Ground Weather RADAR Signal Characterization through Application of Convolutional Neural Networks

THESIS

STEPHEN M LEE, 1st Lt, USAF

AFIT-ENS-MS-20-M-158

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-20-M-158

GROUND WEATHER RADAR SIGNAL CHARACTERIZATION THROUGH APPLICATION OF CONVOLUTIONAL NEURAL NETWORKS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

STEPHEN M LEE, B.S.

1st Lt, USAF

26 March 2020

AFIT-ENS-MS-20-M-158

GROUND WEATHER RADAR SIGNAL CHARACTERIZATION THROUGH APPLICATION OF CONVOLUTIONAL NEURAL NETWORKS

THESIS

STEPHEN M LEE, B.S.
1st Lt, USAF

Committee Membership:

Dr. Lance E. Champagne
Chair

Lt Col Andrew J Geyer, Ph.D.
Reader

AFIT-ENS-MS-20-M-158

Abstract

The 45th Weather Squadron supports the space launch efforts out of the Kennedy Space Center and Cape Canaveral Air Force Station for the Department of Defense, NASA, and commercial customers through weather assessments. Their assessment of the Lightning Launch Commit Criteria (LLCC) for avoidance of natural and rocket triggered lightning to launch vehicles is critical in approving space shuttle and rocket launches. The LLCC includes standards for cloud formations, which requires proper cloud identification and characterization methods. Accurate reflectivity measurements for ground weather radar are important to meet the LLCC for rocket triggered lightning. Current linear interpolation methods for ground weather radar gaps result in over-smoothing of the vertical gradient and over-estimate the risk of rocket triggered lightning, potentially resulting in costly, unnecessarily delayed launches. This research explores the application of existing interpolation methods using convolutional neural networks to perform two-dimensional image interpolation, called inpainting, into the three-dimensional weather radar scan domain. Results demonstrate that convolutional neural networks can improve the accuracy of cloud characterization over current interpolation methods, potentially resulting in fewer launch delays with substantial associated cost savings due to increased capability to meet the LLCC.

Acknowledgements

I would like to thank Mr. William Roeder with the 45th Weather Squadron for his insights. I would also like to thank my faculty research advisor Dr. Champagne, and committee member, Lt Col Geyer, for their support. Finally, I would like to thank my wife and children for their patience.

STEPHEN M LEE

Table of Contents

List of Figures

List of Tables

GROUND WEATHER RADAR SIGNAL CHARACTERIZATION
THROUGH APPLICATION OF CONVOLUTIONAL NEURAL NETWORKS

I. Introduction

1.1 Problem Statement

Can a convolutional neural network (CNN) result in more accurate ground weather radar (WR) gap interpolation over the current linear interpolation model for cloud characterization?

1.2 Background

The 45th Weather Squadron (WS) supports space launch efforts out of the Kennedy Space Center and Cape Canaveral Air Force Station for the Department of Defense, NASA, and commercial customers through weather assessments to satisfy its mission to exploit the weather to assure safe access to air and space[3]. A part of its mission is to assess the Lightning Launch Commit Criteria (LLCC). The LLCC for avoidance of natural and rocket triggered lightning to launch vehicles during ascent must be satisfied to approve space shuttle and rocket launches[4]. The LLCC includes standards for cloud formations, which requires proper cloud identification and characterization methods.

The area covered by a radar beam as the antenna rotates through elevation scans is called its volume coverage pattern (VCP). Part of the radar scan strategy for ground weather radar uses different, contextual based VCPs in order to increase scan efficiency. Unfortunately, this means ground WR suffers from vertical gaps as the elevation angle increases (see figure 1).

Current linear interpolation methods to fit the data result in smoothing and lose small scale information[5]. This smoothing may result in predicted cloud shapes that over-estimate the risk of natural and rocket-triggered lightning. An improvement in vertical interpolation method may identify additional launch opportunities that meet the LLCC and would otherwise be missed.

This is especially applicable to cumulus clouds that visually have extremely sharp edges. Sharp edges imply that these cumulus clouds will have very high gradients of reflectivity at the cloud edges including cloud top. Therefore, the linear interpolation method currently used to fill-in the WR beam gaps will underrepresent the true gradient. This will lead the WR to overestimate the height of the cumulus cloud top, which in turn will lead to falsely evaluate the LLCC Cumulus Rule as violated under conditions where the lauch vehicle trajectory will approach too close to the cloud body.

According to NASA-STD-4010, "The LLCC identify each condition that is required to be met in order to launch. These include criteria for trained weather personnel to monitor the meteorological conditions and implement each launch constraint developed using the following Natural and Triggered Lightning Launch Commit Criteria. The launch operator is required to have clear and convincing evidence that none of the criteria is violated at the time of launch." The LLCC was instituted in response to events where Apollo XII and Atlas/Centaur-67 triggered lightning upon launch. Apollo XII recovered and completed the mission successfully; however, Atlas/Centaur-67 was destroyed, and "since Atlas/Centaur-67 and the rigorous implementation of the LLCC, no other launch vehicles have intercepted or triggered lightning on launch[4]."

