An Approach to Tuning Hyperparameters in Parallel - A Performance Study Using Climate Data

CyberTraining Research Project Final Presentation

Charlie Becker[1], Will D. Mayfield[2], Sarah Y. Murphy[3], Bin Wang[4],
Research assistant: Carlos Barajas[5], Faculty mentor: Matthias K. Gobbert[5]

August 15, 2019

[1]Department of Geosciences, Boise State University
[2]Department of Mathematics, Oregon State University
[3]Department of Civil and Environmental Engineering, Washington State University
[4]Department of Mathematics, Hood College
[5]Department of Mathematics and Statistics, UMBC

Table of contents

1. Introduction

2. Taki

3. Research Problem

4. Building the Network

5. Data Augmentation

6. Parallelization

7. Results and Discussion

Introduction

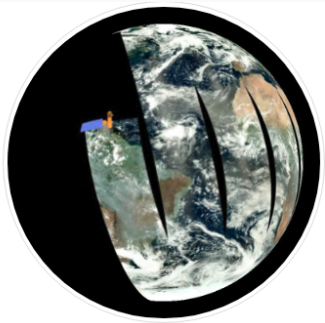
Introduction

A Main Goal: Evaluating performance of statistical learning methods with different software packages and hardware using atmospheric data.

- Atmospheric data provided by NOAA and prepared for the Machine Learning in Python for Environmental Science Problems AMS Short Course. [1]

Tools to use and evaluate:

- Tensorflow
- CPU / GPU



TensorFlow

[1] https://github.com/djgagne/ams-ml-python-course/blob/master/module_2/ML_Short_Course_Module_2_Basic.ipynb

Background

Machine learning with meteorological data

- Machine learning has been used to accurately forecast rain type, clouds, hail, and to perform quality control.[2]
- Can be useful in the place of physically based models by giving meaningful results without the time or computational cost. [3]
- Binary classification of severe weather can be difficult. [4]
  - A false alarm can result in mistrust from the public.
  - A missed alarm can result in expensive damage and loss of life.

[2] Nourani et al. 2019, Ghada et al. 2019, McGovern et al. 2017, Lakshmanan et al. 2015

[3] Nourani et al. 2019

[4] Barnes et al. 2017

Taki

Hardware on Taki

We plan to implement a larger scale application of Tensorflow in the following settings.

CPU

- HPCF2013: 49 compute, 2 develop, and 1 interactive nodes, each containing two 8-core Intel Ivy Bridge CPUs
- HPCF2018: 42 compute and 2 develop nodes, each with two 18-core Intel Skylake CPUs and 384 GB of memory

GPU

- HPCF2018: 1 GPU node containing 4 NVIDIA Tesla V100 GPUs connected by NVLink and 2 Intel Skylake CPUs

Scheduling rules on taki – CPU

How to access 2013/2018 CPU on Taki?

- **2013 batch nodes**
  #SBATCH – –partition=batch
  #SBATCH – –constraint=hpcf2013
- **2018 develop nodes – cnode[001,030]**
  #SBATCH – –account=pi_gobbert
  #SBATCH – –qos=short+
  #SBATCH – –partition=develop
  #SBATCH – –constraint=hpcf2018
- **2018 CPU nodes – 42 nodes with names cnode[002-029,031-044]**
  #SBATCH – –account=pi_gobbert
  #SBATCH – –qos=short+
  #SBATCH – –partition=high_mem

Scheduling rules on taki – GPU

How to access 2018 GPU on Taki?

#SBATCH – –gres=gpu:2 # 2 is the number of requested GPUs
#SBATCH – –constraint=hpcf2018

Research Problem

Research Question

Predict probability of storm rotation using low-level vorciticy thresholds, radar reflectivity, surface wind, and temperature.

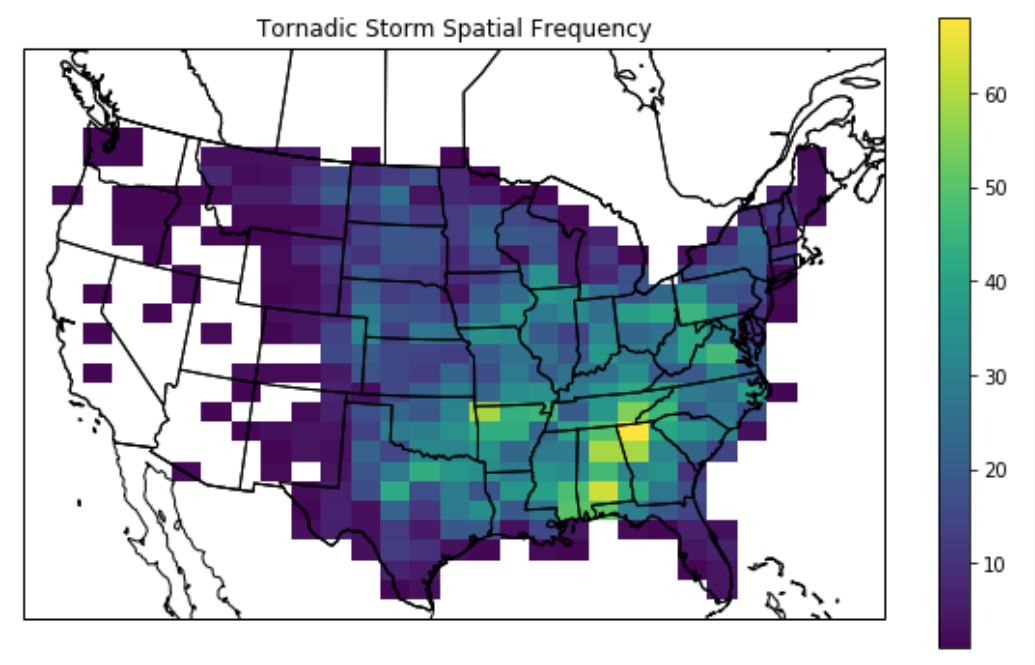


Figure 1: Tornadic storm frequency figure example from the AMS short course. A tornadic storm is defined as a storm with a maximum vorticity greater than 0.005.

