CS230

Generative Modeling of Future Precipitation Patterns

Yuzu Ido
Department of Computer Science
Stanford University
yuzu@stanford.edu

Henri Stern
Earth Systems Program
Stanford University
hsstern@stanford.edu

Abstract

Neural networks are currently under utilized in climate science. This is unfortunate given that they have potential to learn the patterns underlying climate change from many types of long term data. We use PredNet, a convolutional recurrent neural network made to predict the next frame in a video, with an input of 33 years of United States precipitation "images" from 1981 to 2013 to predict that of 2014 (1). While our results were less revealing than was desired, we illustrate that next frame video prediction has the possibility to be used to great effect in field of climate science in the future.

1 Introduction

Over the last few years there has been an explosion in successful machine learning approaches to image based problems. In particular, research focused on autonomous driving and facial recognition has led to many neural net architectures that specialize in interpreting and manipulating data from pictures and video-frames (4). This progress has only slowly been applied to other fields, mostly in the biomedical realm, so we see an opportunity to apply new deep learning-based image processing techniques to climate science (8).

For this project, we apply PredNet, a neural net designed to predict the next frame in a video sequence, to precipitation "images" in an attempt to predict future rain based on past climatological data (1). Specifically, the input is a sequence of 33 grayscale images representing precipitation from 1981 to 2013, where each cell in the rain dataset acts as a pixel with an associated intensity value (2). We then use a deep recurrent convolutional neural network to predict the grayscale image of precipitation in 2014. This is not the first time a guided video prediction problem architecture has been applied to weather dynamics; we have taken inspiration from Requena-Messa, et al. 2021, but it is among a very small number of similar undertakings (3). By testing out a novel climate prediction mechanism, our work will allow us to look into how a deep learning architecture attempts to understand a changing climate over a 34 year time-span.

2 Related work

Related work can be seen in two directions, one of climate science machine learning applications and the other of next-frame video prediction; some papers satisfy both criteria.

The problem of next-frame video prediction formally takes a sequence of n frames $X_{t-n}, \ldots, X_t$ to predict the next frame, $\hat{Y}_{n+1}$. Although the problem initially looks supervised as the next frame serves as a label, it is actually self-supervised as the next frame is already provided in the original data (9). To illustrate the breadth of next frame video prediction technologies available today, Figure

CS230: Deep Learning, Fall 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

1 is graphic from a review paper compiled in 2020 (4). There has been a recent explosion in the field due to the applicability of the technology to the problem of autonomous driving.

Architecture	Approach	Number of Input Frames	Number of Predicted frames	Loss Function	Model	Code
Sequence-to-one	Vukotic [10]	1	Up to 15	L_2 loss	CNN autoencoder	–
	Xue [11]	1	1	KL-divergence and L_1 loss	CNN autoencoder	https://github.com/tfxue/visual-dynamics
	Liu [12]	2	1,2,3	L_1 loss	Pyramid of CNN autoencoder	https://github.com/liuziwei7/voxel-flow
	Liang [6]	5,10	10,20	L_1 and Adversarial loss	GAN+CNN autoencoder+ConvLSTM	–
	Mathieu [13]	4,8	1,8	L_2, L_1, adversarial and gradient difference loss	Pyramid of CNN+GAN	https://github.com/coupriec/VideoPredictionICLR2016
	Villegas [14]	10	up to 128	L_2 and Adversarial loss	LSTM+CNN autoencoder+GAN	https://github.com/ZhongxiaYan/video_prediction
Sequence-to-sequence	Michalski [15]	2, 3, 5	1, 2, 20	L_2 loss	Pyramid of autoencoder and LSTM	–
	Srivastava [16]	10	10	Cross-entropy and L_2 loss	LSTM autoencoder	https://github.com/mansimov/unsupervised-videos
	Denton [17]	5,10	10,20	Cross-entropy, L_2 and Adversarial loss	CNN+GAN+LSTM	https://github.com/ap229997/DRNET
	Oliu [18]	10	10	L_1 loss	CNN+GRU autoencoder	https://github.com/moliusimon/frnn
	Oh [4]	4, 11	1	L_2 loss	CNN autoencoder+LSTM	https://github.com/junhyukoh/nips2015-\action-conditional-video-prediction
	Finn [19]	2, 10	up to 20	L_2 loss	ConvLSTM autoencoder	https://github.com/tensorflow/models/tree/master/research/video_prediction
	Lotter [5]	10	up to 5	L_1 loss	ConvLSTM+CNN	https://github.com/coxlab/prednet
	Wang [20]	3, 10	up to 20	L_1 and L_2 loss	ST-LSTM	https://github.com/ujjax/pred-rnn
	Wang [21]	3, 10	up to 20	L_1 and L_2 loss	Casual LSTM	https://github.com/Yunbo426/predrnn-pp

Figure 1: Observe the wide variety of next frame prediction architectures. Note that PredNet is the third to last entry.

