Working Title: Reverse Design of Meta Surface Stacks

Bachelor Thesis

Friedrich-Schiller-Universität Jena
Physikalisch-Astronomische-Fakultät
Institute of Quantum and Nano-Optics

Tim Luca Turan the third

March 13, 2020

Evaluators

First Assessor:	Prof. Dr. rer. nat. Thomas Pertsch Institute of Quantum and Nano Optics Friedrich-Schiller-Universität Jena
Second Assessor:	M.Sc. Jan Sperrhake Friedrich-Schiller-Universität Jena Institute of Quantum and Nano Optics

1 Introduction

Metamaterials are materials whose physical properties emerge not from the kind of material they are but from their internal geometry. Optical metamaterials are mostly made of repeating sub-wavelength structures. They can be engineered to exhibit properties not found in nature like negative refractive indices [1]. In recent years the focus has shifted to flat 2D metamaterials dubbed metasurfaces. Normal optical devices like lenses or phase plates rely on distances much larger than one wavelength to change the light's properties in contrast metasurfaces have sub-wavelength thickness and enable thin components not possible before [2]. **1.too close to source?** These metasurfaces promise custom components which can be tailored to a wide range of applications just by changing the surface geometry but because of their small size metasurfaces with complex geometry are quite hard to manufacture. Also predicting the optical behavior of metasurfaces cannot be done analytically and involves computationally intensive simulations.

2016 [3] presented an algorithm called SASA to analytically calculate the optical behavior of metasurface stacks if one knows how the surfaces behave individually. This allows for a different approach when designing an optical component. The idea here is to create a target behavior not by using complex geometry but by using simple metasurfaces and stacking them ontop of each other. Designing a component becomes an optimization problem of choosing the right surfaces and then tuning stack parameters like the distance between layers and their rotation angle. This approach is possible because SASA is analytical in nature and many different stacks can be considered very quickly but it also poses some challenges when trying to automate the process. Conventional optimization methods rely on the underlying process being continuous so that if a parameter is tuned slightly it is possible to tell weither that was a step in the right direction. This is a problem because some of the choices involved when building a stack are categorical. For example, choosing the material of the layers or the kind of geometry.

Another recent development has produced a very general and powerful way of dealing with categorical decision problems. Neural Networks have found their way into many different areas and given enough data it should be possible to train a Neural Network to choose the discrete stack parameters to a given optical target. Again SASA being analytical is useful because it can generate a lot of training stacks very quickly. We can now sketch out an algorithm which takes a transmission spectrum as a target and outputs a metasurface stack to reproduce this target:

Preperation:

1. Conventionally simulate a database of metasurfaces with simple geometries
2. Use SASA to generate training stacks based on this database
3. Train a Neural Network on these stacks

Algorithm:

1. Input the transmission spectrum into the Neural Network and receive the stack parameters
2. Use a conventional optimization method to tune the continuous parameters
3. Use the database in conjunction with SASA to determine the optical behavior of the current stack.
4. Repeat 2. and 3. until the target accuracy is reached

In section 3 we will first develop the physical theory to understand metasurface stacks and learn about their limitations. Then we will look into the computer science background of Neural Networks and which network architecture to choose for the problem at hand. The Algorithm is described in detail in section 4 and is evaluated in section 5. All the code written for this project is open source and documented here: https://github.com/TimLucaTuran

Contents

ToDo

2 Notation

Symbols	Explanation
$\mathbf{E}$, $\mathbf{B}$, $\mathbf{k}$	Vectors are written bold
E, B, k	Magnitudes of vectors are written non-bold so that $\lvert\mathbf{E}\rvert = E$
$\hat{S}$, $\hat{J}$, $\hat{w}$	Matrices have an operator hat
$\hat{w}^2_{1,2}$, E^{in}_x	Super and sub indices are used to specify certain elements
$(y_i - y_i')^2$, $(n)^2$	All exponents are outside parentheses to differentiate between index and exponent
∇	Is the Nabla operator $\nabla = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix}$
Δ	Is the Laplace operator $\Delta = \begin{pmatrix} \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \\ \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \\ \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \end{pmatrix}$

3 Background

3.1 S-Matrix Calculus

To implement the desired algorithm we need to be able to calculate the optical behavior of stacked metasurfaces. The mathematical framework we will use is called Scatter Matrix Calculus and this section will give some insight into its physical origin and how to use it. We will start with the Maxwell Equations in matter.

Maxwell Equations

$$\nabla \times \mathbf{E}(\mathbf{r},t) = -\frac{\partial}{\partial t}\mathbf{B}(\mathbf{r},t) \tag{3.1}$$

$$\nabla \cdot \mathbf{D}(\mathbf{r},t) = \rho_{\text{ext}}(\mathbf{r},t) \tag{3.2}$$

$$\nabla \times \mathbf{H}(\mathbf{r},t) = \mathbf{j}(\mathbf{r},t) + \frac{\partial}{\partial t}\mathbf{D}(\mathbf{r},t) \tag{3.3}$$

$$\nabla \cdot \mathbf{B}(\mathbf{r},t) = 0 \tag{3.4}$$

The four involved fields are: $\mathbf{E}$...electric field, $\mathbf{B}$...magnetic flux density, $\mathbf{D}$...electric flux density and $\mathbf{H}$...magnetic field and the sources are the external charges ρ_{ext} and the macroscopic currents $\mathbf{j}$. All the material properties are captured by the $\mathbf{D}$ and $\mathbf{H}$ fields which are defined as:

$$\begin{aligned}\mathbf{D}(\mathbf{r},t) &= \varepsilon_0\mathbf{E}(\mathbf{r},t) + \mathbf{P}(\mathbf{r},t)\\ \mathbf{H}(\mathbf{r},t) &= \frac{1}{\mu_0}[\mathbf{B}(\mathbf{r},t) - \mathbf{M}(\mathbf{r},t)]\end{aligned} \tag{3.5}$$

Where $\mathbf{P}$ is the dielectric polarization and $\mathbf{M}$ is the magnetic polarization. One can read equation (3.5) in the following way: When the electric field $\mathbf{E}$ interacts with matter it exerts a force on all its charges and displaces them by a small amount. This separation of charges results in a counter field $\mathbf{P}$ and the total field $\mathbf{D}$ is now a superposition of $\mathbf{E}$ and $\mathbf{P}$. This set of equations describes the whole electromagnetic spectrum, in this work however we are only interested in visible (VIS) and near infrared (NIR) light where we can make some simplifications. Generally in optics materials are non-magnetizable so $\mathbf{M} = 0$ and there are no free charges $\rho_{\text{ext}} = 0$. Inserting these assumptions into the maxwell equation gives:

$$\nabla \times \mathbf{E}(\mathbf{r},t) = -\mu_0\frac{\partial}{\partial t}\mathbf{H}(\mathbf{r},t) \tag{3.6}$$

$$\varepsilon_0\nabla \cdot \mathbf{E}(\mathbf{r},t) = -\nabla \cdot \mathbf{P}(\mathbf{r},t) \tag{3.7}$$

$$\nabla \times \mathbf{H}(\mathbf{r},t) = \mathbf{j}(\mathbf{r},t) + \frac{\partial}{\partial t}\mathbf{P}(\mathbf{r},t) + \varepsilon_0\frac{\partial}{\partial t}\mathbf{E}(\mathbf{r},t) \tag{3.8}$$

$$\nabla \cdot \mathbf{H}(\mathbf{r},t) = 0 \tag{3.9}$$

Light in Vacuum

Now we can derive the famous wave equation by considering $\nabla\times$ (3.6):

$$\begin{aligned}&\nabla \times \left[\nabla \times \mathbf{E}\right] = \nabla \times \left[-\mu_0\frac{\partial}{\partial t}\mathbf{H}\right]\\ \iff\quad &\nabla(\nabla \cdot \mathbf{E}) - \Delta\mathbf{E} = -\mu_0\frac{\partial}{\partial t}\nabla \times \mathbf{H} \quad \Big| \text{ subs. (3.8) and (3.7)}\\ \iff\quad &\frac{1}{c^2}\frac{\partial^2}{\partial t^2}\mathbf{E} - \Delta\mathbf{E} = -\mu_0\frac{\partial}{\partial t}\mathbf{j} - \mu_0\frac{\partial^2}{\partial t^2}\mathbf{P} + \frac{1}{\varepsilon_0}\nabla(\nabla \cdot \mathbf{P})\end{aligned} \tag{3.10}$$

In vacuum ($\mathbf{P} = 0$ and $\mathbf{j} = 0$) the right side of this equation vanishes and we are left with $\frac{1}{c^2}\frac{\partial^2}{\partial t^2}\mathbf{E} - \Delta\mathbf{E} = 0$ which is solved by the plane wave $\mathbf{E} = \mathbf{E}_0 e^{i(\mathbf{kr}-\omega t)}$ where $\frac{\omega}{k} = c$. This describes the propagation of light through empty space: a sinusoidal oscillation in time and space along the $\mathbf{k}$ direction where $\mathbf{E}$, $\mathbf{B}$ and $\mathbf{k}$ are all perpendicular to each other. **2.this needs details 3.show figure?**

Light in homogeneous, isotropic materials

The next question we can answer is how light propagates through a homogeneous and isotropic material. For us, the dielectric polarization is some linear function of the electric field so $\mathbf{P}(\mathbf{r},t) = \hat{\chi}(\omega,\mathbf{r})\mathbf{E}(\mathbf{r},t)$. An isotropic material behaves the same for all orientations of $\mathbf{E}$ that means $\hat{\chi}(\omega,\mathbf{r})$ becomes a scalar function $\chi(\omega,\mathbf{r})$. If the material is additionally homogeneous, that is the same everywhere independent of $\mathbf{r}$, then $\boldsymbol{\nabla}\chi(\omega,\mathbf{r}) = 0$. With equation (3.7) this gives us $\boldsymbol{\nabla}\cdot\mathbf{P} = 0$ and the wave equation simplifies to:

$$\begin{aligned} \frac{\varepsilon}{c^2}\frac{\partial^2}{\partial t^2}\mathbf{E} - \Delta\mathbf{E} &= 0 \\ \text{where} \quad \varepsilon\,\mathbf{E} := (1+\chi)\mathbf{E} &= \mathbf{E} + \mathbf{P} \end{aligned} \tag{3.11}$$

That means light behaves in these materials exactly as it would in vacuum we only have to account for a decreased speed of light $c' = \frac{c}{\sqrt{\varepsilon}} =: \frac{c}{n}$ with the refractive index n. This is equivalent to a decreased wavelength $\lambda' =: \frac{\lambda}{n}$ or an increased wave vector $k' = \frac{2\pi}{\lambda'} = n\,k$. A complex valued $n := \eta + i\kappa$ even captures the possibility of an exponentially decaying field:

$$\mathbf{E} = \mathbf{E}_0 e^{i(n\mathbf{kr}-\omega t)} = \mathbf{E}_0 \underbrace{e^{-\kappa\mathbf{kr}}}_{\text{decay}} \underbrace{e^{i(\eta\mathbf{kr}-\omega t)}}_{\text{oscillation}} \tag{3.12}$$

Interfaces

The metasurface stacks we want to understand are obviously not one homogeneous material. Rather they contain many interfaces between different materials and we can again use the maxwell equations to predict how light will behave at such an interface.

