CS230

Predicting a fiber Bragg grating's parameters from its transmission spectrum using deep learning

Arushi Arora
Ph.D. Candidate
Department of Electrical Engineering
Stanford University
arushi15@stanford.edu

Abstract

There have been recent advancements in developing strong and narrow resonances in optical fiber Bragg gratings for use as ultra-sensitive strain, pressure and temperature sensors. The sensors are fabricated with certain physical parameters, such as the induced index change, length, and loss, which decide the sensor performance but are unfortunately not known exactly at the time of fabrication. Once the sensors are fabricated, they are characterized, and scientists must use the measured transmission spectrum to find these physical parameters, using which they can design better sensors in the future. While existing methods can easily predict the sensor's transmission spectrum given the physical parameters, the inverse problem of finding the physical parameters from a given transmission spectrum is a much harder problem. This project applies deep learning to the latter problem, that is: Given the transmission spectrum of an optical filter, specifically a fiber Bragg grating, can a neural network predict its essential physical parameters? This project shows that a surprisingly simple neural network consisting of three dense layers is able to learn the relationship between the transmission spectrum and the physical parameters, and can predict the values of the physical parameters with low error.

1 Introduction and Related Work

A fiber Bragg grating (FBG) is a short length (a few mm to a few cm) of optical fiber along which the refractive index of the core region is periodically modulated on a sub-wavelength scale. FBGs have been widely used in many applications including filters in optical communication systems [1], cavity mirrors in lasers [2], and ultra-sensitive strain and pressure sensors [3]. Their typical transmission spectrum is illustrated in Fig. 1. There are two elements of interest in the figure; the first is the large window of wavelengths for which the FBG acts as a strong reflector, the second is the series of strong, narrow slow-light peaks observed on the left edge. These elements (width of the wide reflection peak and the number, width and amplitude of the narrow slow-light peaks) are controlled by the following physical parameters of the FBG: length, index modulation amplitude, type of apodization, period, and FBG loss. The values of these parameters in turn depend on the how the FBG was fabricated, such as the laser pulse intensity, number of laser pulses, and fiber composition, called fabrication parameters.

For sensing applications we wish to create slow light peaks as narrow and strong as possible. In effect, this requires us to optimize the fabrication parameters. However, there are several challenges in this study. One, the relationship between the fabrication parameters and the induced physical parameters is not well known. Thus, typically, the FBG is written with the values of fabrication parameters that have been honed with trial and error. Two, once the FBG is characterized by measuring the transmission spectrum, the analytical calculation of its physical parameters is non-trivial. There is a

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

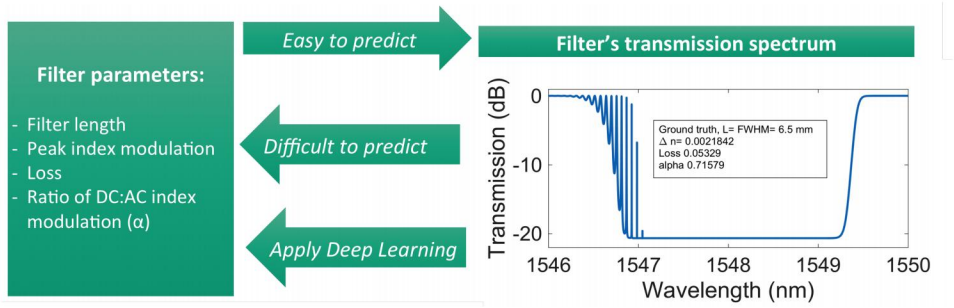

Figure 1: Existing codes can analytically solve for the transmission spectrum (on the right) if given the values of the fiber parameters (displayed inside the spectrum). A deep learning algorithm is applied for going in the reverse direction.

real-coded genetic algorithm that proposed this calculation in [4]. A Markov decision process was employed in optimizing similar structures in [5]. On the other hand, predicting the transmission spectrum from the values of the physical parameters is relatively simple using existing analytical methods, such as the transfer matrix method which solves for the transmitted and reflected electric field amplitudes using electromagnetic theory [6]. Thus, scientists (many of those are graduate students) typically use the latter theory and spend days tuning the values of the physical parameters until the calculated spectrum fits the measured spectrum.

To create an alternative to this time-consuming fitting procedure, this project creates a neural network that learns the correspondence between a transmission spectrum and its corresponding physical parameters. The input to the network are features of the transmission spectrum and the output is a vector of the predicted parameter values. The algorithm reduces the time spent in predicting the physical parameters for a particular spectrum from days to seconds. This speeds up the research efforts to create better ultra-sensitive FBG sensors.