Data

All data used was aquired from the Machine Learning in Python for Environmental Science Problems AMS Short Course [5]

- ˜80,000 Convective storm centroids across central U.S.
- A storm is defined as having simulated radar reflectivity greater than 40 dBZ.
- 32 x 32 x 4 "images"
- Unbalanced data set: majority/minority $\sim$ 19 : 1

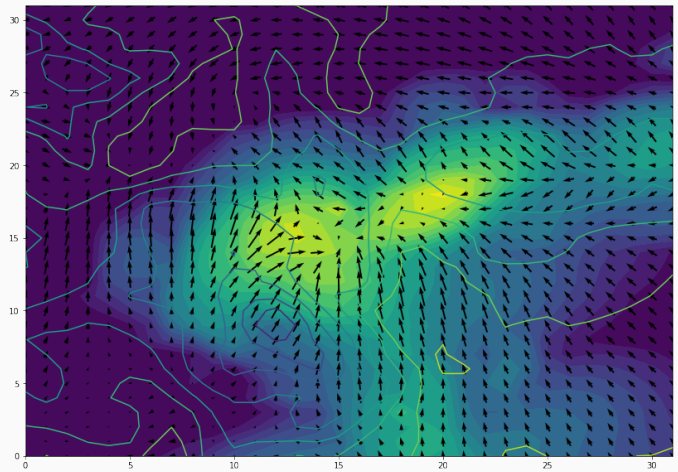


Figure 2: An example image of the radar reflectivity and wind field for a storm

[5] Data can be downloaded from: `https://storage.googleapis.com/track_data_ncar_ams_3km_csv_small/track_data_ncar_ams_3km_csv_small.tar.gz`

