Machine learning of interference patterns

Igor Chełstowski, Michał Karny, Karol Łukanowski, Marcin Pruszczyk
under supervision of prof. dr hab. Jakub Tworzydło

March 4, 2022

1 Goal

The main goal of the project is to learn about the basics of machine learning and its application to simple physical setups. Also, to implement these ideas, building a simple neural networks in Python. We will focus on the implementation in the library called Keras. Additionally, we aim to improve the skills required for an efficient cooperation between members of a scientific group, such as project management, the ability to present obtained results to other members of the group, and the ability to use digital repositories (such as github) to share the developed code.

The physical system of choice is a one-dimensional model of interference. We consider a plane wave of light entering a system of flat mirrors of considerably small thickness. The mirrors are parallel to each other and the wave-vector of the plane wave is perpendicular to their surfaces. We seek a theoretical description of the dependence of the intensity of light escaping the system of mirrors on the wave vector value of the light entering "black box", in which the mirrors are placed. First, we assume that the configuration of the mirrors within the "black box" is known, and from simple theoretical calculations we derive analytically the sought dependence. Then, we teach a neural network to guess the configuration of the mirrors in the "black box" based on plots of transmittance as a function of the wave vector.

2 Machine Learning

As machine learning is a widely discussed topic, both in the scientific community and in the world of economics, there is plenty of literature and video tutorials on it. The description of theoretical concepts behind neural networks lays beyond the scope of this report. However, we list out some of the sources that enabled us to gain sufficient background to obtain the results presented in the latter part of the report.

Written materials

- Introduction to Keras, a Python library for constructing neural networks

Video tutorials

- "But what is a neural network?" from 3Blue1Brown YouTube channel
- "Introducing convolutional neural networks (ML Zero to Hero - Part 3)" from TensorFlow YouTube channel
- prof. Żygierewicz class on machine learning - neural networks (in Polish)
- prof. Żygierewicz class on machine learning - convolutional networks (in Polish)

3 Theoretical model

The results presented in this part of the text were derived by Igor Chełstowski.

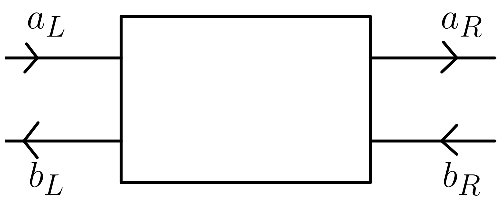

Figure 1: A schematic figure of a "black box". The light enters it from the left with amplitude a_L and with amplitude b_R from right. It also escapes the box to the left with amplitude b_L and to the right with amplitude a_R.

We look for a procedure allowing to formulate relations between the amplitudes of light entering the optical system and amplitudes of light leaving the system.
To do so, we introduce the transfer matrix:

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}, \tag{1}$$

which relates the amplitude of the light entering from the left a_L and leaving to the left b_L with the amplitudes of light leaving to the right a_R and entering from the right b_R in a linear manner

$$\begin{bmatrix} b_R \\ a_R \end{bmatrix} = \mathbf{M} \begin{bmatrix} a_L \\ b_L . \end{bmatrix} \tag{2}$$

This of course allows to express b_L and b_R by a_L and a_R by equations:

$$\begin{cases} b_L = m_{22}^{-1} \left(a_R - m_{21} a_L \right), \\ b_R = m_{11} a_L + m_{12} m_{22}^{-1} \left(a_R - m_{21} a_L \right). \end{cases} \tag{3}$$

In this formalism, propagation through empty space on distance L of light with wave vector k is described by

$$\mathbf{M} = \begin{bmatrix} e^{ikL} & 0 \\ 0 & e^{-ikL} \end{bmatrix}. \tag{4}$$

For a $50:50$ mirror, the transfer matrix is given by:

$$\mathbf{M} = \begin{bmatrix} \sqrt{2} & -1 \\ -1 & \sqrt{2} \end{bmatrix}. \tag{5}$$

As we are considering a one-dimensional system, the wave vector is a single scalar of dimension of inverse length.
We also want to introduce the scattering matrix

$$\mathbf{S} = \begin{bmatrix} r & t' \\ t & r' \end{bmatrix} \tag{6}$$

such that

$$\begin{bmatrix} b_L \\ b_R \end{bmatrix} = \mathbf{S} \begin{bmatrix} a_L \\ a_R \end{bmatrix}. \tag{7}$$

Its matrix elements may be expressed by the matrix elements of the transfer matrix

$$\begin{cases} r = -m_{22}^{-1} m_{21}, & r' = m_{12} m_{22}^{-1}, \\ t = m_{11} - m_{12} m_{22}^{-1} m_{21}, & t' = m_{22}^{-1}. \end{cases} \tag{8}$$

This allows for computationally efficient formulation - one may overwrite the previous values of variables with new ones, according to

$$\begin{cases} t' = m_{22}^{-1}, \\ r = -t' m_{21}, \\ t = m_{11} + m_{12} r, \\ r' = m_{12} t'. \end{cases} \tag{9}$$

Such a procedure consist of one inversion and three matrix multiplications.