2 Dataset and Features

Essentially, the goal is to use the measured transmission spectrum as the input to the algorithm, which outputs a vector containing the values of the FBG's normalized physical parameters (see Fig. 1). Two examples in the dataset are shown in Fig. 2. Using the two sets of parameter values in the existing analytical code [6], the transmission spectra in the center, Fig. 2a and 2c, are generated. As is observed, the spectra are in stark contrast to each other because the value of the α parameter in Fig. 2c is almost half of the value used for Fig. 2a. While Fig. 2a exhibits a great number of narrow peaks on the left-edge, Fig. 2c barely exhibits any peaks. In addition, the resonances in Fig. 2a are of different heights and widths with the peaks becoming progressively narrower with increasing wavelength as seen with better resolution in Fig. 2b and 2d. Clearly, the spectra are highly dependent on the values of the parameters and must be encoded carefully so that the neural network can make the correct predictions.

There could be several ways to encode the input:

1. Use an image of the transmission spectrum: to resolve the number, height and width of a large number of slow-light peaks on the left edge of the spectrum, the image must be of suitably high resolution and size. Since typical algorithms require all input images to be of the same size, we would need 2046 x 2046 or larger resolution images, which will tend to slow down the data processing. In addition, encoding the image itself will also lead to loss of information. Thus, it is not the most accurate way to represent the data.

2. Vectorise the transmission spectrum as $(x_1^{(i)}, x_2^{(i)})$ where $x_1^{(i)}$ is a vector of wavelengths from 1549 nm to 1555 nm and $x_2^{(i)}$ is a vector of the corresponding transmitted power $T^{(i)}$ at those wavelengths. This sounds promising, however, leads to the same problem of resolution. To be able to resolve the highest-density image containing say, twenty slow-light peaks, the vector length for one image shall be 9 million points, or two times nx= 9 million. Since all the other inputs must also have the same size, even images that contain no peaks at

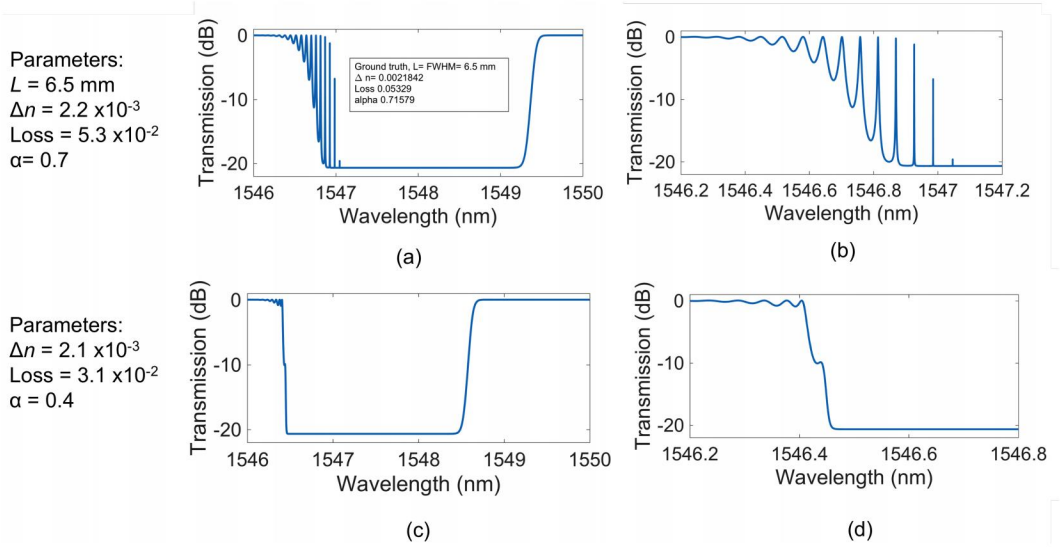

Figure 2: Two examples in the dataset are shown. Using the values of the parameters (on the left) in an existing analytical code, the transmission spectra in the center ((a) and (c)) are generated. The figures are zoomed into their left band-edge to see the peaks with better resolution on the extreme right ((b) and (d)).

all, will be resolved with 9 million points, which is unnecessary. It will cause the weight matrices to be huge and bulky and difficult to train and store.

3. Parameterize the input data as follows (see Fig. 3):
   (a) Let the maximum number of possible peaks maxpk be 100.
   (b) Create an input vector of size (5*maxpk +1, m) where m = number of training samples
   (c) For each peak in each training sample, create a vector as follows:
      i. Presence of peak = 0 (if no peak) or =1 (if peak exists)
      ii. T_{abs}= absolute height of peak (real number between 0 and 1)
      iii. Distance from previous peak = 0 (if first peak), belongs to a real number $(0,\infty)$
      iv. Peak width at half prominence is real number between 0 and 1 (see below)
      v. Peak prominence is a real number between 0 and 1, that measures the eminence of the peak compared to its neighbors [7]
   (d) The last element of the input vector is the bandwidth of the entire spectrum, which is given by the 3-dB width between the left and right edges of the spectrum

Since this is a fairly niche problem, no publicly available dataset exists. Thus, the dataset was created in-house. First, a MATLAB code, based on the transfer matrix theory and written by graduate students in Prof. Michel Digonnet's group at Stanford University, was used to generate the transmission spectra for various parameter values. For this initial project, the fiber length was kept constant as 6.5 mm, and the data was created by only sweeping across the values of peak index modulation, loss, and α (ratio of DC:AC index modulation). With total samples m = 6532, the output matrix 'Y' was of shape (3,m).

