Report on readAFM

Tobias Reisch

August 25, 2017

1   The Project

The goal of this project was to explore the possibilities of using Machine Learning to analyze data obtained from Non Contact Atomic Force Microscopy (NC-AFM). In particular functionalized tip NC-AFM, which allows to achieve atomic resolution [1].

This report contains a brief description of the most important steps in the project, especially

- How to generate the input data.

- How to generate the labels.

- How the Neural Net was built.

- Highlights of the Results and an Outlook.

About the title: The git-repository was named *readAFM* in intentional similarity to *mechAFM*, while *mechAFM* simulates an AFM image with a mechanical model of the tip atoms, *readAFM* was the attempt to understand the output "images" of an AFM, in a sense "read" them.

The code written in the course of this internship is available under https://github.com/SINGROUP/readAFM/.

2 Input

Machine Learning usually needs a lot of training data to work. In our case this would be measured frequency shifts or forces from AFM experiments. Since there is no preexisting database of labeled experimental AFM 'images' available, we decided to use simulated data. We used the *MechAFM* program, developed in our group, based on [2, 3], to generate three dimensional arrays filled with force values. The inputs for *MechAFM* consist of a xyz-file, a file specifying the potentials to use for the molecular dynamics simulation and a set of parameters. The values used that deviate from the default parameters are:

area	8 Å x 8 Å
zlow	position of topmost atom + 4 Å
zhigh	zlow + 4 Å
dx,dy,dz	0.2 Å

The output of *MechAFM* consists of the force acting on the tip in z-direction, the angle of the CO-molecule and the displacement of the tip atom. We only keep the force in z-direction, which is compiled to a *Numpy* array and stored in a hdf5-container. Additionally we store informations like the name string, the position of the atoms, the array-size, etc. as attributes in the hdf5-file. The data is separated randomly into a training and a validation set, with roughly a 80:20 splitting.

3 Labels

For this large amount of simulated AFM data, we need to generate labels automatically. This step should be done very carefully, because the labels will also determine what the output of the Neural Net will look like. If we assign labels containing more information about the molecule than there is in the AFM data, the NN will not be able to train successfully, on the other hand, we want to get as much out of it as possible. We could, of course, ask the NN to predict the chemical sum formula for the molecule, but for this sort of analysis, there are already many different techniques available. And it is the big strength of atomic resolution AFM, that it is able to determine the spatial structure of the molecules. So the labels are a balance between extracting as much information as possible, determining the spatial structure of the molecule and bringing it into a form that is useful for the NN to learn. For example it is easier to ask for a constant length output, than for a variable length output (i.e. the NN outputs a list of atom positions).

For our case we decided it would be best (at the beginning) to produce a xy-map with values that are large when there is an atom at that position and zero otherwise. It is to a probability map of finding an atom at that position, but due to its lack of normalization it certainly is not. Therefor let's call it *signal*. A projection of the atoms coordinates to the xy-plane, but replacing the atoms with Gaussians scaled according to the covalent radius of the element. The formula for the Gaussians looks like this:

$$amp(element) = amp_0 * \frac{covalentradius(element)}{covalentradius(carbon)} \tag{1}$$

$$\sigma = \sigma_0 * \frac{covalentradius(element)}{covalentradius(carbon)} \tag{2}$$

$$signal(\vec{r}) = amplification * \exp(-\frac{(\vec{r} - \vec{r_0})^2}{\sigma^2}) \tag{3}$$

Where amp_0 and σ_0 are the input parameters and $\vec{r_0}$ is the position of the atom. To take into account, that atoms that are further away from the tip than others are less visible, we interpret the Gaussians as Gaussians in three dimensions and evaluate them at every point in the xy-plane at the height (z-value) of the topmost atom. It turned out to be beneficial to use different values for σ in the xy-plane and in the z-direction. See also fig 1.

It is crucial to choose σ_z carefully, since for a to large value all the atoms would appear in the labels whereas the atoms far away from the tip would not appear in the force images, on the other hand for a too small σ_z only the topmost atom would be included in the labels, while in the force curves also lower lying atoms appear.

