Probabilistic Dust Storm Prediction

Context-Senstive Impact Discovery

Sean Flaherty

2017

Mentors: Dr. Huiping Cao, Dr. Colby Brungard, Dr. David DuBois
Acknowledgements: Josue Gutierrez, Merrill Bean, Max Bleiweiss

Motivation

- Dust storms are a hazard to public health and economic development where they occur.
- Predictive models paired with communication systems allow for an early warning system to prepare for and reduce potential damage [1].

Overview

- Goal: Use data mining/machine learning to create a predictive model for dust events on a local scale.
- Previous research: Univariate predictor (500mB geopotential height) using image processing (ZNCC). Only good at predicting large events [2].
- Justification: Want an accurate predictor for meso- and micro-scale dust events.

Gathering Data

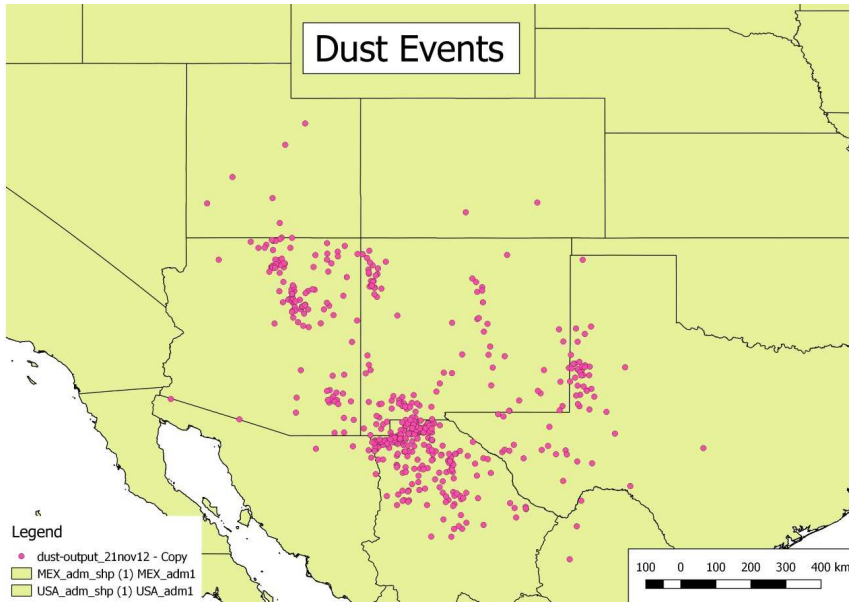
- RAP/RUC forecast model – NOMADS (NOAA Operational Model Archive and Distribution System).
- Forecasts available from NOAA online repository (HTTPS/FTP).
- Data downloaded using GNU wget utility [3].