Building the Network

A simple 1-layer dense neural network (DNN) model

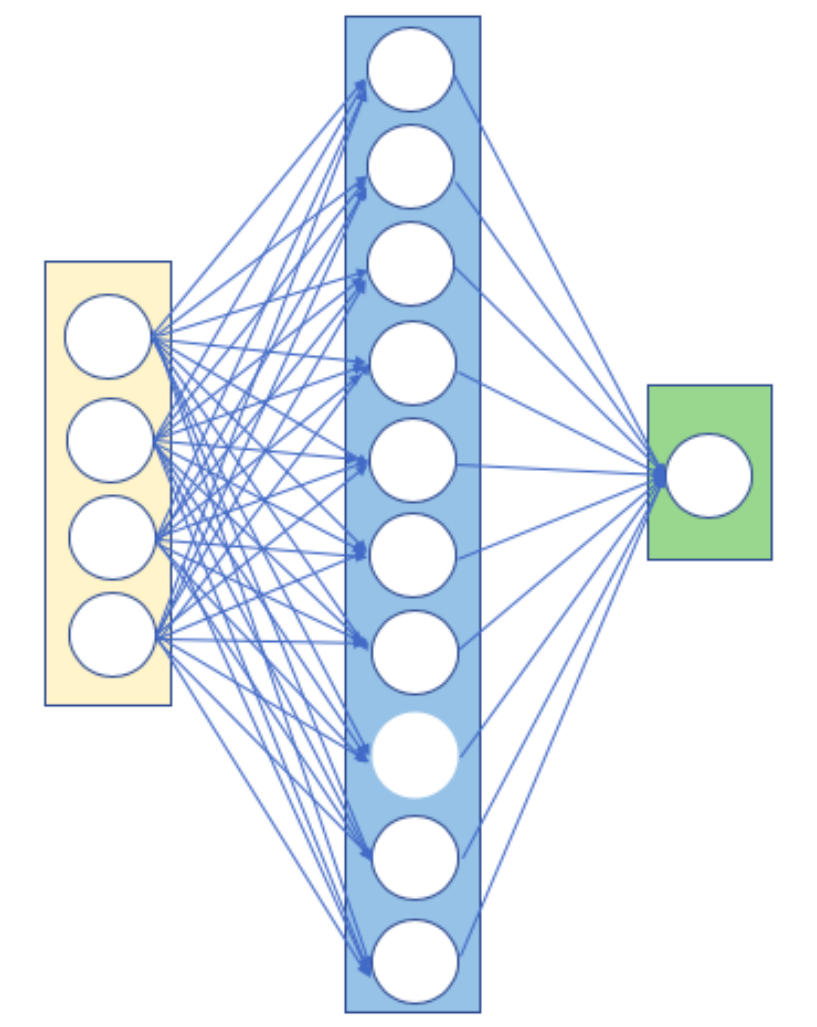


Figure 3: A diagram of the dense neural network

A simple multi-layer convolutional neural network (CNN) model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 32, 32, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 8)         608
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 8)         0
_________________________________________________________________
average_pooling2d_1 (Average (None, 16, 16, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 16)        3216
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 16)        0
_________________________________________________________________
average_pooling2d_2 (Average (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 32)          12832
_________________________________________________________________
activation_3 (Activation)    (None, 8, 8, 32)          0
_________________________________________________________________
average_pooling2d_3 (Average (None, 4, 4, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 17,169
