universität innsbruck

Master Thesis

Solving inverse problems of many-body physics with neural networks

Stefan Obernauer

October 27, 2022

Supervised by Ass.-Prof. Dr. Mathias S. Scheurer
Institute for Theoretical Physics

Abstract

Over the past few years machine learning techniques have found increasing influence as a numerical method in many physical problems. What stands out as a main advantage of those techniques is the ability to analyze large amounts of data which can no longer be examined by conventional methods. An interesting research area here are inverse problems, where machine learning algorithms can be used for reconstructing parameters from an existing dataset. An example of such datasets in condensed matter physics are scanning tunneling microscope images. These images can provide a wide range of information about the underlying system and common neural network architectures, which are widely used for image classification tasks, can help to shed light on the essential parameters involved.

In this thesis we study the ability of deep neural networks to reconstruct parameters of the effective Hamiltonian from scanning tunneling microscope images. For this purpose we define a minimal tight-binding model for twisted bilayer graphene with a nematic phase, which provides the basis for the artificial generation of the images. We can show that a convolutional neural network, which examines STM images at only one energy, is not sufficient to make accurate predictions about the underlying nematic parameters. However, a network architecture examining the STM images at four different energies allows significantly improved statements about the nematic structure. Finally we show that the best way to train the neural network is to feed scaleograms of the local density of states at one lattice site. These scaleograms alone are sufficient to enable a high accuracy in the predictions of the nematic parameters.

Danksagung

An erster Stelle möchte ich mich herzlich bei meinen Eltern und meiner Schwester bedanken. Ohne deren stetige Unterstützung wäre ich als Physikstudent und vor allem als Mensch niemals so weit gekommen. Sie haben mich davor bewahrt, stressige Zeiten unnötig ernst zu nehmen und es mir ermöglicht, meinen Blick stets auf das Wichtige im Leben zu richten.

Ein besonderer Dank gilt meinem Betreuer Ass.-Prof. Dr. Mathias S. Scheurer, welcher dieses interessante Thema vorgeschlagen hat und mir zu jeder Zeit mit seinem Rat zur Seite stand.

Ebenso möchte ich den Mitgliedern der Arbeitsgruppe danken, welche mich offen in ihre Runde aufgenommen haben. Mein besonderer Dank gilt hierbei João Augusto Sobral, welcher mir bei Fragen zu den theoretischen Themen geholfen hat. Eine große Hilfe waren ebenfalls meine Mitstudenten und Bürokollegen Jakob Wessling und Michael Perle, mit denen ich stets auf erbauliche Weise über Physik und die Welt reden konnte.

Contents

1 Introduction

In recent years machine learning techniques have become increasingly important in business and science. This is mainly due to the fact that great progress has been made in image recognition and natural language processing and more and more companies are resorting to the analysis of large amounts of data through machine learning algorithms. It is therefore only natural to ask whether machine learning could also be used as a solution to physical problems. In condensed matter physics in particular, which has always relied on numerical simulations due to its high-dimensional problems, machine learning seems to be a suitable candidate. In fact, numerous papers have been published in recent years that deal with the possibilities of machine learning applied to condensed matter physics [1–3]. There is notable research in the analysis of images of scanning tunneling microscopes (STM), which are an experimental possibility to collect large and complex amounts of data about material properties and thus prove to be a field for machine learning activities [4–8].

In this thesis, STM images are the data type to be examined by the machine learning technique of deep neural networks (DNNs). However, no experimental STM images are used here, as the images are generated numerically on the computer. This methodology makes it possible to train a neural network for a wide variety of realizations of a physical model rather quickly and thus to derive favorable neural network architectures for each of these realizations. A neural network trained on artificial STM images could then be used to examine experimental images. However, this step was omitted in this thesis, since the physical model used is optimized to generate huge datasets quickly and thus is not suitable for a generalization onto experimental data.

The model in question is a toy model for twisted bilayer graphene (TBG) introduced by my supervisor and it resembles the most important characteristics of the graphene system. This tight-binding model is based on a Bloch Hamiltonian $h_{\mathbf{k}}[\{t_j\}]$, which depends on several parameters t_j and is the key ingredient for the artificial generation of the STM images. Several quite unexpected phases have been proposed to be discovered in TBG, one of them being a nematic phase. Such a phase corresponds to an experimentally observed rotational symmetry breaking in the STM images, while translational symmetry is preserved [9]. The rotational symmetry breaking is a feature of the STM image that has proven to be learnable from a neural network [5]. Hence we decided to add a nematic coupling $h_{\mathbf{k}} \to h_{\mathbf{k}}[\{t_j\}] + \Delta h_{\mathbf{k}}[\{\alpha_j\}]$ to the toy Hamiltonian, which is parametrized by α_j and should describe the occurrence of nematic order. As we will see in the following chapters, the generation of the artificial STM images based on the parameters α_j is rather straightforward. However the opposite case, meaning to extract the parameters from the generated images, is a highly

non-trivial task. This raises the exciting question of whether machine learning is able to solve this inverse problem, namely to deduce the corresponding parameters α_j of the underlying Hamiltonian from an STM image with a visible nematic phase. The fact that inverse problems have already been successfully solved with machine learning techniques shows courage for a positive outcome of this experiment [10–14].

In chapters 2-4 I will first present the basic theory necessary for understanding my work. It is a summary of the current state of research and in no way a product of my own cognitive work. The main sources I have used are given at the beginning of each chapter. Chapter 2 begins with a derivation of the most important properties of monolayer graphene and subsequently also for TBG, laying the foundation for a justification of the toy model. In the course of this, a brief overview of the current state of research on twisted double bilayer graphene (TDBG) will also be given. Furthermore a description of the nematic coupling is introduced. In chapter 3 the most important equations for the artificial generation of STM images are derived step by step and the physical meaning of these images is also dealt with in more detail. In chapter 4 the basics of neural networks are presented and the deep learning library Tensorflow, which I used for implementing the various architectures, is discussed. After these three theory chapters the actual part of my work begins, although the model for TBG introduced in chapter 5 was given to me from my supervisor. This chapter depends heavily on the theory in the first two chapters and ends with the successful generation of the STM images for different nematic couplings. Chapter 6 then corresponds to the main results of my work, in which machine learning is applied to the STM images.

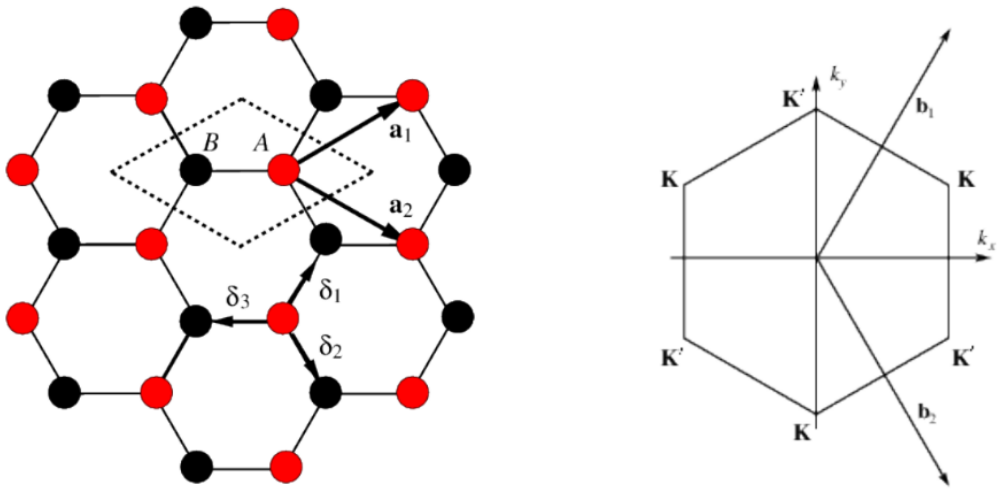
2 Twisted multi-layer graphene

In order to lay a foundation for twisted bilayer graphene (TBG) and twisted double bilayer graphene (TDBG), I will first summarize the most important properties of a simple monolayer graphene system. Based on these definitions, the physics of the layer stackings is derived.

2.1 Single-layer graphene

The derivations in the following monolayer graphene sections mainly follow the papers from Rozhkov [15] and Neto [16].

2.1.1 Geometry



(a) **(b)**

Figure 2.1: In figure (a) we see the honeycomb structure of single-layer graphene with a unit cell (dotted lines) containing two atoms. Each atom corresponds to a triangular sublattice denoted with (A, B) and can be reached via a Bravais lattice vector $\mathbf{R}^{A(B)} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2(+\boldsymbol{\delta}_3)$, where $\mathbf{a}_j$ are primitive vectors, n_j are integers and $\boldsymbol{\delta}_j$ correspond to the nearest-neighbor vectors. Figure (b) shows the corresponding hexagonal Brillouin zone with reciprocal lattice vectors $\mathbf{b}_j$ and high-symmetry corner points $\mathbf{K}^{(\prime)}$ [15].

Figure 2.1 (a) shows that a simple graphene layer consists of carbon atoms arranged in a hexagonal lattice. This lattice structure is based on a triangular Bravais lattice with a diatomic basis (A, B). One often speaks of two sublattices, since each of the two basis atoms is arranged in a triangular lattice itself. The two atoms are separated by a bond length of $a \approx 1.42\,\text{Å}$, which defines a lattice constant $a_0 = \sqrt{3}a$. If the origin of the lattice vectors lies on sublattice A, the bond length also corresponds to the length of the basis vector $\boldsymbol{\delta}_3$, which is also one of three nearest-neighbor vectors

$$\boldsymbol{\delta}_1 = \frac{a_0}{2}\left(\frac{1}{\sqrt{3}}, 1\right), \quad \boldsymbol{\delta}_2 = \frac{a_0}{2}\left(\frac{1}{\sqrt{3}}, -1\right) \quad \text{and} \quad \boldsymbol{\delta}_3 = -\frac{a_0}{\sqrt{3}}(1, 0)\,. \tag{2.1}$$

The two primitive lattice vectors, which define the triangular Bravais lattice $\mathbf{R}^{A(B)} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2(+\boldsymbol{\delta}_3)$ on sublattice A (B), are given as

$$\mathbf{a}_1 = a_0\left(\frac{\sqrt{3}}{2}, \frac{1}{2}\right), \quad \mathbf{a}_2 = a_0\left(\frac{\sqrt{3}}{2}, -\frac{1}{2}\right) \tag{2.2}$$

and in turn allow the specification of the reciprocal lattice vectors

$$\mathbf{b}_1 = 2\pi\frac{\mathbf{R}_{90^\circ}\mathbf{a}_2}{\mathbf{a}_1\mathbf{R}_{90^\circ}\mathbf{a}_2} = \frac{2\pi}{a_0}\left(\frac{1}{\sqrt{3}}, 1\right), \quad \mathbf{b}_2 = 2\pi\frac{\mathbf{R}_{90^\circ}\mathbf{a}_1}{\mathbf{a}_2\mathbf{R}_{90^\circ}\mathbf{a}_1} = \frac{2\pi}{a_0}\left(\frac{1}{\sqrt{3}}, -1\right), \tag{2.3}$$

where $\mathbf{R}_{90^\circ}$ is defined as a 90° rotation matrix. The resulting Brillouin zone (BZ) itself has a hexagonal structure and contains high symmetry points on its corners, which are central to the physical understanding of graphene. These $\mathbf{K}^{(\prime)}$ points are the only two distinguishable corner points of the BZ. Every corner point can be generated by choosing a definition for a pair $\mathbf{K}^{(\prime)}$ (see equation (2.9)) and translating these two points by a reciprocal lattice vector. The corresponding k-space geometry can be seen in figure 2.1 (b).

2.1.2 Tight-binding model

In order to derive the physics of single-layer graphene, a tight-binding description is introduced. Focusing on the basic case, where electrons on one lattice site can only tunnel to their nearest neighbors, the Hamiltonian reads

$$H = -t\sum_{\langle i,j\rangle,\sigma}\left(a^\dagger_{i,\sigma}b_{j,\sigma} + b^\dagger_{j,\sigma}a_{i,\sigma}\right) \tag{2.4}$$

where $a^{(\dagger)}_{\sigma,i}$ and $b^{(\dagger)}_{\sigma,j}$ are electron creation (annihilation) operators with spin $\sigma = (\uparrow, \downarrow)$ respectively defined for carbon sites on the two different sublattices $\mathbf{R}^A_i$ and $\mathbf{R}^B_j$. The Hamiltonian implies that t is the nearest-neighbor hopping energy, as the operators annihilate electrons on one sublattice site while creating electrons on a nearest-neighbor site simultaneously. To solve for the spectrum equation (2.4) can be rewritten in

reciprocal space. This is done by performing a Fourier transformation of the electron operators

$$a_{\mathbf{k},\sigma} = \frac{1}{\sqrt{N}} \sum_i e^{i\mathbf{k}\mathbf{R}_i^A} a_{i,\sigma}, \quad b_{\mathbf{k},\sigma} = \frac{1}{\sqrt{N}} \sum_j e^{i\mathbf{k}\mathbf{R}_j^B} b_{j,\sigma}, \tag{2.5}$$

with the indices $i(j)$ running over all carbon sites of sublattice A(B). Here N denotes the total number of unit cells (UCs). By writing the two operators as a single spinor

$$\boldsymbol{\Psi}_{\boldsymbol{k},\sigma} = (a_{\mathbf{k},\sigma}, b_{\mathbf{k},\sigma})^T, \quad \boldsymbol{\Psi}^\dagger_{\boldsymbol{k},\sigma} = \left(a^\dagger_{\mathbf{k},\sigma}, b^\dagger_{\mathbf{k},\sigma}\right), \tag{2.6}$$

and by rewriting the sum over nearest-neighbors $\langle i,j \rangle$ as a sum over the nearest-neighbor vectors from equation (2.1) the Hamiltonian from equation (2.4) becomes

$$H = \sum_{\boldsymbol{k},\sigma} \boldsymbol{\Psi}^\dagger_{\boldsymbol{k},\sigma} h_{\boldsymbol{k}} \boldsymbol{\Psi}_{\boldsymbol{k},\sigma}, \quad \text{with} \quad h_{\boldsymbol{k}} = -t \begin{pmatrix} 0 & f(\boldsymbol{k}) \\ f^*(\boldsymbol{k}) & 0 \end{pmatrix} \quad \text{and} \quad f(\boldsymbol{k}) = \sum_{j=1}^{3} e^{i\boldsymbol{k}\boldsymbol{\delta}_j}. \tag{2.7}$$

By diagonalizing the Bloch Hamiltonian $h_{\boldsymbol{k}}$ we then arrive at the important expression for the band spectrum:

$$E_\pm(\boldsymbol{k}) = \pm t|f(\boldsymbol{k})| = \pm t\sqrt{3 + 2\cos(a_0 k_y) + 4\cos(\frac{a_0}{2}k_y)\cos(\frac{\sqrt{3}}{2}a_0 k_x)} \tag{2.8}$$

Hence we have a spectrum with two bands ($\pm$) and the plot from figure 2.2 reveals the existence of two band crossings at the corner points of the BZ. We want to Taylor expand equation (2.8) close to these points, which can be defined as

$$\mathbf{K} = \frac{2\pi}{a_0}\left(\frac{1}{\sqrt{3}}, \frac{1}{3}\right), \quad \mathbf{K}' = \frac{2\pi}{a_0}\left(\frac{1}{\sqrt{3}}, -\frac{1}{3}\right), \tag{2.9}$$

yielding

$$E_\pm(\boldsymbol{K} + \boldsymbol{q}) = \pm\hbar v_F |\boldsymbol{q}|, \tag{2.10}$$

where the Fermi velocity is defined as $v_F = \sqrt{3}a_0 t/(2\hbar) \simeq c/300$, $\boldsymbol{q} = \boldsymbol{k} - \mathbf{K}^{(\prime)}$ and it should hold that $|\boldsymbol{q}| \ll |\boldsymbol{K}|$.

We can clearly see that the energy dispersion near the corner points is linear (as a clear distinction to the usual quadratic dependence). Furthermore the electrons behave like relativistic massless Dirac fermions. For this reason the corner points are so-called Dirac points. This special bandstructure also leads to a density of states (DOS) $\rho(E) \approx 2E/(\pi v_f^2)$ linear to the energy. The properties near the Dirac points are of particular importance, since in neutral graphene the physics near these points dominate. The dominance of the $\mathbf{K}^{(\prime)}$ points is more apparent when considering the electronic structure of graphene. It was shown that in the undoped case there is exactly one p_z electron on each carbon site, which leads to two electrons per unit cell [17]. We showed that there are two possible energy bands and since each state can be occupied by exactly two electrons, the valence band has to be fully occupied whilest the conduction band is empty. Therefore the linear DOS vanishes at $E = 0$ and the Fermi energy lies exactly at the crossing point of the bands. These properties are typical for semimetal materials [18].

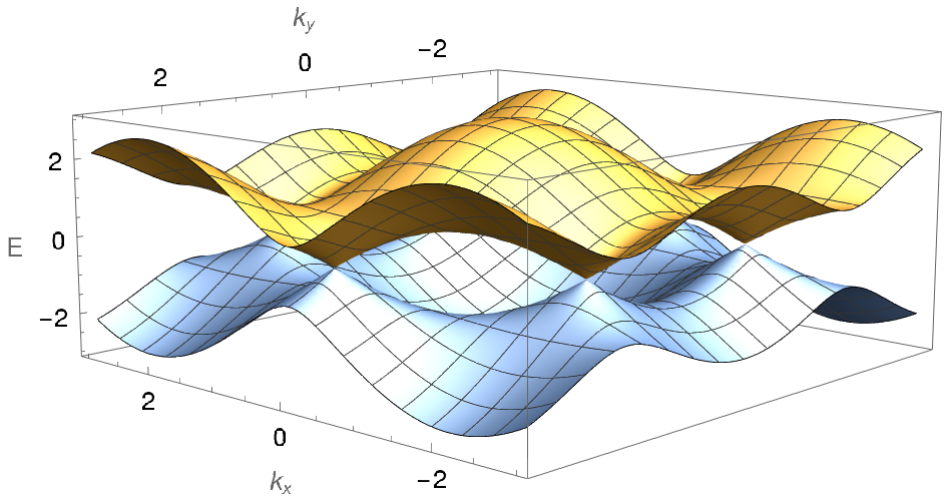


Figure 2.2: Plot of the lowest two bands of single-layer graphene. The Dirac points on the corners of the BZ separate the conduction (upper) and valence (lower) bands. In their vicinity the electron dispersion and DOS becomes linear. In this plot the parameters from equation (2.8) have been set to $t = 1$ and $a = 1$.

