The effect of noise on an optical machine learning model

Sybren Huitink

June 12, 2019

Supervisors:

Prof. Dr. A.P. Mosk Dr. D. van Oosten

Natuur- en Sterrenkunde
Utrecht university

Abstract

To this day machine learning has been executed by computers. In this thesis, we research the possibility, and difficulties, of an experiment that performs machine learning tasks using light propagation. This would be more energy-efficient as well as faster than current machines. We study the effect that noise, which is unavoidable in the real setup, will have on the experiment. We find that small quantities of noise during training can decrease the validation accuracy of a model with a few percents. For a model that is trained without noise, the validation efficiency is not affected strongly by noise.

Contents

1 Introduction

Automatizing tasks is an important development in today's society. After the gain in popularity of computers that could be programmed to apply mathematical operations quickly grew more complex, this led to the question of whether these machines could think like humans. Artificial intelligence is the resulting scientific field, it focuses on producing software that can perform tasks like image and speech recognition, predicting stock markets and self-driving cars. Machine learning is the part of artificial intelligence in which programs learn from previous experience. The algorithm is trained by experience, using lots of test cases to get better at the tasks it should perform.

There are lots of factors that make the tasks mentioned above challenging to program. The human brain recognizes and prioritizes certain features more than others, ignoring information which is not of interest. Deep learning, a kind of machine learning, is a method that allows computers to do that same thing. With deep learning, the computer can make complex concepts out of more simple concepts [4, p.5]. These concepts are memorized in layers, a layer is a collection of nodes. In each node, a computation happens which result is passed on to the nodes in the next layer. As an example of these concepts, early layers might have filters that are able to detect edges or circles in the image and layers further in the network might be able to combine these more simple features to more complex attributes like faces or eyes. The layers make a mathematical transformation of the input to an output that ignores some properties and amplifies others.

The process of deep learning requires a lot of computational power and rapidly increases in time when more complex tasks are to be achieved. In this thesis, we will discuss experiments on a deep learning model that consists of a laser beam propagating through a vapor with a nonlinear optical response to light. An artificial neural network functioning on the principles of light propagation has numerous advantages. Since the information signal is a laser beam its speed is quicker than electronics can ever achieve. The setup is also very power efficient compared to an electronic neural network.

The setup contains spatial light modulators. these devices are capable of changing the intensity and phase of the incoming wavefront. The spatial light modulators can be trained just like a layer in an artificial neural network. Therefore an optical setup is able to act as a deep learning network.

In the experiment noise will be present on the input as well as on the SLM's. How sensitive the setup is to this noise is important for the applicability of the system in real-life applications. This raised the question: "How robust is the optical machine learning model to noise?"

In this thesis, we first explain the basics of deep learning. After, the experimental setup and its different components are discussed, followed by the conversion to the simulated model. Then the performed experiments will be discussed, firstly the response of the noise on the input images and afterward noise on the weights of the model. Finally, we discuss some possible ways to improve the model and its robustness to noise.

2 Theory

The optical deep learning setup this thesis is concerned about is called Vaporware. At the point of writing this thesis, the setup has not been build just yet. Therefore all measurements had to be simulated in python. This simulation is called the Vaporware model.

2.1 Neural networks

To understand the following chapters of this thesis we will explain the basics of artificial neural networks here. As the name suggests a neural network is composed of neurons connected together. An artificial neuron holds a number between zero and one. The neurons depicted by the circles in figure 1. The value of the neuron is one if the neuron is completely activated, the value is zero if the neuron is completely inactivated. In figure 1 It is shown that a typical artificial neural network consists of different layers that are connected.

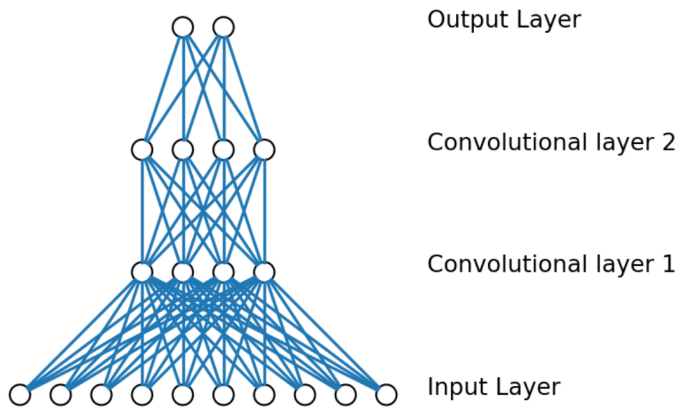

Figure 1: **Schematic architecture of convolutional neural network** This figure shows the nodes as circles. The weights are represented by the blue lines.

A layer determines how much each neuron in the next layer is activated. Since there are different layers the network recognizes more complicated patters at each new layer. All the weights of one previous layer are multiplied with their activation and then summed together. This sum is fed into the activation function of the neuron they are connected to. An activation function determines the activation of the following layer, there are multiple options for activation functions. The most simple activation function is the linear $y(x) = x$ which does not transform the input sum. More complex activation funcions bring the input value anywhere between their asymptotes, examples of this are the a Sigmoid or hyperbolic tangent activation functions, shown in figure 2. Most often activation functions with a symmetry around zero, like the hyperbolic tangent, are better since the convergence will be faster [8].

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

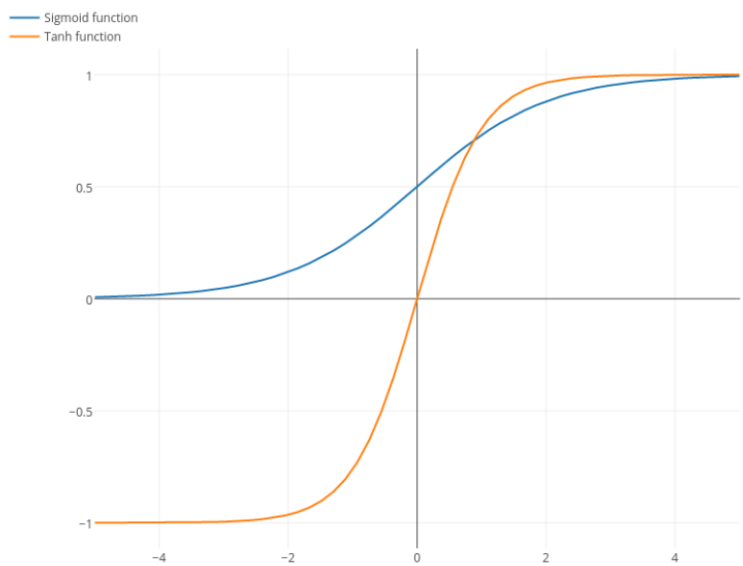

Figure 2: **A graphical representation of the Sigmoid and hyperbolic tangent activation functions [3].**

We might want a neuron to first exceed a certain value before it activates. This may be because the neuron is more or less important than other neurons. This value, called the bias, is first subtracted from the sum of the weights before the activation function is applied. Another activation function that is often used is called the Softmax function. The Softmax function as defined by Ian Goodfellow [4, p.185] is shown in equation 3.

$$\text{Softmax}(x)_i = \frac{e_i^x}{\sum_j e_j^x} \tag{3}$$

The network starts with random weights and biases. A widely used test case for machine learning is the MNIST database which consists of handwritten digits [7]. These images are the input of the artificial neural network. The network has to learn how to recognize the images correctly. the learning of the network is done by training, in training the network is given lots of input images, a prediction is made and a number is calculated which indicates how wrong the network's prediction was. This calculation is executed by the loss function. If

the network is bad at its task the loss function will result in a high number. When the network is trained well the result of the loss function will be low. Therefore minimizing the loss function results in increasing the performance of the network. In machine learning, there are multiple options for loss functions. The most basic example of a loss function is the mean squared error. The mean squared error computes the difference between the predicted and actual value and squares it. This process is iterated over the entire dataset which result a value.