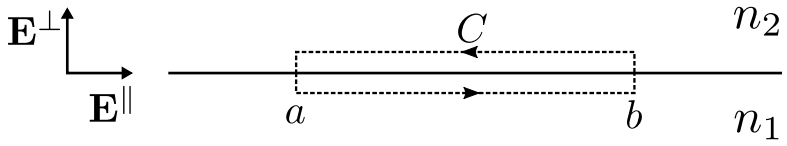


Figure 1: An interface of two materials with different refractive indices n_1 and n_2 and a closed contour C which is tangent to the interface between the points a and b.

Let us consider a closed contour C around an interface as seen in Figure 1 and integrate the first maxwell equation (3.6) over the surface A enclosed by that contour:

$$\begin{aligned} \int_A \boldsymbol{\nabla}\times\mathbf{E}(\mathbf{r},t)\,\mathrm{d}\mathbf{A} &= -\mu_0\frac{\partial}{\partial t}\int_A \mathbf{H}(\mathbf{r},t)\mathrm{d}\mathbf{A} \\ \overset{\text{stokes}}{\Longleftrightarrow} \qquad \int_{C=\partial A} \mathbf{E}(\mathbf{r},t)\mathrm{d}\mathbf{r} &= -\mu_0\frac{\partial}{\partial t}\int_A \mathbf{H}(\mathbf{r},t)\mathrm{d}\mathbf{A} \end{aligned} \tag{3.13}$$

But now we can bring the contour infinitely close to the interface and thus reduce the right hand side of the equation, the total magnetic flux, through the surface, to zero. That leaves us with:

$$\begin{aligned} \int_a^b \mathbf{E}_1\,\mathrm{d}\mathbf{r} \;+ \int_b^a \mathbf{E}_2\,\mathrm{d}\mathbf{r} &= 0 \\ \Longleftrightarrow \qquad \int_a^b \mathbf{E}_1\,\mathrm{d}\mathbf{r} &= \int_a^b \mathbf{E}_2\,\mathrm{d}\mathbf{r} \end{aligned} \tag{3.14}$$

Because a and b were chosen arbitrarily that means that the transverse field components along the path need to be continuous so $\mathbf{E}_1^{\parallel} = \mathbf{E}_2^{\parallel}$. The analog expression $\mathbf{H}_1^{\parallel} = \mathbf{H}_2^{\parallel}$ can be shown by starting with the third maxwell equation (3.8) instead. We can now use these continuity conditions to learn more about the behavior of light at these interfaces.

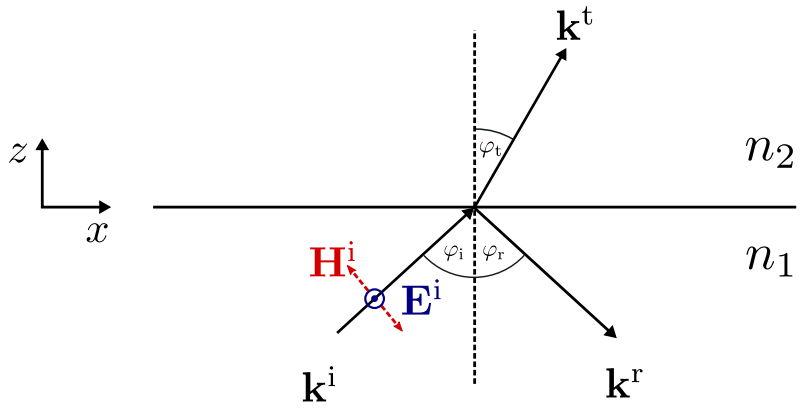


Figure 2: Interface between two materials of different refractive indices n_1 and n_2. The incident light has a wave vector $\mathbf{k}^{\mathrm{i}}$ and is partially transmitted and partially reflected. The fields are shown in transverse electric (TE) polarization where $\mathbf{H}$ is in the x-z plane and $\mathbf{E}$ occilates in the y direction.

Let us consider the same interface from before and have an incident field $\mathbf{E}^{\mathrm{i}}$ interact with it. From figure 2 we can see:

$$\mathbf{k}^{\mathrm{i}} = k_0 n_1 \begin{pmatrix} -\sin\varphi_{\mathrm{i}} \\ 0 \\ -\cos\varphi_{\mathrm{i}} \end{pmatrix}, \quad \mathbf{k}^{\mathrm{r}} = k_0 n_1 \begin{pmatrix} \sin\varphi_{\mathrm{r}} \\ 0 \\ -\cos\varphi_{\mathrm{r}} \end{pmatrix}, \quad \mathbf{k}^{\mathrm{t}} = k_0 n_2 \begin{pmatrix} \sin\varphi_{\mathrm{t}} \\ 0 \\ \cos\varphi_{\mathrm{t}} \end{pmatrix} \tag{3.15}$$

Now we can decompose the electric field into two orthogonal polarizations. One component were $\mathbf{E} = E\,\vec{\mathbf{e}_{\mathbf{y}}}$ called transverse electric (TE) as shown in figure 2 and its orthogonal component where $\mathbf{H} = H\,\vec{\mathbf{e}_{\mathbf{y}}}$ called transverse magnetic (TM). If we apply the continuity condition from before to the TE component we get $\mathbf{E}_1 = \mathbf{E}_2$ at $z = 0$:

$$\begin{aligned} \mathbf{E}^{\mathrm{i}} + \mathbf{E}^{\mathrm{r}} &= \mathbf{E}^{\mathrm{t}} \\ E^{\mathrm{i}} e^{i(k_x^{\mathrm{i}} x)} + E^{\mathrm{r}} e^{i(k_x^{\mathrm{r}} x)} &= E^{\mathrm{t}} e^{i(k_x^{\mathrm{t}} x)} \end{aligned} \tag{3.16}$$

This is only possible for all x if:

$$E^{\mathrm{i}} + E^{\mathrm{r}} = E^{\mathrm{t}} \quad \text{and} \quad k_x^{\mathrm{i}} = k_x^{\mathrm{r}} = k_x^{\mathrm{t}} \tag{3.17}$$

Two basic laws of optics lie in this relation: The law of reflection

$$k_x^{\mathrm{i}} = k_x^{\mathrm{r}} \quad \Rightarrow \quad k_0 n_1 \sin\varphi_i = k_0 n_1 \sin\varphi_r \quad \Rightarrow \quad \varphi_i = \varphi_r := \varphi_1 \tag{3.18}$$

and the Snells law of refraction

$$k_x^{\mathrm{i}} = k_x^{\mathrm{t}} \quad \Rightarrow \quad k_0 n_1 \sin\varphi_i = k_0 n_2 \sin\varphi_t \quad \Rightarrow \quad n_1 \sin\varphi_i = n_2 \sin\varphi_t \tag{3.19}$$

Fresnel Equations

To fully describe an interface we also need know the fraction of the transmitted $t := E^t/E^i$ and the reflected $r := E^r/E^i$ fields. Again using the TE polarization we have an orientation of fields were $\mathbf{E} = E\,\vec{\mathbf{e}}_y$ and $\mathbf{H}$ is contained perpendicular to $\mathbf{E}$ in the x-z plane:

$$\mathbf{H}^{\mathrm{i}} = H^{\mathrm{i}} \begin{pmatrix} -\cos\varphi_{\mathrm{i}} \\ 0 \\ -\sin\varphi_{\mathrm{i}} \end{pmatrix}, \quad \mathbf{H}^{\mathrm{r}} = H^{\mathrm{r}} \begin{pmatrix} \cos\varphi_{\mathrm{r}} \\ 0 \\ \sin\varphi_{\mathrm{r}} \end{pmatrix}, \quad \mathbf{H}^{\mathrm{t}} = H^{\mathrm{t}} \begin{pmatrix} -\cos\varphi_{\mathrm{t}} \\ 0 \\ \sin\varphi_{\mathrm{t}} \end{pmatrix} \tag{3.20}$$

The H_x component is tangent to the interface so it also needs to be continuous across the boundary:

$$\begin{aligned} & H_x^{\mathrm{i}} + H_x^{\mathrm{r}} = H_x^{\mathrm{t}} \\ \Longleftrightarrow \quad & H^{\mathrm{i}} \cos\varphi_1 - H^{\mathrm{r}} \cos\varphi_1 = H^{\mathrm{t}} \cos\varphi_2 \end{aligned} \tag{3.21}$$

Maxwells first equation (3.6) allows us connect the magnitudes of $\mathbf{H}$ and $\mathbf{E}$ via the refractive index $H \sim n\,E$. This changes (3.21) to:

$$n_1\,E^{\mathrm{i}}\cos\varphi_1 - n_1\,E^{\mathrm{r}}\cos\varphi_1 = n_2\,E^{\mathrm{t}}\cos\varphi_2 \tag{3.22}$$

Substituting E^{r} or E^{i} with (3.17) and rearranging we get:

$$t = \frac{2n_1\cos\varphi_1}{n_1\cos\varphi_1 + n_2\cos\varphi_2} \quad \text{and} \quad r = \frac{n_1\cos\varphi_1 - n_2\cos\varphi_2}{n_1\cos\varphi_1 + n_2\cos\varphi_2} \tag{3.23}$$

These are called the Fresnel equations for the TE component. The TM component can be treated analogous which yields:

$$t = \frac{2n_1\cos\varphi_1}{n_2\cos\varphi_1 + n_1\cos\varphi_2} \quad \text{and} \quad r = \frac{n_2\cos\varphi_1 - n_1\cos\varphi_2}{n_2\cos\varphi_1 + n_1\cos\varphi_2} \tag{3.24}$$

For perpendicular incident light at $\varphi = 90°$ these should describe the same situation as we can no longer differentiate between TE and TM components and indeed for this angle they are equivalent if we consider that the TE factors describe the electric field and the TM factors describe the magnetic field. The TE factors become:

$$t = \frac{2n_1}{n_1 + n_2} \quad \text{and} \quad r = \frac{n_1 - n_2}{n_1 + n_2} \tag{3.25}$$

Stacking

We can take this one step further by including interfaces which have incident light from both sides. Let $\mathbf{E}_{\mathrm{out}}$ be the light coming out of the interface and $\mathbf{E}_{\mathrm{in}}$ be the light going into the interface as seen in figure 3:

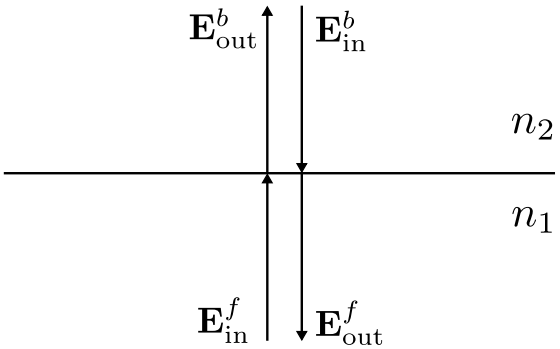


Figure 3: Interface between two materials of different refractive indices n_1 and n_2 were incident light is coming from the front and the back.