We chose PredNet because of it was highly vetted and well reviewed on GitHub and it had a structure we understood well and felt comfortable manipulating.

There are a number of research labs across the globe that are applying machine learning techniques to climate questions. A very recently published (July 2021) of such a paper uses CNNs to identify patterns in extreme precipitation days (10). This paper also makes use the PRISM data set, although it is more concerned with uncovering underlying patterns of behavior rather than predicting future precipitation events. It was through looking at this paper, along with the next frame prediction review paper (4), and Requena-Messa, et al. (3) that we began to formulate our own ideas.

3 Dataset and Features

Our precipitation dataset comes from PRISM Climate Group, managed by the Northwest Alliance for Computational Science Engineering, based at Oregon State University (2). We use the time series datasets named AN81d for daily millimeters of precipitation from 01/01/1981 to 12/31/2014. This uses information from all stations in the PRISM network across the United States and interpolates their data points using methods known as climatologically-aided interpolation (CAI) and RADAR. The AN81d dataset prioritizes finding the best estimate at the expense of temporal consistency. The relevant data is stored in .bil files, an Arc-GIS binary raster file format. All coordinates inside the United States have a non-negative value corresponding to the millimeters of precipitation on that day in that particular location. All coordinates outside the United States have a -9999 placeholder value. Each file is readable into a numpy 2D array, size 621x1405, float-32 type. The grid resolution is 4 km.

First, during visual examination of the data, the -9999 placeholder drowned out any subtle differences in the actual precipitation, so we changed all those to -10. To build our supervised data points (X, Y), we wanted to let X be the set of numpy 2D arrays corresponding to MM/DD/1981 - MM/DD/2013, and Y be the numpy 2D array for MM/DD/2014, for every day MM/DD of the year except February 29. However, this only provides 365 training examples, which we thought inadequate. Thus, our strategy is to crop each original 2D 621x1405 array into 620x1400, then make each 620x1400 into a

hundred 62x140 2D arrays by combining entries from rows that are the same mod 10, and columns that are the same mod 10 – see figure illustrating this for mod 3.

