U-Net Performance Optimization for Quantum Phase Reconstruction

1st Cameron Rodriguez
Civil Engineering Department
Columbia University
New York, United States of America
cjr2210@columbia.edu

2nd Jackson Lee
Physics Department
Columbia University
New York, United States of America
jcl2259@columbia.edu

3rd Fred Garcia
Astronomy Department
Columbia University
New York, United States of America
fbg2107@columbia.edu

***Abstract*—Experimental realizations of Bose-Einstein condensates (BECs) offer to unlock a host of useful and interesting phenomena at the cutting edge of quantum technology. However, experimental probes of BECs are fundamentally incomplete, with experiments measuring snapshots of the density of the condensate, while being unable to measure the corresponding phase - a quantity that is necessary to fully describe the BEC. Here, we present a machine-learning model that can accurately predict the phase field of a BEC, given only the density field as the input. We show that by applying performance optimizations to our model, including pruning and quantization, we can produce a lightweight model that retains a high degree of accuracy. We anticipate that the small memory requirements and robust performance of our model, when combined with suitable hardware, will allow for the rapid characterization of BECs in real-world laboratories[1].**

***Index Terms*—Machine Learning Performance Optimization, Bose-Einstein Condensation, Neural Network Pruning, Quantization**

I. Introduction

Bose-Einstein condensates (BECs) are particularly striking examples of collective quantum phenomena, with a macroscopic fraction of particles in a system occupying the lowest energy quantum state [1]. BECs offer a rich playground for experimental physicists, with studies observing the formation of vortex lattices and vortex-antivortex pairs [2], [3]. Here at Columbia, the first BEC made of dipolar molecules was recently realized [4]; this allowed for unprecedented control of inter-molecular interactions, which in turn paves the way for realization of new quantum phases and powerful quantum simulators [5].

A BEC can be described by a complex-valued condensate wavefunction, $\Psi(x, y)$ (here we consider a BEC in two spatial dimensions, which is experimentally realizable by strongly trapping the system along the third dimension). As with any complex number, $\Psi(x, y)$ can be decomposed into a magnitude, $n(x, y) = |\Psi(x, y)|^2$, and a phase, $\theta(x, y) = arg(\Psi(x, y))$. That is [1]:

$$\Psi(x, y) = \sqrt{n(x, y)}e^{i\theta(x,y)} \tag{1}$$

Quantum mechanical laws dictate that $n(x, y)$ gives the density of the condensate at (x, y), while the gradient of the phase, $\nabla\theta(x, y)$, is proportional to the velocity of the condensate.

[1]see code here: github.com/JackieLee23/HPML-Final-Project

II. Literature Review

Cutting-edge experimental techniques have allowed for the in-situ measurement of the density field of a BEC to become possible [6]. However, measurement of the phase gradient remains experimentally inaccessible. This leads to a challenging problem for experimentalists, who are effectively only able to access partial information about the state of a BEC. To this end, in this study, we present a Machine Learning (ML) model capable of inferring the gradient of the phase by only looking at density snapshots. We note that while previous studies have used the machine learning of density images for vortex detection [7], [8], they have not reconstructed a full phase gradient. The data has been generated computationally in our study, allowing for ground truth density and phase gradient pairs to be used for supervised learning. We are also motivated by the concerns of experimentalists in real labs, where inference must be rapid, often using lightweight models run on edge devices. Thus, we test a series of optimizations to make our model fast and memory efficient, while still being accurate.

