Solar Flare Forecasting: A Novel Deep Learning Approach

Justin Cai

July 31, 2018

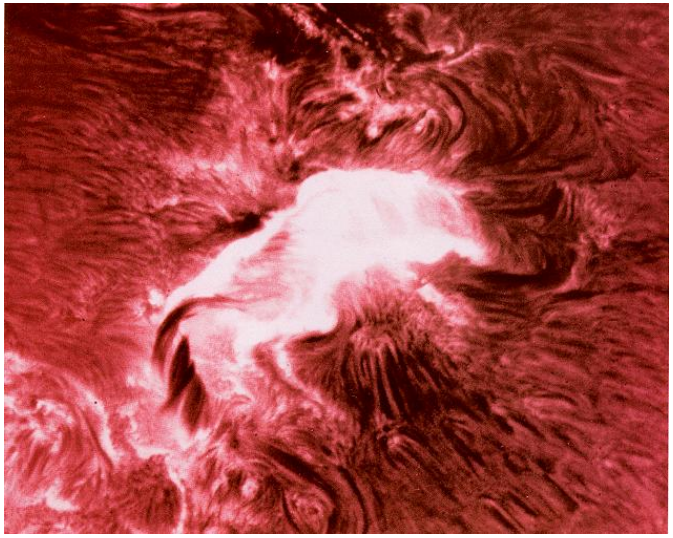
About me

- Junior at CU Boulder, studying computer science
- At LASP, I'm a student working in the Data Systems software engineering group
- First got interested in machine learning in a project using social media to predict flu vaccination rates
- Outside of ML, I'm interested in software engineering, programming languages, and statistics

Laboratory for Atmospheric and Space Physics

Why predict solar flares?

- Solar flares are events that occur near the surface of the sun
- The energy/radiation they release can be harmful to astronauts in space or electronics and other hardware components of satellites and aircraft
- A prediction could inform astronauts to take shelter in a place where the radiation would not affect them



Why use deep learning?

- Previous flare prediction models were being research and developed before/or during the revolution of deep learning, using simpler algorithms like SVM and kNN
- The application of deep learning in problems like image recognition improved results significantly
- The goal of this project is to apply current deep learning methods to the flare prediction problem and find success over the simpler models

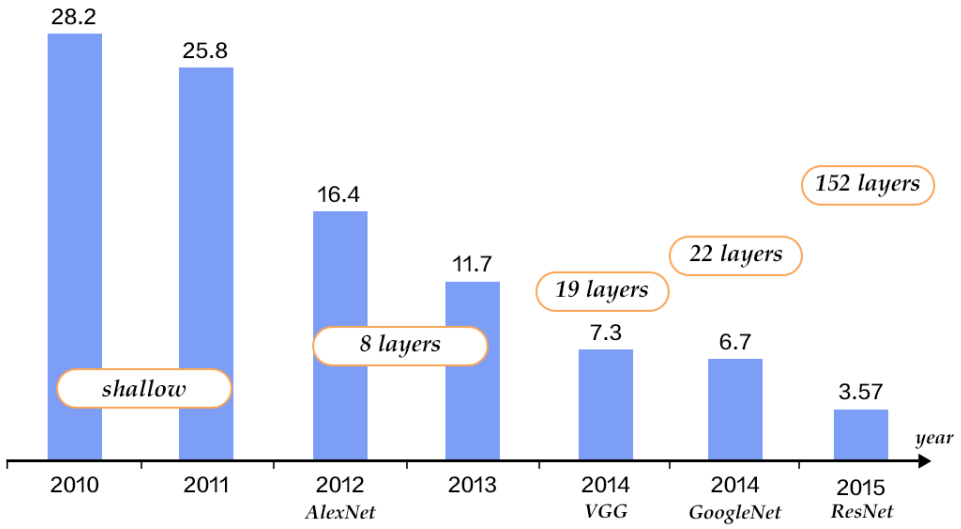


Figure: Error rate of the top models from ImageNet Large Scale Visual Recognition Competition (ILSVRC) with the depth of models. Deep networks have consistently won the competition every year, yielding lower and lower error rates.

