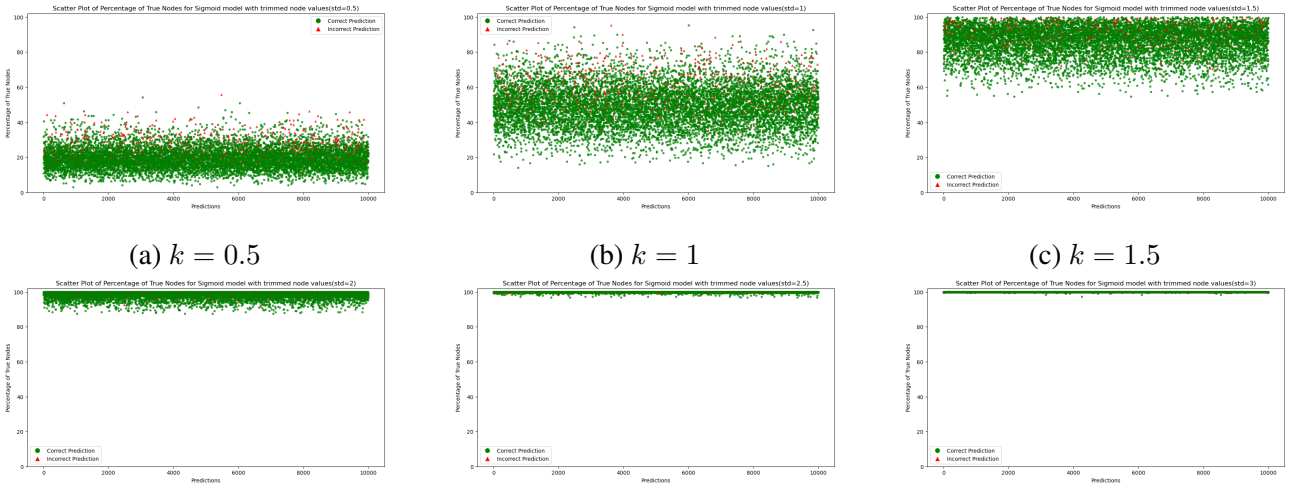
Table 2: Correlation and p-value for each p

As we can see, the more we delete node activations from our recorder list, the higher the magnitude of the correlation. It appears we have a slight negative correlation. As the p-values are very small, we have a good indicator that our results are becoming more significant. This leads us to investigate different ways to trim down our recorded node values.

3 Second Experiment

3.1 Measuring with the accuracy of the prediction

In this experiment, we implement a new type of criterion based on distance from the mean of activations for each node. We compute the activations for each node i during training and define again $R_i = [v_i^L, v_i^H]$, only this time, $v_i^L = \bar{v}_i - k * sd_i$ and $v_i^H = \bar{v}_i + k * sd_i$, where $\bar{v}_i$ is the mean of the activations for node i, sd_i is the standard deviation for that node and k is a constant. We plot the percentages in Figure 6.



(a) $k = 0.5$ (b) $k = 1$ (c) $k = 1.5$

(d) $k = 2$ (e) $k = 2.5$ (f) $k = 3$

Figure 6: Percentage of nodes in range after trimming using different values of k

In table 3, we display the percentage of activations removed for each k, as well as the Point Biserial correlation and its associated p-value.

k	Percentage of activations removed	Point Biserial correlation	p-value
0.5	72%	-0.261	$5 \cdot 10^{-155}$
1	40%	-0.194	$9 \cdot 10^{-86}$
1.5	11%	-0.149	$4 \cdot 10^{-51}$
2	1.3%	-0.115	$4 \cdot 10^{-31}$
2.5	0.05%	-0.036	$3 \cdot 10^{-4}$
3	$6 \cdot 10^{-6}$ %	0.010	0.305

Table 3: Correlation and p-value for each k

- When $k = 0.5$, we observe a medium negative correlation between node activations and accuracy. Furthermore, the number of node activations falling in range is relatively low compared to what we observed before. This is due to the fact that we remove more than 72% of values for each R_i, which makes the new R_i extremely narrow
- When $k = 1$, the percentages are a lot more sparse. We have a large cluster of percentages ranging from 30 to 70%. The correlation is still negative
- When $k = 1.5$ and $k = 2$, our percentages start looking similar to when we did not trim our training activations. However, the percentages are still sparse, and we could capture a slight negative correlation
- When $k = 2.5$ and $k = 3$, we go back to the case where most of the node values are in their range, and the correlation starts becoming close to 0

As our p-values are extremely small, we can infer that these results are significant, and that distance from the mean allows us to deduce some results about the correlation of accuracy and node values in the range. However, our correlation coefficients are still small and insufficient to be able to form a valid conjecture. We now decide to look at the actual outputs of the network rather than the predicted class, i.e., the probabilities from the softmax function.

3.2 Measuring with the probabilities from softmax

Instead of looking at the quality of the prediction in a binary way (i.e., is the sample correctly classified or not), we look at the probability that the softmax function outputs for the prediction the neural network will make. We perform the same experiment as above, only now, given that we are dealing with probabilities, we will compute the Pearson and Spearman correlation coefficient instead of the point biserial correlation.

(a) $k = 0.5$ (b) $k = 1$

(c) $k = 1.5$ (d) $k = 2$

(e) $k = 2.5$ (f) $k = 3$

Figure 7: Percentage of nodes in range with probability of prediction using different values of k