To minimize the loss function the weights and biases of the network have to be changed. To determine how much these have to change we need an optimizer. The optimizer changes the weights and biases of the network by using the loss function to find the right changes to the weights. All optimizers work differently but the general idea is that the optimizer calculates what a change of each weight does to the loss function. This is done by calculating the gradient of the loss function with respect to a weight. The weights are then changed in a way that minimized the loss function, this is done for many small steps to hopefully reach the minimum. It might, unfortunately, happen that it reaches a local minimum. In a local minimum, the loss function has not been fully minimized since it is stuck in a smaller minimum.

Another important parameter is called the learning rate. The learning rate determines the amount by which the weights are changed. A small learning rate means the weights get changed with a small amount each step. Big learning rates can result in always overshooting the minimum while on the other hand, really small learning rates can easily get stuck in a local minimum.

To find the negative gradient of the cost function, the steepest direction to minimize the cost function, neural network optimizers use an algorithm called backpropagation. In back propagation starts by the values at the end of the network, in the case of handwritten digits this is will be ten numbers between zero and one, each indicating how certain the network is of its prediction. A number close to one means the network thinks it is that number.

Now all neurons in the output layer are connected to the previous layer. These weights and biases can be changed in a way that results in an output of 10 numbers which is closer to the perfect output. Going back through all the layers the algorithm finds the best possible way to change the weights and biases. The best possible way to change the weights and biases turns out to be the negative gradient of the cost function.

The purpose of training a network is to build a statistical model that can make predictions of new cases. This is different from memorizing the training data. A good neural network can perform just as well on new data as it would on the data it was trained on.

There is a distinction in machine learning between classic machine learning, where the programmer has to define the features himself that the program then recognizes and makes computations on. And deep learning, where the computer devices its own features it can detect and work with [4, p.10]. The Vaporware model is a deep neural network. A deep learning network has multiple layers of which at one with a nonlinear activation function.

2.2 Convolution layers

A class of neural networks that is in particular interest for us is the convolutional neural network, these networks are especially good for image recognition. The networks make use of the mathematical method called convolution where normal networks use matrix multiplication. To describe a convolutional neural network it is necessary to introduce convolutions first.

Convolution is a mathematical operation of two functions resulting in a new function describing how much the shape is modified by the other. In deep learning, the input is a tensor. What a neural network really does is modify this tensor every time it passes through a layer.

In a convolutional neural network, the tensor will be split into blocks over which the program iterates and calculates values. Now pooling finds the maximum value of a bigger slice of blocks. This slowly extracts data from the image, first, it will find edges or lines, later it will find the smaller details.

Convolution layers can detect patterns in an image, this can be anything simple from edges or corners to complex characteristics like eyes or leaves. Figure 1 shows a possible convolutional neural network. The input could be the pixel brightness and pixel locations of an image, to keep the image clear the output is only two options. This network could, for example, make a distinction between the numbers five and six or cats and dogs. A convolutional neural network can also have other types of layers in between. The convolutional layer receives input and consequently transforms it to give input to the next layer. However, the transformation in convolution layers is different from other types of layers. In a convolution layer, this transformation of the input data is a convolution operator. A regular convolution layer has a variable amount of filters associated with them. Each filter can detect different attributes in an image. A filter is a small matrix with values, as an example see figure 3. The convolution operator uses this matrix to take the inner product of it with the image, then slides one step and takes the inner product with the block after that, after doing this with the entire image an output tensor is created that has enlarged certain features. These features are used by the next layer to detect complex structures. A very important restriction in the Vaporware model is that there is only one possible filter. This filter is determined by light diffraction through the vapor. Since this process obeys the laws of physics there is no simple way to change this filter.

0.34	0.47	0.65
0.76	0.86	0.15
0.21	0.06	0.11

Figure 3: **An example of a convolution window that filters for a certain feature in the image**

An example of a computational advantage over normal matrix multiplication. Firstly since the output after the convolution operator is smaller than

the input. In image recognition, this means that the normally huge number of pixels, which equals to lots of parameters, is reduced by the convolution.

2.3 The advantage of noise

It is common in machine learning to add augmented data to the training set. Augmented data is transformed real data that may be low in quantity. Augmentation can be done by making transformations to the original data set by adding for example noise to it. This especially works for object recognition tasks [4, p.240]. In images for object recognition, there are lots of factors that can be easily simulated. An example of augmented data is rotating the image slightly, this may improve the robustness and increase validation accuracy.

The data structure used to build a network is called a model. The model we use in this thesis is called a Sequential model which is a linear stack of layers [2]. Often in machine learning models, a dropout layer is added, this corresponds in some way to the addition of noise to the weights of the model. This decreases the occurrence of unusually big values in the weights of the model. Adding noise to the weights mostly occurs in recurrent neural networks. "This can be interpreted as a stochastic implementation of Bayesian inference over the weights. The Bayesian treatment of learning would consider the model weights to be uncertain and representable through a probability distribution that reflects this uncertainty. Adding noise to the weights is a practical, stochastic way to reflect this uncertainty." [4, p.242]

The addition of noise to the weights of the neural network can also be thought of as a form of regularization which increases the stability of the network to reduce overfitting.

$$Cost = E * [(\hat{y}(x) - y)^2] \tag{4}$$

To explain this further we start with equation 4. In this equation $\hat{y}(x)$ is the prediction made by the model, y is the actual value. The data set consists of a list $[(x1, y1), (x2, y2), ...]$ where x is the image and y is the number supposed to be written. E can be any function of x and y, depending on the type of regression. Now introducing a random perturbation ϵ_W to the weights and minimizing the cost function drives the weight parameters to values where small perturbations have little effect on its workings. This pushes the neural network to a state where it is insensitive to variations of moderate size. This means the model has not found just a minimum but a minimum that is flatter than the minimum it would have found without any noise [4, p.242]. Such a minimum is more robust to noise which results in better prediction power for real-life applications.

2.4 Regularization

Figure 4 shows two different ways of fitting data. The data points are the function $y = 2x$ with noise added to the data. The right part of figure 4 shows a very complicated function that does reach every data point but does not describe

the general progression of the data well. This is an overfit to the data. The left part of figure 4 is a smooth function that does not reach every data point but describes the general evolution of the data way better.