Using the factors from the Fresnel equations we get:

$$E^b_{\mathrm{out}} = E^f_{\mathrm{in}}\,t^f + E^b_{\mathrm{in}}\,r^b \quad \text{and} \quad E^f_{\mathrm{out}} = E^b_{\mathrm{in}}\,t^b + E^f_{\mathrm{in}}\,r^f \tag{3.26}$$

This can concisely be expressed using matrices and vectors:

$$\begin{pmatrix} E^f_{\mathrm{out}} \\ E^b_{\mathrm{out}} \end{pmatrix} = \underbrace{\begin{pmatrix} t^f & r^b \\ r^f & t^b \end{pmatrix}}_{=:\hat{S}} \begin{pmatrix} E^f_{\mathrm{in}} \\ E^b_{\mathrm{in}} \end{pmatrix} \tag{3.27}$$

We call these matrices mapping field-in to field-out S-matrices. They can be used to describe more than just interfaces. As shown in the paragraph Light in materials when light propagates through homogeneous isotropic material just the phase changes by a factor of $e^{ik_0 nd}$ where d is the distance traveled and $\mathbf{k}_0$ the wave vector in vacuum. We can express this by allowing complex valued t and r:

$$\hat{S}_{n,d} = \begin{pmatrix} e^{ik_0 nd} & 0 \\ 0 & e^{ik_0 nd} \end{pmatrix} \tag{3.28}$$

Analogue the S-matrix for an interface from n_1 to n_2 is:

$$\hat{S}_{n_1,n_2} = \begin{pmatrix} \frac{2n_1}{n_1+n_2} & \frac{n_2-n_1}{n_1+n_2} \\ \frac{n_1-n_2}{n_1+n_2} & \frac{2n_1}{n_1+n_2} \end{pmatrix} \tag{3.29}$$

Say we have a stack of different homogeneous isotropic materials. We are now are now able to write down an S-matrix for every part of the stack that is every interface and every propagation between interfaces as seen in figure 4, but we cannot predict the behavior of the stack as a whole.

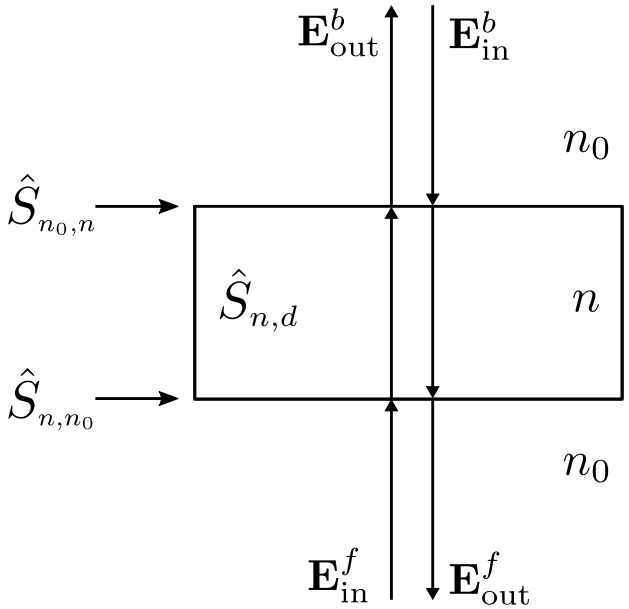


Figure 4: A slab of one homogeneous isotropic material with thickness d in air (n_0). This setup is described by two interface and one propergation S-matrices.

The combined S-matrix $\hat{S}$ consisting of $\hat{S}_1$ and $\hat{S}_2$ cannot be obtained by simply using the matrix product. The result would be:

$$\hat{S}_1 \, \hat{S}_2 = \begin{pmatrix} t_1^f \, t_2^f + r_1^b \, r_2^f & t_1^f \, r_2^b + r_1^b \, t_2^b \\ r_1^f \, t_2^f + t_1^b \, r_2^f & r_1^f \, r_2^b + t_1^b \, t_2^b \end{pmatrix} \neq \begin{pmatrix} t^f & r^b \\ r^f & t^b \end{pmatrix} = \hat{S} \tag{3.30}$$

and this would imply that the transmission from the front t^f increases when the internal reflections of the system r_1^b and r_2^f increase which is the opposite of the behavior we expect. Redheffer [4] studied these kind of systems in the 60s and gave an operator $\star$ which yields the combined S-matrix:

$$\hat{S}_1 \star \hat{S}_2 := \begin{pmatrix} t_2^f (1 - r_1^b r_2^f)^{-1} t_1^f & r_2^b + t_2^f r_1^b (1 - r_2^f r_1^b)^{-1} t_2^b \\ r_1^f + t_1^b r_2^f (1 - r_1^b r_2^f)^{-1} t_1^f & t_1^b (1 - r_2^f r_1^b)^{-1} t_2^b \end{pmatrix} \tag{3.31}$$

For this operation to produce physically valid results certain condition have to be met. We will discuss these in detail in the paragraph Conditions of section 3.2. Applied to the example of figure 4 that gives:

$$\hat{S} = \hat{S}_{n_0,n} \star \hat{S}_{n,d} \star \hat{S}_{n,n_0} \tag{3.32}$$

Polarization

Up to this point we are dealing with vectors of the form $\begin{pmatrix} E_{\text{in}}^f \\ E_{\text{in}}^b \end{pmatrix}$ where E_{in}^f and E_{in}^b are scalar properties. That means we can only describe linear polarized light in one direction. The last generalization we want to make is to extend this formalism to all polarizations. As shown in the paragraph Light in materials a planar light wave propagating along the z axis through a homogeneous isotropic material can be described as:

$$\mathbf{E} = \begin{pmatrix} E_x e^{i(kz-\omega t+\varphi_x)} \\ E_y e^{i(kz-\omega t+\varphi_y)} \\ 0 \end{pmatrix} = \left(E_x\, e^{i\varphi_x}\, \vec{\mathbf{e_x}} + E_y\, e^{i\varphi_y}\, \vec{\mathbf{e_y}}\right) e^{i(kz-\omega t)} \tag{3.33}$$

the waves polarization is determined by the scaling factors of $\vec{\mathbf{e_x}}$ and $\vec{\mathbf{e_y}}$ and can be expressed as a *Jones Vector* $\mathbf{j} \in \mathbb{C}^2$.

$$\mathbf{j} = \frac{1}{\sqrt{E_x^2 + E_y^2}} \begin{pmatrix} E_x \\ E_y\, e^{i\delta} \end{pmatrix} \quad \text{with} \quad \delta := \varphi_y - \varphi_x \tag{3.34}$$

Now all linear operations on the polarization are matrices $\hat{M} \in \mathbb{C}^{2\times 2}$. That means all passive components have a corresponding matrix. A couple examples for components in horizontal position:

$$\begin{aligned} \text{polarizer:} \quad & \hat{M} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ \lambda/4 \text{ plate:} \quad & \hat{M} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} e^{-\frac{i\pi}{4}} \\ \lambda/2 \text{ plate:} \quad & \hat{M} = \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \end{aligned} \tag{3.35}$$

We can now generalize the S-matrix calculus simply by allowing Jones matrices in place of the Fresnel factors: $t \to \hat{T}$ and $r \to \hat{R}$. The star product of two S-matrices becomes:

$$\hat{S}_1 \star \hat{S}_2 = \begin{pmatrix} \hat{T}_1^f & \hat{R}_1^b \\ \hat{R}_1^f & \hat{T}_1^b \end{pmatrix} \star \begin{pmatrix} \hat{T}_2^f & \hat{R}_2^b \\ \hat{R}_2^f & \hat{T}_2^b \end{pmatrix} = \begin{pmatrix} \hat{T}_2^f (\mathbb{1} - \hat{R}_1^b \hat{R}_2^f)^{-1} \hat{T}_1^f & \hat{R}_2^b + \hat{T}_2^f \hat{R}_1^b (\mathbb{1} - \hat{R}_2^f \hat{R}_1^b)^{-1} \hat{T}_2^b \\ \hat{R}_1^f + \hat{T}_1^b \hat{R}_2^f (\mathbb{1} - \hat{R}_1^b \hat{R}_2^f)^{-1} \hat{T}_1^f & \hat{T}_1^b (\mathbb{1} - \hat{R}_2^f \hat{R}_1^b)^{-1} \hat{T}_2^b \end{pmatrix} \tag{3.36}$$

So the general S-matrix is $\hat{S} \in \mathbb{C}^{4\times 4}$ and consist of four separate Jones matrices and the general input vector $\begin{pmatrix} \mathbf{E}_{\text{in}}^f \\ \mathbf{E}_{\text{in}}^b \end{pmatrix}$ consists of two separate Jones vectors.

3.2 SASA

The S-matrix calculus enables us to calculate the optical behavior of a stack if we know the optical behavior of all its layers in the form of their S-matrices. In section 3.1 we derived the S-matrices of propagation through and interfaces between homogeneous isotropic materials. This calculation is based on the Fresnel equations and is therefore analytic. We cannot however calculate the S-matrix of a metasurface analytically. This results in a "Semi Analytic" Stacking Algorithm where the S-matrices of the metasurfaces need to be provided but the remaining S-matrices can be calculated based on the stack parameters.

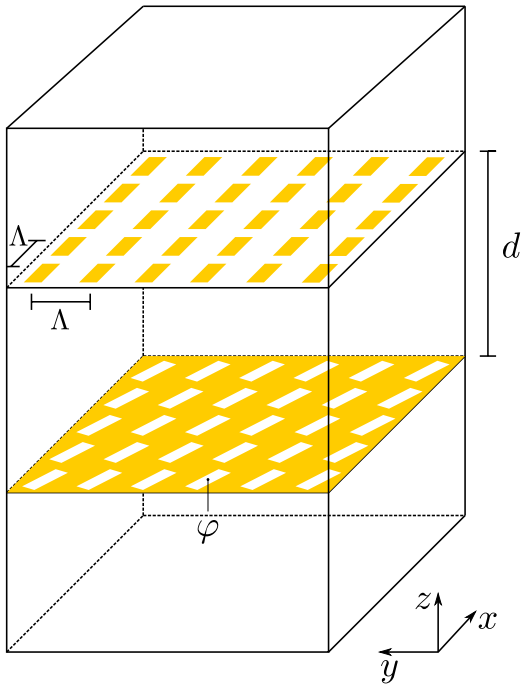


Figure 5: An example for a two metalayer stack. The top layer consist of rectangular gold particles arranged on a square matrix of period Λ. The bottom layer consist of rectangular holes in a gold sheet which are rotated by an angle of φ in respect to the top layer. The two metalayers are separated by a glass spacer of thickness d.