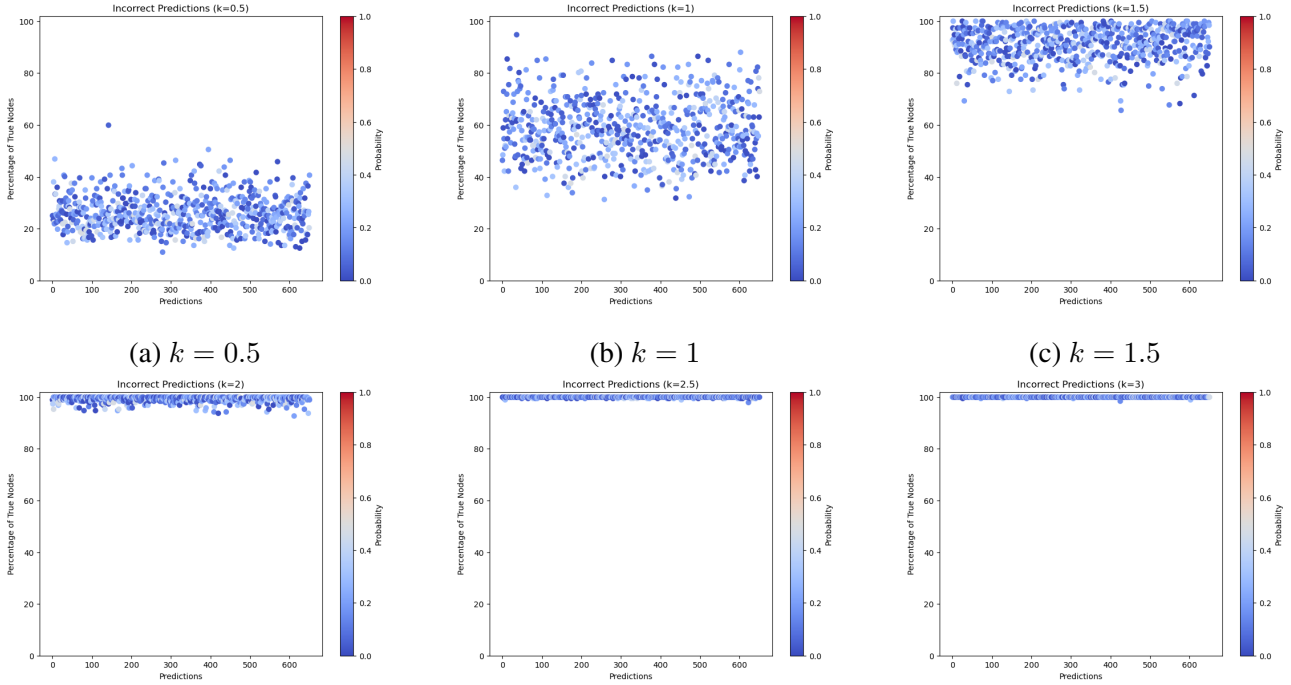
k	Pearson	p-value Pearson	Spearman	p-value Spearman
0.5	-0.5108	0.0	-0.6271	0.0
1	-0.4354	0.0	-0.5892	0.0
1.5	-0.3287	1.66×10^{-250}	-0.5121	0.0
2	-0.2601	2.64×10^{-154}	-0.4605	0.0
2.5	-0.1786	1.96×10^{-72}	-0.3259	4.27×10^{-246}
3	-0.0024	0.8096	-0.0112	0.2617

Table 4: Correlation and p-value for each k using softmax probabilities

We observe a pattern in the plots of incorrect predictions (figure 7) for $k \in \{0.5, 1, 1.5\}$. The more the network is sure of its prediction (i.e. when the dot is red), the less the percentage (i.e., the smaller the number

of nodes activation for the sample fall in their range). Moreover, the negative correlation coefficients (table 4), which have a relatively high magnitude in this experiment, allow us to confirm this conjecture. Thanks to the small p-values, this is a statistically significant result that needs to be investigated further.

In order to confirm this idea, we plot the probability of the correct class (instead of the maximum probability for a sample) only for the incorrect predictions.



(a) $k = 0.5$ (b) $k = 1$ (c) $k = 1.5$

(d) $k = 2$ (e) $k = 2.5$ (f) $k = 3$

Figure 8: Percentage of nodes in range with probability of ground truth using different values of k for incorrect predictions

k	Pearson	p-value Pearson	Spearman	p-value Spearman
0.5	-0.0979	0.0124	-0.0681	value_s
1	-0.0802	0.0406	-0.0436	value_s
1.5	-0.0348	3.75×10^{-1}	-0.0138	value_s
2	-0.0826	0.0349	-0.0519	value_s
2.5	0.0116	0.7666	0.0370	value_s
3	-0.0120	0.7588	-0.0222	value_s

Table 5: Correlation and p-value for each k (probability of ground truth)

Our conjecture is only true for $k = 0.5$ based on the plots (figure 8). However, the correlation coefficients (table 5) are all close to 0. especially the Spearman coefficients. Thus, we cannot conclude much from this analysis. However, we have the result from table 4 that states that the higher the percentage of nodes in range, the lower the probability of predicting the wrong class.

4 More creative approaches

In this section, we showcase some creative approaches to investigate further the impact of node values on the prediction.

4.1 Additional unseen input

Although the test images are different from the training ones, we still observe some similarities. In this subsection, we create 10 new images with a thin brush and 10 others with a thick brush, one of each category for each class. These images will serve in the evaluation problem.

(a) Image with thin brush

(b) Image with thick brush

Figure 9: Example of new images created

Out of the 10 images created with a thin brush, the network assigned the correct class to 5 of them. For the ones created with a thick brush, the network predicted only one image correctly. Despite this low accuracy, the percentage of node activations in range is quite high for all the samples. Using the thin images we always get 100% of nodes in range, whereas for the thick images, the lower bound is 99.61% . As for the correlation, when computing the point biserial correlation for all samples, we get a value of 0.2567 with p-value 0.2744. While the p-value is not significant, we can see that there is a small correlation between percentages and accuracy. This is partly due to the fact that the data made with the thin brush always reaches 100% of nodes in range and accounts for an accuracy of 50%. However, as we do not have enough samples, these results cannot lead to a conclusion.

4.2 Fast Gradient Sign Method

Although ANNs are designed to maximize accuracy, it is always interesting to look at how they behave when predicting wrong samples. As the dataset is simple, our network performs quite well on it, leading to very few misclassified samples out of the testing set. Therefore, to investigate further the network's behavior with misclassified inputs, we decided to use the Fast Gradient Sign Method (FGSM) [4] to generate adversarial examples. An adversarial is an input formed by applying a small perturbation to examples from the dataset, such that the model misclassifies the input with high confidence. Let θ be the parameters of the model, x the input, y the target, and $J(\theta, x, y)$ be the cost function that serves as loss during training. We add a small amount of noise based on the gradient of the loss with respect to the input.

The algorithm:

1. Compute the gradient: $\nabla_x J(\theta, x, y) = \frac{\partial x}{\partial J}$
2. Generate the Adversarial example: we calculate the perturbation η by taking the sign of the gradient, that is, $\eta = \epsilon \cdot sign(\nabla_x J(\theta, x, y))$ where ϵ is a constant that controls the magnitude of the perturbation. The higher the ϵ, the higher the perturbation and thus, the more misclassified samples we will get

3. Update the input: The new adversarial sample x_a is created by adding the perturbation η to the original sample x, that is, $x_a = x + \eta$

The goal of the algorithm is to craft adversarial examples that maximize the loss such that our model provides a false prediction while ensuring that the perturbation is small such that it remains imperceptible. Figure 10 displays some adversarial examples depending on the value of ϵ. The title of each image shows the "ground truth -> adversarial classification."

