UCL
Université catholique de Louvain

École polytechnique de Louvain (EPL)

Investigating Layer Activation Patterns in Neural Networks for Classification Error Detection

Dissertation presented by
Victor DEMUYSERE

for obtaining the Master's degree in
Computer Science and Engineering
Option(s): Artificial Intelligence

Supervisor(s)
Christophe DE VLEESCHOUWER

Reader(s)
Simon CARBONNELLE, Benoît MACQ

Academic year 2017-2018

Abstract

Neural networks are intricate systems that learn to classify and recognize complex patterns. They yield incredible results in many fields like computer vision, speech recognition, text categorization and many others. It is therefore tempting to use them in difficult and important applications such as road tracking for self-driving cars, medical diagnosis, stock market predictions, etc. But our understanding of the intrinsic mechanisms that govern their behavior is still not fully understood. Neural networks are still mostly black-boxes, and blindly trusting their decision on important matters could have disastrous consequences. Therefore, a lot of research is made to find ways to make neural networks more reliable.

In this thesis, we look at the particular problem of detecting when the decision of a network is likely to be incorrect. Most of today's post training methods focus on the last layer of the network. We propose two different approaches that instead look at the entirety of the network. In the first one we search for individual neurons that could indicate a correct or incorrect decision. The second approach looks at patterns in the activation of neurons. It measures the similarity between samples regarding these activation patterns and tries to detect erroneous samples by looking at how similar they are to supposedly close training samples.

We then measure the performances of our methods and compare them between each other and to the state-of-the-art.

Acknowledgements

I would like to greatly thank Prof. Christophe De Vleeschouwer for his great advises and being as available as possible throughout the year.

I would also like to give many thanks to PhD student Simon Carbonnelle for genuinely offering his help. Without mentioning all the really interesting and relevant papers that he sent me.

A great thank to Mr. Benoît Deper that made himself available to support a potential thesis even if things turned out differently.

Finally, I would like to thank my parents and my sisters for all the relevant advises they gave me not only for this thesis but during my entire studies in general, as well as for their endless support.

Contents

Chapter 1

Introduction

1.1 Context and objectives

Artificial neural networks are an intricate system that learn to classify and recognize complex patterns. Image recognition, speech recognition, object detection, text categorization and many others are fields in which neural networks yield astounding results.

The main problem with neural networks is that the mechanisms that allows them to learn such complex concepts are still not well understood. Making them black-boxes. They deliver an output without giving any clues on what brought them to it. Many studies have been conducted to enlighten the intrinsic behaviors of neural networks.

In this thesis, we will focus on categorical neural networks. Networks whose task is to take an entry vector (that can be multidimensional like the pixel values of an image) and classify it in one of different predefined classes. More specifically, we are interested in the degree to which we can trust the decision of a network.

The decision of the network when classifying an entry sample can be compromised on different levels :

- The network was not properly trained (wrong architecture, not enough training, too much training leading to overfitting, ...) and does not classify the sample correctly because of that. Even if that sample is similar to training samples.
- It can also be due to the fact that the network is not perfect and some samples still give it a hard time.
- Finally, it can be because the sample is from a different distribution[1]. This means that the sample does not resemble any of the training samples. We call these samples out-of-distribution. One problem with out-of-distribution samples can be that they simply do not belong to any of the predefined classes. In that case, they will always be misclassified. But, even if they do belong to one of the classes, they can still cause problems to the classification. The reason is that their values may differ from the training samples (called in-distribution samples). They can have unusual contrasts, higher or lower average values, new kind of patterns, ... Many factors that can baffle the network.

This misclassification is to be avoided as much as possible as it can have disastrous impacts in some domains (road-detection in self-driving cars, medical diagnosis, ...). Therefore, multiple researches have been conducted in order to detect if a given sample was classified correctly or not (some of them are described in Section 1.2).

[1]Most of the time, all the samples in a dataset are considered to be from the same distribution.

In this thesis, we explore two different approaches that should allow us to distinguish out-of-distribution and misclassified samples from correctly-classified in-distribution samples.

Our goal is to verify if the intuitions behind these approaches hold against experimentation. We also want to measure how well they allow detecting out-of-distribution and misclassified samples.

We will perform our experiments on a single network architecture and on the MNIST dataset. The reason we keep things simple is to maintain the number of parameters down to the essentials and to limit the amount of external factors that would have to be taken into account. This allows for straight-forward observations and clearer conclusions.

As described in more details later in this manuscript, in order to have examples of out-of-distribution samples, we will train some networks on a subset of MNIST leaving out some classes. The samples belonging to the left out class will serve as out-of-distribution samples. However, these samples are not going to be entirely out-of-distribution since they come from the same initial dataset. Therefore, we will use randomly generated white noise samples in addition to our out-of-distribution samples.

1.2 Related works

A lot of research has been done in the domain of neural networks, trying to enlighten some of the mechanics that govern their behaviors or to improve their efficiency and reliability. Here are some recent and/or interesting works done on the matter.

Calibration: (Guo et al. 2017) show that deep networks are badly calibrated. They define the calibration of a network as how close its confidence[2], when classifying a sample, is to the actual probability of that sample to be correctly classified. Figure 1.1 shows two diagrams that highlight the miscalibration of deep neural networks. These diagrams were constructed by dividing the samples into different categories according to the softmax output of the prevalent class (i.e. their confidence), creating one column per confidence category. The height of each column then corresponds to the actual accuracy of each category. The closer the diagrams are to the identity function, the better the network is calibrated. A perfect diagonal would indicate that the softmax output closely matches the actual accuracy.

They try out different approaches to remedy this miscalibration. The most notable and elegant one being ***temperature scaling***. The idea is to divide the outputs of the neurons of the last layer by a scalar T (called "temperature") before passing them through the softmax function. T is optimized according to negative log likelihood on the validation set. Despite its simplicity, temperature scaling is very efficient on most computer vision problems.

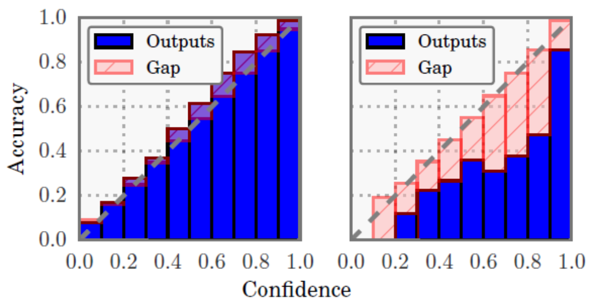


Figure 1.1: Reliability diagrams for a 5-layer LeNet (left) and for a 110-layer ResNet (right) on CIFAR-100 created by (Guo et al. 2017).

[2]We define the notion of confidence in the "Network output" paragraph of Section 2

Misleading image classifying networks: (Szegedy et al. 2013) show that easily classifiable images can be modified in a way that is imperceptible for humans, but that baffles a trained network. Making it confidently classify the images in the wrong categories. Figure 1.2 shows a few examples.

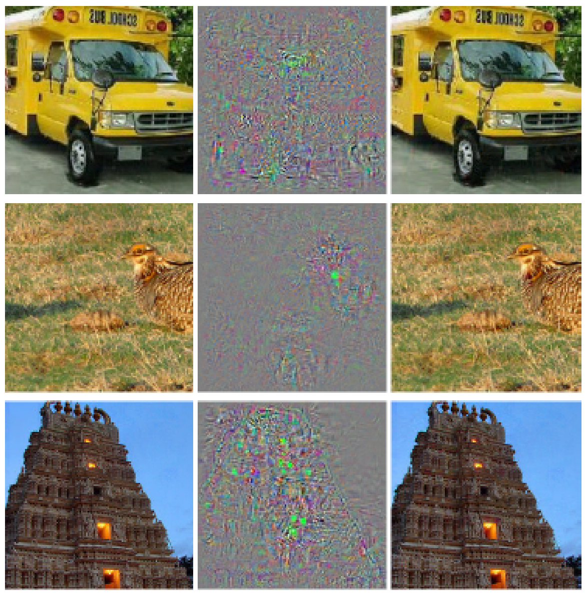


Figure 1.2: Correctly classified images (left), the applied filter magnified 10 times (center) and the modified image (right). All images in the right column are predicted to be an *"ostrich, Struthio camelus"*. Figures from (Szegedy et al. 2013).

Fooling image classifying neural networks: (Yosinski et al. 2014) used adversarial networks to create images that do not resemble anything to humans, but that were classified by deep neural networks with a strong confidence. Figure 1.3 shows some examples.

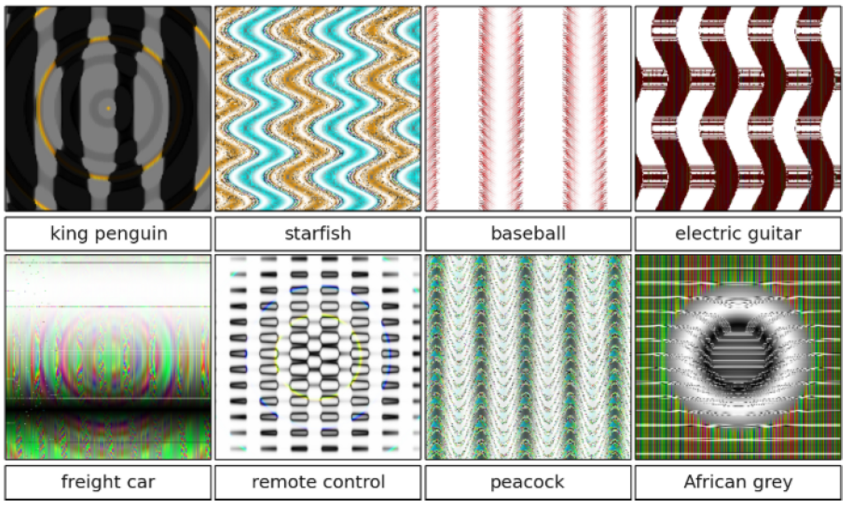


Figure 1.3: Evolved images that state-of-the-art networks trained on ImageNet believe with $\geq 99,6\%$ confidence to be familiar objects. Figure from (Yosinski et al. 2014).

Baseline: (Hendrycks et al. 2016) created a baseline to detecting out-of-distribution and misclassified samples. Their approach consists of applying a threshold on the confidence of the network. Samples for which the confidence is below the threshold are considered wrongly classified. They tested their baseline method in several fields: computer vision, natural language processing, automatic speech recognition, ...

ODIN: (Liang et al. 2017) created a new method called ODIN to detect out-of-distribution samples. It works similarly to the baseline method with a few twists. The first twist is that it applies temperature scaling to the network. The second one is that they slightly modify the input based on the back-propagation of the outputs of the softmax function. The filter applied on the input augments all the softmax outputs. Since it has a stronger impact on in-distribution samples, it makes it easier to distinguish them from out-of-distribution ones with a threshold.

The lottery ticket hypothesis: (Frankle et al. 2018) look into the winning lottery ticket hypothesis. The hypothesis states that a large neural network is able to converge more easily to good accuracies because a subset of its neurons (the winning ticket) were instantiated in such a way that the network could converge to good values. They show that it is possible to find this "sub-network" by pruning the main network, and that the sub-network, if instantiated with the same initial values, can converge in the same amount of training as the total network, and reach the same (sometimes even better) accuracy.

Generalization: (Zhang et al. 2016) show that the common thought that bad generalization is a property of certain types of training or architectures is wrong. They do so by overfitting networks with randomly labeled datasets (i.e. data for which it is impossible to generalize). They show that the only requirement in order to overfit data (i.e. creating a generalization gap) is to have a number of parameters that exceeds the number of data points.

1.3 Proposed new approaches

As described in the previous section as well as in the "Network output" paragraph of Chapter 2, most of today's techniques to detect out-of-distribution and misclassified samples are based on an analysis of the last layer's activation.

In this thesis, we suggest taking a look at a more of the information we can gather about a network instead of focusing on the last layer. We try out two new approaches that look at the activation of neurons to get an insight on the factors that make a network correct or not.

Indicative Neurons We first take a look at neurons individually. We observe their training and influence on the network.

The idea behind this approach is that some neurons indicate a correct or incorrect decision of the network when they are activated. We try to find if such indicative neurons exist. If they do, we try to find which characteristics allow them to be distinguished from the others.

Activation Patterns Secondly, we compare the neurons' activation between samples. Intuitively, similar samples (i.e. samples of the same class or subclass) should activate the neurons of the network in a similar fashion. We call the way a sample activates the neurons of a network: its ***activation pattern***. We then define a measure distance between activation patterns. By storing the activation patterns for each training sample we can then see, for a brand-new sample, if its activation pattern is close to some stored ones.

From this intuition, we try out two ways of detecting erroneous samples :

1. We look at the training samples that have activation patterns that are close to the one of the observed sample. We then look at *how far apart* they are. The intuition is that out-of-distribution and misclassified samples should have activation patterns that are further apart from the closest training samples than regular samples.

2. The second idea is based on the layers of the network. We can look at the individual activation pattern of each layer. The activation pattern of the samples in the first layer are all very

different. The intuition is that the activation patterns of samples of the same class get closer together, layer by layer, until they activate the final layer the same way. If this is true, the activation patterns of out-of-distribution and misclassified samples should make a different progression through the network than regular samples.

We try to exploit this to detect erroneous samples. We compare the progress of a given sample's activation pattern to the progress of the activation patterns of its closest training samples in the last hidden layer. This comparison is used to predict the reliability of the network's decision.

1.4 Outline

In Chapter 2, we start by describing the functioning of categorical fully-connected neural networks as well as back-propagation. This will give us the opportunity to indicate the notations that we are going to use throughout this manuscript. We also describe the main neural networks we trained and how we trained them.

Chapter 3 explores our first approach (about indicative neurons). We start by stating the hypothesis we want to highlight as well as the mechanisms in neural networks that led us to think the approach is relevant (Section 3.1). We also describe some concepts that we use later in the chapter.

We continue by describing our process. We proceed in three phases:

1. In the first phase we discover the correlation between characteristics of the neurons and their impact on the decision of the network.
2. Then, we look at how neurons behave when fed with out-of-distribution and misclassified samples in comparison with correctly-classified in-distribution samples.
3. Finally, we see if we are able to make detectors of out-of-distribution and misclassified samples based on our theory and observations. We then measure the performances of these detectors.

Each phase is separated into two:

1. A first part describing our experiments and their results,
2. And a second one analyzing our observations.

In Chapter 4 we explore our second approach regarding the grouping of activation patterns. Again, we start by presenting our hypothesis as well as what led us to believe this approach was promising. We also describe the concepts that we use in the rest of the chapter.

Here we have two different ideas:

1. One solely concerning the distance between a sample's activation pattern and the activation patterns of close training samples,
2. And the other one about the *progress* of the sample's activation pattern as compared to the close training samples.

For each idea we proceed in two phases:

1. In the first phase we look at the behavior of the activation patterns on different datasets to check if the observations coincide with our hypothesis.
2. We then create out-of-distribution and misclassified sample detectors based on our observations and hypothesis, and record their performances.

Again, each phase is separated into an “experiments and results” section and an “analysis” section.