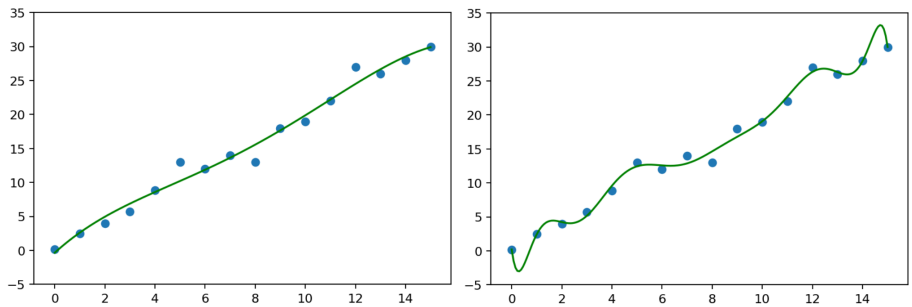

Figure 4: **An example of overfitting.** The data points in blue are fitted in two ways, the right figure has a more complicated function fitted to it.

In regularization, an extra term is added to the cost function which reduces the chance of overfitting. This extra term punishes bumpy features in the error function. It simplifies the error function as a transformation of the right part of figure 4 to the left part of figure 4. The error function therefore changed from E to the regularized error function $\widetilde{E}$.

$$\widetilde{E} = E + \nu\Omega \tag{5}$$

ν is the regularization parameter that controls how much ω affects the error function. During training equation 5 is minimized. A function that fits very well to the training dataset results in a small E while a function that is very smooth has a small $\nu\Omega$ as a result. There are different definitions of Ω. A simple example of a regularization method is the weight decay, which penalizes large weights.

$$\Omega = \frac{1}{2}\sum_i w_i^2 \tag{6}$$

2.5 Theory on noise

Training with noise has been shown to be able to improve the generalization of the network [1]. As we will show in this section training with noise is closely related to regularization. In the modern artificial intelligence field training with noise is mostly forgotten while regularization is still used often to increase performance.

Training a neural network is achieved by minimizing a cost function, an example of a cost function is the sum of squared error function. This function is a sum over the entire training set and its output, the equation can be seen in

equation 7. In this equation, y_k is the output of training unit k, x^n the input and the weight w is a vector. N is the total amount of training units and c the number of outputs, in the case of number recognition c would be ten. t_k^n is the correct output, in the MNIST database this is equivalent to the label, the number that the image is supposed to represent.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{y_k\left(\mathbf{x}^n;\mathbf{w}\right) - t_k^n\right\}^2 \tag{7}$$

We can set the amount of training data to infinity in equation 7 so integral can be applied, this will replace the sums with integrals resulting in equation 8. This is the error function before noise is added. $p(t_k, x)$ is the joint distribution of t_k and x. [1, p.202]

$$E = \frac{1}{2}\sum_{k}\iint\left\{y_k(\mathrm{x};\mathrm{w}) - t_k\right\}^2 p\left(t_k,\mathrm{x}\right)dt_k dx. \tag{8}$$

The joint distribution $p(tk, x)$ is changed to the product of the unconditional density function for the input data $p(x)$, and the target data density conditional on the input vector $p(tk|x)$ [1, p.202]. The input noise can be described by a vector ξ that is added to the input which is also a vector. The noise ξ is randomly distributed by probability distribution $p(\xi)$. Without noise the error function is equation 8, now with noise added the error function becomes equation 9

$$\widetilde{E} = \frac{1}{2}\sum_{k}\iiint\left\{y_k(\mathbf{x}+\xi) - t_k\right\}^2 p\left(t_k|\mathbf{x}\right)p(\mathbf{x})\widetilde{p}(\xi)d\mathbf{x}dt_k d\xi \tag{9}$$

If the amount of noise is small equation 9 can be simplified using a Taylor expansion on $y_k(x+\xi)$. Using a noise distribution with a mean of zero becomes:

$$y_k(\mathrm{x}+\xi) = y_k(\mathrm{x}) + \sum_i \xi_i \left.\frac{\partial y_k}{\partial x_i}\right|_{\xi=0} \tag{10}$$

Since the noise distribution is uncorrelated and has a mean of zero it has the following properties:

$$\int \xi_i\tilde{p}(\xi)d\xi = 0 \qquad \int \xi_i\xi_j\tilde{p}(\boldsymbol{\xi})d\boldsymbol{\xi} = \nu\delta_{ij} \tag{11}$$

Where ν is the variance of the noise distribution. Now using these results in equation 9 the result is

$$\widetilde{E} = E + \nu\Omega \tag{12}$$

The second order derivative is negligible resulting in Ω given by

$$\Omega = \frac{1}{2}\sum_{k}\sum_{i}\int\left(\frac{\partial y_k}{\partial x_i}\right)^2 p(\mathbf{x})d\mathbf{x}, \tag{13}$$

$\nu\Omega$ is the additional term to the cost function due to noise. Now minimizing equation 12 with respect to $y(x)$ results in the a good mapping function. Hereby it is shown that training with input noise is just another form of regularization.

2.6 Light Propagation

In the setup considered in this thesis, light is propagated through a vapor. This vapor can be exited by the light, because of this the atoms will have two possible states. This possibility creates a non linearity in the system. The initial electric field does not stay the same during this propagation, therefore it is necessary to use a formula describing change of the electric field $E(x, y, 0)$ in the beginning to $E(x, y, z)$ at a distant point z. The wavefront propagates in the z direction. $\mathbf{r}$ is the vector indicating the location (x, y, z). The laser produces light with a single wavelength, in complex notation the electric field is then given by [5]:

$$E(\mathbf{r}, t) = \mathrm{Re}\left[E(\mathbf{r})e^{-i\omega t}\right] \tag{14}$$

Writing this formula in decomposed frequency's α and β corresponding to the frequency component in the x and y direction. The wavefront is made up of a combination of plane waves. In Fourier space propagation is a multiplication which translates to a convolution in real space. The Fourier expansion of equation 14 results in the equation

$$E(\mathbf{r}) = \frac{1}{4\pi^2}\int E(\alpha, \beta, z)e^{i(\alpha x+\beta y)}\mathrm{d}\alpha\mathrm{d}\beta \tag{15}$$

We define k_0 as ω/c where ω is the frequency and c is the light speed. Since the light is moving along one axis we use the paraxial approximation. This approximation let's us ignore small angle deviations the light has with respect to this axis. The Helmholtz equation for vacuum then states:

$$\nabla^2 E(\mathbf{r}) + k_0^2 E(\mathbf{r}) = 0 \tag{16}$$

We define $\gamma(\alpha, \beta) = \sqrt{k_0^2 - \alpha^2 - \beta^2}$. γ can be a real or complex, if $\alpha^2 - \beta^2$ is bigger than k_0^2 the solution is damped. If γ is real the wavefront is propagating. If equation 15 is inserted in the Helmholtz equation and the result integrated a useful equation for the electric field is the result. The electric field at each point in space is then given by

$$E(\mathbf{r}) = \frac{1}{4\pi^2}\int E(\alpha, \beta, z = 0)e^{i(\alpha x+\beta y+\gamma z)}d\alpha\mathrm{d}\beta \tag{17}$$

This is the important optics formula in the Vaporware model since it is used to calculate the electric field at each SLM. Thus, the propagation of the wavefront is calculated in Fourier space.