As we consider more complicated systems, such that are composed of many subsystems with well-defined transfer matrices, we look for composition rules - a way to express the action of a complex optical system given that the action of the single subsystems is known.

For a system composed of two subsystems, the compostion rule for the transfer matrices is given by

$$\mathbf{M} = \mathbf{M_2} \cdot \mathbf{M_1}. \tag{10}$$

However, such an operation is numerically highly unstable.
The composition rule for the scattering matrices $\mathbf{S}$ is given by

$$\begin{cases} r = r_1 + t_1' \left(1 - r_2 r_1'\right)^{-1} r_2 t_1, \\ r' = r_2' + t_2 \left(1 - r_1' r_2\right)^{-1} r_1' t_2', \\ t = t_2 \left(1 - r_1' r_2\right)^{-1} t_1, \\ t' = t_1' \left(1 - r_2 r_1'\right) t_2', \end{cases} \tag{11}$$

where r_1, r_1', t_1, t_1' are the matrix elements of the matrix $\mathbf{S}_1$, which is on the left, and r_2, r_2', t_2, t_2' are the matrix elements of the matrix $\mathbf{S}_2$, which is on the right. It turns out to be way more numerically efficient and stable than the composition rule for the transfer matrices.

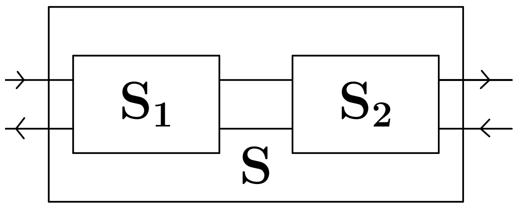

Figure 2: A sketch representing the idea of a complex optical systems. Composition rules allow to treat the action of two scattering matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ as an action of one scattering matrix $\mathbf{S}$.

The quantity relevant from the experimental point of view and for the discussion of the neural network is the transmittance T, given by the formula

$$T = |t'|^2. \tag{12}$$

It measures how much of the intensity of the light is transmitted outside the black box.

4 Realization of the model

4.1 Formulation

The first implementation of the problem is such: we consider one possible type of mirrors with their reflection r, r' and transmission t, t'. We consider a ladder construction of the interferometer - the mirrors may be placed in one of n possible sites. The sites are placed in a fixed distance between each other. We put into the "black box" different numbers of mirrors, from i_{min} to i_{max}. This gives the total number Ω of considered k $\in [0, 2\pi/\text{unit}]$ mirror configurations

$$\Omega = \sum_{i=i_{min}}^{i_{max}} \binom{n}{i}. \tag{13}$$

Using the theory derived in the previous section, we wrote a code in Python calculating, for every mirror configuration, the action of the composite system for a given value of wave vector k. Of course, in such a case one has to take into account the propagation of the light inside the black box. The outcome of such computations are graphs of transmittance as a function of the wave vector k. Such a graph is presented in Figure 3. The plots are for a black box with 20 ladder slots for 50:50 mirrors placed in such a way that the distance between the neighboring ones was 1 in chosen units. The red one corresponds to three mirrors placed at the sites of number 0, 1 and 3, the green one to six mirrors placed at sites 0, 2, 3, 7, 8, 9 and the blue one corresponds to ten mirrors placed at 0, 1, 2, 5, 7, 8, 11, 15, 16, 20. The plots are for $k \in [0, 2\pi/\text{unit}]$