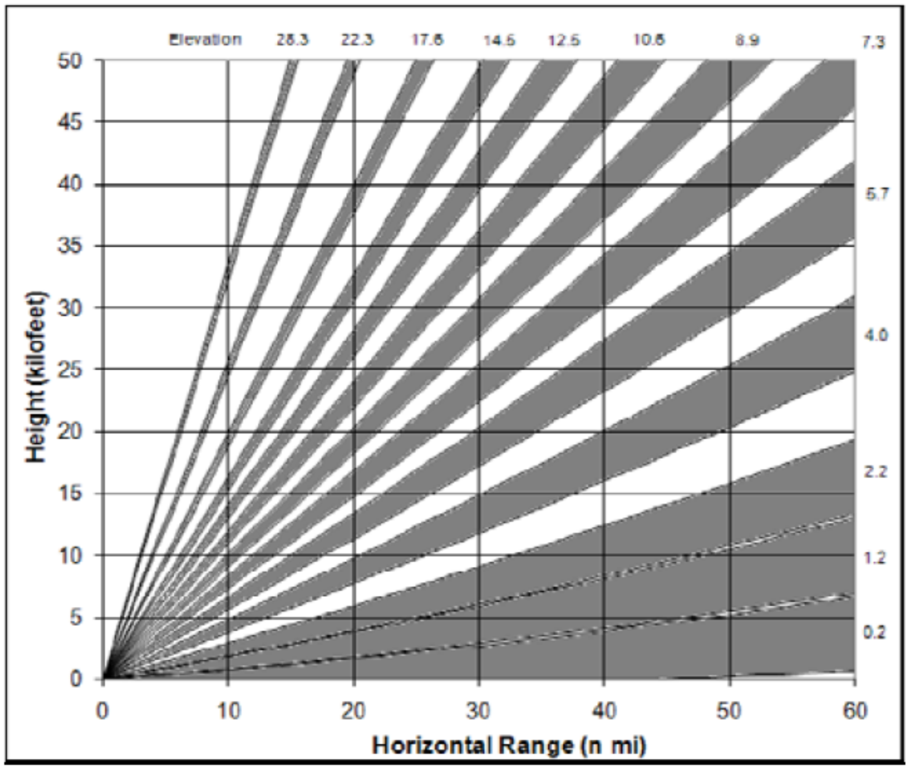


Figure 1. Radar Beam Coverage versus Range [1]

According to the 45th WS, "approximately 35% of launches from CCAFS/KSC are delayed, and 5% scrubbed, due to the Lightning Launch Commit Criteria," which can cost hundreds of thousands, if not millions of dollars per lost launch opportunity[6]. These potential costs drive the 45th WS and space launch community to constantly seek incremental improvements to the LLCC prediction ability.

Data is provided from National Centers for Environmental Information (NCEI) website National Oceanic and Atmospheric Administration (NOAA). NCEI's radar archive includes the Next Generation Weather Radar System (NEXRAD) and Terminal Doppler Weather Radar networks[7]. This research focuses on the base data, called Level-II, which includes the original three meteorological base data quantities: reflectivity, mean radial velocity, and spectrum width, as well as the dual-polarization base data of differential reflectivity, correlation coefficient, and differential phase[8].

1.3 Motivation

The objective of this research is to develop an improved cloud-shape interpolation method. Current WR interpolation uses simple averaging of reflectivity between radar scans and does not perform any extrapolation for missed cloud tops or bottoms[9]. Using CNNs for WR interpolation is a novel solution approach to this problem since it expands the already well-performing CNN model architecture for two-dimensional color image interpolation into a new three-dimensional cloud reflectivity image domain.

1.4 Summary

Chapter 2 reviews current research into weather prediction as well image interpolation using neural networks. Chapter 3 explains the methodology used to develop the ground weather radar reflectivity interpolation model as well as its evaluation criteria. Chapter 4 presents the model results and compares them to the existing linear interpolation model. Chapter 5 covers the the conclusions and proposes possible avenues for future research.

II. Literature Review

2.1 Overview

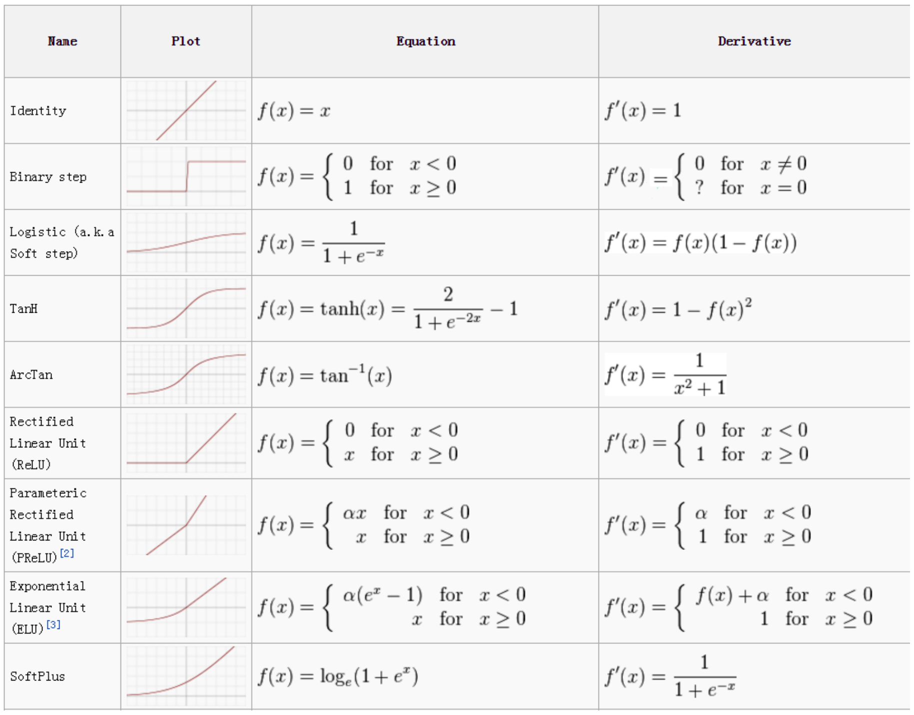
This chapter discusses past research into ground weather radar scan elevation gap linear interpolation methods as well as background for nonlinear interpolation using deep learning neural networks through CNNs.

2.2 Radar Gap Interpolation

Lakshmanan et. al. demonstrate how linear interpolation methods reduce the bias and variance in cloud formation height estimation. Despite the displayed advantages, however, they state that there is not always improvement over traditional scanning techniques[10]. Augst and Hagen have outlined the issues with classic linear regression by highlighting how "measurements with a great elevation angle have less weight to the calculation of the velocity," low elevation angles neglect the vertical wind component of radial velocity, and operational radars have "a lack of data coverage increasing with altitude[5]." Essentially, due to the nature of ground WR data collection, higher elevation angles have less power to calculate the reflectivity cross section, as well as result in larger radar scan gaps, with the possibility of missing portions of the cloud tops entirely without the ability to extrapolate.

2.3 Neural Network Architecture

Neural networks define a series of mapping functions, often called activation functions, to best approximate the relationship between inputs and outputs (see figure 2 for some common activation functions). Activation functions evaluate in chained equations, referred to as the layers of the network, where the functions make up the nodes of the network and the connections are defined by the weighted outputs of each previous layer. Neural networks seek to minimize the error between the approximations and true values by updating the weights through stochastic gradient descent[11].