III. Methodology

A. Dataset

We consider the case where we confine the BEC in a 2d harmonic trap with trapping frequency $\omega_\perp$, and rotate the trap at frequency Ω. We computationally generate the data by using the Gross-Pitaevskii Equation [7]:

$$i\hbar\frac{\partial\Psi}{\partial t} = \Bigg[-\frac{\hbar^2}{2m}\nabla^2\Psi + \frac{1}{2}m\omega_\perp^2 r^2 + g|\Psi|^2 + i\hbar\Omega\Big(x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}\Big)\Bigg]\Psi \tag{2}$$

Where $\hbar$ is the reduced Planck's constant, and m is the mass of the particles making up the BEC. Henceforth (and in the simulation), we use units where $\hbar = \omega_\perp = m = 1$. We sample from cases with $g \in [50, 500]$, and $\Omega \in [0.45, 0.85]$, and perform imaginary time evolution with Fourier split stepping to find the condensate wavefunction [9]. Each parameter combination and initial state gives a different condensate wavefunction, which can then be split into density

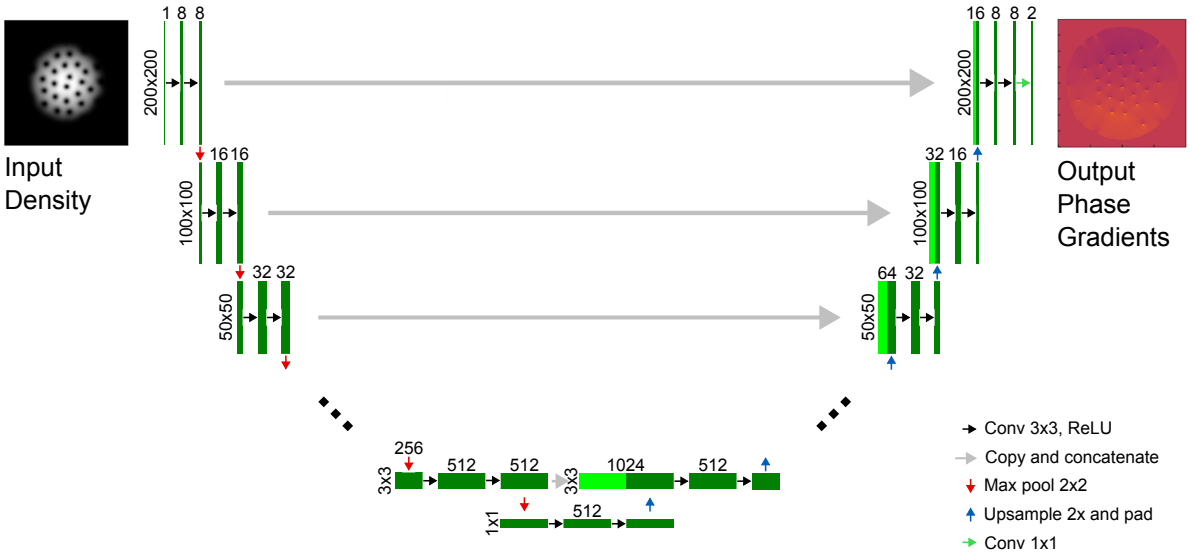

Fig. 1. Diagram of the U-Net. The network takes as input a 200x200 image of the density, and follows an encoding path to a 1x1 image bottleneck with high channel dimension. After the bottleneck, a decoding path subsequently upsamples the inputs, until we arrive at a 2x200x200 output. Skip connections ensure that low-level, local features are not lost.

and phase gradient components. The density is taken to be a 200x200 image, while the phase gradient is a 2x200x200 (an image with channel depth 2), as we have both x and y phase gradients. We generate 1600 wavefunctions, and use a 60/20/20 training/validation/testing split. We note that while the dataset size is small, the machine learning model we use was specifically designed with small training sets in mind, leading to surprisingly robust generalization ability.

B. Machine Learning Model: U-net

We use the U-Net, an image-to-image translation network that was originally designed with the goal of image segmentation while training on a small training set [10]. The U-Net consists of an encoder-decoder architecture, with each encoding and decoding layer consisting of convolutions, and either max-pooling or upsampling. This is then augmented by skip connections, which allow for the transmission of low-level, local information to the final output. An image of the U-Net architecture, with the relevant sizes and channel dimensions of the data, is shown in Figure 3. We utilize a 1x1 image bottleneck, as it is important to capture global information in order to accurately reconstruct phase gradients.