- Δn is sampled linearly from 0.001 to 0.0035
- Loss is sampled logarithmically from 0.01 to 2
- α is sampled linearly from 0.1 to 1

Each of the three numbers was then normalized to the largest value present in each row (0.0035,2 and 1) before being trained by the neural network. The predictions were later scaled up by the same factor to generate the actual values of the parameters.

Second, I wrote a code to extract the features from each calculated spectrum as described above and store them inside a matrix 'X', which was then used as the input to the neural network. The matrix

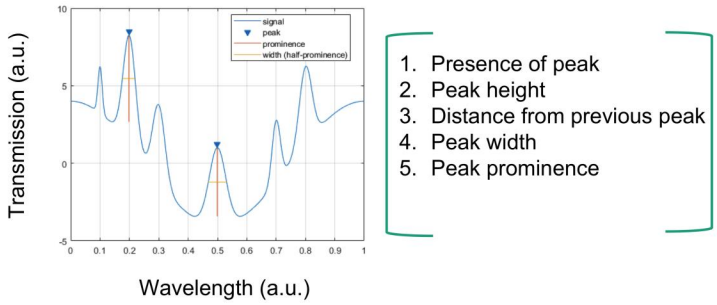

Figure 3: Feature extraction from the transmission spectrum. For each peak in each spectrum, the above listed five features were extracted. Left figure adapted from [7].

Figure 4: Neural network architecture

'X' of shape (501,m) consists of m total samples with each sample consisting of a vector of 501 extracted feature values. The total number of samples were divided into training, validation and test datasets in the following ratio $60\% : 20\% : 20\%$.

3 Methods

The neural network consisted of three fully-connected dense layers of 200, 60 and 3 hidden units respectively (see Fig. 4). The initial layers used ReLU activation while the last layer used a sigmoid activation so that the outputs lay between 0 and 1. The predictions were then multiplied by the scaling constants (0.0035,2,1) to give the actual predictions.

The loss metric for evaluating the performance of the network and optimizing the weights was chosen to be the mean squared error, defined as

$$MSE = \frac{\sum_{i=1}^{3}(Y_{predicted}^{(i)} - Y_{true}^{(i)})^2}{3}$$

The Adam optimizer was used with the default parameters introduced in the original publication of [8]. The code was written with the help of Keras [9], a deep learning Python library that runs on top of Tensorflow [10], another open-source Python library. Keras and Tensorflow were chosen because of their helper functions that made writing this deep learning algorithm much easier and faster.

4 Results and Discussion

As expected, the training loss decreased with increasing number of epochs (see Fig. 5). The training loss converges quickly to a low value of 0.03 after approximately 250 epochs. Since the error on the validation set closely matches the training set, (expected since they come out of the same distribution), the network has not over-fit to the training data. Slightly wider or larger networks gave no substantial improvement in performance. This may indicate that the training dataset size and/or the way the features are encoded are in the input may need to be improved. With such a small dataset, iterating through the entire dataset every epoch was not time-consuming, thus the batch size was kept the same as the number of training samples. In the future when training with a larger amount of data, the batch size may need to be reduced.

Two examples from the test set are shown in Fig. 6. Fig. 6a shows an example where there is an excellent match between the ground truth and the predicted parameter values. These values were then used to generate the predicted curve (red) shown in the bottom figure. The excellent match between the curves leads to the same conclusion of the algorithm having learned the correspondence between

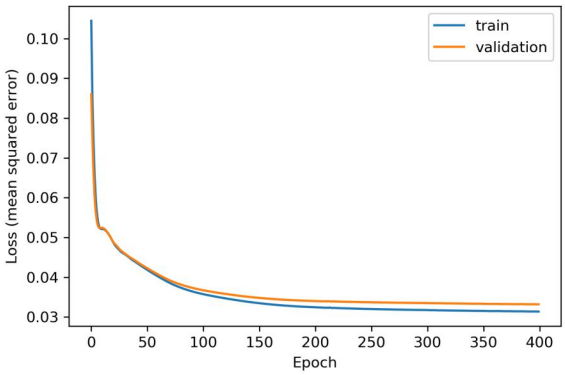

Figure 5: Loss versus number of epochs on the training and validation datasets.