Data Format

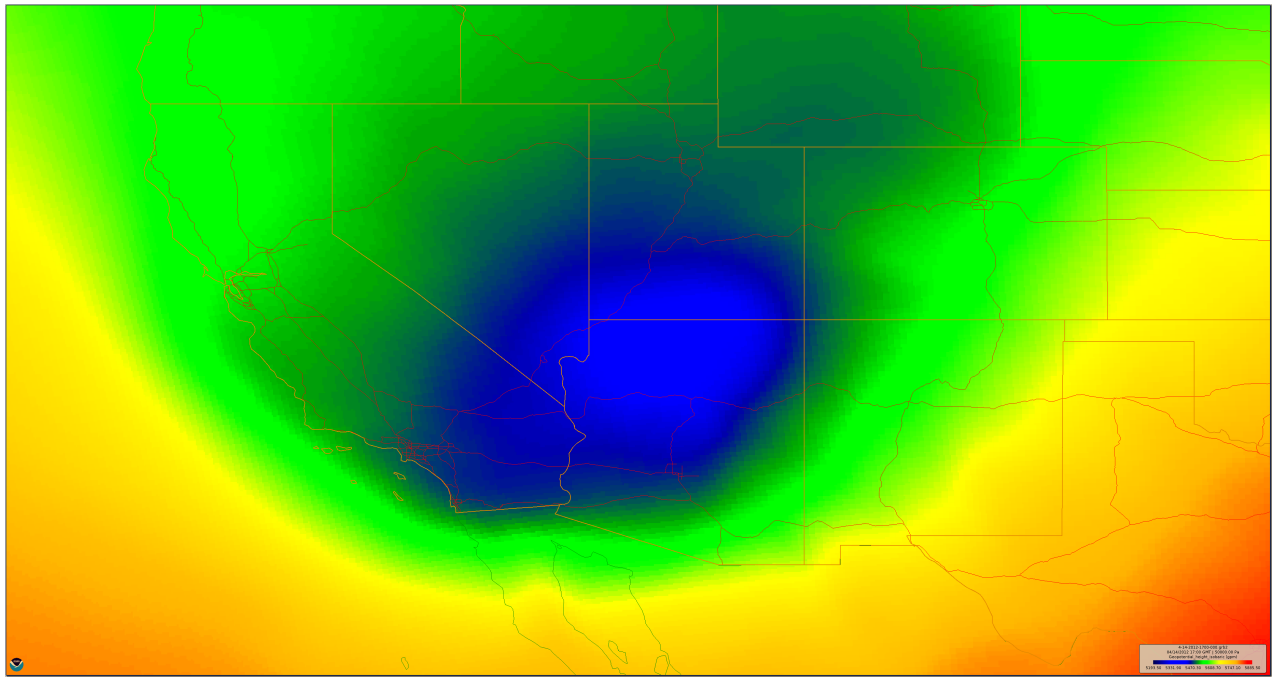
- GRIB – GRIdded Binary (WMO standard for weather data).
- RAP/RUC models update forecasts hourly with 13 and 25.2 km resolutions. Each file contains forecast models for a single time.
- Each file has a number of weather parameters, each of which with a grid of data points corresponding to locations.

Generating Training Data

- Script iterates through list of dates - 4/13/2007 to 4/30/2012 - all .grb2 format (RUC13)
- If dust event on that date, use event's location.
- If not, randomly generate location using normal distribution and lat/lon mean and standard deviation.



GRIB Example



A single parameter's raster shown using NOAA Weather and Climate Tool. This image shows the 500mB geopotential height at 18:00GMT preceding a dust event on April 14, 2012 [4].