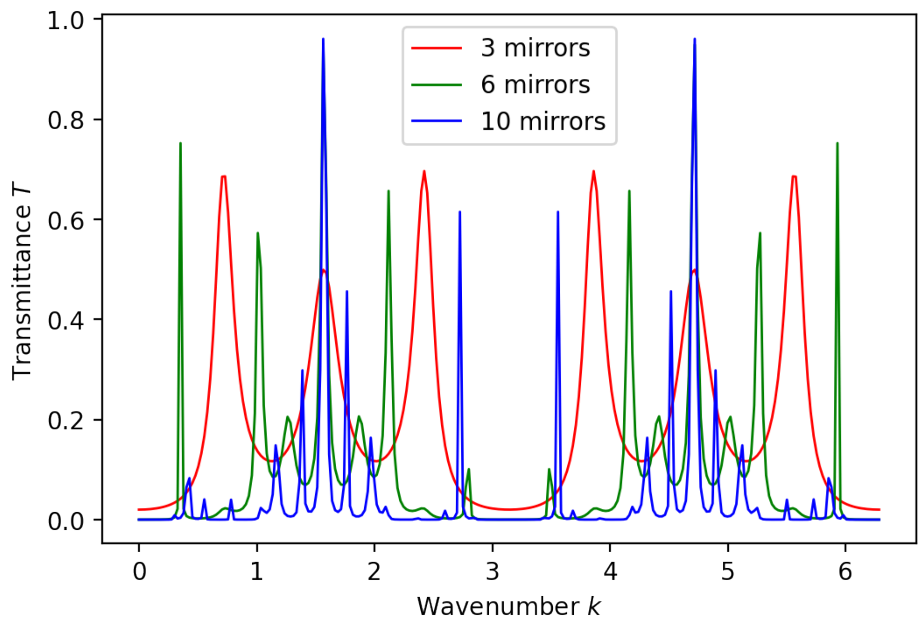

Figure 3: Plots of transmittance as a function of the wavenumber (wave vector) k. The red plot corresponds to three mirrors placed at the sites of number 0, 1 and 3, the green plot to six mirrors placed at sites 0, 2, 3, 7, 8, 9 and the blue one corresponds to ten mirrors placed at 0, 1, 2, 5, 7, 8, 11, 15, 16, 20. The plots are for $k \in [0, 2\pi/\text{unit}]$

Now we set the number of "ladder slots" n and the minimal and maximal number of mirrors, i_{min} and i_{max}. We want to train our neural network in such a way that it is able to guess the number of mirrors inside the black box. And so, the classes for our neural network are labeled by the number of mirrors i. In the class labeled by the number i, there are $\binom{n}{i}$ corresponding configurations.

4.2 Performance metrics of the neural network

We want to investigate whether or not the neural network performs at a sufficiently high level, i.e. if it is able to guess the number of mirrors inside the black box based on a given graph of trasmittance as a function of the value of the wave vector. In order to do so, we introduce performance metrics. Each of them is evaluated for the neural network in the following manner: first, using a program written in Python, we generate a set of input data. In our case it is a set of values of transittance as a function of the wavelength for different mirror combinations in the black box. Afterwards, we introduce labels for the data - the quantity that we want the neural network to guess. In this setup the graphs are labeled by the number of mirrors inside the black box. Then we divide randomly the input data into two sets - a training set and a validation set. The training set is used to tune the parameters of the neural network so that they fit into the input data from the set. It is as if the neural network was preparing for an exam by checking the answers for problems in a problem book. The validation set is used to verify how well it has tuned itself. The neural network is given a graph of transmittanceit never saw before and it guesses the number of mirrors. Then its answer is compared with the correct answer encoded in the input data. Such a sequence of going through all of the input data, starting with division into the training and validation set and ending with comparing the answers of the neural network with the correct ones in the validation data is called an epoch. During each epoch the whole set of input data is ran through.
During one epoch one may calculate the following performance metrics:
precision - the number of correctly guessed members of a given class divided by the number of guesses that a graph corresponds to that class.
recall - the number correctly guessed members of the class divided by the number all the members of the class. **accuracy** - the sum of the number of correctly guessed members of the class with the number of correctly dismissed configurations, i.e. the number of configurations correctly guessed not to be in the given class, divided by all the configurations.
f1-score - the harmonic mean of precision and recall

$$\text{f1} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}. \tag{14}$$

To get a grasp of these concepts, it is helpful to represent them graphically, as in Figure 4.
These performance metrics are defined for a single class. However, one may calculate values of macroscopic performance metrics to evaluate the quality of the neural network as a whole. This is done by properly averaging the basic performance metrics over the classes. Such macroscopic metrics are:
Overall accuracy - the ratio of the number of correctly guessed labels to the number of total guesses.
Macro averaged metrics - arithmetic mean of the metric's values from all the classes.