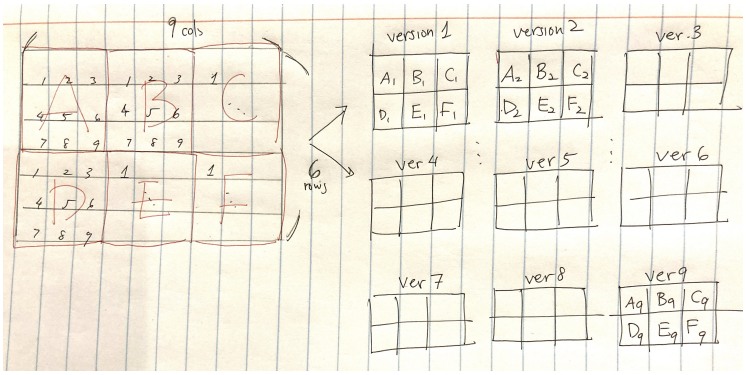

Figure 2: Data Augmentation

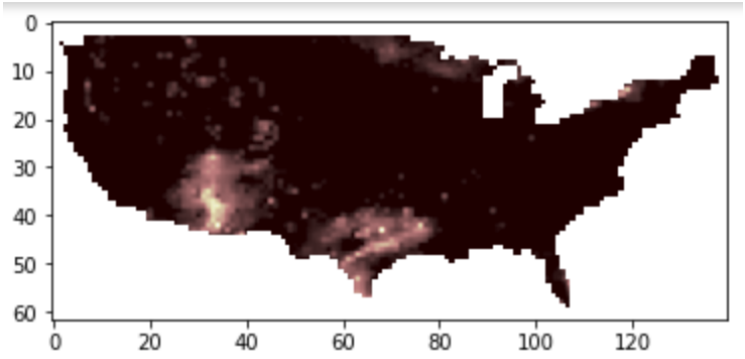

Figure 3: 62x140 image example

Thus for each day, we have 100 versions. Further, due to the model's input size divisibility concerns, we needed to make them 64x144 size, so we padded with -10's, two rows each on the top and bottom, one column each on the left and right. Results are saved to hdf5 files. Doing this on data for all years, the final data points are (X,Y) such that X is the arrays for MM/DD/1981-MM/DD/2013, for a particular version V, and Y is the array for MM/DD/2014, for the same version V. We will thus have 36500 data points, which we split randomly into a 90/5/5 ratio for 32850 train, 1825 test, and 1825 dev. We chose this split because 36500 is not a particularly large data set, so we wanted a more traditional train/dev/test split, and additionally because the PredNet paper used 90/5/5 as well.

Later, we saw that the machine was not learning sufficiently, and thought that it may be due to the large amounts of negative space outside the US in our 62x140 images. Thus, we cropped to an inner box of 44x100 covering the western and central US which had minimal negative space.

4 Methods

We built an adaptation of PredNet for application to the precipitation prediction problem (1). PredNet is a deep convolutional recurrent neural network, designed to take in a sequence of video frames from dash mounted camera footage and predict the next frame, with architecture depicted in Figure 3 (1). Like the name suggests, a convolutional recurrent neural network combines the features of a convolutional neural network (CNN) and a recurrent neural network (RNN). The convolutional neural network is used in computer vision problems and processes image data using filter kernels which act as a sliding window over the images to extract features. The recurrent neural network takes in sequential data such as sound clips and feeds information backward through time so the machine can learn the sequential connections. Thus, this joint structure seemed appropriate for our problem examining sequences of images.

In more detail, according to the PredNet authors, the model is a "series of repeating stacked modules that attempt to make local predictions of the input to the module, which is then subtracted from the actual input and passed along to the next layer. Briefly, each module of the network consists of four basic parts: an input convolutional layer (A_l), a recurrent representation layer (R_l), a prediction layer ($\hat{A}_l$), and an error representation (E_l). The representation layer, R_l, is a recurrent convolutional network that generates a prediction, $\hat{A}_l$, of what the layer input, A_l, will be on the next frame. The network takes the difference between A_l and $\hat{A}_l$ and outputs an error representation, E_l, which is split into separate rectified positive and negative error populations. The error, E_l, is then passed forward through a convolutional layer to become the input to the next layer (A_{l+1}). The recurrent prediction layer R_l receives a copy of the error signal E_l, along with top-down input from the representation layer of the next level of the network (R_{l+1})" (1).

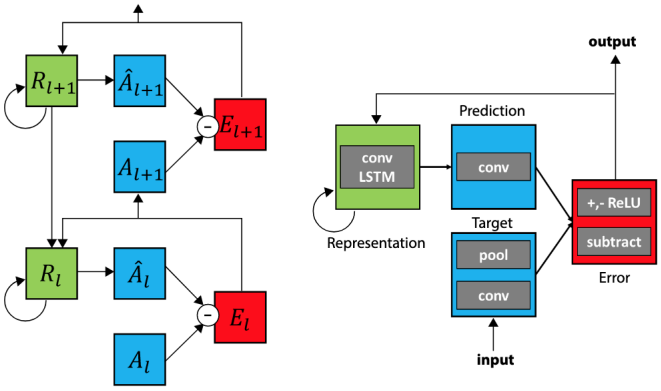

Figure 1: Predictive Coding Network (PredNet). Left: Illustration of information flow within two layers. Each layer consists of representation neurons (R_l), which output a layer-specific prediction at each time step ($\hat{A}_l$), which is compared against a target (A_l) (Bengio, 2014) to produce an error term (E_l), which is then propagated laterally and vertically in the network. Right: Module operations for case of video sequences.

Figure 4: Model architecture from PredNet paper (1)

PredNet has a "loss function implicitly embedded in the network as the firing rates of the error neurons" (1). Our architecture takes in a sequence of the same day across 33 years (1981 - 2013) and use those "frames" to predict the same day in the next year (2014). For example, our network attempts to generate the precipitation of a particular version of January 2nd, 2014 based on the sequential string of that version of January 2nds from 1981 to 2013.

The baseline model for the prediction is the previous frame model, which takes the last image in the X data point (2013) as our prediction for the target, next year (2014).

