Study of quantum fluctuations in a degenerate Fermi gas using machine-learning techniques

Erika Eill
North Carolina State University
eleill@ncsu.edu

Ilya Arakelyan
North Carolina State University
iarakel@ncsu.edu

Bradford Ingersoll
North Carolina State University
bingers@ncsu.edu

Abstract

Recently, machine-learning techniques such as artificial neural networks have proven successful in quantum physics applications [1,2]. Here, we apply such techniques to experimental images of ultracold atomic gases near degeneracy to study transition from classical to quantum physics and calibrate temperature of an atomic sample. Deep learning methods open new possibilities for model-independent identification of quantum regime using individual images.

1 Introduction

In a classical atomic gas, local fluctuations of density n obey Poisson distribution. With an onset of degeneracy in a Fermi system, the gas is no longer classical and the underlying statistics of density fluctuations becomes sub-Poissonian, such that $(\Delta n)^2 < \bar{n}$. A study of density fluctuations in atomic clouds paves a way to detect the emergence of quantum regime and provides robust measurements of thermodynamic properties of the gas. Determination of temperature relies heavily on precise knowledge of experimental parameters and understanding of complex internal interactions within the system. Due to a mediocre signal-to-noise ratio, the same preparation protocol is repeated multiple times. Conventionally, statistical analysis of multiple images is employed to reveal the size of fluctuations [3]. Possible classification based on individual images would provide an entirely new approach to fast and model-independent detection of degenerate regime and lead to new applications of machine-learning techniques in quantum physics of ultracold atomic gases for applications such as a phase transition to the superfluid state.

In this work we successfully realize a novel application of neural network algorithms to the atom number fluctuation problem. We investigate density distributions of laser-trapped ^{6}Li sample by taking 2D light absorption images of atomic clouds near a transition to quantum regime. We implement two distinct sample preparation protocols resulted in two temperature classes, "hot" and "cold". For training purposes, the ground truth labels can be identified from the sample preparation procedure and validated by models describing density fluctuations as a function of temperature using existing physics models [3]. For classification, we employ convolutional and standard neural networks to label individual images.

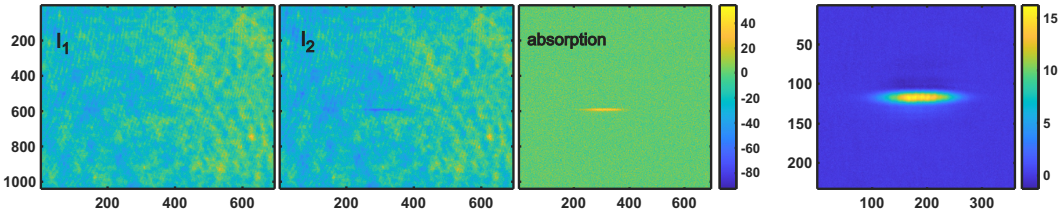

Figure 1: Intensity of the imaging beam before the cloud (I_1), after the cloud (I_2), light absorption $(1 - I_2/I_1) \times 100\%$. Rightmost panel shows resulting density distribution (atom number in each pixel) for an average of 100 images.

2 Methods

2.1 Data Set and Preprocessing

A raw data set of 4300, 12-bit 1040x696 images of atomic density distributions was collected within the "Thermodynamics of weakly interacting Fermi gases" study at AMO laboratory led by Dr. John Thomas at NCSU [4]. The atom clouds are produced using two different experimental protocols, which provide samples at two different temperatures. Ordinarily, the temperature is determined from an atom density distribution using mathematical models that are valid in a few special cases. Here, we discuss another model-dependent technique based on fluctuations in atom density that has a broader range of potential applications to determine the ground truth for each sample preparation protocol.