2.1.3 Dirac-like fermions

Close to the Dirac points the electrons are nearly massless and thus fulfill the massless Dirac equation. This can be shown by writing each of the transformed operators from equation (2.5) as a sum of two new operators. The idea behind this is that we want to separate the description for electrons close to each of the two Dirac points. The new operators are therefore labeled by $(\mathbf{K} \to 1, \mathbf{K}' \to 2)$ and redefine the old electronic operators on each site as

$$a_i \approx e^{-i\mathbf{K}\mathbf{R}_i^A} a_{1i} + e^{-i\mathbf{K}'\mathbf{R}_i^A} a_{2i}, \quad b_j \approx e^{-i\mathbf{K}\mathbf{R}_j^B} b_{1j} + e^{-i\mathbf{K}'\mathbf{R}_j^B} b_{2j}, \tag{2.11}$$

where the index for spin was dropped for convenience. These operators are then used in equation (2.4) and expanded to linear order in $\boldsymbol{\delta}$. The Hamiltonian then includes two copies of a massless Dirac-like Hamiltonian. Each copy describes the physics of electrons in the vicinity of each of the Dirac points $\mathbf{K}^{(\prime)}$. The first quantized Dirac equations read

$$\begin{aligned} -i\hbar v_F \boldsymbol{\sigma} \cdot \nabla \psi_1(\boldsymbol{r}) &= E\psi_1(\boldsymbol{r}) \ \ (\text{close to } \mathbf{K}) \\ -i\hbar v_F \boldsymbol{\sigma}^* \cdot \nabla \psi_2(\boldsymbol{r}) &= E\psi_2(\boldsymbol{r}) \ \ (\text{close to } \mathbf{K}^{'}), \end{aligned} \tag{2.12}$$

where ψ_i are electron wave functions with two components and $\boldsymbol{\sigma} = (\sigma_x, \sigma_y)$ are Pauli matrices. Those wave functions are of special interest when studying symmetries and topological properties of the system. In momentum space and for each of the two

lowest bands (±) with corresponding energies from equation (2.10) they are given as

$$\begin{aligned}\psi_1^{\pm}(\boldsymbol{q}) &= \frac{1}{\sqrt{2}}\begin{pmatrix} e^{-i\theta_{\boldsymbol{q}}/2} \\ \pm e^{i\theta_{\boldsymbol{q}}/2}\end{pmatrix}, \quad \text{for } H_K = \hbar v_F \boldsymbol{\sigma}\cdot\boldsymbol{q}, \\ \psi_2^{\pm}(\boldsymbol{q}) &= \frac{1}{\sqrt{2}}\begin{pmatrix} e^{i\theta_{\boldsymbol{q}}/2} \\ \pm e^{-i\theta_{\boldsymbol{q}}/2}\end{pmatrix}, \quad \text{for } H_{K'} = \hbar v_F \boldsymbol{\sigma}^*\cdot\boldsymbol{q},\end{aligned} \tag{2.13}$$

where the phase is defined as $\theta_{\boldsymbol{q}} = \arctan(q_x/q_y)$. By rotating the phase by 2π the wave function picks up a phase of π and changes sign. This behavior results from the chirality (or helicity), which describes the projection of the momentum on the pseudospin. The states from equation (2.13) are eigenstates of the chirality operators

$$\chi_1\psi_1^{\pm} = \pm\frac{1}{2}\psi_1^{\pm}, \quad \chi_2\psi_2^{\pm} = \pm\frac{1}{2}\psi_2^{\pm}, \quad \text{with } \chi_1 = \frac{\boldsymbol{\sigma}\cdot\boldsymbol{q}}{2|\boldsymbol{q}|}, \quad \chi_2 = \frac{\boldsymbol{\sigma}^*\cdot\boldsymbol{q}}{2|\boldsymbol{q}|} \tag{2.14}$$

and from these definitions we conclude that the pseudospin $\boldsymbol{\sigma}$ can be aligned parallel or antiparallel to $\boldsymbol{q}$. The states in the two different Dirac points have opposite chirality and each $\mathbf{K}^{(\prime)}$ point defines a so called valley [19]. These two valleys are usually denoted by $\xi = \pm 1$ as the Dirac points are often defined as $\mathbf{K}_- = -\mathbf{K}_+$. The new notation and valley index can then be used to write the Dirac Hamiltonian in the compact form [18] [20]

$$H^{\xi\mathbf{K}}(\boldsymbol{q}) = \hbar v_F \begin{pmatrix} 0 & \xi q_x - i q_y \\ \xi q_x + i q_y & 0\end{pmatrix} = \hbar v_F \boldsymbol{q}\cdot(\xi\sigma_x, \sigma_y). \tag{2.15}$$

2.1.4 Symmetries and distant-neighbor hopping

Since the overall aim is to justify a tight-binding model for TBG which is restricted by a minimal set of symmetries, let's first have a look at the symmetries in the monolayer graphene system. The principle idea of breaking symmetries by adding hopping terms in the Bloch Hamiltonian can later be used for the TBG case.

Equation (2.7) gives an off-diagonal Bloch Hamiltonian $h_{\mathbf{k}}$ for the simplest case of nearest-neighbor hopping t_1. Under this off-diagonal form the Hamiltonian preserves time-reversal

$$h^*(-\mathbf{k}) = h(\mathbf{k}) \tag{2.16}$$

and inversion symmetry

$$\sigma_x h(-\mathbf{k})\sigma_x = h(\mathbf{k}), \tag{2.17}$$

where σ is a Pauli matrix in pseudospin space. Both symmetries result in equations constraining the Hamiltonian $h(\mathbf{k})$ at opposite points in reciprocal space. To get a constraint at the same $\mathbf{k}$-point, the two symmetries are combined to TI-symmetry:

$$\sigma_x h^*(\mathbf{k})\sigma_x = h(\mathbf{k}) \tag{2.18}$$

We have showed with equation (2.15) that our off-diagonal and Hermitian matrix can be written in the general form $h(\mathbf{k}) = h_1(\mathbf{k})\sigma_x + h_2(\mathbf{k})\sigma_y$. Putting this expression in equation (2.18) immediately shows the invariance of graphene under this TI-symmetry. However adding a term $h_3(\mathbf{k})\sigma_3$ implies $h_3(\mathbf{k}) = -h_3(\mathbf{k}) \Rightarrow h_3(\mathbf{k}) = 0$, effectively forbidding a potential difference on the two sublattices. The protection of Dirac points in graphene can therefore be related to the fact, that no term proportional to σ_z is present in the Hamiltonian. A breaking of either time reversal or inversion symmetry opens a gap between the two bands [19]. A Semenoff mass term $m\sigma_z$ is sufficient to break inversion symmetry, whilest time-reversal symmetry in general can be broken by adding an external magnetic field. Haldane [21] showed that in graphene a complex intrasublattice hopping $t_2 e^{-i\phi}\sigma_z$, which corresponds to an alterning (but net) magnetic flux through the UC, is sufficient to break time-reversal symmetry. A monolayer graphene system with t_1-hopping, the t_2-hopping term just mentioned and a Semenoff mass term corresponds to the original Haldane model. We will later use these ideas and define our own Haldane model for TBG [22–24].

2.2 Twisted bilayer graphene

Even if a single layer of graphene already has exciting electronic properties, new insights can be gained by stacking multiple layers of graphene on top of each other. It has been found that very interesting physics happens when bilayers of graphene are twisted in relation to one another. The resulting geometry is called a moiré pattern. Theoretical investigations have shown that such moiré patterns lead to a band structure containing two moiré bands per valley and per spin [25]. These bands are very flat, in contrast to the linear bands in single-layer graphene. The bandwidth depends on the twist angle between the layers, leading to magic angles ($\theta \approx 1.1°$) where the bandwidth almost vanishes completely. Such a material is called magic angle twisted bilayer graphene (MATBG). Because the energy in such bands changes only very slowly with the electron momentum, the electrons are strongly correlated. Such strongly correlated systems are of particular interest, as it is hoped that this will provide a better understanding of the insulating and superconducting states in cuprates. In fact, in 2018 [26, 27], a team from MIT was able to detect such phases in TBG as well [28].

2.2.1 Geometry

An example of a moiré structure can be seen in figure 2.3 (a). Looking at this pattern it can bee seen that light and dark areas alternate periodically. A separate description can be assigned to each of these areas. In the bright areas one speaks of AA stacking, since each atom from the upper layer has a corresponding partner from the lower layer. In the dark areas on the other hand there is always an atom without a partner, which is called AB or BA stacking [19]. One can clearly see that the AB/BA

stacking areas create a honeycomb lattice, whereas the AA stacking areas sit in the middle of these hexagons and form a triangular lattice. In general every twist angle between two stacked graphene layers is sufficient to create a moiré pattern, however, exact lattice translation symmetries only result if so-called commensurate angles are found. In various papers from the last few years authors have introduced equations and geometrical properties resulting from these commensurate angles [29–34]. In the following the most important findings are given. The commensurate angles for a corresponding superlattice are given as

$$\cos\theta = \frac{m^2 + 4mn + n^2}{2(m^2 + mn + n^2)} \tag{2.19}$$

where m and n are integers. The main focus lies on twist angles between $\theta \in (0, \pi/3)$ as angles outside of the given range can be attained via the symmetries of the superlattice. Those symmetries are determined by the location of the twisting center, however, the twisting center has no influence on the commensuration condition. The commensuration and therefore lattice properties like the superlattice constant $L(m, n)$ solely depend on the twist angle θ. The lattice vectors of the superlattice can be given through the lattice vectors of single-layer graphene ($\mathbf{a}_i$)

$$\mathbf{L}_1 = n\mathbf{a}_1 + m\mathbf{a}_2, \;\; \mathbf{L}_2 = -m\mathbf{a}_1 + (n+m)\mathbf{a}_2, \tag{2.20}$$

leading to a superlattice period

$$L = \frac{ra_0}{2\sin(\theta/2)}. \tag{2.21}$$

One can show [30] that the superlattice constant L has a $r = |m - n|$ times bigger length than the moiré lattice vectors L_M, which in principle can be chosen for any angle θ. It follows that the lattice vectors of the superlattice unit cell $\mathbf{L}_i$ and the moiré unit cell (mUC) $\mathbf{L}_M^i$ coincide for $r = 1$. An example for this condition is shown in figure 2.3 (a), where $(m, n) = (8, 9)$ leading to a twist angle $\theta = 3.89°$. Figure (b) shows the k-space for the same configuration. It can be seen that the four valleys $\mathbf{K}_\pm^{(l)}$ of the two stacked layers are folded into two new $\bar{\mathbf{K}}^{(\prime)}$ points of the moiré BZ (mBZ). In this case of $r = 1$ the Dirac points $\mathbf{K}_+^{(1)}$ and $\mathbf{K}_-^{(2)}$ ($\mathbf{K}_+^{(2)}$ and $\mathbf{K}_-^{(1)}$) map to $\bar{\mathbf{K}}$ ($\bar{\mathbf{K}}'$). At small twist angles it becomes clear that $|\mathbf{K}_+^{(1)} - \mathbf{K}_+^{(2)}| \ll |\mathbf{K}_+^{(1)} - \mathbf{K}_-^{(2)}|$ meaning that a coupling between the Dirac points $\mathbf{K}_+^{(1)}$ and $\mathbf{K}_-^{(2)}$ is restrained, though they are folded to the same Dirac point in the reciprocal moiré space. This valley charge conservation, corresponding to the $U_\nu(1)$ symmetry, allows to group the states in the vicinity of $\mathbf{K}_+^{(1)}$ and $\mathbf{K}_+^{(2)}$ into a single valley, again denoted with $\xi = +$. The same holds for their time reversal opposites being grouped to valley $\xi = -$ again for small twist angles [29].

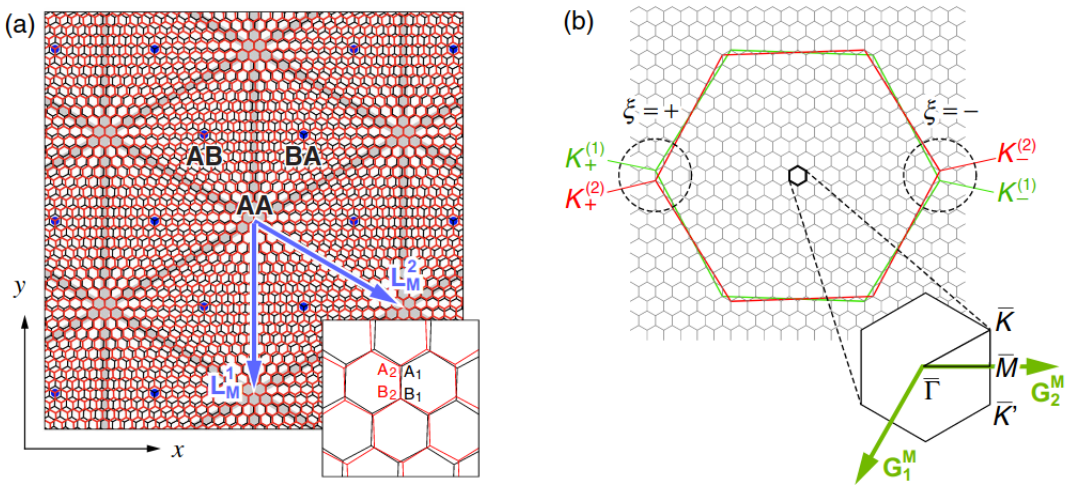


Figure 2.3: Figure (a) gives the atomic structure and moiré lattice vectors of two stacked graphene layers with a twist angle $\theta = 3.89°$, $(m, n) = (8, 9)$. Figure (b) shows the corresponding BZ folding of the two single-layer BZs (green and red lines) leading to a reduced moiré BZ. The reciprocal lattice vectors of the moiré lattice are given as $\mathbf{G}_i^M$ [35].

2.2.2 Symmetries

The following symmetries section is mainly based on Ref. [29]. Besides the translational symmetry achieved via the commensuration condition in equation (2.19) there are further microscopic symmetries depending on the location of the twisting center. Taking the AA stacking case as a starting point, three available high-symmetry points for the twisting center are the honeycomb center and the two carbon atoms. In the first case we have the largest point group, namely D_6. This group is generated by C_6 rotation on the one hand, on the other hand there is a rotational symmetry that exchanges the two layers: C_{2x}. This symmetry can also be found if we place the twisting center on one of the carbon atoms. However, the point group is now reduced to D_3, since the superlattice is only invariant under C_3 rotations. Just like monolayer graphene the system again is invariant under time reversal T.

Wannier obstruction

It is found that a tight-binding model is constrained under the resulting representations at the high-symmetry points. A proper description of the two nearly flat bands (per valley and per spin) can only be attained by placing the Wannier functions in the middle of the AB/BA regions, leading to a honeycomb structure. By setting up a corresponding tight-binding theory it turns out that the mentioned symmetries are not sufficient to explain the robustness of the Dirac points determined in experiments. For this reason approximate symmetries, which are not yet visible in the commensurate structures, are introduced. As can be seen in continuum theory, a nonlocal

$C_2T = (C_6T)^3$ symmetry is able to describe the protection of Dirac points, but at the same time leads to an obstruction when trying to construct well-localized Wannier states. This behavior results from the $U_\nu(1)$ symmetry, since both monolayer valleys grouped in the bilayer valley have the same chirality, leading to a total chirality not equal to zero. However, a two band tight-binding Hamiltonian always has to result in a total chirality of zero. The connection of this chirality obstruction with mirror obstruction resulting from C_{2x}-symmetry has been shown by Ref. [29]: To define the chirality operator for TBG one can first look at an open region of the mBZ which completely covers the Dirac points. The smoothness of a basis of Bloch wave functions can then be secured by defining C_2T as

$$c_{\mathbf{k}} \to \sigma_x K c_{\mathbf{k}}, \tag{2.22}$$

where K is complex conjugation and $c_{\mathbf{k}}$ is an electron operator in momentum space. Now the Hamiltonian is constrained by this symmetry and has to be of the form

$$H(\mathbf{k}) = n_0(\mathbf{k}) + n_1(\mathbf{k})\sigma_x + n_2(\mathbf{k})\sigma_y. \tag{2.23}$$

This further allows for a definition of the chirality as the winding of $(n_1(\mathbf{k}), n_2(\mathbf{k}))^T$. In this basis the operators have to transform under C_{2x} either as

$$c_{\mathbf{k}} \to \sigma_x c_{C_{2x}\mathbf{k}}, \tag{2.24}$$

for opposite eigenvalues at the high symmetry point M or as

$$c_{\mathbf{k}} \to \eta_M c_{C_{2x}\mathbf{k}}, \tag{2.25}$$

for identical eigenvalues η_M. As a consequence $(n_1(\mathbf{k}), n_2(\mathbf{k}))^T$ has to transform as

$$(n_1(\mathbf{k}), n_2(\mathbf{k}))^T \to (n_1(C_{2x}\mathbf{k}), -n_2(C_{2x}\mathbf{k}))^T \text{ for opposite EVs,} \tag{2.26}$$
$$(n_1(\mathbf{k}), n_2(\mathbf{k}))^T \to (n_1(C_{2x}\mathbf{k}), +n_2(C_{2x}\mathbf{k}))^T \text{ for identical EVs.} \tag{2.27}$$

Looking at equation (2.26) one sees that the loops around K and K' of the mBZ have the same number of windings, while equation (2.27) leads to different ones. As the number of windings around the Dirac cones corresponds to the chirality in their vicinity it follows that chirality obstruction can be taken into account by defining the action C_{2x} via equation (2.24) [29] [36]. We will use this fact again by defining our own minimal tight-binding model for TBG.

Solving the obstruction

As our own minimal model should take into account this obstruction of the Wannier states, it is of particular interest of how to integrate it in the simplest possible way. Luckily the obstruction can be solved by adding additional bands in a Haldane-like tight-binding model. The solution shown by Zou [29] is a four-band model (for each

valley and spin) where we have a pair of two bands with opposite Chern number. This four-band model can be attained by placing two orbitals on each site of the two site UC of the honeycomb lattice (spanned by the AB/BA sites). As we only want to focus on a single valley it is clear that C_6 and T won't preserve the valley. However, all the other symmetries and combinations of C_6T will preserve it leading to a single valley point group generated by C_6T and C_{2x} [36]. It is therefore important to note that $(C_6T)^2 = C_3$. We will use the fundamental idea of this model later.