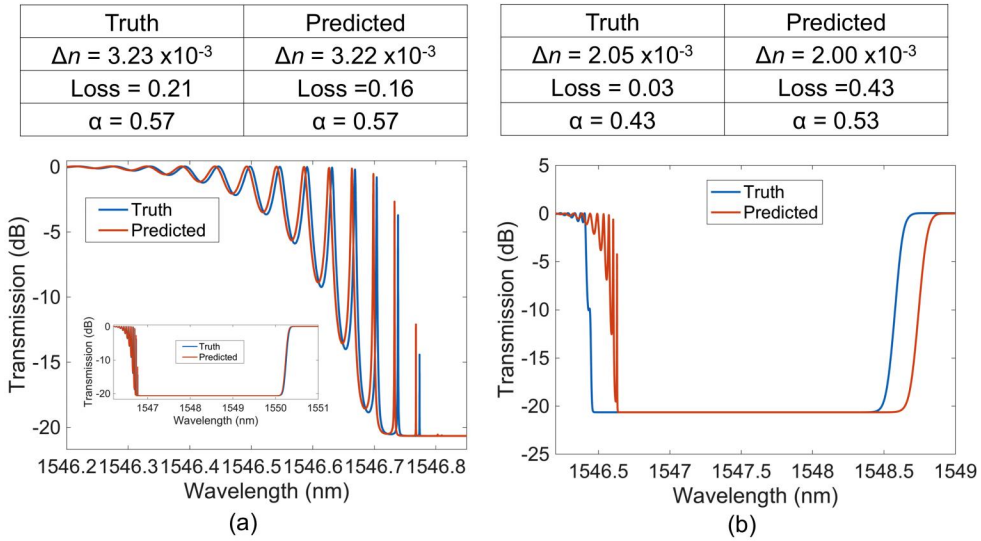

Figure 6: Two examples from the test set where (a) exhibits an excellent match between ground truth and the predicted parameter values, and (b) with a not-so-good match.

the input and output very well. On the other hand, Fig. 6b shows an example where the algorithm doesn't do so well, which is typically in cases where there are a small number of peaks. The predicted loss and α are much too high compared to the ground truth. The reason for the prediction failure could be due to not having enough training samples for these cases, or that the spectral shift that was not taken into account in the parameters fed to the algorithm. Using algorithms that automatically find the features of the spectra (such as LSTM) instead of hand-engineering the features as done here may possibly improve performance in such cases. Given the limited time, these can be addressed in future versions of this project.

5 Conclusion and Future Work

A deep learning algorithm was employed to predict the parameters of an optical fiber Bragg grating from its transmission spectrum. The features were extracted from the spectrum and sent through a neural network to find the predicted parameters. It was found that an extremely simple neural network performs excellently on most cases, reducing the time for predictions from days to seconds. This shall be an excellent resource for scientists who fabricate and design fiber Bragg grating based sensors and improve the efficiency of their research.

6 Acknowledgements

Many thanks to Jay Whang, TA for CS 230 for his help in figuring out the best way to featurize the input data and for his excellent feedback. I also thank all CS230 instructors and other TAs for the invaluable teaching material and helpful office hours.

References

[1] G. P. Agrawal, and S. Radic, "Phase-shifted fiber Bragg gratings and their application for wavelength demultiplexing", IEEE Photon. Technol. Lett. 6, 995 (1994).

[2] G. A. Ball and W. H. Glenn, "Design of a single-mode linear-cavity erbium fiber laser utilizing Bragg reflectors", J. Lightwave Technol. 10, 1338 (1992).

[3] G. Skolianos, A. Arora, M. Bernier, and M. J. F. Digonnet, "Slow light in fiber Bragg gratings and its applications," Journal of Physics D: Applied Physics 49, 463001 (October 2016).

[4] G. Cormier, R. Boudreau, and S. Theriault, "Real-coded genetic algorithm for Bragg grating parameter synthesis", J. Opt. Soc. Am. B 18, 1771 (2001).

[5] T. Hughes, and Y. Shi, "Optimization of Optical Structures Using Markov Decision Processes" CS 229 project (2015). `http://cs229.stanford.edu/proj2015/342_report.pdf`

[6] T. Erdogan, "Fiber grating spectra", Journal of lightwave technology 15, 1277 (1997).

[7] See MATLAB findpeaks function `https://www.mathworks.com/help/signal/ref/findpeaks.html#buff2uu`

[8] D. P. Kingma, and J. L. Ba, "Adam: a Method for Stochastic Optimization", International Conference on Learning Representations, 1–13 (2015).

[9] Francois Chollet. Keras. `https://github.com/fchollet/keras` (2015).

[10] Abadi M. et al. "TensorFlow: Large-scale machine learning on heterogeneous systems", (2015) Software available from tensorflow.org.