CS230

Deep Networks for Optical Phase Retrieval

Hunter Swan
Department of Physics
Stanford University
orswan@stanford.edu

Abstract

Modern tools in optics allow for generation of laser beams with essentially arbitrary intensity patterns. However, in practice doing so requires solution of a computationally intensive problem know as *optical phase retrieval*, which has no known good algorithms. We investigate the solution of this phase retrieval problem via deep neural networks. While the accuracy of solutions generated in this way is not competitive with state of the art algorithms, they are adequate for many applications and have the advantage of being much faster to compute.

1 Introduction

Anyone who has played with a laser pointer is probably familiar with the fact that most laser beams look like nondescript spots. If one were to take a picture of the cross section of such a laser beam, you would find that the intensity is very nearly a Gaussian, having a form like $\exp\left(-x^2/\sigma^2\right)$. However, it is possible to shape a laser into an essentially arbitrary pattern of light. Such beams are of great technological and scientific importance, being useful for such applications as precision machining, optical levitation, and analog computing.

Generating laser beams with a desired intensity is a computationally non-trivial task. The most efficient method (in terms of laser power) involves a hard mathematical problem called *phase retrieval* (or sometimes "optical phase retrieval," to distinguish it from some closely related problems which are also called "phase retrieval"). To describe the method, we must first establish some basic laser physics: A laser's light field is described mathematically by a complex-valued vector field. The values of this vector field in any one plane determine the values at all other planes, via the dynamical equations obeyed by light. If the laser beam is directed through a lens, the functional form of the beam after the lens (briefly, "output beam") is related to that before the lens (briefly, "input beam") via a 2-dimensional Fourier transform.

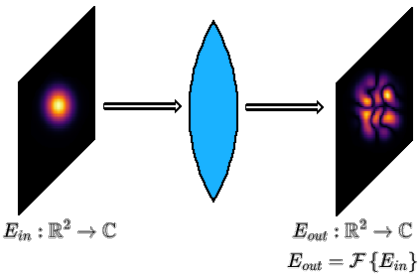

Figure 1: A lens performs a Fourier transform on the electric field E of light. Combined with the ability to modify the phase of the input beam, this allows us to create arbitrary output intensity patterns.

It turns out that for a given input beam, one can generate essentially any output beam provided one can fully manipulate the *phase* of the input beam at each point in space. The advantage of this approach is that altering the phase of light does not affect its total power, meaning this method is maximally efficient (lossless). The phase of a laser beam can be controlled effectively with a gadget called a spatial light modulator

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

(SLM). An SLM consists of an array of pixels (typically 1000×1000), each of which can impart an arbitrary constant phase to the light incident upon it. With sufficient pixel density, one can impart essentially arbitrary spatially varying phase to a laser. However, the task of determining what phase should be applied to achieve a desired output beam is highly non-trivial. This is the phase retrieval problem, and the best algorithms are approximate and slow.

The precise mathematical statement of the phase retrieval problem is:

> Given two real-valued functions $g, G : \mathbb{R}^n \to \mathbb{R}$ with $\|g\|_{L^2} = \|G\|_{L^2}$, find a phase-valued function $\phi : \mathbb{R}^n \to S^1 \subset \mathbb{C}$ such that $|\mathcal{F}\{g\phi\}| = G$, where $\mathcal{F}$ denotes the Fourier transform.

In practice, we modify this basic statement in several ways to make it easier to implement on a computer:

- Firstly, we replace ϕ above with $e^{i\psi}$ for some real valued $\psi : \mathbb{R}^n \to \mathbb{R}$, and the problem is then to find the latter. This has the practical advantage that real scalars are easier to represent than complex phases. It also introduces a few technical difficulties which we discuss later.
- Exact solutions to the phase retrieval problem can be prohibitively difficult to find, so we instead seek to minimize the error $\| \left| \mathcal{F}\left\{ge^{i\psi}\right\}\right| - G\|$ to some reasonable tolerance. Here $\|\cdot\|$ can be any convenient norm. We will use the L^2 norm in all that follows.
- Lastly, we discretize the problem onto a grid and use a discrete Fourier transform (DFT) $\mathcal{D}$ to approximate the continuous Fourier transform $\mathcal{F}$. If we use an N-point discretization in each spatial dimension, then the functions g, G become approximated by arrays $\tilde{g}, \tilde{G} : [N]^n \to \mathbb{R}$, where $[N] = \{1, 2, \ldots, N\}$. These arrays can of course be thought of as vectors in $\mathbb{R}^{N^n}$. The DFT $\mathcal{D}$ is a linear operator on the vector space of all such arrays, and we choose a normalization of the DFT such that it is unitary[1].