Importantly geometric transformations like rotation and mirroring can be applied directly to Jones matrices. For example let $\hat{M}$ be the Jones matrix of an optical component then the Jones matrix of the rotated component $\hat{M}_\varphi$ is calculated as:

$$\hat{M}_\varphi = \hat{\Theta}(-\varphi)\,\hat{M}\,\hat{\Theta}(\varphi) \quad \text{where} \quad \hat{\Theta}(\varphi) = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \tag{3.37}$$

On this basis we can rotate S-matrices in a similar manner:

$$\hat{S}_\varphi = \begin{pmatrix} \hat{\Theta}(-\varphi)\hat{T}^f\hat{\Theta}(\varphi) & \hat{\Theta}(-\varphi)\hat{R}^b\hat{\Theta}(\varphi) \\ \hat{\Theta}(-\varphi)\hat{R}^f\hat{\Theta}(\varphi) & \hat{\Theta}(-\varphi)\hat{T}^b\hat{\Theta}(\varphi) \end{pmatrix} \tag{3.38}$$

Flipping and mirroring of S-matrices is done analogously. With all mathematical operations well defined we can write down SASA in pseudo code:

Input: Stack = [Layer 0, Layer 1, ...]
Output: Stack S-matrix
S-mats = [ ]
for $i = 0$; $i <$ len(Stack); $i = i + 1$; **do**
  if *Stack*[i] **is** *metasurface* **then**
    add layer $\hat{S}$ to S-mats
  else
    calculate propagation S-matrix $\hat{S}_{n_i,d_i}$
    add $\hat{S}_{n_i,d_i}$ to S-mats
  end
  apply geometric transformations to added S-matrix
  calculate interface S-matrix $\hat{S}_{n_i,n_{i+1}}$
  add $\hat{S}_{n_i,n_{i+1}}$ to S-mats
end
$\hat{S} = \mathbb{1}$
for $i = 0$; $i <$ len(*S-mats*); $i = i + 1$; **do**
  $\hat{S} = \hat{S} \star S$-mats[$i$]
end
return $\hat{S}$

4.fix cursive?

SASA pseudocode: A Layer variable holds all the information of that layer, that is the S-matrix if its a metalayer and the refractive index n and thickness d otherwise.

Conditions

For this algorithm to produce physical valid output some conditions need to be met. Equations that are boundary conditions to the algorithm will be marked bold. It has been shown 5.needs cite that the S-matrix calculus can only be used when the metamaterials in the stack interact with each other via the far field. That is only the zeroth diffraction order of a metasurface can be non evanescent and the higher orders need to have decayed enough when they reach the next metasurface.

$$\lambda > \max(n^f, n^b) \cdot \Lambda \tag{3.39}$$

(3.39) ensures that only the zeroth diffraction order is non evanescent but additionally the spacer between the metasurfaces as seen in figure 5 needs to be thick enough so that all higher order modes have decayed sufficiently. There is a calculation here [3] which concludes that if we require:

$$e^{\mathrm{Im}(k_z)d} < e^{-2} \approx 0.14 \quad \text{then} \quad d > \frac{\Lambda}{\pi\sqrt{1 - \frac{\Lambda^2 n^2}{\lambda^2}}} \tag{3.40}$$

The factor e^{-2} is chosen arbitrarily. A smaller value increases the agreement of the S-matrix calculus and more rigoros methods which also take into account the near field.

Symmetries

We can use geometric transformations to examine the S-matrices of metasurfaces which satisfy certain symmetries. Lets start with mirror symmetry. The Jones matrix $\hat{M}'$ of a mirrored component $\hat{M}$ can be calculate by:

$$\hat{M}' = \hat{F}^{-1}\hat{M}\hat{F} \quad \text{where} \quad \hat{F} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{3.41}$$

If a metasurface has mirror symmetry like the ones seen in figure 5 the S-matrix needs to satisfy $\hat{S} = \hat{S}'$ and that means all the contained Jones matrices need to satisfy the condition too:

$$\text{Let} \quad \hat{T}^f = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad \text{then}$$
$$\left(\hat{T}^f\right)' = \hat{F}^{-1}\hat{T}^f\hat{F} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \overset{!}{=} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \hat{T}^f \tag{3.42}$$
$$\Rightarrow \qquad \hat{T}^f = \begin{pmatrix} A & 0 \\ 0 & D \end{pmatrix}$$

We can also examine rotational symmetry with the rotation matrix $\hat{\Theta}(\varphi)$. A metasurface of a C_4 symmetry for example square shaped metaparticles, is equivalent if rotated 90° so the S-matrix should be the same too $\hat{S} \overset{!}{=} \hat{S}_{\pi/2}$:

$$\hat{\Theta}(\pi/2) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad \text{and} \quad \hat{T}^f_{\pi/2} = \hat{\Theta}(-\pi/2)\hat{T}^f\hat{\Theta}(\pi/2) \overset{!}{=} \hat{T}^f$$
$$\Rightarrow \qquad \hat{T}^f = \begin{pmatrix} A & B \\ -B & A \end{pmatrix} \tag{3.43}$$

So if a metasurface has both mirror and C_4 symmetry its $\hat{T}^f$ matrix is shaped:

$$\hat{T}^f = \begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix} \tag{3.44}$$

That means in this case the star product is commutative and the stack will behave the same for light coming from the bottom and light coming from the top. And it should be noted that these considerations apply to all Jones matrices and not only to $\hat{T}^f$

3.3 Neural Networks

Artificial Neural Networks (ANN's or short NN's) are a kind of data structure inspired by the biological neurons found in nature. They can be used to find a wide range of input output relations. One classic example is mapping pictures of hand written digits to the actual digits. Rather than explicitly programmed, NN's are trained on a dataset (X, Y) of correct input output pairs.

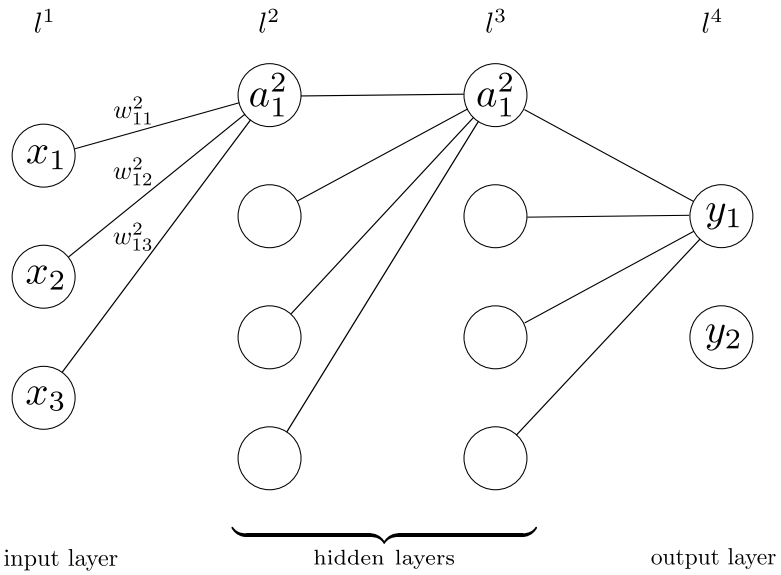


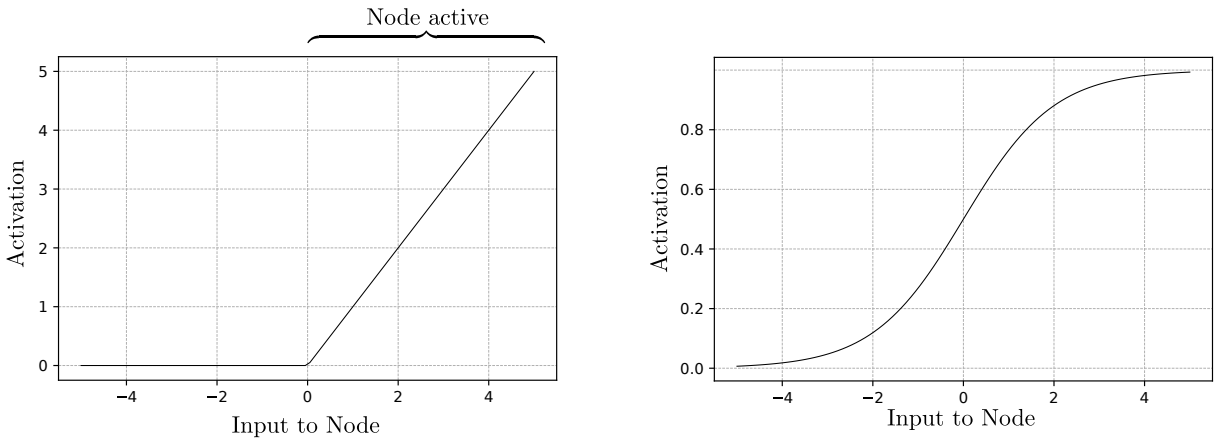
Figure 6: The most simple kind of NN is called densely connected or multilayer perceptron. For clarity only connections to the top most node of each layer are shown.

Multilayer Perceptron This kind of classic NN consist of single nodes or neurons which are organized into layers. The terms node and neuron can be used interchangeably. Every node is connected to all the nodes of the previous and the next layer. For this reason the network is called dense or densely connected. Each node holds a value called activation a where the activation to the first layer is the input to the network, here: (x_1, x_2, x_3). The nodes are connected by weights w which specify how much one node should influence the next and every node has a bias b to control at what total input activation the node itself should become active. To calculate the activation of a node one has to multiply all the activations of the previous layer with their respective weights w, add the bias b and finally apply a non-linear activation function σ as seen in Figure 7. To describe this process mathematically we are going to use the usual index notation where superscripts specify the layer and subscripts the node. So a^2_1 is the activation of the first node in the second layer. To characterize a weight two subscripts are needed for the end and beginning of the connection. For the example in figure 6 that means:

$$a^2_1 = \sigma\left(\sum_i w^2_{1i}\, x_i + b^2_1\right) \tag{3.45}$$

However it is more convenient to stop considering every node individually and to view the involved quantities as vectors and matrices. So that (3.45) can be written as:

$$\mathbf{a}^l = \sigma\left(\hat{\mathbf{w}}^l \mathbf{a}^{l-1} + \mathbf{b}^l\right) \tag{3.46}$$



(a) rectified linear unit (ReLu)

(b) sigmoid

Figure 7: Two examples of activation functions σ. Especially the ReLu function has a destinct on and off state similar to a biological neurons.