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 2. Common neural network activation functions[2]

The output f from each layer is defined as

$$\mathbf{f}^{(1)} = g^{(1)}\left(\mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}\right) \tag{1}$$

$$\mathbf{f}^{(2)} = g^{(2)}\left(\mathbf{W}^{(2)T}\mathbf{f}^{(1)} + \mathbf{b}^{(2)}\right) \tag{2}$$

$$\cdots$$

$$\mathbf{f}^{(n)} = g^{(n)}\left(\mathbf{W}^{(n)T}\mathbf{f}^{(n-1)} + \mathbf{b}^{(n)}\right) \tag{3}$$

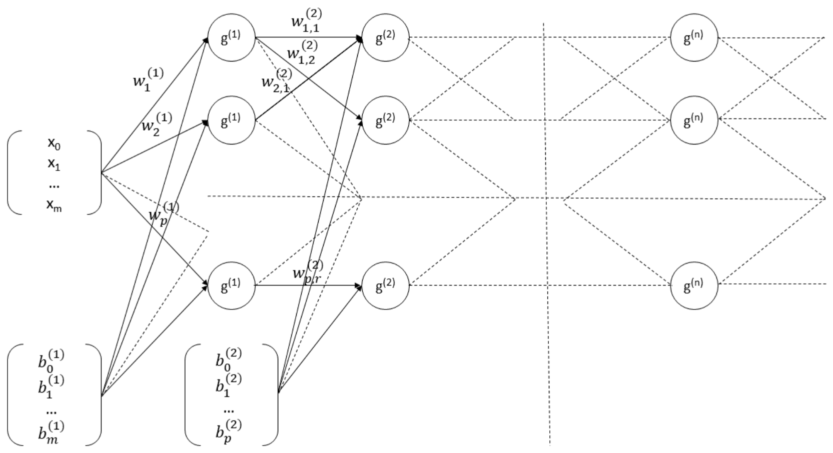


Figure 3. Simple feedforward neural network

for n layers where g is the activation function used for each layer, x is the input for the initial layer, W is the weights for the inputs, and b is the bias for the function.

CNNs utilize at least one convolution step to emphasize the relationship between observations close to each other when organized into a grid-like structure. This is accomplished by using a kernel of a smaller size than the input data to filter the image into feature maps for the activation functions.

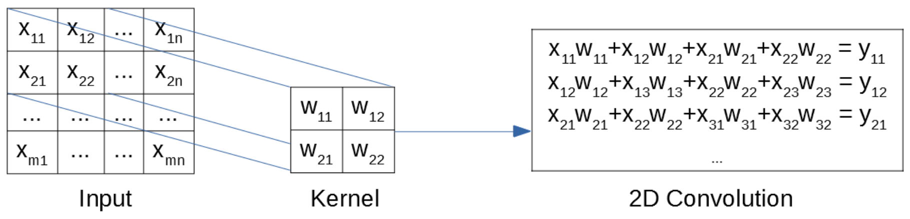
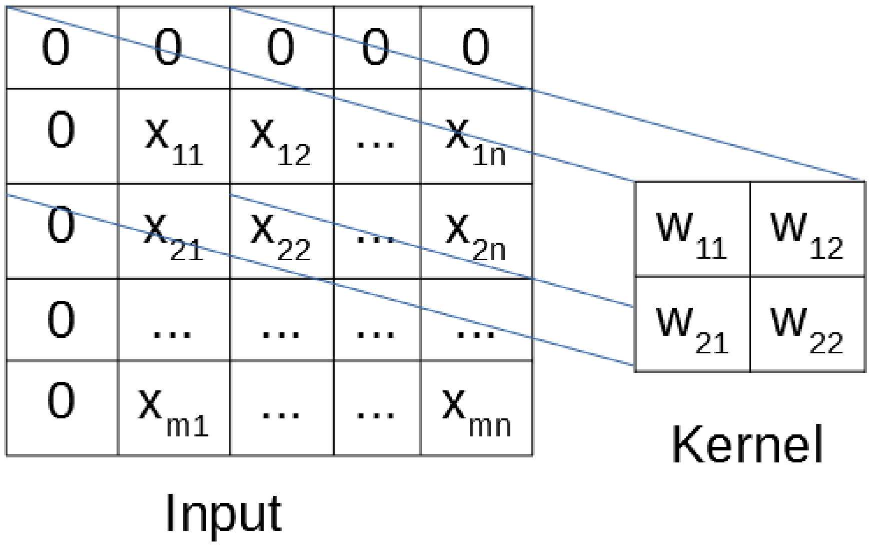

x_{11} x_{12} ... x_{1n}
x_{21} x_{22} ... x_{2n}
... ...
x_{m1} ... ... x_{mn}
w_{11} w_{12}
w_{21} w_{22}
$x_{11}w_{11}+x_{12}w_{12}+x_{21}w_{21}+x_{22}w_{22} = y_{11}$
$x_{12}w_{12}+x_{13}w_{13}+x_{22}w_{22}+x_{23}w_{23} = y_{12}$
$x_{21}w_{21}+x_{22}w_{22}+x_{31}w_{31}+x_{32}w_{32} = y_{21}$
...
Input
Kernel
2D Convolution

Figure 4. 2D Convolution

Since the kernel is the same throughout a layer, the relationships between observations in the grid are equally weighted. Additional convolutional layers allow indirect connections between nodes that would not normally be possible since the direct connections are limited by the size of the kernel. Methods to increase the performance of CNNs include pooling the outputs of nodes into summary statistics that reduce the dimensionality of the output space, as well as padding the edges of the grid with empty values in order to allow a kernel to fully map the features along the outside of the grid. See figure 5 for an example of padding.



0 0 0 0 0
0 x_{11} x_{12} ... x_{1n}
0 x_{21} x_{22} ... x_{2n}
0 ... ...
0 x_{m1} ... ... x_{mn}
Input
w_{11} w_{12}
w_{21} w_{22}
Kernel

Figure 5. Padding allows the kernel to fully scan edges and corners

2.4 Photo Image Interpolation using Neural Networks

Notable work using CNNs for two-dimensional image interpolation, or inpainting, includes the method introduced at the 32nd Conference on Neural Information Processing Systems[12]. Now referred to as Multi-Column Generative Advesarial Networks (GANs), these special CNNs selectively mask portions of image data in order to make predictions of the missing contents. This method incorporates methods from context-encoders[13] where portions of an image are predicted based on their surroundings, and GANs [14], which introduce a separate branching network that attempts the classify the predictions as real or fake and minimizes the ability of the network to discriminate between the truth values and the generated values. A multi-column GAN requires a pre-trained classification network that is incorporated into the context based interpolator that attempts the trick the discriminator. As the efficacy of two-dimensional photo image inpainting increases it leaves an opportunity to start at the ground up for three-dimensional images such as radar scans.