After these refinements of the basic phase retrieval statement, the problem we try to solve in this work is as follows:

> Given real arrays $g, G : [N]^n \to \mathbb{R}$ with $\|g\| = \|G\|$, find an array $\psi : [N]^n \to \mathbb{R}$ minimizing $\| \left| \mathcal{D}\left\{ge^{i\psi}\right\}\right| - G\|$.

Again, $\|\cdot\|$ here denotes L^2 norm. In most of this work we focus on the case $n = 1$ of one spatial dimension. The quantity $\| \left| \mathcal{D}\left\{ge^{i\psi}\right\}\right| - G\|$ we refer to as the *phase retrieval error* and denote by $E(g, G, \psi)$.

2 Related work

Laser beam shaping is a well established field, and existing approaches to solving the phase retrieval problem fall into two categories, geometric and iterative. On the geometric side, [1] summarizes the state of the art. In related work [7], the authors use a partial differential equation to solve the phase retrieval problem. On the iterative side, the paradigmatic approach is the "Gerchberg-Saxton algorithm" (GSA) [4], and generalizations due to Fienup [2]. More details on these methods can be found in the appendix 8.1.

As for neural network applications to this problem, to my knowledge no one has applied a neural network specifically to optical phase retrieval. There are some closely related problems (which are also called "phase retrieval", possibly with other adjectives) with applications in e.g. crystallography which have successfully employed deep learning techniques [8, 9]. In [8], the authors use a hybrid method, combining both conventional iterative methods and a UNet, to get improved performance on a somewhat different phase retrieval problem (distinguished from the optical phase retrieval problem by different input data). In [9], the authors use a "Y-Net" convolutional network to infer an unknown optical field from two intensity measurements in different planes.

[1] This technical detail may be ignored by readers who find it confusing. We mention it because (1) unitarity of the Fourier transform is required for the statement of the phase retrieval problem to make sense as given, and (2) most programming languages implement DFT's which are not unitary.

3 Dataset and Features

The problem we seek to solve is purely computational, and it turns out to be easy to generate unlimited quantities of data for training. The most straightforward way to do this is to pick functions $g, \psi : [N]^n \to \mathbb{R}$ at random and compute $G = \mathcal{D}\left\{ge^{i\psi}\right\}$. Then ψ provides a solution to the phase retrieval problem defined by g, G.

There is some flexibility in the distribution from which we draw g and ψ, but it is best to have them close to the functions encountered in practice. The details of how to accomplish this are relegated to the appendix 8.2. A typical example of g, ψ and the resulting G are shown in fig. 2.

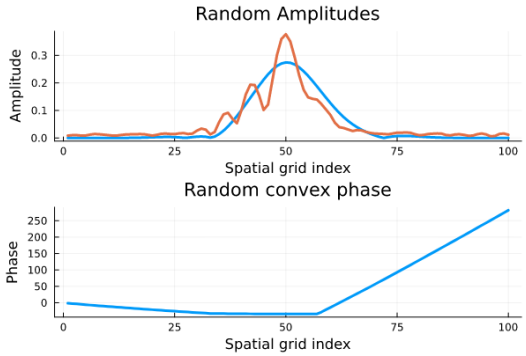

Figure 2: A typical random input and output beam amplitude and random convex phase. The input amplitude and phase are chosen independently, and the output amplitude is computed by a Fourier transform.

Since this work is meant to be exploratory, we focused on 1D data with a discretization size $N = 100$.

One additional subtlety that arose in our work concerns a method for unsupervised learning that we devised. In this approach (described below in the "Methods" section), we provide the neural network only the functions g, G, with no ψ. For this method, we don't need to compute a ψ at all, and instead generate g and G separately, using essentially the same distribution, as described in the appendix 8.2.

4 Methods

4.1 General considerations

We used the Flux.jl machine learning stack [5, 6] built in the Julia language for our experiments, for reasons of speed and extensibility. We experimented with two broad approaches to learning the phase retrieval problem, one supervised and the other unsupervised. These two methods are described in more detail below.