Training During training the network output $\mathrm{NN}(\mathbf{x}) := \mathbf{y}'$ is calculated through repeated use of (3.46) and is then compared with the known correct output $\mathbf{y}$ by a cost function $C = C(\mathbf{y}, \mathbf{y}')$. The goal of the training is to minimize this function C. The cost function might simply be the mean squared difference between $\mathbf{y}$ and $\mathbf{y}'$:

$$C_{\mathrm{mse}}(\mathbf{y}, \mathbf{y}') = \sum_i \left(y_i - y_i'\right)^2 \tag{3.47}$$

but there are different cost functions for different kind of outputs. For example a network which predicts continuous values needs a different loss than one predicting categories. More on this in section 4.2. Now we can quantify how well the NN is performing but how should the weights and biases be changed to improve this performance? Here the Algorithm *Backpropagation* is used and allows an efficient approximation of the gradients for $\mathbf{b}^l$ and $\hat{\mathbf{w}}^l$. These are used to gradually change the weights and biases to minimize the cost function. A comprehensive explanation of Backpropagation can be found here: [5].

Convolutional Neural Networks An area where NNs have been very successful is image recognition or more general computer vision but the described multilayer perceptron has a number of weaknesses for this kind of task. Let's say our input is a n by n gray scale image. This can be expressed as a $n \times n$ matrix, flattened and fed into the input layer (see figure 8). But now the number of weights to the next layer $\hat{\mathbf{w}}^2$ is $n \cdot n \cdot \left|\boldsymbol{l}^2\right|$ which soon becomes unfeasible. As described in the section on notation $\boldsymbol{l}^2$ is here the activation vector of the second layer and not $\boldsymbol{l}$ squared.

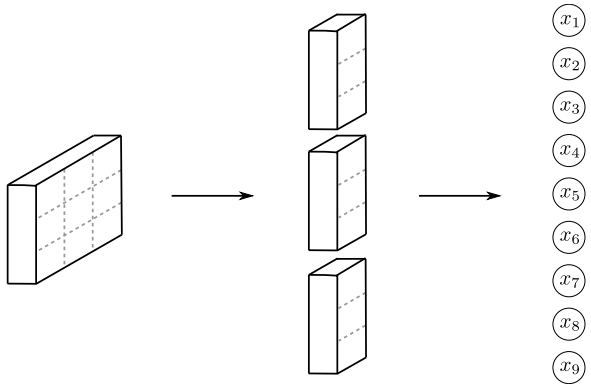


Figure 8: Flattening of a 3×3 matrix to fit the input of a multilayer perceptron.

Computational limits aside there is another problem. Imagine an image with the letter T in the top

right corner. If this letter moves to a different position as in figure 9 the networks reaction will be completely different because the weights and biases involved are completely different. So the NN cannot learn the concept "letter T" independent of its position in the picture. Also the information about the distance between pixels is lost.

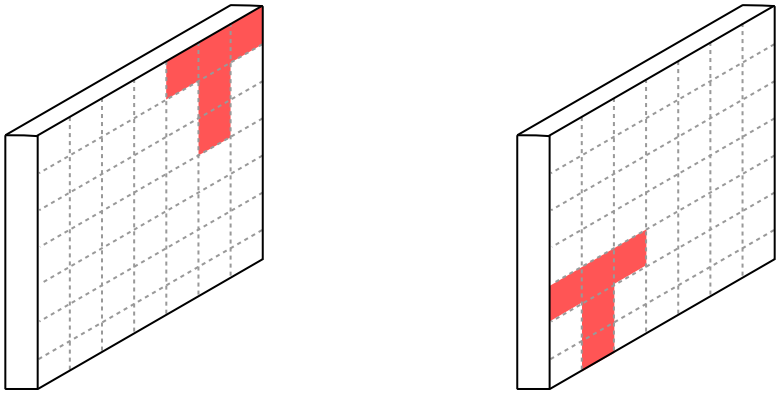


Figure 9: Two pictures of a T at different positions where the red color signifies a high value in the grayscale image. After the flatten operation seen in figure 8 very different nodes are active.

These problems led to the development of a new kind of layer called *Convolution*. A fixed size squared matrix called *kernel* is shifted over the matrix and at every position the the point wise product between kernel and matrix is calculated and summed as shown in figure 10:

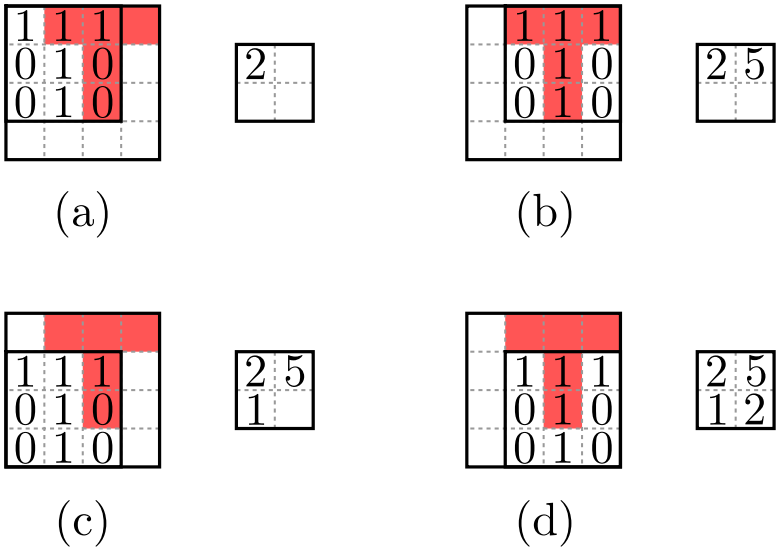


Figure 10: Example of a convolution. The 3×3 kernel is shifted over the image one step at the time. The red color in the image represents a pixel value of 1. For example in picture (a) the point wise product between kernel and image is zero everywhere except at two positions where a one in the kernel meets a one in the image. Because there are less valid positions for the kernel than pixels in the image the result is smaller in size.

The result of this operation called feature map of that kernel (see figure 11). Notice how the greatest value of the feature map is at the position of the letter T. So with only a small number of weights the convolution is able to detect the T independent of its position in the image. This is still slightly misleading because this "T kernel" was intentionally constructed to find the "T". In a real convolutional layer the kernel values are trained via Backpropagation similar to the weights of a Multilayer Perceptron as described in the paragraph Training.

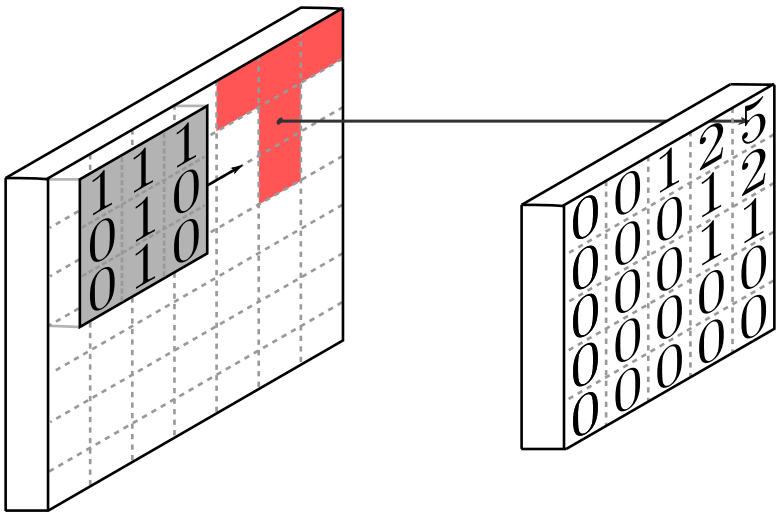


Figure 11: Example of a convolution where white pixel are 0 and red pixel are 1. The 3×3 Kernel is shifted over the image and when its directly over the The "T kernels" feature map is greatest at the position of the original letter.

One convolutional layer contains not only one but a number of different kernels k. The resulting k feature maps are stacked in the "z direction" so that the shape of the $n \times n$ matrix transforms to $(n-2) \times (n-2) \times k$. In a Convolutional Network (ConvNet) multiple of these layers are used so that it can find "patterns in patterns". For the letter detection example one could imagine the first layer to detect various edges and the next layer to detect letters in the position of these edges.

Pooling Layers For a big image and a large number of kernels the output shape of a convolutional layers is still $\mathcal{O}\big((n)^2 \cdot k\big)$, so quite large. Also notice how in figure 11 the "T kernel's" feature map is not only active at the exact position of the T but in the general region. The solution to this is to downsample the output with a *Pooling Layer*. Here a smaller kernel, usually 2×2 is shifted over the matrix two steps at a time and at every position an operation is performed to reduce the number of values to one. This could be taking the maximum or the average of that 2×2 region. This operation reduces the matrix in the x and y dimension by a factor of 2 as shown in figure 12:

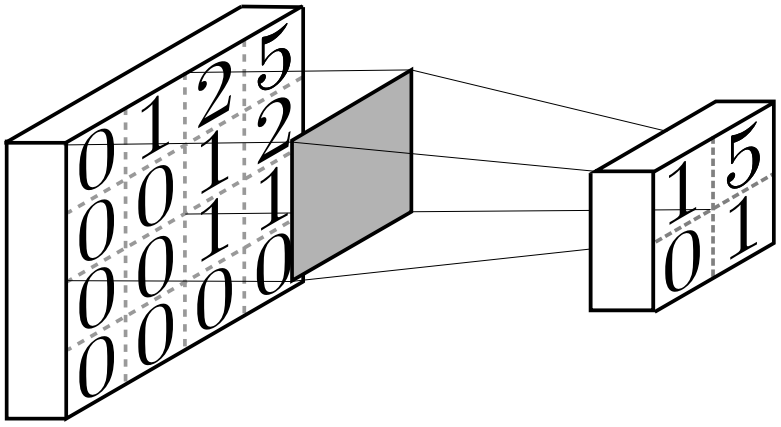


Figure 12: Example of a Max Pooling Layer. For every 2 by 2 field the maximum is calculated. After applying first the convolution and then the pooling layer the information "T in the top right corner" is still there and size of the resulting matrix is very manageable.

