Tornado Storm Data Synthesization using Deep Convolutional Generative Adversarial Network

Carlos A. Barajas[1], Matthias K. Gobbert[1], and Jianwu Wang[2]

[1] Dept. of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21250, USA, {barajasc,gobbert}@umbc.edu,
[2] Dept. of Information Systems, University of Maryland, Baltimore County, Baltimore, MD 21250, USA, jianwu@umbc.edu

Abstract. Predicting violent storms and dangerous weather conditions with current models can take a long time due to the immense complexity associated with weather simulation. Machine learning has the potential to classify tornadic weather patterns much more rapidly, thus allowing for more timely alerts to the public. A challenge in applying machine learning in tornado prediction is the imbalance between tornadic data and non-tornadic data. To have more balanced data, we created in this work a new data synthesization system to augment tornado storm data by implementing a deep convolutional generative adversarial network (DCGAN) and qualitatively compare its output to natural data.

Keywords: deep learning, data augmentation, data synthesization, DCGAN, TensorFlow, Keras

1 Introduction

Forecasting storm conditions using traditional, physics based weather models can pose difficulties in simulating particularly complicated phenomena. These models can be inaccurate due to necessary simplifications in physics or the presence of some uncertainty. These physically based models can also be computationally demanding and time consuming. In the cases where the use of accurate physics may be too slow or incomplete using machine learning to categorize atmospheric conditions can be beneficial [1].

A forecaster must use care when using binary classifications of severe weather such as those which are provided in this paper. The case of a false alarm warning can be harmful to public perception of severe weather threats and has unnecessary costs. On the one hand, an increased false alarm rate will reduce the public's trust in the warning system [2]. On the other hand, a lack of warning in a severe weather situation can cause severe injury or death to members of the public. Minimizing both false alarms and missed alarms are key in weather forecasting and public warning systems.

With advances in deep learning technologies, it is possible to accurately and quickly determine whether or not application data is of a possibly severe weather condition like a tornado. Specifically one can use a supervised neural network

such as a convolutional neural network (CNN) for these binary classification scenarios. These CNNs require large amounts, hundreds of thousands and even millions, of data samples to learn from. Without an ample amount of data to learn from a CNN has no hope of achieving accurate predictions on anything except the original training data provided. Of the 183,723 storms in the data set used in this work only around 9,000 entries have conditions which lead to tornadic behavior in the future [3]. This imbalance of tornado versus no tornado results in a situation where a machine is very good at predicting no potential tornado but is very bad at predicting when there is a tornado imminent hence false negatives.

It is for these reasons there is a real motivation to acquire more data that would result in tornadic conditions. This heralds the need of synthetic data to bolster the amount of data used for training a neural network. Synthetic data must be generated such that it is indistinguishable from real data and can be used in conjunction with the natural data to train a neural network on a more balanced data set which produces less if any false negatives.

2 GAN based Data Augmentation

Each generative adversarial network (GAN) has not just one neural network but rather two networks which compete against each other to generate the best synthetic data possible. There are two parts of a GAN that make it an effective producer of synthetic data. The generator takes in random data and uses this to generate fake data. The discriminator understands what real data looks like and because of this it is capable of determining whether or not any given data is fake or real. Together these two pieces make a GAN capable of producing synthetic data similar to given data that is indistinguishable from the original data qualitatively. The process of training the GAN means providing the discriminator enough data that it can judge whether or not provided data is fake or real. Given this feedback the generator must adapt by generating more realistic data such that discriminator cannot tell the difference between the falsified data and the real natural data. If the generator can fool the discriminator, which is designed to be an expert on the data, then the synthetic data is considered just as good as naturally collected data. Data obtained from this properly tuned GAN is typically considered more robust and a much more effective method for training than the primitive method of duplication [4].