Within each of the approaches, we tried a variety of network architectures. Our baseline for the purposes of this class was a 6-layer dense network trained via the supervised learning approach. The metric by which we compare our models is the phase retrieval error (described in the Introduction) $E(g, G, \hat{\psi}) := \| \left| \mathcal{D}\left\{ge^{i\hat{\psi}}\right\} \right| - G \|$, averaged over a large sample of g and G, with $\hat{\psi}$ the corresponding output from the network.

For each network, we trained with an Adam optimizer and tuned the learning rate in the "panda" style, watching the loss descend until it stagnated, and then reducing the learning rate and continuing to train. We generally continued this until reducing the learning rate yielded no further benefit. An example of the loss throughout a training cycle is shown in fig. 3.

Our datasets typically had 10^5 data points, processed with minibatch size 64. Different minibatch sizes affected the training speed slightly, but since we trained essentially until the loss stagnated, the minibatch size had little effect on final accuracy of the model.

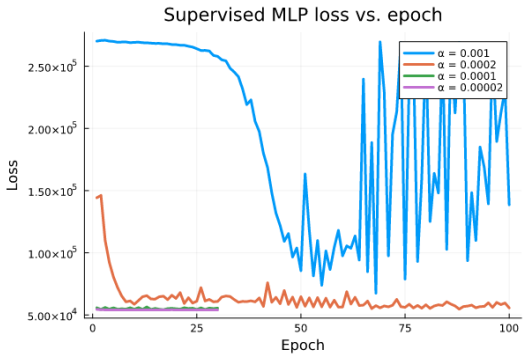

Figure 3: Loss vs. training epoch for several sequential values of the learning rate for the baseline model of a 6-layer dense network with supervised learning.

4.2 First approach: Supervised learning

To train a network via supervised learning, a single datum consists of a pair (g, G) (the input to the network) and a phase ψ. The network predicts a phase $\hat{\psi}$ and the loss is the mean-squared error (squared L^2 norm distance) $\|\psi - \hat{\psi}\|^2$.

In addition to the baseline 6-layer dense network, we also tried a basic convolutional network with this approach. As described in the "Experiments/Results/Discussion" section, the performance of these networks was rather poor.

4.3 Second approach: Unsupervised learning

To train a network via unsupervised learning, we again provide the network a pair (g, G) as input and receive $\hat{\psi}$ as output. However, as loss we directly use the phase retrieval error $\| \left| \mathcal{D} \left\{ g e^{i\hat{\psi}} \right\} \right| - G \|$, which is the metric we seek to minimize.

The main difficulty in implementing this method is that the usual implementations of the discrete Fourier transform are not compatible with differentiable programming. In particular, the main fast Fourier transform (FFT) implementation in Julia (FFTW [3]) cannot be used in a Flux.jl model or loss function. To work around this, we implemented our own version of a DFT. The details are described in the appendix 8.3.

With this unsupervised method, we again implemented a 6-layer dense network and basic convolutional network.

5 Experiments/Results/Discussion

The metric we optimize for is $E(g, G, \hat{\psi}) = \| \left| \mathcal{D} \left\{ g e^{i\hat{\psi}} \right\} \right| - G \|$. For comparison, with $N = 100$, running 1000 iterations of the GSA for randomly generated g, G (as in our data generation scheme described above) yields a mean $E(g, G, \hat{\psi}) \approx 0.014$.

Qualitatively, the first important result is that the supervised learning method did not perform well for any of the networks we tried. In all cases, the loss stagnated quickly during training. Though the model output phases bore some aesthetic similarity to the ground truth phases, they failed to attain quantitative agreement even on training set data.

By contrast, networks trained via the unsupervised method worked reasonably well. All models were readily trainable, and the output phases solved the phase retrieval problem reasonably well, as quantified by the phase retrieval error.

Table 1 summarizes metric performance of various models and GSA.

A typical output beam amplitude produced by Fourier transforming a given input amplitude times a phase output by one of these models is shown in fig. 4.