Example Network Architecture Now all the building blocks for a complete ConvNet are available. Repeatedly alternating convolution and pooling layers changes the input from wide in x and y dimension and narrow in z to a long z-strip. At the very end this strip is fed into one densely connected layer which is in turn connected to the output neurons. An example architecture of this kind is shown in

figure 13:

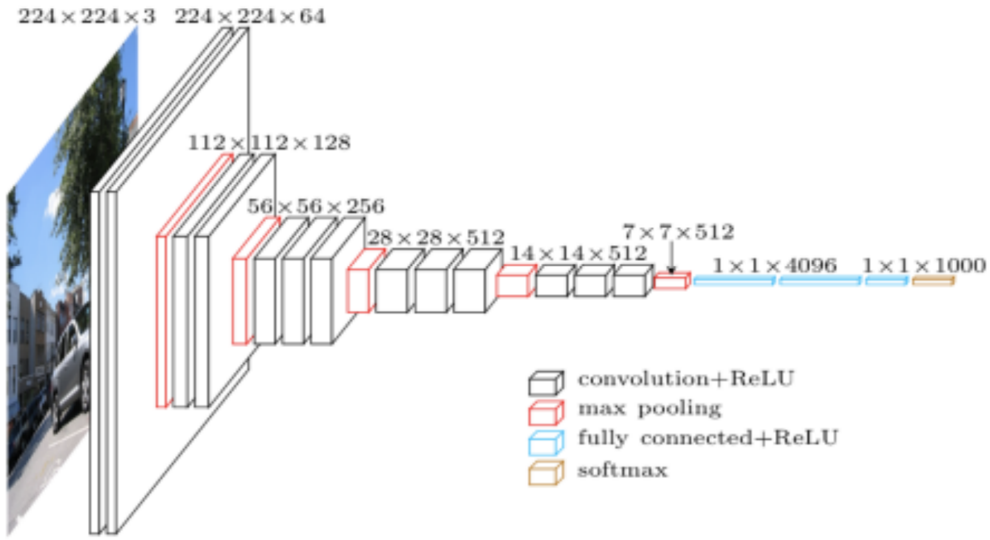


Figure 13: Example for a complete ConvNet. Note the input is in this case an RGB image so there are three layers in the z dimension corresponding to the different colors. In this case the first layer has 64 kernels of size $3 \times 3 \times 3$. The next convolution then has 128 kernels of size $3 \times 3 \times 64$. [citation needed]

1D ConvNets The input to the desired algorithm is a target spectrum to which the Network should output some parameters (more on this in the section 4.2). This input data is a function $I(\lambda)$ so only one dimensional but all the same ideas apply. Convolutional kernels are sized $1 \times 3 \times z$ and pooling kernels are $1 \times 2 \times z$. Both are only shifted in one direction as see in figure 14. These 1D convolutions might detect features like rising and falling edges and the later layers might combine these features into concepts like peaks and troughs but as always in machine learning what the network actually does to reach its objective is not controlled by the programmer.

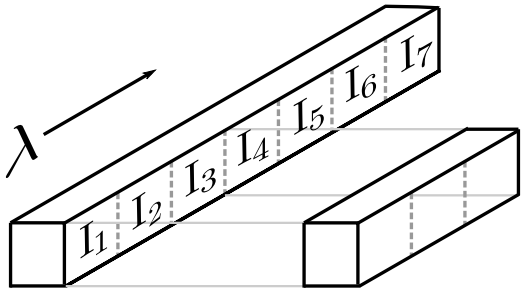
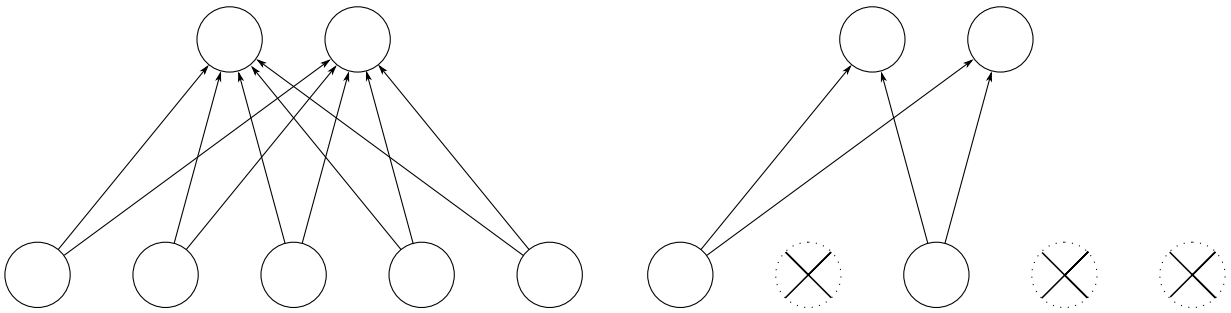


Figure 14: Example of a 1D convolution. A 1×3 kernel is shifted over a spectrum I discretized at 9 wavelengths

Dropout Layer In 2014 Srivastava et al.[6] presented a method to prevent overfitting and speed up the training process of large Neural Networks. During training they randomly drop a number of neurons in a layer along with all connections to and from these neurons. This prevents the neurons from co-adapting **6.explain co-adapting** and because there are less weights and biases to tune for each step the training becomes overall faster. The process of dropping some neurons is shown in figure 15:



(a) no dropout (b) with dropout

Figure 15: Example of a dropout applied to the bottom layer. Only two of the neurons remain active and only their weights and biases are modified during this training step. Figure found in [6] (modified)

4 Algorithm

Before starting to implement the algorithm we need to define which metasurface stacks the algorithm is allowed to produce. This was already partly motivated in the introduction: We want easy to manufacture metasurfaces which can produce a high variety of transmission spectra when stacked. Additionally we want the transmission spectra to be the same for every point of entry along the base of the stack as we are interested in custom filters rather than things like holograms. (Which can also be done using metasurfaces [7]) With this in mind we are going to use a structure which is periodic in the x and y direction and consists of one repeating meta-atom. These kind of surfaces can be manufactured via ... **7.ask jan how these are made.**

As shown in section 3.2 paragraph Symmetries we have to use at least a meta-atom of C_4 symmetry to produce different transmissions for the x and y polarization. The simplest geometry which fulfills this is the rectangle. Another limitation given by the manufacturing process is the number of metasurface layers in a stack. Stacks with more than two layers become increasingly hard to manufacture thats why we are going to limit the algorithm to two layer stacks even though theoretically arbitrary stacks can be calculated via SASA.

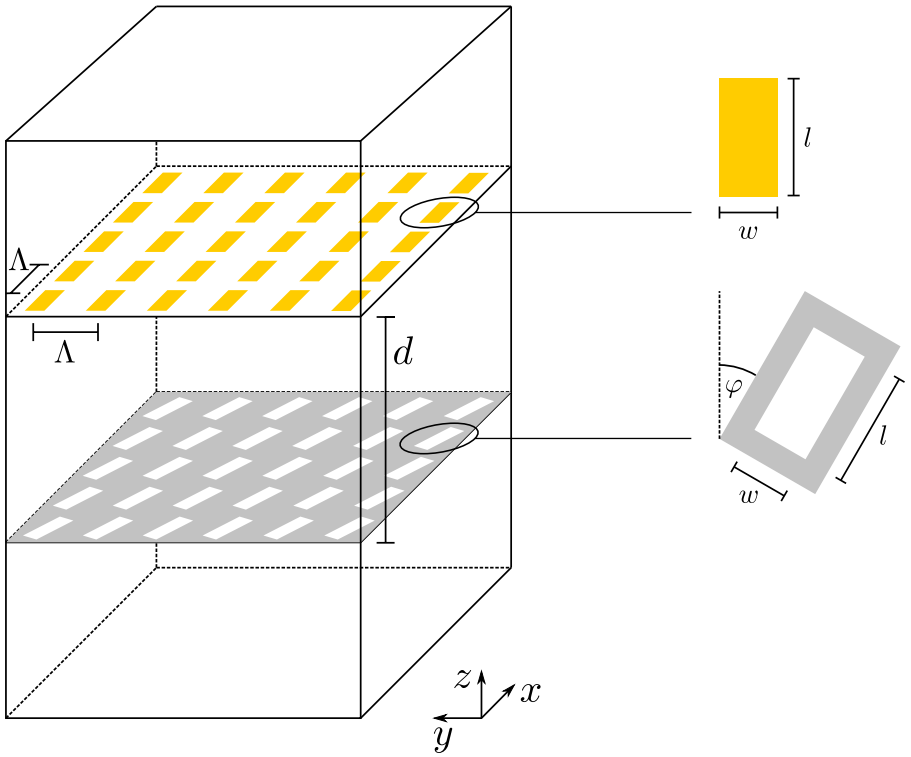


Figure 16: A two layer stack to visualize all the design parameters $\mathcal{D}$ which can be set by the algorithm. These include stack parameters $\mathcal{S}$ and two sets of layer parameters $\mathcal{L}_1$ and $\mathcal{L}_2$. So $\mathcal{D} = (\mathcal{S}, \mathcal{L}_1, \mathcal{L}_2)$. The stack parameters consist of the distance between the metasurfaces d and their rotation angle φ so $\mathcal{S} = (d, \varphi)$. The layer parameters are the width w, length l and thickness t of the meta-atom, the period of meta-atoms Λ and the

The last decision about the metasurfaces geometry is a little heuristic. A large variety of transmission spectra includes some that are very absorbent everywhere except for specific wavelengths and some where almost all wavelengths are transmitted. A metal metasurface absorbs a light when the electrons in the meta-atoms can oscillate in resonance. For rectangular meta-atoms this is the case for very specific wavelengths so metasurfaces made of these produce the former kind of spectrum. The opposite can be achieved when using rectangular holes. These metasurfaces have many different resonance frequencies and only the wavelengths that would have "fit" into the holes are not absorbed. That means the hole geometry can be use to produce the latter kind of spectrum and we have to

include both geometries to reach the greatest variety of transmission spectra.

