MOSES Data Inversion With Convolutional Neural Networks

Roy Smart

Montana State University
roytsmart@gmail.com

I. Introduction

The *Multi-Order Solar EUV Spectrograph* (MOSES) is a unique instrument developed by Charles Kankelborg's research group at Montana State University that captures solar images in three diffraction orders: $m = 0, +1,$ and -1. Unlike most spectrographs, MOSES is not equipped with a slit to restrict the field of view on the dispersion axis. The advantage of this configuration is that spectral and spatial information is simultaneously viewed by the instrument, allowing interesting solar features to be more easily identified. However this lack of spatial restriction by a slit means that spectral and spatial information are convolved together to form the images captured by MOSES. These images are known as *overlappograms*.

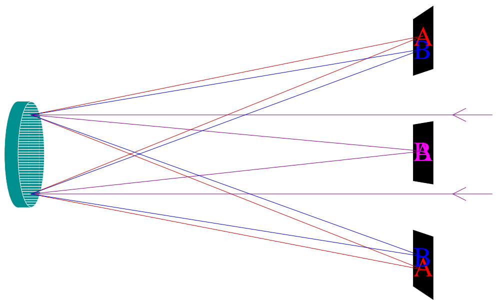

Figure 1: *Layout of the MOSES instrument demonstrating how different wavelengths are overlapped onto each detector [1].*

The main objective of the MOSES instrument is to determine Doppler shifts of the structures observed in the solar transition region. To find this quantity, we must perform what we call an *inversion*. This is best described as taking the 2-dimensional spectral and spatial information from each of the three detectors and constructing a spectral *cube* in three dimensions, with two spatial dimensions and one spectral dimension.

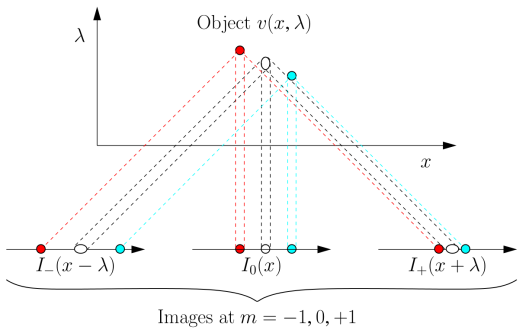

Figure 2: *Diagram demonstrating how compact objects in the data cube are convolved to form the images at each detector [1].*

The main challenge with inverting the MOSES data is that it is an obviously ill-posed problem, i.e. the spectral cube contains more information than is provided by the instrument. This is mathematically described using the expression

$$I_m(x', y') = \int_B v(x' - m\lambda, y', \lambda) d\lambda. \quad (1)$$

Where $I(x', y')$ is the intensity measured by the instrument, $v(x', y', \lambda)$ is an object located in the spectral cube, m is the spectral order, (x', y') are the detector coordinates, and the domain B is the passband of the instrument.

Since Equation (1) doesn't have a unique solution, there are many possible cubes that could produce the same images captured by MOSES. Consequently, we must use physical constraints to attempt to trim down the large number of potential solutions to the inversion problem.

II. Motivation

Many researchers throughout the history of the MOSES research project have developed computational tools to solve the inversion problem. These methods include Smoothed Multiplicative Algebraic Reconstruction Technique (SMART)[2], Fourier backprojection and pixon inversion [3] and are discussed in detail in the literature.

The above methods have proved successful in revealing the interior structure of the transition region [1]. However there is still room for progress to be made towards full inversions of the entire MOSES dataset. These algorithms have a tendency to produce what is known as *plaid*. This is a phenomenon where bright objects on the spectral cube tend to be smeared across the direction of the spectral projections, producing decidedly unphysical inversions.

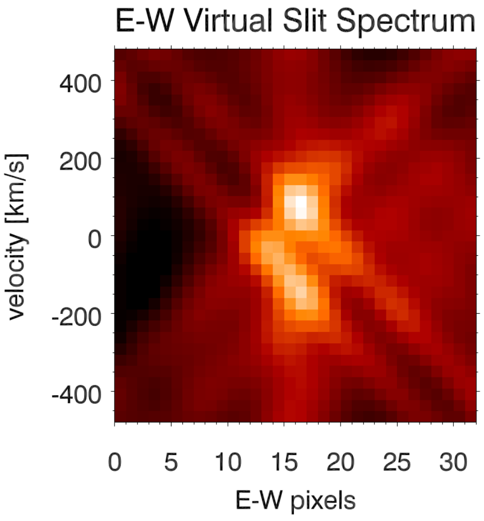

Figure 3: *An example of plaid. Notice how the bright pixels in the center are smeared diagonally (projection dimension) across the image. This is an example of an unphysical inversion. [4]*

Developing an inversion method that is resistant to plaid would be very beneficial to Kankelborg's research group, as such an inversion would allow researchers to extract more scientific information about the structure of the transition region.