2.2.3 Nematicity

The following description of nematicity is taken from Ref. [37]. In addition to the superconducting and insulating phases, there are other notable phases of TBG that can be attributed to the strong correlations between the electrons. One of them is a nematic phase, which in the case of TBG is defined by a break in C_3 symmetry at various points in the proposed phase diagram. The representation of the nematic order parameter is rather straightforward on a hexagonal lattice. It is defined by the two components

$$\mathbf{\Phi} = (\phi_1, \phi_2)^T, \tag{2.28}$$

and it should transform under the irreducible representation E of D_3 in the case of our suggested four-band minimal model. As a first guess it can be assumed that the nematic order can be oriented arbitrarily, which can be expressed by the parameterization

$$\mathbf{\Phi} = \phi(\cos 2\theta, \sin 2\theta)^T, \tag{2.29}$$

with an angle θ determining the orientation of the director $\mathbf{n} = (\cos\theta, \sin\theta)^T$. However it turns out that the nematic director cannot take any direction, but is limited to a certain number of angles due to crystal anisotropies. This can be studied in Landau theory, where the anisotropy is taken into acount with the cubic term in the action

$$S_{nem}[\mathbf{\Phi}] = S_0[\mathbf{\Phi}] + \underbrace{\frac{\gamma}{6}\int_x (\phi_+^3 + \phi_-^3)}_{\frac{\gamma\phi^3\cos 6\theta}{3}}, \tag{2.30}$$

where the last term is integrated over a spatial coordinate and imaginary time $x = (\mathbf{r}, \tau)$ and $\phi_\pm = \phi_1 \pm i\phi_2$. The final form of the cubic term suggests that there are different solutions for the minimum, depending on the sign of γ:

$$\gamma > 0: \quad \theta = \frac{(2n+1)\pi}{6} \tag{2.31}$$

$$\gamma < 0: \quad \theta = \frac{2n\pi}{6} \tag{2.32}$$

As it holds that $\mathbf{\Phi}(\theta + \pi) = \mathbf{\Phi}(\theta)$ the allowed solutions are given for $n = 0, 1, 2$ and are hence threefold degenerate. In real space this corresponds to a strengthening of

the lattice bonds like shown in figure 2.4. We will be able to observe these bond strengthenings on our artificially generated STM images.

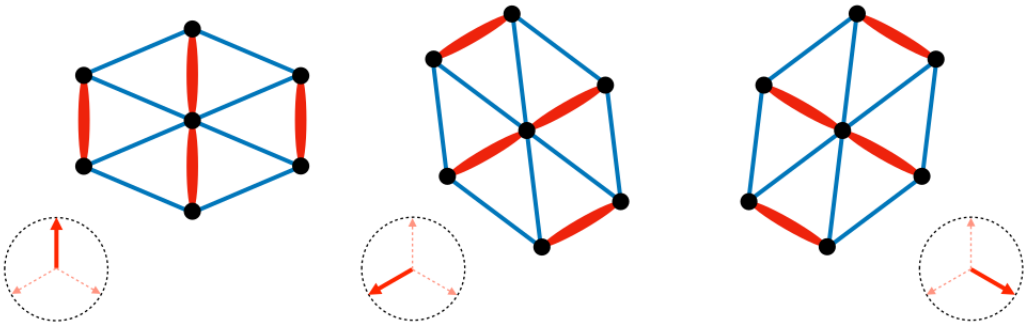


Figure 2.4: Band ordering for threefold degenerate nematic oder in real space [37].

2.3 Twisted double bilayer graphene

The interesting electronic properties of TBG only become visible when the twist angle θ has been tuned closely to a magic angle. For this reason in experiments with TBG the focus lies on setting up θ as precisely as possible. In order to circumvent this difficult task further possibilities are sought to fine tune flat moiré bands. One approach is to push the idea of graphene layer stacking further. Then the next emergent multi-layer systems are trilayer graphene and twisted double bilayer graphene (TDBG). The latter one is composed of two AB-stacked graphene bilayers, which are then rotated by an angle θ, leading again to a moiré superlattice. This system seems to be particularly interesting, because (in contrast to TBG) the bandwidth of the moiré bands can be controlled via an external electric displacement field D. It has been shown that under certain ranges of this applied field a correlated insulator phase can be generated, which doesn't rely on an accurate magic angle between the layers [38, 39].

The displacement field D not only influences the band structure of TDBG, but also the symmetries of the system. First of all, TDBG has point group D_3, which means that the system, like TBG, is invariant under C_3 and C_{2x} rotations. However, a displacement field breaks the C_{2x} symmetry, lowering the point group to C_3. The symmetry considerations of TBG cannot easily be transferred to TDBG, since TDBG is not invariant under C_2 rotations. Since in TBD the combined symmetry C_2T protects the Dirac cones, it is not surprising that in TDBG those are absent [40]. Nevertheless TDBG systems are an interesting and thriving research topic and though my work focuses on a minimal TBG system it hopefully will also be generalizeable to TDBG models.

3 Scanning tunneling microscope

This chapter follows the derivations and explanations from Bruus [41]. The basic principle of a scanning tunneling microscope (STM) is that electrons can tunnel from the probed material to a tip of a measuring device. This tunneling can happen because of a small overlap of the wave functions from both materials, which is described by a tunnel matrix. In order to be able to describe this matrix and thus also the electron current that flows through the device, the concept of Green's function plays a key role.

3.1 Many-body Green's function

The idea of a many-body Green's function goes back to the non-interacting Green's function, which is also called a propagator as it gives a wavefunction $\Psi(\mathbf{r}, t)$ known at time t' at a later time t via the equation

$$\Psi(\mathbf{r}, t) = \int d\mathbf{r}' G(\mathbf{r}'t', \mathbf{r}t)\Psi(\mathbf{r}', t'), \tag{3.1}$$

where Green's function is given in the form

$$G(\mathbf{r}t, \mathbf{r}'t') = -i\Theta(t - t')\langle\mathbf{r}|e^{-iH(t-t')}|\mathbf{r}'\rangle. \tag{3.2}$$

In analogy to this non-interacting propagator, the retarded Green's function is defined for the many-body case

$$G^R(\mathbf{r}\sigma t, \mathbf{r}'\sigma' t') = -i\Theta(t - t')\langle\{\Psi_\sigma(\mathbf{r}t), \Psi^\dagger_{\sigma'}(\mathbf{r}'t')\}\rangle, \tag{3.3}$$

where $\{..., ...\}$ is the anti-commutator as we are dealing with fermions. The retarded Green's function can also be viewed as a propagator since equation (3.3) describes the amplitude with which a particle, starting in position $\mathbf{r}'$ at time t', propagates to a new position $\mathbf{r}$ at a later time t, completely analogous to the interpretation of equation (3.2). The big difference to the non-interacting propagator is that now the whole many-body Hamiltonian is taken into account and therefore correlations between the electrons can also be treated properly. The function is called retarded, because only times $t > t'$ are allowed. There are more types of Green's functions treating different cases for the allowed time evolution. The Green's function in real space $G^R(\mathbf{r}\sigma t, \mathbf{r}'\sigma' t')$

can be connected to every other basis $|\nu\rangle$ via

$$G^R(\mathbf{r}\sigma t, \mathbf{r}'\sigma' t') = \sum_{\nu,\nu'} \psi_\nu(\sigma\mathbf{r}) G^R(\nu\sigma t, \nu'\sigma' t') \psi^*_{\nu'}(\sigma'\mathbf{r}'), \quad \text{with} \tag{3.4}$$

$$G^R(\nu\sigma t, \nu'\sigma' t') = -i\Theta(t-t')\langle\{a_{\nu\sigma}(t), a^\dagger_{\nu'\sigma'}(t')\}\rangle. \tag{3.5}$$

In a translational invariant system we are further allowed to write $G^R(\mathbf{r}, \mathbf{r}')$ as $G^R(\mathbf{r} - \mathbf{r}')$ and connect it to the Green's function $G^R(\mathbf{k}\sigma t, \mathbf{k}'\sigma' t') = -i\Theta(t-t')\langle\{a_{\mathbf{k}\sigma}(t), a^\dagger_{\mathbf{k}'\sigma'}(t')\}\rangle$ in k-space through the relation

$$G^R(\mathbf{r} - \mathbf{r}', \sigma t, \sigma' t') = \frac{1}{V} \sum_{\mathbf{k}} e^{i\mathbf{k}(\mathbf{r}-\mathbf{r}')} G^R(\mathbf{k}, \sigma t, \sigma' t'), \tag{3.6}$$

where we used the fact that $G(\mathbf{k}, \mathbf{k}') = \delta_{\mathbf{k},\mathbf{k}'} G(\mathbf{k})$. In the case of free electrons with a corresponding Hamiltonian $H = \sum_{k\sigma} \xi_{\mathbf{k}\sigma} c^\dagger_{\mathbf{k}\sigma} c_{\mathbf{k}\sigma}$ the retarded Green's function takes the form

$$G^R_0(\mathbf{k}\sigma, t - t') = -i\Theta(t-t') e^{-i\xi_{\mathbf{k}}(t-t')}, \tag{3.7}$$

where Green's function is diagonal in $\mathbf{k}$ and σ. A Fourier transform of equation (3.7) gives

$$G^R_0(\mathbf{k}\sigma, \omega) = \lim_{\eta \to 0^+} \frac{1}{\omega - \xi_{\mathbf{k}} + i\eta}, \tag{3.8}$$

which is further used for the definition of the spectral function

$$A_0(\mathbf{k}\sigma, \omega) = -2\Im[G^R_0(\mathbf{k}\sigma, \omega)] = 2\pi\delta(\omega - \xi_{\mathbf{k}}). \tag{3.9}$$

As the spectral function is simply a delta function in the case of non-interacting electrons, an energy excitation ω can only occur if the added electron is in state $\xi_{\mathbf{k}} = \omega$. In general the spectral function does not simply correspond to a delta function and is given through

$$A(\nu, \omega) = -2\Im[G^R(\nu, \omega)]. \tag{3.10}$$

A very important relation is, that the occupation of an electron state is given as

$$n_\nu = \langle c^\dagger_\nu c_\nu \rangle = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} A(\nu, \omega) n_F(\omega), \tag{3.11}$$

where n_F is the Fermi-Dirac distribution. This equation makes it clear that the spectral function allows an interpretation similar to that of the density of states and that conclusions about the DOS are possible by measuring the spectral function.

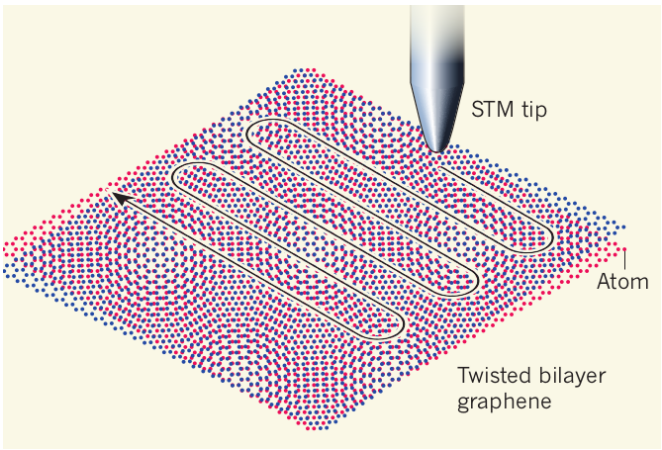


Figure 3.1: A STM tip scanning across TBG measuring the tunneling current [42].

3.2 STM

The set-up of an STM experiment consists of a material to be examined and a conductive tip that is moved across the material (see figure 3.1). Electrons can tunnel between the two materials because the wave functions overlap if the distance is small enough. This tunneling current can be measured via contacts attached to the two materials and gives information about the topography of the scanned system [42]. This behavior is also made visible in the Hamiltonian of both materials, which receives an additional term

$$H_{12} = \sum_{\nu\mu} \left(T_{\nu\mu} c^\dagger_{1,\nu} c_{2,\mu} + T^*_{\nu\mu} c^\dagger_{2,\mu} c_{1,\nu} \right), \tag{3.12}$$

where $c_{1,\nu}$ and $c_{2,\mu}$ are electron operators corresponding to the Hamiltonians of the two systems H_1 and H_2. The tunneling matrix is given through

$$T_{\nu\mu} = \int d\mathbf{r} \psi^*_\nu(\mathbf{r}) H(\mathbf{r}) \psi_\mu(\mathbf{r}), \tag{3.13}$$

where $H(\mathbf{r})$ is the one-particle Hamiltonian and ψ_j are electron wavefunctions for both systems given in first quantization language. A current flowing from system 1 to system 2 is defined by the changing of particle numbers and hence can be written in a mathematical form most prominent from the Heisenberg picture

$$I = \frac{d}{dt} N_1 = i[H, N_1] = -i \sum_{\nu\mu} \left(T_{\nu\mu} c^\dagger_{1,\nu} c_{2,\mu} - T^*_{\nu\mu} c^\dagger_{2,\mu} c_{1,\nu} \right). \tag{3.14}$$

Since the overlap of the wave functions is only very weak and the tunneling decreases exponentially with distance, the current can be calculated down to the lowest order in $T_{\nu\mu}$. As can be seen in equation (3.14), only one term for the second order is missing. After applying linear response theory and doing a Fourier transformation, the current

can be written as a function of the spectral density as follows

$$I = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} \sum_{\nu\mu} |T_{\nu\mu}|^2 A_1(\nu,\omega) A_2(\mu,\omega+eV)[n_F(\omega+eV) - n_F(\omega)], \tag{3.15}$$

where the last factor shows a dependence on the difference of Fermi-Dirac distributions and $V = V_2 - V_1$ is a difference of applied voltages to the respective systems.

An STM experiment to measure the local density of states is done by an measurement of the differential conductance dI/dV. A nearly constant DOS in the tip material leads to a simplification of the sum over μ in equation (3.15), namely

$$\sum_{\mu} |T_{\nu\mu}|^2 A_2(\mu,\omega+eV) \approx \text{ const.} \tag{3.16}$$

Hence equation (3.15) can be written as

$$\frac{dI}{dV} \propto \int_{-\infty}^{\infty} d\omega \left(\frac{\partial n_F(\omega+eV)}{\partial\omega} \right) \sum_{\nu} A_1(\nu,\omega), \tag{3.17}$$

which can be further simplified by noticing that the derivative of the Fermi-Dirac distribution approaches a delta function at low temperatures:

$$\frac{dI}{dV} \propto \sum_{\nu} A(\nu, -eV) \tag{3.18}$$

This is a straightforward equation to experimentally measure the spectral function

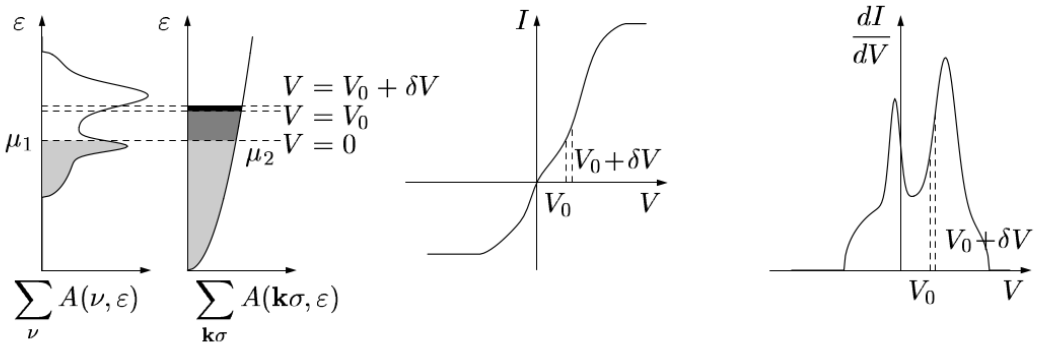


Figure 3.2: A difference in the chemical potential μ_i induced by an applied voltage results in a current and a differential conductance, which successfully measures the spectral function of the probed material $A(\nu,\epsilon)$ [41].

and therefore the local density of states by measuring the change of current when altering the voltage. It also allows to compute artificial STM images by defining a

proper spectral function. Figure 3.2 shows the relation between the DOS of the probed material and the measured differential conductance for a tip with nearly constant DOS. It can be seen that an applied voltage $+V_0$ shifts the chemical potential μ_2 of the tip thus leading to a current to the probed material. The differential conductance can then be derived by adding a small voltage δV leading to an adjusted current dI. Varying the voltage V_0 from positive to negative values results in a dI/dV spectrum resembling the LDOS of the probed material above and below the chemical potential μ_1.

4 Machine learning

This chapter is mainly based on the explanations and derivations from Nielsen [43] and Géron [44]. The overall aim of my work is to develop a computer program that recognizes and examines patterns in STM images that are not analyzable by the human eye. As we have seen from the references in the introduction such tasks are being increasingly solved with machine learning techniques. In general machine learning means to program a computer in such a way that it can learn from existing data sets. In the best case, the computer can then apply its gained "experience" to new data. A common example are classification tasks, in which the program learns the connection between some training data and its classifications (labels). When the data is supplied with the labels, this is called supervised learning, in contrast to unsupervised learning, where no help of this kind is provided for the program. For the learning task a model with tuneable parameters is defined and these parameters are optimized to output the correct labels. The corresponding quantity to minimize in the optimization process is called loss function. The selection of the model architecture is key in categorizing machine learning programs. In our case of building an image classifier, one type of architecture is particularly suitable: artificial neural networks

4.1 Artifical neural networks

The very first artificial neural network (ANN) was invented back in the 1940s in an attempt to explain the neural processes in animal brains [45]. Since then a variety of architectures have been proposed. They all share the common idea that just like the logical building blocks of our brain, which are called neurons, artificial neurons can be build up to form huge networks on their own solving increasingly difficult tasks. The simplest type of artificial neuron is a threshold logic unit (TLU). An example can be seen in figure 4.1 (a). Such a neuron processes an input array $\mathbf{x}$ through three steps. First it gives a weight w_j to every single input x_j and sums them up $z = \mathbf{x} \cdot \mathbf{w}$. Next it applies a so-called activation function and finally it returns the result. A simple choice for an activation function is a step function, then the output can only take the values 0/1 leading to a binary classifier. Training this kind of classifier would mean to find the suitable weights w_j under which the TLU outputs the correct classification (which is either 0 or 1) for various inputs. Usually a trainable bias neuron is also added to the summation $z = \mathbf{x} \cdot \mathbf{w} + \mathbf{b}$, increasing the tunability of the output. A single TLU can only perform very simple logical operations and even a whole layer of TLUs, which is called a perceptron, quickly reaches its limitations. However if several perceptron

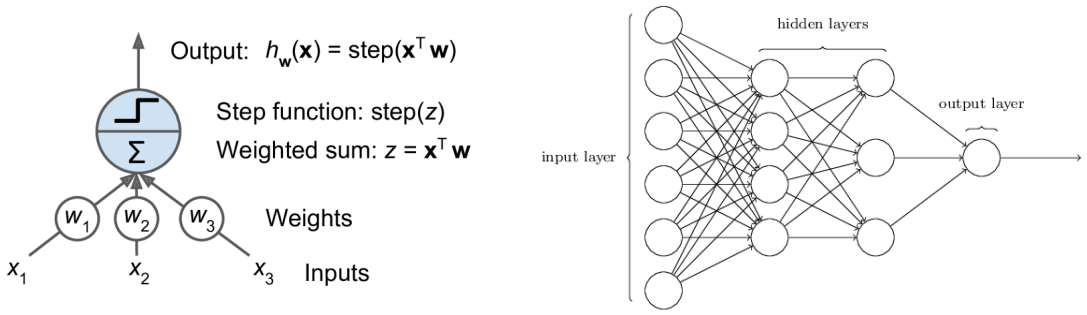


Figure 4.1: Figure (a) shows the functioning of a single threshold logic unit (TLU) [44]. Figure (b) shows the alignment and stacking of TLUs leading to a multi layer perceptron (MLP) [43].

layers are connected one after the other the resulting deep neural network (DNN) is capable of solving highly complex problems. In figure 4.1 (b) an example architecture for such a multilayer perceptron (MLP) is shown. One sees that the neurons in the first hidden layer are connected to all the input neurons. In the second hidden layer, in turn, all neurons are connected to all neurons in the previous layer. So the output from one neuron is used as input for all neurons in the next layer. In this way, any number of layers can be lined up one after the other, leading to networks which can accomplish increasingly subtle tasks. As all neurons are connected to the ones in the previous layer one often speaks of dense layers.