C. Model Performance Benchmarks

In order to train the model, and to benchmark the performance of the model, we utilize a masked mean squared error. As seen in Figure 2, outside of a central region, the density of the condensate drops to close to 0. As the physics outside of this central region is irrelevant experimentally, we mask out regions where the density drops below a very low threshold value. Thus, we choose a loss function that is the pixelwise mean squared error between the ground truth and predicted

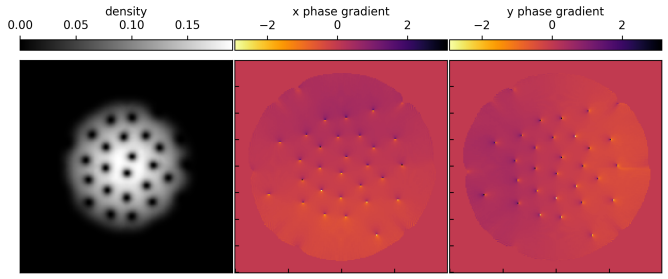

Fig. 2. An example taken from the training set. We show the final density states (left panel, in arbitrary units) of the Bose-Einstein Condensate – which is what is measurable in experiments – and the corresponding x and y phase gradients (here calculated as finite differences between pixels) in the center and right, respectively.

phase gradients, only considering pixels where the density has not been masked out. This loss function is used both to train the model, and also benchmark the performance of the model.

D. Optimization Strategies

We employ a series of optimization strategies to improve the memory utilization and inference latency of the model - vital for experimentalists who need to rapidly analyze data.

1) Optimizer Selection: Optimizer selection plays a crucial role in the training process, and can often allow for better accuracy to be achieved in a much shorter period of training time. Here, we test three different optimizers: standard stochastic gradient descent, Adam [11], and AdamW [12]. While stochastic gradient descent directly updates weights, with a step proportional to the estimated gradient, Adam and

AdamW take advantage of momentum from the gradient and the second-order gradient to take better steps.

2) Neural Network Pruning: Neural networks are often over-parameterized, resulting in redundant weights that unnecessarily consume computational resources [13]. A common approach to solving this issue is neural network pruning, which consists of removing the "least important" parameters of a network in hopes of reducing its size and computational cost. The "importance" of a parameter is defined differently depending on the pruning technique used.

In this work, we explore unstructured (fine-grained) pruning, in which individual weights deemed as "least important" are removed, rather than removing entire neurons, filters, or channels. In addition, weights are pruned globally (across all layers of the network) according to their magnitude, rather than on a layer-by-layer basis. While unstructured pruning can achieve substantial model compression, it generally results in irregular sparsity patterns that are difficult to exploit for computational acceleration. Structured pruning, however, produces more hardware-friendly sparsity patterns and can lead to more efficient implementations. The implementation of structured and/or local pruning strategies into the U-Net architecture described in Section III could be explored going forward.

The pruning techniques considered in this study are listed as follows:

- **Pruning**: Weights are pruned once to achieve the target sparsity level, with no subsequent retraining.
- **Pruning + Fine-Tuning**: Weights are pruned to the target sparsity level, after which the model is retrained using the remaining weights.
- **Iterative Pruning + Fine-Tuning**: Weights are pruned and the model is retrained in multiple iterations until the target sparsity level is reached.

The overall goal of this section is therefore to implement each of the pruning methods described above, and to benchmark their accuracy against the original dense model as the pruning ratio is increased.

3) Post-training Quantization: Quantization is particularly important in model deployment. In this section, we study various quantization strategies to quantize the model post training, to make its deployment and inference less resource intensive. This section employs several methods, balancing accuracy and size compression of the model. Note, the quantization benefits we explore in this model is namely for model compression and not yet bench-marked for performance.