2.5 Radar Image Interpolation with CNNs

Current ground WR work uses CNNs for "nowcasting," or radar scan extrapolation for precipitation. Examples include using a recurrent neural network combined with a CNN to develop a prediction layer with a image processing convolutional layer[15], as well as current work using downsampling/upsampling CNN blocks to predict subsequent radar scans[16]. These are useful for rainfall predictions, but there is currently no work using image interpolation using neural networks on ground WR scans for cloud characterization.

The use of radar super resolution attempts to use high resolution scans as truth data in order to improve the estimates from lower resolution scans[17]. These structures use a modified GAN framework for training in order to obtain a high amount of visual similarity between the different radar scan resolutions for post-processed two-dimensional representations. The limitations of super resolution data to develop a more generalized framework include the limited accessibility of high resolution scans as well as the fact that super resolution frameworks do not account for masked or missing scans.

III. Methodology

3.1 Data Description

The data in this study consists of 1246 radar scans from NOAA for 20 to 24 June 2016 in Tampa, Florida. Scans were extracted into netCDF file format using NOAA's weather and climate toolkit[18]. The reflectivity, elevation, azimuth, and distance measurements of each scan were read into numpy array files. Scan sizes were standardized between scans with different volume coverage patterns by padding unobserved indeces in scans with zeros. Final array sizes were 1556 distance measurements around 360 azimuth angles and 11 elevation scans with a reflectivity measurement at each index. Radar scans were randomly separated into training, validation, and test sets, with a training set of 873 radar scans, validation set of 251 radar scans, and test set of 122 radar scans.

Suitable truth data is required for predicting the reflectivity of unobserved elevation angles. This study assumes that a model that can predict reflectivity at masked elevation angles will be sufficient to predict the reflectivity at unknown elevation angles. Since NOAA does not record reflectvity for echo top scans below 18.5 decibels relative to the reflectivity factor (dBZ), dBZ being a unit of measurement comparing the return of a radar signal called the reflectivity, this study will be limited to cloud top predictions at 18.5 dBZ and above. Input data included all scan information where the masked elevation angle was assigned 99999 for reflectivity, and output data consisted of the predictions for reflectivity at the masked elevation angle. The masked elevation angle was chosen randomly between each input observation with a common random seed between runs.

3.2 Model Architecture

The CNN architecture consists of three convolutional layers followed by two fully connected dense layers. The first convolutional layer has eight filters and a kernel size of 1*32*128. The activation function for the first convolutional layer is an exponential linear unit to accommodate negative reflectivity values. The second convolutional layer has 32 filters and a kernel size of 1*16*64. The third convolutional layer has 128 filters and a kernel size of 1*8*32. These both had rectified linear unit activation functions to fit piecewise linear functions to the input data to accommodate the irregularities of the input data. The number of filters increased with each layer to increase the level of abstraction, and the kernel size was lowered to accommodate the limit on the number of parameters the model could store.

The first dense layer has 256 nodes with a tanh activation function to standardize the outputs between -1 and 1, and the final layer has an output space equal to a single elevation scan of 1*1*360*1556 with a linear activation function. See figure 6 for a layout of the layers.

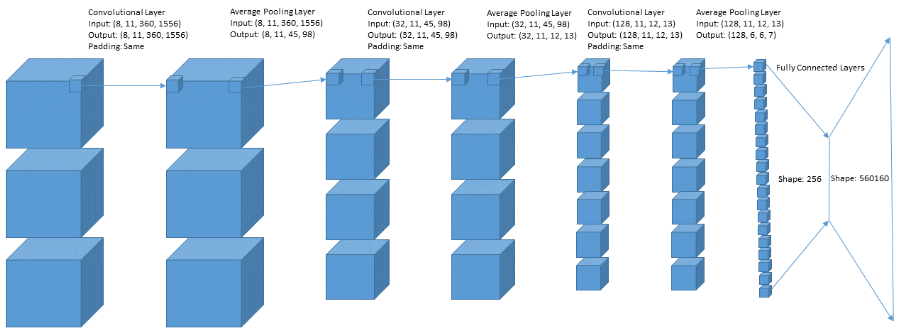


Figure 6. CNN Architecture

The outputs are normalized between layers by subtracting the mean and dividing by the standard deviation to avoid covariance shift, which could cause overfitting or introduce computational complexity, and the kernel weights are average pooled after each convolutional layer into summary statistics for the next layer to reduce the dimensionality and increase computational power.

3.3 Model Evaluation

The trained CNN model with the lowest validation set mean squared error will be used for test set comparison with the base model. Models will be compared on mean error,

$$ME = \Sigma_{i=1}^{n}\Big(\frac{y_i - \hat{y}_i}{n}\Big) \tag{4}$$

mean squared error (MSE),

$$MSE = \Sigma_{i=1}^{n}\Big(\frac{(y_i - \hat{y}_i)^2}{n}\Big) \tag{5}$$

mean absolute error (MAE),

$$MAE = \Sigma_{i=1}^{n}\Big(\frac{|y_i - \hat{y}_i|}{n}\Big) \tag{6}$$

maximum absolute error,

$$MaxAE = \max_{i}\{y_i - \hat{y}_i\} \tag{7}$$

and R-Squared,

$$R^2 = 1 - \frac{SS_e}{SS_t} = 1 - \Sigma_{i=1}^{n}\Big(\frac{(y_i - \hat{y}_i)^2}{(y_i - \bar{y})^2}\Big) \tag{8}$$

where n is the number of observations, y_i is the true value of the observation, $\hat{y}_i$ is the predicted value of the observation, SS_e is the sum of the squared error between the predicted and true observation values, and the SS_t is the total sum of all squared differences between the true values and their mean.