Atom density can be determined by measuring absorption of light passing through a cloud. If $I_1(x, y)$ is the transverse intensity distribution of light propagating along z-axis through an atomic cloud of density $n(x, y, z)$, a fraction of light is absorbed by the cloud resulting in the intensity distribution $I_2(x, y)$. In a linear-absorption regime,

$$I_2(x, y) = I_1(x, y) \times e^{-OD(x,y)}, \tag{1}$$

where optical density $OD = -ln(I_2/I_1)$ is proportional to an integrated atom density $n_2(x, y) = \int n(x, y, z)dz$. Thus, taking an image intensity distributions $I_{1,2}$ before and after atoms determines n_{2D}. Since simultaneous measurement of I_1 and I_2 is not possible, two pictures are taken, one with the atoms present (I_2) and one without (I_1). Typical images of intensities I_1, I_2, absorption $(I_1 - I_2)/I_1$ and the resulting density distribution $n_{2D}(x, y)$ are shown in Fig. 1. The pixel size is 1.31 μm (vertical) by 2.62 μm (horizontal).

2.2 Ground Truth Determination

Density fluctuations are imprinted into the absorption image in Fig. 1, but their extraction using conventional statistical analysis relies on precise knowledge of experimental parameters. For any given small region (superpixel) of an image, the corresponding count number would vary when the same experiment is repeated. The variation is due to the Poissonian statistics of light used to image atoms, technical noise of the video camera and, finally, fluctuations in the local atom number. Using Eq. 1, the variance of the optical density in a given superpixel is given by

$$(\Delta OD)^2 = (\Delta I_1)^2 + (\Delta I_2)^2 + (CameraNoise)^2 + (\Delta N_{atom})^2, \tag{2}$$

assuming all contributing effects are uncorrelated.

Taking multiple images of I_1 and I_2 under similar experimental conditions, we obtain the mean value of the density (atom number per superpixel) using Eq. 1 and its variance using Eq. 2. A typical images of the mean density and density variance of 100 images are shown in Fig. 2. The superpixel size is is 2.62 μm (vertical) by 10.5 μm (horizontal). The colorbar shows atom number per superpixel.

For every superpixel in Fig. 2 we can verify if the atom number variance is sub-Poissonian, *i.e.*, $(\Delta N_{atom})^2 < N_{atom}$, by plotting the two quantities as shown in Fig. 3 for two different preparation protocols. Each data point in Fig. 3 corresponds to a superpixel in Fig. 2 (right panel). The two trends show two distinct temperature regimes, with the hotter one (red) closer to the classical thermodynamic limit $(\Delta N_{atom})^2 = N_{atom}$ (dashed line).

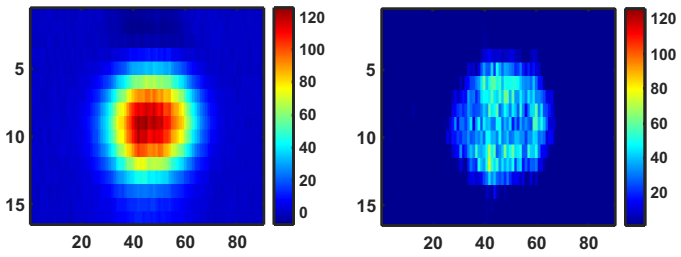

Figure 2: Mean density distribution (atom number per superpixel) of 100 images (left panel) and the corresponding variance (right panel).

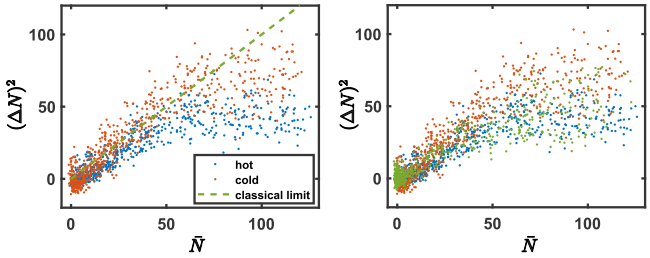

Figure 3: Left panel: Training sets, "cold" (blue), "hot" (red) and the classical limit $(\Delta N_{atom})^2 = \bar{N}_{atom}$ (dashed line). Right panel shows a test set (green) along with the two training sets.