Trainable params: 17,169
Non-trainable params: 0
```

Data Augmentation

Data Augmentation Techniques

- Neural networks trained with unbalanced dataset may be biased toward predicting the majority classes.
- Augmentation techniques to handle class imbalancy of the data set:
  - Undersample the majority clas;
  - Oversample the minority class;
  - Generate synthetic images.
- For example:

Figure 4: Images of kittens generated by rotation, flipping, and scaling. [6]

[6] https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Data Augmentation Techniques (Cont.)

For DNN model, we used imblearn.over_sampling.RandomOverSampler to over-sample the minority class(es) by picking samples at random with replacement.

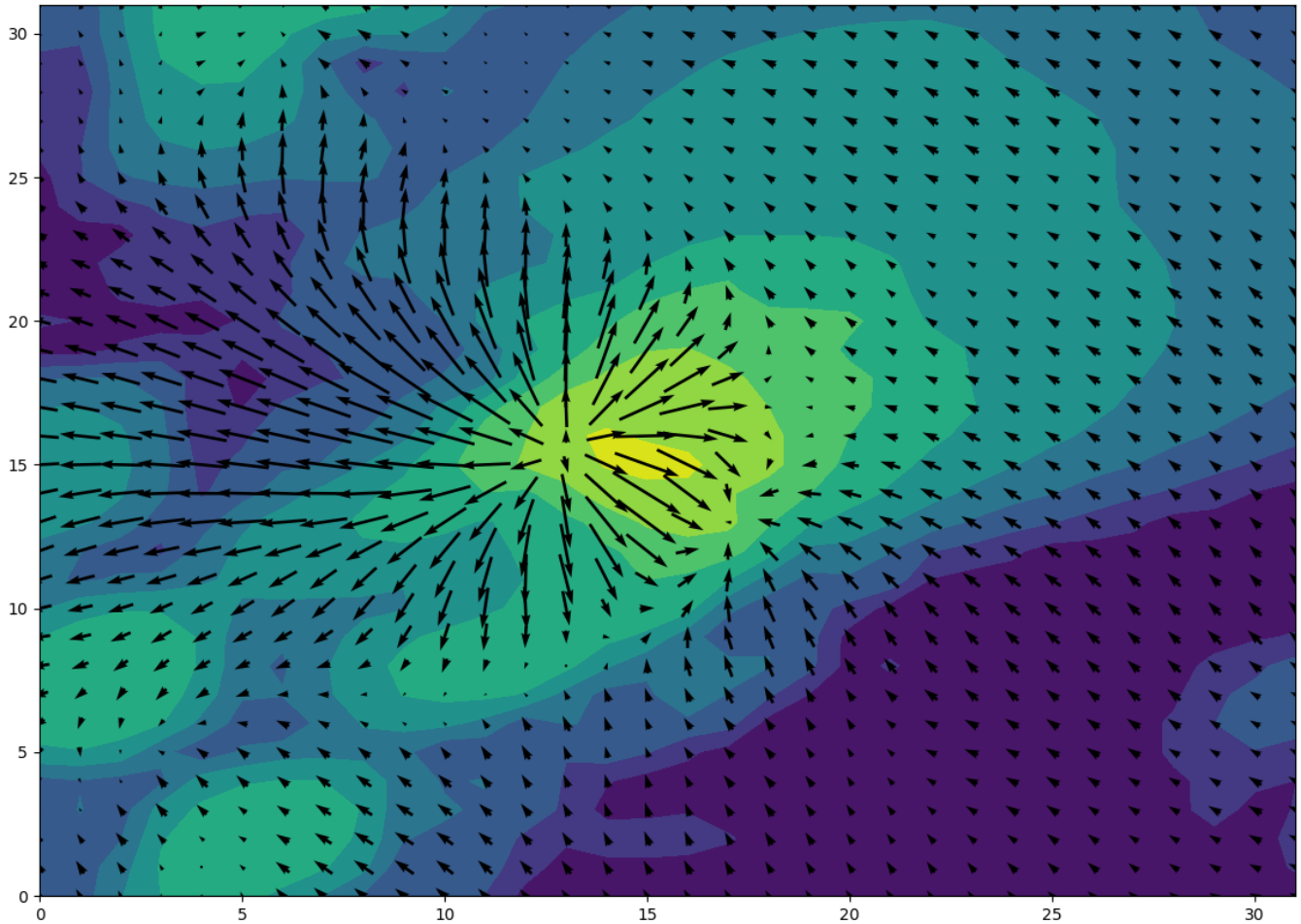
Issues: This method only takes rank 2 array as input.

For CNN model,

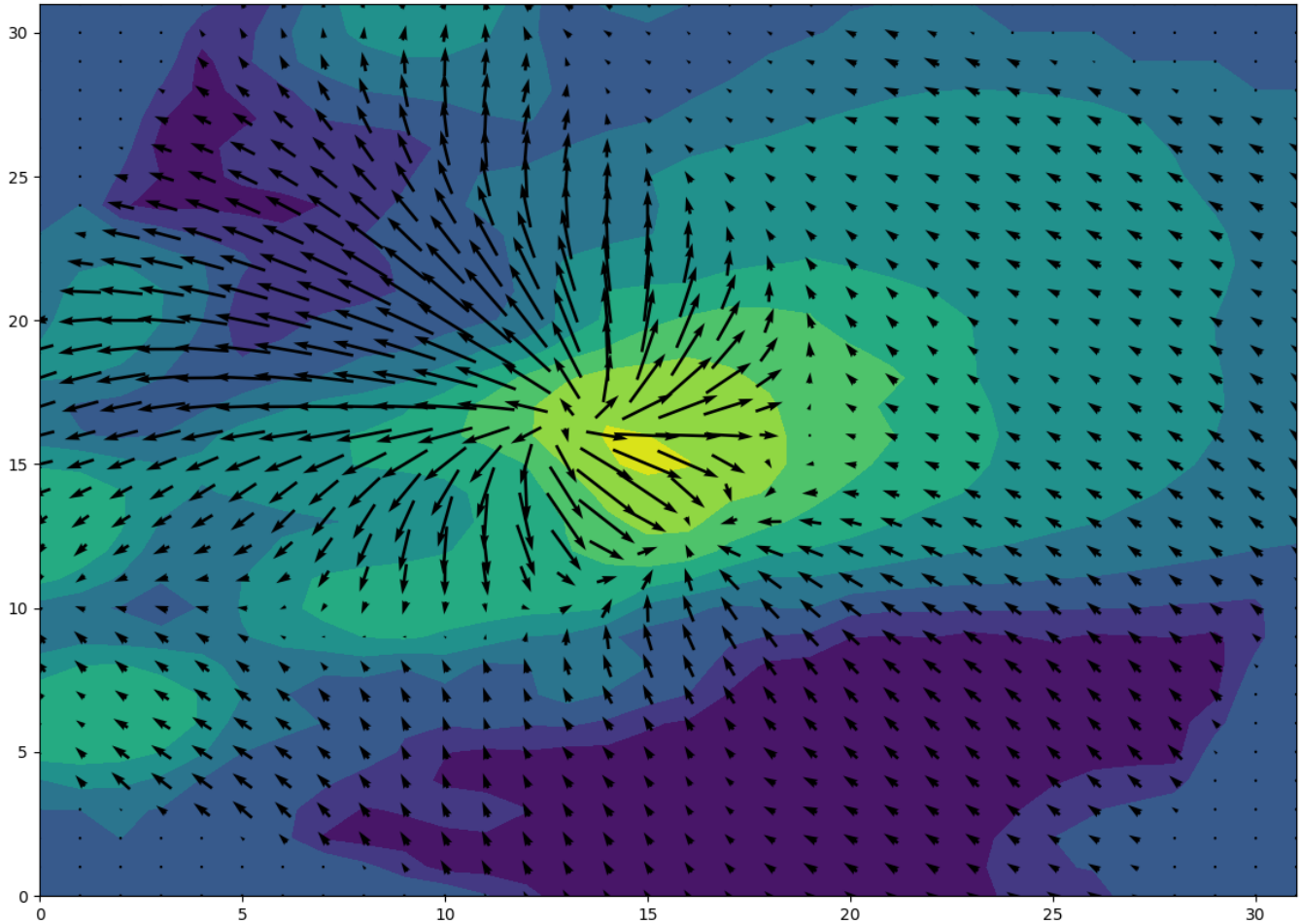
- Online augmentation – use keras.preprocessing.image.ImageDataGenerator to generate real-time tensor image data by rotation, and flip.