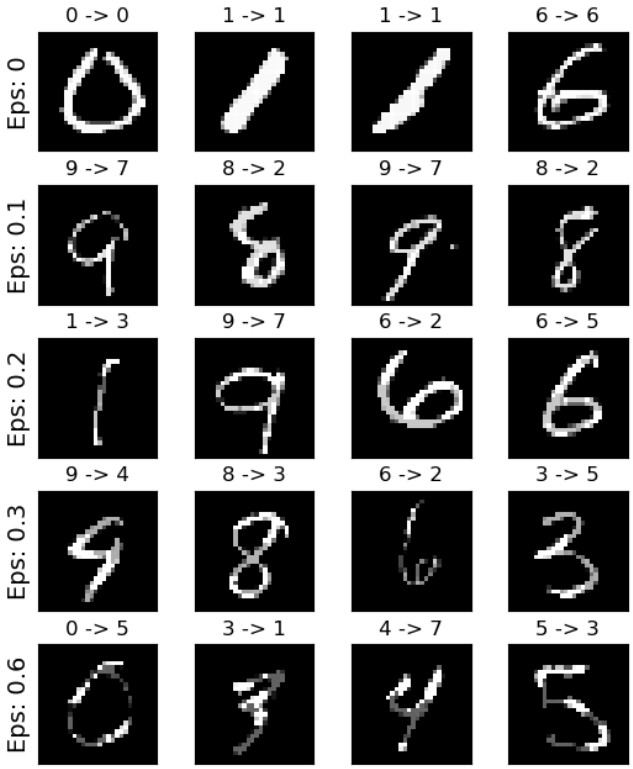
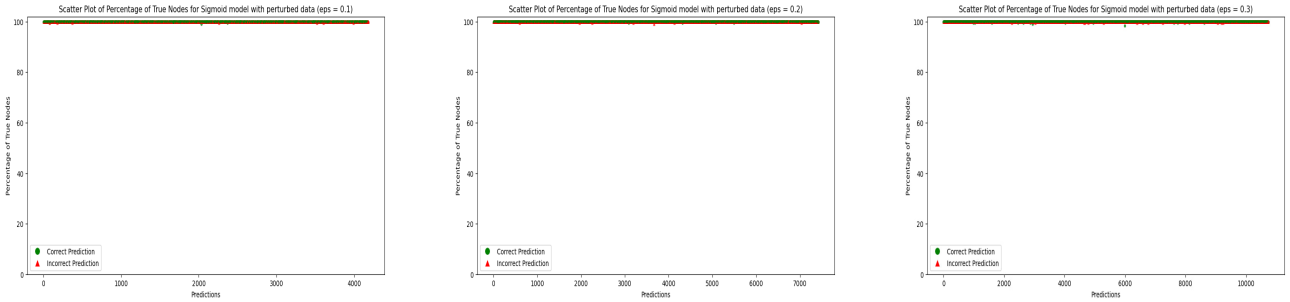


Figure 10: Influence of the value of ϵ on the adversarial samples

We notice that the effect of the attack on the samples is evident starting from $\epsilon = 0.3$. Nevertheless, it is quite trivial for humans to discern the digits. We first train the model with our original trainset and record the node activations. Similar to the first experiment, we compute the value ranges R_i for each node v_i. Then, we generate adversarial samples using FGSM for ϵ in $0.1, 0.2, 0.3$. For each ϵ, we obtain n_ϵ adversarial samples. We evaluate our model on these samples, as well as n_ϵ test samples from our original test set. We then plot the percentage of node values that fall into their range R_i for each sample (Figure 11. Then, we compute the Point Biserial correlation in Table 6.



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.2$ (c) $\epsilon = 0.3$

Figure 11: Scatter plot for different value ranges depending on different ϵ

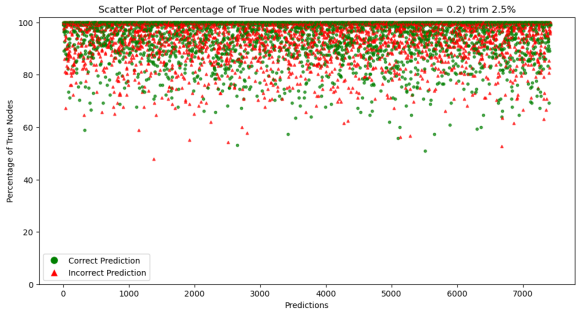
ϵ	$2n_\epsilon$	Point Biserial correlation	p-value
0.1	4182	0.0101	0.5110
0.2	7421	0.00987	0.39494
0.3	10738	-0.0077	0.42047

Table 6: Correlation and p-value for each ϵ

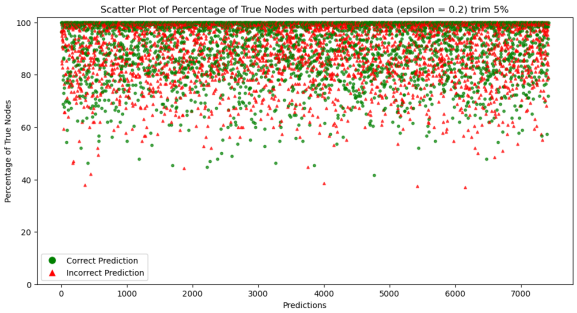
We can see that increasing ϵ does not really affect the percentage of node values falling into the range. Moreover, although statistically insignificant because of high p-values, the correlation coefficient is very small for all ϵ. We arbitrarily decide to continue our analysis with $\epsilon = 0.2$. We continue our investigation and trim the node activations recorded during training using a similar approach to our first experiment. For p in $2.5, 5, 10, 15$, we trim the highest $p\%$ and the lowest $p\%$ of the node values' activations recorded while training, then plot and calculate the correlation. We can see from the plots and tables below that the percentages are more sparse, but the correlation is close to 0, no matter the method we try.

p	Point biserial correlation	p-value
2.5%	-0.0031	0.786
5%	0.0064	0.581
10%	0.0087	0.453
15%	-0.0090	0.4362