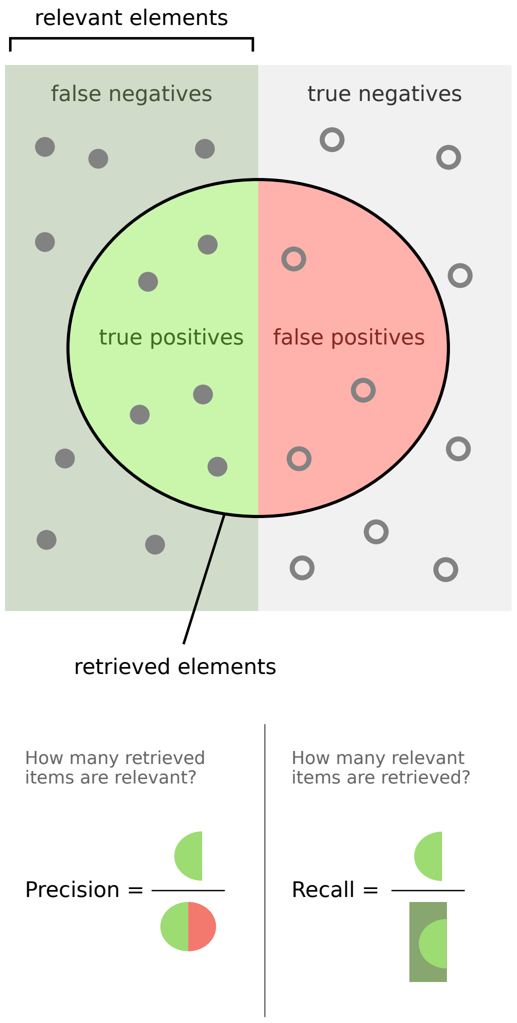

Figure 4: A sketch illustrating the concepts of two performance metrics: accuracy and recall. In the considered setup, for a given label, true positives are correctly guessed labels, false positives are guesses of the considered label assigned to configurations not belonging to the class, false negatives are incorrect assignments of an element of the considered class to a different label and true negatives are correctly not assigning an element of a different class to the considered class. Figure taken from https://en.wikipedia.org/wiki/Precision_and_recall

5 Results

We trained neural networks implemented in the Python library Keras to guess the number of mirrors. We started off with the simplest possible setup and then came up with modifications of that setup to understand better what actually is happening in the neural network and to work out some of the arising problems manifesting themselves in very high or very low values of performance metrics of the neural network.

5.1 First Setup

We started off with the implementation of the black box with 20 ladder slots. We considered configurations in which we had from 3 up to 10 50:50 mirrors. The 20 ladder slots were placed in such a way that the distance between the neighboring ones is 1 in chosen units. We considered graphs of transmittance with the wave vectors k in the interval $k \in [0, 2\pi/\text{unit}]$. At this point we did not manipulate the input data in any way. The input data were of the same form as plotted in Figure 3.

The corresponding plots of the relevant performance metrics for the setup are plotted in Figure 5. It is worth noting that in this figure and in Figures 6 and 7 he low-opacity colored lines correspond to the true values of the measured metrics, whereas their high-opacity counterparts result from applying a smoothing operation to make the plot easier to read. One sees that the overall accuracy has a very large value, it is of the order of 80%. Also one notices that the performance metrics take extremely large values for the classes of configurations with the number of mirrors from 7 to 10. The large values are also obtained extremely quickly. This caused our concern, as it may hint that the neural network did not learn based on the shapes of the transmittance graphs, but rather based on the total transmittance. One may of course intuitively predict that the configurations with larger number of mirrors should transmit less light. One easily notices that the performance metrics have much lower values for classes of configurations with a lower number of mirrors, i.e. for configurations with 5 or less mirrors. This of course causes the macro averaged metrics to have relatively low values, $\approx 50\%$. Also, the performance metrics for these classes decrease with each epoch. This might indicate that the neural network learns not to guess the configurations with less than 5 mirrors. This is sensible since the classes do not have the same number of members. In the class of 10-mirror configurations, one has $\binom{20}{10} = 184756$ elements. In the class of 3-mirror configurations, one has $\binom{20}{3} = 1140$ elements, so over 160 times less elements. whci means that the elements of the 3-mirror class is a tiny fraction fraction of the number of all of the configurations, more precisely it is 0.19% of the number of all the configurations. This means that without any learning it is much better to simply guess for the classes with larger numbers of mirrors and to completely dismiss the classes with low numbers of mirrors.