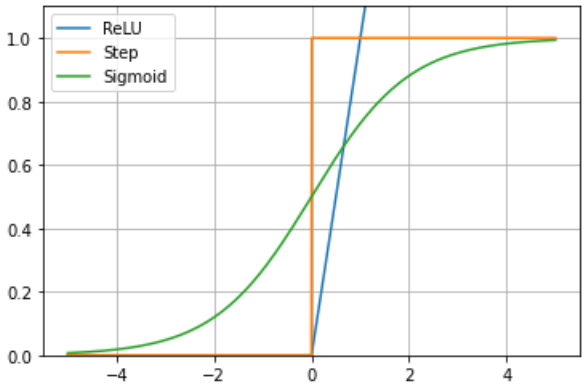


Figure 4.2: Comparison of possible activation functions for an artificial neuron.

However, in order for a DNN to solve complex problems, the weights and biases have to be trained with the help of training data. A desired behavior for the weight updates would be that small changes in the output can be triggered by small changes of the weights. This cannot be achieved with a simple step function, since here the values can jump back and forth abruptly between 0 and 1. For this reason, various activation functions were introduced, of which the Rectified Linear Unit function (ReLU)

$$ReLU(z) = max(0, z) \tag{4.1}$$

and the sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{4.2}$$

have proven their worth today. As can be seen in figure 4.2 the sigmoid function approaches the step function for very large and small values. In the middle range, however, it is smooth, which leads to a continuously changing output when the weights are tweaked. The sigmoid function is often used at the output layer for classification tasks. In this way, the output value can be interpreted as a probability of belonging to a class. The ReLU is the standard function for the hidden layers, as it has proven its usefulness in practical use. In order to understand how the activation function works on the neurons in the various layers, it is advisable to introduce a notation for these kind calculations. The notation in the following equations is based on Ref. [43]. The activation of a neuron j in layer l can be linked to the activation of a neuron in layer $(l-1)$ by the equation

$$a_j^l = f\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) = f\left(z_j^l\right), \tag{4.3}$$

where k runs over the neurons in layer $(l-1)$ and w_{jk}^l is the weight between the neuron j in layer l and the neuron k in layer $(l-1)$. This equation can also be written in matrix form $a^l = f\left(w^l a^{l-1} + b^l\right) = f\left(z^l\right)$ where the activation function is applied elementwise.

4.2 How training works

4.2.1 Gradient Descent

With equation (4.3) we now have a mathematical form that describes the output of each neuron in each layer. The question now is how the weights w_{jk}^l can be trained in each layer so that the network finally outputs the correct output vector a^L. The basic method to achieve this is Gradient Descent (GD). The idea of GD is to minimize a cost function C. This is done by calculating the gradient of the cost function ∇C with regard to its parameters, in this case the weights and biases. Since the gradient always points in the direction of greatest slope, a minimum of the cost function can be achieved by pushing the parameters in the "opposite" direction. The size of this adjustment is determined with a learning rate η. Calculating and substracting the gradient from the parameters over and over again thus means that a minimum of the cost function can be found over time. The resulting algorithm for updating the weights and biases reads

$$w_{jk}^{l'} = w_{jk}^l - \eta \frac{\partial C}{\partial w_{jk}^l}, \quad b_j^{l'} = b_j^l - \eta \frac{\partial C}{\partial b_j^l}. \tag{4.4}$$

A basic choice for the cost function C is the mean squared error (MSE)

$$C_{\text{MSE}} = \frac{1}{2n} \sum_x \parallel y(x) - a^L(x) \parallel^2, \tag{4.5}$$

where $a^L(x)$ is the final layer output of a training sample x, $y(x)$ is the true output of that sample and the sum runs over all training samples n. As this cost function is an average over cost functions applied to a single sample $C = 1/n \sum_x C_x$, the gradients in equation (4.3) have to be calculated for every single training sample. However this time consuming task can be improved by dividing the training data into mini-batches. The gradients are then calculated and averaged for only a small number of randomly selected samples. After the weights and biases are adjusted, a new mini-batch is selected. This continues until all training samples have been processed. One run through the training dataset is called an epoch.

4.2.2 Backpropagation

Backpropagation is an algorithm to efficiently calculate the gradients from equation (4.4). As its name suggests, the starting point for calculation lies at the output layer of the network. The main idea behind this method is to calculate errors δ_j^l for neurons j in layers l during one run through the model (the forward pass) and to relate those errors to the gradients. An error can be defined by

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}, \tag{4.6}$$

then the total cost of the network would be changed by $\delta_j^l \Delta z_j^l$ if the weighted sum of a single neuron gets tweaked by $f\left(z_j^l + \Delta z_j^l\right)$. This implies that the overall cost can't be minimized anymore if δ_j^l is already very small, justifying the definition of the error. With the help of this error the steps of the backpropagation algorithm read as follows:

$$\delta^L = \nabla_a C \odot f'(z^L) \tag{4.7}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot f'(z^l) \tag{4.8}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l, \quad \frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{4.9}$$

After a full forward pass, during all activations a^l have been computed, the algorithm starts at caclulating the error of the output layer δ^L. This error can then be used to calculate the error from the previous layer, δ^{L-1}. Applying equation (4.8) over and over again will determine all errors through all layers, which are then used to calculate every gradient fo the cost function in the network via equation (4.9).

4.2.3 Better optimizers

As we have seen, many steps are necessary for a training epoch in order to adjust the parameters well. Training can therefore take a very long time, especially in very deep networks with millions of parameters. On the one hand, this can result in various problems that seem to occur in very deep networks (e.g. unstable gradients), on the other hand, the gradient descent algorithm also has potential for improvement. The following equations and explanations in this section rely mainly on Ref. [44].

Momentum optimization

A first way to improve the GD algorithm is to take into account the old gradient from the previous calculation step when updating the new parameters. Such an algorithm can be derived from the analogy of a ball rolling towards the global minimum. With normal gradient descent the ball will always have the same velocity, but with momentum optimization the ball will experience an acceleration. This allows the algorithm to converge faster [46,47]. The algorithm for the weights can be represented mathematically as follows

$$m' = \beta m + \eta \nabla_w C, \tag{4.10}$$

$$w' = w - m, \tag{4.11}$$

where we dropped the indices for the layer and exact position of the neuron. Here m corresponds to the so-called momentum and β is a new hyperparameter, which is called friction in analogy to the physical image of an accelerating ball. A common value for this hyperparameter is $\beta = 0.9$.

Adaptive Gradient Algorithm (AdaGrad)

Another way to further consider the gradients calculated in the previous step is to use them to adjust the learning rate η. For this, the following algorithm for the weights of a neuron was proposed by Duchi [48]

$$g' = g + \nabla_w C \otimes \nabla_w C, \tag{4.12}$$

$$w' = w - \frac{\eta}{\sqrt{diag(g) + \epsilon}} \cdot \nabla_w C, \tag{4.13}$$

where ϵ is a very small constant to avoid division by zero and g' is the outer product matrix of the summation over the square roots of all previous gradients. This means that the learning rate will be downscaled if the cost function is very steep. This behavior should lead to updates that approach the global minimum more directly.

RMSProp

An often encountered problem with AdaGrad is that the algorithm scales down the learning rate too much and that the learning never converges. The RMSProp algo-

rithm tries to fix this by limiting the summation from equation (4.12) to the most recent gradients via an exponential decay

$$g' = \beta g + (1 - \beta)\nabla_w C \otimes \nabla_w C, \tag{4.14}$$

where the hyperparameter β is usually set to 0.9.

Adaptive Moment Estimation (Adam)

The standard optimizer for machine learning nowadays is Adam, which combines the advantages of momentum optimization and RMSProp [49]. The algorithm is given as

$$m' = \beta_1 m + (1 - \beta_1)\nabla_w C, \tag{4.15}$$
$$g' = \beta_2 g + (1 - \beta_2)\nabla_w C \otimes \nabla_w C, \tag{4.16}$$
$$\hat{m} = \frac{m'}{1 - \beta_1^t}, \tag{4.17}$$
$$\hat{g} = \frac{g'}{1 - \beta_2^t}, \tag{4.18}$$
$$w' = w - \frac{\eta}{\sqrt{\hat{g}} + \epsilon} \cdot \hat{m}, \tag{4.19}$$

where t corresponds to the current number of iteration when starting with iteration 1. The special update rule for $\hat{m}$ and $\hat{g}$ comes from the fact that both m and g are biased towards their initialization value 0.

4.3 Convolutional neural networks

An approach to feed images into a network with the knowledge of dense layers would be to assign a neuron to each pixel with the corresponding intensity and to convert the two-dimensional image of size $(m \times n)$ into a one-dimensional input vector of size $(m \cdot n)$. However, this reduction in dimensionality means that information about the spatial structure of the image is lost. Pixels that are far apart are treated exactly the same as those that are close together. A network architecture that can include the spatial structure in the learning process is called a convolutional neural network (CNN).

The main idea is that each pixel is assigned a neuron, but the input layer retains the two-dimensional structure of the image. The neurons from the first hidden layer are connected to the input neurons via a receptive field. Each neuron in the hidden layer has its own receptive field which is shifted across the entire imput image. The length per shift is called the stride and is a hyperparameter of the network. The size of the receptive field is called the kernel size and has often an extent of 5×5 or 3×3. An example for a kernel size of 5×5 is shown in figure 4.3. As can be seen in the figure the hidden layer has fewer neurons than the input layer. Sometimes it is preferred

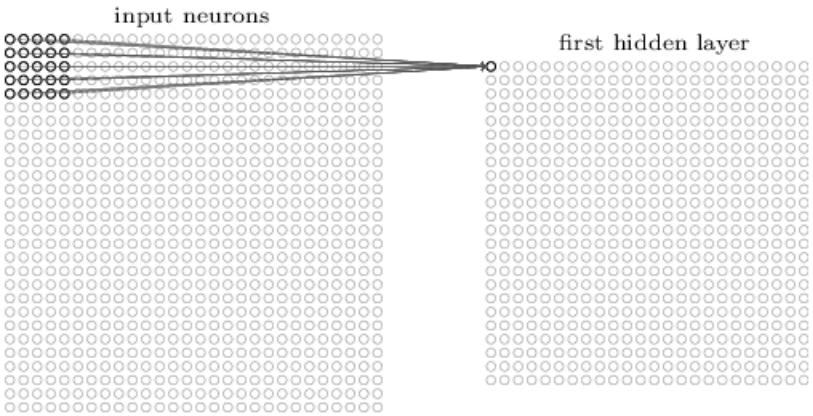


Figure 4.3: Main idea behind a convolutional layer [43].

that the number of neurons remains constant. For this purpose, additional neurons with the value 0 are inserted around the neurons of the input image. The method is called zero padding.

The reason for the high efficiency and speed of a CNN is that the weights and biases are the same for all neurons in the convolutional layer. Because of this, a single layer extracts exactly one feature from an image. In order to recognize several features of an image, further convolutional layers are stacked, leading to several feature maps. The output of a neuron in layer l at position (j, k) is defined as

$$a^l = f\left(b + \sum_{l=0}^{s}\sum_{m=0}^{p} w_{l,m} a^{l-1}_{j+l,k+m}\right), \tag{4.20}$$

where the sums run over the weights and neurons of a receptive field in layer $(l-1)$ with kernel size $(s \times p)$. Equation (4.20) can also be written as $a^l = f\left(b + w * a^{l-1}\right)$, where $*$ is the convolutional operator. Because of this operation, a CNN has far fewer parameters than a simple dense layer. The number of parameters can be reduced even further by inserting a pooling layer after a convolutional layer. A pooling layer has the same form as a convolutional layer, with a receptive field connected to the previous layer. However, the interest here is not in learning weights, rather in reducing information. A max-pooling layer for example simply takes the maximum value from the receptive field. As a result, information about the exact position of the feature is neglected. An example for a max-pooling layer with kernel size 2×2 can be seen in figure 4.4.

A complete CNN consists not only of convolutional and pooling layers, it combines these new concepts with the previous ideas of a DNN. Usually, after the last pooling layer, the neurons from different feature maps are brought into a single one-dimensional vector, this is done by a so called flattening layer. This layer is then used as an input layer for a dense layer. Based on this, different DNN models can then be used again to optimize the output.

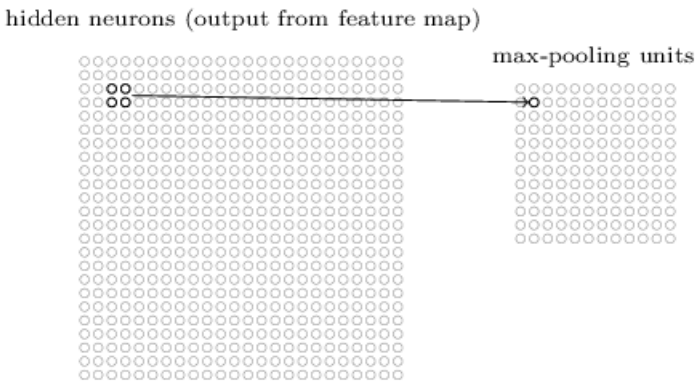


Figure 4.4: Main idea behind a pooling layer [43].

4.4 Tensorflow

As we have seen, there are many things to consider when constructing a DNN. From the most basic elements like multidimensional arrays to the implementation of optimizers and backpropagation, there is a lot of programming work involved. To simplify the construction of deep networks, François Chollet published the high-level API Keras [50]. Keras focuses on building models via a simple use of layers, but does not provide any low-level mechanics itself and therefore needs a backend program. Meanwhile only Tensorflow is accepted as backend and Tensorflow even comes with its own Keras implementation tf.keras. The core of Tensorflow is similar to NumPy, but instead of arrays the mathematical operations are executed on tf.Tensor (immutable) or tf.Variable (mutable) objects. A simple 2×2 array to store and update model weights can be defined with these code lines:

```
import tensorflow as tf

w = tf.Variable([[1., 2.],
                 [3., 4.]])
```

Tensorflow not only supports standard mathematical functions, but also offers a variety of functions that are useful for machine learning applications. These operations can easily be offloaded to GPUs and TPUs. However, the biggest advantage of Tensorflow is that, compared to NumPy, it can automatically calculate the gradient of any differentiable tensor. When calculating the gradient using tf.GradientTape, Tensorflow even records all operations on trainable parameters (tf.Variables) and can therefore use them for the backward pass in the backpropagation algorithm. Tensorflow thus offers all the necessary low-level functions so that the high-level Keras components can function properly. An example for a CNN model I used in this thesis built with Keras is:

```
from tensorflow import keras
```

```
model = keras.models.Sequential([
        keras.layers.InputLayer(input_shape=(65,65,1)),
        keras.layers.Conv2D(filters=16,
                            kernel_size=(3,3),
                            strides=(1,1),
                            activation='relu',
                            padding="valid"),
        keras.layers.MaxPool2D(pool_size=(2,2),
                               strides=(2,2)),
        keras.layers.Conv2D(filters=32,
                            kernel_size=(3,3),
                            strides=(1,1),
                            activation='relu',
                            padding="valid"),
        keras.layers.MaxPool2D(pool_size=(2,2),
                               strides=(2,2)),
        keras.layers.Flatten(),
        keras.layers.Dense(64, activation='relu'),
        keras.layers.Dense(32, activation='relu'),
        keras.layers.Dense(4, activation='linear')
        ])
```

This model takes images of size 65×65 and feeds them through two convolutional and max-pooling layers, then through two dense layers and an output layer with 4 neurons. The loss function and optimizer are then defined via

```
model.compile(optimizer=keras.optimizers.Adam(),
              loss=tf.keras.losses.MeanSquaredError())
```

and the training process can be started with:

```
model.fit(training_data, validation_data, epochs)
```

The standard values for Adam optimizer in tf.keras are $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e - 07$. It becomes clear that powerful neural networks can be built and trained with just a few lines of code. This makes Tensorflow very accessible, however, by using the the low-level API also very complex models and training algorithms can be implemented [51, 52].

5 Generating the STM images

5.1 A minimal tight-bind model for TBG

The first step of this thesis is to find a simple model for which artificial STM images can be generated easily and quickly. Since this question is very general, we decided on a model of TBG. The following toy model of TBG offers the possibility of implementing a four-band model and thus the key features of the system rather quickly. This tight-binding model was introduced by my supervisor and a justification for the basic ideas was attempted in the theory part. One of the key features of the resulting model are the valley preserving symmetries, which are listed in the following:

1. C_3 rotation perpendicular to the plane.
2. $C_{2x} : (x, y) \to (x, -y)$ two-fold in-plane rotation along the x-axis exchanging the two layers.
3. The product of time-reversal and two-fold rotation perpendicular to the plane of the system: C_2T.