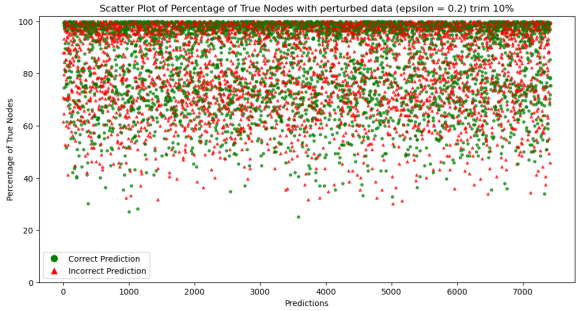
Table 7: Correlation and p-value for each p with adversarial samples



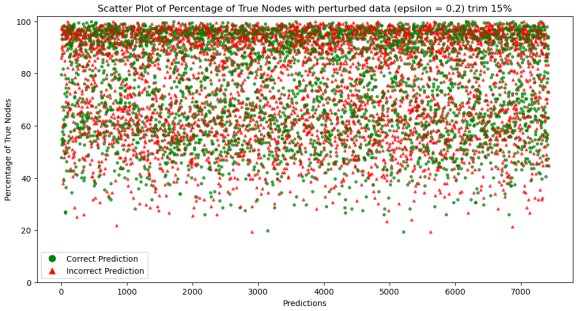
(a) $p = 2.5$



(b) $p = 5$



(c) $p = 10$



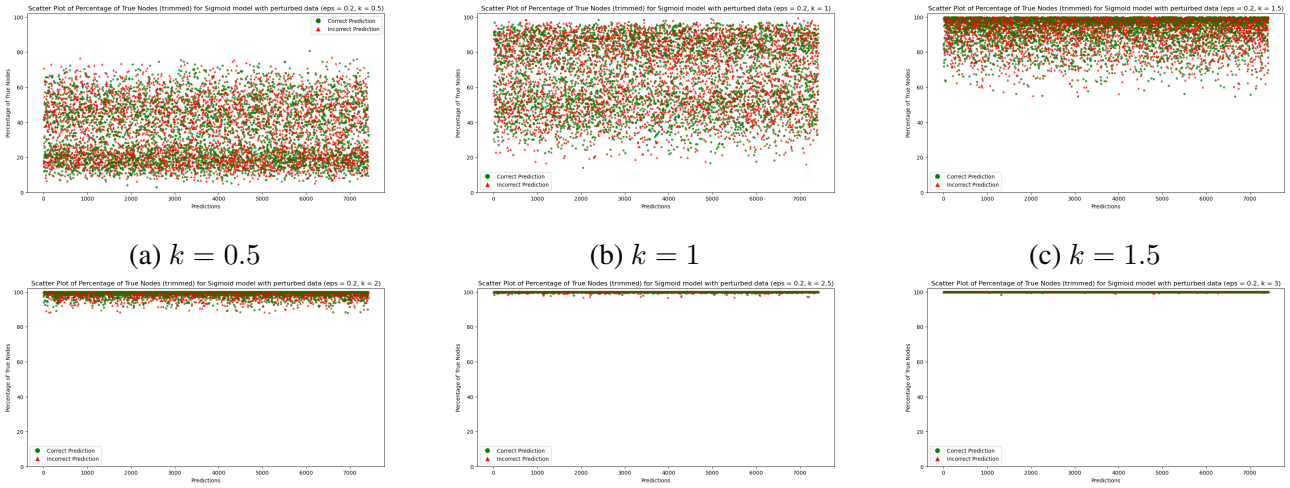
(d) $p = 15$

Figure 12: Percentages of node values in range after trim with adversarial images (Sigmoid)

We also try to trim the node activations using the distance from the mean and $k * std$.

k	Percentage of activations removed	Point Biserial correlation	p-value
0.5	72%	-0.020	0.08395
1	40%	0.0037	0.7482
1.5	11%	0.021	0.0722
2	1.3%	0.008	0.483
2.5	0.05%	-0.024	0.036
3	$6 \cdot 10^{-6}$ %	0.0126	0.2776

Table 8: Correlation and p-value for each k with adversarial samples ($\epsilon = 0.2$)



(a) $k = 0.5$ (b) $k = 1$ (c) $k = 1.5$

(d) $k = 2$ (e) $k = 2.5$ (f) $k = 3$

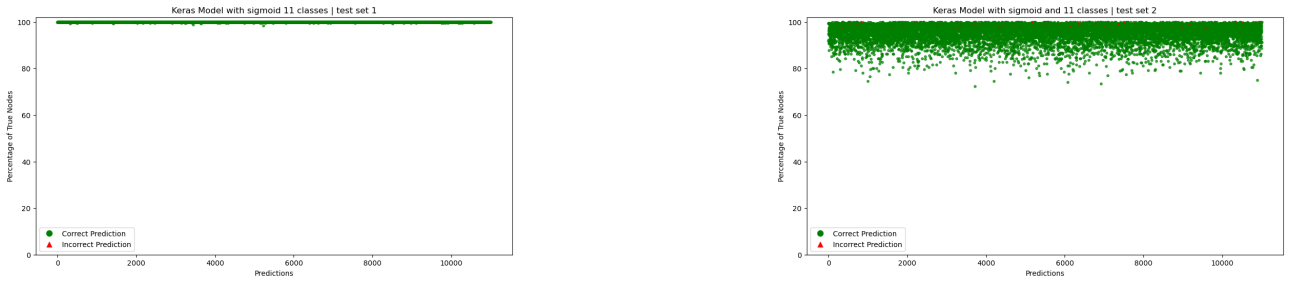
Figure 13: Percentage of nodes in range after trimming using different values of k with adversarial samples ($\epsilon = 0.2$)

Even with adversarial samples, the conjecture that high node percentages and better accuracy are correlated is not correct. However, as our p-values are high for some values of k, we fail to reject the hypothesis.