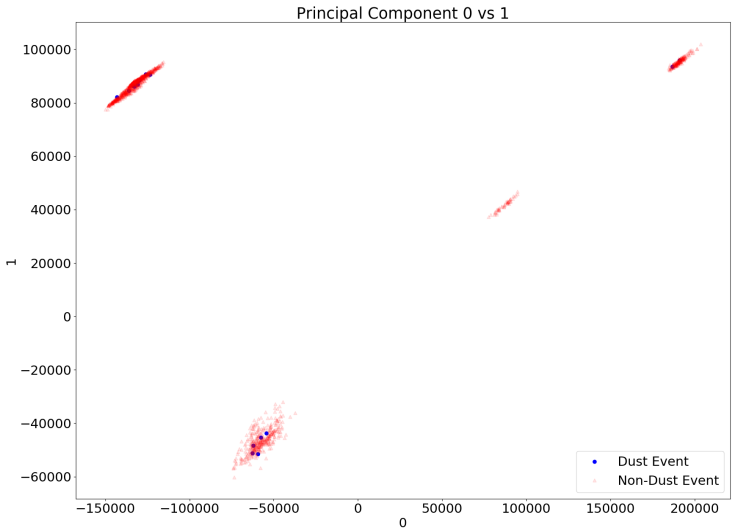
Reading GRIB files

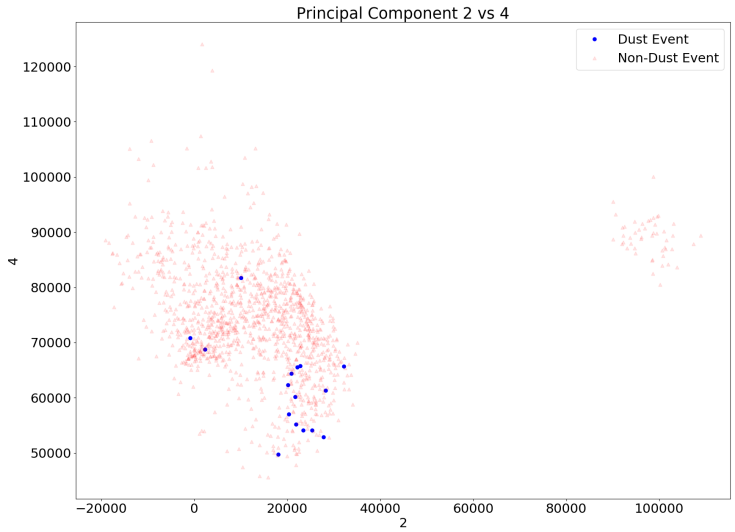
- pygrib Python library – allows opening of .grb, .grb2 files in Python
- Opening a GRIB creates a file iterator, with each object in it a weather parameter [5].
- Each weather parameter has various attributes, including a raster of latitudes/longitudes and data for the parameter.
- Weather data gets stored into CSV files for easier lookup.

PCA

- Principal component analysis reduces the dimensionality of the data.
- Transforms data into subspaces with the most spread between points - explains most of the variance in the data.
- RUC dataset has 315 dimensions for each instance – could make algorithms less effective (curse of dimensionality).
- Normalize principal components for use in NN.

PCA plots



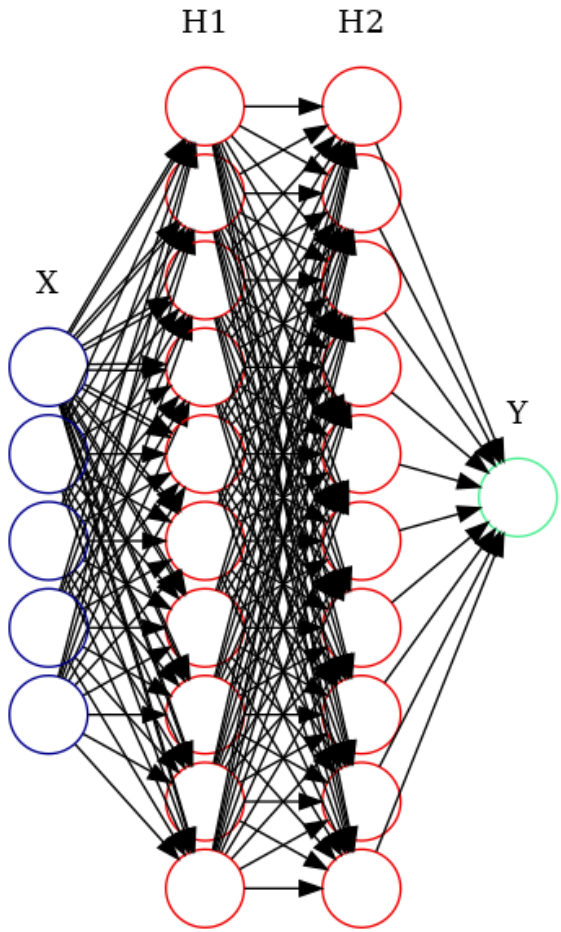


Plots of principal components against each other. Left shows little no distinction between dust and non-dust events, while right shows some.

Algorithms

- Feedforward NN with PCA
- RNN/LSTM with PCA

Try each algorithm and see which one provides the best accuracy.