The transformations in momentum space can be attained by introducing the Pauli matrices ρ and σ in sublattice and orbital space, respectively. The electron operators in real space on each site i of the honeycomb lattice for each orbital $l = 1, 2$ are then given as $c_{i,l}$. This corresponds to momentum space operator of the form $c_{\mathbf{k},l,\rho}$ with $\rho = 1, 2$ denoting the sublattice. A convenient assumption would be that the orbitals can be chosen to transform trivially under C_3 rotations. This means that there is no change of sublattice and orbital and that $\rho_z/2$ is involved in the transformation as a generator of rotations around the z-axis:

$$C_3 c_{\mathbf{k}} C_3^\dagger = \rho_0 \sigma_0 e^{-\rho_z \mathbf{a}_1 \mathbf{k}/2} c_{C_3\mathbf{k}} \tag{5.1}$$

C_{2x} also acts trivially on the sublattice space and the action on orbital space in case of Wannier obstruction has already been shown in equation (2.24) leading to the transformation

$$C_{2x} c_{\mathbf{k}} C_{2x}^\dagger = \rho_0 \sigma_x c_{C_{2x}\mathbf{k}}. \tag{5.2}$$

It has been shown with equation (2.22) that C_2T will transform one orbital into one another. As the action of this symmetry will also exhange sublattices the transformation can be written as

$$C_2T c_{\mathbf{k}} C_2T^\dagger = \rho_x \sigma_x c_{\mathbf{k}}. \tag{5.3}$$

The action of C_{2x} and C_2T on the two Wannier orbitals can be seen in figure 5.1 (a). The form of the orbitals (blue(up)/red(down)) corresponds to Wannier functions underlying the D_3 point group

$$W_\pm = \pm c_1 y(y^2 - 3x^2) - c_2(x^2 + y^2)^2, \tag{5.4}$$

where $c_{1,2}$ are constants to fine tune the form of the orbitals.

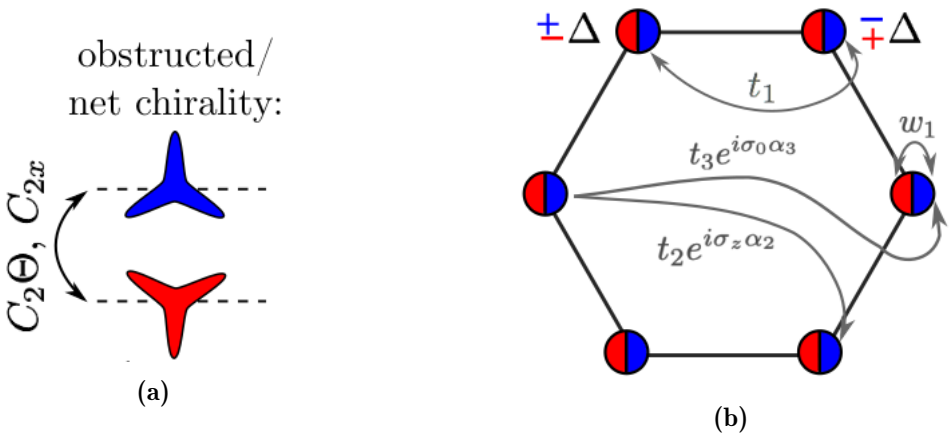


Figure 5.1: (a) Non-trivial action on the Wannier orbitals of the emergent single-valley symmetries. (b) Resulting hopping terms on the two-orbital honeycomb lattice in order to break additional symmetries. These figures were provided by my supervisor.

Since the model should be as simple as possible and should only cover symmetries from a single valley, we now want to try to break all possible symmetries which are thinkable on a honeycomb lattice. This is done by an increasing number of hopping terms leading to a Haldane Hamiltonian:

1. The nearest neighbor term t_1 can be chosen to be the usual form of monolayer graphene leading to the spectrum of equation (2.8).

2. The next-nearest neighbor term $t_2e^{i\sigma_z\alpha_2}$ can be complex in general and will break time-reversal symmetry. It has to have opposite flux for the different orbitals in order to attain a vanishing Chern number in the fully coupled model.

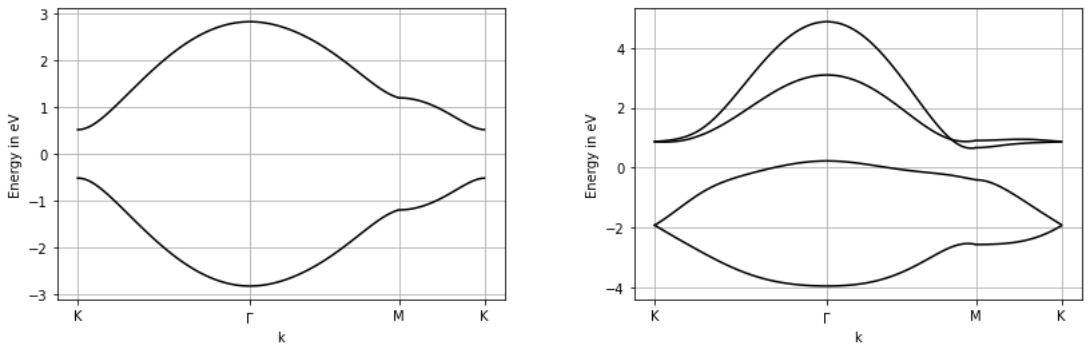
3. The third-nearest neighbor term $t_3e^{i\sigma_0\alpha_3}$ will break inversion symmetry. So far we have introduced a Haldane-like model for one of the orbitals with a non-zero Chern number. A model with the same band structure and opposite Chern number for the other orbital can then be defined via a C_2T-related copy of this Haldane, which means to send $\alpha_2 \to -\alpha_2$.

4. By applying an onsite-coupling $w_1 \sum_{\mathbf{k}} c^\dagger_{\mathbf{k}} \rho_0 \sigma_x c_{\mathbf{k}}$ Dirac cones will be induced on the coupled four-band model.

The resulting 2×2 Haldane Hamiltonian for one of the orbitals has the form

$$\begin{aligned} h_{\mathbf{k}} = & t_1 \left(\sigma_x + \sigma_x \cos\left(\mathbf{k}\mathbf{a}_1\right) - \sigma_y \sin\left(\mathbf{k}\mathbf{a}_1\right) + \sigma_x \cos\left(\mathbf{k}\mathbf{a}_2\right) - \sigma_y \sin\left(\mathbf{k}\mathbf{a}_2\right)\right) \\ & + t_2 \left(\cos(\mathbf{k}\mathbf{a}_2 - \alpha_2) + \cos(\mathbf{k}(\mathbf{a}_1 - \mathbf{a}_2) - \alpha_2) + \cos(\mathbf{k}\mathbf{a}_1 + \alpha_2)\right) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ & + t_2 \left(\cos(\mathbf{k}\mathbf{a}_2 + \alpha_2) + \cos(\mathbf{k}(\mathbf{a}_1 - \mathbf{a}_2) + \alpha_2) + \cos(\mathbf{k}\mathbf{a}_1 - \alpha_2)\right) \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \\ & + t_3 \left(\sigma_x \cos\left(\mathbf{k}(\mathbf{a}_2 - \mathbf{a}_1) + \alpha_3\right) + \sigma_y \sin\left(\mathbf{k}(\mathbf{a}_2 - \mathbf{a}_1) + \alpha_3\right)\right) \\ & + t_3 \left(\sigma_x \cos\left(\mathbf{k}(-\mathbf{a}_2 - \mathbf{a}_1) + \alpha_3\right) + \sigma_y \sin\left(\mathbf{k}(-\mathbf{a}_2 - \mathbf{a}_1) + \alpha_3\right)\right) \\ & + t_3 \left(\sigma_x \cos\left(\mathbf{k}(\mathbf{a}_1 - \mathbf{a}_2) + \alpha_3\right) + \sigma_y \sin\left(\mathbf{k}(\mathbf{a}_1 - \mathbf{a}_2) + \alpha_3\right)\right) \end{aligned} \tag{5.5}$$

where the lattice vectors $\mathbf{a}_i$ are defined according to monolayer graphene from equation (2.2), but with $a_0 = 1$. By taking a copy of this Hamiltonian for the second orbital, sending $\alpha_2 \to -\alpha_2$ and coupling those two models one gets a 4×4 Hamiltonian with four energy bands. The resulting band structure for the one-orbital Haldane and the two-orbital one are shown in figure 5.2.



(a) **(b)**

Figure 5.2: Band structure along the high-symmetry points for the one-orbital (a) and two-orbital (b) Haldane model.

5.2 Nematic order parameter

The nematic order parameter $\mathbf{\Phi} = (\phi_1, \phi_2)^T$ couples to the electrons in the general form

$$\Delta h_{\mathbf{k}} = \boldsymbol{\phi} \cdot \boldsymbol{g}_{\boldsymbol{k}} = \phi_1 g_{1,\mathbf{k}} + \phi_2 g_{2,\mathbf{k}}, \tag{5.6}$$

where the matrix-valued $g_{j,\mathbf{k}}$ should be constrained by the irreducible representations of D_3 while being invariant under C_2T:

$$C_3 : \boldsymbol{\phi} \to R_3\boldsymbol{\phi}, \;\; R_3 := \frac{1}{2}\begin{pmatrix} -1 & -\sqrt{3} \\ \sqrt{3} & -1 \end{pmatrix} \tag{5.7}$$

$$C_{2x} : \boldsymbol{\phi} \to R_{2x}\boldsymbol{\phi}, \;\; R_{2x} := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5.8}$$

$$C_2T : \boldsymbol{\phi} \to \boldsymbol{\phi} \tag{5.9}$$

To find these $g_{j,\mathbf{k}}$, one can define the pair of functions $f_{1,\mathbf{k}}$, $f_{2,\mathbf{k}}$ which are smooth and periodic on the Brillouin zone, real-valued, and obey

$$\mathbf{f}_{R_3\mathbf{k}} = R_3\mathbf{f}_{\mathbf{k}}, \;\; \mathbf{f}_{R_{2x}\mathbf{k}} = R_{2x}\mathbf{f}_{\mathbf{k}}, \;\; \mathbf{f}_{\mathbf{k}} := (f_{1,\mathbf{k}}, f_{2,\mathbf{k}})^T. \tag{5.10}$$

Focusing on the lowest lattice harmonics of this infinite-dimensional function space [40]

$$\mathbf{f}_{\mathbf{k}} = \frac{8}{3}\left(\cos(k_y) - \cos(\frac{\sqrt{3}k_x}{2})\cos(\frac{k_y}{2}), \sqrt{3}\sin(\frac{\sqrt{3}k_x}{2})\sin(\frac{k_y}{2})\right)^T, \tag{5.11}$$

and using equations (5.1)-(5.3) one can then write

$$\mathbf{g}_{\mathbf{k}} = \alpha_0\rho_0\sigma_0\begin{pmatrix} f_{1,\mathbf{k}} \\ f_{2,\mathbf{k}} \end{pmatrix} + \alpha_1\rho_0\sigma_x\begin{pmatrix} f_{1,\mathbf{k}} \\ f_{2,\mathbf{k}} \end{pmatrix} + \alpha_2\rho_0\sigma_y\begin{pmatrix} -f_{2,\mathbf{k}} \\ f_{1,\mathbf{k}} \end{pmatrix} + \alpha_3\rho_z\sigma_z\begin{pmatrix} -f_{2,\mathbf{k}} \\ f_{1,\mathbf{k}} \end{pmatrix}, \tag{5.12}$$

where α_j are some real-value expansion coefficients. For the orientations of $\boldsymbol{\Phi} = (\cos(2\theta), \sin(2\theta))^T$ there are only two possible solutions like shown in equations (2.31)-(2.32): $\theta = 0$ (and the symmetry-related $\theta = \pi/3, 2\pi/3$) and $\theta = \pi/6$ (and the symmetry-related $\theta = \pi/2, 5\pi/6$).

5.3 Computing the STM response

In order to compute the differential conductance in real space $dI/dV(\mathbf{x})$ it is necessary to compute Green's function as can be seen from equations (3.10) and (3.18). In our case of a 4×4 Hamiltonian with two Wannier orbitals per lattice site Green's function takes the form

$$G^R_{\alpha\beta}(\mathbf{R}-\mathbf{R}', \omega) = \frac{1}{V}\sum_{\mathbf{k}} e^{i\mathbf{k}(\mathbf{R}-\mathbf{R}')}G^R(\mathbf{k}, \omega), \tag{5.13}$$

$$G^R_{\alpha\beta}(\mathbf{k}, \omega) = \lim_{\eta\to0^+}\left(\frac{1}{\omega - h_{\mathbf{k}} + i\eta}\right)_{\alpha\beta}, \tag{5.14}$$

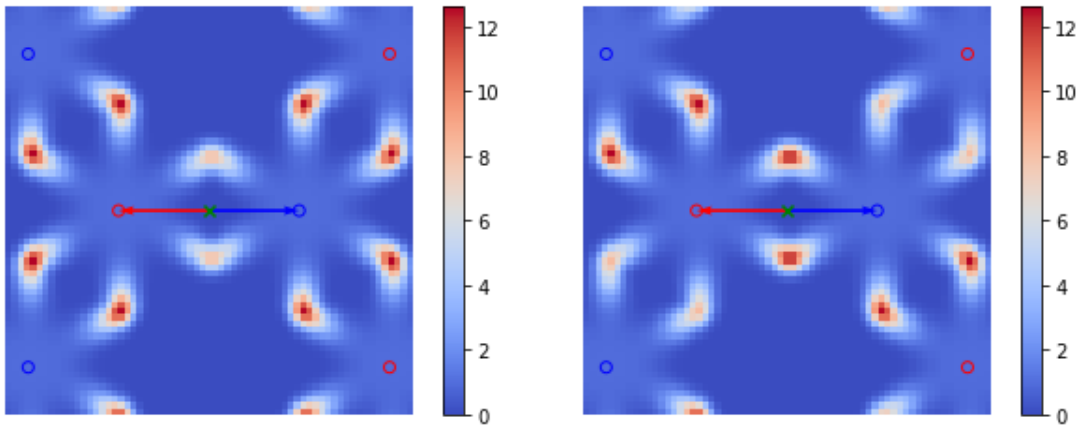
where compared to equations (3.6) and (3.8) we ignored the spin and we are on the translation invariant honeycomb lattice with Bravais lattice vectors $\mathbf{R}^{(\prime)}$. We can then

use equation (3.4) to change to a real space basis $|\mathbf{x}\rangle$

$$\frac{dI}{dV}(\mathbf{x},\omega) \propto \Im\left[\sum_{j,k,\alpha,\beta} W_{\mathbf{R}_j\alpha}(\mathbf{x}) G^R_{\alpha\beta}(\mathbf{R}_j - \mathbf{R}_k, \omega = -eV) W^*_{\mathbf{R}_k\beta}(\mathbf{x})\right], \tag{5.15}$$

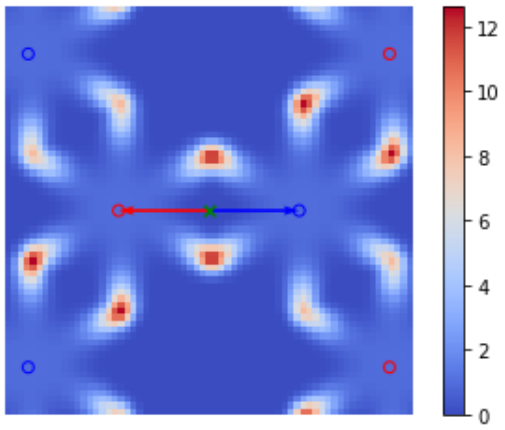
where $W_{\mathbf{R}_j\alpha}(\mathbf{x})$ are Wannier functions from equation (5.4). The indices α and β correspond to four different realizations of the Wannier functions, as each of the two orbitals $W_\pm$ can be placed on each of the two sublattices. The index $\mathbf{R}_j$ corresponds to the fact that the Wannier function is computed in several UCs, therefore leading to a function $W_{\mathbf{R}_j\alpha}(\mathbf{x}) = W_\alpha(\mathbf{x} - \mathbf{R}_j - \boldsymbol{\delta})$ with $\boldsymbol{\delta}$ corresponding to the atom basis and therefore fixing the sublattice.

With equations (5.14)-(5.15) and the previously defined 4×4 Hamiltonian we have all the ingredients to generate some artificial STM images for a four-band TBG model. Figure 5.3 shows some examples for various nematic director values at an energie of $\omega = -1\,\text{eV}$, $\eta = 0.01$, a total of $k = 2500$ vectors out of the first BZ and nematic parameters $\alpha_0 = 0.06$, $\alpha_1 = 0.04$, $\alpha_2 = 0.07$ and $\alpha_3 = 0.05$. The bond strengthening behavior of the nematic phase can be clearly seen and distinguishes the different director values. The energy was chosen to lie within the energy bands from figure 5.2 (b). The parameter η and the number of k-vectors were chosen in a way that the images can be generated quickly but also don't deviate too much from the optimal case of $k \to \infty$. The nematic parameters were sampled from the interval $(0.01, 0.1)$. The choice on the lower boundary comes from the fact that the nematic order can already be seen well at a value of $\alpha_j = 0.01$.

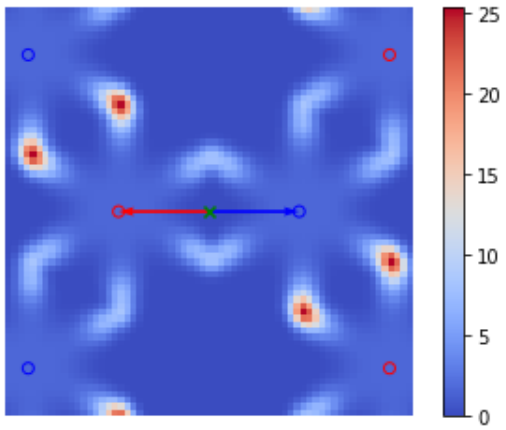


(a) $\theta = 0$

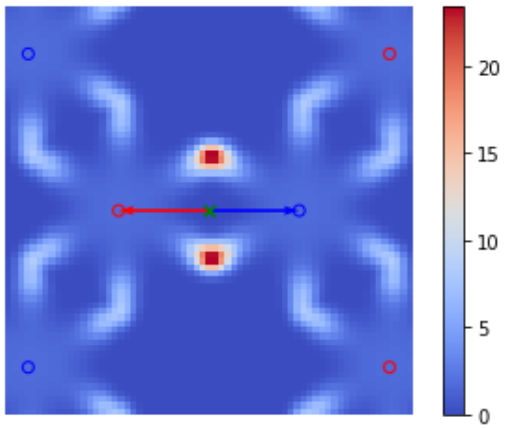
(b) $\theta = \pi/3$



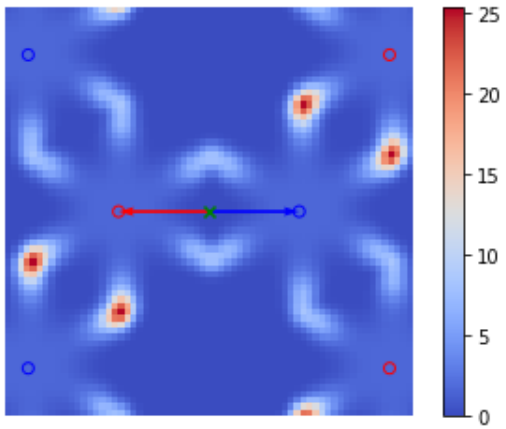
(c) $\theta = 2\pi/3$



(d) $\theta = \pi/6$



(e) $\theta = \pi/2$



(f) $\theta = 5\pi/2$

Figure 5.3: STM plot at an energie of $\omega = -1\,\text{eV}$ for various values of the nematic director θ. The blue and red circles correspond to the two sublattices A/B. The green cross marks a Bravais lattice point and the arrows are the basis vectors pointing to the two atoms.