4.3 Additional class

We introduce an additional class to better understand our network's behavior when faced with noisy data. This concept, known as 'fooling images' [12], involves classifying input data that does not correspond to any recognizable pattern, such as a digit. To implement this, we augment our training set with 6,000 extra training samples. These samples are generated by uniformly sampling values from 0 to 1. As usual, we will record the node activation values during training. For the evaluation problem, we will proceed in performing two tests with two distinctive test sets:

1. The first set will consist of the usual test set from MNIST and 1,000 randomly generated images. These images consist of pixels ranging from 0 to 1, similar to the input patterns encountered during training
2. The second set will consist of the usual test set from MNIST and 1,000 random noise data. This data will be created as follows:
   - First, we perform a coinflip c, resulting in either $c = -1$ or $c = 1$
   - Next, we generate a random number, r, between 0 and 1
   - Each of the sample's values will be assigned a value of $c + r$, with random c and r for each value



(a) Test set 1 (b) Test set 2

Figure 14: Percentages of node values in range for each test set

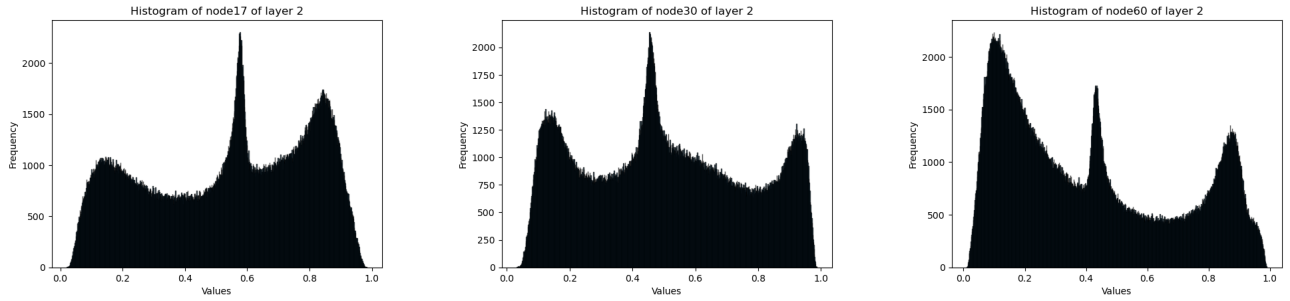
Testing set	Accuracy	Point Biserial Correlation	p-value
Test set 1	97%	0.0220	0.0208
Test set 2	99%	-0.0530	$2.58 \cdot 10^{-8}$

Table 9: Correlation using the two testing sets

While the accuracy with the second test set is slightly higher than for the first set, it appears that the percentage of nodes in the range is less cluttered around 100%. However, both test sets let us conclude that there is a very negligible correlation between accuracy and the number of node values in range.

4.4 Gaussian Mixture Models

We decided to narrow down our investigation to the last layer, as it is the one determining the output. Upon looking at the distribution of our node activations, we notice that we have some clusters that are forming (Figure 15).



(a) Node 17 (b) Node 30 (c) Node 60

Figure 15: Histogram of node activations during training for different nodes of layer 2

The plots allow us to visualize that the node activations have a multimodal distribution. We recall that a Gaussian distribution has a probability density function defined as follows:

$$p(x|\mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{9}$$

with μ denoting the mean and σ the standard deviation of the distribution. We assume now that each point x_n has been produced by some latent variable z that is a categorical variable representing d underlying distributions

from the data. Each point would be mapped to one specific distribution by considering z as a one-hot vector, e.g. if $z = (0, 0, 1)^T$ then our data point would belong to the third distribution. However, given that in reality, we do not a one-hot vector, we assign z with a probability distribution:

$$p(\mathbf{z}) = \boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_k)^T, \text{ with } 0 \leq \pi_i \leq 1 \text{ and } \sum_{i=1}^{d} \pi_i = 1 \tag{10}$$

Each of our data points belongs to all the distributions with a certain probability now. We will then use Gaussian distributions for each of these distributions. Each Gaussian would have its associated mean and standard deviation, and they would be mixed with each other using the coefficients π_i. This is the definition of a Gaussian Mixture Model. For a GMM with d Gaussian components, we write:

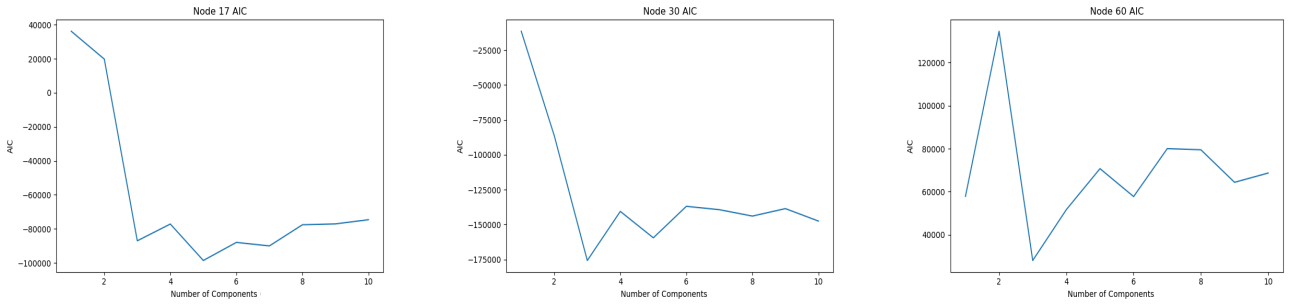
$$\mathcal{N}(\mu, \sigma^2) = \sum_{i=1}^{d} \pi_i \mathcal{N}(x|\mu_i, \sigma_i^2) \text{ with } 0 \leq \pi_i \leq 1 \text{ and } \sum_{i=1}^{d} \pi_i = 1 \tag{11}$$

Our next analysis will consist of fitting a Gaussian mixture model for each node in the last layer, extracting the different normal distributions in the multimodal distribution, extract approximately 95% of values for each normal distribution using the formula $\bar{x} \pm 2std$, where $\bar{x}$ denotes the mean and std the standard deviation. Each normal distribution will then give us a new list of values, for which we extract the range R_i as usual. Thus, for each node, given d the number of Normal distributions that the multimodal distribution contains, we will get d ranges. We then present the model with our testing set and see the percentage of node activation that falls within one of the d ranges for each node. Our first approach is to find the number of normal distributions for each node using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) scores. These scores are computed as follows:

$$AIC = -2 * MLE + 2p \tag{12}$$

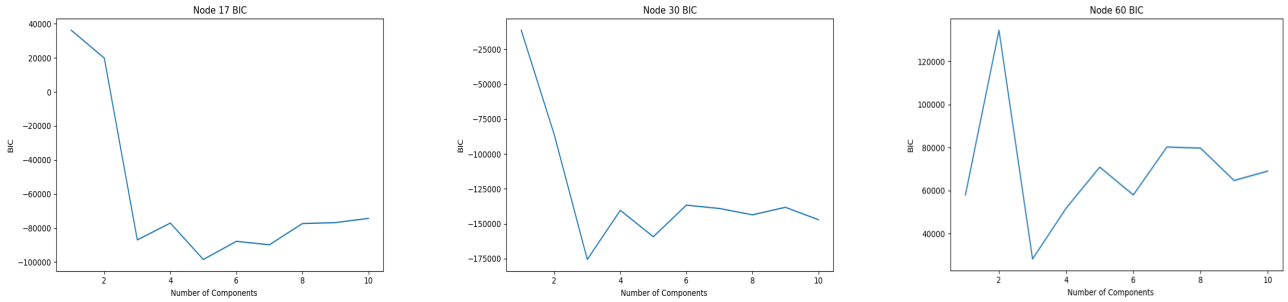
$$BIC = -2 * MLE + p * log(N) \tag{13}$$