A mean error near at or near zero indicates that the predictions should not be substantially skewed to over or under estimate the true value of the observations. A low MSE indicates that there is little variability in the difference between the predicted values and the observed values, and is of particular note because predictions that are further from the true value will have more impact on the value of the MSE, which is valuable when the precision of characterization is important to penalize more for being further from the actual shape. The MAE is important because it characterizes a one-to-one scaling of error in relation to each other. The maximum absolute error indicates an upper limit of how far off a prediction may lie from the true value. The R-Squared is the best measure of accuracy because it represents how much variability is explained by the model.

3.4 Hardware and Software

All model training and evaluation was conducted on a Windows 10 Pro for Workstations enabled computer with a Intel(R) Xeon(R) Gold 5120 CPU, 512 GB RAM, and NVIDIA Quadrio GP100 GPU, as well as Python 3.7 packages Tensorflow-GPU 1.13.1, Keras 2.2.4, CUDA 10.1, and CuDNN 7.6.0, with all necessary dependencies.

IV. Results and Analysis

4.1 Overview

The proposed model out-performed the existing model on all recorded statistical measures for the test set.

Table 1. Summary Statistics

	Mean Error	MSE	MAE	Max Absolute Error	R Squared
Base Model	-0.8048	62.56	3.541	61.50	0.4769
CNN Model	-0.1513	33.26	2.18	31.00	0.7219

4.2 Analysis

The mean error of the base model tends to over-estimate the reflectivity values, as expected. The CNN model also tends to overestimate, but at an average of less than one-fifth of the magnitude exhibited by the base model. The reduction of MSE using results from the CNN model to nearly half shows a substantial reduction in the smoothing problem since the predictions fit the actual values much closer. This continues to be the trend when measuring the MAE, so even when the weights of large errors and small errors are equal the CNN model continues to have nearly half the amount of error. Since precipitation commonly occurs at reflectivity values around or above 20 dBZ, a maximum error of 31.00 dBZ is relatively large as an upper bound, but there are obvious improvements from the maximum observed error of 61.50 dBZ from the base model, which displays a direct improvement in the magnitude of the over-estimated values. Finally, the improvement of the R-Squared by nearly 25% shows that much more of the variability in the observed values is explained by the CNN.

4.3 Training the Model

Figure 7 shows the training performance of the CNN prior to testing against the base model. The training MSE shows general improvement across epochs, the time required to perform a full training cycle that updates the function weights, with the validation MSE following the trend. The fact that the validation MSE does not diverge substantially from the training indicates that the model is actually training to explain a general framework of the input features, and is not simply trying to overfit and predict the outputs. This means the model should perform consistently just as well from new, independent observations from the training, with similar performance characteristics.

The primary consideration from figure 7 is that the sporadic performance of the model across training epochs indicates room for improvements in the model structure. This means the model likely has room for hyperparameter tuning, which are the user defined parameters that are not updated by the model training. Most likely the learning rate, or the scaling of the gradient for the weight parameter updates, is too large, which causes the weights to update too dramatically. Essentially, the updates overshoot the optimal weight values and converges to a suboptimal solution. A lower learning rate would require longer training but is more likely to have a stable gradient descent during backpropagation. A good alternative is to mix approaches so that the learning rate can reduce between epochs, so that the gradient descent will not get stuck early on by a low learning rate, but it reduces in later epochs so that it will not overshoot the optimal or a nearly optimal solution. The learning rate used was the default value of 0.001. Other possible hyperparameters to tune include experimenting with different numbers of layers, kernel sizes, number of filters, or activation functions. The effects of changing those may have dramatic, unpredictable effects because they change the inherent nature of the model.

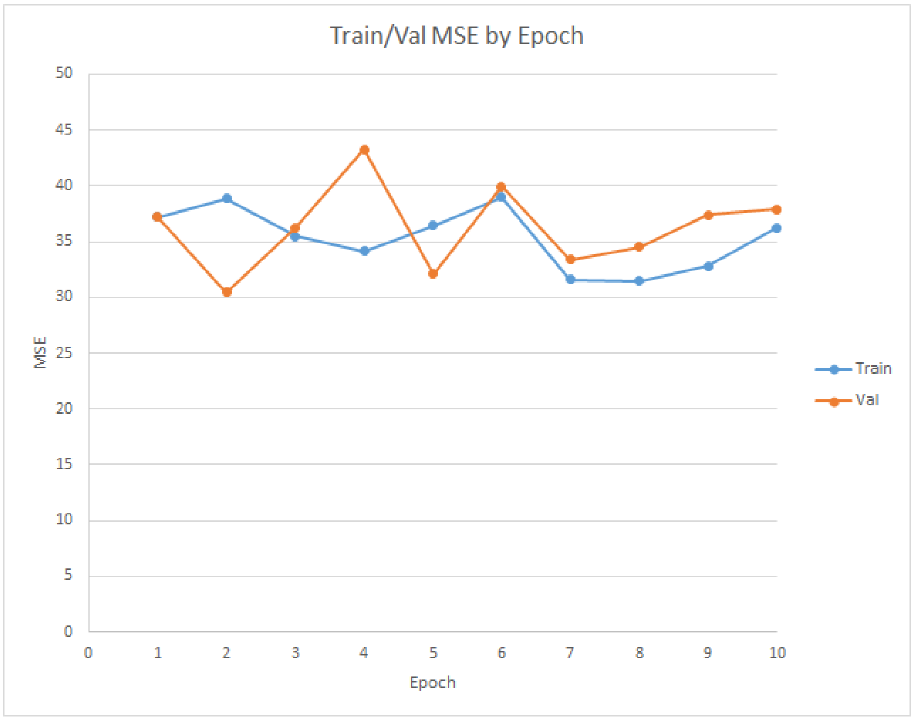


Figure 7. MSE by Epoch

4.4 Limitations

This research has been heavily constrained by hardware limitations. Data storage has been the biggest issue, where one hour of unprocessed radar scans can be in the upwards of 200 MB in size, and less than one week of processed scans can take up over 100 GB of hard drive space. The data arrays are very large, and require close to 10 GB of dedicated GPU memory and 90 minutes of training time per epoch with the current settings to perform mathematical operations. Hyperparameter tuning and model architecture restructuring can not only crash the GPU memory if the number of function weight parameters increase, but may also slow training with no guarantee of prediction improvements. These all result in problems where additional models are time intensive to build with no performance guarantee, and saving the prediction errors increases the data storage requirements.