III. Proposed Inversion Method

I. Machine Learning

Thus far, all of the attempts to invert the MOSES data have relied on traditional, human-designed algorithms to carry out the computation. However, in the past few years, great strides have been made in machine learning, where a computer program is trained to solve a particular problem by providing examples of the problem along with each corresponding solution. Feedback is then provided to the network so that it may utilize minimization techniques to converge on a solution.

I propose that a method based off of machine learning might allow some progress to be made towards the goal of plaid-resistant MOSES inversions. This is made possible by the fact that the learning algorithm could be penalized for producing plaid solutions to the inversion problem and rewarded for producing physical solutions.

While machine learning sounds like a fantastic approach to problem solving, there is still a great deal of challenges to contend with. First, the structure of the learning algorithm is very important in determining its problem-solving ability, and unlike traditional algorithms, the right structure can often only be found by trial and error. Furthermore, the learning process takes a large number of examples to converge on the solution. Solving these problems will be discussed in the following pages.

II. Artificial Neural Networks

Artificial neural networks (ANNs) are machine learning algorithms that are modeled off of the structure of the animal brain. ANNs can be described as a system of *neurons* that have the ability to communicate with each other. The behavior of each neuron is dictated by an activation function. This function maps neuron inputs to outputs, e.g. determines when the neuron is *active*. Active neurons propagate information through the network while dormant neurons restrict it.

Communication is facilitated through a series of connections between the neurons, where each connection has an associated weight representing the amount of influence one neuron may have on another. The weights are tuned based off of experience and taken with the activation function, allow the neural network to learn.

The neurons are usually organized into layers, where each layer can be interpreted as a different computational step used to solve the problem. The first layer is known as the input layer, and the number of input neurons corresponds to the amount of information supplied to the network. The output layer is the last layer, and the number of output neurons equals the amount of information to be expected in the solution of the problem being solved. In between the input and output layers, there are a number of hidden layers used to compute the solution. The amount of hidden layers and the number of neurons per hidden layer depends on the complexity of the problem.

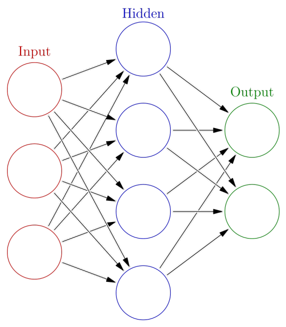

Figure 4: *An example of a simple neural network.*

Using this simple idea, many different species of neural networks may be constructed. The choices of network depth, number of connections, and activation function are examples of what are known as *hyperparameters* which determine the behavior of the network.

III. Convolutional Neural Networks

Vanilla ANNs usually connect each neuron in one layer to every other neuron in the preceding layer. This configuration is sufficient when the number of input neurons is small. However, as the number of input neurons increases (for example, as large as an image), the huge number of free parameters quickly leads to overfitting. Furthermore, the Vanilla ANN treats pixels with large spatial separation equally, which is often unnecessary for image processing where close spatial relationships are more important.

Convolutional Neural Networks (CNNs) solve this problem by connecting only a small number of the input neurons (known as the receptive field) to the layers below, known as convolutional layers.

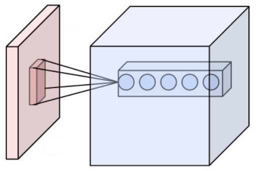

Figure 5: *The red box represents the input neurons, with the receptive field connected to the next five convolutional layers.*

The receptive fields are tiled such that they overlap one another. Overlapping receptive fields, taken with weight sharing in each convolutional layer, provides translational invariance. Translational invariance is very important as physically valid solutions will exhibit this type of symmetry. There are many other layers that go into a CNN that will not be discussed here, but they include: pooling, RELU, and fully connected layers. These layers serve to increase the computational power and decrease the training time of the network.

CNNs have been used successfully in many fields, most notably in image recognition [5]. We are confident that convolutional neural networks will be best suited to MOSES data inversion because spectral features in MOSES images are separated spatially, and CNNs are optimized to detect spatial features in images.