where MLE denotes the maximum likelihood estimator, p the number of parameters in the model and N the number of activations. The lower the AIC (respectively BIC), the better the fit of our model. We can see that the two scores depend on a high MLE, but will be penalized by the number of parameters they have. In addition, we hypothesize that we could have 10 clusters, one for each class. Before testing that, we allow d to range from 1 to 10 and for each node, we will select the d that minimizes the AIC and BIC.



(a) Node 17 (b) Node 30 (c) Node 60

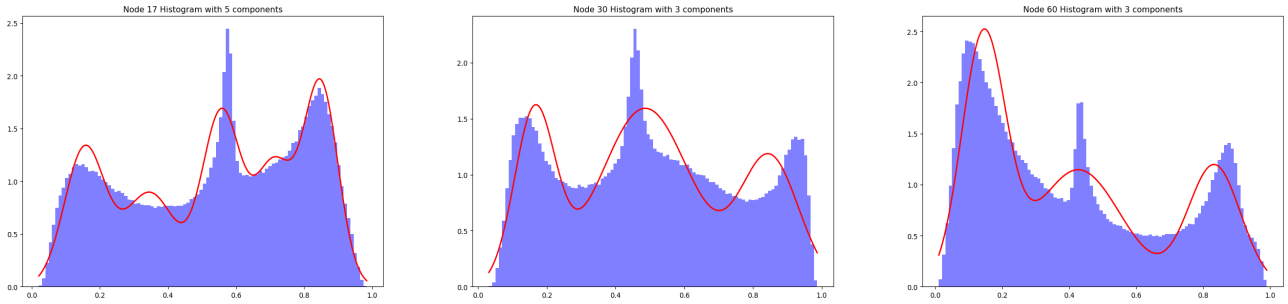
Figure 16: AIC for each d for different nodes



(a) Node 17 (b) Node 30 (c) Node 60

Figure 17: BIC for each d for different nodes

We can see from figure 16 and figure 17 that AIC and BIC are minimized for the same d_i for each node i, as their curve is quite similar. We check that conjecture for each node and assert that this is true. Now that we obtained the values d_i that minimize the scores, we take for each node the associated model and plot it next to the distribution of its activations in figure 18.



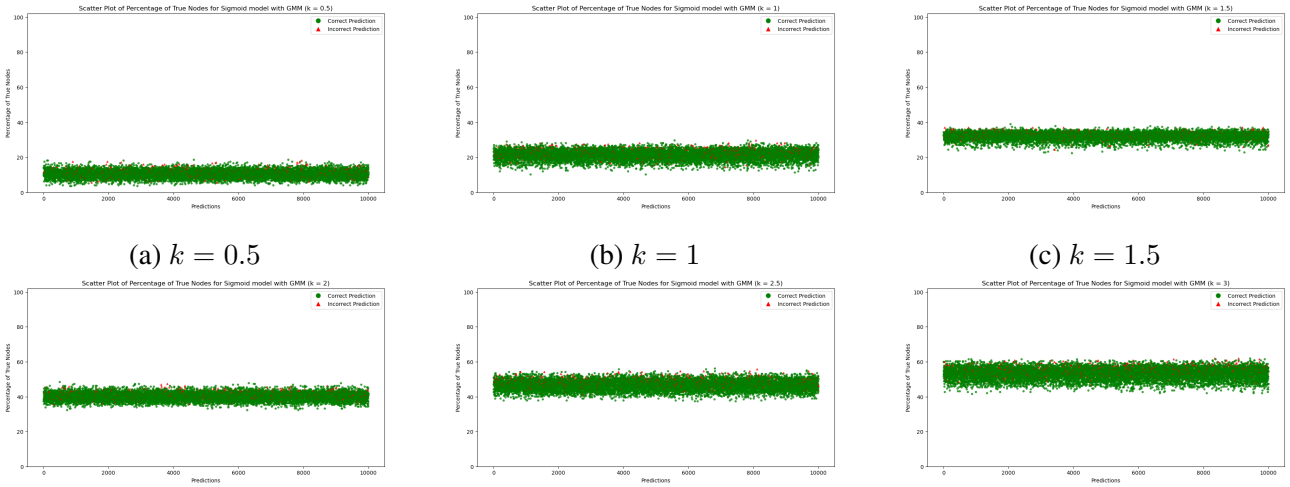
(a) Node 17 (b) Node 30 (c) Node 60

Figure 18: GMM with node activations' distribution

Now, we obtain d_i ranges for each node and perform our experiment. Results are displayed in figure 19 and table 10.

k	Point Biserial correlation	p-value
0.5	-0.1032	$4.033 \cdot 10^{-25}$
1	-0.1174	$4.639 \cdot 10^{-32}$
1.5	-0.102	$1.27 \cdot 10^{-24}$
2	-0.1305	$3.0133 \cdot 10^{-39}$
2.5	-0.16296	$1.782 \cdot 10^{-60}$
3	-0.1690	$5.421 \cdot 10^{-65}$