Previous work on flare prediction

- Bobra & Couvidat achieved a true skill statistic (TSS) of 0.761 using features calculated from vector magnetic field data and a support vector machine (SVM) with a RBF kernel
- Nishizuka et al. tested three different algorithms, SVM, k nearest neighbors (kNN), and extremely randomized trees (ERT). They used similar features to Bobra & Couvidat, although they detected active regions with their own algorithm. Additionally, they added UV brightening features, time derivatives of features, and flare history features. They achieved a TSS of 0.912 using kNN

Methods - data

- To predict flares, we'll need magnetic field data
  - Michelson Doppler Imager (MDI) - for magnetic field, it can only records line of sight, but has been recording for longer
  - Helioseismic and Magnetic Imager (HMI) - along with line of sight, can record full vector data
- HMI released a derivative product that automatically tracks active regions, called Spaceweather HMI Active Region Patch (SHARP/HARP)
- Each SHARP is sequence of observations of an active region, giving various spaceweather quantities calculated from the vector magnetogram

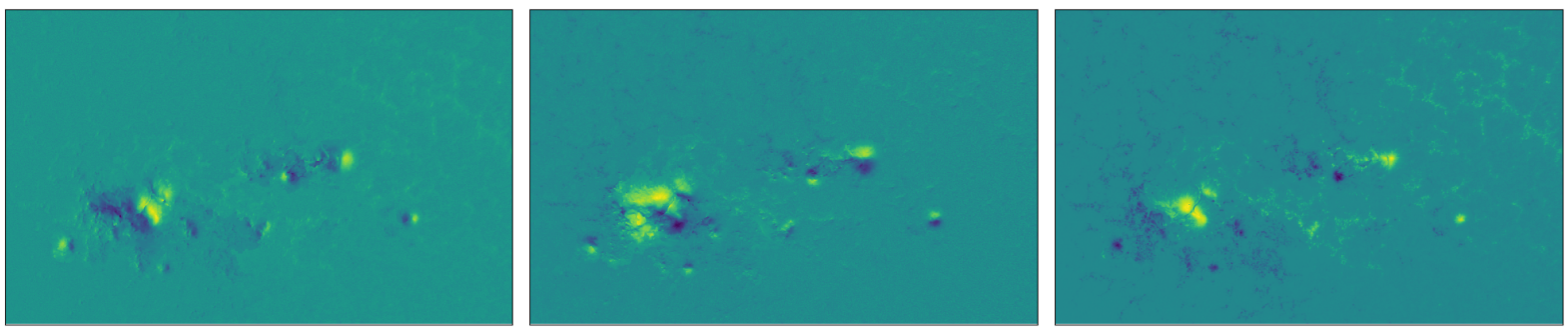


Figure: Example active region snapshot. These three images show the westward, southward, and radial components of the vector magnetogram.