The most straightforward solution to the hardware limitations is to improve the hardware, especially the data storage. Increasing data storage is particularly important to save the distributions in error between multiple models. This would enable the assessment of the model's bias to over or under predict radar scans, as well as if there are any areas where the predictions are consistently bad or random large errors unbalance the overall error estimate. This could also make room to save the distribution of network weight updates to observe the characteristics of the network, such as high activity nodes or even nodes that die by having their weights go to zero, which would help provide insight into a more efficient architecture or observe the sensitivity to changes in activation functions or number of nodes. Another solution is to use image downsampling to reduce the size of the input arrays. This would allow an increase in layer complexity since fewer parameters would be needed, but this would place an unknown limit on the prediction accuracy due to the loss of small scale data.

If hardware is not a limiting factor then more data may provide more insight. Image upsampling with super-resolution radar images would provide finer detail scans that can be used for truth data. Using cloud scans from throughout the year, as well as cloud scans in other geographical areas, would test the model's ability to generalize. The current model is built with the assumption that radar scans are independent from each other, but alternative model frameworks should be explored that can account for the time aspect, whether the immediate previous radar scan, or the seasonality of scans.

One final consideration is that ground WR has a scan threshold of 18.5 dBZ for echo top readings due to diminishing returns with radar power frequency bands, which is a low priority for most precipitation based predictions, but cloud bodies exist at lower dBZ values for overall shape, a measurement important for the LLCC.

Higher frequency radar scans that captured the entire cloud body may improve true prediction precision.

V. Conclusion and Future Work

5.1 Conclusion

A CNN model will likely reduce the number of delayed or canceled rocket launches due to fewer LLCC rule violations since the CNN model is less likely to over predict cloud reflectivity values, which could range in a cost savings between \$150,000 to \$1,000,000 for even a single recovered launch depending on the vehicle[6]. The variance in predictions for the CNN model is also lower, so it is likely the predictions will consistently avoid the over-smoothing issue from the current method that leads to so many LLCC rule violations that result in delayed rocket launches. The improved accuracy of the cloud characterization will also provide additional insight into the weather properties as a general product for the weather community.

5.2 Future Work

This research was constrained to purely consider the quantitative values associated with cloud shape characterization (reflectivity). Future work should also consider qualitative cloud characteristics to better distinguish between bodies for predictive purposes. This could be implemented through introducing a new branch in the neural network that separately categorizes the cloud characteristics and adds that information as part of the interpolation process. Not only will this help inform predictions by accounting for commonalities within groups such as cumulus clouds or anvil top clouds, for example, but the missing elevation scans are a characteristics themselves due to the VCP. VCP's, which utilize the context of the current weather to determine elevation scans, indicate the severity of the precipitation present, which can be used to inform predictions.

Another thing to consider is a framework that incorporates the temporal aspect of the scans, such as time of day or season. Even moreso important is that, though treating the WR scans as independent images performs relatively well, they are not truly independent. As the clouds are scanned they propagate through the atmosphere. This is similar to viewing a moving object through a picket fence - though at any one time the whole image is not visible, the entirety of the object will be revealed within incremental observations. This information can be used to improve the interpolation as the cloud moves through space.

Finally, the scope of the radar scans should be increased. Not only can the ground WR scan gap interpolation be greater generalized by utilizing scans from different geographical regions, radar scans themselves are not limited to weather. As contextual information is introduced into the model framework the network can be trained to account for non-weather related objects, and as the network gains an understanding of the properties of these foreign objects it should be trained to account for, and even interpolate the shape and movement of the objects. This would lead to a true framework that can generalize three-dimensional radar images based on context and characteristics, as well as object permanency as it moves through space in time.