IV. Implementation

I. Selecting Hyperparameters

We are not yet sure how best to select the hyperparameters for the network. Most networks in the literature consist of approximately five convolutional layers, one or more pooling layers, one or more RELU layers, and one fully connected layer. Determining the topography of the network is one of the major challenges that lies ahead.

II. Training Dataset

The training dataset needs to contain the data gathered by MOSES and the spectral cube that produced the result. Since we do not yet know the answer to the MOSES inversion problem, we will need to look to other sources to provide the training data. An obvious choice is the data collected by the IRIS satellite. IRIS is a traditional slit spectrograph, and thus has independent dimensions for spectral and spatial data.

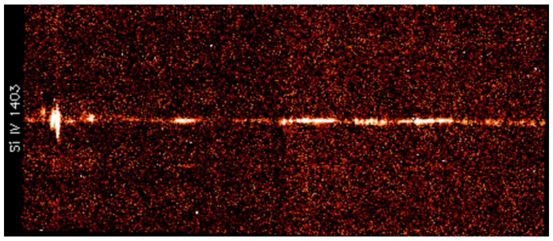

Figure 6: *An example of the data collected by IRIS in the Si IV line. The horizontal dimension is spatial, while the vertical dimension is spectral [6].*

The idea then, is to use the information collected by IRIS to simulate MOSES data. IRIS has collected a large dataset during its mission, so there should be sufficient data available to train the CNN.

III. Hardware

Training large, intricate neural networks has only recently become tractable for personal computers. While CPUs have only been demonstrating modest floating-point performance gains over the past few years, GPUs have been following an exponential trend.

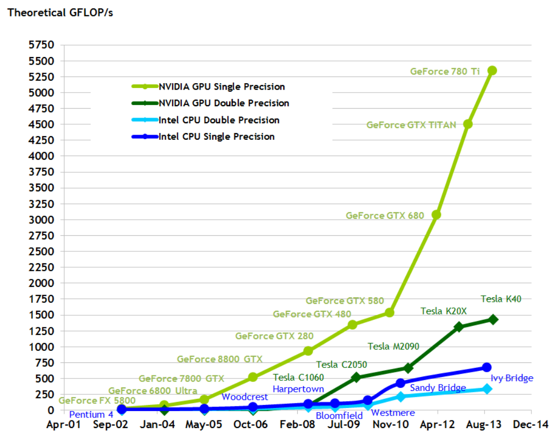

Figure 7: *Comparison of GFLOP/s for CPU and GPU [7].*

Furthermore, GPUs are designed with massive parallelism in mind, which can be utilized by a neural network algorithm. Therefore, it is desirable to take advantage of current GPU performance to quickly train neural networks.

IV. Software

Developing such an advanced neural network would take a great deal of time, especially for a GPU implementation. Luckily enough, one has already been developed for us; *Caffe* is a convolutional neural network program, developed at Berkley for image processing [8].

Caffe allows the user to define their network as a simple script, and then it trains the network on an Nvidia GPU automatically. Once the network is trained it may be executed on either a CPU or GPU. This will prove useful as we can guarantee that the network will still be usable, regardless of hardware developed in the near future.

References

[1] Charles C. Kankelborg J. Lewis Fox and Roger J. Thomas. A transition region explosive event observed in He II with the moses sounding rocket. *The Astrophysical Journal*, 719:1132–1143, August 2010. `http://solar.physics.montana.edu/MOSES/papers/2010/FoxKankelborgThomas2010.pdf`.

[2] T. Rust, L. Fox, C. Kankelborg, H. Courrier, and J. Plovanic. MOSES Inversions using Multiresolution SMART. 224:414.06, June 2014.

[3] J. L. Fox, C. C. Kankelborg, and T. R. Metcalf. Data inversion for the Multi-Order Solar Extreme-Ultraviolet Spectrograph. 5157:124–132, November 2003.

[4] T. Rust and C Kankelborg. Imaging spectroscopy with moses sounding rocket data. *SPD*, 2015.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[6] iris rast cii1336 oi1356 siiv1403 mgiik2796sji 1400 raster position 0 20160129 0549 j, January 2016. `http://www.lmsal.com/solarsoft//irisa/data/level2/2016/01/29/20160129_054924_3664251603/www/raster/iris_rast_CII1336_OI1356_SiIV1403_MgIIk2796SJI_1400_Raster_Position_0_20160129_0549_j.html`.

[7] Cuda c programming guide. website, September 2015. `http://docs.nvidia.com/cuda/cuda-c-programming-guide/`.

[8] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM. `http://doi.acm.org/10.1145/2647868.2654889`.