We use the two trends of Fig. 3, "hot" and "cold", as training data sets for a new collected data to be classified. For that, we pick 248 "hot" and 271 "cold" images that have a low standard deviation from the mean (<5%) in the total atom number, cloud size, and laser power. That ensures that the observed atom number variance is of statistical nature and not due to variation in total atom number between different samples. For each set, we compute the mean atom number per superpixel and the corresponding variance to generate a plot shown in Fig. 3. The test set consists of 98 "cold" images taken on a separate date, with the mean atom number slightly different from that of the training data. We do not mix the two data sets to apply, for instance, a cross-validation method since normalizing data sets is dangerous: bigger and smaller clouds might have exactly the same temperature and result in the same trend as in Fig. 3. Re-scaling one of the data sets would lead to calculating a wrong value of temperature. The test set (green) is shown along the training sets (blue and red) in Fig. 3, right panel.

We consider three methods to classify a test set. Using regression analysis we fit each training trend to a polynomial and classify the test data points based on their proximity to the polynomial. Also, we partition the two training trends by a polynomial to minimized the resulting classification error and classify the test data points based on their location with respect to the boundary. Finally, we classify the test set using the nearest-neighbor method. Note, that the three approaches include bagging: each superpixel is a weak classifier, all superpixels combined provide the final classification.

3 Hot and Cold Data Sets Separability

To apply neural network protocols for classification we must first verify that the two data sets are separable and can be distinguished.

3.1 Regression analysis

First, we fit each trend in the training set using polynomials of various power p and use the adjusted R-square statistic as an indicator of the fit quality when higher powers (more fit parameters) are added to the polynomial. Results for "cold" data set are shown in Table 1.

P	2	3	4	5	6
adjusted R-square	0.84	0.84	0.85	0.85	0.86

Table 1: Adjusted R-square for polynomial fits of various powers.

We find that using a higher power polynomial does not improve the fit quality significantly. A simpler model, the second power polynomial in the form $p_0 + p_1 x + p_2 x^2$, is used below and in the Partitioning method subsection that follows. Moreover, a physical model of the system suggests the parabolic dependence. The corresponding polynomial fits are shown in Fig. 4 (left panel). We also plot the test data set along with "hot" and "cold" polynomial fits (Fig. 4, right panel). We use the sum of squared residuals (SSR) between the test data set and each fit as a relative indicator for the purpose of classification. We find (SSR_{cold})=162 and (SSR_{hot})=475. The SSR-value is normalized by the number of the corresponding data points; for the test data set we only consider points with $\bar{N}_{atom} > 20$ to avoid contribution from data with a low signal-to-noise ratio. For comparison, the best polynomial fit to the cold training set results in a normalized SSR of 53.

Based on the SSR-values we classify the test set as "cold" that can also be seen in Fig. 4.

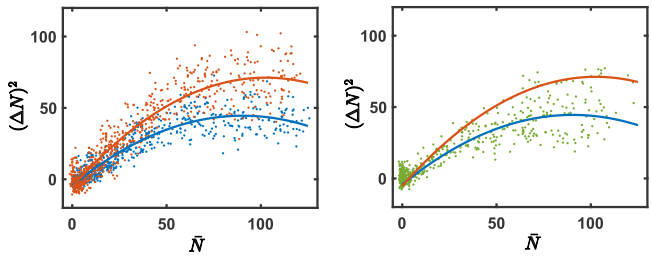

Figure 4: Left panel: Polynomial fits (solid line) to "hot"(red) and "cold"(blue) data. Right panel: Location of test data set (green) with respect to polynomial fits.

3.2 Partitioning Method

We use partitioning to establish a polynomial boundary between the "cold" and "hot" training data sets. We introduce an error function that for a given polynomial combines resulting classification errors for each superpixel for both sets. We then minimize the error function with respect to the polynomial coefficients p_0, p_1, and p_2. The error for each data point is weighted by its distance from the polynomial, we only include points with $\bar{N}_{atom} > 20$. The resulting partitioning in shown in Fig. 5 (left panel).

We then apply the partition to classify the test set as shown in Fig. 5 (right panel). Each superpixel of the test set is a weak classifier, which is identified based on its position with respect to the polynomial. The combined unweighted classification error for class "cold" is 27% (172 out of 237 correctly identified), for class "hot", 73% (65 out of 237 correctly identified). Based on the boundary polynomial we classify the test set as "cold" that can also be seen in Fig. 5.