6 ML results

6.1 Orientation of nematic director

6.1.1 Discrete director values

To start and test whether a neural network is capable of recognizing meaningful structures in the artificially generated STM images, a simple case with a simple network should first be examined. Before examining the role of the nematic parameters α_j, we first want to examine the nematic director θ. Although the director can take continuous values when neglecting crystal anisotropies, the simplest case corresponds to the three discrete solutions $\theta = 0, \pi/3, 2\pi/3$ (or $\theta = \pi/6, \pi/2, 5\pi/6$). So the first task for an ANN is to predict the correct class for θ. Since this task looks relatively simple at first glance and is even possible for human eyes in the most cases, I chose a simple structure of three layers for the network:

1. An input layer, which consists of 4,225 neurons due to an image size of 65×65.
2. A hidden layer with 512 neurons and ReLU activation function.
3. An output layer with 3 sigmoid neurons.

The output value of the sigmoid neurons is interpreted as a probability value of belonging to one of the three classes (directions). In total there are 2,165,251 trainable parameters in the network. Adam is chosen as an optimizer and the cost function to be optimized is the MSE from equation (4.5). The optimizer and cost function will remain the same for the rest of all machine learning tasks. A total of 39,500 sample images were generated, of which 33,000 serve as training data, 6,000 as validation data and 500 as test data. The images were sampled at random energies between $-4\,\mathrm{eV}$ and $4\,\mathrm{eV}$ and with random $\alpha_j \in (0.01, 0.1)$ configurations. The results of the trained network are very promising for more sophisticated tasks. Out of the 500 images in the test dataset, only 7 were incorrectly classified. These are images on which no clear structure can be seen even for the human eye. All other images have been classified correctly, two examples can be seen in figure 6.1.

6.1.2 Continous director values

In general, the values of the director θ need not be discrete. A more realistic choice would therefore be $\theta \in (0, \pi)$. In this case, the human eye can no longer easily extract

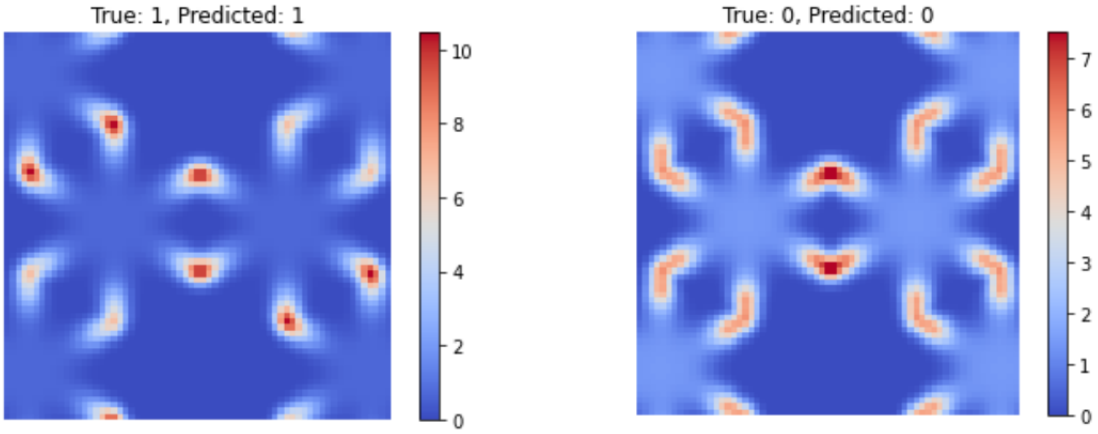


Figure 6.1: Example of two correctly classified STM images in the case of discrete director values. Class 0 corresponds to $\theta = 0$, class 1 to $\theta = \pi/3$.

the exact value of the nematic director. This increasing complexity of the problem should also be taken into account when selecting the neural network:

1. A convolutional layer with 16 filters, a kernel size of 3×3, a stride of one and a ReLU activation function. As no padding was used the layer dimensions are $(63, 63, 16)$.

2. A max-pooling layer with size 2×2 and stride 2 which halves the size of the image.

3. A convolutional layer with 32 filters, a kernel size of 3×3, a stride of one and a ReLU activation function. As no padding was used the layer dimensions are $(29, 29, 32)$.

4. A max-pooling layer with size 2×2 and stride 2.

5. A layer that flattens the pooling layer of size $(14, 14, 32)$ resulting in 6,272 input neurons for the following dense layer.

6. A dense layer with 64 neurons and ReLU activation function.

7. A dense layer with 32 neurons and ReLU activation function.

8. An output layer with 1 neuron and linear activation function.

This CNN has a total of 408,385 parameters. Here one can see one of the main advantages of CNNs in comparison to simple DNNs with solely dense layers: Although the network is very complex and has many layers, the number of parameters is smaller than in the network with only one dense layer from the previous task. Here a total of 62,000 sample images were generated, of which 39,500 serve as training data, 10,000

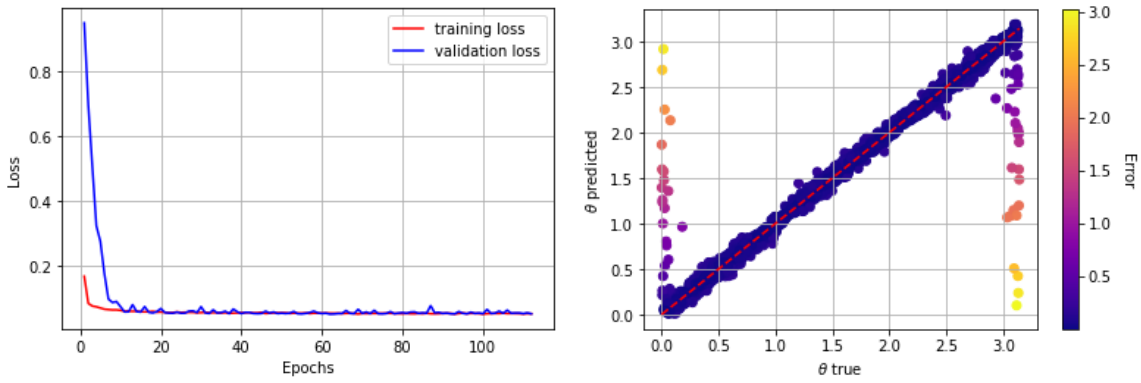


Figure 6.2: Figure (a) shows the course of the loss function (MSE) for the training and validation set over the number of trained epochs. Already from 20 epochs the error is already very small and needs about 100 more epochs to converge. Figure (b) shows the true values of the nematic director plotted against the predictions from the network. The coloring of the datapoints simply corresponds to the error $|\theta_{pred} - \theta_{true}|$ of every single point.

as validation data and 2,500 as test data. The images were sampled at a fixed energy of $-1\,\mathrm{eV}$ and with random $\alpha_j \in (0.01, 0.1)$ configurations. In figure 6.2 (a) one can see the evolution of the MSE over the training epochs for the training data set and the validation set. It can be seen that the error of the training set reaches a very small value after just a few epochs, whereas the error of the validation set is still a few epochs behind. In the end, however, both sets converge in the similar error range. It should be noted that for all machnine learning training tasks I used an early stopping callback with a patience of 50 epochs. This means that the training stops as soon as the loss in the validation dataset does not change for 50 epochs. For this reason, the point of convergence is already at about 60 epochs in this case. Since an epoch lasts only a few seconds due to the small number of parameters, the network trains extremely quickly. Since these learning curves plots always look very similar for successful trainings, even on different tasks, I'll leave this one as representative example, but I'll refrain from such plots for the next results. In figure 6.2 (b) one can see that the network does not have to be improved further. The predictions of the nematic director θ on the test dataset for values other than 0 and π are already very close to the true values with which the STM images were generated. Since the values of 0 and π correspond to the same point on the unit circle, it is not surprising that the network reaches its limitations here.

The function keras.layers.GaussianNoise was used to check the reliability of the predictions. It lays a Gaussian noise over the training and validation data right at the beginning of the training process. In our case it is used as a fast method to check if precise statements can still be made about the test data set, which is left exactly the same without Gaussian noise. Here, too, it could be shown that the network delivers equally good results. The result is shown in figure 6.3.

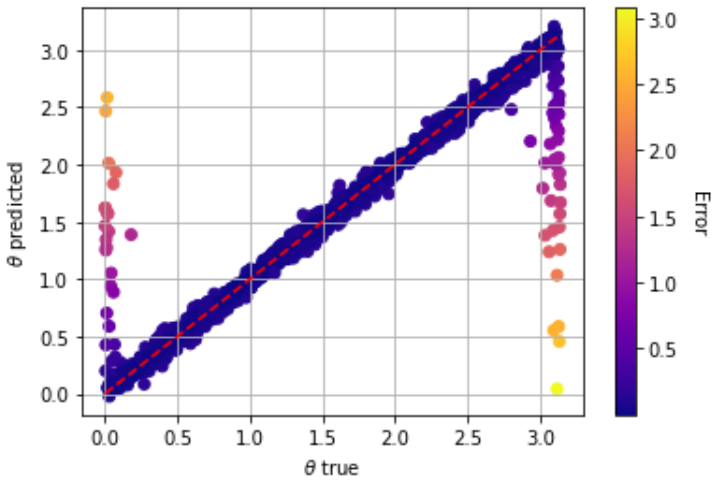


Figure 6.3: The true values of the continous nematic director plotted against the predictions from the network. The coloring of the datapoints simply corresponds to the error of every single point.

6.2 Form of nematicity

6.2.1 All nematic parameters

Next, attention should now be drawn to the form of nematicity. It should be investigated whether it is possible for a CNN to extract the nematic parameters α_j from the STM images. Exactly the same network should be used as for the case of the continuous director. The only difference is that there are now four output neurons, slightly increasing the total parameter count to 408,484. A total of 20,000 sample images were generated, of which 15,700 serve as training data, 3,000 as validation data and 1,300 as test data. The images were sampled at a fixed energy of $-1\,\mathrm{eV}$ with a fixed nematic director $\theta = 5\pi/6$ and with random $\alpha_j \in (0.01, 0.1)$ configurations. The results for the predictions on the test dataset are shown in figure 6.4. As can be clearly seen, the network does not succeed in making satisfactory predictions for any of the α_j over the entire parameter space. Especially for α_3 there seems to be no learning process at all. Furthermore, attempts were made to improve the results with deeper networks and a far higher number of filters and neurons in the dense layers. In addition, common machine learning methods such as batch normalization and dropout layers were used. However, after a large number of variations of the network, no fundamental improvement could be detected. Also a change in the image size and an increased number of images could not produce any better results. It must therefore be assumed that the problem of solving all four parameters at once is too complex for the currently sampled dataset. There are now two ways to further test this assumption: First, to reduce the complexity of the problem by solving for fewer nematic parameters. Secondly, increasing the quality of the training data, for

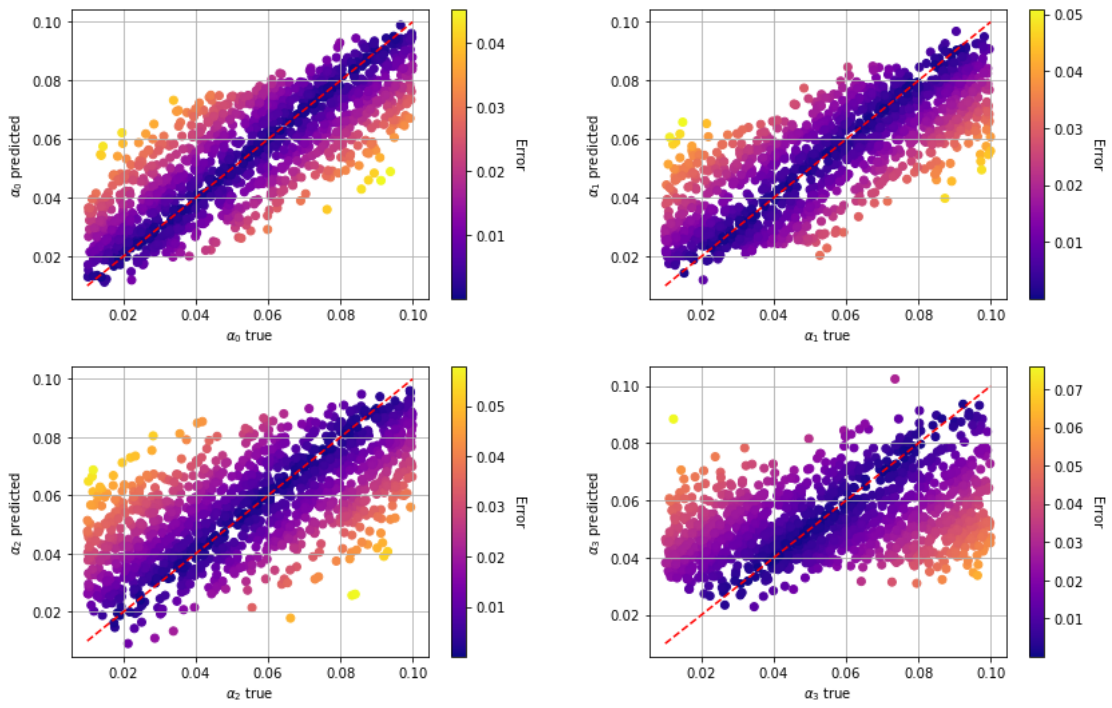


Figure 6.4: The plots show the true values of the nematic parameters plotted against the predictions from the network. In this case the network was trained with only one energy channel, leading to unsatisfying predictions. The coloring of the datapoints simply corresponds to the error of every single point.

example by sampling at several energies and thus increasing the number of channels in the network.

6.2.2 Only two nematic parameters

First, the inverse problem should be simplified by only determining two parameters, namely α_0 and α_3. We again use the same CNN and sample images at the same conditions like in the previous tasks. A total of 9,600 sample images were generated, of which 6,100 serve as training data, 2,400 as validation data and 1,100 as test data. As can be seen in figure 6.5, the predictions for α_0 are already on the right track, however, the predictions for α_3 remain at an unacceptable level. We can therefore say with good conviction that the problem is in principle too complex to be solved with samples at just one energy.

6.3 Adding more energy channels

Regardless of the complexity of the neural network, the STM samples provided are not sufficient to be able to make accurate predictions about the nematic parameters.

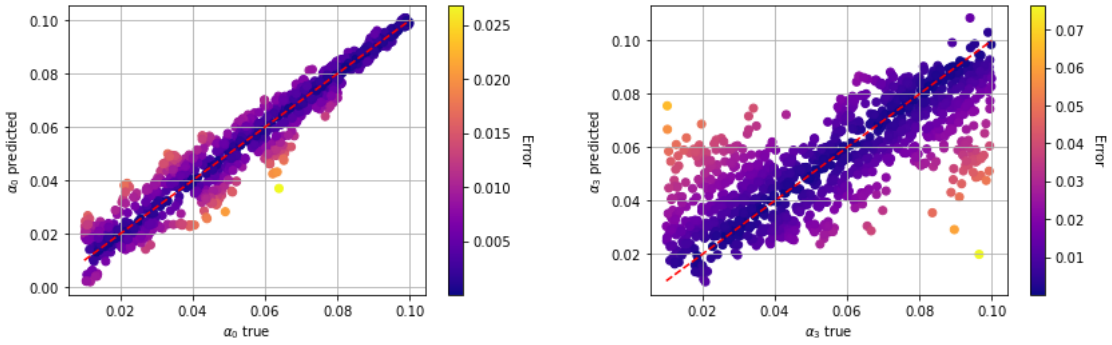


Figure 6.5: The plots show the true values of the nematic parameters α_0 and α_3 plotted against the predictions from the network. Even a simplification of the problem by reducing the amount of nematic parameters doesn't lead to accurate predictions. The coloring of the datapoints simply corresponds to the error of every single point.

For this reason, the variety of the training samples should be increased by creating plots not only at an energy $-1\,\mathrm{eV}$, but at four different energies $(-2, -1, 1, 2)\,\mathrm{eV}$. Consequently, for a randomly sampled α_j combination at a director angle of $\theta = 5\pi/6$, there are four different images. These four different images can then be transferred in parallel to Tensorflow for training. First, each energy channel is trained separately using the same CNN model that we have used before. The only difference is that at the end of each channel there is no output layer, but the last dense layers of the four channels are concatenated and fed again through one dense layer with 128 neurons and ReLU activation, with an output layer again at the end. The network model is shown in figure 6.6 and has a total amount of 1,650,436 trainable parameters.

6.3.1 Only two nematic parameters

First we want to check whether the training of the simpler variant with only two parameters can be improved by the additional energy channels. With the 3 additional energies, we no longer have 9,600 images in total, but per channel. The allocation key to training, validation and test data has remained the same. The results for the predictions on the test dataset with the four-channel model can be seen in figure 6.7. It is clear to see that this CNN has perfectly learned the intricacies of STM images. The deviation of the predictions from the true values is only extremely minimal, even the predictions of the parameter α_3, which was previously a lot more difficult to learn, are only just a tiny bit less accurate than their partner's. With these encouraging results, we can now extend to the most general case with all four parameters.