5 Experiments/Results/Discussion

We consistently ran into issues with AWS, particularly surrounding their limits on data storage and RAM. When working with a large image-based data set, we needed access to a high capacity compute node. Through Henri's research position in the Department of Earth System Science, we had access to the Sherlock computing cluster at the Stanford Research Computing Center. We used a compute node with 24 processors, 100 GB of RAM, and terabytes of available temporary storage. We did have to settle for a system without a GPU, but that only slowed us down a little. That was not a problem since we were not being charged by hour used as we would have been on AWS.

Hyperparameter tuning

We began our experimentation process with the hyperparameters that the original PredNet paper used. Then, in the first series of tests and redesigns, we used combinations of learning rates in

$\{0.01, 0.001, 0.0001\}$, batch sizes in $\{4, 32, 64\}$, varying numbers of samples per epoch, varying numbers of epochs, mean absolute error and mean squared error, three and four layer network structures, and a parameter particular to the PredNet structure, the layer loss weights. Because each layer of PredNet has its own associated loss, the authors of the paper allow for two different means of taking those losses into consideration. In both cases, the loss associated with the first layer is given 100% of its weight, then either each successive layer is given 10% weight or each successive layer loss is discounted with 0% weight. The PredNet authors chose the Adam optimizer for their algorithm, and we did not modify that.

In this stage of hyperparameter tuning and experimentation, we ran over 30 different versions of the model. Each time, we ran into several consistently reappearing issues. We saw repeatedly that the validation error was lower than the training error. We saw that after a small number of epochs, the training error would cease to decrease monotonically and begin to fluctuate within a relatively small range. One possible explanation could be that because our data set is relatively small (less than 40000 examples), and each data point looks relatively similar, the machine sufficiently learns with just a few epochs, but this theory is not supported by the results in the following section. While we have not calculated the statistical significance of this observation, it is worth pointing out that we frequently saw the lowest loss appear after 7 epochs. It is unclear as to why that would occur so many times. Included below is a graph of typical training and validation loss per epoch curves.

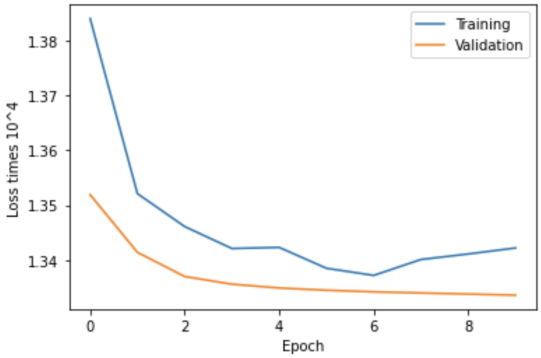

Figure 5: Training and validation loss for an illustrative run. Note that validation loss is anomalously lower than training loss and training loss does not decrease monotonically.

In this first round of experimentation, we found that we minimized loss when the hyperparameters were set to Learning Rate = 0.001, Batch Size = 32, Batches per Epoch = 20, a three layer architecture, Layer Loss Weights = (1, 0, 0). With this set up, we reached a minimum loss of 1.3356×10^{-4}.

After several rounds of hyperparameter tuning where we tried variations in batch size, learning rate, loss function, and layer-level loss weight, we found that although the numerical loss seemed to be decreasing during training, our model was consistently predicting exactly the same thing. Regardless of hyperparameters chosen, the model predicted a value of -10 for every pixel. An example of this can be seen in Figure 6.

Results

Looking at the situation we were presented with, we theorized that the sparseness of the pixels with true data values rather than placeholder values might be the source of the issue. The United States is far from rectangular and the PRISM data set only has values within the boundaries of the continental United States. As such, we decided to investigate what would happen if we selected a inner box that was simultaneously large enough to include identifiable weather patterns and small enough to include as few placeholder pixels as possible. As described in Section 3, we selected a rectangle of the Western and Central United States. We then set about the process of experimentation and hyperparameter tuning again. We will refer to this new set of iterations as the inner-box model, and the previous as the preliminary model.