Finally, in Chapter 5, we summarize what we have learned and observed in each chapter and compare our detectors together. We also compare their performances against the baseline method (Hendrycks et al. 2016) and against the state-of-the-art ODIN method (Liang et al. 2017).

We then describe leads to further researches.

We finish with a general conclusion.

Note. *Throughout this manuscript, whenever we define a concept that we use again later, we put its name in* ***bold-italic****. All these highlighted concepts are listed in the Index along with a reference to the page at which they were first defined.*

Chapter 2

Definition of fully connected neural networks and notations

In this paper, we are going to be analyzing fully connected classification neural networks. It thus seems natural to briefly present them. It will at the same time give us the opportunity to present the notations we will be using throughout this thesis.

Neurons As its name suggests, the building blocks of a neural network are neurons. A neuron is a function characterized by an array of weights $\vec{w}$, a bias b and an activation function $g()$. Given an input vector $\vec{x}$ with the same length as $\vec{w}$, the output a (also called the activation) of the neuron is given by:

$$a = g(\vec{w} * \vec{x} + b) \tag{2.1}$$

For easier notations, we consider the bias as the first value of the weight vectors (making the weight vector one component bigger), i.e. $w_0 = b$. We thus implicitly add the value 1 at the beginning of every input vector. Equation 2.2 thus becomes:

$$a = g(\vec{w} * \vec{x}) \tag{2.2}$$

For the rest of this thesis we refer to $\vec{w} * \vec{x}$ as z.

Note. *A neuron is considered activated when its output value is higher than zero. For most activation functions this is the case whenever z is above zero.*

In most of today's networks, the chosen activation function $g()$ is the ReLU function:

$$relu(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ z & \text{otherwise} \end{cases} \tag{2.3}$$

We use this function for all the neurons of the networks built for this thesis. It is interesting to note that

$$\frac{\partial relu(z)}{\partial z} = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.4}$$

General architecture A neural network is a computer system that takes a number of values as input (called features, pixels for image inputs for example) and that produces an output vector. It is composed of several layers of neurons.

Layers Each layer l is made out of r^l neurons. They all have an input and an output vector. We will call the output vector of layer l : $\vec{o}^l$. The input vector of the first layer of a network is simply the features vector. For all the other layers, the input corresponds to the output vector of the previous layer.

The input vector is fed to all the neurons composing the layer. Each component of the output vector is simply the output of one of its neurons. Therefore, each component n of $\vec{o}^l$ is given by:

$$o_n^l = a_n^l = \begin{cases} relu(\vec{o}^{l-1} * \vec{w}_n^l) & \text{if } l > 1 \\ relu(\vec{x} * \vec{w}_n^l) & \text{if } l = 1 \end{cases} \tag{2.5}$$

Where o_n^l is the n^{th} component of the output of layer l, a_n^l is the output of neuron n of layer l, $\vec{x}$ is the input sample and $\vec{w}_n^l$ is the weight vector of the n^{th} neuron of layer l.

Network output Here, we will use neural networks as classifiers. Most of the neural networks that need to classify samples in c classes, with $c > 2$ use one scalar output per class. They each correspond to the outputs of the c neurons of the last layer. The highest output indicates the chosen class.

More often than not we do not just want to know which class prevails, but also how much it prevails compared to the other classes. If we just take the output of the neurons of the final network as such, the results can be hard to interpret. Therefore, we apply a ***softmax*** function on these outputs. Here is its formula:

$$out_n = \frac{e^{o_n^L}}{\sum_{m=1}^{c} e^{o_m^L}} \tag{2.6}$$

Where out_n is the final output for the n^{th} class. o_n^L is the output of the n^{th} neuron of the last layer (L is the number of layers in the network).

The outputs for each class now resemble probabilities since they add up to one. This is why the output of the softmax function for each class are often seen as the probability that the analyzed sample belongs to that class. The output of the prevalent class is then considered as the ***confidence*** of the network regarding the classification of a sample. We will be using confidence in this context throughout the thesis.

Learning Classification neural networks are part of the supervised spectrum of machine learning[1]. We train them on pre-labeled datasets. At the end of the training, the network should be able predict the labels of never-seen-before samples (this ability is called generalization).

The training is done by repeatedly updating the parameters of the network. The parameters are the weights and biases of each neuron. They are instantiated with arbitrary values.

The most widespread technique to update the parameters of a network is back-propagation. It is based on stochastic gradient descent. An error function $E(X, \theta^t)$ is defined. X are all the pairs $\{\vec{x}^i, \vec{y}^i\}$ in one batch of the training data, with $\vec{y}^i$ the expected output for sample $\vec{x}^i$. θ^t represents the parameters of the network at iteration t. The goal of the stochastic gradient descent is to find the parameters θ minimize the error $E(X, \theta)$.

At each iteration we modify the weights proportionally to their impact on the error:

$${w_{nm}^l}^{t+1} = {w_{nm}^l}^t - \alpha \frac{\partial E(X, \theta^t)}{\partial w_{nm}^l} \tag{2.7}$$

Where ${w_{nm}^l}^t$ is the m^{th} weight of the n^{th} neuron of layer l at iteration t. We can visualize it as the link between the node m of layer $l-1$ and the node n of layer l. α is the learning rate.

[1]Some researches are being conducted on unsupervised learning with neural networks, but the main use cases are based on supervised learning and these are the ones we are interested about in this manuscript.

The error function (also called cost function or loss function) that is used by most networks is the categorical cross-entropy:

$$E(X,\theta^t) = -\frac{1}{N}\sum_{\vec{y}\in X}\sum_{k=1}^{c} y_k \ln(\hat{y}_k) \tag{2.8}$$

Where N is the number of samples in X, y_k is the expected value of the output corresponding to class k when the network is fed with a sample $\vec{x}$ and $\hat{y}_k$ is the actual value of that output.

We will now expand equation 2.7. But we will make the assumption that there is no softmax function applied on the last layer in order to greatly simplify the math. This won't have an impact on the deductions we make later.

We notice that $E(X,\theta^t)$ can be expressed as the average of the error E^i of each sample $\vec{x}^i$ in X. If

$$E^i = -\sum_{k=1}^{c} y_k^i \ln(\hat{y}_k^i) \tag{2.9}$$

then

$$E(X,\theta^t) = \frac{1}{N}\sum_{i=1}^{N} E^i \tag{2.10}$$

Equation 2.7 then becomes:

$${w_{nm}^l}^{t+1} = {w_{nm}^l}^{t} - \alpha\frac{1}{N}\sum_{i=1}^{N}\frac{\partial E^i}{\partial w_{nm}^l} \tag{2.11}$$

We have

$$\frac{\partial E^i}{\partial w_{nm}^l} = \frac{\partial E^i}{\partial a_n^l}\frac{\partial a_n^l}{\partial w_{nm}^l} \tag{2.12}$$

Where a_n^l is still the output of neuron n of layer l. From equations 2.5 and 2.4 we get:

$$\frac{\partial a_n^l}{\partial w_{nm}^l} = relu'(\sum_{j=0}^{r^{l-1}} o_j^{l-1} w_{nj}^l) o_m^{l-1} = relu'(z_n^l) o_m^{l-1} = \begin{cases} o_m^{l-1} & \text{if the neuron is activated } (z_n^l > 0) \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

for $l > 1$ and

$$\frac{\partial a_n^l}{\partial w_{nm}^l} = \begin{cases} x_m & \text{if the neuron is activated } (z_n^l > 0) \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

for $l = 1$, with x_m the m^{th} component of entry sample $\vec{x}$. z_n^l is the scalar product of the input ($\vec{o}^l$ or $\vec{x}$) with the weight vector $\vec{w}_n^l$ of the n^{th} neuron of layer l (with the included bias).

To express $\frac{\partial E^i}{\partial a_n^l}$ in a readable way, we define a variable δ as such:

$$\delta_n^l \equiv \frac{\partial E^i}{\partial a_n^l} \tag{2.15}$$

Using the equation 2.9 of cross entropy and knowing that $a_n^L = \hat{y}_n$ we get

$$\delta_n^L = -\frac{y_n}{\hat{y}_n} \tag{2.16}$$

for the last layer.

For the hidden layers we have:

$$\delta_n^l = \frac{\partial E^i}{\partial a_n^l} = \sum_{j=1}^{r^{l+1}} \frac{\partial E^i}{\partial a_j^{l+1}}\frac{\partial a_j^{l+1}}{\partial a_n^l} = \sum_{j=1}^{r^{l+1}} \delta_j^{l+1}\frac{\partial a_j^{l+1}}{\partial a_n^l} \tag{2.17}$$

Since

$$a_j^{l+1} = relu(\sum_{k=0}^{r^{l+1}} w_{jk}^{l+1} a_k^l) \tag{2.18}$$

we get

$$\frac{\partial a_j^{l+1}}{\partial a_n^l} = relu'(z_j^{l+1}) w_{jn}^{l+1} \tag{2.19}$$

resulting in the final equations

$$\delta_n^l = \sum_{j=1}^{r^{l+1}} \delta_j^{l+1} relu'(z_j^{l+1}) w_{jn}^{l+1} \tag{2.20}$$

$$\frac{\partial E^i}{\partial w_{nm}^l} = relu'(z_n^l)\delta_n^l o_m^{l-1} \tag{2.21}$$

$$= relu'(z_n^l) o_m^{l-1} \sum_{j=1}^{r^{l+1}} \delta_j^{l+1} relu'(z_j^{l+1}) w_{jn}^{l+1} \qquad \text{(for } l < L\text{)} \tag{2.22}$$

$$= -relu'(z_n^l) o_m^{l-1} \frac{y_n}{\hat{y}_n} \qquad \text{(for } l = L\text{)} \tag{2.23}$$

Notice that for each neuron n of every layer l, the gradients $\frac{\partial E^i}{\partial w_{nm}^l}$ applied to its weights are null if the neuron is not activated.

Our networks The networks on which we perform our experiments for this thesis are quite simple. They are fully connected with the architecture depicted in Figure 2.1. We trained them on the ***MNIST*** database[2] for a single epoch (going through the training data once) with a batch size of 100 and a learning rate of 0.001. We did not train the network too much on purpose. We wanted the results to be imperfect to increase the amount of incorrect examples. But we still wanted a reasonable accuracy in order for our experiments to be representative for functional networks. We initialized the weights with random values (with a uniform distribution) and the biases to one.

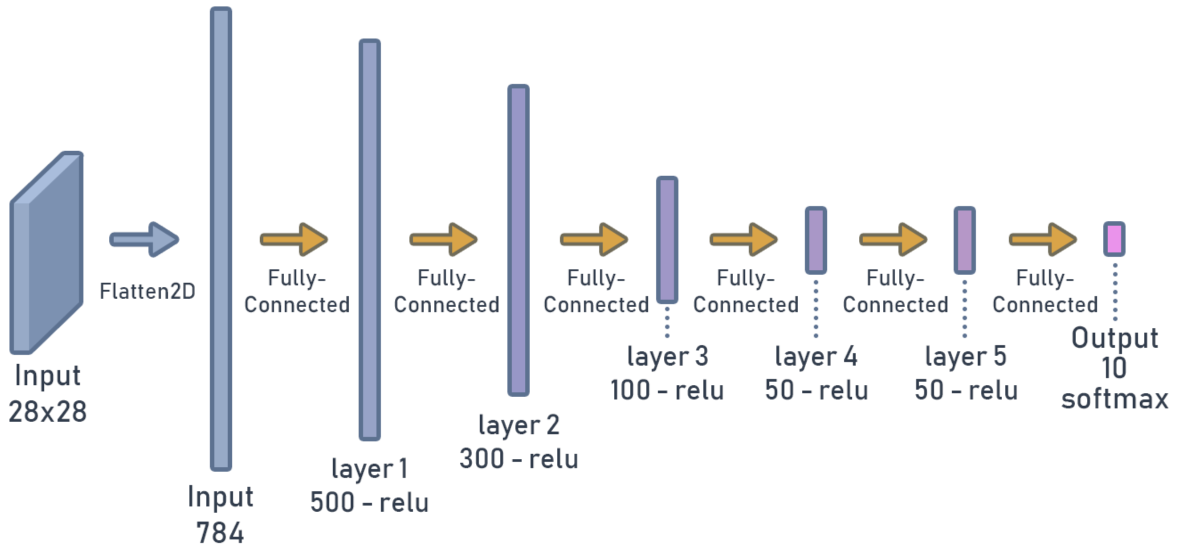


Figure 2.1: Architecture of the tested networks.

[2]MNIST is a database of 28x28 black and white images depicting handwritten digits from 0 to 9. Each pixel is a value representing its lightness on a scale from 0 (totally black) to 255 (totally white). The database consists of 70000 different image/label pairs. 60000 samples are used to train the network and 10000 are used to test the network's accuracy on never-seen-before examples.

Frequently used techniques Lots of network architectures make use of techniques that allows them to reach an even better accuracy. The most popular ones are:

Dropout: Introduced by (Srivastava et al. 2014), this technique consists of randomly dropping (deactivating) neurons in the network during training. This prevents the neurons from co-adapting too much and slows down overfitting, allowing the network to generalize better.

Batch normalization: During the training of a neural network it is often difficult for a layer to learn correct parameters when the distribution of the outputs of previous layer is constantly changing. This phenomenon called internal covariate shift is addressed by (Ioffe et al. 2015). They solved this problem by normalizing the inputs of each layer. Networks that use batch normalization converge much faster and reach higher accuracies than their non-batch-normalized counterparts.

We do not however use any technique other than softmax on our networks. We do so in order to restrain the amount of external factors that we have to take into account in our observations to the essential.

Chapter 3

Analyzing the behavior of individual neurons

3.1 Concept, hypothesis and terminology

In this chapter, we look at the training of each neuron separately. We try to find behaviors that could indicate that the weights and bias of a neuron have converged to good values. Such neurons would be considered "strong" and their activation should be an indicator that the network is making a correct decision.

The theory that we would like to test is whether it is possible, through this analysis of the training, to find neurons whose activation indicates whether the network is making a correct decision or not.

When we look at the equations of back propagation in Chapter 2 we notice that certain characteristics of neurons could indicate a good convergence:

1. **Number of activations** We see in equations 2.21 and 2.11 that the more a neuron is activated during the training, the more its weights were updated. Giving them more opportunities to converge to meaningful values. During our experiments, we are going to count the number of times a neuron was activated during the training of our networks. We will call this count the ***activation count***. Summing the activations of all the neurons for every training sample was quite time-consuming. So instead, we counted the activations for one sample in every batch (the first one). Since we were using batches of 100, the maximal activation count for the neurons of the network trained on MNIST is 600.

2. **Convergence** We consider that a neuron has converged when the gradients applied to its weights are small. Let us take a look at what a small gradient indicates.