We take the following approaches to quantization and compare them:

- **Mixed Precision Quantization**: Keeps critical layers in float32, quantized less critical layers to int8
- **Per-Channel Quantization**: Uses one scale factor per output channel
- **Per-Tensor Quantization**: The most aggressive of them all, uses one scale factor for the entire weight tensor

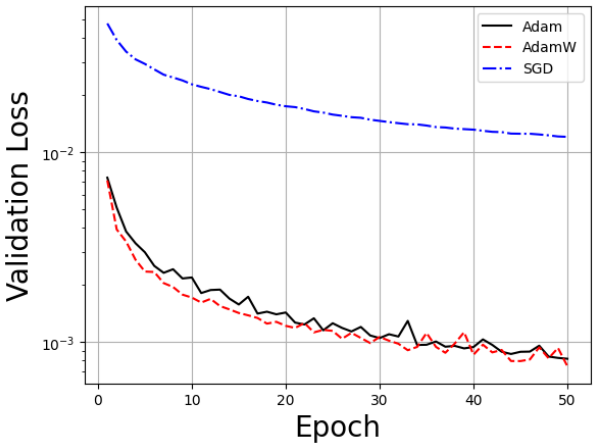

Fig. 3. Validation loss vs. epoch during training for three different optimizers. Results were calculated by averaging three separate runs for each optimizer.

We save these models to disk after quantization, read them back and re-scale them to int32, perform inference, and benchmark the accuracy.

IV. Results and Discussion

A. Optimizer Selection

We show the results of optimizer selection on the validation loss during the course of training in Figure 3, averaged over 3 independent training runs. We see that while Adam and AdamW have similar performance in terms of validation loss, SGD converges much more slowly, with a much higher loss after the same number of training epochs. We hypothesize that the underperformance of SGD is due to some parameters having much higher gradients than others (evidence of this is also seen in the pruning section, where many parameters can be pruned without affecting the accuracy), and without an adaptive step size and momentum, SGD fails to converge quickly. For the rest of the paper, we thus proceed with the Adam optimizer.

B. Neural Network Pruning

In this section, we present the results of the parameter pruning study conducted on the U-Net architecture. Here, we implemented the three pruning techniques mentioned in Section I. Individual weights are pruned globally according to their magnitude. If the model is retrained, 1/5th of the original epoch count and 1/10th the original learning rate are employed. This choice is intended to promote the refinement of the remaining weights, rather than a complete restructuring of the network.

Figure 4 illustrates the weight distributions at the three stages of the pruning process. The changes observed after each step closely match reference results presented in the lecture notes, as well as findings reported in the literature [13]. In particular, the pruning step clearly removes weights with the smallest magnitudes, while the subsequent retraining step produces a modified distribution reflecting the fine-tuning

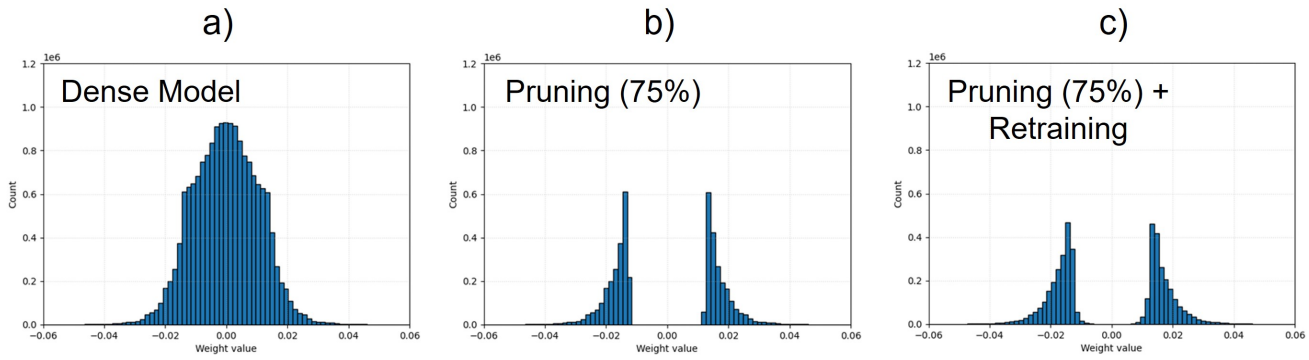

Fig. 4. Weight distributions at the three stages of the pruning process: (a) the original dense model, (b) the model after pruning 75% of the original weights, and (c) the pruned model after retraining.

of the remaining weights. For the iterative pruning and fine-tuning approach, the process shown in 4 is repeated until the target sparsity is reached.