3 Set-up

Figure 5 shows a schematic representation of the setup. A laser displaying the input image with monochromatic light falls on the SLM, the voltages produced in the SLM are the equivalent of the weights of the network. The input image has a 28x28 resolution, the SLM's can also have this resolution. Since most SLM's

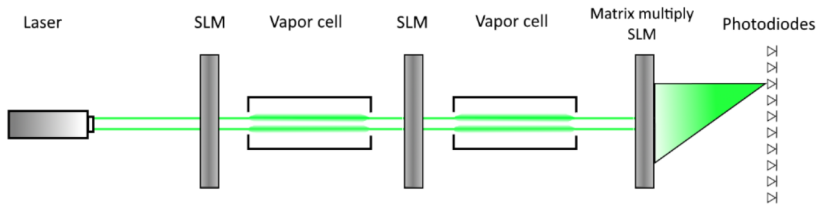

Figure 5: **Schematic figure of the Vaporware setup**

nowadays have more pixels this could be achieved by grouping these pixels together into a superpixel. In the Vaporware model, the SLM's are modeled by special layers that are linear, every pixel rotates the phase of the light from 0 to 2π. The amount of rotation is trainable. The light is then propagated through the vapor, the propagation is modeled as a convolution layer. The resulting electric field has a phase and intensity that, in the Vaporware setup, is passed through a vapor cell. This is modeled in the Vaporware model by multiple layers. The process is split into multiple propagation layers and saturation layers. The propagation layers are modeled by convolution layers that are not trainable. The saturation layers add extra phase shift to the propagated light. These layers are trainable. After some iterations of the steps above the light will reach the last spatial light modulator which has a higher resolution. This spatial light modulator also changes the intensity of the light and focuses all the incoming light on the photodiode representing the right number. The field on the photodiodes has an intensity that is calculated using

$$I(\mathbf{r}) = E(\mathbf{r})E^*(\mathbf{r}) \tag{18}$$

This light intensity is the result input for the ten photodiodes. The voltage's on the photodiodes are the output of the Vaporware setup. The diode with the highest voltage stands for the input number.

This setup shown in figure 5 has two SLM's and two vapor cells, the number of SLM's could be expanded to many more to make a more complex network since it would have more layers. This could be especially useful for more complex tasks than number recognition, an example of this would be recognizing people's faces. In the Vaporware model, there are two propagation and two saturation layers which is a significant simplification that does save a lot of computations. The saturation of the vapor acts as the activation function, activating some areas of the image more than others. The Vaporware model has some additional layers to improve computational efficiency. Firstly, a dropout layer added after the last convolution layer (light propagation). The dropout layer randomly disables some neurons each iteration resulting in a more robust network that trains faster. Right before the last step, a multiplication layer scales the incoming data by multiplying every value with a constant. The intensity is important for the model. The very last step is an activation function, Softmax (3), that transforms every value to a number between zero and one.

3.1 MNIST database

The database we used in this thesis for every experiment is called the MNIST database. The MNIST database consists of handwritten digits all with a resolution of 28x28 pixels. Half of the digits are written by American census bureau employees, the other half is written by high school students [7]. The database consists of 60000 images intended for training and 10000 images that are intended for testing. Testing is also called validating. Every image also has a label connected to it that indicates the number it is supposed to be. Nowadays the MNIST database is the most standard dataset to research machine learning. Figure 6 shows some example images from the database.

Figure 6: **Examples from the MNIST database**

3.2 SLM

A clever device that is needed to make the Vaporware setup work is the spatial light modulator. Throughout the setup the get wavefront of the starting laser light has to be changed. A spatial light modulator is able to manipulate the input light wavefront in multiple ways. In the past photographic emulsions were being used for modulating light beams, this had limitations since it needs a significant time to respond to the light. For this reason scientists experimented to be able to to enhance the speed of converting an optical or electrical signal to light beams. The resulting device of these experiments is the spatial light modulator or SLM. Optical SLM devices can have multiple functions. Examples are that an SLM can amplify its input data but also change the wavelength of the incoming light.

There are multiple types of SLM's that can be used. In this chapter two of these options will be discussed: Liquid crystal SLM's and deformable mirror SLM's.

3.2.1 Liquid crystal SLM

A liquid crystal display is a set of liquid crystal cells, these displays are used frequently in devices like mobile phones or television screens. Each cell is controlled by the amount of voltage applied to it. The voltage controls the intensity of the light reflected or transmitted from the cell. Liquid crystals are interesting because they possess the qualities of a liquid as well as a solid. These molecules are rod-shaped and are able to stack upon each other. An SLM works in the

same way as a display and is able to change the phase delay by applying a voltage to it. Applying an electric field across the cell induces an electric dipole in every liquid crystal molecule which changes their natural orientation. Changing this orientation can be completed in times under a microsecond [5, p.191].

3.2.2 Deformable mirrors

Another possible spatial light modulator uses lots of little mirrors, the orientation of these mirrors can be changed by applying a voltage.

3.3 Vapor

The setup would not be complete without the vapor the light propagates through. The laser is sent through vapor cells which contain a vapor consisting of a single element. In the Vaporware setup, the vapor is modeled as having one exited and one unexcited state. This is not completely realistic since gases generally have more than two possible states. To closely resemble the simulations the chosen gas should preferably have a ground state and one reasonably stable exited state. The frequency, or energy, of the laser light should be changed to equal the energy difference between these two states. The vapor has to be partially in the exited, this can be achieved by tuning the intensity of the laser. The possibility of an atom in the vapor to switch between states after interacting with the laser light makes the model nonlinear. This non-linearity is necessary for machine learning to achieve higher validation accuracy then what linear networks can achieve.

If a layer is activated by a linear function the activation of the next layer is just a linear function of the input of the previous layer. Since this is just a more complex linear function these two layers could be replaced by one larger layer. Without this non-linearity, the process is equal to just a matrix multiplication which is in principle a single layer.

4 Results

For the following results two models are trained, the Vaporware model and a Convolutional neural network [2] from the Keras website. The Vaporware model is trained with the Adam optimizer. The Adam algorithm is an effective alternative to the older stochastic gradient descent algorithm to update the weights of the network after training. Important about the Adam optimizer is that the learning rate is not constant but changes during the training process [6]. The Keras convolutional neural network is trained with the Adadelta optimizer, an optimizer that changes the learning rates for different weights.

4.1 Vaporware model visualized

To gain a better understanding of the transformations the model applies to the input wavefront we visualized this process step by step. Figure 7 shows a graphical representation of how the image is propagated through the setup. The phase is indicated by the color, the other parameter is brightness which is less evident in Figure 7. The brightness will not be altered by the first two SLM's, the phase will be changed.

One important difference between the model and the real setup is that in the real setup the propagation and saturation would happen with an infinite amount of steps. Whereas in this example this process happens in discrete steps.