This allows for the generation of a plethora of new data which is distinct from all previously generated data and also unique in that it is not an augmented version of any of the original data. The more interesting prospect is that, given data which is tornadic in nature, a GAN can be used to generate synthetic data that is also promised to be tornadic [5]. With a GAN, new original data can be generated for the training of the CNN that would be used for prediction. This CNN when given real natural data from an upcoming storm would be able to accurately and instantly deliver a verdict of tornadic or not, rather than waiting for a simulation to finish days later.

3 Data

The data set used in this analysis was obtained from the Machine Learning in Python for Environmental Science Problems AMS Short Course, provided by David John Gagne from the National Center for Atmospheric Research [6]. Each file contains the reflectivity, 10 meter U and V components of the wind field, 2 meter temperature, and the maximum relative vorticity for a storm patch, as well as several other variables. These files are in the form of $32 \times 32 \times 3$ images describing the storm. We treat the underlying data as an image and push it through the CNN as if it were a normal RGB image. This allows our findings to generalize to other non-specialized CNNs. Figure 1 shows two examples image from one of these files. Storms are defined as having simulated radar reflectivity of 40 dBZ or greater as seen in Figure 1 (b). Reflectivity, in combination with the wind field, can be used to estimate the probability of specific low-level vorticity speeds. In the case of Figure 1 (a) the reflectivity and wind field were not sufficient enough to cause future low-level vorticity speeds. The dataset contains nearly 80,000 convective storm centroids across the central United States.

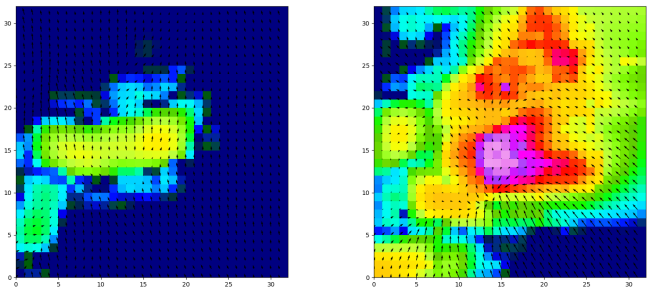
We preprocessed the original NCAR storm data containing 183,723 distinct storms, each of which consists of $32 \times 32 \times 3$ grid points, and extracted composite reflectivity, 10 m west-east wind component in meters per second, and 10 m south-north wind component in meters per second at each grid point giving approximately 2 GB worth of data. We use the future vertical velocity as the output of the network. This gives us 3 layers of data per storm entry producing a total data size of $183{,}723 \times 32 \times 32 \times 3$ floats to feed into the neural network. We use 138,963 storms for training the model and 44,760 storms for testing the accuracy of the model. We track the total wall time for training and testing over both image sets.

With only a handful of tornadic cases present in the base dataset we used primitive augmentation techniques to bolster the number of tornadic cases to be fed into the DCGAN. The only primitive augmentation technique used was rotation. Reflection and translation could generated a storm that might not physically possible.

4 Results: DCGAN Weather Synthesization

The input data was the original input data inflated by primitive augmentation techniques covered in Section 3. The constants for training were: learning rate of 0.001, batch size of 128, data multiplier of 1, and 1 GPU. The images produced by the generator are logged every 25 epochs and evaluated qualitatively as they improve in realism. For more implementation details and additional information, see [7, Appendices A and B].

Consider Figure 2 which contains several rows of images. Each row represents three images generated at the listed epoch number. The first row of the figure contains three images generated before any training was done. whereas the last row represents three images generated after all the training had been done. The



(a) (b)

Fig. 1. Sample images of radar reflectivity and wind field for a storm which (a) does not and (b) does produce future tornadic conditions.

images in the first row are more like noise than real weather which is to be expected as the generator takes in raw noise from a Gaussian distribution. The second row of images are from the 25 epoch marker. Each image has some of the hallmarks of tornadic storm. There are clear attempts at nesting reflectivity levels such putting higher reflectivity in centers or groups. Yet the generator has not been able to really gauge how sensitive the ranges should be and is mixing high and low reflectivity where one might expect the centers to be areas with the most reflectivity. Additionally the wind velocities appear to be very random and non-sensical. At 50 epochs the generator has learned how to more properly gauge relative reflectivity levels. The sizes of the high reflectivity clusters seem very small and uneventful. Additionally there is not enough variation in reflectivity. The images are just mostly high levels of reflectivity rather than concentrated area of high reflectivity which transitions to very low reflectivity over time. The wind patterns are being created in ways that they are moving in ways relative to the clusters of reflectivity which is a positive sign.