We again tested a wide variety of batch sizes, learning rates, model depth, epoch quantity, and layer loss weights. We did notice that with the inner-box model, the validation loss was now larger than the

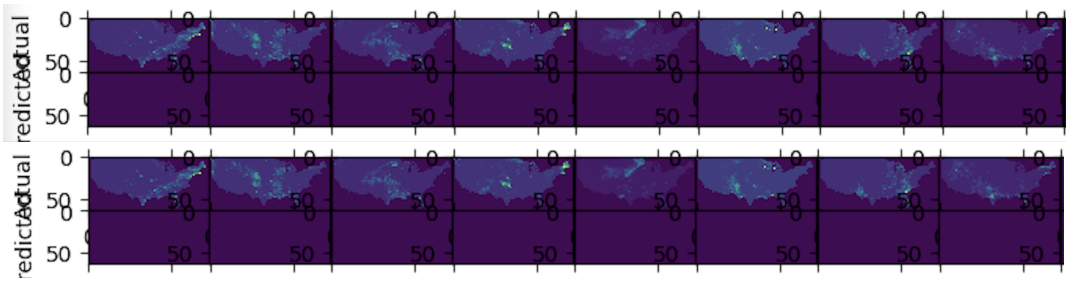

Figure 6: 2 samples of preliminary model results; in both sets of images actual values are shown over predicted values. Note that all predicted images are the same regardless of the actual values.

training loss, and we had more consistent decreases in the loss across epochs. This seemed promising, however when we went to evaluate the model outputs, we found that we were in exactly the same situation as we had been with the preliminary model (see Figure 7).

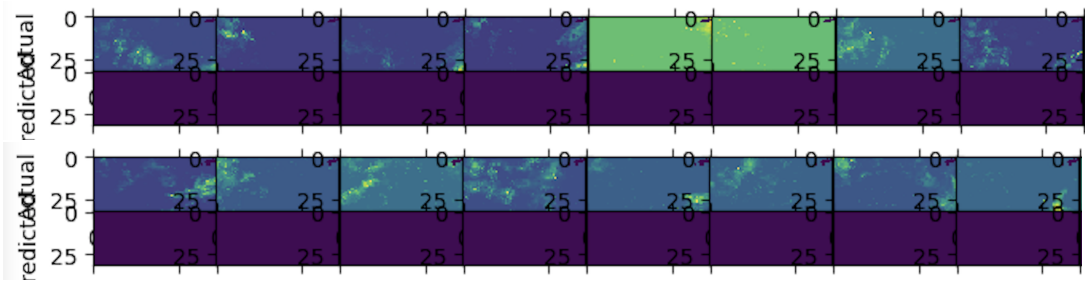

Figure 7: 2 samples of Inner-Box version's results; in both sets of images actual values are shown over predicted values. Note that all predicted images are the same regardless of the actual values.

This set of predictions was generated by a machine whose hyperparameters were learning rate = 0.001, batch size = 32, batches per epoch = 20, a three layer architecture, layer loss weights = (1, 0.1, 0.1). It achieved a minimum loss of 1.2795×10^{-5}, an order of magnitude better than the preliminary machine.

Discussion

This abject failure to learn is confounding on a number of levels. Most crucially is that even when we shifted to an inner-box, where the vast majority of values were non-negative, we still ended up with the same ubiquitous -10 valued prediction. This is hard, near impossible, to explain especially given the fact that it seems that if the machine were to select one value to broadcast over the whole system, a non-negative value would yield lower losses.

Models vs Baselines		
Model Version	Model MSE	Previous Frame MSE
Preliminary	0.001119	0.000655
Inner-Box	0.000439	0.000700

Figure 8: Comparing model prediction performance to baseline prediction performance.

Figure 8 is also curious. It makes sense that the blank images produced by the preliminary model (see Figure 6) are farther from the ground truth than the previous frame image (i.e., the same day of from the previous year) because at least the previous frame has some sort of weather, some non-negative values. What doesn't make sense is that when we restricted the domain in the inner-box version of the model, the same generated blank images actually perform better than the previous frame baseline. Why would this region demonstrate different behavior from the system as a whole? Furthermore, the numbers seem to indicate that the inner-box model is doing more than twice as well as the preliminary model and there is nearly identical baseline error between the two systems. And yet, we can see no

discernible qualitative difference between the outputs of the two models. We have searched the image generation code for an error that might have caused this and found nothing.

We must also note that our model's set up and structure is not significantly different from the original paper's. The order of magnitude of the size of our data sets and the size of the images under consideration are similar. And yet, their paper shows their model vastly outperforming the previous frame baseline, while we do not obviously replicate that success. One possible explanation is the nature of the physics involved in the two use-cases. The original PredNet paper deals with frame prediction from footage taken from dash-mounted cameras (1). This means that their system was designed predominantly to understand and predict kinematic physics. The atmosphere is a chaotic fluid whose physical properties, particularly surrounding phenomenological propagation and spontaneous appearance, are far different from the classical Newtonian case considered in the original paper. Obviously, we cannot know for sure if the machine is learning these physical properties, but it is a strong possibility.