To determine $sigma_z$ rigorously we used a concept known from Information Theory, Mutual Information, which has recently been introduced to our group [4]. The idea is that Mutual Information is large when two distributions contain
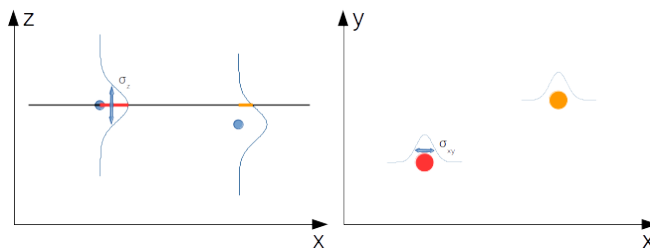
Figure 1: Schematic explanation of the labelling process. The atoms with a large *signal*-value at the height of the topmost atom (left image) appear brighter (darker red) in the projection to the xy-plane (left image).

similar information and small else. We calculated MI vs. σ_z curves 2, and they show a clear maximum.
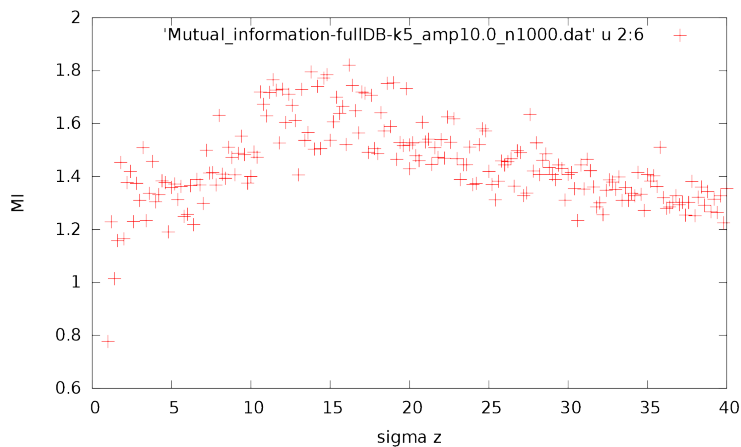

Figure 2: Mutual Information vs. σ_z for three dimensional molecules in random orientations. The spread can be explained by the fact, that the 1000 samples to calculate MI at every point were drawn randomly every time.

4 CNN

In recent years there have been great advances with deep learning and Convolutional Neural Nets [5] and using them for automated analysis of experiments [6]. We try to apply this technique in our case to analyze AFM data. For our task using a CNN seems feasible, for several reasons:

- Its translational invariance allows to use the same filters for different sized images of the same resolution.

- The use of filters of a particular size can account for the flexibility of the CO-molecule at the end of the tip.
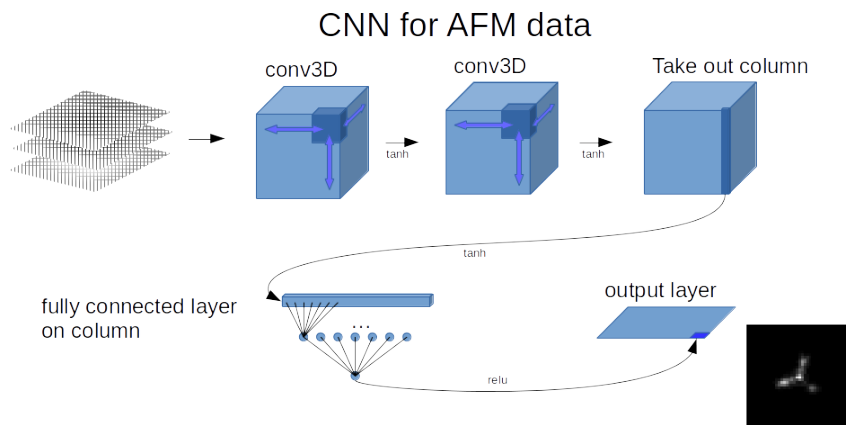

Figure 3: Schematic representation of our NN, see explanation in the running text.

Our dataset can be imagined as a tensor of order 4, three indices for the spatial degrees of freedom (x,y,z) and one index for the "input channels", when treating color images this number is three, for the three color channels RGB. In our case there is just one input channel, the forces. In the first convolutional layer, this tensor is convolved with 16 filters of the size 0.8 x 0.8 x 0.8Å, the convolution preserves the spatial size of the tensor, but the number of "channels" grows to 16. After applying the hyperbolic tangent function as nonlinearity, we repeat the convolution step with 32 filters.

After the 3D convolutions preserved the order of the tensor, now we want to reduce it to a two dimensional image. To achieve this, we flatten the axes of the tensor corresponding to z and the channels and use them as input to a fully connected layer. After the fully connected layer we apply a rectified linear (ReLU) operation as nonlinearity and we receive the output. To stay with the language of tensors: Now we have a tensor of order three, two spatial degrees

of freedom and one output channel. The size of the spatial axes is conserved. For a schematic explanation see fig. 3.

For this Neural Net the filters of the convolution and the weights and biases can be trained with gradient descent. We used stochastic training with the Adam Optimizer [7], which was especially designed for this purpose.