4.1 Implementation

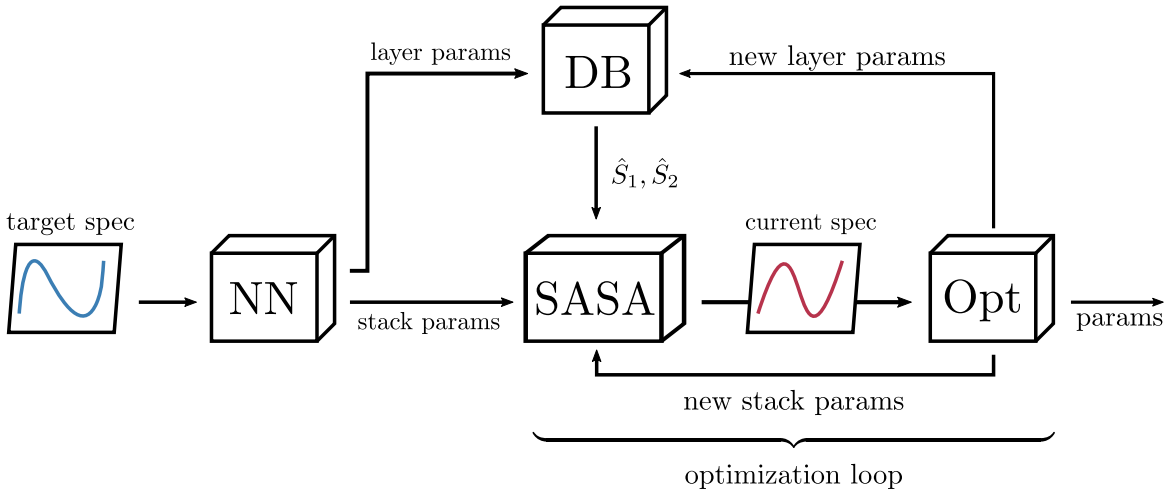


Figure 17: The algorithm tries to find to a given transmission spectrum the kind of two layer meta surface stack which can reproduce this target. The input spectrum is passed to a convolutional neural network which outputs its guess for the stack and layer parameters. The Database module looks at the two sets of layer parameters and interpolates between stored S-matrices to give an estimate for the two S-matrices describing the two layers. The SASA algorithm calculates the resulting current transmission spectrum and passes it to the optimizer. This last module compares it to the target spectrum and adjusts the continuous parameters to minimize the difference between the two. Find the details to all parts of the algorithm in the sections below.

NN:	convolutional neural network trained to map spectra to stack and layer parameters
DB:	database of FMM simulated single layers
SASA:	algorithm calculating $\hat{S}_{\text{stack}} = \hat{S}_{\text{stack}}(\hat{S}_1, \hat{S}_2, \varphi, h)$
Opt:	optimizer changing parameters to minimize the difference between the current and target spectrum
$\hat{S}_1$, $\hat{S}_2$	S-matrices of the top and bottom layer
layer params	two sets of continuous parameters: $p = (w, l, t, \Lambda)$ and discrete parameters: $m \in$ (Al, Au), $g \in$ (rectangles, rectangular holes) w...width, l...length, t...thickness, Λ...Period, m...material, g...geometry
stack params	φ...rotation angle, h...distance between layers
new params	the Opt. only changes the continuous parameters, the discrete ones, e.g. material, remain unchanged
optimization loop	this loop is repeated until the target accuracy is reached

4.2 Network

Input:	Spectrum $I = (I_\mathrm{x}, I_\mathrm{y})$ a $\lambda \times 2$ array $I_{\mathrm{x/y}}$...X- and Y-transmission spectra, λ...number of wavelengths
Output:	two sets layer parameters $p = (w, l, t, \Lambda)$, m, g and stack parameters φ, h w...width, l...length, t...thickness, Λ...Period, m...material, g...geometry φ...rotation angle, h...distance between layers

Network Architecture This module is a 1D Convolutional Neural Network instead of the simple Multi Layer Perceptron. It was chosen to utilize the translational invariance of ConvNets. For example the concept "peak" should be learned independent of its position in the spectrum. As described in section 3.3 a ConvNet provides this functionality. Another constraint on the network architecture arises from the different kind of outputs. $p = (w, l, t, \Lambda)$, φ and h are continuous and m, g are discrete/categorical. These need different activation functions σ to reach the different value ranges. The continuous outputs are mostly bounded by physical constraints and $m, g \in [0, 1]$ as they are *one hot encoded* meaning $1 \rightarrow$ "The layer has this property" and $0 \rightarrow$ "The layer does not have this property".
The different outputs also need different cost functions $C(y, y')$ during training where y' is the networks output and y is the known solution. For the continuous output one can simply use the mean squared error

$$C_\mathrm{mse}(\mathbf{y}, \mathbf{y}') = \sum_i \left(y_i - y_i'\right)^2 \tag{4.48}$$

as all outputs are equally important and the cost function should be indifferent on whether the networks prediction is over or under target. For the categorical output the network learns quicker with the *Categorical Cross-Entropy* error.

$$C_\mathrm{ce}(\mathbf{y}, \mathbf{y}') = -\sum_i y_i \log y_i', \tag{4.49}$$

This error treats false positives and false negatives differently. A false positive does not increase the overall cost as $y_i = 0$ anyway but for a false negative ($y_i = 1$, $y_i' = 0$) $C_\mathrm{ce} \rightarrow \infty$. This is wanted behavior because it does not matter if the network outputs some probability for a wrong class as long as it outputs a higher probability for the correct class. The final architecture is similar to the example given in figure 13 while meeting the above-mentioned constraints:

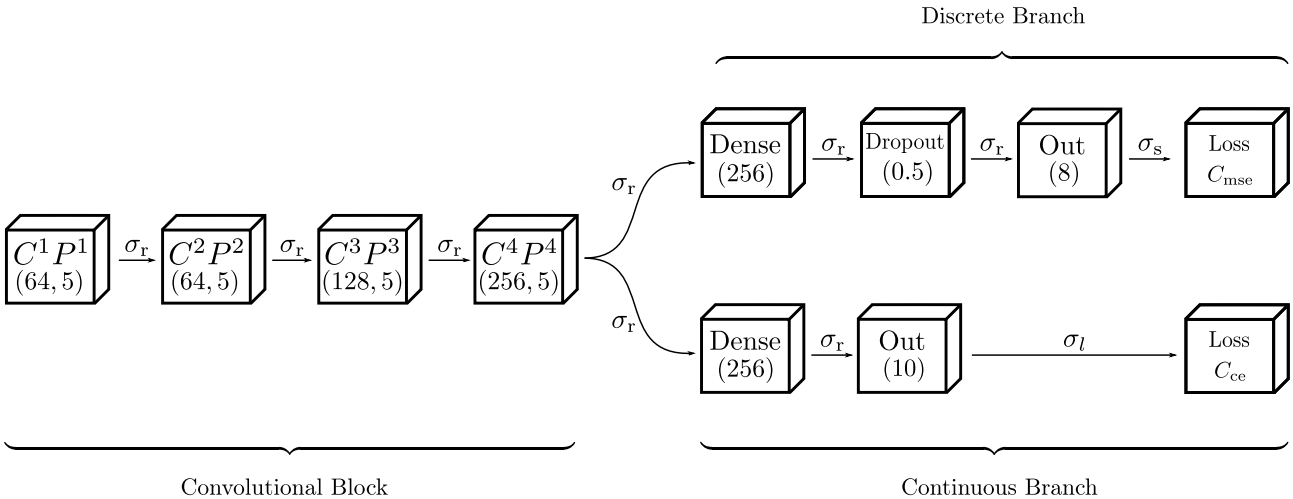
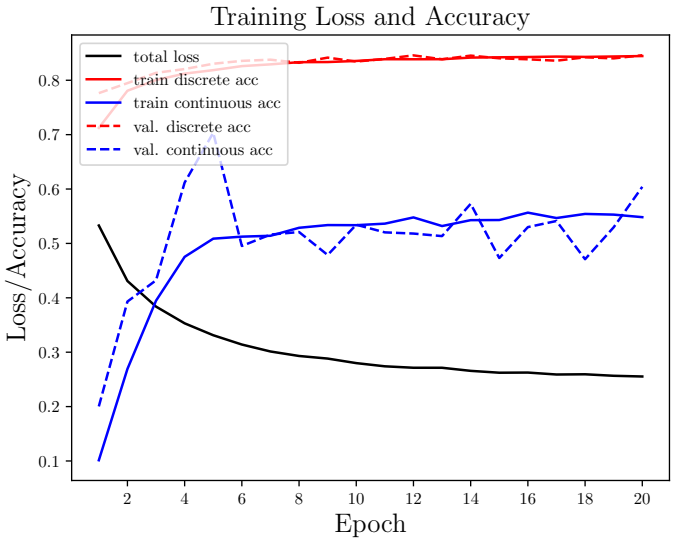
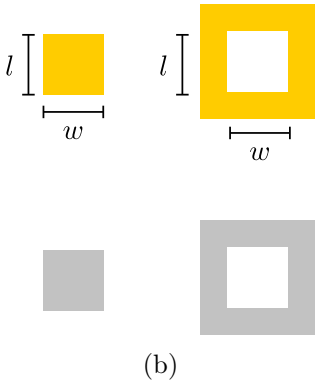


Figure 18: The network starts with 4 pairs of convolutional and pooling layers. The convolutions are characterized by (*number of kernels*, *kernel size*). The kernel size is always 5 and the number of kernels is gradually increased. Then the Network splits into a discrete and a continuous branch via two Dense layers with (*number of neurons*). In the discrete branch a dropout is applied to the dense layer where (0.5) is (*fraction of neurons to drop*). All the internal activations σ_r are ReLu's and the final activations σ_s and σ_l are a sigmoid and a linear function.

Network Training To train a Neural Network one needs a training set (X, Y) of known input output pairs. In this case they are generated using the pre simulated single layers in the database which are randomly combined into a stacks. The single layers are simulated with the Fourier Modal Method (FMM). Then the stacks X- and Y-transmission spectra (I_x, I_y) are calculated via SASA. That means $(I_x, I_y) \in X$ are the networks input and the random parameters $(p_1, m_1, g_1, p_2, m_2, g_2) \in Y$ are the output. For the first geometry we used squares and square holes of Aluminium and Gold as seen in figure Using this approach the following accuracy is reached:



(a)



(b)

Figure 19: Shown are the total loss and the training and validation accuracy for both the continuous and the discrete branch in figure (a). After each epoch the network is validated on data not used in training to check for overfitting. Figure (b) shows the used geometry of squares and square shaped holes of Aluminium and Gold. They have width w, length l and thickness t. In the full layer these meta atoms are arranged periodically as seen in figure 5.

Training and validation accuracy are very similar which indicates that there is no overfitting. The discrete accuracy quickly reaches a maximum of $\sim 80\%$ which is less than expected for this classification problem. The issue here lies not within the networks architecture but in the data generation process. In section 3.2 we have shown that for this geometry the transmission spectrum is the same for both directions. That means the data generation can result in two different stacks which produce the same spectrum. Lets consider a stack where one layer is Aluminium and the other is Gold. As both of them produce the same spectrum one time the network is taught that the first layer is Gold and another time its taught the complete opposite. Actually if the network is trained this way it only ever predicts stacks with layers of equal materials because this is the only setup it can get right.