The salient feature of neural network pruning is its ability to reduce model size and complexity while largely preserving accuracy. Therefore, we are interested in examining the accuracy degradation associated with each of the three pruning techniques as the pruning ratio increases. Accuracy is measured relative to the original dense model, and the results are shown in Figure 5.

For the case of pruning alone without retraining (shown in green), accuracy decreases monotonically as the pruning ratio increases. This behavior is expected, since the model is not retrained to compensate for the removed network connections.

When the model is retrained only once after reaching the target sparsity (shown in blue), pruning ratios of up to 99% can be achieved without any loss in accuracy. In fact, this model slightly outperforms the original dense model. This behavior has been reported in previous studies as well [13] and can be attributed to the removal of weights that contribute minimally to the model output. By eliminating these low-importance parameters, the network is able to better focus on learning the most influential connections, leading to improved generalization. However, employing pruning ratios beyond 99% results in a collapse in the model's accuracy. This is expected, and generally occurs for pruning ratios much less than 99%.

Finally, the iteratively pruned and fine-tuned model (shown in red) achieves a pruning ratio of 99.5% with no loss in accuracy, and up to 99.9% while remaining within 5% of the original model's performance. This result is particularly impressive, as it demonstrates that a network with approximately 15 million parameters can be reduced to roughly 15 thousand parameters while still maintaining an accuracy of 95%.

Overall, this section benchmarks the impact of neural network pruning on the U-Net architecture, with results that align well with those reported in the literature. Notably, even without iterative pruning and retraining, a pruning ratio of 99% can be achieved with no loss in accuracy. Given this potential for such large compression, future work could investigate which layers

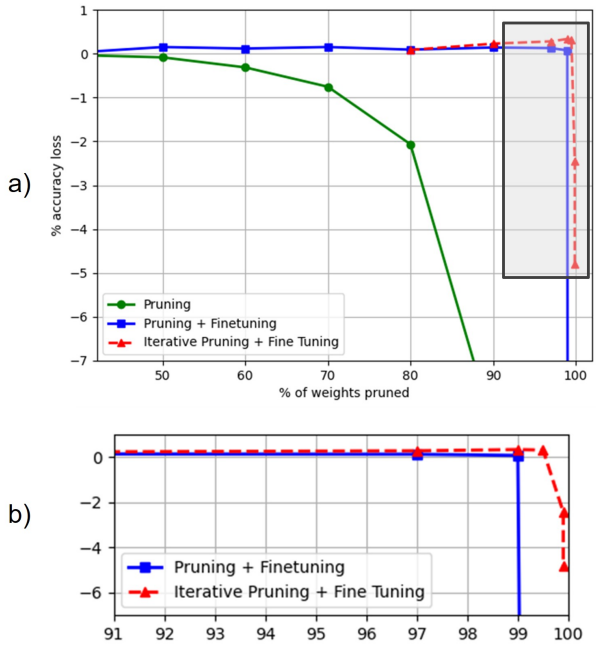

Fig. 5. (a) Accuracy loss of the pruned model as a function of pruning ratio. The green line shows pruning without retraining, the blue line shows pruning with fine-tuning, and the red dashed line shows iterative pruning with fine-tuning. (b) Zoomed-in view of the gray area in (a), highlighting the differences between the latter two methods.

contain the largest proportion of pruned parameters and use this knowledge to inform potential architectural modifications to the U-Net.

We did not evaluate the computational speedup of the pruned models, as the hardware used in this study was unable to exploit sparse computations. However, characterizing the achievable speedup on suitable hardware could be an interesting next step. Finally, as mentioned in Section I, implementing structured pruning methods and comparing their accuracy and performance benefits would also be a valuable direction for future work.