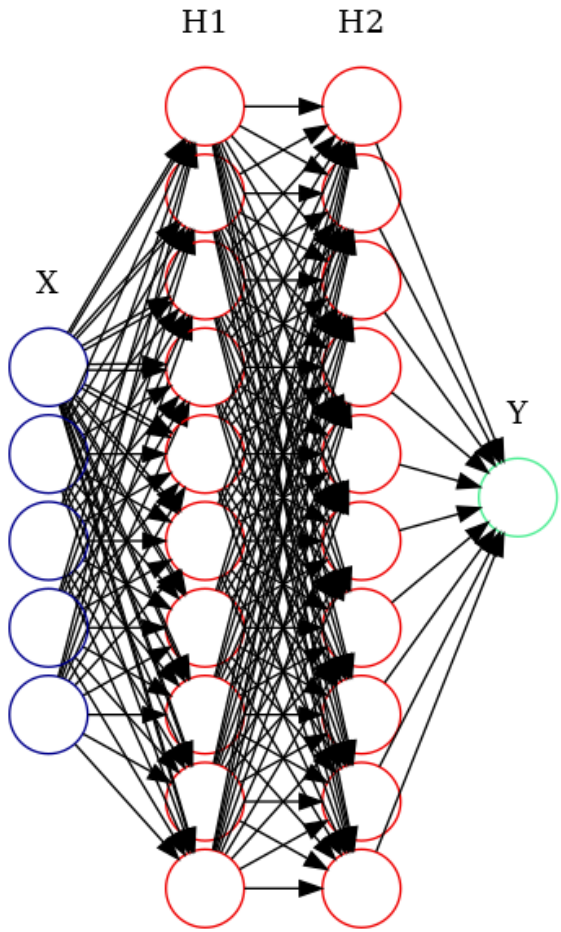


Implementation of algorithms

- TensorFlow Python library
- Machine learning utility for creating computation graphs, doing lazy evaluations, using GPU for faster processing.
- Automates backpropagation and optimization algorithms [6].

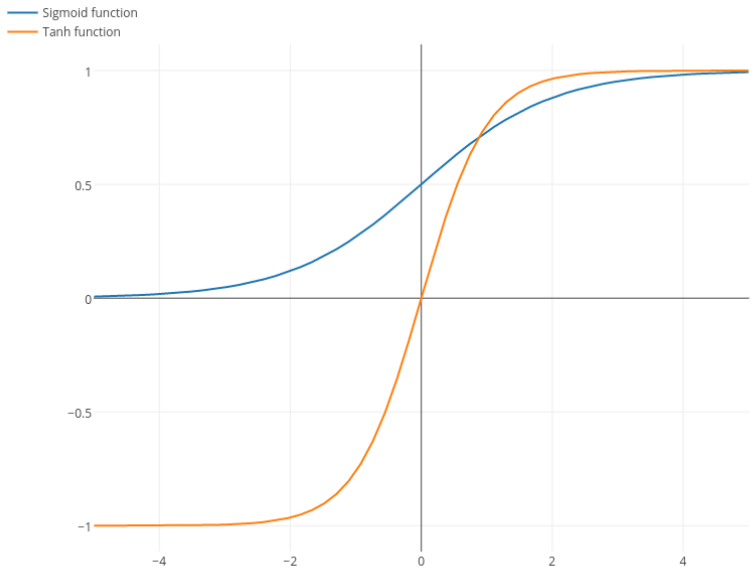
Feedforward Neural Network

- 10 principal components as inputs
- 25 nodes per hidden layer
- 2 output nodes
- Tested between 1 and 8 hidden layers