Methods - data

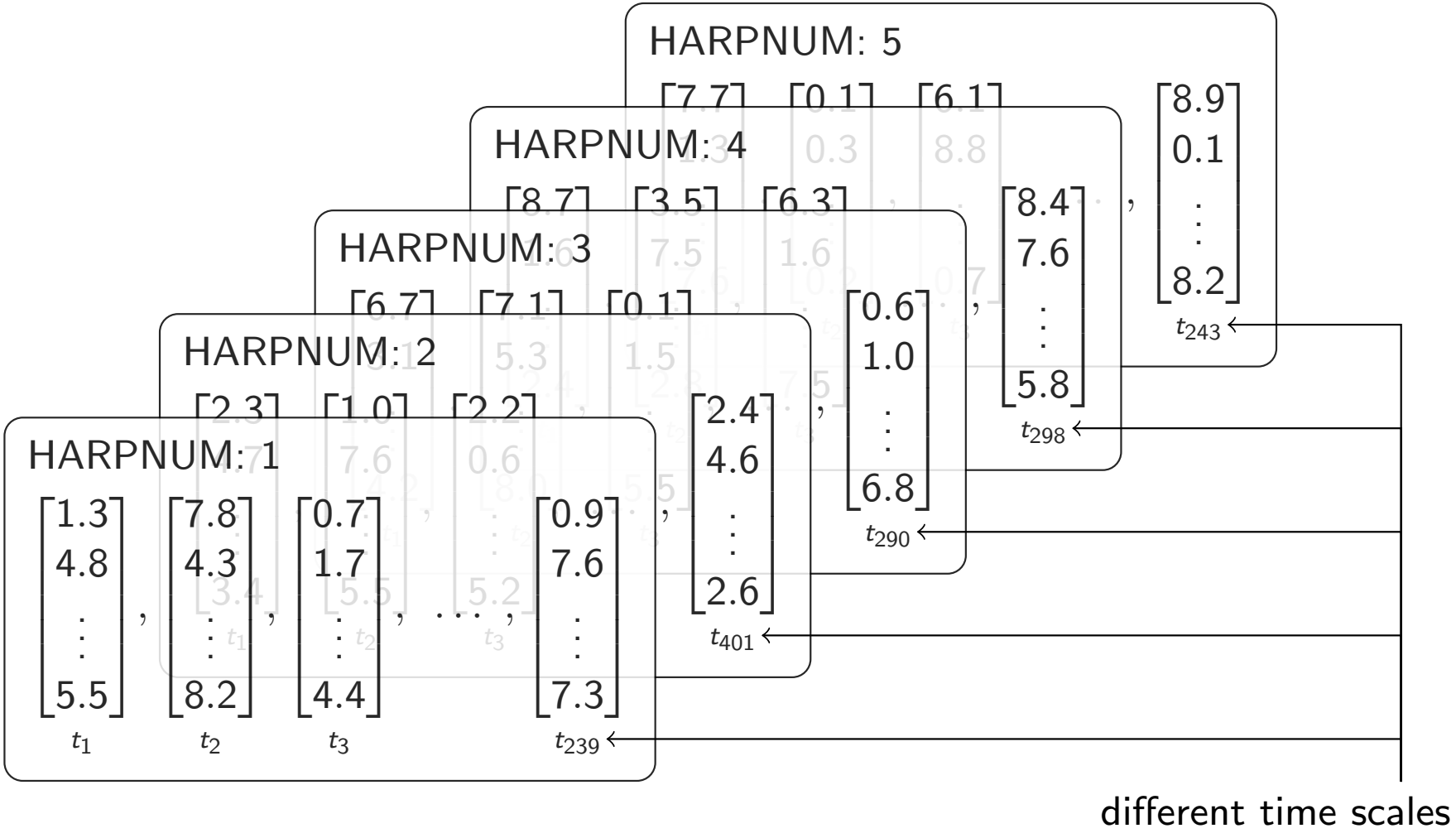


Figure: Representation of collection of SHARP sequences.