- Offline augmentation – use skimage.transform.rotate to generate synthetic images by rotations at the preprocessing stage

Main Observations: Data augmentation doesn't have a noticeable effect on the metrics of interests for this data set.

Data Augmentation Techniques – Sample Synthetic Image (Cont.)



Data Augmentation Techniques – Sample Synthetic Image (Cont.)



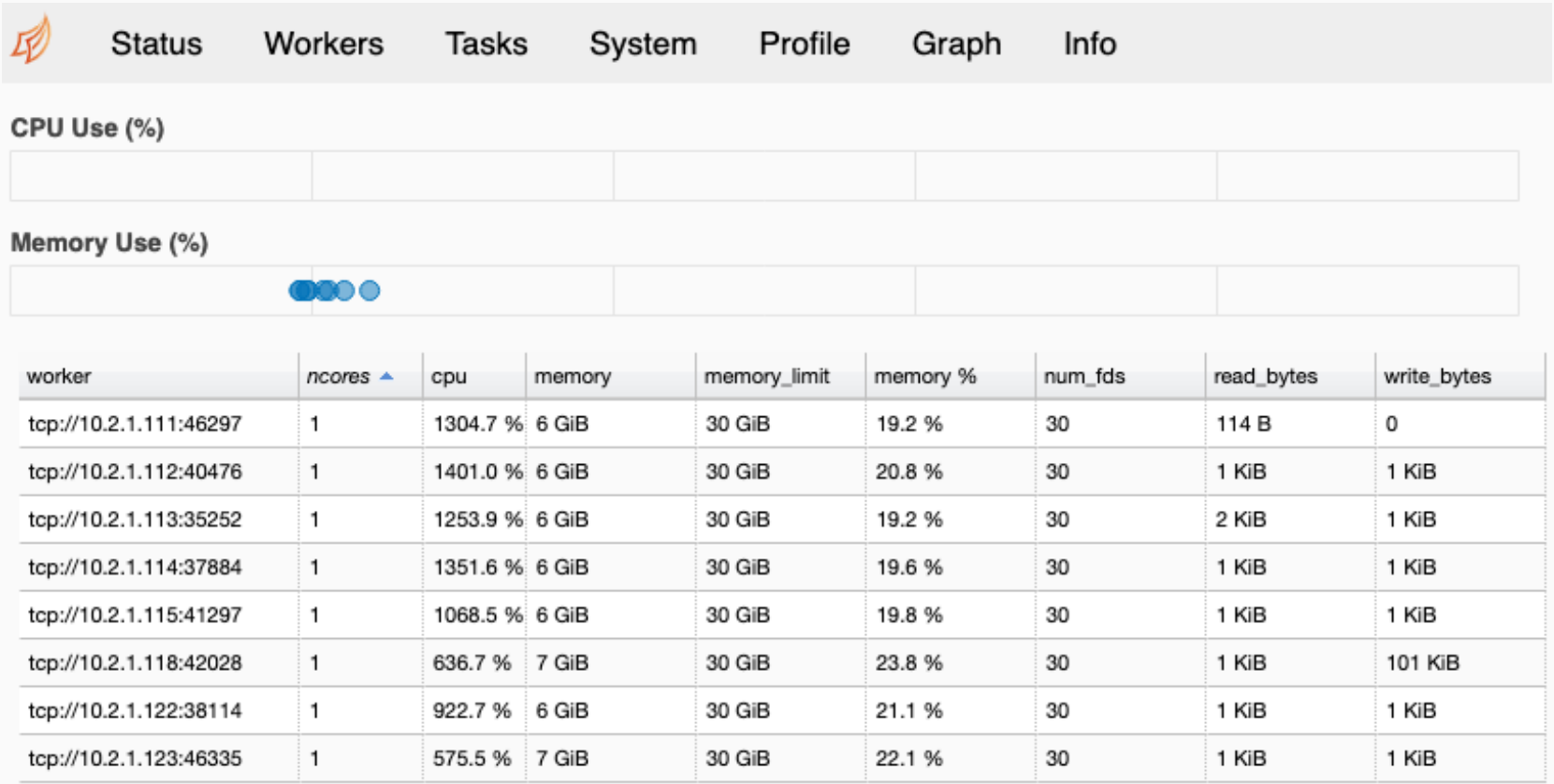
Parallelization

Parallelization of Hyperparameter Tuning

We have attempted to semi-parallelize hyperparameter tuning using a built-in gridsearch (or randomsearch) function in Scikitlearn (paired with keras) across a single node, by manually assigning the number of processes to run the models simultaneously.

Fully parallelizing distributed Keras models is not generically supported within keras/dask_distributed. However, there has been recent development linking dask_distributed directly with tensorflow. We tried to test this process.

Dask Distributed dashboard



worker	*ncores* ▲	cpu	memory	memory_limit	memory %	num_fds	read_bytes	write_bytes
tcp://10.2.1.111:46297	1	1304.7 %	6 GiB	30 GiB	19.2 %	30	114 B	0