Table 10: Correlation and p-value for each k with GMM

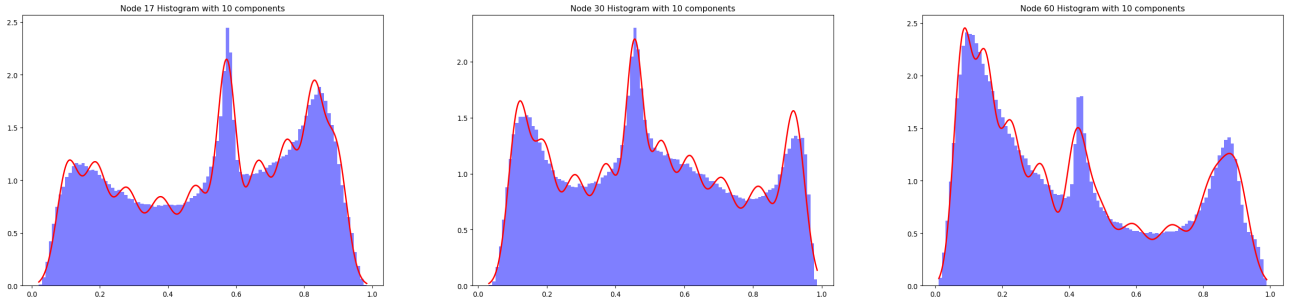


(a) $k = 0.5$ (b) $k = 1$ (c) $k = 1.5$

(d) $k = 2$ (e) $k = 2.5$ (f) $k = 3$

Figure 19: Percentage of nodes in range with GMM using different values of k

We have observed a slight negative correlation. The results are significant since the p-values are extremely small. To further this experiment, we assume that there would exist c clusters for each node, with c the number of output layers (i.e., $c = 10$). We perform the experiment again, but using c components for each GMM instead of finding the value that minimizes AIC and BIC.

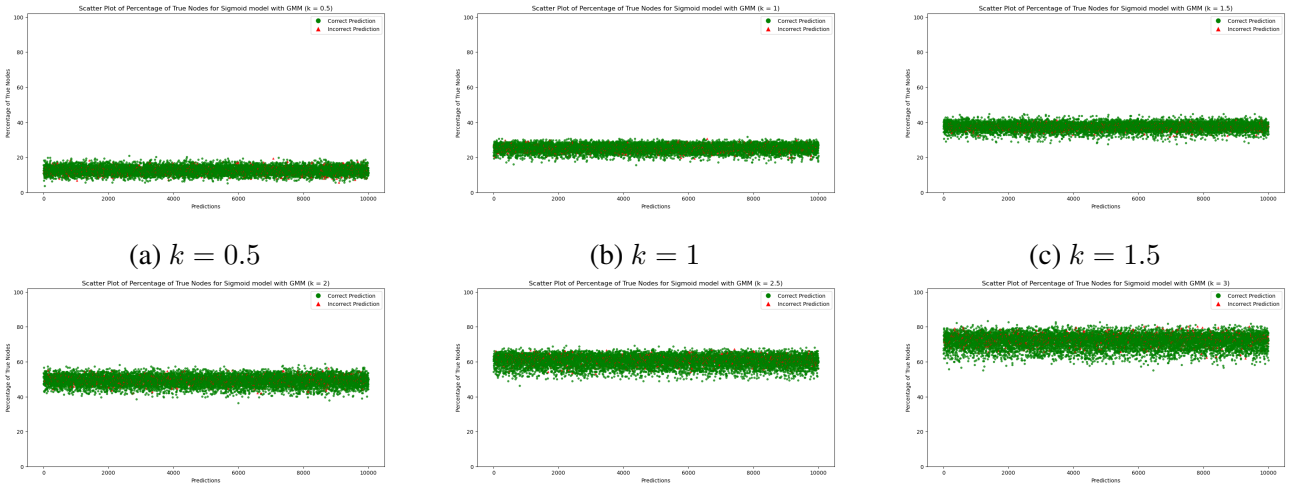


(a) Node 17 (b) Node 30 (c) Node 60

Figure 20: GMM (10 components) with node activations' distribution

k	Point Biserial correlation	p-value
0.5	0.00273	0.784
1	0.00335	0.7375
1.5	-0.00427	0.6693
2	-0.03765	0.00016
2.5	-0.07468	$7.6 \cdot 10^{-14}$
3	-0.0859	$7.5 \cdot 10^{-18}$

Table 11: Correlation and p-value for each k with GMM (10 components)



(a) $k = 0.5$ (b) $k = 1$ (c) $k = 1.5$

(d) $k = 2$ (e) $k = 2.5$ (f) $k = 3$

Figure 21: Percentage of nodes in range with GMM (10 components) using different values of k

The plots in figure 21 are similar to the ones in figure 19, where most of the data points are clustered together. However, we can see an increase of the value of these percentages, most likely due to the fact that now each node has 10 associated ranges. We can notice a correlation very close to 0 in table 11. For $k = 2.5$ and $k = 3$ where the p-values are significant, the correlation is close to 0. While the GMM approach did not give us insights on a possible correlation between accuracy and percentage of activations in range, it did give us the idea to investigate the behavior of nodes when confronted with a certain class.

5 Final Experiment

In this experiment, we use the model with 11 output nodes (the 11th being the "noise" class). Instead of recording the nodes activations during training regardless of the class of the training sample, we decide to use 11 lists for each node to record its activation depending on the class of the sample.

When training the network, we use the original training set and add 10% of the size of the set with uniform random samples (generated from 4.3). Then, depending on the class of the training sample, we record the nodes activations in the appropriate list. Figure 22 presents the distribution of node 2 in the first layer for two different classes.

