5 Training with Electronic Snapshots

For this exercise, you will be working with theoretical electronic snapshots for a quantum model of interacting electrons on a square lattice. These snapshots are obtained during a Monte Carlo simulation of the model. In this model, electrons live on vertices (sites) of a 10×10 lattice. Being spin-1/2 fermions, electrons show up in four different ways at any given site when their image is taken (a quantum measurement is made). They either do not occupy that site, one of the species occupies the site (either a spin-up or a spin-down), or both species occupy the site (double occupancy). Of course, no two fermions of the same spin species can occupy the same site; be in the same exact quantum state.

The system we are going to work with is a spin-balanced system; we have an equal number of spin-up and spin-down electrons in every snapshots. Moreover, you are given snapshots of only spin-up electrons after the measurement. The reason will be clear later. In every image, each of the 100 pixels can take two integer values: 0 or 1, indicating whether there is a spin-up electron at that site. On average, there are 45 spin-up electrons in each snapshot. You are given two sets of data (`Theory_TrainingSet_High.dat` and `Theory_TrainingSet_Low.dat`), one containing images taken at a relatively low temperature, and one containing images taken at a relatively high temperature.

The goal of this exercise is to use these theory data to train a convolutional neural network to recognize whether a snapshot belongs to the low temperature or the high temperature data set. Ideally, the network will be able to do this by learning the physics of what is happening in the system at these extreme limits (whether the electrons organize themselves in particular ways) and then we can try to infer that physics!

You are also given four sets of data obtained from an experiment that uses lasers and fermionic atoms (playing the role of electrons) to emulate our theory model (`Expt_TestSet_#.dat` see the first reference below). These are 12×12 images with pixels that take three different values: 0, indicating outside the range of the microscope, 1, indicating an atom, and -1, indicating no atom or a double occupancy. One of the challenges of this exercise is to be able to use your trained neural network and determine which of these four experimental data sets contains high temperature snapshots and which one contains low temperature snapshots.

Begin with the code for multiplying two numbers and load the training data sets. Each set has 10,000 snapshots organized in rows. There are 104 columns. You should ignore the numbers in the first 4 columns and reshape the remaining 100 in each row into 10×10 images. Visualize a few of the snapshots at each temperature using `imshow`. Then, create the matrix of data X with a shape $(20000, 10, 10, 1)$ and place all the images from the two temperatures in there. Also create the matrix of labels Y with a shape $(20000, 2)$. Separate your data into 80% for training and 20% for validation during the training. Organize them into X_train, X_test, Y_train, T_test as before.

In the next step, construct your neural network and use a convolutional layer (`Conv2D`) with a couple of kernels of size 3×3 or 5×5 as your first layer. You can use an "input_size" of $(10, 10, 1)$ and rectified linear unit (relu) as your activation functions. Next, flatten the feature maps using `model.add(Flatten())` and add a fully-connected layer before the output layer with two perceptrons. Your activation function for the output layer should be "softmax".

Perform the training and monitor the accuracies and losses. Do you see any signs of overfitting? You should be able to get around 80% in your testing accuracy.

One way to figure out what the machine has learned is to examine the kernels. Find a way to visualize your trained kernels and interpret what you find.

Now load and prepare your experimental snapshots to be run through the neural network. The objective is to find out what temperature region each set belongs to (low or high). You need to do that by obtaining the network output for each data set and examining the average. Note that you have to adjust the sizes of these snapshots to be able to run them through your machine, and you also have to convert pixel values to have the same convention as in your training data.

More reading:
"String patterns in the doped Hubbard model", Chiu at al., Science 365, 251 (2019)
URL: `https://www.science.org/doi/10.1126/science.aav3587`
The experimental data come from this paper. They can be downloaded from `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/1CSVBV`

"Machine learning phases of strongly correlated fermions" K. Ch'ng, et al., Phys. Rev. X 7, 031038 (2017)
URL: `https://journals.aps.org/prx/abstract/10.1103/PhysRevX.7.031038`

"Visualizing strange metallic correlations in the two-dimensional Fermi-Hubbard model with artificial intelligence" Khatami et al., Phys. Rev. A 102, 033326 (2020)
URL: `https://journals.aps.org/pra/abstract/10.1103/PhysRevA.102.033326`