tcp://10.2.1.112:40476	1	1401.0 %	6 GiB	30 GiB	20.8 %	30	1 KiB	1 KiB
tcp://10.2.1.113:35252	1	1253.9 %	6 GiB	30 GiB	19.2 %	30	2 KiB	1 KiB
tcp://10.2.1.114:37884	1	1351.6 %	6 GiB	30 GiB	19.6 %	30	1 KiB	1 KiB
tcp://10.2.1.115:41297	1	1068.5 %	6 GiB	30 GiB	19.8 %	30	1 KiB	1 KiB
tcp://10.2.1.118:42028	1	636.7 %	7 GiB	30 GiB	23.8 %	30	1 KiB	101 KiB
tcp://10.2.1.122:38114	1	922.7 %	6 GiB	30 GiB	21.1 %	30	1 KiB	1 KiB
tcp://10.2.1.123:46335	1	575.5 %	7 GiB	30 GiB	22.1 %	30	1 KiB	1 KiB

Figure 7: An example image from the Dask Distributed dashboard

Results and Discussion

Results for Parallel Study – CPU

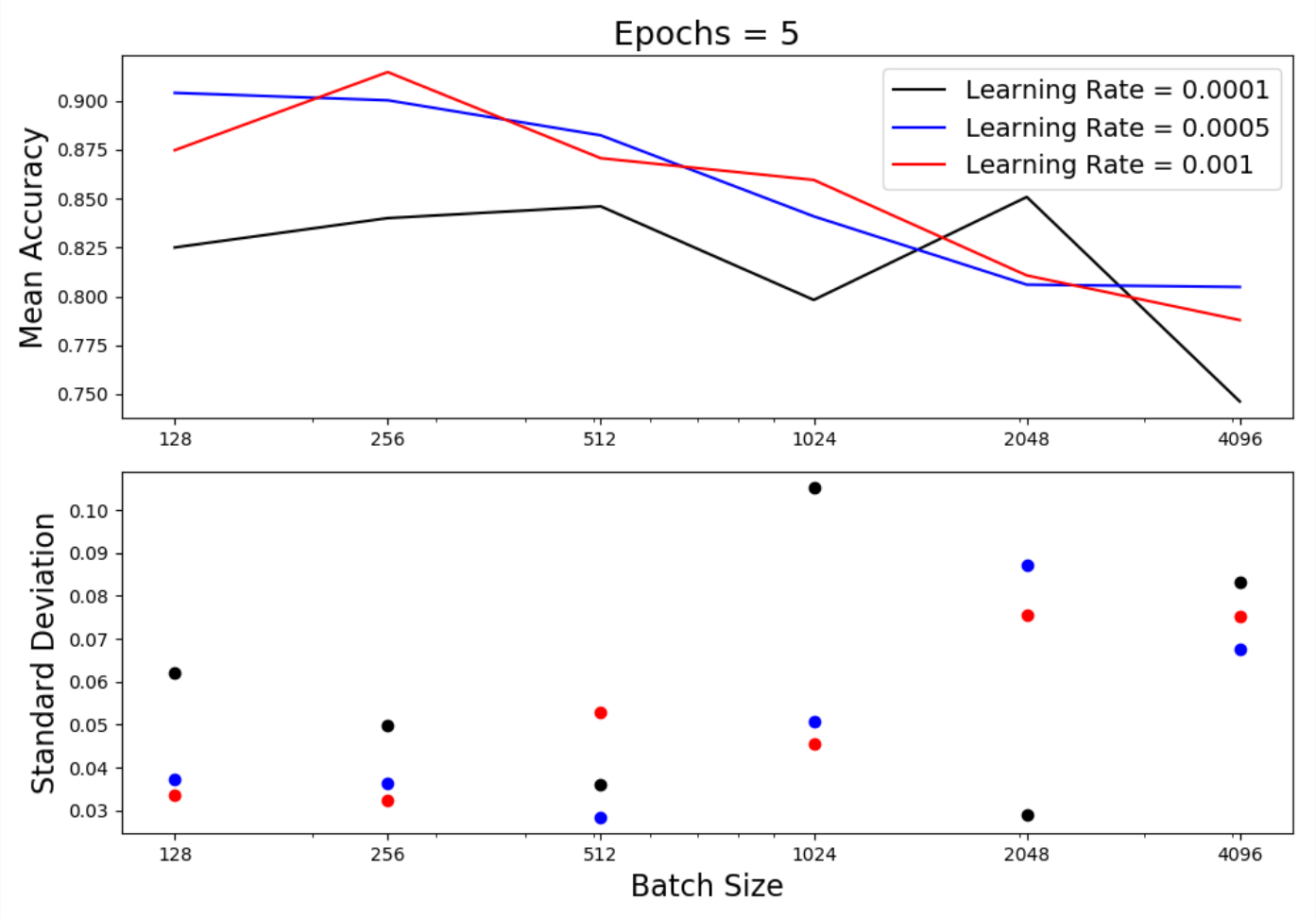


Figure 8: Accuracy results for a range of batch sizes and learning rates with 5 epochs

Results for Parallel Study – CPU (Cont.)

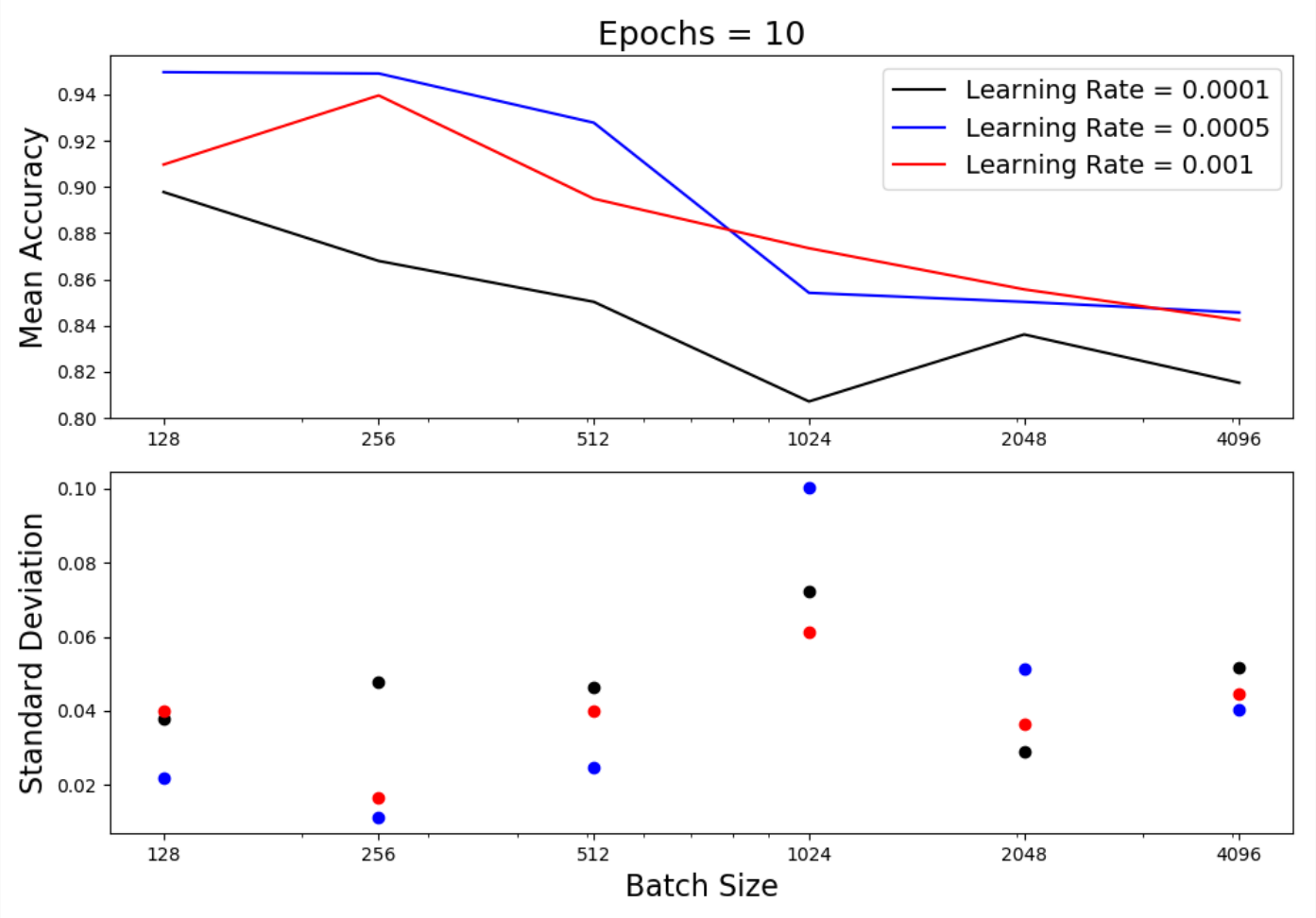


Figure 9: Accuracy results for a range of batch sizes and learning rates with 10 epochs

Results for Parallel Study – CPU (Cont.)

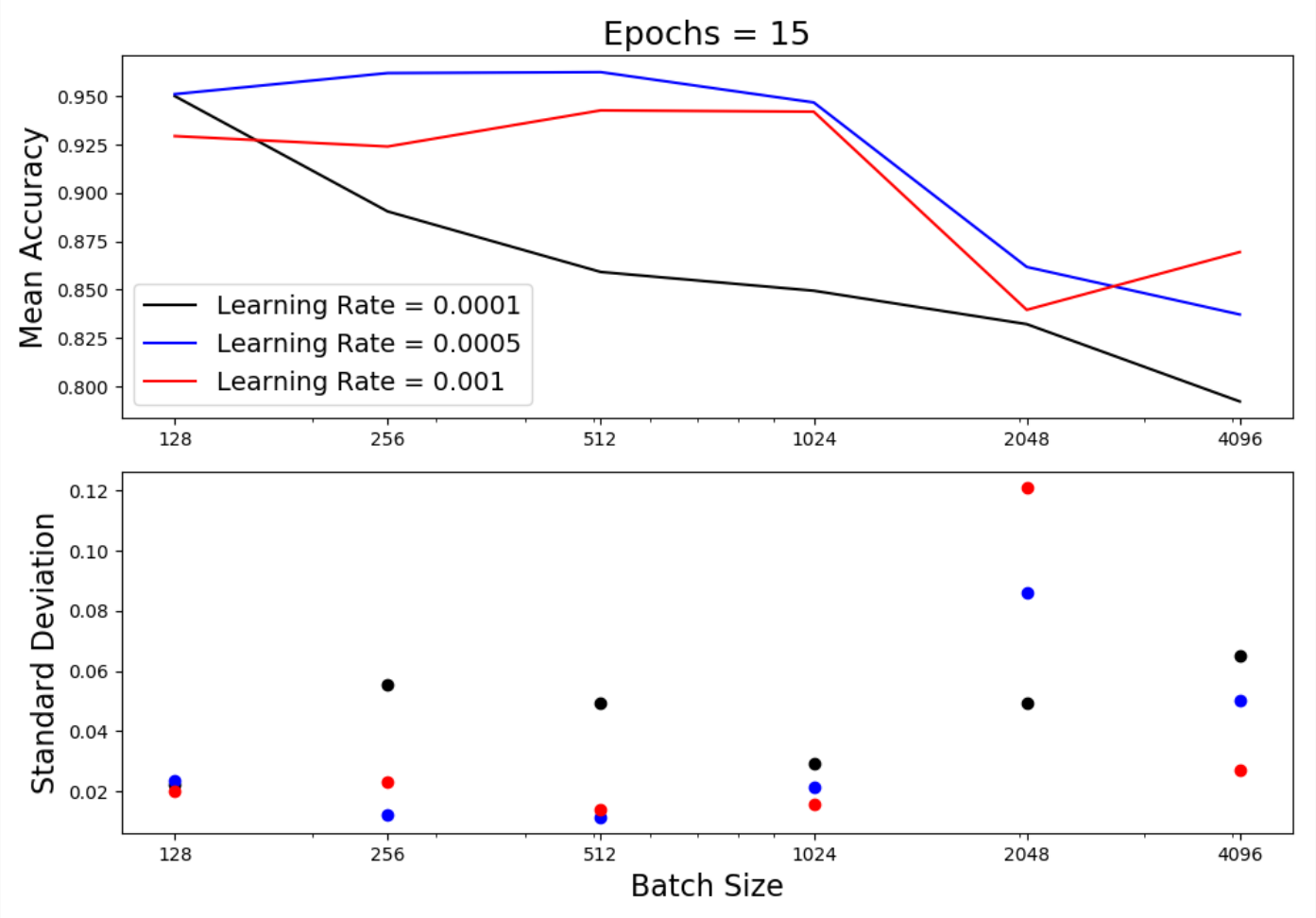


Figure 10: Accuracy results for a range of batch sizes and learning rates with 15 epochs