   When the m^{th} weight of a neuron n of a layer l is updated, the gradients $\frac{\partial E^i}{\partial w^l_{nm}}$ applied to the weight are given by equation 2.22 in Chapter 2. We rewrote the equation (slightly differently) here:

$$\frac{\partial E^i}{\partial w^l_{nm}} = relu'(z^l_n) o^{l-1}_m \delta^l_n \tag{3.1}$$

$$= \begin{cases} relu'(z^l_n) o^{l-1}_m \sum_{j=1}^{r^{l+1}} \delta^{l+1}_j relu'(z^{l+1}_j) w^{l+1}_{jn} & \text{(for } l < L\text{)} \\ -relu'(z^l_n) o^{l-1}_m \frac{y_n}{\hat{y}_n} & \text{(for } l = L\text{)} \end{cases} \tag{3.2}$$

   Where E^i is the cross-entropic loss for one input sample $\vec{x}^i$.

   We can see that the gradients of a weight w^l_{mn} are proportional to $\delta^l_n = \frac{\partial E^i}{\partial a^l_n}$, the derivative of E^i with respect to the activation of neuron n of layer l.

Since

$$\delta_n^l = \sum_{j=1}^{r^{l+1}} \delta_j^{l+1} relu'(z_j^{l+1}) w_{jn}^{l+1} \qquad \text{(for l<L)} \qquad (3.3)$$

$$\delta_n^l = -\frac{y_n}{\hat{y}_n} \qquad \text{(for l=L)} \qquad (3.4)$$

We observe that the magnitude of δ_n^l depends on two things:

- The derivatives of E^i with respect to the outputs $\hat{y}_n^i$ of the network:

$$\frac{\partial E^i}{\partial \hat{y}_n^i} = -\frac{y_n}{\hat{y}_n} \qquad (3.5)$$

  Note. *It is interesting to observe for cross-entropic loss that -$\frac{\partial E^i}{\partial \hat{y}_n^i}$ is proportional to E^i.*

- And what we here call the influence I_n^l of a neuron n of layer l on the output $\hat{y}$ of the network:

$$I_n^l = \begin{cases} \sum_{j=1}^{r^{l+1}} I_j^{l+1} relu'(z_j^{l+1}) w_{jn}^{l+1} & \text{for } l < L \\ 1 & \text{for } l = L \end{cases} \qquad (3.6)$$

All together, this means that if the gradient applied to a weight w_{mn}^l is small (but not null), we can deduce one of three things:

- Either the error was small (because -$\frac{\partial E^i}{\partial \hat{y}^i}$ was small for all i),
- Or the impact I_n^l of neuron n of layer l on the output was small,
- Or both.

Knowing that the activation of a specific neuron is associated with a small classification error is a good sign that this neuron has converged to meaningful values. The previous observations tell us that we can find neurons that lead to small classification errors by looking at the magnitude of the gradients applied to their weights. We just need to make sure that these neurons have a big impact on the output.

Here, we will use an approximation of the impact of a neuron that doesn't require looking at the activations of all the neurons for all the samples. We try out two different approximations in fact: the l1 influence I_{1n}^l of the n^{th} neuron of layer l:

$$I_{1n}^l = \begin{cases} 0 & \text{if the neuron does not activate for any training sample} \\ 1 & \text{if } l = L \\ \sum_{k=1}^{r^{l+1}} \mid w_{kn}^{l+1} \mid I_k^{l+1} & \text{otherwise} \end{cases} \qquad (3.7)$$

and the l2 influence:

$$I_{2n}^l = \begin{cases} 0 & \text{if the neuron does not activate for any training sample} \\ 1 & \text{if } l = L \\ \sqrt{\sum_{k=1}^{r^{l+1}} (w_{kn}^{l+1} I_k^{l+1})^2} & \text{otherwise} \end{cases} \qquad (3.8)$$

We noticed later on that the l2 influence brings out more information regarding the neurons. From now on, this will be the one we are referring to when we talk about the ***influence***.

To make sure a neuron has converged correctly, we must thus check if the gradient applied to its weights stabilized around small values and see if the neuron still has a big influence.

To measure the decrease of the gradients applied to the weights of a neuron, we used a new metric that we called the ***average gradient***. We compute the average gradient ag_n^l of a neuron n in layer l like so:

$$ag_n^l = \sum_{t=T-xT}^{T} \sum_{m=0}^{r^{l-1}} \frac{\partial E(X, \theta^t)}{\partial w_{nm}^l} \tag{3.9}$$

Where T is the total number of training iterations. r^{l-1} is the size of the previous layer (i.e. the number of weights of the neuron). x is an arbitrarily chosen fraction. It lets us chose on which percentage of the last iterations we want to compute our average gradient. We tried different values for x and noticed that $x = 0.5$ showed the best results.

In the next sections we are going to experiment with these newly defined measures to find out if they indeed allow us to look at neurons in a way that brings us information on the correctness of the decision of a network.

Concepts and definitions Let us first define a few other concepts that we are going to be using in this chapter:

- **Confidence** An important concept that we already introduced in the "Network output" paragraph of Chapter 2 is the confidence of a network. What we call the confidence is the value of the dominant component of the softmax output when classifying a sample. We often assimilate the confidence to the probability of the network's decision to be correct. But as (Guo et al. 2017) pointed out, it is not always the case. Therefore, they defined the ***calibration*** of networks. A network is considered well calibrated if the confidence output when classifying any sample reflects the probability that this sample is correctly classified. In order to measure how well a network is calibrated, they used the ***Expected Calibration Error*** (***ECE***) . They divide the classified samples in different categories based on the confidence with which they were classified. Each category corresponds to an interval of confidence. We call B_m the set of all samples in category m. They then compute for each category the average confidence $conf(B_m)$ and the average accuracy with which the samples where classified $acc(B_m)$.

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(arg\,\underset{n}{max}(\hat{y}_n^i) = arg\,\underset{n}{max}(y_n^i)) \tag{3.10}$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}^i \tag{3.11}$$

  With $\hat{p}^i$ corresponding to the value of the highest component of the softmax output when classifying a sample $\vec{x}^i$. The notation $1(a = b)$ is evaluated to a 1 if the equality is true and to 0 otherwise.

  The ECE is then given by:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} |acc(B_m) - conf(B_m)| \tag{3.12}$$

  With M being the number of categories and N the total number of classified samples.

  We are going to use the concept of categories based on confidence throughout this thesis. We will call them ***confidence categories***. We will always use ten categories (making the intervals 0-10%, 10-20%, ...).

- **Neuron accuracy** In this chapter, we are going to need a metric that reflects whether a neuron is associated with correct or incorrect decisions of the network. We defined such a metric and called it the ***neuron accuracy***. To compute the neuron accuracy of a neuron n we feed the network with test samples. We count the total amount of samples that activated the neuron (that we will call the *total activation*). We then count, within the samples that activated the neuron, how many of them were correctly classified (that we will call the *correct count*). The neuron accuracy of the neuron is simply given by $\frac{\text{correct count}}{\text{total activation}}$.

3.1.1 Outline

- In the first phase, we try to find correlations between the characteristics of a neuron and the way it activates regarding the correctness of the network's decision.
- In phase 2, we look at how the findings in phase 1 change when we process out-of-distribution and misclassified samples.
- In phase 3 we build an out-of-distribution and misclassified samples detectors based on our previous conclusions. We then measure their performances.

3.2 Phase 1: Collecting data on the neurons and finding correlations

3.2.1 Experiments and results

We trained a network as described in the "Our networks" paragraph of Chapter 2 and got an accuracy of 95.31%.

Once the network was trained, the first factor that we wanted to measure was the calibration. We used the ECE defined in Section 3.1 and got a result of 0.57% which is really small. In order to have a visual representation of the calibration, we created a diagram similar to the ones in Figure 1.1. In this diagram (Figure 3.1), the orange bars represent the $acc(B_m)$ for each category and the blue bars represent the $conf(B_m)$ (both in percent). The number of samples in each category are depicted on top of the accuracy columns. The two blue columns on the left are far from the expected accuracy (i.e. the average confidence), but it is because there are too few samples in these categories for the numbers to be representative.

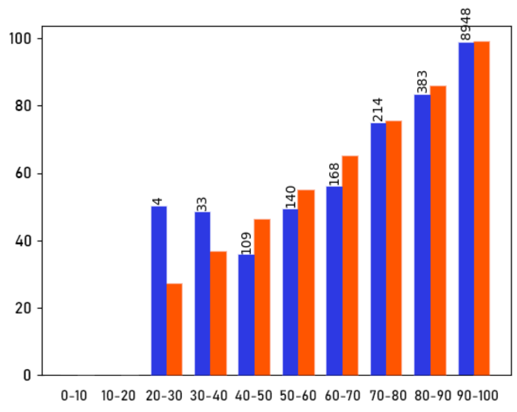
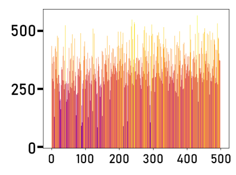


Figure 3.1: Diagram dividing test samples into 10 different categories according to the confidence with which the network classifies them. The x axis shows the interval of every category in percent. For each one, the average confidence is represented in orange while the actual accuracy is represented in blue (both in percentage). The number of samples in each category is presented on top of the accuracy columns.

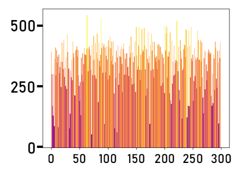
We see that the network is already well calibrated, even without the use of techniques such as temperature scaling. Therefore, the rest of our experiments will not push towards the calibration aspect of our method, but will instead explore the other points of interest of this approach. Namely, detecting out-of-distribution and misclassified samples.

For the first facet of interest, we needed to record some data. For each neuron, we computed the activation count and the average gradient (defined in Section 3.1). Once the training was completed we computed the influence of each neuron as well (according to equation 3.8). At last, we computed their neuron accuracy (defined in Section 3.1 as well) using the testing samples.

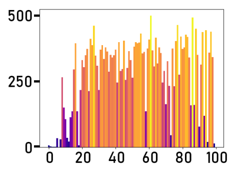
We then plotted the results of all these measures. We made five diagrams per measure. One for every hidden layer. The vertical axis represents the measure itself and horizontal axis always represents the neurons (one column per neuron). In order to highlight correlations between the measured data, we sorted the neurons in growing order of average gradient (Figures 3.2, 3.3 and 3.4) and in growing order of activation count (Figures 3.5, 3.6 and 3.7).



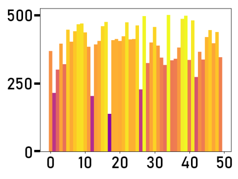
(a) Layer 1



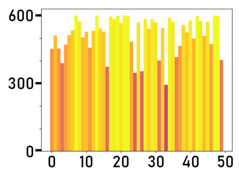
(b) Layer 2



(c) Layer 3

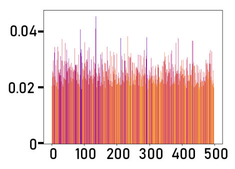


(d) Layer 4

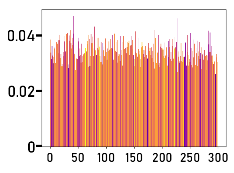


(e) Layer 5

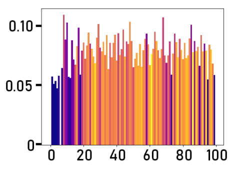
Figure 3.2: Diagrams representing the activation count of each neurons for each layer. Each column represents a neuron. The neurons are sorted in growing number of average gradient. The colors represent the activation count as well (blue for low values, dark orange for mid values and bright yellow for the highest values). Since we checked for one sample per batch if a neuron was activated, we get a total of 600 tests, which is the maximum activation count obtainable.



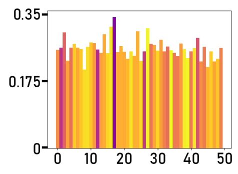
(a) Layer 1



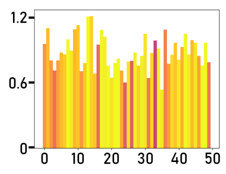
(b) Layer 2



(c) Layer 3

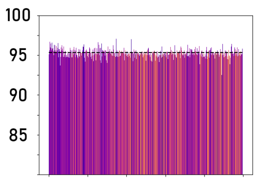


(d) Layer 4

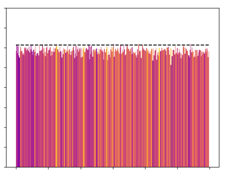


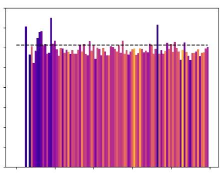
(e) Layer 5

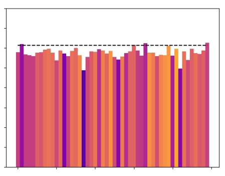
Figure 3.3: Diagrams representing the influence of each neurons for each layer. The neurons are sorted in growing number of average gradient. The colors represent the activation count (blue for low values, dark orange for mid values and bright yellow for the highest values).



(a) Layer 1


(b) Layer 2


(c) Layer 3


(d) Layer 4

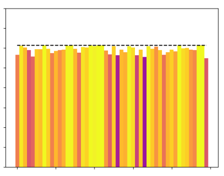
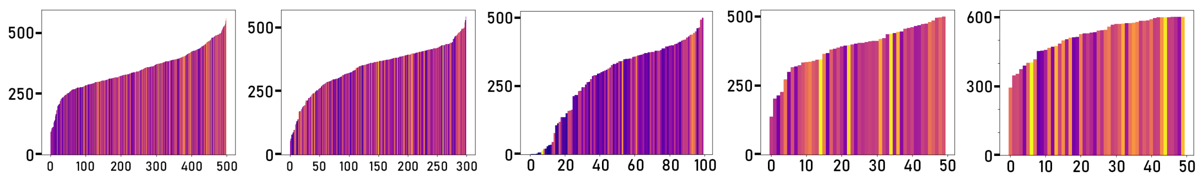
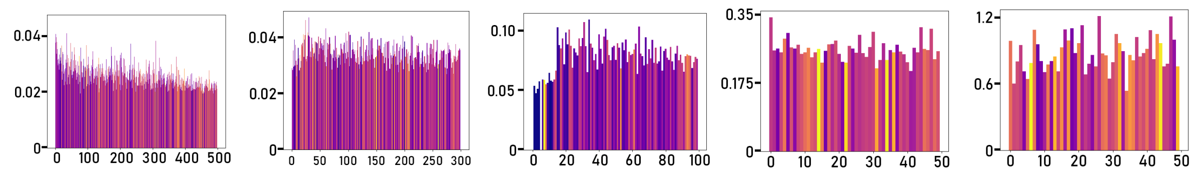

(e) Layer 5

Figure 3.4: Diagrams representing the neuron accuracies. As a reminder, the neuron accuracy is computed by taking the amount of test samples that activate the neuron and are correctly classified and dividing this amount by the total number of sample that activate that neuron. The neurons are sorted in growing number of average gradient. This time, the colors do not represent the number of times each neuron was activated during training (i.e. the activation count), but the number of times each neuron was activated during testing. Again, blue represents low values, dark orange mid values and bright yellow represents the highest values. The general accuracy of the network is represented by the dotted line (95.31%).