To produce the images in figure 7 the color has to be changed from the HLS color model to the RGB color model first. The HLS model is a description of colours more natural to how humans perceive colours. Each pixel in the model is characterized by the Hue, light and saturation. Hue represents the phase of the light. Light is equivalent to the amplitude of the light and saturation has a constant value of 1/2. The saturation implies the amount of black or white mixed with the colour. A saturation value of 0 has a completely black colour as a result, a value of 1 means it is completely white. The RGB model characterizes the amount of red, green and blue of a pixel.

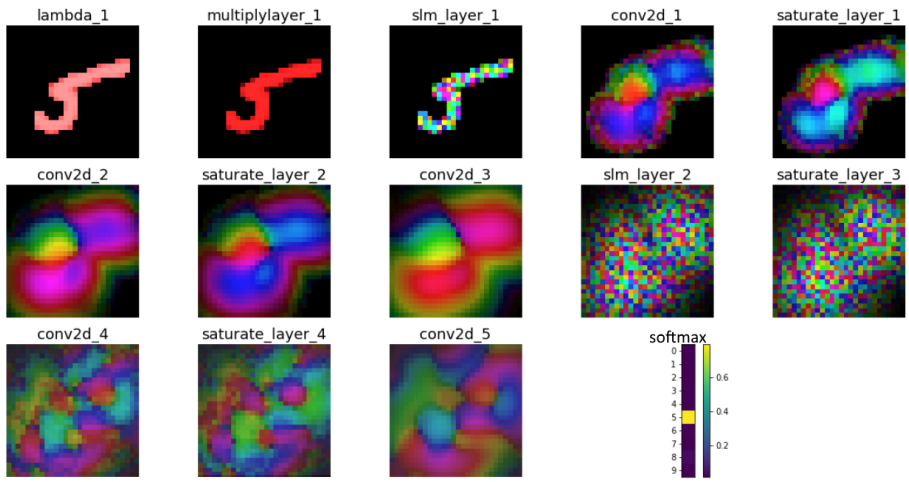

Figure 7: **The propagation of the wavefront of the input digit through the setup [12].**

In figure 7 the shape of the input digit is entered and its intensity is increased. The first SLM adds a complex phase to this image. This wavefront is propagated through the vapor which saturates it. The saturation works in a way as an activation function that activates certain areas more than others. Before the second SLM, the shape of the five is nearly unrecognizable. After the second spatial light modulator, the wavefront becomes more complex and unrecognizable. After the last convolution layer, matrix multiplication is used, This is followed by flattening the two-dimensional data to a one-dimensional array which has weights connected to the actual activation function. This results in the output shown in the last picture of figure 7. These last steps are left out of figure 7 since it is inconvenient to visualize.

The last layer, Softmax, shows the output layer of the setup. It is the resulting array of numbers after the Softmax function is applied. This indicates which digit the model predicts and with what certainty. In Figure 7 the image is correctly classified as five, the number associated with the five is 0.795 which is much higher than all other numbers as is shown in equation 4.1. This shows the activation of the output neurons, the sum of these ten values always has to add up to equal one.

$$[0.01868886, 0.01946426, 0.01822988, 0.02118771, 0.02244342,$$
$$0.79521817, 0.02382111, 0.01758016, 0.04066996, 0.02271442] \quad (19)$$

4.2 Input noise

To research the decreased validation accuracy as a result of distortions on the input images the data set had to be manipulated. Therefore we added Gaussian noise to the data set since that is the noise most often encountered in nature. It can be seen in figure 8 how much images get affected with certain amounts of noise.

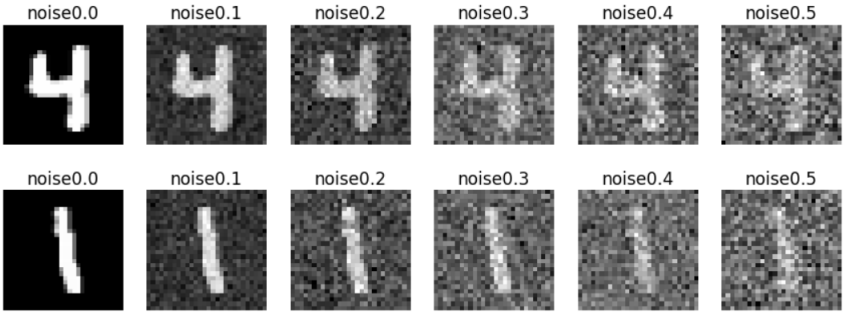

Figure 8: **Examples of Gaussian noise on input images.** This figure shows the effect of noise on images of the MNIST database. The number above every image indicates the standard deviation of the noise.

Before the noise is applied the data set is first normalized. This means that every pixel has a value between zero and one. A pixel with value one means it is white and a value of zero means the pixel is black. The values in between are all variations of grey. Now noise with different standard deviations can be applied to the data set. The addition of a noise distribution means that every pixel has a chance to become brighter or darker. If a pixel with value one gets brighter this is the new maximum value for brightest, completely white, possible pixel. Figure 8 shows that the numbers are more unrecognizable if more noise is added.

Figure 9 is the resulting validation accuracy after noisy input images with different amounts of noise added to the input data of the models. To be able to compare the response of the vaporware model to input noise we did the same experiment on a Keras convolutional neural network that is especially good for predictions of the MNIST database [2]. The models were trained on the dataset without noise, it turns out that in this case, the validation accuracy degrades faster than when it is also trained with noisy images. Evidence of this is shown in the appendix. Some numbers are more susceptible to false classification than others. An example is the number eight since it can easily transform into other numbers like the nine or zero. The numbers zero and one are the most robust to noise and have little chance of being transformed to other numbers. [11].

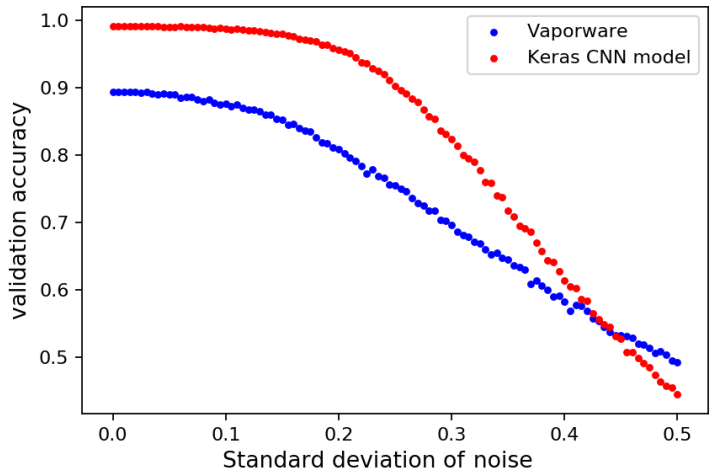

Figure 9: **Validation accuracy versus the standard deviation of the noise added to the MNIST database. Both the Vaporware and the Keras[2] model are analyzed.** A different noise pattern is added to every image of the database. Both models were trained for 40 epochs.