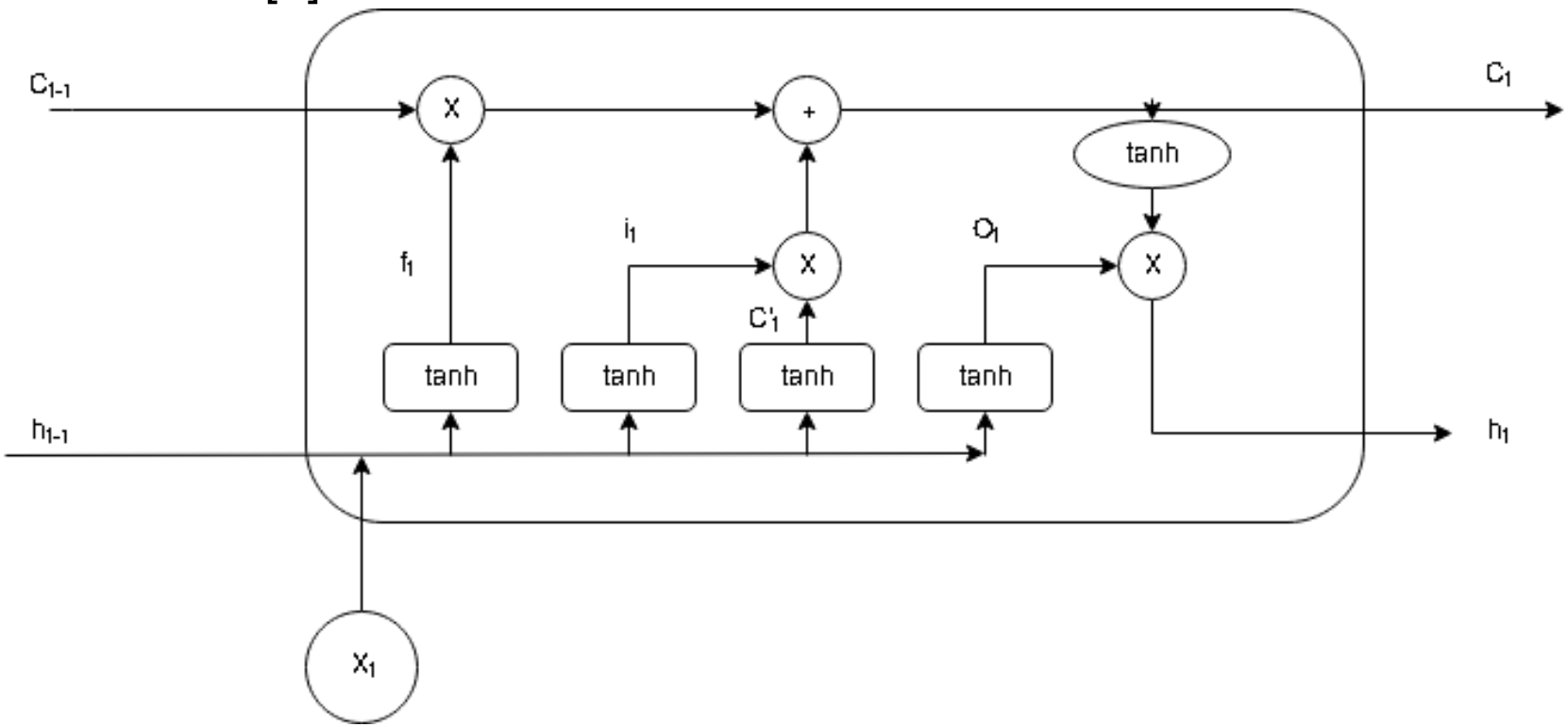
Feedforward Neural Network, cont'd

- Use tanh activation function for hidden layers - steeper optimization gradient [7]
- Sigmoid output layer for values between 0 and 1
- Makes classification based on greatest value in output layer
- Weighted cross-entropy with logits loss function - weigh output losses according to imbalance in data set - about 11% of events are dust.