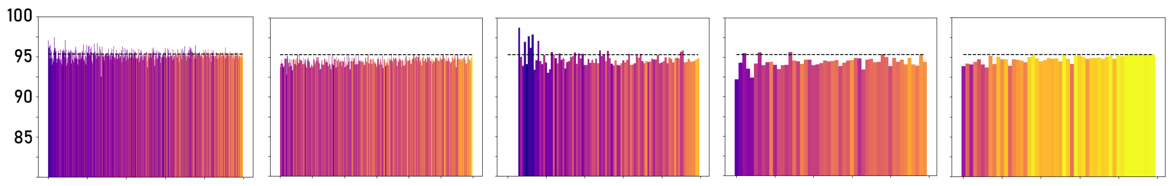
(a) Layer 1 **(b)** Layer 2 **(c)** Layer 3 **(d)** Layer 4 **(e)** Layer 5

Figure 3.5: Diagrams representing the activation count of each neurons for each layer. The neurons are sorted in growing number of activation count. The colors represent the average gradient (over the last 50% of the training). Blue for low values, dark orange for mid values and bright yellow for the highest values.



(a) Layer 1 **(b)** Layer 2 **(c)** Layer 3 **(d)** Layer 4 **(e)** Layer 5

Figure 3.6: Diagrams representing the influence of each neurons for each layer. The neurons are sorted in growing number of activation count. Again, the colors represent the average gradient (blue for low values, dark orange for mid values and bright yellow for the highest values).



(a) Layer 1 **(b)** Layer 2 **(c)** Layer 3 **(d)** Layer 4 **(e)** Layer 5

Figure 3.7: Diagrams representing the neuron accuracies. The neuron accuracy is computed by taking the amount of test samples that activate the neuron and are correctly classified and dividing this amount by the total number of sample that activate that neuron. The neurons are sorted in growing number of activation count. This time, the colors do not represent the average gradient, but the number of times each neuron was activated during testing. Again, blue represents low values, dark orange mid values and bright yellow represents the highest values. The general accuracy of the network is represented by the dotted line (95.31%).

3.2.2 Analysis

Different phenomenons emerge from the diagram we just presented:

- In the diagrams ordered by average gradient (Figures 3.2, 3.3 and 3.4) there does not seem to be any correlations, except for those depicting the third layer. It would seem that some neurons that have a low average gradient (on the left of the diagrams) behave abnormally. Figure 3.2c shows that these neurons were sparsely activated during the training, and we see in Figure 3.3c that they also present a very small influence. Finally, Figure 3.4c shows that the accuracy of these neurons is quite chaotic, with some of them showing very high levels of accuracy and others very low levels of accuracy (not even visible in the graph).

- When ordered by activation count (Figures 3.5, 3.6 and 3.7), we can also see that neurons on the left of the third layer diagrams have a small activation count and influence. They are simply the same neurons as described in the previous point. They are also on the left of these diagrams since they had really low activation counts.

But this time, other correlations can be observed: the first main one is that the accuracy seems to grow in average as the activation count rises. Especially in the last layers. The second observation is to be seen in the accuracy diagram and the influence diagram: it looks like their variance diminishes when the activation count augments.

We can draw several conclusions from these observations:

1. **Weak neurons** There does not seem to be "strong" neurons as described in Section 3.1, but normal neurons and weak neurons. The weak neurons are the ones in layer 3 that have small average gradients, activation counts and influences. We see that they have a quite chaotic neuron accuracy.

   In the line of our theory in Section 3.1, we could try to use their activation or non-activation to detect when the network is taking a bad decision. But we can already see that it wouldn't work. Their chaotic comportment and the fact that there is only a few of them in the third layer should not allow for a regular behavior.

   Our interpretation of how these neurons appeared is that the weights of these neurons were initialed in such a way that they almost never got activated to start with. When they did get activated during the training, they misled the decision, causing the gradients to strongly diminish the values of the weights between that neuron and the next layer. Cutting them off. This would explain why their neuron accuracy is chaotic: they were not trained to participate in the decisions of the network and just randomly get activated once in a while.

   Such interpretation could be tested by training our network sample by sample (batches of 1) and plotting the evolution of the influence of weak neurons and sane neurons during the training. We could then compare these plots to graphs representing the evolution of the gradient vector applied to these neurons at each iteration (again, one graph per neuron). If the proposed interpretation is correct, we should be able to see strong decreases in the influence of weak neurons each time a gradient is applied to its weights (i.e. each time the neuron is activated and has an impact on the output). But the question of how weak neurons and normal neurons emerge is outside the scope of this thesis. We leave this experiment to further research. We also leave the question of why this happens only in the third layer to another study.

2. **Activation count** As expected, a better activation count is representative of a better neuron accuracy. But we also observe a decrease in the variance of the neuron activation as the activation count increases. This could be due to two factors :

   - One, an increased activation count is synonym of more stable neurons, and this stability is reflected in the variance of the neuron accuracy.
   - Two, it is simply due to the fact that neurons that were often activated during training have a stronger chance to be activated during testing (this is reflected by the color gradient we observe in figures 3.7). Since more examples are taken into account in the measuring of the average accuracy, it is not surprising that the variance of the average gets diminished.

3. **Average gradient** We also saw that there is no correlation between the average gradient and the influence or neuron accuracy, except for the distinction between weak and normal neurons. This would mean that the average gradient clearly marks the limit between weak and normal neurons. But that, once a neuron belongs to one of the two categories, its gradient progress doesn't reflect the quality of its training.

4. **Number of activated neurons** One last observation caught our attention. Most of the neuron accuracies are below the general accuracy of the network. This can only mean that wrongly

classified samples activate more neurons in average. It is a really intriguing behavior, and we decided to further investigate on this phenomenon in the next section as well.

Now that we have a better idea of how the characteristics of neurons influence their behavior, we want to see how they behave/activate when fed with out-of-distribution samples. In the next phase, we conduct experiments that will look into these behaviors. They allow us to draw further conclusions about the observations that we made in this first phase.

3.3 Phase 2: Observing the behavior of neurons on out-of-distribution datasets

3.3.1 Experiments and results

In this second phase, we want to observe how the network behaves when given out-of-distribution samples.

To obtain out-of-distribution samples in big quantity, we resorted to two techniques. The first one was to train the network on a subset of the MNIST classes. That way, the samples from classes that the network didn't see during training become out-of-distribution samples. These samples are really interesting, because they are from the same manifold, i.e. really similar to the training samples and therefore harder to distinguish from them. We chose the classes of samples representing ones and fives as excluded samples. We chose these two classes because experiments we conducted before (that were left out of the manuscript) showed us that the architecture of network we provided distinguished ones really easily and fives with lots of trouble. This gap in classification accuracy allows for richer observations during testing. We call the set of samples belonging to the "one" class the ***ones set*** and the set of samples belonging to the "five" class the ***fives set***. The set of testing samples that belong to classes known by the network (i.e. used during training) is referred to as the ***normal set*** in the rest of this thesis.

We decided to change the architecture of our network a little to fit this new distribution. Instead of having ten neurons on the output layer, we put eight.

After the training of the new network, we plotted the same diagrams as in Figures 3.2 to 3.7 to check if the same behaviors were observable, and they were.

As a second technique to gather out-of-distribution samples, we simply created pure white noise samples. Samples for which each of the 28 by 28 values were chosen randomly in the range $[0, 255]$. To make sure our noise samples trigger a notable behavior in our new network, we decided to filter the samples and keep only the ones that were classified with a confidence above 80%[1]. We call this set the ***noise set***.

The first thing we wanted to see was how the neurons were activated when the network was fed with out-of-distribution and misclassified samples as opposed to correctly classified normal samples. To do so, we realized a few plots showing the average number of activation per neuron when fed with the different datasets.

We fed the network with the following sets: the ones set, the fives set, the noise set, the samples from the normal set that were correctly classified and the samples of the normal set that were incorrectly classified. Each samples of these sets were classified with a certain confidence. We decided to divide the samples of each set into confidence categories according to these confidences. Dividing each set into ten subsets (one per category). This is because the behavior of the network when fed with samples that trigger a strong confidence might be different from when fed with ones that trigger a low confidence.

[1]As a reminder, the notion of confidence we use in this thesis is defined in the "Network output" paragraph of Chapter 2

For every subset, we computed the number of times each neuron was activated. We then plotted the average activation of each neuron (i.e. the number of time it was activated over the total amount of samples in the dataset) in the diagrams of Figure 3.8. Every bar corresponds to a neuron. As before, we sorted the neurons within each graph in growing order of activation count. The height of each bar represents the average activation of a neuron for a given subset.

We also plotted the diagrams for the normal samples depicting the digit 3 separately. We added them in order to see how the number of activation behaved for a single class of the normal samples. It seemed logical since the out-of-distribution diagrams each represent a single "class" at the time.

We realized that the observations are comparable at any level of confidence, so we just show the graphs for the $> 90\%$ confidence category because it contains the largest amount of data.

To lighten the page, we only show the diagrams for the first, third and fifth layer as they are sufficient to visualize the main behaviors.

Another thing we did was to compute the average percentage of activated neurons per layer, per confidence category and per sample set. We display them underneath each diagram.

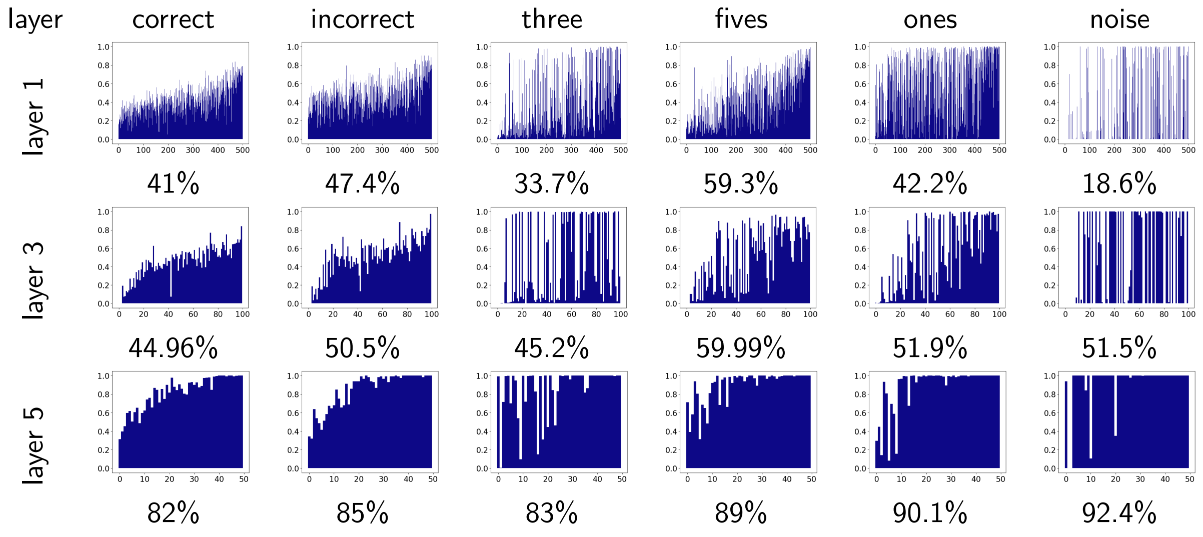


Figure 3.8: Diagrams showing the average activation of each neuron for different set of samples. Each diagram plots the results for the samples that were classified with a confidence above 90% of one of the following set: the correctly classified normal set, the misclassified normal set, the set of samples depicting a three, the fives set, the ones set and the noise set respectively. Each column represents one neuron. They are sorted in growing number of activation during training (i.e. the activation count). The average percentage of activated neurons is given under each diagram.

We try to interpret these results in the next section.

3.3.2 Analysis

Class specific neuron patterns By looking at the diagrams, we noticed that gaps in the number of activations between close neurons get bigger as the samples get further away from normal. This is probably due to the fact that there are more intrinsically different samples (since there are multiple classes) present in the normal sets. To verify this, we added the diagram for a single class of the normal set (the threes) and indeed got an oscillating result as well. This oscillation reflects the fact that neurons are either activated almost every time by a certain class or rarely activated by that class. Values in between are rare. This tells us that samples of a certain class activate very specific patterns of neurons. These patterns could potentially be exploited to see if a sample is correctly classified or not. Indeed, it is interesting to see if a sample classified as an eight activates the same neurons as most eights. We do not follow this trail further in this chapter because (and the attentive reader might have figured it out already) this is exactly the track we follow in our second approach (Chapter 4).

Activations corresponding to noise samples It is also interesting to note that the noise samples, despite their randomness, tend to all activate a very specific portion of the neurons as well. Even more than the other classes. We looked at the class the network assigns them to and saw that they were almost all classified as twos. We retrained several networks and noticed that they always classify noise samples in one or two specific classes. The classes that come back are the twos, the threes, the fives and the eights. We analyzed the MNIST database further to see what these digits had in common. We measured how "spread" the samples were in average for each class. We measured the spread of a sample by multiplying the value of each pixel with its distance from the center and adding up these values for all the pixels. The digits that had the strongest spread were the ones cited before and the zeros.

This shows that the network is strongly influenced by the variance and the spread of digits. Almost as much as the shapes themselves.

The reason why the zeros do not become a typical noise class is probably that there is a strong negative zone in the center. In other words, each network sets up a mechanism that prevents samples that have high values in the center pixels from activating the neurons in the same way a zero would.

We still do not know why noise samples are never classified as other classes however. We let the interpretation to the reader.

Number of activated neurons Finally, we noticed that out-of-distribution and wrongly classified samples activate more neurons than correctly classified normal samples. This corresponds to what we observed in the "Number of activated neurons" paragraph in Section 3.2.2.

We further experiment and speculate on this phenomenon in phase 3.

3.4 Phase 3: Usability of the data on out-of-distribution detectors

3.4.1 Experiments and results

As we observed in Section 3.2, the one measure recorded during training that actually has a visible influence on the quality of a classification is the number of activation during the training (i.e. the activation count). This is slightly visible in Figure 3.8. The distribution of the spikes spreads to the left as the samples get more different from the in-distribution samples. This shows that, when more neurons with high activation count are activated, we are more likely dealing with an in-distribution sample. From this observation we defined a ***neuron activation coefficient*** for each sample. The formula for the neuron activation coefficient for a layer l and a sample x^i being given by:

$$neuron\_activation\_coefficient(\vec{x}^i) = \sum_{n=1}^{r^l} 1(a_n^l \neq 0)(\max_{n,l}(ac_n^l) - ac_n^l) \tag{3.13}$$

Where r^l denotes the number of neurons in layer l, a_n^l defines the output of the n^{th} neuron of layer l and ac_n^l is the activation count of that neuron. $1(a_n^l \neq 0)$ is equal to 1 if true, 0 otherwise.

By computing the coefficient this way, samples that have a higher score are the ones that activate lots of neurons and for which the activated neurons have a smaller average activation count. This means that out-of-distribution and misclassified samples should score higher than correctly-classified normal samples.

We implemented out-of-distribution detectors based on a threshold on this neuron activation coefficient. Samples that score above the threshold are considered out-of-distribution or misclassified (here considered as the positive category). Samples that score below are considered well-classified and in-distribution (considered as the negative category). We could then plot the ***ROC curves***[2] of these detectors, allowing us to have a mathematically relevant representation of the expressiveness of our coefficient. We created ten detectors per layer. One for each confidence category.