Methods - data

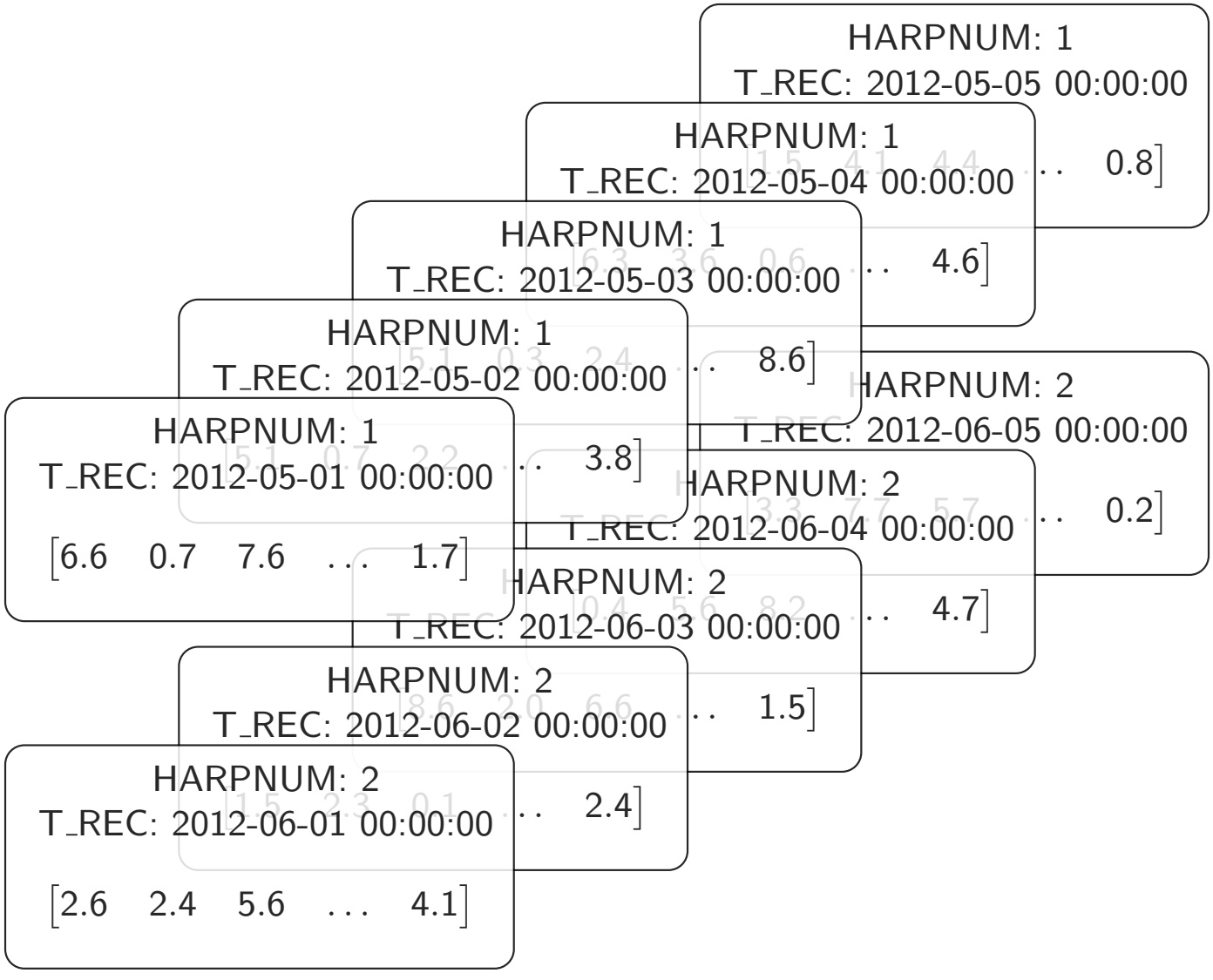


Figure: Representation of collection of SHARP observations.

Methods - data

- Geostationary Operational Environmental Satellite (GOES) measures X-ray flux, the metric which classifies flares
- They provide a catalog of flare events, giving information about the event, like time, class, and (if there was) what NOAA AR produced the event
- Most HARPs are associated with a NOAA AR, so the two data sets can be joined on that fact
- Knowing this, we can assign a label to each SHARP observation indicating if a $\{X, M, X \text{ or } M\}$ class flare happened in $\{6, 12, 24, 48\}$ hours

Methods - MLP

- Multilayer perceptrons (MLP) are the most basic and fundamental deep learning model
- Composed of layers, each of which are functions parameterized by an affine transformation and a nonlinear activation function
- In flare prediction, this model can be applied just like any of the simpler machine learning models. The model would SHARP observations and produce an output of whether or not a flare will occur
- Without augmenting the original data, the model has "tunnel vision", i.e. no information about past events

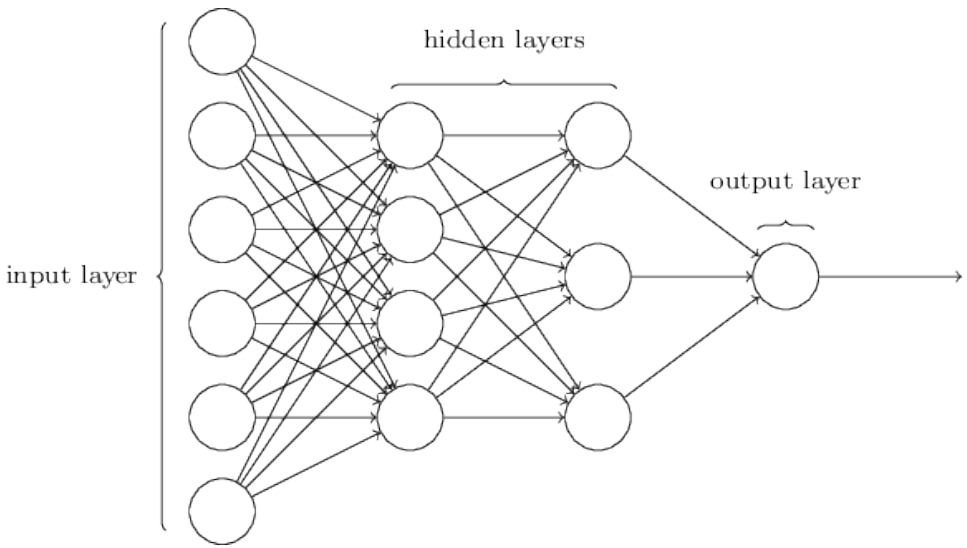


Figure: Diagram of a MLP.