Results for Parallel Study – GPU

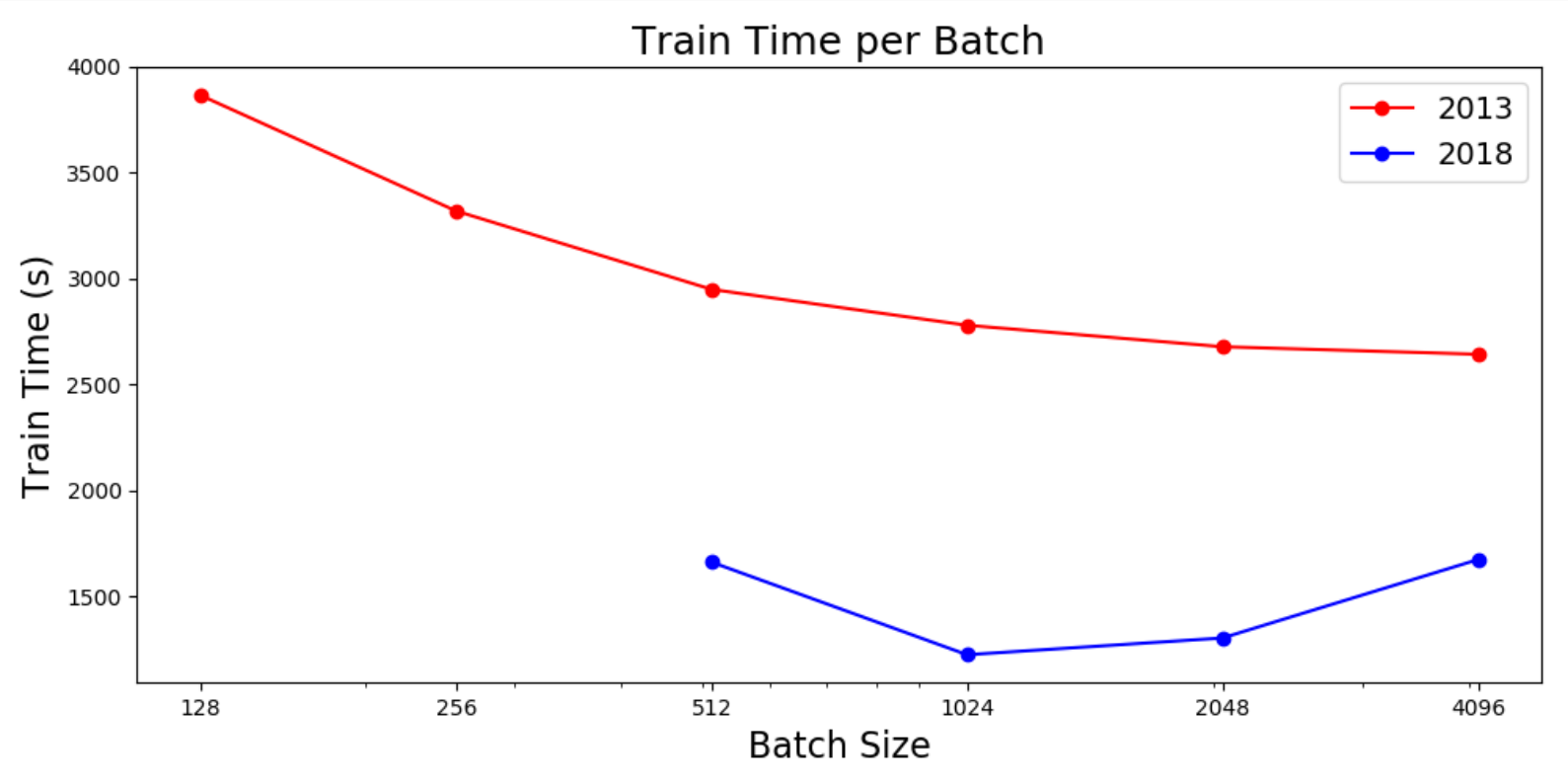


Figure 11: The training time for each batch size for runs on 2013 (red) and 2018 (blue) nodes.

Summary

- Proposed a general framework for augmentation and a parallel tuning scheme for hyperparameters.
- Tested on 2013 and 2018 Taki nodes.
- Efficient parallelism is most comprehensively shown by the timing results in the 2013 and 2018 configurations. The 2018 cluster has twice (32) as many available processors as the 2013 nodes (16).

Potential Future Directions

- Parallelize the hyperparameter tuning process using GPU enabled tensorflow on taki
- Complete a more robust study using a dataset that can train on an order of days rather than hours.

References

- Lindsey R Barnes, Eve C Gruntfest, Mary H Hayden, David M. Schultz, and CharlesBenight. False Alarms and Close Calls: A Conceptual Model of Warning Accuracy.Weather and Forecasting, 22(5):1140–1147, October 2007.
- Wael Ghada, Nichole Estrella, and Annette Menzel. Machine Learning Approach toClassify Rain Type Based on Thies Disdrometers and Cloud Observations.Atmosphere,10(251):1–18, May 2019.
- R. Lagerquist and D.J. Gagne II. Basic machine learning for predicting thunderstormrotation: Python tutorial. `https://github.com/djgagne/ams-ml-python-course/blob/master/module_2/ML_Short_Course_Module_2_Basic.ipynb`, 2019.
- Valliappa Lakshmanan, Christopher Karstens, John Krause, Kim Elmore, AlexanderRyzhkov, and Samantha Berkseth. Which Polarimetric Variables Are Important forWeather/No-Weather Discrimination?Journal of Atmospheric and Oceanic Technology,32(6):1209–1223, June 2015.
- Amy McGovern, Kimberly L Elmore, David John Gagne II, Sue Ellen Haupt, Christo-pher D Karstens, Ryan Lagerquist, Travis Smith, and John K Williams. Using ArtificialIntelligence to Improve Real-Time Decision-Making for High-Impact Weather.Bulletinof the American Meteorological Society, 98(10):2073–2090, October 2017.
- Vahid Nourani, Selin Uzelaltinbulat, Fahreddin Sadikoglu, and Nazanin Behfar. Artifi-cial Intelligence Based Ensemble Modeling for Multi-Station Prediction of Precipitation.Atmosphere, 10(2):80–27, February 2019.