We tested the detectors on different sets. The first set was simply the normal set described in Section 3.3. For this set, we consider misclassified samples as the out-of-distribution (i.e. positive) samples. The other sets were made by combining samples of the normal set with samples of one of the out-of-distribution sets described in Section 3.3 (i.e. the ones, fives and noise sets). The ROC curves resulting from applying the detectors on these new sets for each layer are represented in Figure 3.9. Again, the confidence categories all show the same behavior. Therefore, we only show the plots for the highest confidence category.

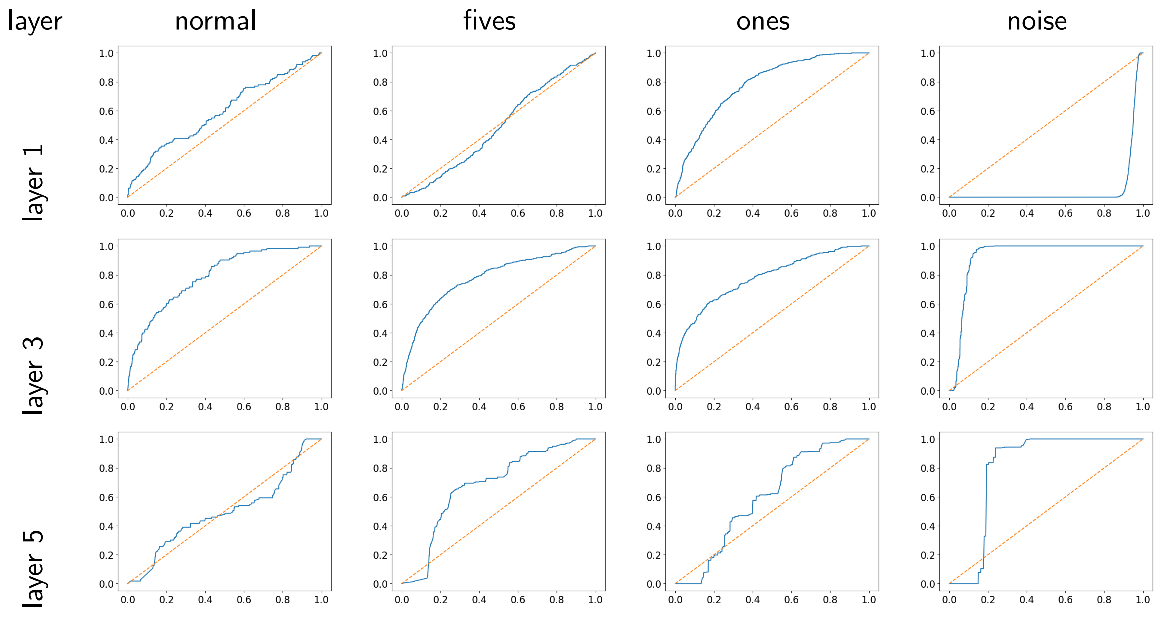


Figure 3.9: ROC curves of out-of-distribution and misclassified samples detectors based on a threshold on the neuron activation coefficient. Samples that score higher than the threshold are considered out-of-bound or misclassified (considered the positive category). The first column shows the curves when applying the detector on the normal set, trying to detect misclassified samples. The columns 2, 3 and 4 depict the curves when when applied on a combination of the normal set and one of the out-of-distribution sets (fives, ones and noise respectively).

We can see that the detectors are able to distinguish out-of-distribution samples to a certain degree. However, the detectors use the number of activated neurons and the average activation count *together* to classify the data. But we also wanted to see how detectors based on each of these factors *individually* performed.

Therefore, we repeated the previous steps to construct detectors. But instead of using the neuron activation coefficient, we created detectors that use the number of activated neurons and other detectors that use the average activation count. The average activation count is off course computed by adding up the activation count of all the activated neurons and dividing that number by the number of activated neurons.

The ROC curves of these new detectors are plotted in figures 3.10 and 3.11 respectively. We still only show the graphs for the samples classified with the highest confidence.

It is important to note that, unlike the other detectors, the detectors based on the average activation count considers samples with a score above the threshold as correct and in-distribution.

[2]ROC (Receiver Operator Characteristic) curves are a common visual tool to apprehend the quality of a binary classifier. The curve represents different values of threshold. The vertical axis corresponds to the true positive rate and the horizontal axis represents the false positive rate. A curve that reaches the top left corner is ideal as it detects all the true positive samples without mistaking a negative for a positive (no false negatives).

This is because the average activation count is supposedly higher for these samples.

We analyze these curves in the next section.

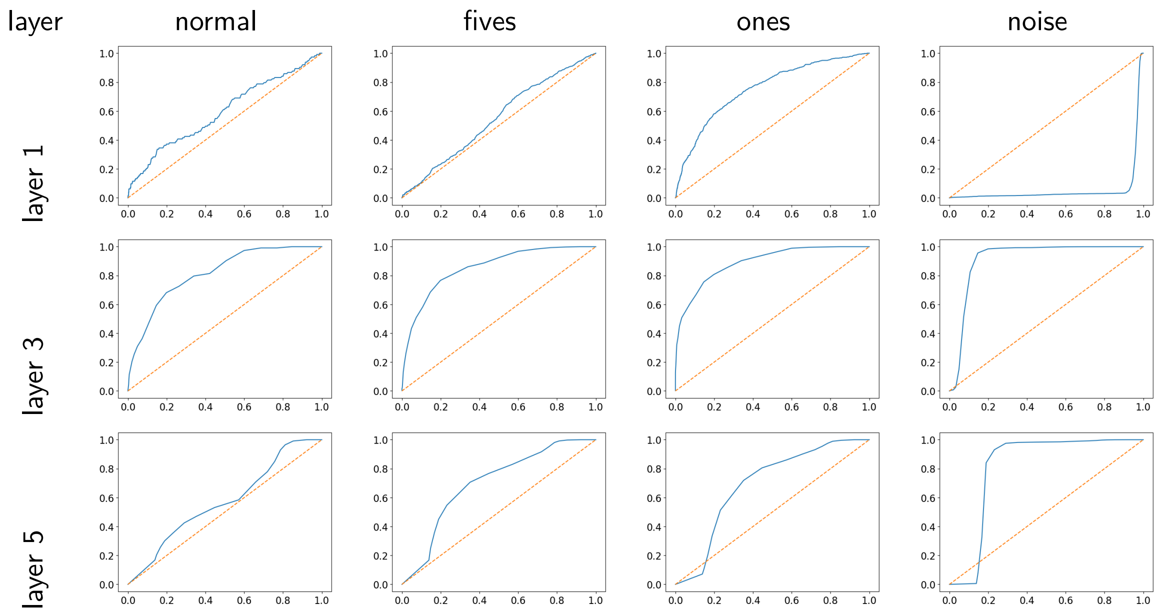


Figure 3.10: ROC curves of out-of-distribution and misclassified samples detectors based on a threshold on the number of activated neurons. Samples that score higher than the threshold are considered out-of-bound or misclassified (considered the positive category). The first column shows the curves when applying the detector on the normal set, trying to detect misclassified samples. The columns 2, 3 and 4 depict the curves when when applied on a combination of the normal set and one of the out-of-distribution sets (fives, ones and noise respectively).

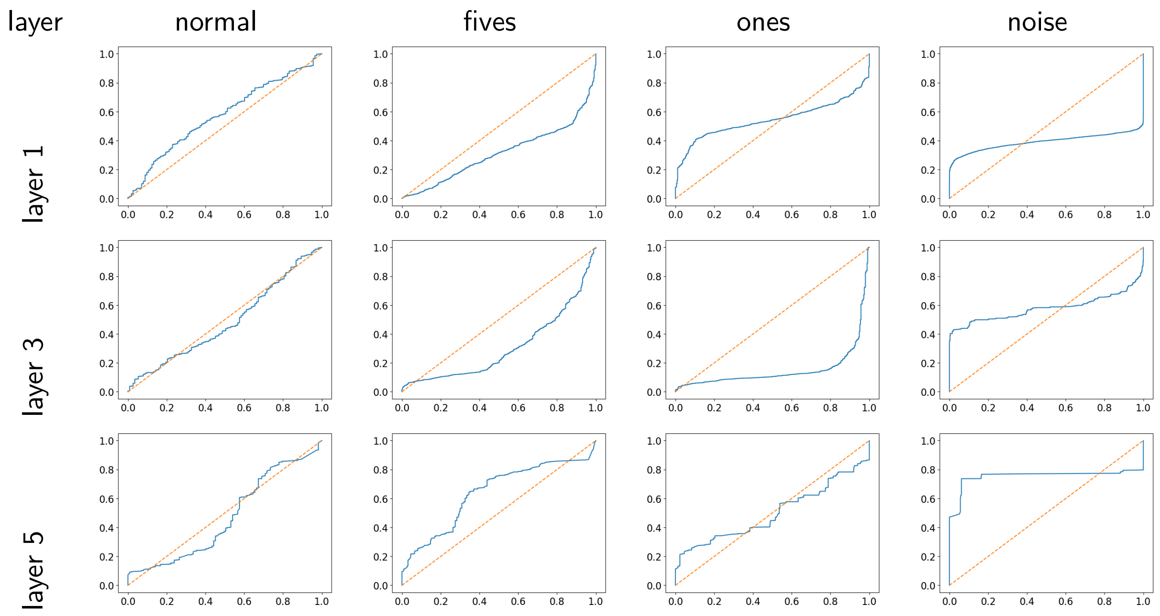


Figure 3.11: ROC curves of out-of-distribution and misclassified samples detectors based on a threshold on the average activation count. This time, unlike the other graphs, the samples in the positive category are the correctly classified in-distribution ones. Meaning that samples with a score above the threshold are considered correct and in-distribution. The first column shows the curves when applying the detector on the normal set. The columns 2, 3 and 4 depict the curves when when applied on a combination of the normal set and one of the out-of-distribution sets (fives, ones and noise respectively).

3.4.2 Analysis

When we look at figures 3.10 and 3.11 we realize that the positive results of our neuron activation coefficient based detectors were only due to the counting of activated neurons. The use of the mean activation count was useless, even disruptive for the lower and middle layers.

What can we learn from these observations?

Number of activated neurons It seems that the number of activated neurons is the best indicator of abnormality of a sample (i.e. how different it is from the training data). Our interpretation of this is that a specific activation pattern corresponds to each class (as observed in Figure 3.8). When the network sees a sample for which it is not sure of the affinity, it tends to consider it as potentially belonging to multiple classes. It thus activates the patterns corresponding to each one of the concerned classes.

This interpretation goes hand in hand with the theory of our next approach about clusters of activation pattern within classes. We present leads that could find a link between this approach and the next one further in Chapter 5. In that Chapter, we also compare our number-of-activated-neuron based detector to different benchmarks.

Activation count Despite the fact that a high activation count seemed to be synonym of a more accurate neuron in the last layers, the center diagrams of Figure 3.11 show us that the average activation count of a sample has the exact opposite meaning in the center layers of the network. Namely, out-of-distribution and wrongly classified samples tend to have a higher average activation count. Here is a possible interpretation of this phenomenon.

The activation count of a sample is not representative of how well a neuron was trained at all[3]. A high activation count just means that a neuron gets activated more often in general. The reason neurons with high activation counts have a higher accuracy is simply that their frequent activation allows their accuracy to get closer to the network's overall accuracy[4]. And since most of the neuron accuracies are below the overall accuracy, getting closer to it is a synonym of augmentation.

This explains why neurons that have a high activation count also have a higher neuron accuracy, independently of the actual quality of their activation. But it doesn't explain why the average activation count is higher for out-of-distribution samples than for normal ones.

One explanation could be the presence of two types of neurons in the middle layers. Some neurons are class specific while others are more generic. Class specific neurons have lower chances of getting activated, and when they are, it is more likely by a sample that clearly belongs to the corresponding class. Generic neurons, on the other hand, are not as precise (they are probably activated to indicate the presence or absence of a general feature in the sample). They are therefore activated more often (i.e. bigger activation count).

Since both in and out-of-distribution samples activate generic neurons, but out-of-distribution samples activate less class specific neurons, we end up with average activation counts that match our observations.

However, this interpretation doesn't explain why out-of-distribution and misclassified samples activate more neurons in general. According to it, the opposite would make more sense. Since both interpretations are still at their speculative stage, it is difficult be sure of where they diverge and where they agree. But further work trying to reconcile both interpretations could lead to interesting discoveries.

[3]Remember Section 3.1, where we explained that a high activation count was synonym of more updates of the weights during the training. We thought that neurons whose weights were activated more often during training would have converged better since they had more opportunities to find good values.

[4]We can indeed see that the curve in Figure 3.7e would be even more regular if the neurons were sorted by number of activation instead of by activation count

Average gradient and influence Coming back to our initial hypothesis, we learned that the average gradient combined with the influence allows us to uncover weak neurons. But these weak neurons are just passive in the network, and thus do not help in the detection of out-of-distribution samples.

Chapter 4

Clustering on neurons' activation patterns

4.1 Concept, hypothesis and terminology

When a human is asked to classify an image he sees for the first time, the confidence with which he classifies the image is proportional mainly to two things: how many examples close to this image he saw before, and how close those examples actually are to the image. This is interesting because, not only are humans good at knowing what they don't know, but a human can also present examples from his memory that express why he thinks an image belongs to a certain class.

Our second approach aims at reproducing this behavior with neural networks. The notion of how close two examples are together for humans depends on our intrinsic perception of these examples. Here, we will consider that the way a neural network perceives a sample is how its neurons activate. When we talk about the activation of a neuron, we are talking about the value of its output. A neuron that outputs a zero is considered inactive, and a neuron whose output is greater than zero is considered active. Each sample activates the neurons differently, forming different activation patterns.

Two similar examples (should) activate the network in a similar fashion. This means that out-of-distribution samples should have activation patterns that are not close to the activation patterns of any training sample.

Note. *For text clarity, we will now refer to a sample for which we are currently analyzing the activation pattern as "the sample", whereas the previously seen samples (the training samples for which we recorded the activation patterns) to which we compare that "sample" will be referred to as "examples".*

With activation patterns representing the way a network perceives a sample, finding which previously seen examples are similar to this sample (continuing the human/network comparison) comes down to finding the examples that have a close activation pattern. Once we have a group of similar examples, we can measure the average distance between the activation pattern of the sample and the rest of the group (giving us an idea of how "similar" it is to the close examples). A sample that is not similar to its close examples is probably an outlier.

Distance In order to group the activation patterns together we need to define a distance measure between them. Since our goal is to find out whether activation patterns contain information on their own, regardless of the strength of the activation, we decide to binarize the patterns. Each layer's activation is represented by an array of 1 and 0. A 1 represents the fact that the corresponding neuron was activated for a given example and a 0 means it wasn't. The distance between two vectors can then be measured by the Hamming distance:

$$d(\vec{a}, \vec{b}) = \sum_{n=1}^{r^l} 1(a_n \neq b_n) \tag{4.1}$$

With r^l being the length of layer l. $1(a_n \neq b_n)$ is evaluated to 1 if $a_n \neq b_n$, 0 otherwise. As you can see, we measure the distance between layers instead of over the entire network at once. This is for two reasons:

- The behavior of the activations is probably very different between early layers and layers close to the output. We want to be able to observe these different behaviors as closely as possible.
- As explained later, the progress itself of some measures through the layers will be important.