Figure 9 shows the validation accuracy versus the standard deviation of the noise added to the input dataset. It can be seen that the Keras CNN model is barely affected by little noise but degrades faster with higher levels of noise. The degrading data points show that if there is noise on the input images the models have trouble identifying the number correctly. With little noise, the validation accuracy of the models will not be affected much by input noise. Higher noise, with a standard deviation of 0.2 or higher, linearly degrades the validation accuracy of the models. The Keras CNN model is more robust to lower noise levels but degrades faster with higher input noise. In conclusion, if little noise is present on the input it will not affect the model much. From higher standard deviations the validation accuracy has a steeper decrease. If the noise is increased further it will eventually become stationary at a validation accuracy of approximately ten percent. Since there are ten numbers this is the probability of a completely random guess being the right number.

4.3 Noise on Weights

We now want to investigate the performance of the Vaporware model when we add noise not to the input data but on the weights of some layers of the model. The layers we add noise to are the SLM layers of the Vaporware model. There are three different methods of how we added noise to the weights of the model. First, we can add noise after the model is already trained. The second possibility is to add noise after every epoch during training and to add

noise during validation so after the model is trained. This resembles the most realistic situation. The third is by adding noise only after every epoch and without adding noise while validating. The last is the most unrealistic but the results are interesting to compare the other methods with. Since the SLM's in the setup change the phase of the incoming light the physical representation of this noise is a phase noise, this means the phase of the light will be changed more or less because of this noise. The phase noise, in radians, is calculated by multiplying the standard deviation by 20π.

Figure 10 is a raw data visualization of the effect of noise on the weights of the model. In this figure, the noise is only added to the spatial light modulator layers after training the model is trained for 40 epochs. This is before the model validates a data set. For each phase shift, twenty data points are gathered with a new random noise added to the weights every time. The Vaporware model also simulates the propagation and saturation of the light as layers. In our analysis, these layers are left unaffected by noise.

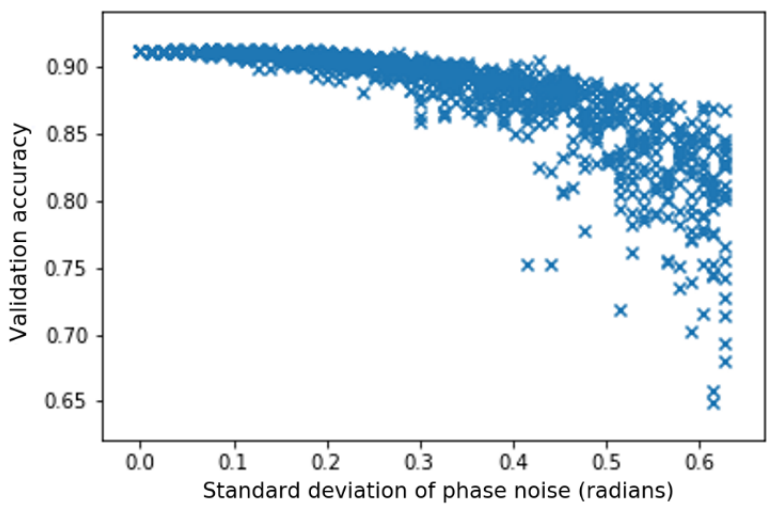

Figure 10: **Validation accuracy of the Vaporware model versus noise added to the spatial light modulators after training.** The standard deviation ranges from 0 to 0.65 radians, this is split in 25 steps. In this plot there are 20 data points for each phase noise, resulting in a total of 500 data points. The model was trained for 40 epochs

Figure 10 shows how opposed to noise on input data, the spread in the validation accuracy becomes larger when the noise levels on the weights is higher. This is because the phase noise, which is different for each datapoint, sometimes has more effect on the accuracy. Therefore there is a spread in data points. To visualize this data in a more meaningful way the mean of twenty data points

and its standard deviation can be shown, the result of this is shown in figure 11. The following figures are the results for a training time of 20 epochs.

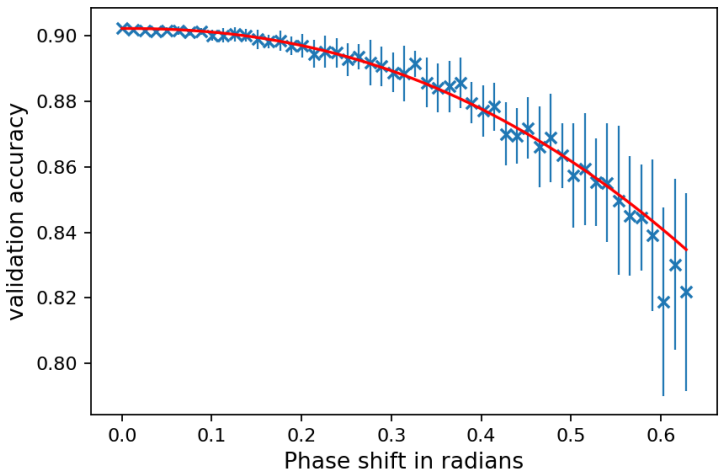

Figure 11: **The effect of noise on the first two SLM layers of the Vaporware model after the model was trained.** The red line is the fit of the data points. The vertical blue bars is the standard deviation of the datapoint. This result is obtained after the model was trained for 20 epochs.

In figure 11, noise is only added to the weights after the model was trained. The red line is the fit of the data points, the fit took into account the standard deviation of the data points. The error bars indicate the standard deviation of the data points at that particular phase noise. The code used for fitting the data points can be found in the appendix. These data in figure 11 fits the curve

$$y = 0.91 + -0.19 * x^{2.24}, \tag{20}$$

In the fit it was assumed that the validation accuracy degrades by a power law. The same experiment can be performed on the Keras model [2].

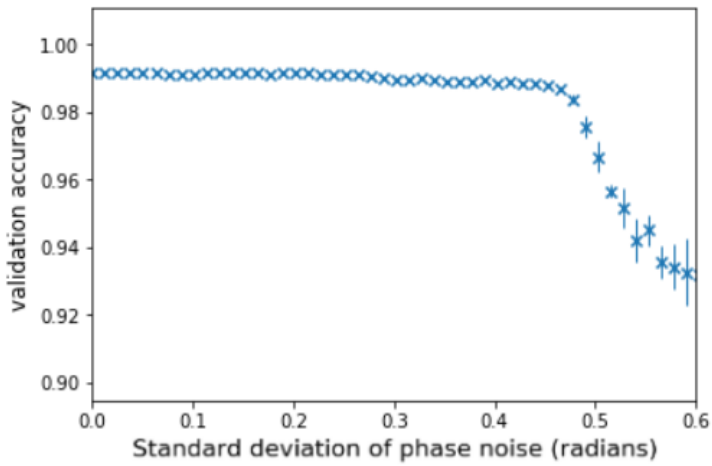

Figure 12: **Validation accuracy versus the phase noise on the two convolution layers of the Keras model** The noise was added to the weights after the model was trained. The vertical blue bars show the standard deviation of the validation accuracy at each phase noise.