6.3.2 All nematic parameters

Like in the case of only two nematic parameters α_j the total amount of images is now 4 times larger compared to the single-channel model, which means that 60,000

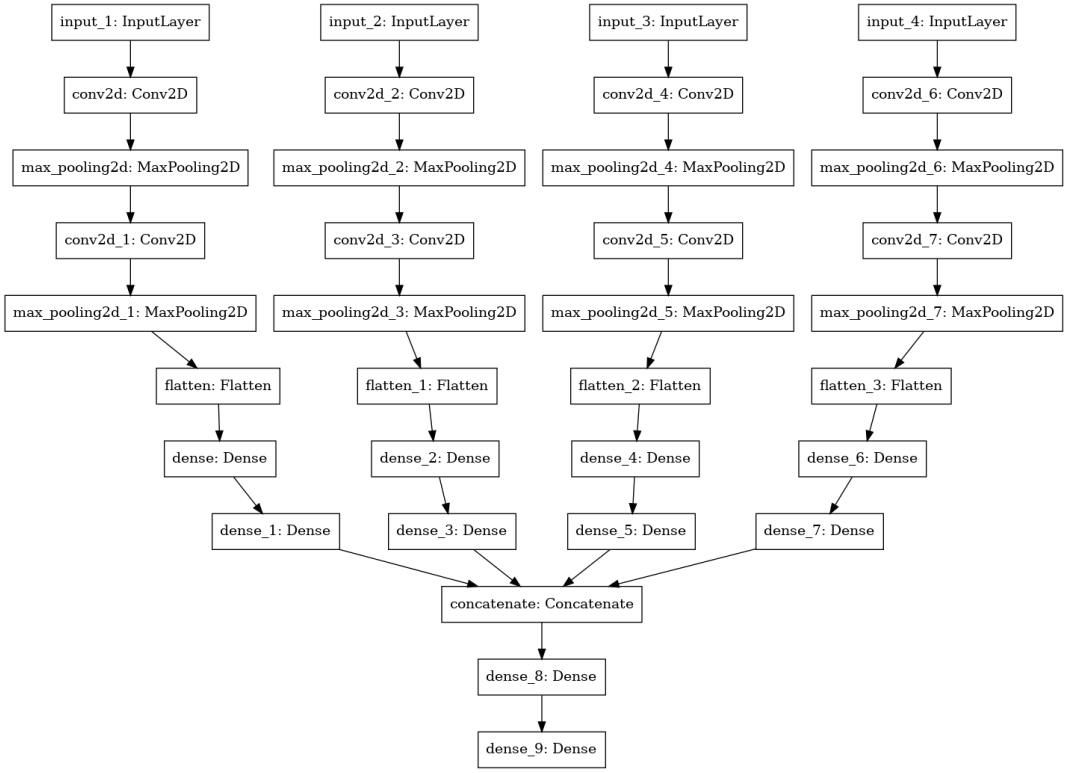


Figure 6.6: The CNN model has been expanded to include three more energy channels, resulting in additional layers in the network to bring the channels together. This image was generated with the Keras funtion keras.utils.plot_model().

images are now involved in the training and testing process. The loss function has converged after approximately 300 epochs and the results are shown in figure 6.8. It can be clearly seen that in the case of four energy channels the network is able to make meaningful predictions about the test dataset. In contrast to the single channel problem, predictions involving α_2 are now the most unreliable. But even this parameter can be determined much better than any other parameter in the single channel case. Nevertheless, we don't want to be completely satisfied here and add another possibility for a channel.

6.4 Adding LDOS at a single point

We have seen that an increase in energy channels leads to a significant improvement in the results. The question now remains as to how we can continue to take this into account. As the pixels of the STM image can be viewed as a local LDOS at fixed energy, let us now focus on the LDOS at a single STM point at different energies. Such an LDOS(ω) leads to a simple graph, as shown in figure 6.9 (a). A honeycomb lattice point, namely a sublattice point A, was selected as the fixed point of this plot.

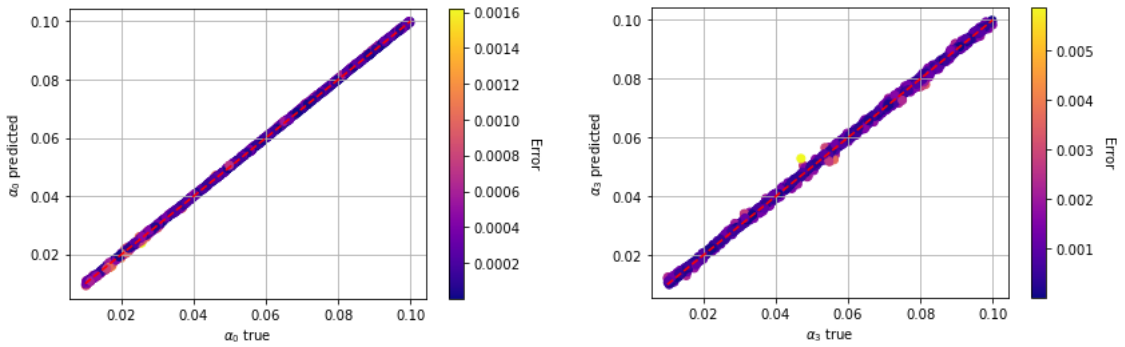


Figure 6.7: This plots shows the true values of the nematic parameters α_0 and α_3 plotted against the predictions from the network. An increase to a total of four energy channels leads to almost perfect predictions. The coloring of the datapoints simply corresponds to the error of every single point.

A continuous wavelet transform (CWT) is recommended as a method of converting such a plot into an image that can be evaluated by a CNN in the same way as the STM image channels. A CWT produces images, so-called scaleograms, which can easily be passed on to CNNs. A CWT works in a similar way to a Fourier transform, with the main difference that accuracy in the time domain is preserved. This method has been used successfully by Ref. [10]. According to this reference, a CWT of a discrete 1D dataset (the LDOS at any energy in our case) with size N_f can be written as

$$\tilde{f}(t,s) = \frac{1}{\sqrt{s}} \sum_{i=0}^{N_f-1} f_i \psi \left(\frac{(i-t)\Delta}{s} \right), \tag{6.1}$$

$$\{f_0, f_1, ..., f_{N_f-1}\} = \{f(t_{min}), f(t_{min}+\Delta), ..., f, (t_{max})\} \tag{6.2}$$

where $\psi(t)$ is the so-called mother wavelet function. This funtion is translated by a parameter t and scaled by a parameter s. In our case we stick to the choice made by Ref. [10] and also choose the real Morlet form

$$\psi(t) = e^{-t^2/2} \cos(5t). \tag{6.3}$$

However, I did not implement the function myself, but resorted to the PyWavelets package. With this package, corresponding scaleograms with the size 65×65 were created for 5,000 alpha configurations in the previous training dataset, for 2,000 alpha configurations in the validation dataset and for 1,000 configurations in the test dataset. As each alpha configuration comes with four energy channels and now also a scaleogram channel, we have a total of 40,000 images. These images can then be fed into the same CNN shown in figure 6.6 with an additional scaleogram channel. An example of a scaleogram that can run through this channel is shown in figure 6.9 (b). The resulting predictions of the nematic parameters are shown in figure 6.10. As it turns out, the scaleograms of the LDOS images are the missing piece of the puzzle to

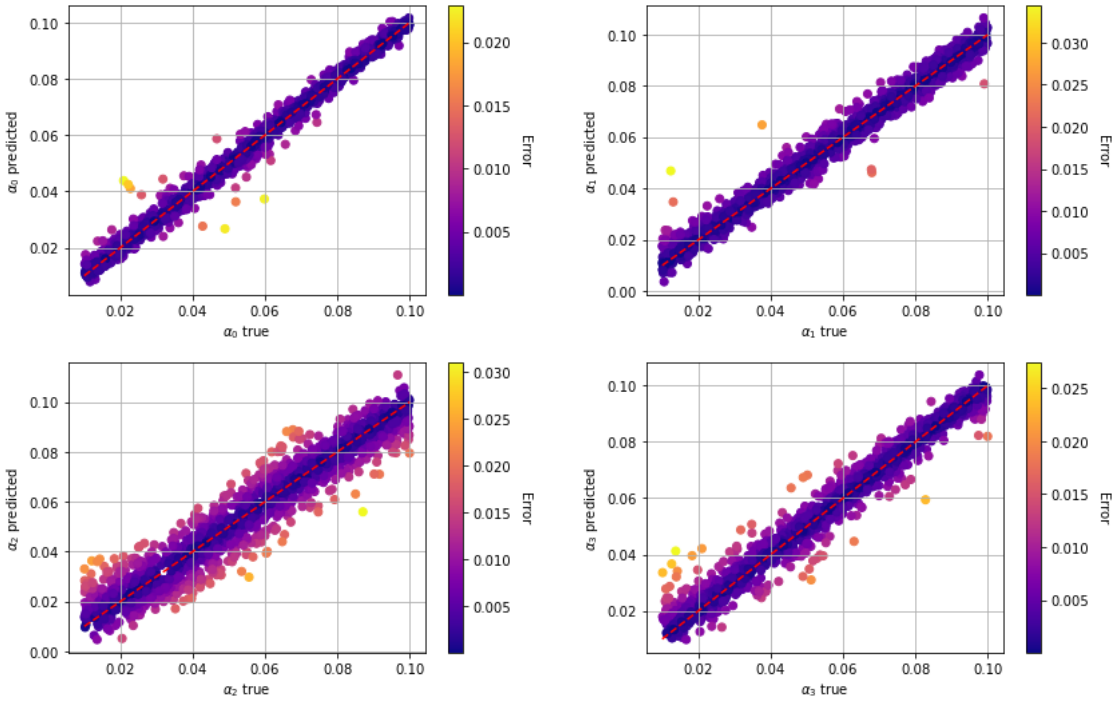
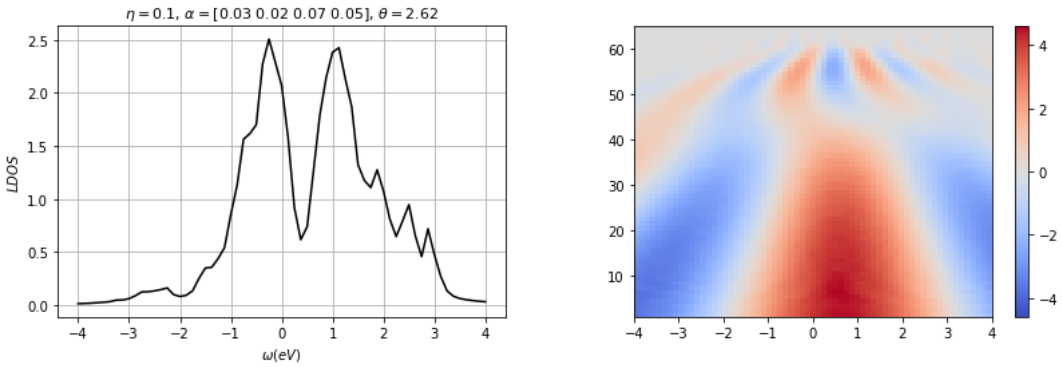


Figure 6.8: The plots show the true values of the nematic parameters plotted against the predictions from the network. Additional energy channels lead to a significant improvement in the accuracy of the predictions. The coloring of the datapoints simply corresponds to the error of every single point.

create a nearly perfectly trained network. In order to verify the results even further, we want to take into account the fact that STM images and scaleograms aren't inherently perfect and are always subject to noise. For this purpose, the LDOS images are generated again, but this time with a small contribution of Gaussian noise with a standard deviation of 0.05. Figure 6.11 shows the difference between a LDOS plot without (left) and the same image with (right) noise. Figure 6.12 shows an STM image with Gaussian noise with the same standard deviation. The training process was now repeated with the adapted images. The results in figure 6.13 show that even with Gaussian noise the results are more accurate than for the case with no scaleogram channel and no noise. We can therefore say that an increase in the energy channels in combination with an LDOS(ω) channel solves our inverse problem with sufficient accuracy. Since each additional energy channel requires additional effort in the generation of STM images, especially in the experimental case, it would be interesting to know whether a single scaleogram channel is already sufficient for precise results.



(a) (b)

Figure 6.9: Figure (a) shows the LDOS(ω) at a fixed lattice point. The plot was generated with $\eta = 0.1$, a fixed configuration of the nematic parameters and director and 100 k-vectors. The amount of k-vectors was drastically reduced to increase the speed of numerical calculations. Figure (b) shows the corresponding scaleogram. The transformation was done with a continous wavelet transform, where the mother wavelet function has the real Morlet form and the scale was chosen to match the size of the STM images.

6.5 Only LDOS at a single point

In order to illustrate the enormous influence of the LDOS(ω), the network is now trained again, but only with a single scaleogram channel. The scaleograms remain the same from the previous task, hence with an underlying Gaussian noise. In figure 6.14 it can be clearly seen that the scaleogram channel alone is sufficient to produce good results for the network. Although the STM images contribute to further improvement, they are not nearly as important as the scaleograms. Hence when deciding whether to increase accuracy with additional energy channels or a scaleogram channel, these results strongly suggest that the LDOS(ω) channel takes precedence. With available resources and time, the accuracy can then be improved by adding energy channels.

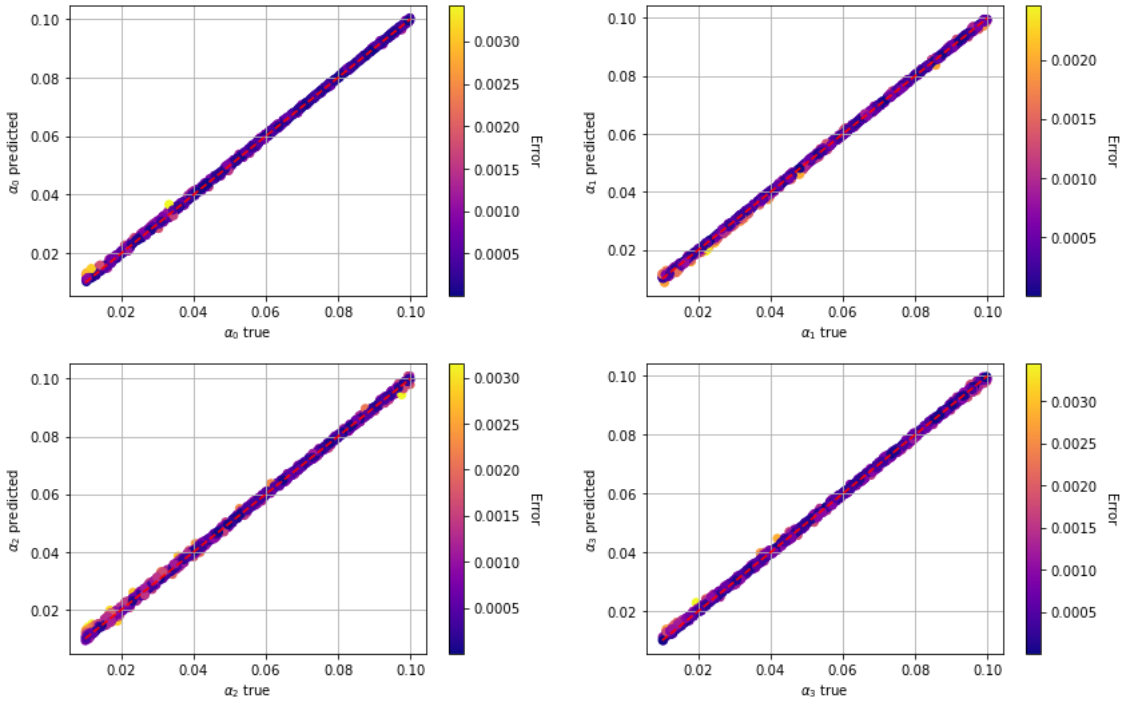


Figure 6.10: The plots show the true values of the nematic parameters plotted against the predictions from the network. A combination of energy channels with one LDOS(ω) channel leads to nearly perfect predictions of the network. The coloring of the datapoints simply corresponds to the error of every single point.

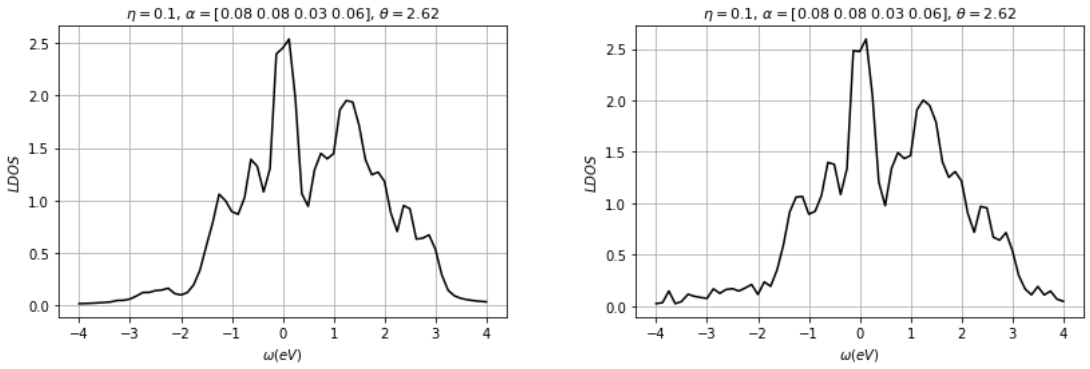


Figure 6.11: LDOS(ω) at a fixed lattice point with randomly sampled nematic parameters and director value without (left) and with Gaussian noise (right) with a standard deviation of 0.05.

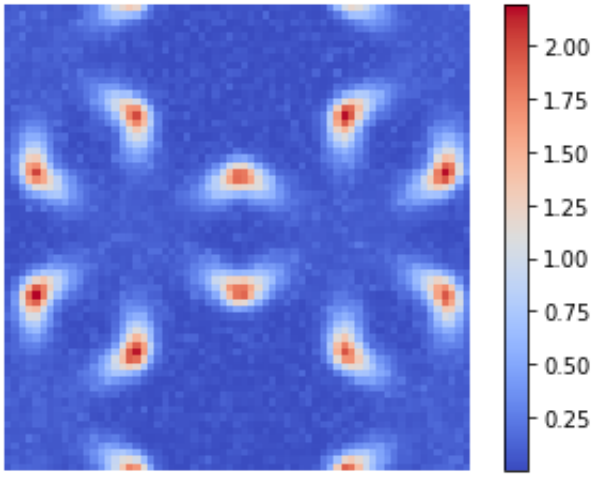


Figure 6.12: A STM plot overlayed with Gaussian noise with a standard deviation of 0.05.