One problem arises when measuring the distance the way we do. When measuring the distance between two binary vectors, if either one or zero is outnumbered in both vectors, the maximum distance between these vectors is the sum of the outnumbered category in each one of them. We solved this problem by using percentiles to define if a neuron's activation should correspond to a 0 or a 1. We take a fraction f between 0 and 1. For a given sample, we look at the outputs of the neurons of one layer. We then compute the binary pattern of a layer l (of size r^l) by assigning a 0 to all the neurons whose output belongs to the $(f * r^l)$ smallest outputs and a 1 to the others. This technique is a little less rigorous, but it allows us to differentiate the patterns much better. Since each vector now has the same number of 1 and 0, the fraction that allows to use the full spectrum of distances is $f = 0.5$ (that way, neither 0 nor 1 are outnumbered).

Grouping There are several ways to group similar patterns based on distance. We are going to explore two of them:

1. **Clustering**: For each layer we divide the examples into a certain amount of clusters based on the distances between their activation patterns. We will use k-means as clustering algorithm. The close examples of a sample are simply the ones from the clusters it is assigned to. The group of close examples is then different in each layer.

   We explain in Section 4.2.1.1 how we choose the number of clusters in each layer.

   Here are some concepts we are going to use:

   - The ***silhouette coefficient*** (or ***silhouette score***) is a measure that indicates how strongly a sample belongs to a cluster. The value is in the $[-1, 1]$ interval. 1 corresponds to a sample that is perfectly classified while -1 represents the fact that the sample should belong to another cluster. A value of 0 corresponds to a sample that is in an overlap between two clusters. If a is the mean distance between a sample and all the other samples from its cluster and b is the mean distance between that sample and all the sample from the second most fitting cluster, the silhouette score is given by $\frac{b-a}{max(a,b)}$.
   - The ***center sample*** of a cluster is the sample that is the closest to the center of the cluster.
   - The network predicts the class of all the samples that are fed to it. Each example in a cluster has its own predicted class. We will call the class that is the most represented in a cluster the ***dominant class*** of that cluster.

     When a sample is fed to the network we can find to which cluster it belongs in each layer and its class is predicted. For every layer it is now possible to divide the samples into two groups: the ones for which the dominant class of the cluster they were assigned to matches their predicted class and the ones for which it doesn't match. We will call these groups the ***matching samples*** and ***mismatching samples*** respectively.

- We will use a coefficient that we call the ***Cluster Distance Coefficient*** (***CDC***). The CDC of a sample $\vec{x}^i$ on a layer l is computed as follows:

$$CDC_l(\vec{x}^i) = \frac{\mid cluster(\vec{p}_l^i) \mid d(\vec{p}_l^i, center(cluster(\vec{p}_l^i)))}{\sum_{\vec{p}_l^j \in cluster(\vec{p}_l^i)} d(\vec{p}_l^j, center(cluster(\vec{p}_l^i)))} \tag{4.2}$$

  Where $d(\vec{a}, \vec{b})$ is the euclidean distance between vectors $\vec{a}$ and $\vec{b}$ and $\vec{p}_l^i$ is the activation pattern of $\vec{x}^i$ in layer l. $cluster(\vec{p}_l^i)$ is the set of training samples' activation patterns that belong to the cluster of $\vec{p}_l^i$. $center(C)$ is the center vector of the group of vectors C.

  The CDC of a sample $\vec{x}^i$ on a layer l represents the distance between the sample's activation pattern in layer l and the center of the corresponding cluster, divided by the average distance to the center of the training samples from the same cluster.

2. **Nearest neighbors**: This grouping technique is quite straight forward: for a given sample we find the k closest examples in a given layer. In this chapter we are going to use $k = 300$[1].

   Again, we need to define a few concepts:

   - We will call the k examples that are the closest of a sample in a certain layer l the l^{th} ***layer group***. We also give a specific name to the examples of the layer group of the last layer: the ***final group***. We give a specific name to this group because we are going to analyze the relation between the samples of this group in all the other layers as well.
   - The average distance between a sample's activation pattern and the activation patterns of its group will be called the ***average group distance***.
   - The percentage of examples of a group that share the same predicted class as the main sample will be called the ***matching class percentage***.

Theories As in Chapter 3, we would like to see if it is possible to distinguish samples that the network will classify wrongly (samples it is unfamiliar with) from samples that will be classified correctly. We want to know if we can make this distinction by looking at the activation patterns of the neurons. In particular, if comparing the activation pattern of a sample with previously seen examples allows us to tell whether it is an outlier or not.

To represent familiar and unfamiliar samples, we are going to use the same datasets as presented in Section 3.3. Namely, we will train the networks on a subset of MNIST. This subset includes all the samples representing digits from zero to nine from which we removed the samples representing ones and fives. We consider these removed samples as two new datasets that we will call the ***ones*** set and the ***fives*** set. The remaining MNIST samples will be called the ***normal*** set. We also created a set of random inputs (with a uniform distribution) forming samples of pure noise. From these samples of noise, we selected only the ones that triggered a confidence[2] above 80% when fed to the network. We call these selected samples the ***noise*** set.

We will consider correctly classified normal samples as examples of familiar samples. The noise, ones and fives set as well as wrongly classified normal samples will be considered examples of unfamiliar samples.

We have two ideas of how one could use the grouping of similar activation patterns to detect abnormal samples:

1. The first one is the most intuitive one and is based on clustering. When we run a network on a sample, we record the activation pattern in one layer. We compare the pattern to the recorded

[1]It seemed like a reasonable amount because, since we are using the 10000 MNIST test samples for our experiments it does not reach the amount of samples of an entire class. It also leaves room for the sample to be in a subclass but without being a too small number.

[2]As a reminder, we are using the confidence as defined in Chapter 2 in the "Network output" paragraph.

patterns of previously seen examples and find out which cluster it belongs to. We measure how close the pattern is to the center of the cluster and see how it compares to the overall density of the cluster. We combine this information with the dominant class of the cluster to get a prediction of the class this sample should belong to. We then put this prediction against the actual output of the network and see if they match. The idea is that the integration of a sample in its cluster should indicate if the sample is regular or unusual.

This approach may seem like we are simply double-checking the decision of our network by comparing it to the output of a k-means classifier. But this is not exactly the case. Here, the k-means classifier doesn't analyze the raw data, but uses the activation patterns inside the network as features. This means it classifies the intrinsic representation of the data.

2. The second idea reveals less information about the "memories" of a network, but more about the way a network processes a sample. The intuition is that a network "encodes" a sample (Shwartz-Ziv et al. 2017) such that, at each new layer, its representation (here, the activation pattern for that layer) gets closer and closer to samples of the same class. Until the final layer, where it is finally classified. In theory, two samples that are classified in the same class are so because of one of two reasons. Either they are very similar from the beginning, or it is as a result of how the network encoded them. Ultimately, samples of the same class, but different subclasses that have very different entry values end up activating the final layer the same way.

   Here, we will group a sample with close previously seen examples by finding the k closest examples.

   When we group samples according to their activation pattern in the last hidden layer (i.e. the final group), we can see how the density of that group behaves as we go backwards through the layers. We would like to record these behaviors for regular samples (samples that are close to the archetype of their class or subclass) and for unusual samples (samples that don't resemble any of the training samples). We then want to see if the evolution of the activation patterns of a sample compared to the evolution of the rest of the final group through the layers allows to have an idea of how "normal" that sample is. A sample that is close to the examples of its final group in the last layer, but that gets further and further away as we move back in the layers of the network, even though the other examples stay tightly together, would be suspicious.

4.1.1 Outline

We proceed per idea :

1. Idea 1 :
   - In the first phase we look at the behavior of the cluster distance coefficient in different use cases to see how it behaves regarding out-of-distribution and misclassified samples. We see if these behaviors coincide with our hypothesis.
   - In phase 2 we try to construct threshold-based detectors that uses the observations of the previous phase. We measure the performances of our detectors and plot their ROC curves.
2. Idea 2 :
   - In the first phase, we look at the progress through the layers of the average distance between a sample and its final group. We do this separately for correctly-classified in-distribution samples, misclassified in-distribution samples, samples from the ones dataset, the fives dataset and the noise dataset. This shows us if there is indeed a difference between the progress of the activation patterns of regular samples and unusual samples.

     We then repeat the same tests, but instead of measuring the average group distances, we measure the matching class percentages.

- In phase 2, we build two different neural-network-based out-of-distribution and misclassified samples detectors. They both use the observations that were made in the previous phase as well as our hypothesis. We also measure the performances of these detectors and plot their ROC curve (we explain in Section 4.3.2.1 how we create ROC curves for network-based binary classifiers).

4.2 Idea 1: Direct clustering observations

4.2.1 Phase 1: Observing how well different samples lie within their clusters

In this first phase we test how well our intuition applies in practice. We find out if some measures are indicators of erroneous classification.

4.2.1.1 Experiments and results

We used the same network as in Section 3.3 and trained it on the same dataset (MNIST from which removed the ones and the fives $\rightarrow$ 8 classes in total) in order to be able to test the accuracy of our method on detecting out-of-bound samples later on.

We first needed to prove that clusters of activation patterns indeed represent meaningful groups of samples. To do so, we recorded the layers' activation patterns of the training samples and discovered the clusters among them using K-means. In order to discover the number of intrinsic clusters, we applied K-means multiple times. Each time with a different number of clusters going from 2 to 32. Every time, we computed the mean silhouette coefficient of all the samples. We then chose the number of clusters that led to the best average silhouette score. For each layer, the chosen number of cluster was 8. We then found, for each cluster, the dominant class as well as the class of the center sample. It turns out each layer has exactly one cluster per class. This is good news since it means that the activation patterns do represent meaningful group of samples. In Chapter 3 we saw that classes of samples activate specific patterns of neurons already since layer 1. But we weren't sure whether each class had exactly one archetype of activation pattern that was different from all the other classes. Now knowing that it is the case, we can go on with our experiments.

Let us then move on to our first experiment. As for our first method, we start by creating a visual representation of the relevant information. This will allow us to see where the results correspond and diverge from our expectations. Only once we have wrapped our heads around this information can we move on to the next step. In the next step (phase 2) we check if the observations made here allow detecting potentially erroneous behaviors in the network.

The relevant information we want to take a look at in this context is the relationship between the distance that separates a sample from the center of its cluster and the reliability of its classification. To highlight this relationship visually, we plotted the average cluster distance coefficient (defined in Section 4.1) per layer of different datasets. Namely, the normal, the ones, the fives and the noise sets. We divided the normal set in two by separating the correctly classified samples from the incorrect ones.

The contrast between the average CDC of the correct normal samples set and the other datasets should show us if there exists a correlation.

Before creating the diagrams for each layer, we decided to further divide the datasets to make finer observations. For each layer we split the datasets in two: matching samples and non-matching samples (see definition in Section 4.1).

We thus created ten diagrams per layer, two per sets. In each diagram we separated the samples according to the confidence with which they were classified. We made ten confidence categories (see the "Concepts and definitions" paragraph of Section 3.1) and computed the average CDC for each one of them. The columns in the diagram represent the confidence category and are sorted from the weakest confidence (0-10%) to the strongest confidence (90-100%). The height of each column corresponds to the average cluster distance coefficient.

The layer on which the differences are the most visible is the second layer. Its diagrams are represented in Figure 4.1.

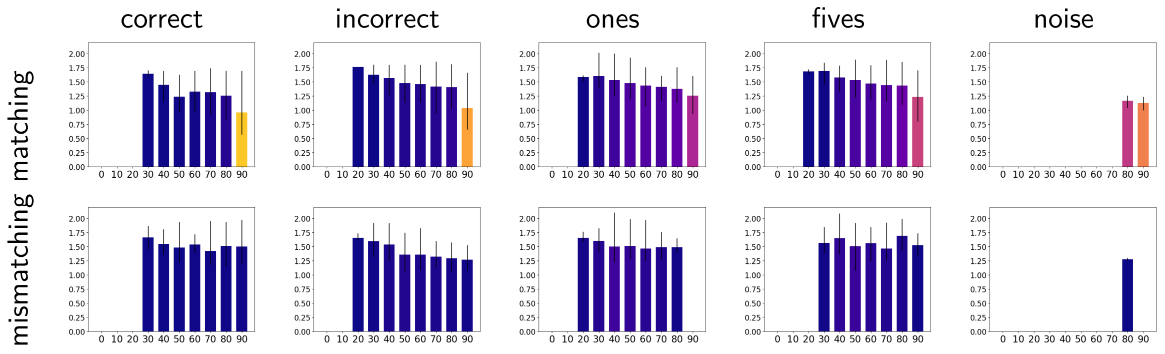


Figure 4.1: Diagrams representing the average CDC per confidence category of different sets. Each column represents one of the ten confidence categories and are sorted (left to right) from weakest confidence (0-10%) to strongest confidence (90-100%). The colors represent the amount of samples corresponding to each category. Blue corresponds to low values, orange mid values and yellow the highest values. The black lines on top of each column represent the total range of a category. Samples whose predicted class matches with their cluster have been separated from the ones that do not match. Diagrams representing matching samples are presented in the top row while samples representing mismatching samples are represented in the bottom row.

4.2.1.2 Analysis

Quite some information emerges from these diagrams. Let us begin by pointing out what we expected:

- In the matching sets, we can see that the average CDC is indeed lower in the normal sets than in the others for samples with a high confidence.
- There is a correlation between the confidence and the CDC. It was expected that the average CDC would drop as the confidence would increase since the network would be more familiar (i.e. more confident) about samples that activate the network in a regular fashion (i.e. samples that have an activation pattern close to the previously seen patterns of the same class or subclass).

Now what we didn't foresee:

- The average CDC of confidence categories below 90% do not allow for a clear distinction between in and out-of-distribution samples. This means that out-of-distribution samples are processed in the same way as perplexing normal samples.
- The noise samples seem to all be in the matching set. This reinforces what was previously observed, namely that the network has one class that stands way out for noise patterns.

The fact that there is a difference between the mean CDC of the normal confidently classified samples and the out-of-distribution confidently classified samples means that it is possible to create an out-of-distribution detector. However, seeing how broadly the CDCs can vary within one dataset compared to how small the difference is between in and out-of-distribution, makes us think that the distinction will be quite hard to make. In the next section we build a few threshold based detectors to see how well the CDC allows us to distinguish the samples.

A little note could be made on the clusters in the early layers. As observed in the experiments, the activation patterns are already mostly grouped by class in the first layer. This is really fast. It reinforces the proposed interpretation made in Section 3.3, that suggested that each class has typically one way of activating the neurons.

It also explains why noise samples all activate the network in such similar patterns all the way from layer one. The network "has" to fit them in a class directly. It thus picks the class that they are the closest to (apparently, the same class for every noise samples) and activates according to the pattern archetype of that class.