75 and 100 epochs are where the generator has a solid grasp of what it should be doing and the differences between these stages are subtler than previous ones. The storms for 75 and 100 epochs have a rich variation of high and low reflectivity especially when compared to previous epoch counts. Each has a clear set of centers that smoothly transition into lower reflectivity as you move away them. The wind patterns have a clear dependence on the centers position and intensity. This is especially apparent when compared to previous epoch markers. At 75 epochs the storms were either very mundane or had a single high activity center or two. The majority of the storms looked well formed but still not as distinct as would be expected. The 100 epoch benchmark produced the most varied and interesting data set. All of the storms have different shapes, sizes, intensities, and centers. There is a clear and smooth transition from high reflectivity to low reflectivity. The wind patterns seem to interact with the centers and have purpose.

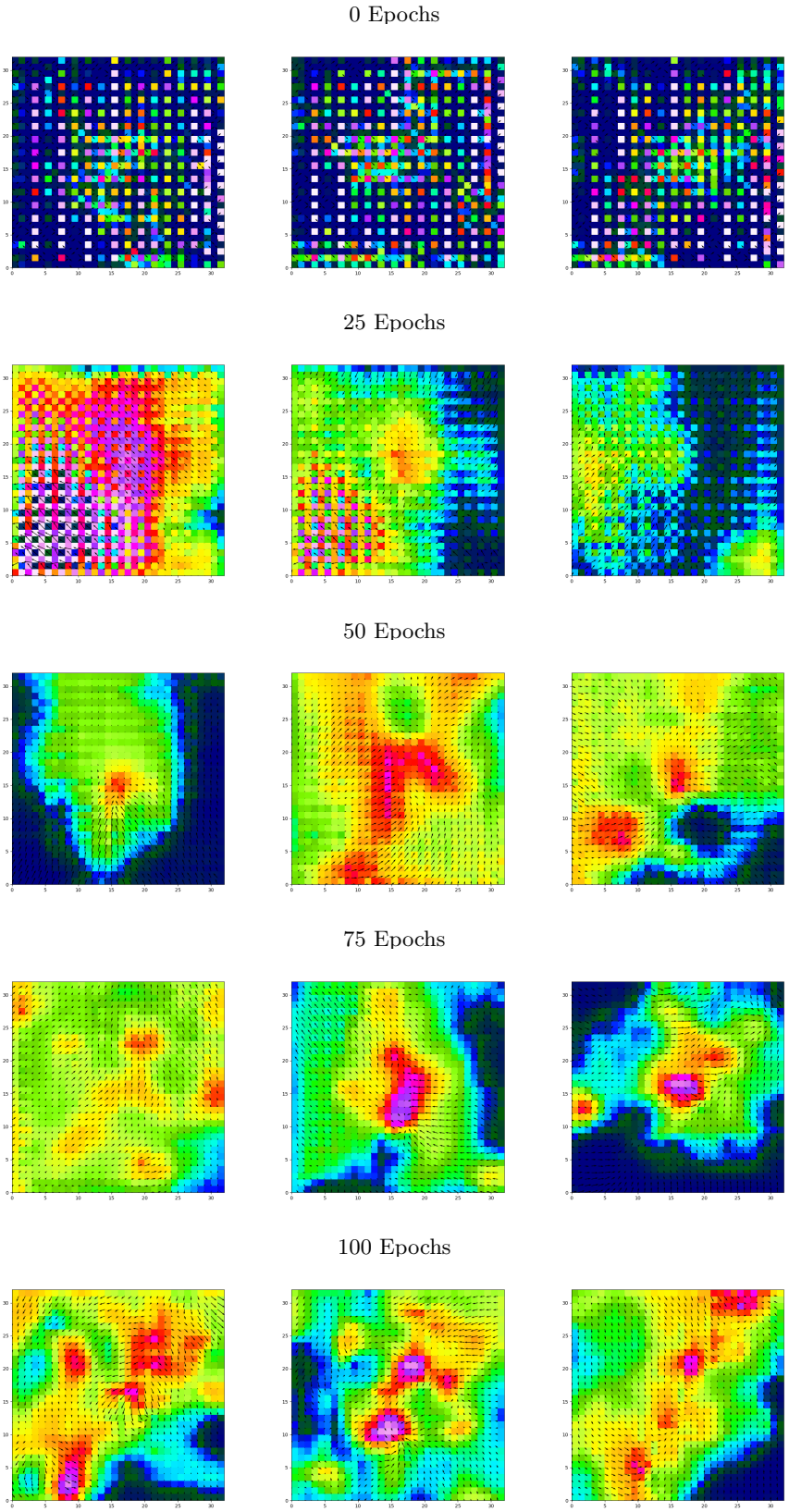


Fig. 2. Sample images of GAN generator output at 0, 25, 50, 75, 100 epochs.

5 Conclusions

The DCGAN is the first complex network trained using the new framework from [3]. The images started out as nothing more than plottable noise. As the epochs progress every new set of generated images slowly gained additional qualities that put them closer to the realm of realism. At the 75 epoch mark most of the storms were mostly indistinguishable from the real storms in the data set, however there were small features which were still not accurate to the careful eye of a layperson. At 100 epochs the images were completely indistinguishable from the real storm data set by a layperson. The transitions, the realistic concentrations of reflectivity, the obvious correlation of wind velocity relative to the concentrated activity, and the rich variety of storms produced were all quality. The data set might be able to be used to train the predictive network to predict real storms as the data is indistinguishable by the casual observer.