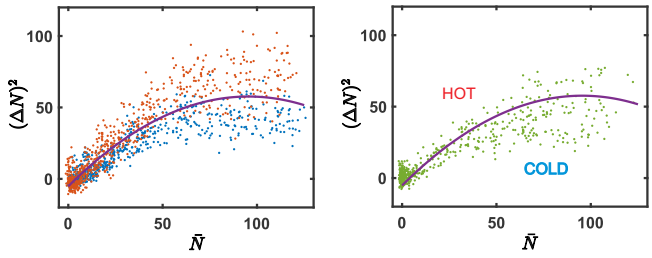

Figure 5: Partition polynomial (solid) divides "hot" and "cold" data sets (left panel). Right panel: application of partition polynomial to classification of test data (green).

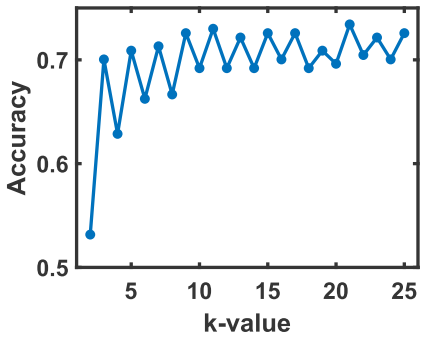

Figure 6: Accuracy of classification prediction for the nearest-neighbor method for various values of k.

3.3 Nearest-neighbor Method

We apply the nearest-neighbor method for $k = 2$ to $k = 25$ to classify the test data set. For every superpixel of the test set with $\bar{N}_{atom} > 20$, we find nearest neighbors from a combined "cold" and "hot" sets using Euclid metric and identify the corresponding majority class. We then combine weak classifiers together to obtain the classification accuracy. The calculated accuracy is plotted versus value of k in Fig. 6. We find that the test set can be identified as "cold" with the about 71% accuracy at large values of k. Unfortunately, we do not have enough test data to conduct cross-validation type techniques to verify if this result is robust or corresponds only a specific choice of training and test sets. We point out that the obtained accuracy matches well with the classification accuracy (1-error) obtained using Partitioning method.

4 Application of Neural Networks

Once we verified that the two datasets, hot and cold, are separable we proceeded with classification algorithms using Artificial Neural Networks. Upon completion of the data pre-processing described in Methods, each dataset consisted of 145 black-and-white, 12-bit images with a size of 16$\times$ 90 pixels. Each pixel's value represents a corresponding number of atoms. We applied both Convolutional Neural Network (CNN) and Standard Neural Network (SNN) methods to classify individual pictures from test sets.

4.1 Convolutional Neural Network

Application of neural network algorithms is a standard approach in image recognition problems. In this work, we used a single-convolution layer, fully connected network, a simplified version of the 14-layer network employed in a 3-class problem of reference [7]. Our network comprises 7 layers as shown Fig. 7 and was numerically implemented [8] using MATLAB Deep Learning Toolbox. The input layer is followed by a 2-dimensional convolutional layer with about 16 filters (maps) of a typical size of 1$\times$4 pixels. The convolution is activated by a linear rectifier, followed by the maximum value pooling of variable size and stride. The next layer of fully connected neurons is further classified using normalized exponential (softmax) loss function.

The network implementation and optimizations were performed as follows. We divided images of both classes between a training set of 200 and a test set of 90 using random sampling. We trained the network for a given set of hyper parameters: size and number of filters, size and stride of pooling. We then processed test images through the network and computed accuracy, which we used as the metric of the network classification performance. To reduce sampling bias, we repeated the algorithm 25 times for random train/test partitions and averaged the resulting accuracy.