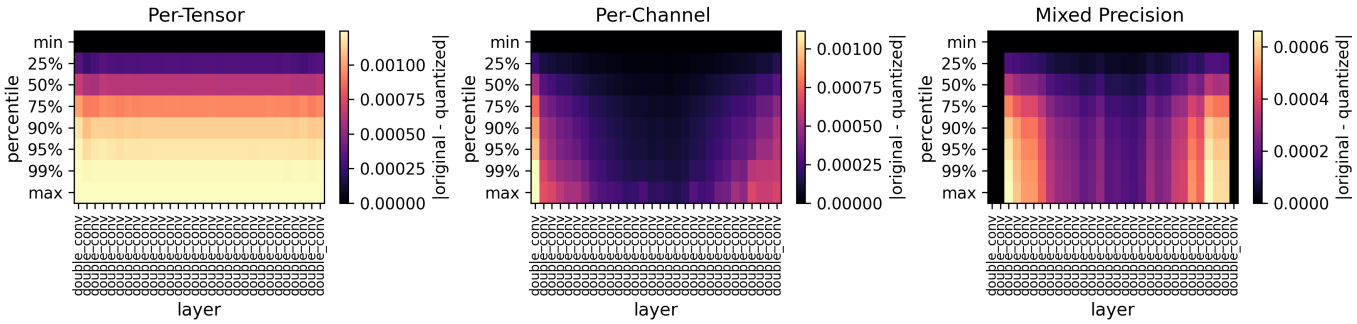

Fig. 6. This figure shows a heat map of the quantization strategy used. Each pixel is colored by the absolute residual between the original and quantized values. In the y axis, we show the different statistical measures of the weight differences while on the x axis, we show the layers.

C. Post-training Quantization

The results of the quantization strategy for the three approaches can be seen in Figure 6, which shows the corresponding residual heat maps. The y axis shows the percentiles, ranging from minim to maximum absolute different, with the median (50%) indicating the typical weight change for the quantization. The x axis shows the layers themselves, which we limit to the first 30. In short, darker colors indicate smaller weight differences (better quantization preservation), while brighter colors show larger weights differences, and hence large quantization error.

In Figure 6 (left panel), per-tensor quantization applies uniform quantization, giving similar error patterns across layers. Looking at the per channel approach, we see that the differences are mostly present in the outer layers, with the most difference being in the first layer. In the mixed precision strategy, we chose to skip the outer 2 layers. The U-Net architecture also limits us in quantization, since most of the information is transferred through a bottleneck middle layer, preventing us from quantifying these layers too much. Nonetheless, these show a quantitative view of our approach to quantization post training. All layers show similar error patterns because per-tensor quantization uses the same scale factor for the entire weight tensor.

The results of this quantization strategy can be qualitatively seen in the predictions, which we show in Figure 7. The very right column shows the difference between the original prediction by the U-Net (first column, second row) and the predicted maps of the quantized model. In this column, darker orange is better. The per tensor approach (third row from the top) shows the error relative to the original model ($|\Delta\text{orig}|$) to be the highest out of all the approaches. Note, this is across all the entire image in the masked region. This is then followed by the per channel and mixed precision, which progressively have lower residuals, consistent with expectations.

Looking at the final results of the size reduction and error increase, we get the following. The quantization study shows Mixed Precision achieves the best balance, reducing model size by $\sim 4\times$ with minimal accuracy loss (only 1% error increase vs Original). Per-Tensor provides similar compression but incurs 33% higher errors, while Per-Channel slightly under performs with higher maximum errors despite better mean error. All quantized models achieve 16.6 MB file sizes compared to the 66.2 MB Original, nearly $4\times$ compression while maintaining reasonable accuracy for BEC phase prediction.

V. Conclusion

A. Summary of Findings

We find that we are able to train and subsequently optimize a U-Net to accomplish the task of phase gradient prediction, given only density snapshots. The appropriate choice of optimizer proved to be vital for convergence during training. Furthermore, pruning and quantization greatly reduced model size and memory footprint, while maintaining high levels of accuracy.