4.2.2 Phase 2: Usability of the data on out-of-distribution detectors

Here, we apply our observation made in phase 1 to try to detect out-of-distribution samples. We build detectors based on the CDC of samples and analyze their performance to see how good the CDC is in revealing abnormal samples.

4.2.2.1 Experiments and results

We implemented a simple out-of-distribution and misclassified samples detector. The detector uses a threshold on the CDC of samples. If the CDC is above the threshold, the sample is considered erroneous. The sample is classified as correctly-classified and in-distribution otherwise. We decided to test the detector on different sets of samples to see how it performed on each one separately. The main sets were assembled by combining the normal samples set with one of the out-of-distribution set (ones, fives and noise sets). We also tested the detector on the normal set to see how well it detects misclassified samples.

Since we observed in the previous section that the CDC behaves differently for each confidence category, we tested our detector on samples of each confidence category individually. We also tested it on every main set with all the categories together and separated the matching and mismatching samples for each.

What we mean by "testing" the detector is plotting the ROC curves for each dataset. The erroneous samples are considered as the positive category.

It turned out that the detector performed as well on all the confidence categories together as when separating them. It is due to the fact that normal samples classified with a confidence below 90% are rare enough not to have a big impact on the false positives. Thus, from now on, we won't separate the datasets by confidence category for the CDC based detector.

The ROC curves for each set are presented in Figure 4.2. We only show the diagrams of layer two as it got the best results.

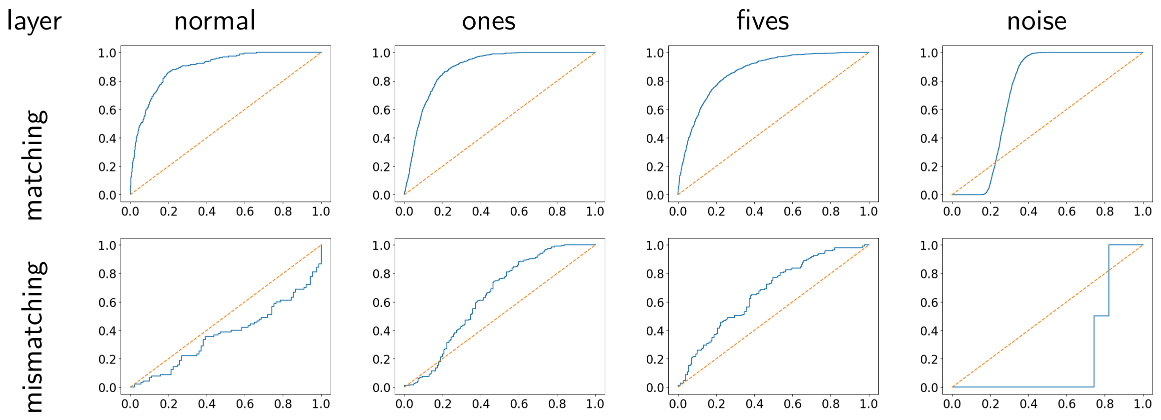


Figure 4.2: ROC curves of the out-of-distribution detector for each main dataset. On top are the ones when fed with matching samples and below are the ones on mismatching samples.

4.2.2.2 Analysis

As expected, if we omit mismatching samples, the CDC allows to make the difference between in-distribution and out-of-distribution samples as well as between correctly and incorrectly classified samples with a reasonable accuracy. We compare the performances of this approach with our other approaches as well as some general benchmarks in Chapter 5.

4.3 Idea 2: Consistency of the clusters when backtracking through the layers

As explained briefly in Section 4.1, given a certain sample, we are going to find the k training samples that have activation patterns that are the closest to that sample's activation pattern in each layer. We called these group of training samples the layer groups (and the last layer's layer group: the final group).

As said in Section 4.1, the activation patterns of a sample that truly belong to the same class/subclass as the samples from its final group should not go too far from them as we go back in the layers. Samples that do not truly belong to the same class/subclass should drift apart from the other samples as we recede through the layers.

4.3.1 Phase 1: Observing the consistency of clusters through the layers for different samples

4.3.1.1 Experiments and results

What we want to observe here is simply the behavior of the activation patterns with respect to their layer groups and their final group through the layers. Two main relations between a sample and its groups are interesting to measure. We defined them in Section 4.1:

- The average group distance: This measure is especially interesting to compute for the final group of a sample in each layer. It gives us a good idea of how a sample drifts apart or stays tightly together with the samples from its final group.
- The matching class percentage: This is only interesting to measure for the layer groups. There is no need to compute it for the final group in each layer since it doesn't change from layer to layer.

We thus looked at the behavior of these two measures for different samples. We used the same network as in Section 3.3 trained on MNIST from which removed the ones and the fives. We computed the mean matching class percentage and average group distance for the samples of the different datasets described in Section 4.1 (the normal set, the ones set, the fives set and the noise set). We separated the misclassified normal samples from the correctly-classified ones to form two sets. This allowed us to compare the behaviors of our measures between erroneous samples and correctly-classified in-distribution ones.

We created two diagrams per dataset. In the first diagram, we computed the average final group distance of each of its samples in every hidden layer. We then plotted the average over all its samples for every layer separately. In the second diagram, for every sample in a set, we computed the matching class percentage of each layer group (one per hidden layer). And again, plotted the average per layer. The plots are presented in Figure 4.3.

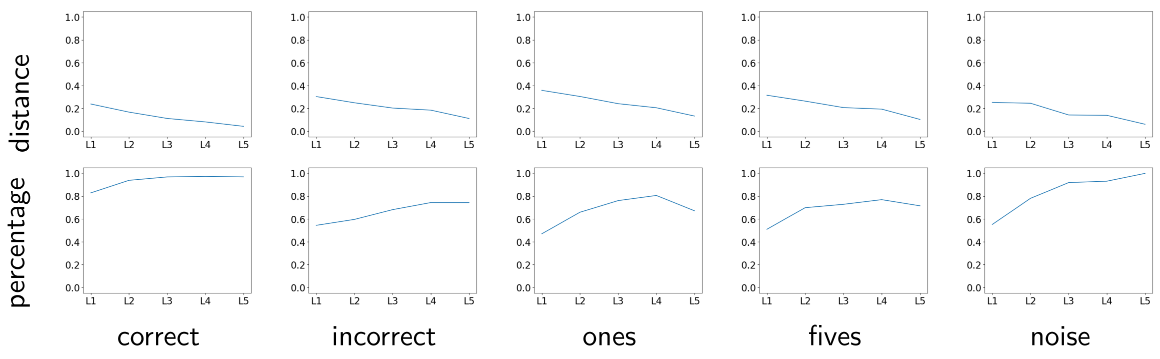


Figure 4.3: The figures of the top row represent for different datasets the mean average group distance between samples and their final group in each layer. The bottom row shows diagrams for these datasets depicting the evolution of the average matching class percentage through the layers. The horizontal axis represents the layers each time (layer 1 on the far left and layer 5 on the far right). (the group of a sample is the set of training samples that are similar to that sample in a given layer)

4.3.1.2 Analysis

Matching class percentage We can see in Figure 4.3 that the matching class percentage curve stays much higher in the correctly-classified normal dataset than in the others as we move back in the layers. Meaning that normal samples tend to have activation patterns that are actually close to the activation patterns of samples from the same class/subclass. And so since the first layers of the network. On the other hand, the activation patterns of out-of-distribution samples artificially converge to samples of the same class. What we call artificial convergence is the fact that the network tries to make samples fit in the class they are closest to. Making their patterns a little more similar to the patterns of the samples of that class at each layer.

In the next section we try to see if we can use this observation to detect out-of-distribution samples consistently.

Average last layer group distance In the same figure, we see that the average group distance is slightly higher overall for out-of-distribution and misclassified samples. Showing that the latter are more isolated in the activation pattern spectrum throughout the layers. This means that, despite the efforts of the network of merging the activation patterns in the last layers, the patterns of out-of-distribution samples are still different. The difference between the figures is less flagrant than for the matching class percentage, but we will still see if we can predict if a sample is normal and well-classified or not using this information.

4.3.2 Phase 2: Usability of the data on out-of-distribution detectors

4.3.2.1 Experiments and results

As done in the previous sections, we wanted to measure to which extent the observations made in phase 1 allow us to create an out-of-distribution samples detector. The quality of the detector gives us a good idea of how much information the behavior of the average group distance (regarding the final group) and the matching class percentage gives us regarding how normal[3] a sample is.

We created two detectors. One based on the average group distances of each layer regarding the final group, and the other one based on the matching class percentage of each layer. Unlike before, we won't make simple threshold based detectors. This is because we do not just want to analyze a single value at the time. We want our detectors to take the progress of the values through the layers into account as well.

[3]By normal, we mean how close a sample is to the archetype of its class or subclass.

Therefore, we trained one neural network per detector. The neural networks each take five input features (one per layer) and output a single value. The expected output is 0 for in-distribution samples and 1 for out-of-distribution samples. Here is how we created the datasets to train and test each network:

We merged the testing samples of the MNIST database with the training samples of MNIST that corresponded to ones and fives. We also added noise samples. We made sure the amount of in and out-of-distribution samples were equal. We then split the dataset in two to create a training and a testing set. For every sample of each set, we computed the average last layer group distance and matching class percentage of each layer. A 1 or a 0 was assigned to each sample depending on whether it was normal or not.

We trained each network on batches of size 100 for 1000 epochs with a learning rate of 0.001. The networks have the same number of neurons per hidden layer as the ones used throughout this thesis (described in "Our networks" in Chapter 2).

After the training, both networks reached an accuracy of 85%. We decided to test them on the test sets we just described. But, for each network, instead of taking the rounded output as the chosen class, we applied a threshold on the output. By varying the value of the threshold we could plot the ROC curves of each network. This gives us the means of comparison with the detectors of the previous sections (the comparison is done in Chapter 5). The curves are depicted in Figure 4.4.

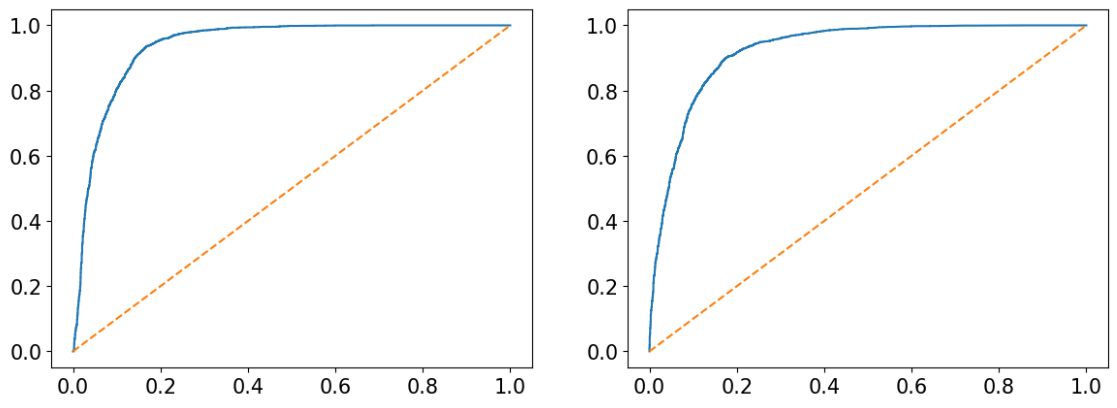


Figure 4.4: ROC curves of the neural-network-based detectors. The schema on the left represents the curve of the network based on distance and the schema on the right represents the curve of the network based on the percentage.

4.3.2.2 Analysis

We can already see that the performances of our detectors are much better than what we obtained in the previous sections. We will make general comparison between all our approaches as well as some general benchmarks in Chapter 5.

Chapter 5

General analysis

5.1 Summary

Independent Neuron Analysis In Chapter 3 we focused on the analysis of neurons independently. We looked at measures computed around the training of the network. This allowed us to detect what we called "weak" neurons. Neurons that are shut off by the network during the training process. Unfortunately, being able to differentiate such neurons from regular ones didn't bring us closer to the detection of samples prone to misclassification.

We also discovered evidences that two types of neurons emerge in the normal (non-weak) neurons:

1. Neurons that get activated by specific classes and
2. Neurons that separate samples based on more generic characteristics (they get activated when the characteristic is present/absent in the sample and do not otherwise). They therefore get activated more often.

Not all neurons are necessarily one of the two types. It is possible that some neurons belong somewhere in between.

Another behavior we discovered is that error-prone samples activate more neurons than normal ones when classified. Based on this discovery, we created one of the out-of-distribution detectors that we will analyze later in this chapter. We will call this detector the ***neuron count detector***.

Activation Pattern Analysis In Chapter 4 we looked at the neurons' activation patterns of the network. We had two intuitions in mind:

1. Samples that are close together should activate the neurons in similar ways. Creating similar activation patterns. Therefore, out-of-distribution samples and samples that do not resemble the other samples of its class/subclass (i.e. error prone samples) should have activation patterns that do not resemble any (or very few) of the training sample's activation patterns. We thus defined a distance measure between activation patterns. We then separated the training samples into clusters according to this distance measure. We could then measure the distance between the sample's activation pattern and the center of the cluster it was assigned to. We compared it to the distance that separated the training samples in that sample's cluster from the center. We then used that comparison to create a second out-of-distribution detector. Let us call that detector the ***cluster detector***.
2. The second intuition was that the network tends to automatically bring the activation patterns of samples that are supposedly from the same class together layer by layer. The distance between similar samples getting smaller as we progress in the layers. Out-of-distribution samples

also get pushed closer to other samples, but they should be less similar to these samples in the early layers than regular samples. This should allow us to detect them.

We measured the average distance between a sample s and the training samples that had the closest activation patterns in the last layer. We then recorded this average distance between the sample s and the same training samples in each layer. Our third detector is a neural network that takes these distances per layer as inputs. We will call this detector the ***distance network detector***.

Based on this second intuition, we created a fourth detector. It is also a neural network that takes one input per layer. But instead of taking the average distance between the analyzed sample and the close training samples as input, it takes, for each layer, the percentage of close training samples that share the same predicted class as the analyzed sample. We call this fourth detector the ***percentage network detector***.

5.2 Comparison

We want to compare the performances of the detectors described in Section 5.1 together. In order to have some benchmarks, we also compare them to the baseline method and a state-of-the-art technique.

The baseline method was introduced by (Hendrycks et al. 2016). This method simply consists of applying a threshold on the confidence outputted by the network when classifying a sample[1]. We will refer to this detector as the ***baseline detector***.

The state-of-the-art technique we are going to be comparing our detectors to is the ODIN method (Liang et al. 2017). Their technique detects out-of-distribution samples by analyzing the output of the last layer and slightly modifying the samples before feeding them to the network. They distort a sample, feed it to the network and apply the temperature scaling (described in the "Calibration" paragraph of Section 1.2) algorithm to their softmax output. A threshold on the final output allows them do distinguish out-of-distribution samples from in-distribution ones with great accuracy. The intuition behind this technique is that the input distortion boosts the confidence. Since the impact of the distortion is stronger on in-distribution samples than on out-of-distribution samples, it makes it even easier to differentiate them by looking at the confidence. They outperform the baseline technique from far.