Appendix A. Python Code

```
import tensorflow as tf
import matplotlib.pyplot as plt
import netCDF4 as nc
import numpy as np

import keras as ks
from keras.models import Sequential, load_model
from keras.layers import Dense, Activation, Conv3D, Flatten,
                          AveragePooling3D, BatchNormalization,
                          Dropout, Reshape
from keras import optimizers, callbacks, regularizers
from keras import backend as K
from keras.callbacks import ModelCheckpoint

import shutil
import time
import os

gpu_options = tf.GPUOptions(allow_growth=True)
session = tf.InteractiveSession(config=tf.ConfigProto(gpu_options= \
                                    gpu_options))
init = tf.global_variables_initializer()

def processData(filename,filepath):
    raw_data = nc.Dataset(filepath+filename)
```

```
refl = raw_data.variables['Reflectivity']
refl = np.lib.pad(refl,((0,11-refl.shape[0]),
                        (0,0),
                        (0,1556-refl.shape[2])
                        ),'constant')

elev = raw_data.variables['elevationR']
az = raw_data.variables['azimuthR']
dist = raw_data.variables['distanceR']

elev_m = np.tile(elev, (1556,1,1))
elev_m = np.moveaxis(elev_m, 0, -1)
elev_m = np.lib.pad(elev_m,((0,11-elev_m.shape[0]),
                            (0,0),
                            (0,1556-elev_m.shape[2])),
                            'constant')

az_m = np.tile(az, (1556,1,1))
az_m = np.moveaxis(az_m, 0, -1)
az_m = np.lib.pad(az_m,((0,11-az_m.shape[0]),
                        (0,0),
                        (0,1556-az_m.shape[2])),
                        'constant')

dist_m = np.tile(dist, (11,360,1))
```

```
    dist_m = np.lib.pad(dist_m,((0,0),
                                (0,0),
                                (0,1556-len(dist))),
                                'constant')

    imageFile = np.stack([refl,elev_m,az_m,dist_m])
    imageFile = np.ma.filled(imageFile,fill_value=0)

    return imageFile

def getFullFilePaths(directory):
    filePath = []
    for file in os.listdir(directory):
        filePath.append(os.path.abspath(os.path.join(directory, file)))

    return filePath

CLEAN_SORT_DATA = False

if CLEAN_SORT_DATA == True:
    np.random.seed(1)
    for f in os.listdir('./Raw_Data/'):
        data = processData(f,'./Raw_Data/')
        j = f.replace('.nc','')
        np.save('./Image_Files/{}.npy'.format(j),data)
    source = ''./Image_Files"
    dest1 = ''./Train_Images"
    dest2 = ''./Val_Images"
    dest3 = ''./Test_Images"
    files = os.listdir(source)
    for f in files:
        randNum = np.random.rand(1)
        if randNum < 0.7:
            shutil.move(source + '/'+ f, dest1 + '/'+ f)
        elif randNum > 0.9:
            shutil.move(source + '/'+ f, dest3 + '/'+ f)
        else:
            shutil.move(source + '/'+ f, dest2 + '/'+ f)

def maxSquareError(y_true, y_pred):
    return K.max(K.square(y_true - y_pred))

def varSquareError(y_true, y_pred):
    return K.var(K.square(y_true - y_pred))

def rSquare(y_true, y_pred):
    resSS = K.sum(K.square(y_true - y_pred))
    totSS = K.sum(K.square(y_true - K.mean(y_true)))
    return (1 - resSS/(totSS + K.epsilon()))

def meanError(y_true, y_pred):
    return K.mean(y_true - y_pred)


def graphHistory(history):
    '''Function for graphing the training and validation loss'''
    # summarize history for loss / metric
    fig = plt.figure()
    plt.plot(history.history['mse'])
    #plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.ylabel('MSE')
    plt.xlabel('Epoch')
    plt.legend(['Tr. Loss', 'Val. Loss'])
    plt.show()


def makeCNN():
    model = Sequential()
    model.add(Conv3D(8,
                    (1, 32, 128),
                    padding='same', data_format= ''channels_first",
                    input_shape=(4,11,360,1556)))
    model.add(BatchNormalization(axis=1))
    model.add(Activation('elu'))
    model.add(AveragePooling3D((1, 8, 16),
                                padding='same',
                                data_format='channels_first'))
    model.add(Conv3D(32,
                    (1, 16, 64),
                    padding='same',
                    data_format= ''channels_first"))
    model.add(BatchNormalization(axis=1))
    model.add(Activation('relu'))
    model.add(AveragePooling3D((1, 4, 8),
                                padding='same',
                                data_format='channels_first'))
    model.add(Conv3D(128,
                    (1, 8, 32),
                    padding='same',
                    data_format= ''channels_first"))
    model.add(BatchNormalization(axis=1))
    model.add(Activation('relu'))
    model.add(AveragePooling3D((2, 2, 2),
                                padding='same',
                                data_format='channels_first'))
    model.add(Flatten())
    model.add(Dense(256, activation='tanh'))
    model.add(Dense(1*1*360*1556, activation='linear'))

    model.compile(optimizer='adam',
                  loss='mse',
                  metrics=['mae',
                           maxAbsoluteError,
                           varError,
                                  rSquare,
                                  meanError])

        return model

    class DataGenerator(ks.utils.Sequence):

        def __init__(self, file_list, batch_size=4, shuffle=True):
            """Constructor can be expanded,
               with batch size, dimentation etc.
            """
            self.file_list = file_list
            self.batch_size = batch_size
            self.shuffle = shuffle
            self.on_epoch_end()

        def __len__(self):
            'Take all batches in each iteration'
            return int(len(self.file_list)/self.batch_size)

        def __getitem__(self, index):
            'Get next batch'
            # Generate indexes of the batch
            indexes = self.indexes[index:(index+1)]

            # single file
            file_list_temp = [self.file_list[k] for k in indexes]

            # Set of X and y
            X, y = self.__data_generation(file_list_temp)

            return X, y

        def on_epoch_end(self):
            'Updates indexes after each epoch'
            self.indexes = np.arange(len(self.file_list))
            if self.shuffle == True:
                np.random.shuffle(self.indexes)

        def __data_generation(self, file_list_temp):
            'Generates data containing batch_size samples'
            excluded_angles = np.random.choice(11,1,replace=False)

            X = np.empty((self.batch_size,4,11,360,1556))
            y = np.empty((self.batch_size,1*1*360*1556))
            # Generate data
            for i, f in enumerate(file_list_temp):
                data = np.load(f, allow_pickle=True)
                X[i,] = data
                X[i,0,excluded_angles,:,:] = 99999
                y[i,] = data[0,excluded_angles,:,:].flatten()
            return X, y
```

```
model = makeCNN()
model.summary()

trainDir = ``./Train_Images"
valDir = ``./Val_Images"
testDir = ``./Test_Images"

trainImages = getFullFilePaths(trainDir)
valImages = getFullFilePaths(valDir)
testImages = getFullFilePaths(testDir)

trainGen = DataGenerator(trainImages)
valGen = DataGenerator(valImages)
testGen = DataGenerator(testImages)

TRAIN_MODEL = False

if TRAIN_MODEL == True:
    np.random.seed(52)
    hist = model.fit_generator(generator=trainGen,
                               epochs=10,
                               verbose=1,
                               validation_data=valGen,
                               max_queue_size=8,
                               callbacks=[ModelCheckpoint(``./model.h5"
                                       save_best_only=True,
                                       monitor='val_loss',
                                       mode='min')])#,
                                       history])

bestModel = ks.models.load_model(''./model.h5",
                                     custom_objects={
                                     'maxAbsoluteError': maxAbsoluteError,
                                     'varError': varError,
                                     'rSquare': rSquare,
                                     'meanError': meanError
                                     })

graphHistory(hist)

np.random.seed(52)
elevation_angles = [0,1,2,3,4,5,6,7,8,9,10]
totSquareError = 0
totAbsoluteError = 0
testMaxAbEr = 0
totSS = 0
totSumError = 0
i = 0
h = 0

for f in testImages:
```

```
data = cp.load(f, allow_pickle=True)
excluded_angles = cp.random.choice(11,1,replace=False)
for k in excluded_angles:
    if (k - 1 < 0) or (10 < k + 1):
        testPredict = cp.zeros_like(data[0,k,:,:])
        totSS = totSS + cp.sum(cp.square(data[0,k,:,:] - \
                cp.mean(data[0,k,:,:])))
    elif (data[0,k+1,:,:] != cp.zeros_like(data[0,k,:,:])).any():
        testPredict = cp.divide(cp.add(data[0,k+1,:,:],
                                data[0,k-1,:,:]),
                                2)
        totSS = totSS + cp.sum(cp.square(data[0,k,:,:] - \
                cp.mean(data[0,k,:,:])))
    elif (data[0,k-1,0,0] == cp.zeros_like(data[0,k,:,:])).all():
        testPredict = cp.zeros_like(data[0,k,:,:])
        totSS = totSS + cp.sum(cp.square(data[0,k,:,:] - \
                cp.mean(data[0,k,:,:])))
    else:
        testPredict = cp.divide(cp.add(data[0,k+1,:,:],
                                data[0,k-1,:,:]),
                                2)
        totSS = totSS + cp.sum(cp.square(data[0,k,:,:] - \
                cp.mean(data[0,k,:,:])))
    testError = data[0,k,:,:] - testPredict
    i = i + np.prod(testError.shape)
    squareError = cp.square(testError)
```

```
            testMaxAbEr = max(cp.max(cp.abs(testError)),testMaxAbEr)
            totAbsoluteError = totAbsoluteError + cp.sum(np.abs(testError))
            totSquareError = totSquareError + cp.sum(squareError)
            totSumError = totSumError + cp.sum(testError)
        h += 1
        print(``Finished step ", h)

testMSE = totSquareError/i
testMAE = totAbsoluteError/i
testMeanError = totSumError/i
testVar = testMSE - np.square(testMeanError)
testRSq = 1 - totSquareError/totSS

np.random.seed(52)
testCNN = bestModel.evaluate_generator(testGen)

print(``The test MSE using averages is ", testMSE,
            `` and the MSE using the CNN is ", testCNN[0], ``.")
print(``The test MAE using averages is ", testMAE, `` and the MAE using th
print(``The test mean error using averages is ", testMeanError,
            " and the mean using the CNN is ", testCNN[5], ``.")
print(``The test variance using averages is ", testVar,
            " and the variance using the CNN is ", testCNN[3], ``.")
print(``The test R^2 using averages is ", testRSq,
            " and the R^2 using the CNN is ", testCNN[4], ``.")
print(``The test max absolute error using averages is ", testMaxAbEr,
" and the max using the CNN is ", testCNN[2], ''.")
```