Methods - LSTM

- Long short term memory (LSTM) networks are a special type of recurrent neural network (RNN)
- RNNs process sequences; at each time step they produce an output and a state, the state and/or output along with the next time step are fed back into the model and the process repeats
- The state allows for the network to remember things when processing each step
- LSTMs extend this with a concept of learnable gates, which limit how much a certain values can flow through the model.
- The gates allow LSTMs to forget their state when it becomes irrelevant

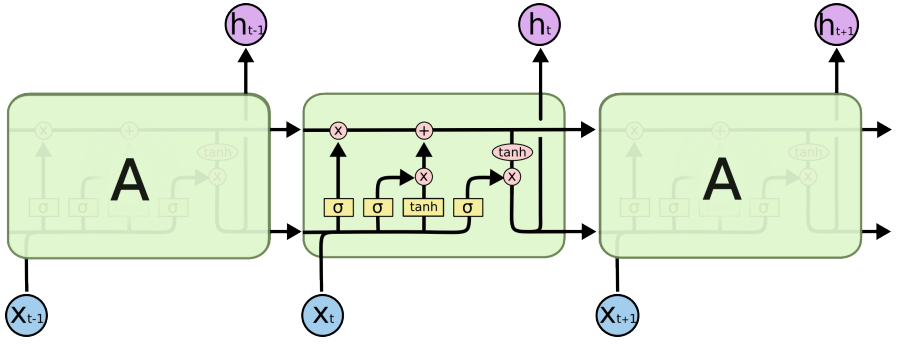


Figure: Diagram of a LSTM cell.

Methods - LSTM

- LSTM are a natural fit for the problem, as our data is a collection of sequences
- History of flares in a region has shown to be a strong predictor in previous models, which LSTMs could learn
- Looking at each time step independently is removing things like how quantities are changing over time, which LSTMs also could learn

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

Figure: Equations for a LSTM cell. f_t, i_t, and o_t are the forget, input, and output gates. c_t is the cell state, x_t is the input, and h_t is the output. The subscript t refers to the input/output at that time step. W, U and b are weight matrices that are learned during training.

Methods - metrics

- According to Bobra & Couvidat, they believe that true skill statistic (TSS) should be used when comparing the performances
- Difference between recall and false alarm rate
- Compared to other metrics, TSS is independent of the class-imbalance ratio (N/P)

$$TSS = \frac{TP}{TP + FN} - \frac{FP}{FP + TN}$$

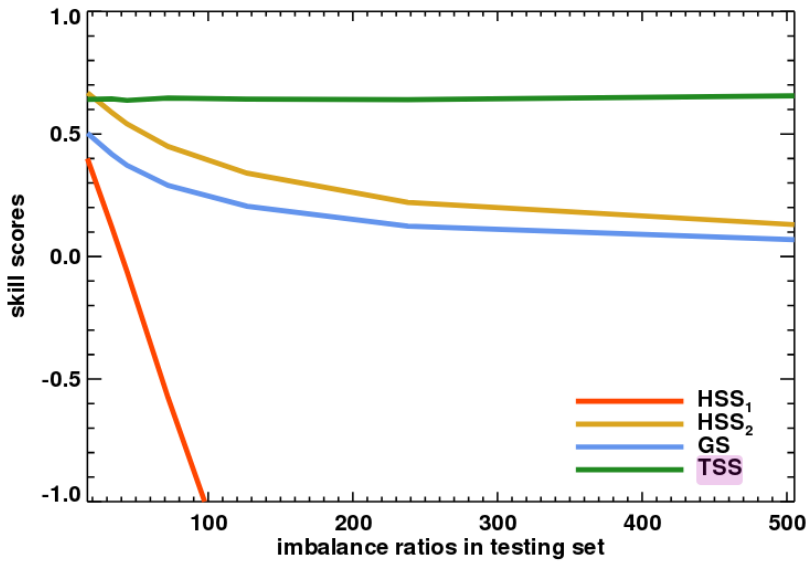


Figure: Dependence of class-imbalance ratio in the test set and scores of a SVM classifier from Bobra & Couvidat (2015).

Implementation

- Data contains SHARP observations at roughly an hourly cadence
- Preprocessed by taking the natural log of features with big scales ($max(\text{feature}) > 10^8$) and then standardizing each feature.
- All of the models were implemented in Keras - a high level deep learning framework
- Training models on a machine with two Titan V GPUs

First training attempt

- The first attempts at training models produced low TSS scores
- The problem was the imbalance of positive events (a flare happened within 24h) and negative events (a flare did not happen within 24h)
- Since there were so few positive events (2% of all observations are positive events), any predictor can achieve a low loss by predicting everything as a negative event.
- Two ways to fix this problem traditionally: weighted loss function and resampling (through undersampling and oversampling)