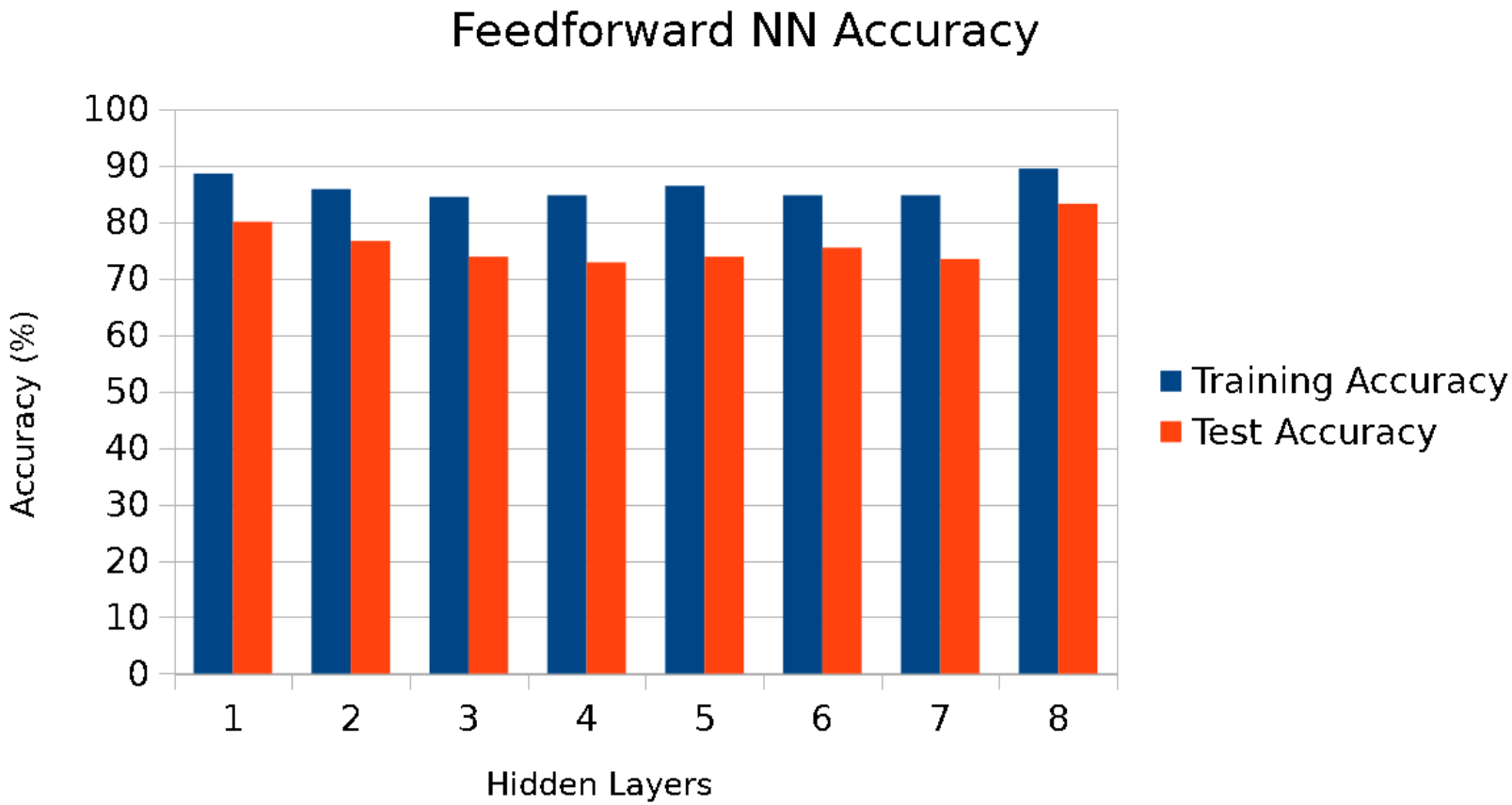


LSTM

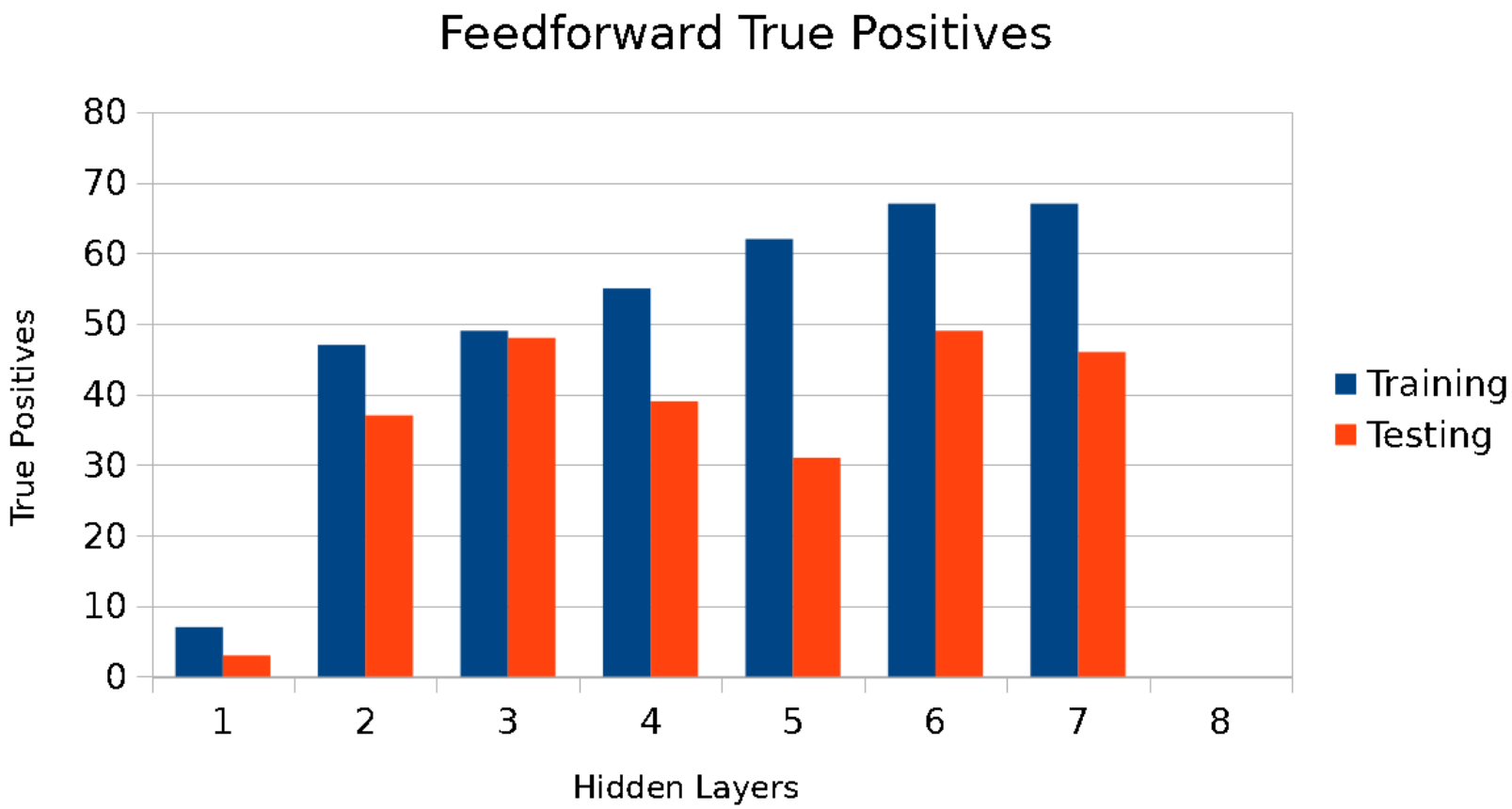
Long short-term memory (LSTM) - RNN that stores past predictions in memory to create a more context-sensitive prediction. [8]