6 Conclusion/Future Work

According the quantitative success metrics (Figure 8), the inner-box model architecture was the most successful. But, a qualitative observation of the results show that neither the preliminary model nor the inner-box model yielded results of any significance. Unfortunately, it is difficult to understand this failure. We did expect the inner-box model to be better performing because of the lack of placeholder values in the system. However, we cannot explain why that machine's improvements in MSE do not translate to qualitative improvements in the predicted image.

We decided against performing feature extraction on the model because of the apparent lack of learning. This is unfortunate because we believe that if the model had been more successful, the feature extraction portion might have actually been the most useful and important part. It is possible that in the course of attempting to predict the next year's weather based on a climatological record, the model could learn something about the fundamental nature of climate change. In the future, if the problems in this system's architecture could be found, there might be much to learn from the feature extraction process.

A natural consequence of strong next-frame video prediction technology Zhou et al. refer to as Sequence-to-Sequence capacity (4). In the course of this project we used a set of n images to predict the $(n+1)^{st}$ image. Once that is done, you are then left with $n+1$ images and you can repeat the process to generate an $(n+2)^{nd}$ image, an $(n+3)^{rd}$ image, and so on. This way, you can extend prediction capacity farther and farther into the future. Lotter et al. found that their prediction capacity decayed rapidly and that they could only reasonably predict five frames into the future (1). However, that paper only considered seven images to predict an eight, so it is no surprise that, when the majority of reference frames in use have themselves already been simulated, prediction capacity drops precipitously. In our set up, we used more than 30 images in the reference set, so we may not be limited to five steps forward as Lotter et al. were. If we had more time, and a better trained network, it would be interesting to see what would happen if we extended the predictions beyond just the next frame.

Although this project did not turn out as we had hoped, we learned a lot along the way and are content in the knowledge that we spent many many hours trying to do the best possible job we could.

7 Contributions

This project was a team effort– all portions of the project were worked on by both of us, with one person typically taking the lead for a part. We contributed equally to the generation and execution of the idea. For the initial stage of the project up to the milestone, Henri led the preliminary research with related works while Yuzu worked on data preprocessing in Colab. Then, after deciding on the PredNet algorithm, Henri focused on reading the provided GitHub code while Yuzu focused on learning the AWS system. Due to memory constraints and rejection from AWS for more resources, we switched to the Stanford Sherlock research machines, with Henri taking the lead on this. After this, we worked equally on the model.

References

[1] Lotter, William, et al. Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. 2016. https://arxiv.org/abs/1605.08104

[2] "PRISM Climate Data." PRISM Climate Group, Northwest Alliance for Computational Science Engineering, 2021, https://prism.oregonstate.edu

[3] Requena-Messa, et al. EarthNet2021: A large-scale dataset and challenge for Earth surface forecasting as a guided video prediction task. 2021. https://arxiv.org/abs/2104.10066

[4] Y. Zhou, H. Dong and A. El Saddik, "Deep Learning in Next-Frame Prediction: A Benchmark Review," in *IEEE Access*, vol. 8, pp. 69273-69283, 2020, doi: 10.1109/ACCESS.2020.2987281.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[6] Chollet, F., et al., 2015. Keras. Available at: https://github.com/fchollet/keras.

[7] Bengio, Yoshua. How auto-encoders could provide credit assignment in deep networks via target propagation. CoRR, 2014.

[8] Shamout, F., et al. "Machine Learning for Clinical Outcome Prediction." IEEE Reviews in Biomedical Engineering, Biomedical Engineering, IEEE Reviews in, IEEE Rev. Biomed. Eng, vol. 14, Jan. 2021, pp. 116–126., doi:10.1109/RBME.2020.3007816.

[9] Oprea, Sergiu, et al. A Review on Deep Learning Techniques for Video Prediction. 2020., doi:10.1109/TPAMI.2020.3045007.

[10] Davenport, F. V., Diffenbaugh, N. S. (2021). Using machine learning to analyze physical causes of climate change: A case study of U.S. Midwest extreme precipitation. Geophysical Research Letters, 48, e2021GL093787. https://doi.org/10.1029/2021GL093787