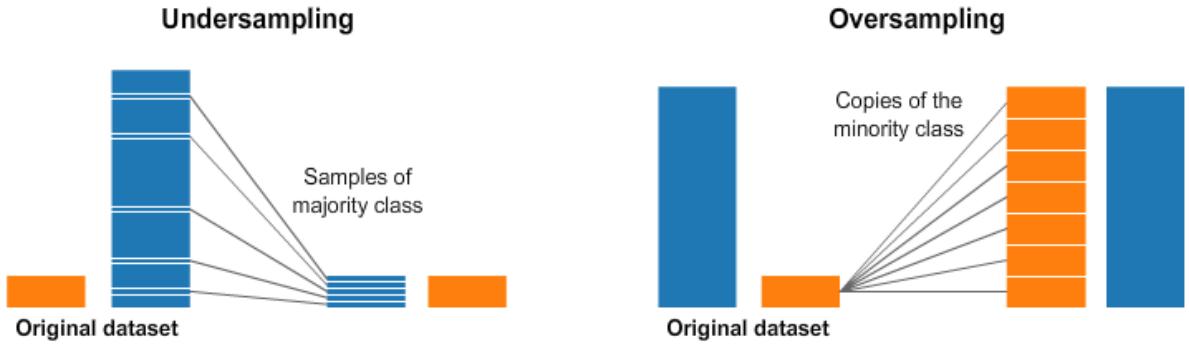


Figure: A visualization of undersampling and oversampling.

Resampling in flare prediction

- Undersampling and oversampling can be applied to SHARP observations, simply remove or duplicate individual observations
- However, applying these techniques to SHARP sequences is trickier
- Duplicating individual observations and inserting them into sequences would ruin the structure of the sequence
- Removing sequences that contain only negative events can help balance the classes, but not perfectly balance
- You could duplicate sequences that have positive events, but depending on the ratio of positive events to negative events within the sequence, duplication could lead to a bigger imbalance

Results

- Three models trained:
  - MLP trained on vector time steps with weighted loss (WL) - MLP w/ WL
  - MLP trained on over and undersampled vector time steps - MLP w/ RS
  - LSTM trained on vector sequences with WL - LSTM w/ WL
- For reference - Bobra & Couvidat achieved 0.76 and Nishizuka et al. achieved 0.91

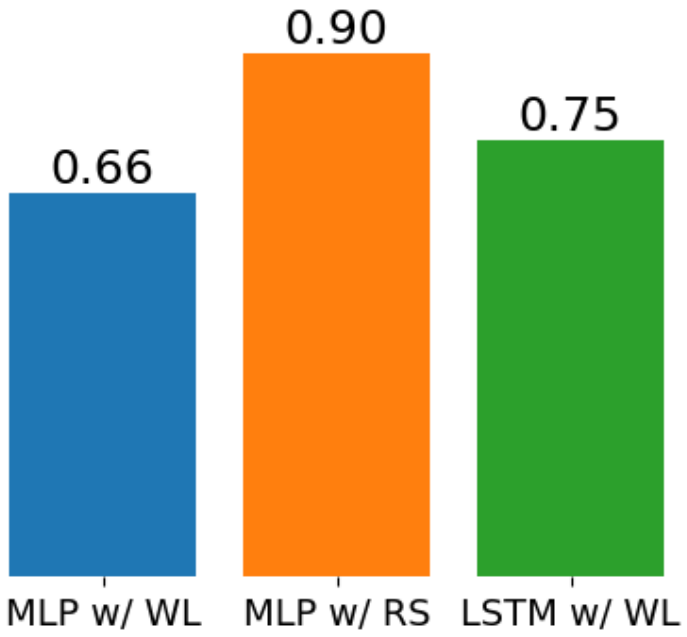


Figure: TSS of trained models.

Results

- Oversampling and undersampling seem to be highly effective
- The model with tunnel vision (had no information from the past) had the highest TSS
- Possible hypotheses
- LSTMs are too complex - need a simpler way of incorporating past information (see future experiments)
- Different activation functions - I trained the LSTMs with tanhs, while the MLPs with ReLUs. (changing to ReLU would bring the LSTM closer to the MLP model)
- Past information is not as useful as we think/SHARP parameters are stronger predictors than we thought
- Bad hyperparameters