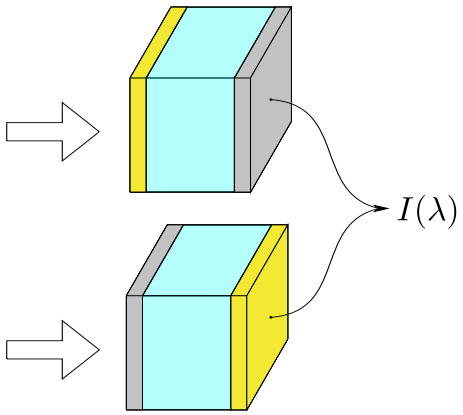


Figure 20: A stack of one Gold and one Aluminium layer separated by a glass spacer. If the layers are made of meta atoms of the square geometry as seen in figure orientations produce the same spectrum which leads to issues when using completely random stacks to train the network.

This is well known problem that arises when training a network on a problem where there are many solutions to one given input. **8.cite stuff** We can prevent this issue simply by allowing only one of the equivalent orientations into the training data. This does not limit the capabilities of the trained network as the variety of input spectra stays the same.

4.3 Database

Input:	layer parameters $p = (w,\, l,\, t,\, \Lambda)$, material m geometry g
Output:	approximation of the S-matrix of this layer $\hat{S} = \hat{S}(p,\, m,\, g)$

The database consist of $\sim$ 5000 S-matrices of single layers which were simulated with the Fourier Modal Method (FMM) on a compute cluster. Its input are the layer parameters width, length, thickness and period, so $p = (w,\, l,\, t,\, \Lambda)$. This module first looks for the n closest neighbors of p. To do that first the input is scaled:

$$\tilde{p}_i = \frac{p_i - p_i^{\min}}{p_i^{\max} - p_i^{\min}} \quad \text{so that} \quad \tilde{p}_i \in [0,\, 1] \tag{4.50}$$

Now the distance d to every entry in the database satisfying the material geometry combination is calculated and the n entries with the smallest distance are selected:

$$d(p,\, q) = \sum_i |p_i - q_i| \tag{4.51}$$

The output $\hat{S}(p)$ is calculated via Inverse Distance Weighting [8] so that more distant entries have a smaller effect. Let $(q_1,\, \ldots,\, q_n)$ be the n closest neighbors to p with stored S-matrices $(\hat{S}_1,\, \ldots,\, \hat{S}_n)$ then:

$$\begin{aligned} \hat{S}(p) = \sum_{j=1}^{n} w_j\, \hat{S}_j \quad \text{where} \quad w_j = \frac{1/d_j^2}{\sum_i 1/d_i^2} \\ \text{so that} \quad \sum_j w_j = 1 \end{aligned} \tag{4.52}$$

To have this interpolated approximation $\hat{S}(p)$ be close to the result of rigoros simulation FMM(p) the simulated grid of parameters needs to be sufficiently dense. **9.Insert some calculation of what is sufficiently dense**

4.4 Optimizer

Input:	current spectrum I a $\lvert\lambda\rvert \times 2$ Array,
	current continuous parameters $p = (w, l, t, \Lambda, h, \varphi)$
	h...spacer height, φ...layer rotation
Output:	improved continuous parameters $\tilde{p}$

The optimizer is at the core a Downhill-Simplex [9] minimizing the mean-squared-difference between the current spectrum I_c and target spectrum I_t so it minimizes $C_\mathrm{mse}(I_\mathrm{c}(p),\, I_\mathrm{t})$ This standard method is however unable to follow the constraints and has to be modified in that regard. To achieve this one can introduce a distance to the boundary D. Let p be a single parameter with lower bound p^l and upper bound p^u, then:

$$D\Big(p,\, p^l,\, p^u\Big) = \begin{cases} p^l - p, & \text{for } p < p^l \\ 0, & \text{for } p^l \leq p \leq p^u \\ p - p^u, & \text{for } p^u < p \end{cases} \tag{4.53}$$

In this way one can penalize the simplex for stepping over the set boundaries by using a total loss L which depends on the sum of all distances D_i:

$$L(I_\mathrm{c},\, I_\mathrm{t},\, p) = \underbrace{C_\mathrm{mse}(I_\mathrm{c},\, I_\mathrm{t})}_{\text{find target}} + \underbrace{\left[\sum_i D\Big(p_i,\, p_i^l,\, p_i^u\Big)\right]^2}_{\text{stay within bounds}} \tag{4.54}$$

10.align underbraces

The choice of power depends on how much the simplex should be penalized for stepping over a boundary. All our conditions are requirements for physical approximations and

11.Maybe 3D plot of an example loss function

5 Junk

A bunch of collected junk

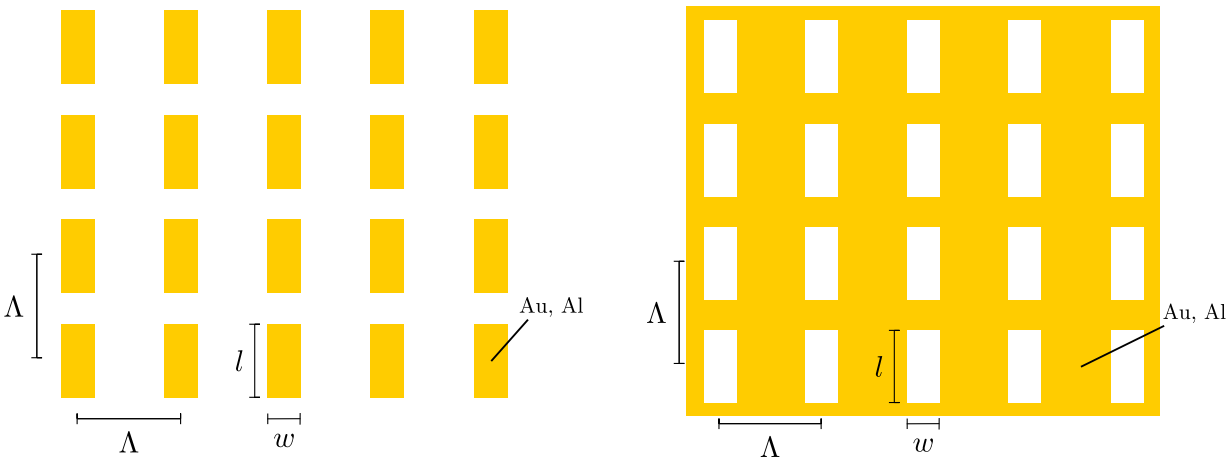
5.1 Meta Surfaces

Definition

Stacks

Geometry

The idea behind choosing a type of meta surface was: To use a simple and easy to manufacture geometry and achieve complex transmission spectra by stacking these simple layers on top of each other. It is essential to use particles of at least C^4 rotational symmetry to enable the layer to affect x and y polarizations differently. A fitting geometry are rectangular meta surface particles.



(a) rectangles

(b) rectangular holes

Figure 21: The rectangular particles of width w and length l are arranged on a square matrix with Periode Λ in x and y direction. They are made of Gold or Aluminium. The second geometry (b) is inverse to the first in that wherever there was material now light can transmit freely. Both layers also have thickness t.

5.2 S-Matrix Calculus

Jones Formalism A planar lightwave propagating along the z axis trough a homogeneous material can be described as:

$$\mathbf{E} = \begin{pmatrix} E_x e^{i(kz-\omega t+\varphi_x)} \\ E_y e^{i(kz-\omega t+\varphi_y)} \\ 0 \end{pmatrix} = \left(E_x\, e^{i\varphi_x}\, \vec{\mathbf{e_x}} + E_y\, e^{i\varphi_y}\, \vec{\mathbf{e_y}}\right) e^{i(kz-\omega t)} \tag{5.55}$$

the waves polarization is determined by the scaling factors of $\vec{\mathbf{e_x}}$ and $\vec{\mathbf{e_y}}$ and can be expressed as a *Jones Vector* $\mathbf{j} \in \mathbb{C}^2$.

$$\mathbf{j} = \frac{1}{\sqrt{E_x^2 + E_y^2}} \begin{pmatrix} E_x \\ E_y\, e^{i\delta} \end{pmatrix} \quad \text{with} \quad \delta := \varphi_y - \varphi_x \tag{5.56}$$

Now all linear operations on the polarization are matrices $\hat{M} \in \mathbb{C}^{2\times 2}$. That means all passive components have a corresponding matrix. A couple examples for components in horizontal position:

$$\begin{aligned} \text{polarizer:} \quad & \hat{M} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ \lambda/4 \text{ plate:} \quad & \hat{M} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} e^{-\frac{i\pi}{4}} \\ \lambda/2 \text{ plate:} \quad & \hat{M} = \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \end{aligned} \tag{5.57}$$

By using matrices one can easily compute the effect of multiple components using the matrix product and find the behavior of a rotated component $\hat{M}_\varphi$ using the standard rotation matrix.

$$\hat{M}_\varphi = \hat{R}(-\varphi)\, \hat{M}\, \hat{R}(\varphi) \quad \text{where} \quad \hat{R}(\varphi) = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \tag{5.58}$$

SASA

$$\hat{S} = \begin{pmatrix} \hat{T}^f & \hat{R}^b \\ \hat{R}^f & \hat{T}^b \end{pmatrix} \tag{5.59}$$

12.How do symmetries in the MS result in S-Matrix symmetries.

Applying Jones calculus to MS

13.Here: Under which conditions can one describe the effect of a MS with a jones matrix?

Boundary Conditions

14.Maybe mark equations used as boundary conditions for the algorithm differently(e.g. fat)

5.3 SASA

Here I want -Intuition for single Layers: how does the transmission change for different Λ, l, w, ϕ -Why is the S matrix needed (Fabry-Perot-Effects, ...)

6 Sources

References

[1] R. A. Shelby. Experimental verification of a negative index of refraction. *Science*, 292(5514):77–79, apr 2001.

[2] Nanfang Yu and Federico Capasso. Flat optics with designer metasurfaces. *Nature Materials*, 13(2):139–150, jan 2014.

[3] C. Menzel, J. Sperrhake, and T. Pertsch. Efficient treatment of stacked metasurfaces for optimizing and enhancing the range of accessible optical functionalities. *Physical Review A*, 93(6), jun 2016.

[4] R. M. Redheffer. On a certain linear fractional transformation. *Journal of Mathematics and Physics*, 39(1-4):269–286, apr 1960.

[5] Michael Nielsen. http://neuralnetworksanddeeplearning.com/chap2.html, December 2019.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan 2014.

[7] J. P. Balthasar Mueller, Noah A. Rubin, Robert C. Devlin, Benedikt Groever, and Federico Capasso. Metasurface polarization optics: Independent phase control of arbitrary orthogonal states of polarization. *Physical Review Letters*, 118(11), mar 2017.

[8] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference on -*. ACM Press, 1968.

[9] R. Mead J. A. Nelder. A simplex method for function minimization. *The Computer Journal*, 8(1):27–27, apr 1965.