Figure 12 shows that the Keras model is much more robust to noise on the weights after training than the Vaporware model. The network performs almost at a validation accuracy equal to the validation accuracy without noise until a phase noise of 0.45 is added to the weights. From that point, the validation accuracy quickly degrades. More research is needed to figure out what causes this difference in robustness. The Keras model is a normal neural network and does not have any optical meaning. Therefore "standard deviation of phase noise" is not the correct name for the axis. However, to be able to compare it to the Vaporware model the standard deviation of noise is multiplied by 20π which results in the equivalent phase noise.

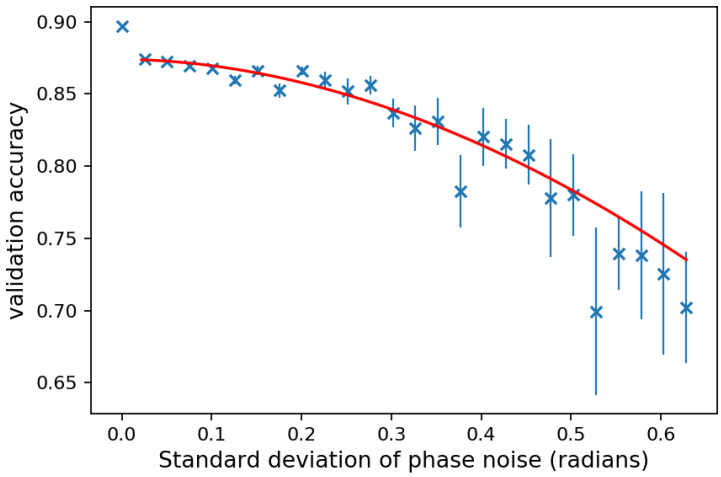

Figure 13: **Validation accuracy versus noise on the first two SLM layers of the Vaporware model during and after training the model.** The red curve is the fit of the data points, the standard deviation of the validation accuracy is shown as the blue error bars. The model was trained for 20 epochs.

In figure 13, noise is added to the weights after the model was trained and also after every epoch. These data points fit the curve shown in equation **??**

$$y = 0.87 + -0.33 * x^{1.89}. \tag{21}$$

The first data point in figure 13 has not been taken into account of the fit because of the steep drop. The effect of phase noise between the first and second data point has to be investigated in more detail.

A comparison of figures 13 and 11 shows that when the Vaporware model is trained with weight noise it performs worse than when noise is only added to the weights before the model validates data. This is not in line with most literature describing noise on neural networks [10]. Figure 14 compares equations 20 and 21.

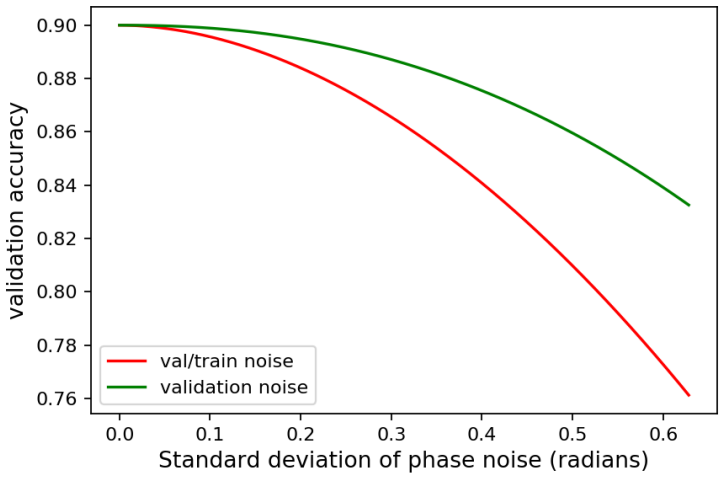

Figure 14: **Validation accuracy versus phase noise, one model trained and validated with noise, the other is only subject to noise during validation.** The red curve shows the fit of the validation accuracy after noise is added during training and during validation. The green curve shows the resulting validation accuracy when noise is only added to the SLM's during the validation of the data. Both models were trained for 20 epochs.

In figure 14 the starting value is set to the same value, 0.90, to be able to compare the difference of the slopes of the curves better. Figure 14 clearly shows that when noise is applied on both the training and validating phase the validation accuracy decreases more rapidly.

The last possible way to add noise to the weights is to only add noise during the training. The result of this experiment is shown in figure 15.

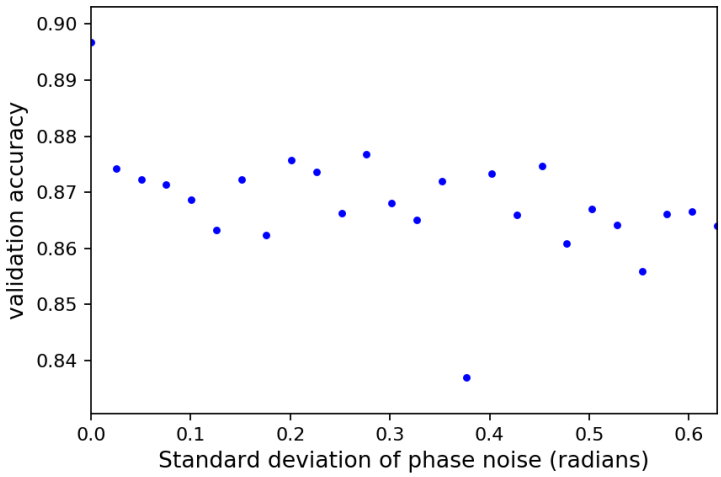

Figure 15: **Validation accuracy of the Vaporware model versus noise added during the training. 26 data points are shown.** The model was trained for 20 epochs.

figure 15 shows the response of the model if the noise is added only while training the model, no noise is added during the validation phase. In figure 15 it is interesting to note that there is a steep drop in validation accuracy from the first data point without noise to the second data point with a small amount of noise. Except for the case with zero phase noise, the accuracy seems to decrease slightly when more noise is present. This decrease is very small. It would be interesting to examine the region between the first and second data points.

Another question that arose is if the amount of epochs the model is trained with has an effect on the models' robustness to noise.

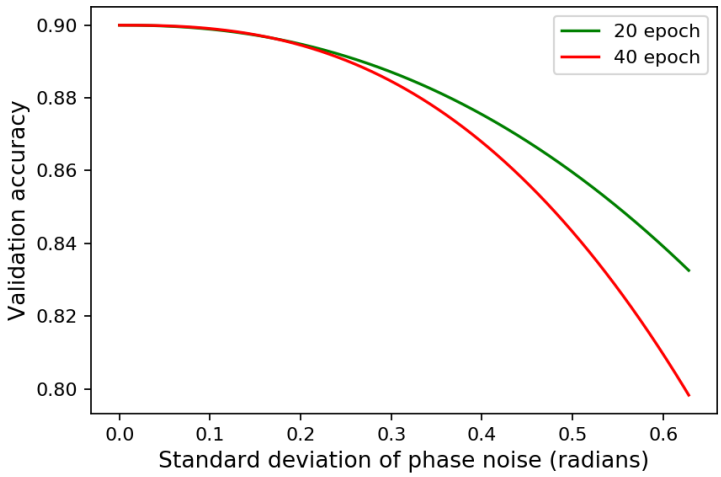

Figure 16: **The validation accuracy versus phase noise after 40 epochs and 20 epochs.** Noise is added to the weights after the Vaporware model has been trained. Only the fitted function of the data is shown.