To use gradient descent we need to determine a scalar value as "cost" function which can then be minimized. We start out calculating a value for the "accuracy" which is the summed squared difference between the prediction of the NN and the label.

$$accuracy = \sum_{x,y}(v_{pred}(x,y) - v_{sol}(x,y))^2 \tag{4}$$

Where v(x,y) should represent the greyscale value at the position (x,y). This measure works to train the NN and is a good characteristic for the convergence of a training session. However, for small σ_{xy} and small amp, the gradient descent would train the weights and biases to a state where they would just output zero everywhere. To prevent this, we adapted the accuracy function by multiplying it at each position (pixel) with the solution, see Eqn. (5). Because v_{sol} is 0 everywhere except at the positions of the atoms, this represents an extra penalty for false negatives (showing no atom even when there should be one). The $\frac{1}{amp}$ compensates for the fact that v_{sol} ranges between 0 and amp, not 0 and 1.

$$accuracy_{weighted} = \sum_{x,y} \frac{1}{amp} * v_{sol}(x,y) * (v_{pred}(x,y) - v_{sol}(x,y))^2 \tag{5}$$

If only the weighted accuracy is used, there is no penalty to false positives (showing atoms where none should be) anymore. To prevent this, the cost function is made up by a mix of both accuracies, the proportionality can be determined by a parameter c (costWeight in the *parameters.in* file).

$$cost = c * accuracy + (1 - c) * accuracy_{weighted} \tag{6}$$

The accuracy and trainability of this Neural Net can be changed in a several ways: By changing the filter size and number in the convolutions, using different sized fully connected layers (a FC layer with 64 neurons already works, but using larger FC layers improves the result drastically!) and of course adding more layers. The effect of these changes will be discussed in the next section.

5 Results

Although the goal of the project is clearly set to use ML to understand better AFM images of arbitrary molecules, it was more practical to start with a simplified task. At first we had a database filled with single atoms at random positions, mainly to debug and as a sanity check. Then we followed the "historical" route and produced a database of flat molecules (where the heavy atoms (everything except H) are all in one plane). The first functionalized tip NC-AFM experiments with atomic resolution were also performed on flat molecules, because a corrugation in z complicates the task even more. Which is what we did in a third step. We used 1000 molecules from [8] and simulated each in 100 random orientations.

The analysis for the flatDB worked quite well. Our CNN was able to predict the correct positions of the atoms quite well, or a few examples see figure 4. It became apparent that especially a larger hidden layer improves the performance a lot.

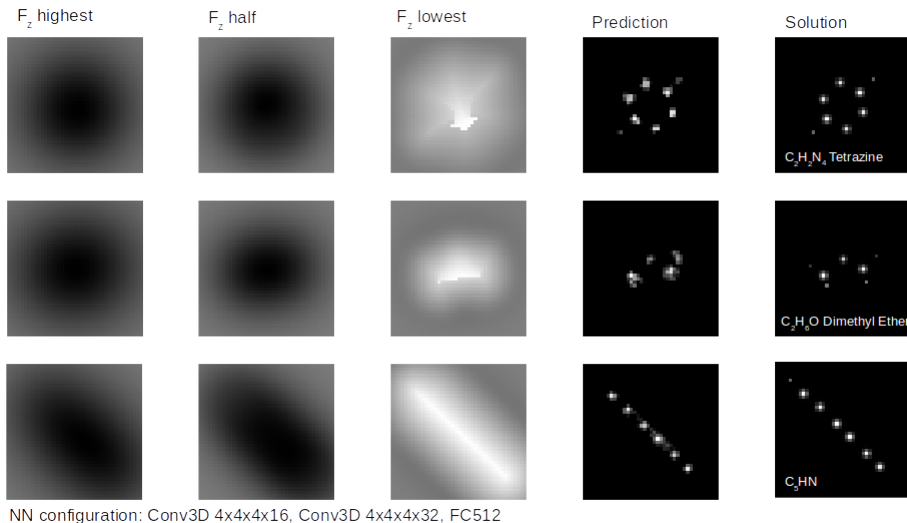NN configuration: Conv3D 4x4x4x16, Conv3D 4x4x4x32, FC512

Figure 4: Flat molecules analyzed with the NN configuration: Conv3D 4x4x4x16, Conv3D 4x4x4x32, FC512

For the database filled with random three dimensional molecules in random orientations there is the extra problem, that not all parts of the molecule are visible. To find the optimal σ_z we tried to maximize the Mutual Information for the F_z-tensor and the solution-"image" as a function of σ_z, this value agreed quite well with the previous estimate (just from looking at it). However, the

resulting predictions are not as good as they are for the flat molecules. In figure 5 the results of the most basic configuration are shown, the positions of the atoms are not predicted as well as for the flat molecules. We were also not able to improve them much with bigger hidden layers.