Method	GSA	Supervised		Unsupervised	
Model		MLP	CNN	MLP	CNN
Metric	0.014 ± 0.016	0.896 ± 0.157	1.044 ± 0.182	0.070 ± 0.061	0.081 ± 0.070

Table 1: Metric performance for various models and algorithms. Listed values are mean $\pm$ one standard deviation, evaluated on a test set of 10^4 examples.

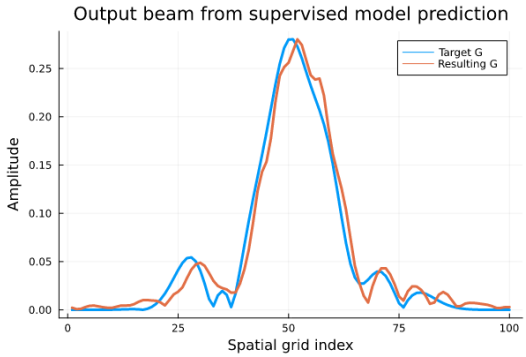

Figure 4: Comparison between a target output amplitude and the output amplitude produced by Fourier transforming the given input amplitude times the phase output by a 6-layer dense network, trained with the unsupervised method. The close agreement between the two curves indicates a small phase retrieval error in this case.

A noteworthy feature of the models we trained is that they run significantly faster than GSA, by a factor of about 50.

5.1 Hyperparameter tuning and regularization

As mentioned, the learning rate was tuned by hand by watching the progression of the loss during training. Since we trained to stagnation, the mini-batch size had little effect on the final model performance.

Since we had access to unlimited data, we did not need to use regularization. We initially used 10^4 data points for training and observed some overfitting with the unsupervised method (test set loss higher than training set loss), but after increasing the dataset to 10^5 points this went away.

6 Conclusion/Future Work

Though neither of our methods surpassed the standard GSA for solving the phase retrieval problem, the unsupervised method came reasonably close. It may have the potential to achieve higher accuracy than GSA with more refinement and training and a more sophisticated architecture. A significant advantage of our deep learning approach to the phase retrieval problem is that, after training, computing solutions to phase retrieval problems is substantially faster than GSA.

Directions for future work include:

- Try more sophisticated network architectures. Our work was substantially limited by available memory on the computers we used for training. With more computing resources we expect larger networks to give superior performance.
- Experiment with phase retrieval in 2 dimensions, which is the most useful for laser beam shaping applications.
- Implement a differentiation rule for FFTs in Flux.jl, as described in the appendix 8.2. This would provide an order of magnitude speedup for the DFT layer in the loss function and backpropagation.

7 Contributions

I performed all work described here solo.

References

[1] Fred M Dickey. *Laser beam shaping: theory and techniques*. CRC press, 2018.

[2] James R Fienup. "Phase retrieval algorithms: a comparison". In: *Applied optics* 21.15 (1982), pp. 2758–2769.

[3] Matteo Frigo and Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation", pp. 216–231. DOI: 10.1109/JPROC.2004.840301.

[4] R. W. Gerchberg and W. O. Saxton. "A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures". In: *Optik* 35.2 (1972), pp. 237–246.

[5] Michael Innes et al. "Fashionable Modelling with Flux". In: *CoRR* abs/1811.01457 (2018). arXiv: 1811.01457. URL: https://arxiv.org/abs/1811.01457.

[6] Mike Innes. "Flux: Elegant Machine Learning with Julia". In: *Journal of Open Source Software* (2018). DOI: 10.21105/joss.00602.

[7] Alexander Kratsch et al. "Solving the logarithmic Monge-Ampère equation with a RK4-algorithm for beam shaping purposes of femtosecond laser beams with spatial light modulators". In: *Laser Resonators, Microresonators, and Beam Control XX*. Vol. 10518. SPIE. 2018, pp. 259–270.

[8] Baopeng Li et al. "Phase retrieval based on difference map and deep neural networks". In: *Journal of Modern Optics* 68.20 (2021), pp. 1108–1120.

[9] Rujia Li et al. "Physics-enhanced neural network for phase retrieval from two diffraction patterns". In: *Opt. Express* 30.18 (Aug. 2022), pp. 32680–32692. DOI: 10.1364/OE.469080. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-30-18-32680.

8 Appendix

8.1 Literature of phase retrieval

In the analytical theory of phase retrieval, there are certain special cases (low dimension or high symmetry) which can be solved exactly in an appropriate limit (geometric optics) that is often met in practice. The theory of such solutions provides substantial insight into the nature of solutions to the phase retrieval problem. This theory is detailed in [1].