Future experiments

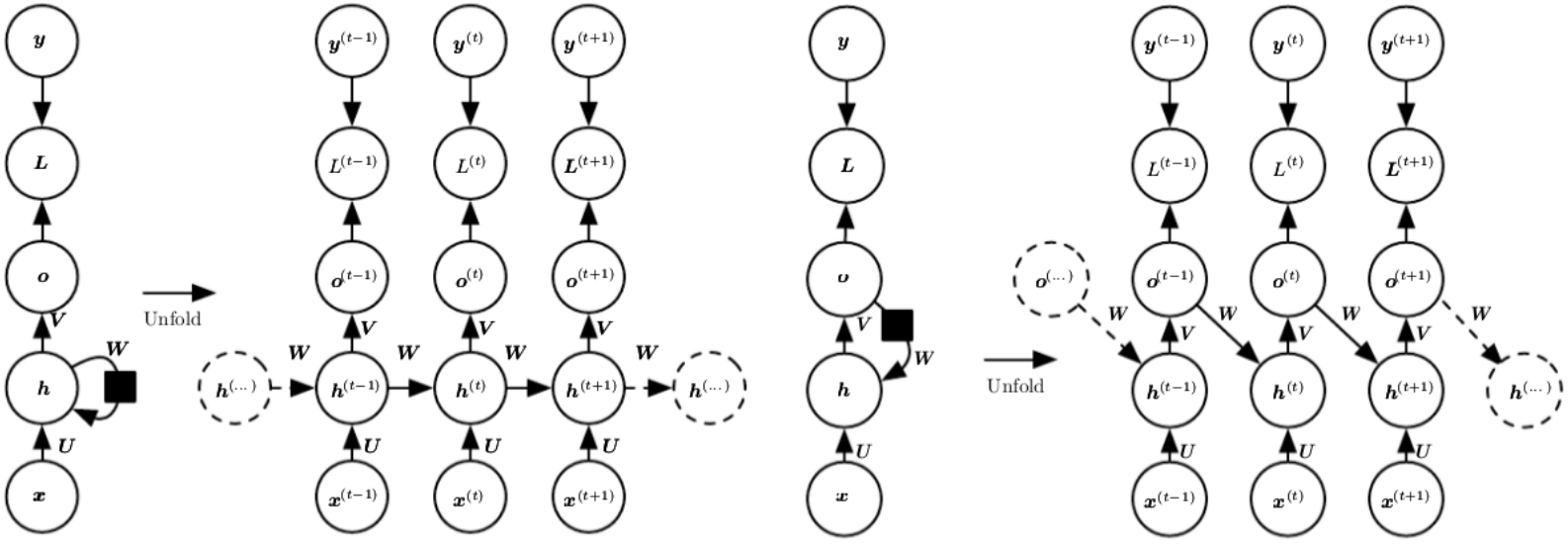
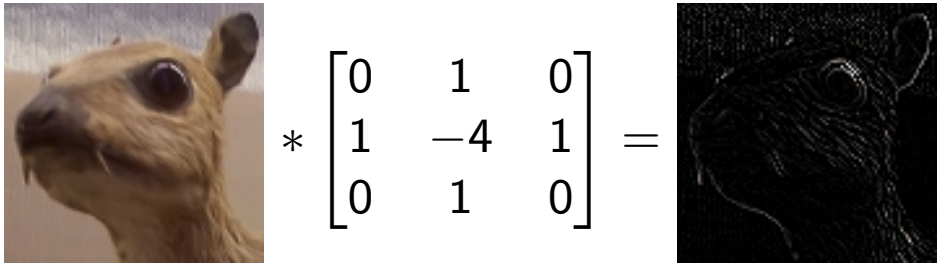


Figure: Regular RNN (left) vs teacher forcing RNN (right). The recurrent connection in the teach forcing RNN comes from the output, instead of hidden layer.