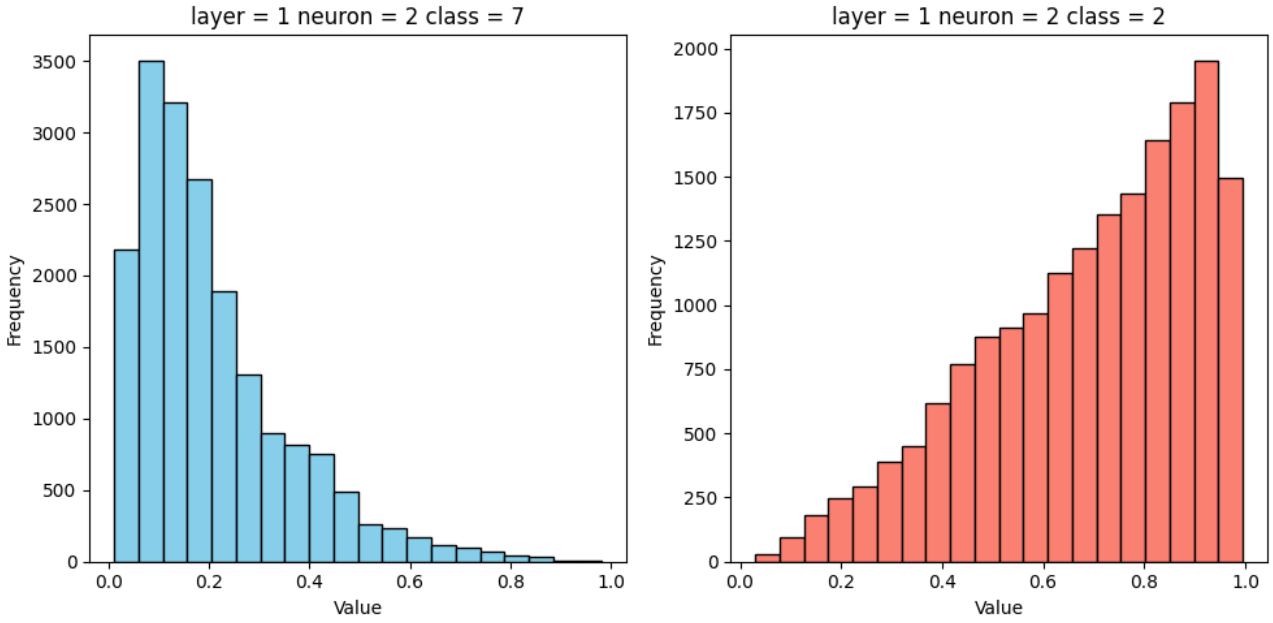


Figure 22: Node activations for node 2 in layer 1 for class 7 and 2

The two test sets are the same as in 4.3. For both sets, we compute the percentage of nodes falling in the range for each class and obtain a list of 11 percentages for each node for each sample. Then, we compute the difference g_s between the highest percentage and the one below it for a sample s. Using this value of g_s, we look at the accuracy of the predictions when the difference g_s is less than 0.1 (i.e., less than 10%) and when g is greater or equal to 0.1 (i.e., greater or equal to 10%). As usual, we calculated the correlation coefficient between g_s for each sample s and the accuracy and obtained the following results:

For both sets, we noticed that for 95% of the time, if a class has more than 90% of node activations in range, then the network predicts that the sample belongs to this specific class.

1. For the first set (composed of the MNIST test set and 1,000 samples with values between 0 and 1), we found that for the samples l that have g_l to be less than 10% (i.e when we have two or more classes that have roughly the same percentage), the accuracy of the network is around 73%. However, for the samples h that have g_h to be greater than 10% (i.e., one class has clearly more node values in range than the rest), the accuracy is around 99%. We also compute the point biserial correlation and find a value of -0.4417 with p-value close to 0. This means we have a positive correlation that is quite significant.

2. For the second test (composed of the MNIST test set and 1,000 samples with random values between -1 and 2), we obtain similar results than with the first test set. for the samples l that have g_l to be less than 10% (i.e when we have two or more classes that have roughly the same percentage), the accuracy of the network is around 75%. However, for the samples h that have g_h to be greater than 10% (i.e., one class has clearly more node values in range than the rest), the accuracy is around 99%. We also compute the point biserial correlation and find a value of 0.4569 with p-value close to 0. Again, this means we have a positive correlation that is quite significant.

These results indicate that the higher the difference between the percentages, the higher the accuracy. In other words, we have a lead to start characterizing the prediction power of the ANN using its node values.

6 Code Implementation

The code, as well as the plots and data, can be found in this GitHub repository.

7 Conclusion

7.1 Conclusion of the thesis

In this thesis, we intend to gauge the prediction power of ANNs by analyzing the behavior of node activations in response to various experiences and perturbations. Our investigation involved employing different techniques, including the examination of adversarial samples, Gaussian Mixture Models (GMMs), and additional classes paired with random noise. While the original conjecture appeared to be false at first, we have been able to prove a similar version of it through a series of experiments. First, we have observed that trimming the node values during training based on the distance from the mean of a node activation had a slight negative correlation with the accuracy. Furthermore, when combining this approach and examining probabilities of predictions instead of binary classification of correct/incorrect prediction, we have seen that the percentage of node activations in the range has a negative correlation with the probability of predicting a wrong class. Finally, our most notable result is finding that when separating the training activation by classes, we obtain a positive correlation between the difference of percentages of the highest and second highest g_s with accuracy. Therefore, one could assume that by looking at the difference g_s for a sample, if that difference is greater than 10%, then the prediction is most likely correct.

7.2 Future work

Despite the insights gained from this research, several ideas for future research remain open. One promising direction is to explore the threshold that g_s is subjected to in order to maximize the accuracy when g_s is greater than that threshold and minimize the accuracy otherwise. Additionally, investigating the generalizability of our findings across different network architectures, datasets, and tasks could provide valuable insights into the broader applicability of our conclusions. Finally, as stated before, these results are purely experimental. Thus, further theoretical results are required in order to advocate for the validity of the demonstrated conjectures. Overall, by continuing to investigate the relationship between node activations and network performance, we can advance our understanding of ANNs and contribute to the development of more interpretable and reliable machine-learning systems.

8 Acknowledgments

I would like to expres my gratitude toward Professor Leo Liberti for making this thesis possible and for his great supervision throughout my internship. His insights and mentoring have been crucial in shaping the trajectory of my research. I would also like to thank Matei Atodiresei for his collaboration and support throughout our work together. Lastly, I would like to express my deep appreciation to Jeremie Dentan for his invaluable contributions and key ideas that greatly influenced my work.

9 References

[1] Douglas G Altman. *Practical statistics for medical research*. Chapman and Hall/CRC, 1990.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Wouter Gevaert, Georgi Tsenov, and Valeri Mladenov. Neural networks used for speech recognition. *Journal of Automatic Control*, 20, 01 2010.

[4] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[6] Diana Kornbrot. *Point Biserial Correlation*. 10 2005.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[8] Quoc V. Le, Will Zou, Serena Yeung, and Andrew Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Conference on Computer Vision and Pattern Recognition*, 2011.

[9] Y. LECUN. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[10] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[11] Mavuto Mukaka. Statistics corner: A guide to appropriate use of correlation coefficient in medical research. *Malawi medical journal : the journal of Medical Association of Malawi*, 24 3:69–71, 2012.

[12] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[13] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*, 42(1):259–268, 2015.

[14] Thomas Douglas Victor Swinscow, Michael J Campbell, et al. *Statistics at square one*. Number Ed. 10. Bmj London, 2002.

[15] Khoa Tran, Olga Kondrashova, Andrew Bradley, Elizabeth Williams, John Pearson, and Nicola Waddell. Deep learning in cancer diagnosis, prognosis and treatment selection. *Genome Medicine*, 13, 09 2021.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[17] Jiarong Xu, Junru Chen, Siqi You, Zhiqing Xiao, Yang Yang, and Jiangang Lu. Robustness of deep learning models on graphs: A survey. *AI Open*, 2:69–78, 2021.