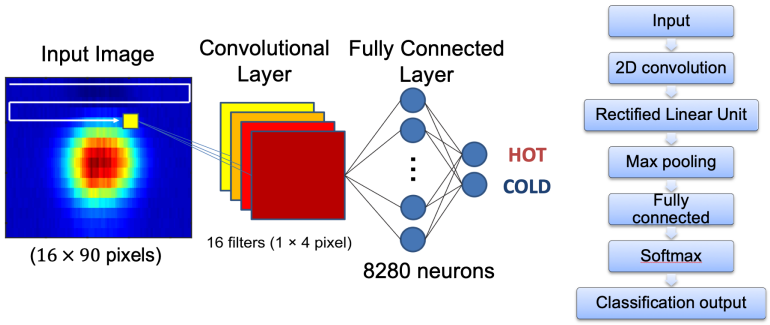

Figure 7: CNN model with 7 layers. An input image is classified using a neural network shown on the right.

4.2 Standard Neural Network

In addition to the CNN (our primary experimental focus), we decided to further experiment with neural networks by also creating an experiment utilizing a standard neural network [7]. Rather than working with the 2-D images and a convolutional layer for image classification, we flattened the data and fed it into a more traditional neural network model. Using the Keras library within Python we created the following model:

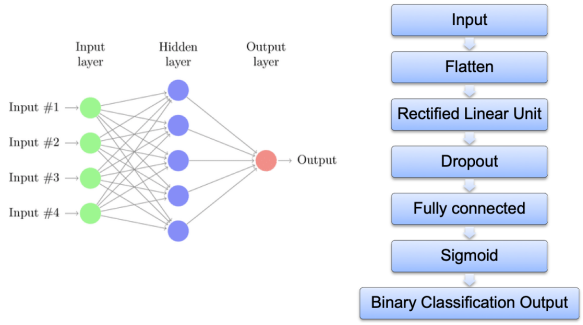

Figure 8: Standard NN model layers.

The hyper parameters that we optimized upon included dropout rate, number of hidden neurons, batch size, number of epochs, and the adam optimizer learning rate. In order to perform an exhaustive optimization of the hyper parameters we used GridSearchCV to test every possible combination with a cross validation using 10 k-folds. The testing we performed found the optimal values to be: Dropout Rate = 0.1, Hidden Neurons = 20, Batch Size = 96, Epochs = 100, and Adam Learning Rate = 0.001.

5 Discussion

5.1 Convolutional Neural Network

We optimized the hyperparameters of the network by manually varying the number of filters at the convolution layer and changing filter, pooling, and stride sizes. During optimization, we maintained the same aspect ratio for all three sizes. We found the maximum accuracy of 83% for the following set of parameters: 20 filters of 1 by 4 pixels pooled in size 1×4 with a stride step of 1×4 that corresponds to the total of 8280 neurons. The resulted accuracy of the converging model and its loss rate is shown in Fig. **??**. A typical training sequence converges after about 100 iterations in less than 10 seconds when implemented on a single Nvidia Quadro P2000 GPU of a mid-range desktop PC. We found that increasing the number of neurons or adding another convolutional layer does not improve the validation accuracy but increases the computation time significantly beyond the scope of this project.

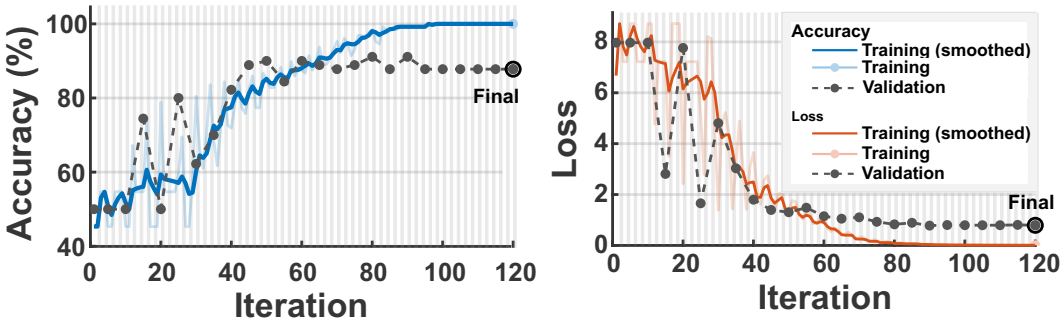

Figure 9: Accuracy and Loss of a converging CNN model. A typical performance (accuracy) of the network is about 80%.