The criteria on the basis of which we are going to compare the performances of the detectors are the following:

1. The ROC curves of each one of them (plotted in Figure 5.1).
2. The AUROC: the Area Under ROC (curve) (Figure 5.2).
3. The False Positive Rate when the True Positive Rate reaches 95% (Figure 5.3).

Unfortunately, we were not able to reproduce the ODIN method on our networks. Therefore, we use the data presented in their paper. This has some implications :

- Since they do not test their method on the detection of misclassified samples, these numbers will be left out of our comparison.
- The noise samples they try to detect were generated using a Gaussian distribution instead of uniform distribution like we did.
- To represent out-of-distribution samples they used the not-MNIST dataset. Their results are then probably slightly better than they would be on ones and fives samples since not-MNIST samples are not from the same initial distribution.

[1]The confidence we are referring to is described in the "Network output" paragraph of Chapter 2

- In their paper, they just show one ROC curve. It is the ROC curve that corresponds to the distinction of TinyImageNet (considered out-of-distribution here) from CIFAR-10 (considered the in-distribution, i.e. the data on which their network has been trained). Since this CIFAR-10 is a more complex dataset than MNIST, we consider this curve as a lower-bound to the performances of ODIN. We show this curve in Figure 5.1 as well.

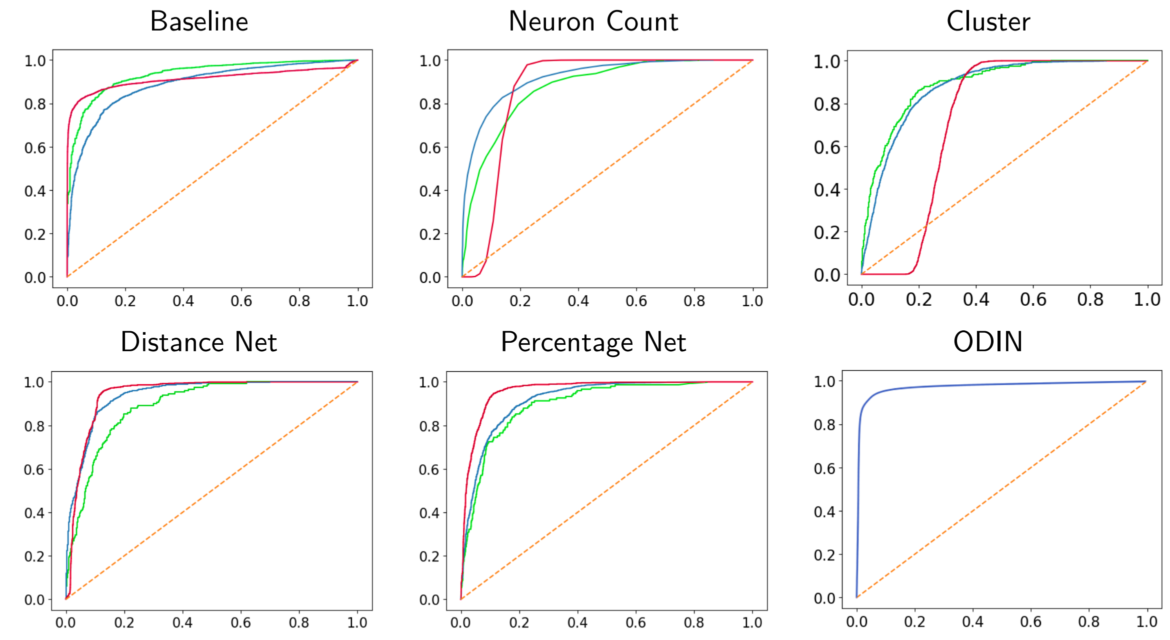


Figure 5.1: ROC curves of the different detectors. The blue lines represent the scores when tested on the MNIST dataset (where ones and fives were considered out-of-distribution). The red lines represent the score when tested on the dataset made by combining the normal MNIST samples (all the samples except the ones depicting ones and fives) with the noise samples. The green lines represent the score when applying the detectors on the small normal MNIST dataset (i.e. without the ones and fives) where the goal is to detect misclassified samples. The ODIN diagram comes from (Liang et al. 2017). It represents the ROC curve of their method when distinguishing TinyImageNet (out-of-distribution) from CIFAR-10 (in-distribution) samples.

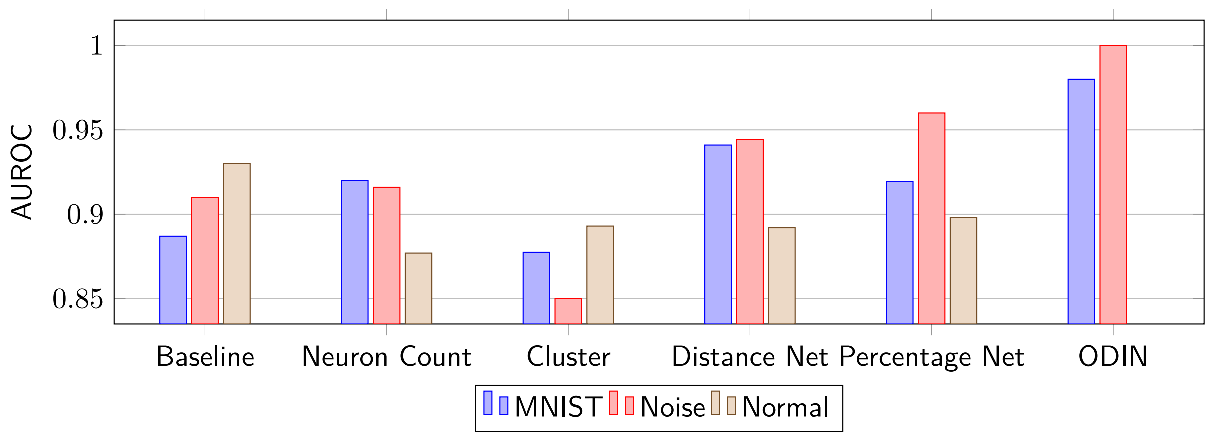


Figure 5.2: Diagram depicting the AUROC (area under ROC curve) of each detector. The blue columns represent the score when tested on the MNIST dataset (where ones and fives were considered out-of-distribution). The red columns represent the score when tested on the dataset made by combining the normal MNIST samples (all the samples except the ones depicting ones and fives) with the noise samples. The brown columns represent the score when applying the detectors on the small normal MNIST dataset (i.e. without the ones and fives) where the goal is to detect misclassified samples.

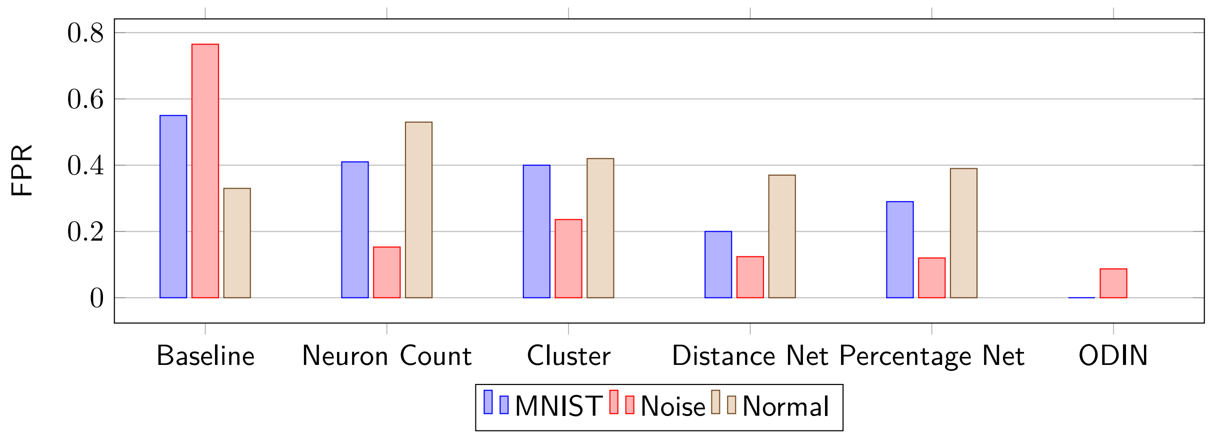


Figure 5.3: Diagram depicting the false positive rate when the true positive rate reaches 95% of each detector. Of course, the smaller the value, the better. The blue columns represent the score when tested on the MNIST dataset (where ones and fives were considered out-of-distribution). The red columns represent the score when tested on the dataset made by combining the normal MNIST samples (all the samples except the ones depicting ones and fives) with the noise samples. The brown columns represent the score when applying the detectors on the small normal MNIST dataset (i.e. without the ones and fives) where the goal is to detect misclassified samples.

We can see that our methods do not yield better results than the state-of-the-art. But it is not their purpose. The approaches explored in this thesis are there to present new leads to explore. They still need to be optimized through discoveries made by further researches. The fact that they all outperform the baseline (in detecting out-of-distribution samples) means that they have a good potential to lead somewhere. Some ideas of further research are given in the next section.

5.3 Scalability and future work

In this paper we discovered different interesting mechanisms. Some of which led to interpretations that could be further explored :

- In Section 3.2.2 we discovered the existence of weak neurons in the third layer. These neurons seem to be essentially useless in the network. Further research could try to understand how and why these neurons became weak. This could help to create guidelines in order to avoid such neurons when creating a network, making networks more efficient.

  One of the possible interpretation we provide in Section 3.2.2 on how they appear is that the weights of these neurons were initialed in such a way that they almost never got activated to start with. When they did get activated during the training, they misled the decision, causing the gradients to strongly diminish the values of the weights between that neuron and the next layer. Cutting them off. This would explain why their neuron accuracy is chaotic: they were not trained to participate in the decisions of the network and just randomly get activated once in a while.

  This interpretation could be tested by training our network sample by sample (batches of 1) and plotting the evolution of the influence of weak neurons and sane neurons during the training. We could then compare these plots to graphs representing the evolution of the gradient vector applied to these neuron at each iteration (again, one graph per neuron). If the proposed interpretation is correct, we should be able to see strong decreases in the influence of weak neurons each time a gradient is applied to its weights (i.e. each time the neuron is activated and has an impact on the output).

  Further research could also include resizing the network (for example making the third layer smaller) to observe how it impacts the appearance of weak neurons.

- In section 4.2.1.2 we discovered that out-of-distribution and misclassified samples activate more neurons than regular samples. We also discovered that, in some layers, the average activation count (term defined in Section 3.1) was higher for out-of-distribution samples.

  Experiments could be led to see if these results are reproducible on different network architectures and different datasets like CIFAR-10 and 100. If they are, digging a little more into the matter could shed light on the link between these two phenomenons and/or why they take place in general.

- Another question that deserves to be looked more into is: why do noise samples activate the neurons in a specific pattern that exactly correspond to the activation patterns of one or two classes (as observed in Section 3.3.2). The reason behind this might be that the first layer automatically activates in a few different patterns, automatically putting samples in one of the subclasses that the network created.

  One might want to look into this by varying the way noise samples are generated. Maybe starting from the samples of the class in which noise samples are classified, and progressively inverting more and more pixels until the image is just white noise. At each step, recording how the activation patterns behave.

Besides these interpretations to explore, other works could be done to further deepen the discoveries made in this thesis:

- Modifying the architecture of the network in multiple different ways. Changing the number of layers, increasing or decreasing the size of the layers, ... Seeing how our observations hold on these different architectures or how they are influenced could teach us a lot.

- Use the same technique as (Yosinski et al. 2014) to generate high-confidence meaningless images. Then testing how well our method allow to detect them.

- By recreating these experiments with new datasets such as CIFAR-10 or CIFAR-100, we could see if our discoveries hold on more complex samples.

- Finally, we could observe the impact of the use of techniques such as dropout (Srivastava et al. 2014), batch normalization (Ioffe et al. 2015) or even convolutional layers on our results.

Chapter 6

Conclusion

Neural Networks are complex systems. The way they learn to “understand” intricate datasets is still full of hidden mechanisms that are yet to be discovered.

In this thesis, we explored different intuitions on the behavior of neural networks regarding the classifications of out-of-distribution and misclassified samples.

We started by defining these intuitions. For each, we then proceeded to a series of experiences and analysis. The results of these allowed us to uncover interesting new mechanisms. We exploited these mechanisms to try to detect out-of-distribution and misclassified samples. We measured the performances of our detectors and compared them together as well as to a baseline and a state-of-the-art technique.

Finally, we suggested different ideas on further research based on our observations.

Our approaches do not surpass the state-of-the-art, but clearly outperform the baseline when it comes to detecting out-of-distribution samples, showing that further researches based on these mechanisms could lead to interesting discoveries and/or new applications.

A lot of research is done around the domain of artificial neural networks. Tons of papers, thesis, books, digging in the complex mechanisms that make them up. We might never be able to grasp this technology in its entirety, but, as for the rest of science, it feels good to see so many people working together to build a common knowledge, little by little, experiment by experiment.

Bibliography

Chollet, François et al. (2015). *Keras*. `https://keras.io`.

Frankle, Jonathan and Michael Carbin (2018). "The Lottery Ticket Hypothesis: Finding Small, Trainable Neural Networks". In: DOI: `arXiv:1803.03635v1`. arXiv: `1803.03635`. URL: `http://arxiv.org/abs/1803.03635`.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger (2017). "On Calibration of Modern Neural Networks". In: ISSN: 1938-7228. arXiv: `1706.04599`. URL: `http://arxiv.org/abs/1706.04599`.

Hendrycks, Dan and Kevin Gimpel (2016). "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: pp. 1–13. arXiv: `1610.02136`. URL: `http://arxiv.org/abs/1610.02136`.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: ISSN: 0717-6163. DOI: `10.1007/s13398-014-0173-7.2`. arXiv: `1502.03167`. URL: `http://arxiv.org/abs/1502.03167`.

Liang, Shiyu, Yixuan Li, and R. Srikant (2017). "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: 2017. arXiv: `1706.02690`. URL: `http://arxiv.org/abs/1706.02690`.

Martn Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

Shwartz-Ziv, Ravid and Naftali Tishby (2017). "Opening the Black Box of Deep Neural Networks via Information". In: pp. 1–19. arXiv: `1703.00810`. URL: `http://arxiv.org/abs/1703.00810`.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15, pp. 1929–1958. ISSN: 15337928. DOI: `10.1214/12-AOS1000`. arXiv: `1102.4807`.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus (2013). "Intriguing properties of neural networks". In: pp. 1–10. ISSN: 15499618. DOI: `10.1021/ct2009208`. arXiv: `1312.6199`. URL: `http://arxiv.org/abs/1312.6199`.

Yosinski, Jason, Jeff Clune, Anh Nguyen, Jason Yosinski, and Jeff Clune (2014). "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images". In: ISSN: 10636919. DOI: `10.1109/CVPR.2015.7298640`. arXiv: `1412.1897`. URL: `http://arxiv.org/abs/1412.1897`.

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals (2016). "Understanding deep learning requires rethinking generalization". In: ISSN: 10414347. DOI: `10.1109/TKDE.2015.2507132`. arXiv: `1611.03530`. URL: `http://arxiv.org/abs/1611.03530`.

Index