Future experiments



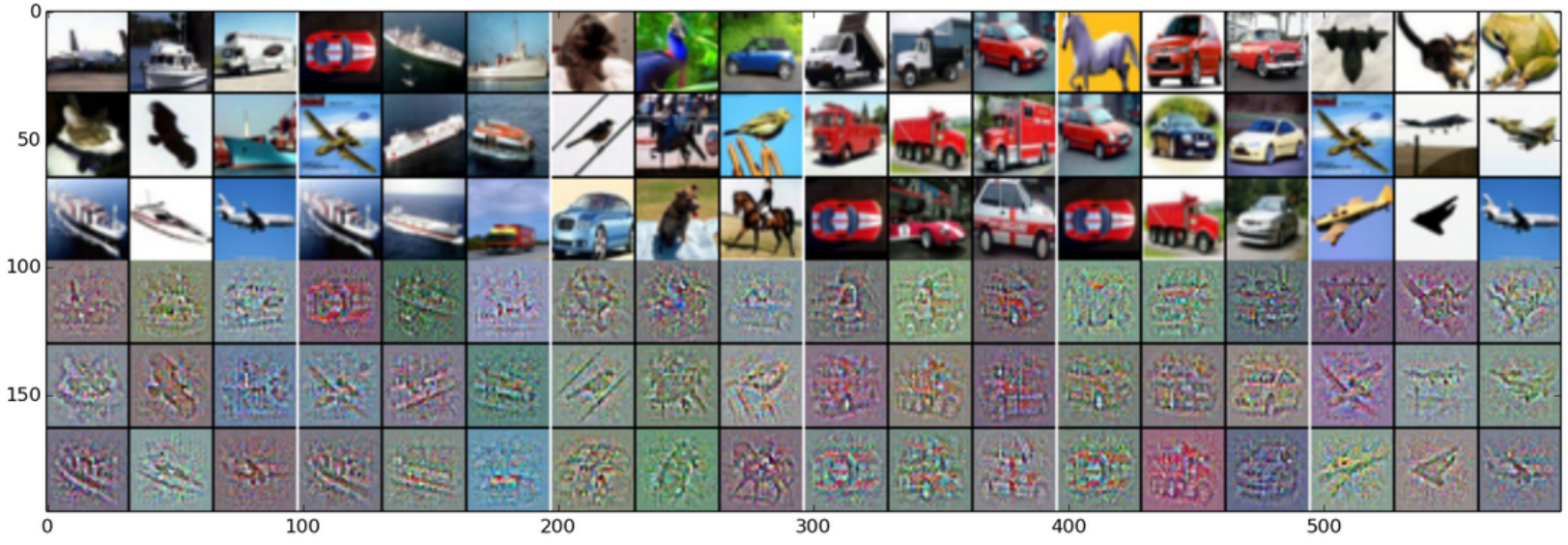


Figure: (Top) Example of convolution operation that detects edges. (Bottom) Example kernels from an image recognition CNN and which images produced the strongest activation.

Future experiments

- CNN + time information - with a CNN, we can still apply all other forms of incorporating past information like a LSTM, teacher forcing RNN, and a window of images. These models would be very complex in the number of features but could possibly be the most powerful models
- Add more features from feature engineering to MLP model
- Turn into a regression problem - take a full disk magnetogram and estimate X ray flux in the next 24h

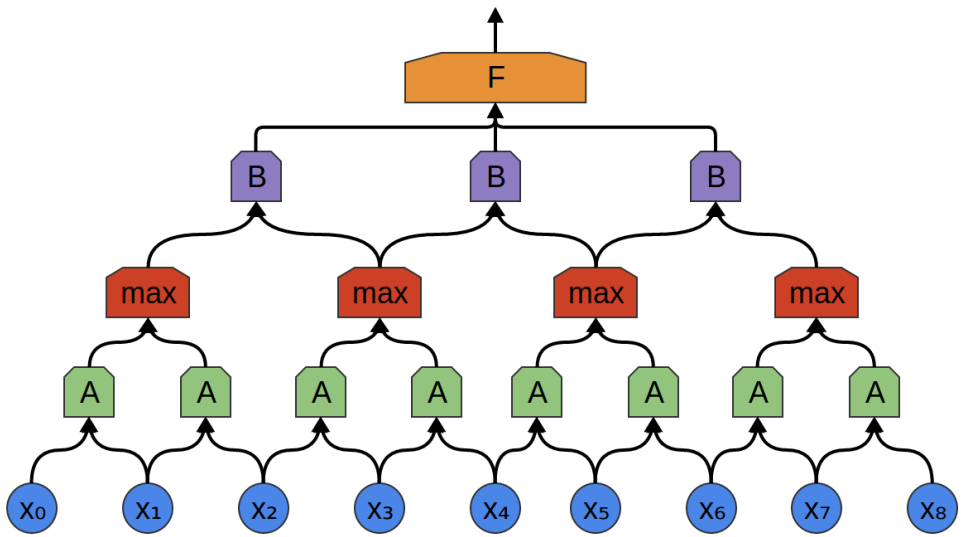


Figure: Example of a 1 dimensional CNN. Each x would be the value of some spaceweather quantity at some time.

Acknowledgements

I would like to thank Wendy Carande, Jim Craft, Kim Kokkonen, Laura Sandoval, Tracey Morland, and Andrew Jones for listening, supporting, and allowing me to work on this project.

Q&A

email: justin.cai@lasp.colorado.edu