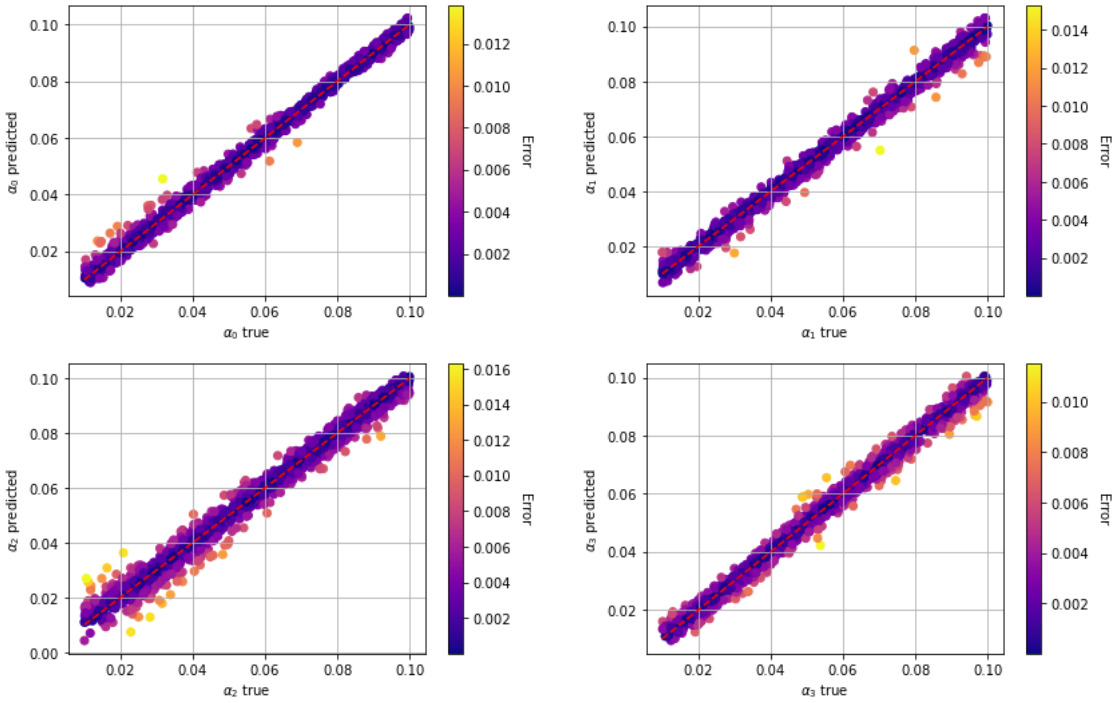


Figure 6.13: The plots show the true values of the nematic parameters plotted against the predictions from the network. Even with a Gaussian noise added to the STM plots and to the LDOS(ω) plots the predictions remain at a high accuracy level. The coloring of the datapoints simply corresponds to the error of every single point.

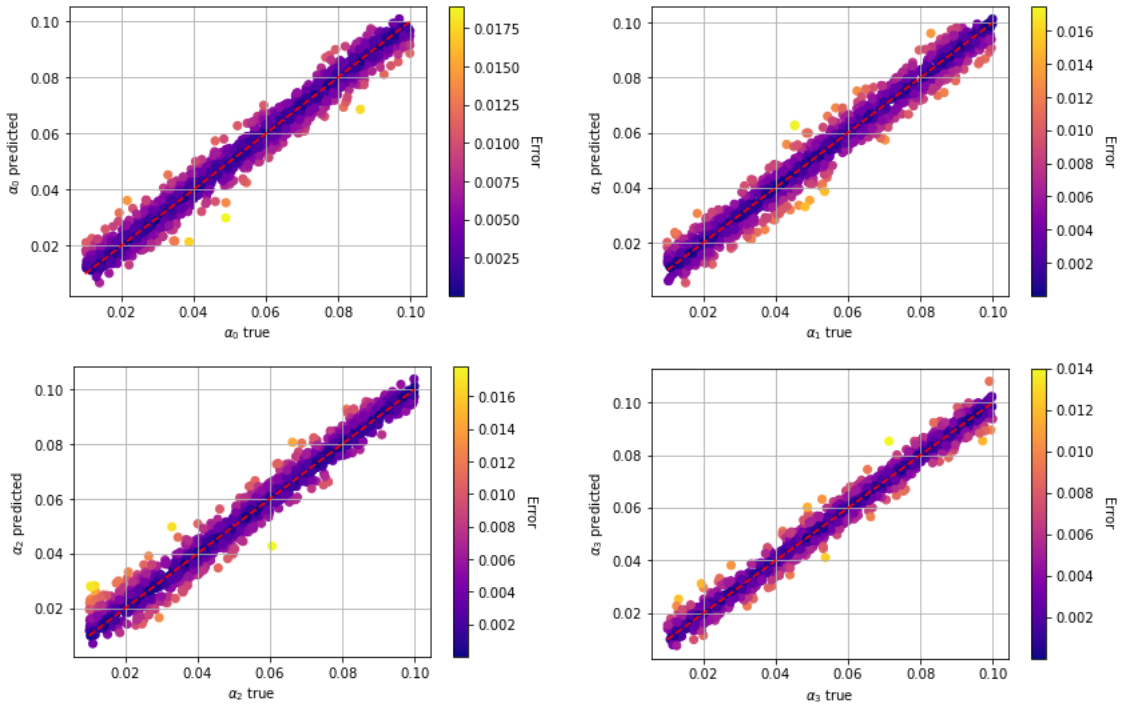


Figure 6.14: The plots show the true values of the nematic parameters plotted against the predictions from the network. A single scaleogram channel of the LDOS(ω) plots is sufficient to yield very good predictions, even in the case of underlying Gaussian noise. The coloring of the datapoints simply corresponds to the error of every single point.

7 Conclusion and outlook

We studied the ability of deep neural networks to analyze artificial STM images in search for the underlying parameters of the effective Hamiltonian. As it turns out the accuracy of this task depends on the variety of images feeded to the network. A single channel of STM images generated at one energy isn't sufficient to solve for the nematic parameters, however a combination of multiple energy channels with one scaleogram channel of the LDOS leads to satisfying results of the networks predictions.

We showed this by starting the thesis with a review of the current theory for monolayer graphene which in turn created a foundation for the description of twisted bilayer graphene. The most important properties for TBG were presented and a description of the symmetries was given, which then allowed us to better understand the model proposed by my supervisor. On the basis of this model and the connection of an STM image to the density of states of the material, we were then able to successfully produce STM images on the computer. These images were then examined by neural networks in the main part of my work. It could be shown that the task of predicting the nematic director with only discrete values is feasible for a simple ANN with only one hidden layer. The introduction of convolutional layers then also makes it possible to determine continuous director values, which is no longer an easy task for the human eye. This network architecture is then used to determine the values of the nematic parameters. As it turns out, this is not easily possible for samples at one energy and alternatives must be considered. One of them is to introduce additional energy channels and indeed this shows a significant improvement in the results. But the best idea seems to be to introduce an additional channel that measures the LDOS at a single lattice point and is fed into the network as a scaleogram via a continuous wavelet transform. Very accurate results can be produced with this methodology and the additional energy channels can even be dispensed with.

The next natural step would now be to apply the network architectures and methodology we have been working on to experimental STM images, in the best case even for TDBG. This is not possible with the model we used, since it is a minimal tight-binding model, but the entire physics can only be mapped with a continuum model. In the case of a continuum model for TDBG, particular attention must be paid to the fact that the nematicity can be broken on different scales. It has been argued in Ref. [40] that a breaking of the C_3 symmetry can happen either on the atomic scale of the graphene layers, which is called graphene nematicity, or on the scale of the superlattice, which is called moiré nematicity. These two cases are differentiated in order to clarify whether the nematic phase is actually tied to the correlated electron physics in the flat bands or due to an instability of the underlying bilayer graphene.

This distinction has important consequences for the formulation of the respective inverse problem, since the nematic parameters have to be formulated in different ways. As example in the case of intravalley graphene nematicity the number of nematic parameters increases to 10, while a description for moiré nematicity even gets by with only 2 parameters. In order to train a network for the form of nematicity or the nematic director under this large number of parameters, it is necessary to set reasonable values for the other parameters. Hence it will not be easy to find values that are also suitable for a generalization to experimental data. It must also be clarified which of the parameters are to be trained at all and for which parameters constant values can be assumed. Additionally when more parameters are randomly sampled at the image generation it will be more difficult for a network to provide meaningful predictions for all of them. If these issues can be answered and the methodology from this work can be successfully applied to a continuum model for TDBG, applying a trained model to experimental data will be the next complicated task. An important point here will definitely be the selection of the STM image area feeded into the network. Cropping experimental STM images to match the artificial STM images or vice versa in a way that the network can properly work is certainly not easy.

There are undoubtedly new challenges with a generalization to TDBG, however it isn't bold to expect a positive outcome of it. We have seen that deep neural networks are able to solve highly complex problems, often in an astonishing good way. I think there is no reason why the methodology presented here should not overcome even higher barriers, again leading to positive surprises.

Bibliography

[1] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91:045002, Dec 2019.

[2] Edwin Bedolla, Luis Carlos Padierna, and Ramón Castañeda-Priego. Machine learning for condensed matter physics. *Journal of Physics: Condensed Matter*, 33(5):053001, nov 2020.

[3] Juan Carrasquilla. Machine learning for quantum matter. *Advances in Physics: X*, 5(1):1797528, jan 2020.

[4] Ce Wang, Haiwei Li, Zhenqi Hao, Xintong Li, Changwei Zou, Peng Cai, Yayu Wang, Yi-Zhuang You, and Hui Zhai. Machine learning identification of impurities in the STM images. *Chinese Physics B*, 29(11):116805, nov 2020.

[5] Jeremy B. Goetz, Yi Zhang, and Michael Lawler. Detecting nematic order in STM/STS data with artificial intelligence. *SciPost Physics*, 8(6), jun 2020.

[6] William Taranto, Samuel Lederer, Youngjoon Choi, Pavel Izmailov, Andrew Gordon Wilson, Stevan Nadj-Perge, and Eun-Ah Kim. Unsupervised learning of two-component nematicity from stm data on magic angle bilayer graphene, 2022.

[7] Frédéric Joucken, John L. Davenport, Zhehao Ge, Eberth A. Quezada-Lopez, Takashi Taniguchi, Kenji Watanabe, Jairo Velasco, Jérôme Lagoute, and Robert A. Kaindl. Denoising scanning tunneling microscopy images with machine learning, 2022.

[8] Mani Valleti, Qiang Zou, Rui Xue, Lukas Vlcek, Maxim Ziatdinov, Rama Vasudevan, Mingming Fu, Jiaqiang Yan, David Mandrus, Zheng Gai, and Sergei V. Kalinin. Bayesian learning of adatom interactions from atomically-resolved imaging data, 2020.

[9] Yuhang Jiang, Xinyuan Lai, Kenji Watanabe, Takashi Taniguchi, Kristjan Haule, Jinhai Mao, and Eva Y. Andrei. Charge order and broken rotational symmetry in magic-angle twisted bilayer graphene. *Nature*, 573(7772):91–95, jul 2019.

[10] Noah F. Berthusen, Yuriy Sizyuk, Mathias S. Scheurer, and Peter P. Orth. Learning crystal field parameters using convolutional neural networks. *SciPost Phys.*, 11:011, 2021.

[11] Yi Hong Teoh, Marina Drygala, Roger G Melko, and Rajibul Islam. Machine learning design of a trapped-ion quantum spin simulator. *Quantum Science and Technology*, 5(2):024001, jan 2020.

[12] Saientan Bag and Rituparno Mandal. Interaction from structure using machine learning: in and out of equilibrium. *Soft Matter*, 17:8322–8330, 2021.

[13] Xiao-Han Wang, Pei Shi, Bin Xi, Jie Hu, and Shi-Ju Ran. Deep machine learning reconstructing lattice topology with strong thermal fluctuations, 2022.

[14] Simiao Ren, Ashwin Mahendra, Omar Khatib, Yang Deng, Willie J. Padilla, and Jordan M. Malof. Inverse deep learning methods and benchmarks for artificial electromagnetic material design, 2021.

[15] A.V. Rozhkov, A.O. Sboychakov, A.L. Rakhmanov, and Franco Nori. Electronic properties of graphene-based bilayer systems. *Physics Reports*, 648:1–104, aug 2016.

[16] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim. The electronic properties of graphene. *Rev. Mod. Phys.*, 81:109–162, Jan 2009.

[17] P. R. Wallace. The band theory of graphite. *Phys. Rev.*, 71:622–634, May 1947.

[18] Gonçalo Catarina, Bruno Amorim, Eduardo V. Castro, Eduardo V. Castro, Eduardo V. Castro, João M. V. P. Lopes, João M. V. P. Lopes, and Nuno Peres. Twisted bilayer graphene: Low-energy physics, electronic and optical properties, jun 2019.

[19] Eva Y. Andrei and Allan H. MacDonald. Graphene bilayers with a twist. *Nature Materials*, 19(12):1265–1275, nov 2020.

[20] Jérôme Cayssol. Introduction to dirac materials and topological insulators. *Comptes Rendus Physique*, 14(9-10):760–778, nov 2013.

[21] F. D. M. Haldane. Model for a quantum hall effect without landau levels: Condensed-matter realization of the "parity anomaly". *Phys. Rev. Lett.*, 61:2015–2018, Oct 1988.

[22] Doru Sticlet and Frédéric Piéchon. Distant-neighbor hopping in graphene and haldane models. *Physical Review B*, 87(11), mar 2013.

[23] J. L. Mañes, F. Guinea, and María A. H. Vozmediano. Existence and topological stability of fermi points in multilayered graphene. *Physical Review B*, 75(15), apr 2007.

[24] Jonathan Atteia. *Topology and electronic transport in Dirac systems under irradiation.* Theses, Université de Bordeaux, December 2018.

[25] Rafi Bistritzer and Allan H. MacDonald. Moiré bands in twisted double-layer graphene. *Proceedings of the National Academy of Sciences*, 108(30):12233–12237, jul 2011.

[26] Yuan Cao, Valla Fatemi, Shiang Fang, Kenji Watanabe, Takashi Taniguchi, Efthimios Kaxiras, and Pablo Jarillo-Herrero. Unconventional superconductivity in magic-angle graphene superlattices. *Nature*, 556(7699):43–50, mar 2018.

[27] Yuan Cao, Valla Fatemi, Ahmet Demir, Shiang Fang, Spencer L. Tomarken, Jason Y. Luo, Javier D. Sanchez-Yamagishi, Kenji Watanabe, Takashi Taniguchi, Efthimios Kaxiras, Ray C. Ashoori, and Pablo Jarillo-Herrero. Correlated insulator behaviour at half-filling in magic-angle graphene superlattices. *Nature*, 556(7699):80–84, mar 2018.

[28] Allan H. MacDonald. Bilayer Graphene's Wicked, Twisted Road. *Physics Online Journal*, 12:12, May 2019.

[29] Liujun Zou, Hoi Chun Po, Ashvin Vishwanath, and T. Senthil. Band structure of twisted bilayer graphene: Emergent symmetries, commensurate approximants, and wannier obstructions. *Phys. Rev. B*, 98:085435, Aug 2018.

[30] Pilkyung Moon and Mikito Koshino. Optical absorption in twisted bilayer graphene. *Phys. Rev. B*, 87:205404, May 2013.

[31] Xiao Chen, Shuanglong Liu, James N Fry, and Hai-Ping Cheng. First-principles calculation of gate-tunable ferromagnetism in magic-angle twisted bilayer graphene under pressure, 2020.

[32] G. Trambly de Laissardière, D. Mayou, and L. Magaud. Localization of dirac electrons in rotated graphene bilayers. *Nano Letters*, 10(3):804–808, feb 2010.

[33] J. M. B. Lopes dos Santos, N. M. R. Peres, and A. H. Castro Neto. Continuum model of the twisted graphene bilayer. *Physical Review B*, 86(15), oct 2012.

[34] Lede Xian, Salvador Barraza-Lopez, and M. Y. Chou. Effects of electrostatic fields and charge doping on the linear bands in twisted graphene bilayers. *Phys. Rev. B*, 84:075425, Aug 2011.

[35] Mikito Koshino, Noah F. Q. Yuan, Takashi Koretsune, Masayuki Ochi, Kazuhiko Kuroki, and Liang Fu. Maximally localized wannier orbitals and the extended hubbard model for twisted bilayer graphene. *Phys. Rev. X*, 8:031087, Sep 2018.

[36] Hoi Chun Po, Liujun Zou, T. Senthil, and Ashvin Vishwanath. Faithful tight-binding models and fragile topology of magic-angle bilayer graphene. *Physical Review B*, 99(19), may 2019.

[37] Rafael M. Fernandes and Jörn W. F. Venderbos. Nematicity with a twist: Rotational symmetry breaking in a moiré superlattice. *Science Advances*, 6(32):eaba8834, 2020.

[38] Minhao He, Yuhao Li, Jiaqi Cai, Yang Liu, K. Watanabe, T. Taniguchi, Xiaodong Xu, and Matthew Yankowitz. Symmetry breaking in twisted double bilayer graphene. *Nature Physics*, 17(1):26–30, sep 2020.

[39] Cheng Shen, Yanbang Chu, QuanSheng Wu, Na Li, Shuopei Wang, Yanchong Zhao, Jian Tang, Jieying Liu, Jinpeng Tian, Kenji Watanabe, Takashi Taniguchi, Rong Yang, Zi Yang Meng, Dongxia Shi, Oleg V. Yazyev, and Guangyu Zhang. Correlated states in twisted double bilayer graphene. *Nature Physics*, 16(5):520–525, mar 2020.

[40] Rhine Samajdar, Mathias S Scheurer, Simon Turkel, Carmen Rubio-Verdú, Abhay N Pasupathy, Jörn W F Venderbos, and Rafael M Fernandes. Electric-field-tunable electronic nematic order in twisted double-bilayer graphene. *2D Materials*, 8(3):034005, may 2021.

[41] H. Bruus, K. Flensberg, and Ø.R.L.N.B.I.K. Flensberg. *Many-Body Quantum Theory in Condensed Matter Physics: An Introduction*. Oxford Graduate Texts. OUP Oxford, 2004.

[42] Mathias Scheurer. Spectroscopy of graphene with a magic twist. *Nature*, 572:40–41, 08 2019.

[43] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[44] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc., 2019.

[45] Frederic B. Fitch. Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133. *Journal of Symbolic Logic*, 9(2):49–50, 1944.

[46] Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 12 1964.

[47] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[48] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(null):2121–2159, jul 2011.

[49] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[50] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[51] Tensorflow documentation. https://www.tensorflow.org/.

[52] Keras documentation. https://keras.io/.