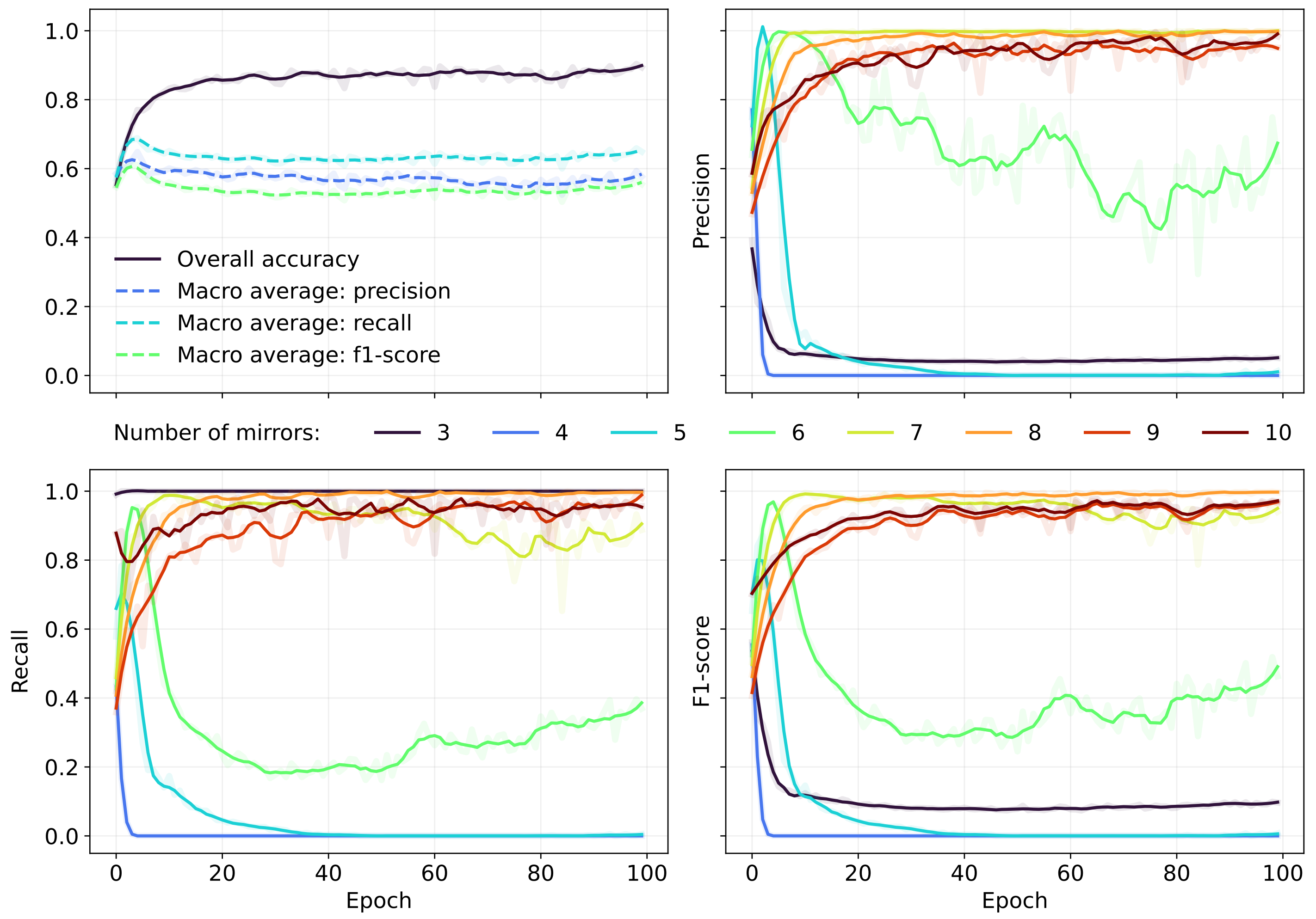

Figure 5: Plot of performance metrics as a function of consecutive epochs for the system with 20 ladder slots and the number of mirrors form 3 to 10. Input data unnormalized. The low-opacity colored lines correspond to the true values of the measured metrics, whereas their high-opacity counterparts result from applying a smoothing operation to make the plot easier to read.

5.2 Normalized input data

We want our neural network to learn to guess the number of mirrors based on the shape of the transmittance graphs rather than based on the total transmittance. This has a simple physical motivation, as it is sometimes very hard to have a light source producing light of very precise constant value of intensity for all measurements. In order to minimize the effect of the the "total transmittance dependence on the number of mirrors", we normalize the transmittance graphs for every configuration. We do that by rescaling the values of the transmittance by dividing it by the maximal value $T(k)$ in the interval $k \in [0, 2\pi/\text{unit}]$, which we denote by T_{max}.
And so we plot

$$T_{norm}(k) = T(k)/T_{max}. \tag{15}$$

Due to such a normalization, one has $T_{norm}(k)$ taking the maximal value 1 somewhere on the interval $k \in [0, 2\pi/\text{unit}]$ for each configuration. This of course makes it impossible for the neural network to learn how to assign the configurations to the classes simply by learning by heart the maximal values of the transmittance.
An example of such a normalized graph of transmittance is presented in Figure 6. The red plot corresponds to three mirrors placed at the sites of number 0, 1 and 3, the green one to six mirrors placed at sites 0, 2, 3, 7, 8, 9 and the blue one corresponds to ten mirrors placed at 0, 1, 2, 5, 7, 8, 11, 15, 16, 20. The plots are for $k \in [0, 2\pi/\text{unit}]$.