B. Recommendations for Future Research

Future work could explore the combination of both quantization and pruning, while taking advantage of specialized hardware designed to give performance gains when utilized alongside those techniques. Techniques such as structured pruning could also be explored, to realize gains on non-specialized hardwares. Our results represent a first step towards a machine learning model that could be applied in real laboratories, pushing the boundaries of quantum science and technology.

C. Contributions

Here is a broad summary of the contributions of each of the authors. Cameron Rodriguez - pruning optimizations, writing, editing; Jackson Lee - optimizer studies, writing, editing, data generation; Fred Garcia - quantization studies, writing, editing. The authors have nothing else to disclose.

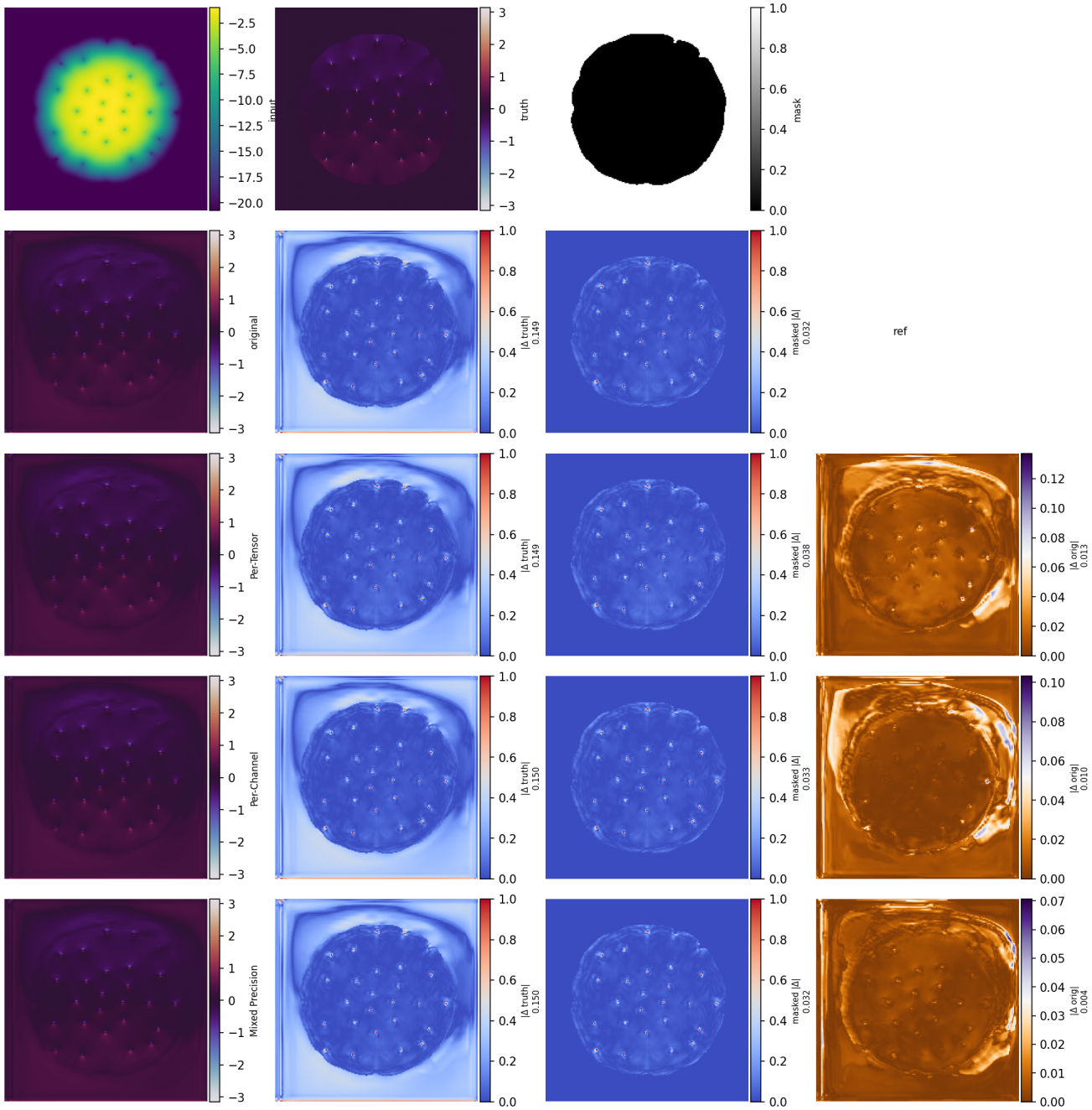

Fig. 7. Quantitative comparison of BEC phase reconstruction using quantized UNet models. (row 1) We show the reference input atom density, ground truth phase from experimental/simulated data, and mask identifying regions to ignore. For rows 2-4, for each quantization method, reconstructed phase, absolute error relative to ground truth, masked error (excluding ignored regions), and deviation from the full-precision model. The thirds row from the top is the per-tensor approach, follower by the per channel, and then the mixed precision.

ACKNOWLEDGMENT

We thank Professor Kaoutar El Maghraoui and the TAs.

REFERENCES

[1] F. Dalfovo, S. Giorgini, L. P. Pitaevskii, and S. Stringari. Theory of Bose-Einstein condensation in trapped gases. *Reviews of Modern Physics*, 71(3):463–512, April 1999. arXiv:cond-mat/9806038.

[2] F. Chevy. Vortices in Bose-Einstein Condensates: Experiments. In Panayotis G. Kevrekidis, Dimitri J. Frantzeskakis, and Ricardo Carretero-González, editors, *Emergent Nonlinear Phenomena in Bose-Einstein Condensates: Theory and Experiment*, pages 191–207. Springer, Berlin, Heidelberg, 2008.