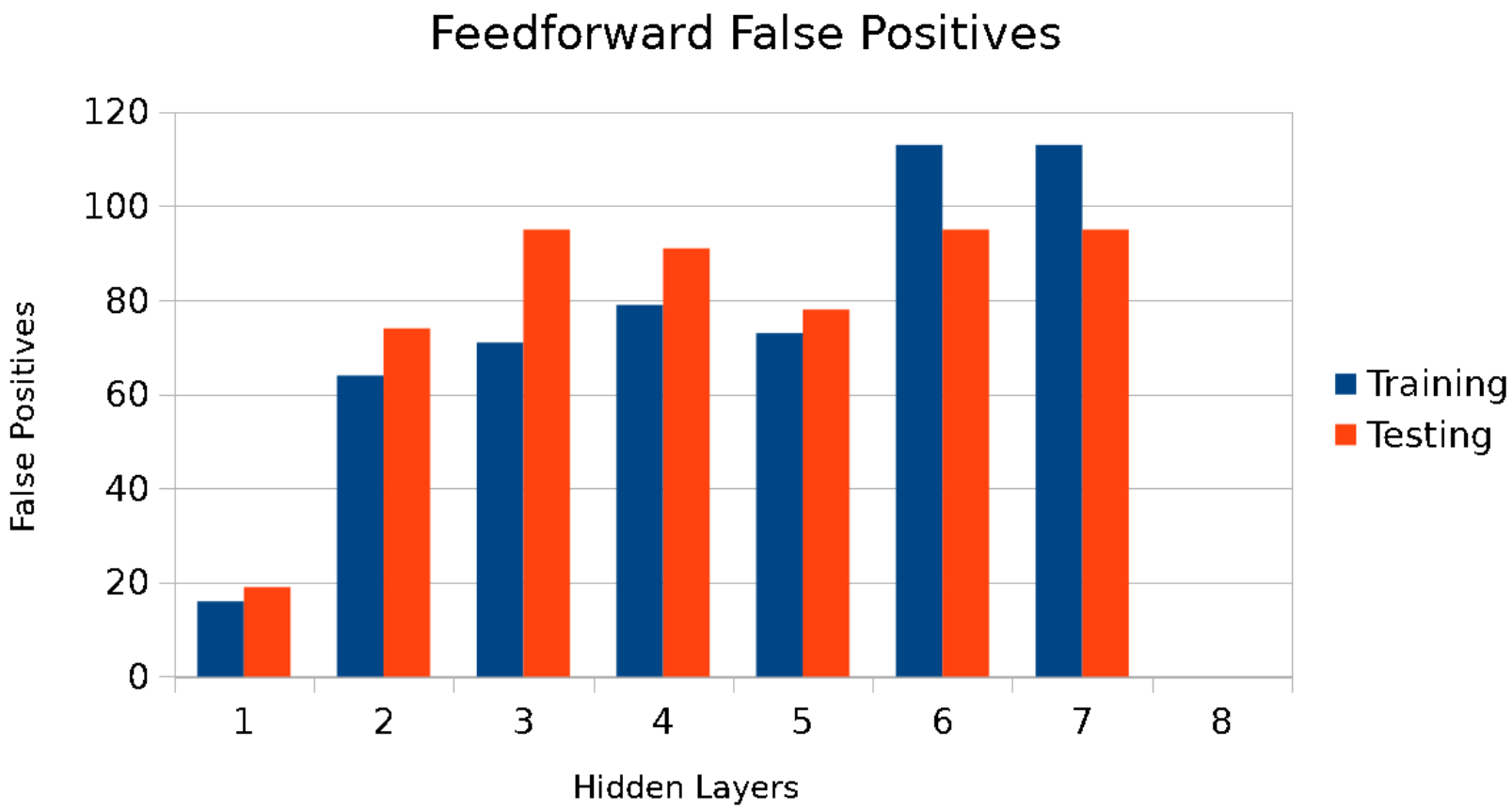
Feedforward Results, 1



Feedforward Results, 2



Feedforward Results, 3



LSTM Results

- 89.4685% training accuracy - but one problem:
- Guessed negative on every prediction!
- Could be possible that even with compensation in loss function, the imbalanced data set makes it easier for the RNN to assume that every event is non-dust.

Conclusion

- Feedforward provides good preliminary results - 80-85% training accuracy; potential for more accurate models.
- Current version of LSTM did not work as hoped - future models with a 1-hour resolution that predict 24-hour sequences may work better.
- Possibility for predictions off of raster images using RNN or CNN models.

References

[1] UNEP GEAS. "Forecasting and early warning of dust storms." UNEP. Feb. 2013. Web. May 24 2017.
[2] Armenta, Rebecca B. "Geopotential height patterns at 500mb associated with dust storms in the United States/Mexico border region during January-May of 2011-2014." May 2016 New Mexico State University. Access May 31 2017.
[3] "GNU Wget 1.18 Manual." GNU Project. Web. Jun. 27 2017.
[4] "Rapid Refresh (RAP)." National Centers for Envrionmental Information. NOAA. Web. Jun. 27. 2017.
[5] "pygrib documentation." Github. Dec. 29 2014. Web. Jun. 14 2017.
[6] "Getting Started with TensorFlow." TensorFlow. Web. Jul. 5 2017.
[7] LeCun, Y. et. al. "Efficient BackProp." Yann LeCun's Home Page. Web. Jul. 25 2017.
[8] Olah, C. "Understanding LSTM Networks." Colah's blog. Aug. 27 2015. Web. Jul. 20 2017.