Bibliography

1. W. Roeder and D. Short, "The new weather radar for americas space program in florida: Scan strategy design," 2009. 45th Weather Squadron, Patrick AFB, FL.

2. S. Sharma, "Activation functions in neural networks." `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6/`, September 2016.

3. "About us." `https://www.patrick.af.mil/About-Us/Weather/`, May 2019.

4. NASA, "NASA standard for lightning launch commit criteria for space flight," June 2017. NASA STD 4010.

5. A. Augst and M. Hagen, "Interpolation of operational radar data to a regular cartesian grid exemplified by munichs airport radar configuration," *Journal of Atmospheric and Oceanic Technology*, vol. 34, pp. 495–510, March 2017.

6. W. Roeder and T. McNamara, "A survey of the lightning launch commit criteria," 2005. 45th Weather Squadron, Patrick AFB, FL.

7. NOAA, "Radar data." `https://www.ncdc.noaa.gov/data-access/radar-data`, June 2019. National Centers for Environmental Information.

8. NOAA, "NEXRAD." `https://www.ncdc.noaa.gov/data-access/radar-data/nexrad`, June 2019. National Centers for Environmental Information.

9. Office of the Federal Coordinator for Meteorological Services and Supporting Research, "Federal Meteorological Handbook No. 11 Wsr-88D Meteorological Observations Part C Wsr-88D Products and Algorithms," *Fcm-H11C-2017*, no. 11, 2017.

10. V. Lakshmanan, K. Hondl, C. Potvin, and D. Preignitz, "An improved method for estimating radar echo-top height," *Weather and Forecasting*, vol. 28, April 2013.

11. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

12. Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, "Image inpainting via generative multi-column convolutional neural networks," in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, (USA), pp. 329–338, Curran Associates Inc., 2018.

13. D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," *CVPR*, pp. 2536–2544, 2016.

14. I. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, Y. Bengio, "Generative adversarial nets," *University of Montreal, Dept of Operations Research, Montreal, Canada*, pp. 1–9, 2014.

15. E. Shi, Q. Li, D. Gu, Z. Zhao, "A method of weather radar echo extrapolation based on convolutional neural networks," *Lecture Notes in Computer Science*, vol. 10704, pp. 16–28, January 2018.

16. M. Choma, "Extrapolation of radar echo with neural networks." `https://medium.com/pocasi/extrapolation-of-radar-echo-with-neural-networks-f87772f70db2`, August 2019.

17. K. Armanious, S. Abdulatif, F. Aziz, U. Schneider, B. Yang, "An adversarial super-resolution remedy for radar design trade-offs," *EUSIPCO*, 2019.

18. S. Ansari, "Noaa's weather and climate toolkit." `https://www.ncdc.noaa.gov/wct/index.php`, January 2020.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 26–03–2020	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2018 — Mar 2020

4. TITLE AND SUBTITLE Ground Weather RADAR Signal Characterization through Application of Convolutional Neural Networks	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Lee, Stephen, M., 1st Lt	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENS-MS-20-M-158
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) William P. Roeder, 45 WS/SYR, 1201 E. H. White II St., MS 7302, Patrick AFB, FL 32923-3238, COMM 321-853-8410, DSN 467, Email: william.roeder@us.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** 45 WS/SYR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The 45th Weather Squadron supports the space launch efforts out of the Kennedy Space Center and Cape Canaveral Air Force Station for the Department of Defense,NASA, and commercial customers through weather assessments. Their assessment of the Lightning Launch Commit Criteria (LLCC) for avoidance of natural and rocket triggered lightning to launch vehicles is critical in approving space shuttle and rocket launches. The LLCC includes standards for cloud formations, which requires proper cloud identification and characterization methods. Accurate reflectivity measurements for ground weather radar are important to meet the LLCC for rocket triggered lightning. Current linear interpolation methods for ground weather radar gaps result in over-smoothing of the vertical gradient and over-estimate the risk of rocket triggered lightning, potentially resulting in costly, unnecessarily delayed launches. This research explores the application of existing interpolation methods using convolutional neural networks to perform two-dimensional image interpolation, called inpainting, into the three-dimensional weather radar scan domain. Results demonstrate that convolutional neural networks can improve the accuracy of cloud characterization over current interpolation methods, potentially resulting in fewer launch delays with substantial associated cost savings due to increased capability to meet the LLCC.

15. SUBJECT TERMS

CNN, weather, interpolation, rocket, lightning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Lance. E. Champagne, AFIT/ENS
U	U	U	UU	45	**19b. TELEPHONE NUMBER** *(include area code)* 937-255-3636, x4646;lance.champagne@afit.edu