Figure 16 shows the surprising result that when the Vaporware model is trained for 40 epochs it is actually less robust to noise added on the weights after training than when it is trained for a 20 epochs. The noise is added to the weights only after the model is trained. This is a possible sign of overfitting which has the consequence that more fitting makes the model less able to perform its task well. Overfitting can be recognized by a growing gap between training error and test error. [4, p.111].

The weights of a layer are composed of two variables. First there are the weights that influence how much the input signal is strengthened or weakened. Second, there is the bias that determines the value a signal has to overcome to be passed on. In figure all above figures where noise is added to the weights noise is also added to the bias.

5 Conclusion

In this thesis, we investigated the sensitivity of the Vaporware model to noise. In obtaining our results, we assumed the simulated model behaves the same as the actual setup would. We found that the Vaporware model is affected little by noise on the input images, even with an unlikely high noise on the input images the Vaporware models' validation accuracy is only slightly decreased. For noise on the spatial light modulators: if noise is added during training the models' validation accuracy reduces more than when the network is trained without any noise. If a small amount of noise is present while the model is trained the validation accuracy decreases a few percents compared to when the model is trained without any noise. The Vaporware model is sensitive to overfitting which should be taken into account in further experimentation.

6 Discussion

The results gained by testing different noises on the models can be used to select which spatial light modulators should be used in this setup. The spatial light modulators do not need a high resolution, high robustness seems to be more important. This holds for experiments on the MNIST database. The resolution may be more important if the network has to do more complicated tasks. The steep drop in validation accuracy in figures 13 and 15 has to be investigated more thoroughly since this could mean a large drop in accuracy even with small amounts of noise. This would decrease the usefulness of a physical machine learning model.

The code we used in this thesis to add noise while training the model may have a defect that has to be researched more thoroughly. The model is saved after training for one epoch, the noise is then added and it is trained for another single epoch. This procedure is repeated for the number of epochs we wanted to train the model. Since the model starts training again each time after one epoch the optimizer, Adam in this case, may not work the same way it would when it is applied for multiple epochs without intermission. The Adam optimizer has momentum which is remembered during training, this may be disrupted by the method we used in this thesis. The same experiments should be performed with other optimizers to be able to conclude this. Another note for future investigation is that, in the experiments of noise on the weights, the same amount of noise was added to the weights themselves as well as the bias. In a real setup, this may behave differently.

Whether an optical machine learning model will surpass the traditional computational neural networks is still questionable. The idea does sound implausible at first but after writing this thesis I am convinced that it has a good chance, the advantages of optics are amazing. Low in energy consumption and unbeatable calculation speed are two very attractive attributes. My prediction is that, after it is proven this setup works, the Vaporware model will grow very quickly in popularity as a research topic. The first company that implements an improved

version of Vaporware in for example cars or supercomputers may take the lead in a new period of computer technology.

The steep drop seen in figure 13 is a problem since the validation accuracy seems to drop a few percents even with little noise. There may be a solution to this problem, from the resulting models it can be seen that the steep drop in accuracy is only present when the model is trained with noise. If the model is only validated with noise the decrease in validation accuracy is more gradual and therefore not a problem of immediate significance. A possible way to work around this problem would be to train the model with very expensive spatial light modulators that are not affected much by noise. The resulting configuration of the spatial light modulators can be transferred to less expensive spatial light modulators that only have to validate data. If the Vaporware model is realistic enough the model could be trained in a simulation where noise is not present and the resulting weights can be transferred to the setup. This process is called transfer learning and has been used in optical machine learning experiments before [9].

References

[1] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[2] François Chollet et al. Keras. `https://keras.io`, 2015.

[3] ekoulier. `https://stats.stackexchange.com/questions/330559/why-is-tanh-almost-always-better-than-sigmoid-as-an-activation-function`, 2016.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998.

[8] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[9] Xing Lin, Yair Rivenson, Nezih T. Yardimci, Muhammed Veli, Yi Luo, Mona Jarrahi, and Aydogan Ozcan. All-optical machine learning using diffractive deep neural networks. *Science*, 361(6406):1004–1008, 2018.

[10] Jocelyn Sietsma and Robert JF Dow. Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79, 1991.

[11] Smajida. Analysis of the effect of noisy data on cnn performance with mnist. `https://github.com/smajida/mnist-fun`, 2016.

[12] Bram Verreussel. On the importance of nonlinearity for physical neural networks. 2019.

7 Appendix

```
import numpy as np
import scipy
from scipy.optimize import curve_fit

f = lambda x, a, c : a * x**c + y0

p0 = [-1, 3]
best_vals, covar = curve_fit(f, X_values, Y_values, p0, sigma)
```

The code above is the code used to fit the data points in the results section. f defines the function to be fitted, a and c are the variables which will be changed to find the best possible values. y0 is the starting point which should equal to the validation accuracy of the fist data point. The *curve_fit* function then finds the best values using the x values, y values and the standard deviations of this data.

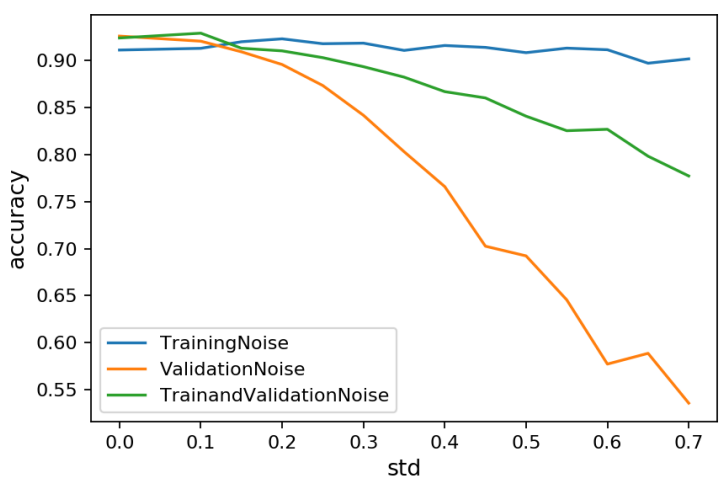

Figure 17: **Noise on the input images of an older version of the Vaporware model**

Figure 17 shows the response of the earlier vaporware model to noise on input images. The blue line is noise added to input images only during the training. The green line is the result of noise added to both the training images and validation images. Orange is the result of a model trained without noisy images but validated with noisy images. This affects the validation accuracy the most. The findings here are in line with the expectations from literature

that state 'training with noise increases the robustness of the model'[10].

7.1 Keras

The model was build in the Application programming interface Keras. In our case it was a program that uses Tensorflow functions to create neural networks. Keras is developed for fast experimentation that leaves the most complicated programming out. This allows the user to easily research ideas. Keras is written in the python programming language. In this thesis we used Colaboratory to execute Python code. Colaboratory is a Jupyter notebook environment that runs on a cloud. This allowed us to acces a GPU which increases the training speed of the models. In this thesis python version 3.6.7 and Keras version 2.2.4 is used.