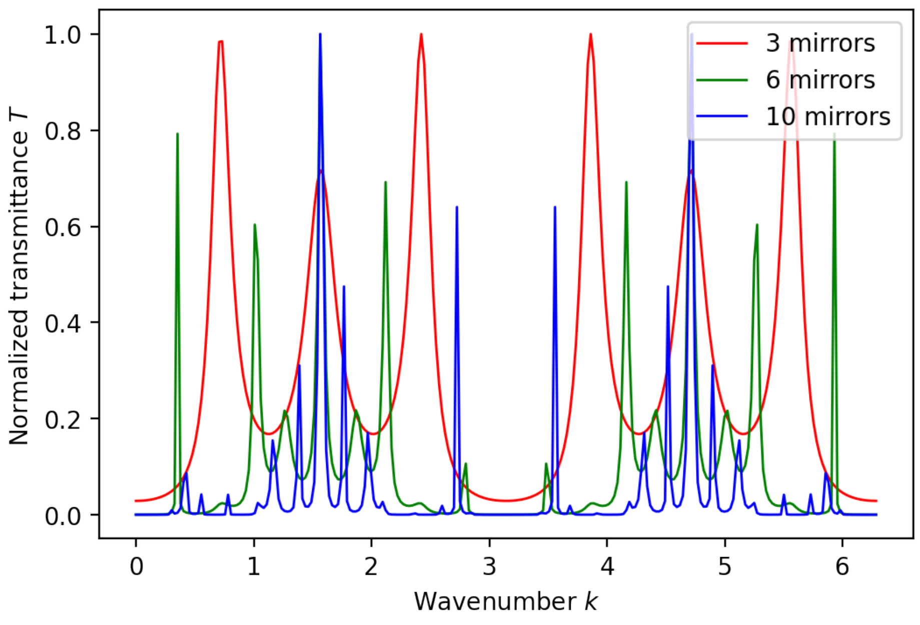

Figure 6: Plots of normalized transmittance as a function of the wavenumber (wave vector) k. The input data have been normalized according to formula (15). The red plot corresponds to three mirrors placed at the sites of number 0, 1 and 3, the green plot to six mirrors placed at sites 0, 2, 3, 7, 8, 9 and the blue one corresponds to ten mirrors placed at 0, 1, 2, 5, 7, 8, 11, 15, 16, 20. The plots are for $k \in [0, 2\pi/\text{unit}]$

The plots of the performance metrics as functions of consecutive epochs are presented in Figure 7. One sees that the plots are a bit more irregular compared to the ones obtained for the setup in which the input data were not normalized. Also, it seems that the neural network was learning more slowly, the value of the overall accuracy is a bit lower too. This shows that we have in fact eliminate the effect of "total transmittance dependence on the number of mirrors", at least to some extent. However, we still see that for later epochs, the performance metrics take extremely low values for classes of configurations with low numbers of mirrors.