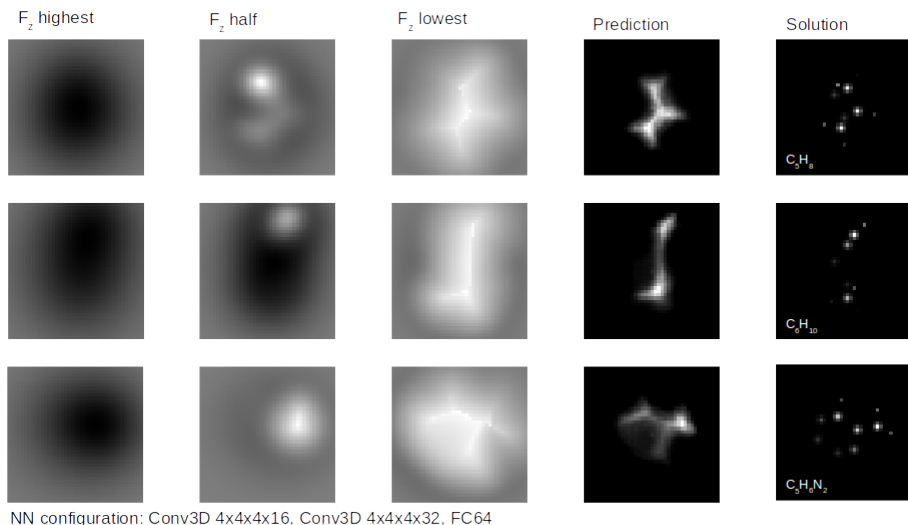NN configuration: Conv3D 4x4x4x16, Conv3D 4x4x4x32, FC64

Figure 5: Random molecules analyzed with the NN configuration: Conv3D 4x4x4x16, Conv3D 4x4x4x32, FC64

Here is a table of configurations that were tried:

conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh hl64 relu
conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh hl128 relu
conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh hl256 relu
conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh hl512 relu
conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh hl1024 relu
conv3d(8x8x8x16) tanh conv3d(8x8x8x32) tanh hl64 relu
conv3d(8x8x8x16) tanh conv3d(8x8x8x32) tanh hl256 relu
conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh hl64 relu conv2d(4x4x16) relu
conv3d(4x4x4x16) tanh conv3d(4x4x4x32) tanh conv2d(4x4x32) tanh hl64 relu

We also tested for which $\sigma_{xy}/$amp combinations successful training was possible and when it would just converge to 0 everywhere, we were able to prove that the improved cost function made the values for $\sigma_x y/$amp much lower. It is also possible to avoid local minimal by starting from a larger σ_{xy} and use that weights and biases as a starting point to train for smaller σ_{xy}.

8

6 Outlook

Possibly adding an extra convolutional layer or more hidden layers could improve the results. An option also worth considering might be to apply extra operations on the output layer. These could either be some filters, e.g. a search for local maxima, or again a Neural Net, e.g. convolutional layer that is trained separately from the CNN.

Maybe the cut through the Gaussians at the height of the topmost atom, is not optimal. From a physical perspective doing the cut at the lowest position of the tip could also make sense.

We already contacted experimentalists, about NC-AFM data which we could use our NN on. Potential insights include whether the training on data simulated with *MechAFM* works, or if a more sophisticated model is necessary. It would also test our model for its robustness against experimental artifacts.

The exchange with experimentalists is also very valuable with regard to the future direction of this project: What form of possible applications for automated (NC-)AFM experiments would be the most welcome?

References

[1] Leo Gross, Fabian Mohn, Nikolaj Moll, Peter Liljeroth, and Gerhard Meyer. The chemical structure of a molecule resolved by atomic force microscopy. *Science*, 325(5944):1110–1114, 2009.

[2] Prokop Hapala, Georgy Kichin, Christian Wagner, F Stefan Tautz, Ruslan Temirov, and Pavel Jelínek. Mechanism of high-resolution stm/afm imaging with functionalized tips. *Physical Review B*, 90(8):085421, 2014.

[3] Sampsa K Hämäläinen, Nadine van der Heijden, Joost van der Lit, Stephan den Hartog, Peter Liljeroth, and Ingmar Swart. Intermolecular contrast in atomic force microscopy images without intermolecular bonds. *Physical review letters*, 113(18):186102, 2014.

[4] Yashasvi Ranavat. https://github.com/SINGROUP/MutualInformation.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Oren Z Kraus, Ben T Grys, Jimmy Ba, Yolanda Chong, Brendan J Frey, Charles Boone, and Brenda J Andrews. Automated analysis of high-content microscopy data with deep learning. *Molecular systems biology*, 13(4):924, 2017.

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

[8] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022, 2014.