[3] D. V. Freilich, D. M. Bianchi, A. M. Kaufman, T. K. Langin, and D. S. Hall. Real-Time Dynamics of Single Vortex Lines and Vortex Dipoles in a Bose-Einstein Condensate. *Science*, 329(5996):1182–1185, September 2010. Publisher: American Association for the Advancement of Science.

[4] Niccolò Bigagli, Weijun Yuan, Siwei Zhang, Boris Bulatovic, Tijs Karman, Ian Stevenson, and Sebastian Will. Observation of Bose–Einstein condensation of dipolar molecules. *Nature*, 631(8020):289–293, July 2024. Publisher: Nature Publishing Group.

[5] Weijun Yuan, Siwei Zhang, Niccolò Bigagli, Haneul Kwak, Claire Warner, Tijs Karman, Ian Stevenson, and Sebastian Will. Extreme Loss Suppression and Wide Tunability of Dipolar Interactions in an Ultracold Molecular Gas, May 2025. arXiv:2505.08773 [cond-mat].

[6] Kali E. Wilson, Zachary L. Newman, Joseph D. Lowney, and Brian P. Anderson. *In situ* imaging of vortices in Bose-Einstein condensates. *Physical Review A*, 91(2):023621, February 2015.

[7] Friederike Metz, Juan Polo, Natalya Weber, and Thomas Busch. Deep-learning-based quantum vortex detection in atomic Bose–Einstein condensates. *Machine Learning: Science and Technology*, 2(3):035019, June 2021. Publisher: IOP Publishing.

[8] Jing Ye, Yue Huang, and Keyan Liu. U-net based vortex detection in Bose–Einstein condensates with automatic correction for manually mislabeled data. *Scientific Reports*, 13(1):21278, December 2023.

[9] Juha Javanainen and Janne Ruostekoski. Split-step Fourier methods for the Gross-Pitaevskii equation. *Journal of Physics A: Mathematical and General*, 39(12):L179–L184, March 2006. arXiv:cond-mat/0411154.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].

[11] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

[12] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. arXiv:1711.05101 [cs].

[13] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.