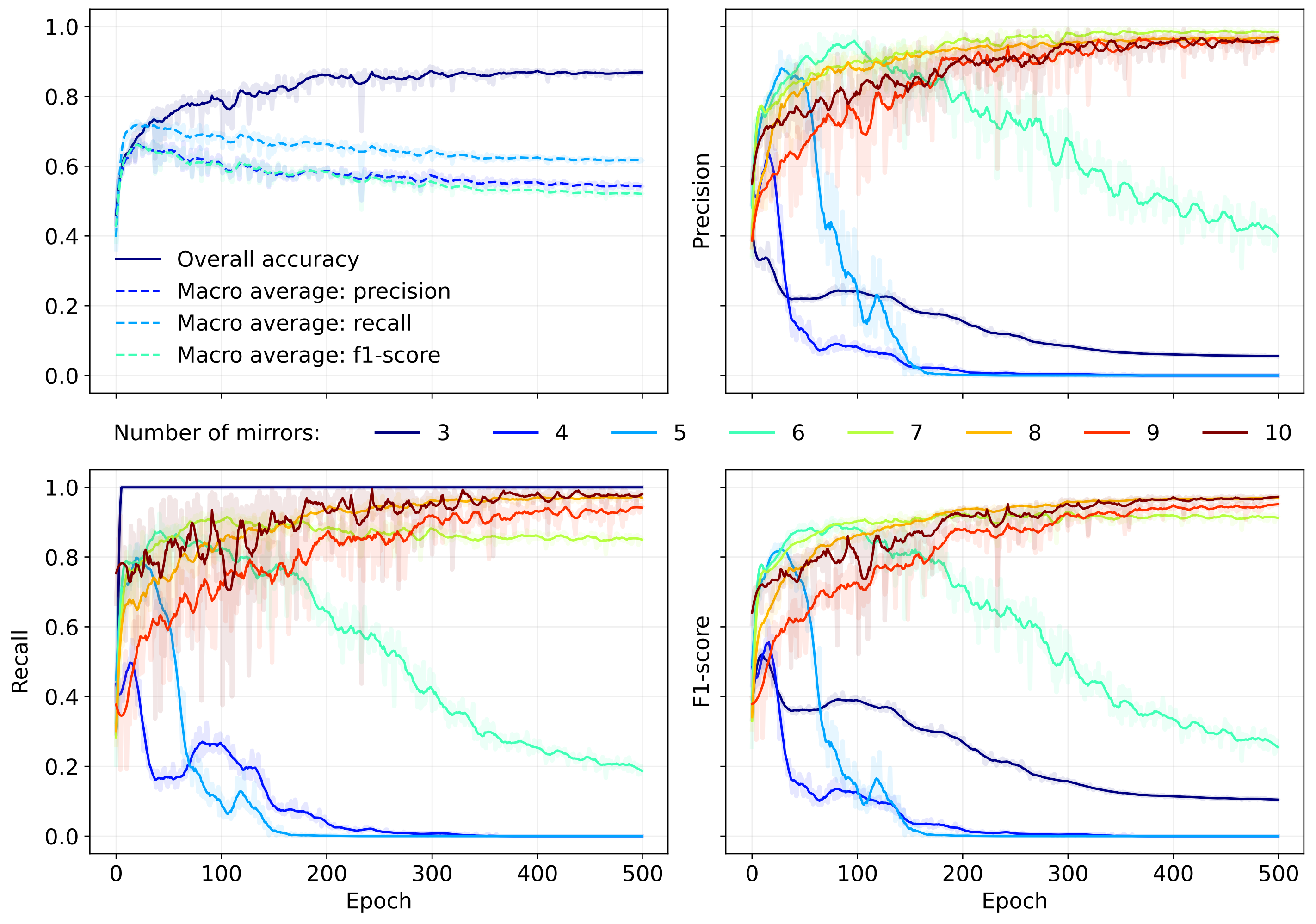

Figure 7: Plot of performance metrics as a function of consecutive epochs for the system with 20 ladder slots and the number of mirrors form 3 to 10. Input data normalized according to equation (15).The low-opacity colored lines correspond to the true values of the measured metrics, whereas their high-opacity counterparts result from applying a smoothing operation to make the plot easier to read.

5.3 Choice of number of mirrors

In order to tackle the problem of the neural network not guessing the labels of classes with low numbers of mirrors, we change the setup. Once again we have a black box with ladder sites spread in such a way that the neighboring sites are at a distance of 1 unit from each other. We put 50:50 mirrors on the ladder sites. This time however we consider 40 ladder sites and configurations with the number of mirrors from 6 to 20. Just as in the preceding paragraph, the input data was normalized using the formula (15).

The plot of the performance metrics as a function of consecutive epochs is presented in the Figure 8. First thing to notice is that the neural network is learning considerably more slowly for this setup in comparison with the setup with 20 ladder sites - the plots for the first to setups were for the first 500 epochs of the network learning, the third one is for the first 1500 epochs. Moreover, the value of the overall accuracy is much smaller - it is of the order 60%, for 20 ladder sites it was around 80%. This is of course to be expected, as the task is harder - there are more classes for the configurations to be assigned to. Interestingly, this time the macro averaged performance metrics coincide with the value of the overall accuracy. This indicates that this time the classes are treated equivalently - none of the classes seem to be disregarded. This notion may be supported also by the observation that the values of the performance metrics systematically increase for every class. This means that the pathological behavior from previous setup is no longer to be observed. One also notices that this timeit is easier for the neural network to correctly assign the labels to configuration with a low number of mirrors.