Acknowledgment

This work is supported in part by the U.S. National Science Foundation under the CyberTraining (OAC–1730250) and MRI (OAC–1726023) programs. The hardware used in the computational studies is part of the UMBC High Performance Computing Facility (HPCF). Co-author Carlos A. Barajas was supported as HPCF RA as well as as CyberTraining RA.

References

1. Nourani, V., Uzelaltinbulat, S., Sadikoglu, F., Behfar, N.: Artificial intelligence based ensemble modeling for multi-station prediction of precipitation. Atmosphere **10**(2), 80–27 (2019)
2. Barnes, L.R., Gruntfest, E.C., Hayden, M.H., Schultz, D.M., Benight, C.: False alarms and close calls: A conceptual model of warning accuracy. Weather and Forecasting **22**(5), 1140–1147 (2007)
3. Barajas, C.A., Gobbert, M.K., Wang, J.: Performance benchmarking of data augmentation and deep learning for tornado prediction. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 3607–3615. IEEE (2019)
4. dos Santos Tanaka, F.H.K., Aranha, C.: Data augmentation using GANs. ArXiv **abs/1904.09135** (2019)
5. Bok, V., Langr, J.: GANs In Action. Manning (2019)
6. Lagerquist, R., Gagne II, D.J.: Basic machine learning for predicting thunderstorm rotation: Python tutorial. https://github.com/djgagne/ams-ml-python-course/blob/master/module_2/ML_Short_Course_Module_2_Basic.ipynb (2019)
7. Barajas, C.A., Gobbert, M.K., Wang, J.: Tornado storm data synthesization using deep convolutional generative adversarial network (DCGAN): Related works and implementation details. Tech. Rep. HPCF–2020–19, UMBC High Performance Computing Facility, University of Maryland, Baltimore County (2020). URL http://hpcf.umbc.edu