5.2 Standard Neural Network

After optimization of the hyperparameters, our final standard neural network model produced a training accuracy of 96%, a validation accuracy of 85%, and a test accuracy of roughly 91% over 100 epochs. Upon plotting accuracy values over 100 epochs, we noticed that there was a constant oscillation of the accuracy over its overall trend (see Fig. 10). We looked into this and discovered that neural networks exhibit this behavior when the input dataset is small. If we are to continue this experiment and refine the model further, we would seek to have a larger input dataset to train upon.

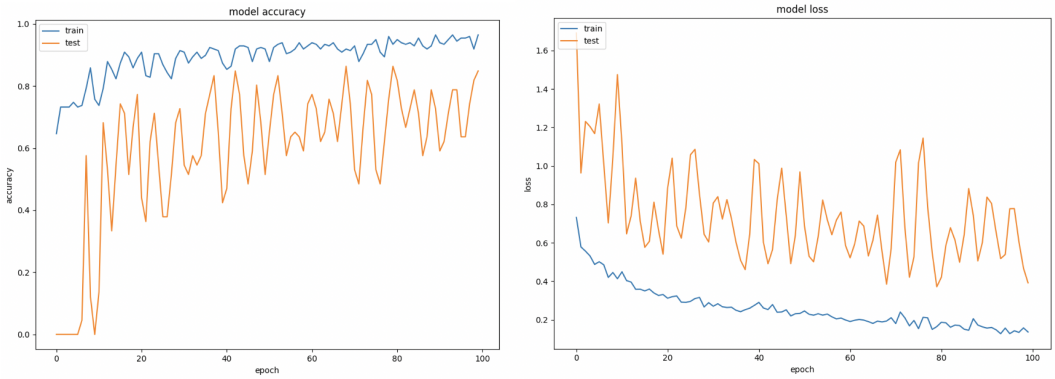

Figure 10: Left panel: Accuracy of SNN model over 100 epochs. Right panel: Loss of SNN model over 100 epochs.

6 Conclusions

In summary, we introduced an atom number fluctuation problem and described how ground truth classification is traditionally done through statistical analysis of a batch of images. Such analysis requires precise knowledge of experimental conditions and large data sets. We introduced and implemented an entirely new approach of classifying individual images using neural networks and obtained test data classification accuracy of 83 to 91%. Such high fidelity allows us to conclude the temperature regime of the system in a single experimental run. As outcomes of neural networking are inherently hard to interpret, one might need to be cautious about such high values of accuracy not typical to conventional statistical analysis. Further in-depth study is required to confirm our findings and determine if the classification is based on the variance feature and not on another parameter such as the atomic sample size nor specific to the employed data set. Such investigation would require more experimental data and significant computational resources.

Finally, we note that the applied techniques have potential applications in mixing computer-generated data with actual experimental data for training purposes and can open new possibilities to study

phase transitions in various quantum systems where the change of a system class can be governed by multiple factors including those unknown or hard-to-identify.

References

[1] Carrasquilla, J., & Melko, R.G. (2017) Machine learning phases of matter. *Nature Physics* **13**:431-434.

[2] Dubessy, R., De Rossi, C., Badr. T., Longchambon, L., & Perrin, H. (2014) Imaging the collective excitations of an ultracold gas using statistical correlations. *New Journal of Physics* **16**(12):122001.

[3] Sanner, C., Su, E.L., Keshet. A., Gommers, R., Shin, Y., Huang, W., & Ketterle, W. (2010) Suppression of Density Fluctuations in a Quantum Degenerate Fermi Gas. *Physical Review Letters* **105**:040402.

[4] Please refer to https://www.physics.ncsu.edu/jet/

[5] Please refer to https://keras.io/

[6] Krizhevsky, A., Sutskever, I., & Hinton, G.E., (2012) ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25*:1097.

[7] Rem, B.S., Käming, N., Tarnowski, M., Asteria, L., Fläschner, L., Becker, C., Sengstock, K., & Weitenberg, C. (2019) Identifying quantum phase transitions using artificial neural networks on experimental data. *Nature Physics* **15**:917-920.

[8] Numerical implementation of Neural Networks of Section 4 and data pre-processing of Section 2 can be found at https://github.com/totallybradical/csc522_project