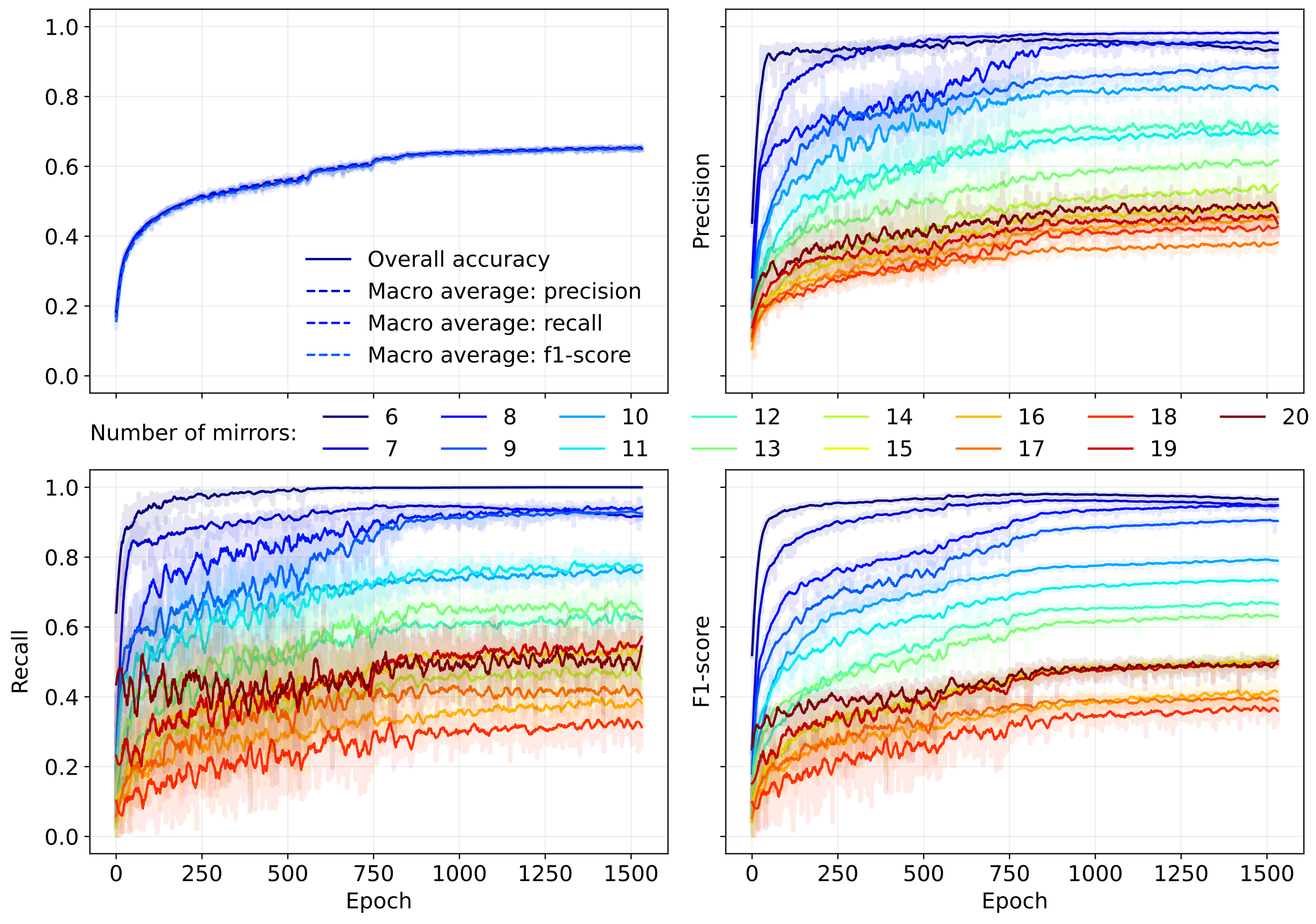

Figure 8: Plot of performance metrics as a function of consecutive epochs for the system with 40 ladder slots and the number of mirrors form 6 to 20. Input data normalized according to equation (15). The low-opacity colored lines correspond to the true values of the measured metrics, whereas their high-opacity counterparts result from applying a smoothing operation to make the plot easier to read.

6 Outlook

Another model is such that on each discrete site in the black box, placed in the same distance from the neighboring ones, we put a one mirror, but this time we have two types of mirrors, with different transmission and reflection coefficients. In this case the total number Ω of considered mirror configurations is

$$\Omega = 2^n, \tag{16}$$

where n is the number of ladder sites. Of course other models are possible - many different types of mirrors, irregularly placed ladder sites. One could also consider a setup in which the light entering the black box is not monochromatic, there is some noise on the laser producing the light and so on. Nevertheless, the considered model, despite being a very simple toy model, gives enough of a physical context to be the starting point of discussion on applications of machine-learning in physical sciences.