Generic laser beam shaping problems do not strictly meet the requirements for the analytical theory, but in many applications the analytics are close enough. When this is not the case, there is a certain partial differential equation (the Monge-Ampere equation) which handles more complicated geometries [7]. However, this PDE is highly non-linear and horrendously difficult to solve, significantly limiting its practicality.

The analytic theory is only valid in a certain limit (geometric optics), and when going beyond this limit, the only show in town is iterative methods. The paradigmatic approach here is the "Gerchberg-Saxton algorithm" (GSA) [4], which uses fast Fourier transforms (FFTs) and an iterative update rule to approximate solutions to the phase retrieval problem. This method is slow and only moderately accurate, but it is very general and robust. There are several variations of GSA which are sometimes used, such as the "Hybrid Input Output" algorithm [2]. Some variations improve accuracy in some regions of space at the expense of others, but none substantially improve the basic work flow.

8.2 Dataset generation

In principle, we could choose the g, ψ of the phase retrieval problem via any distribution, but in practice it is advantageous to choose them close to what we expect in applications. To elaborate, there is a theoretical justification for the existence of a true solution map $(g, G) \mapsto \psi$, arising from optimal transport theory. On the basis of this, we would expect that a neural network which is sufficiently expressive to approximate this map should be trainable on triples produced as above

via any distribution which covers the space of all g, ψ reasonably well. However, in practice this is of course intractable for even moderately fine discretizations, due to the high dimensionality of the spaces involved. Hence we focus our training on g, ψ chosen to look close to what we expect in practical applications.

In applications to laser beam shaping g represents an input laser beam. Commonly occurring laser beams are typically well approximated by low-order expansions of Hermite-Gaussian polynomials, and this is how we generate random g for our data sets. To avoid undesirable boundary effects, we make sure that the characteristic size of g is a bit smaller than the width of the discretization grid.

When generating data for our unsupervised learning method, we use essentially the same Hermite-Gaussian expansion for both g and G. The only difference between the two is that we allow G to have more spectral content in higher modes, reflecting the fact that in applications the output beam typically has more flexible geometry.

From phase retrieval theory, it is known that solutions ψ to the phase retrieval problem are typically close to being convex. Moreover, there are specific quantitative constraints on the gradient of ψ in terms of the support of the functions g, G. On the basis of these theoretical considerations, when generating random phases ψ we choose them to be convex and have an overall scale which is consistent with the domain over which we discretize g, G. It turns out that for an N point discretization, the second derivative of ψ should be of order $\lesssim \pi/N$

A technical difficulty in training models on this data is that there are certain degeneracies which affect the phase ψ. In particular, given a solution to the phase retrieval problem ψ_0, we may generate another solution by either of the following operations:

- For any $c \in \mathbb{R}$ we may add c to each component of ψ_0. This is the *global phase shift invariance*, and is an intrinsic property of the phase retrieval problem.
- To each component of ψ_0 we may add an arbitrary multiple of 2π. This is the *local phase ambiguity*, and is a consequence of our representation of phase retrieval solutions by real-valued arrays, rather than phase-valued (i.e. taking values in $S^1 \subset \mathbb{C}$) arrays.

Our method for generating convex phases mostly resolves the second issue (provided the curvature of the phase is sufficiently small, which is satisfied in practice). To fix the first issue, we fix all phases to be zero at a prescribed point (the left edge of the domain).

8.3 Differentiable DFT implementation

There are two ways to implement a differentiable DFT in Julia:

- The most straightforward solution is to realize the DFT as a matrix operation. This can be automatically differentiated by Flux.jl. The downside of this approach is that the DFT now has $\mathcal{O}(N^2)$ complexity, which slows down data generation and training compared to an FFT implementation.
- Alternatively, Flux.jl allows one to implement a custom differentiation rule for an operation like an FFT. The backpropagation rule for an FFT layer turns out to be simply an inverse FFT. This has the benefit of an $\mathcal{O}(N \log(N))$ complexity.

For reasons of simplicity and available time, we implemented the first option. With our chosen grid size $N = 100$, a matrix multiplication DFT is about 5 times slower than an FFT. Training models incorporating a matrix DFT is still feasible.