UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

AUTOMATED LOCATION OF BIRD ROOSTS USING NEXRAD DATA AND IMAGE SEGMENTATION

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

KATHERINE E. AVERY

Norman, Oklahoma

2020

AUTOMATED LOCATION OF BIRD ROOSTS USING NEXRAD DATA
AND IMAGE SEGMENTATION

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Acknowledgements

This work was supported by the grant NSF-DGE–1545261: NRT: Aeroecology as a Test-bed for Interdisciplinary STEM Training.

The computing for this project was performed in part at the OU Supercomputing Center for Education and Research (OSCER) at the University of Oklahoma (OU) and in part on a Microsoft Azure machine, courtesy of an AI for Earth grant.

Special thanks to my advisor, Dr. Amy McGovern, for helping navigate obstacles, technical or otherwise, and providing crucial guidance and expertise. Thank you to Dr. Andrew H. Fagg and Dr. Eli Bridge for being on my committee and giving advice and feedback.

I would like to acknowledge Amy McGovern, Katy Felkner, Grant Eckstein, Amanda Burke, David Harrison, and Ryan Lagerquist for helping to correct data labels.

I greatly appreciate Duncan Walter for both providing emotional support throughout this project and answering questions when my project took an unexpected detour into frontend development.

Finally, thank you to Robert Daro and Eli Jergensen for their friendship and ongoing support.

Table of Contents

List Of Figures

Abstract

Weather surveillance radars can effectively detect flying animals, such as groups of birds, bats, and insects. Further, these radars are demonstrably useful for detecting the existence of certain bird roosting locations, particularly those of many species of swallows. The roosts appear on radar as distinctive rings of high reflectivity. Detecting and locating bird roosts have a variety of applications from ecological conservation to wind farm placement and air traffic control.

In this thesis, I first detect the presence of a roost in NEXt Generation Weather RADar (NEXRAD) images of purple martin and tree swallow roosts and improve upon Chilson et al. (2019)'s work by altering the network architecture and data preprocessing. Because determining the exact locations of bird roosts is more useful than detection, I also use image segmentation to attempt to locate roosts. I apply a standard U-Net, which shows promising results for determining roost location, achieving a true positive rate of around 0.80 and true negative rate of around 0.85. To increase performance, the dataset is augmented 32 times its original size through a variety of image transformations.

Chapter 1

Introduction

Locating and tracking bird roost locations is important for biological conservation, research, citizen science, and agriculture, among other applications (Bauer et al. 2017b). For example, knowing roost locations can help inform decisions about wind turbine placement and the use of bird-safe glass in windows. According to Johnson et al. (2002), over 100 million birds run into man-made objects and about 3.5 million birds migrate over wind farms in the United States every year. Birds can also cause problems for farms and local ecology by eating crops and spreading diseases among environments, or they can have positive agricultural and environmental impacts by eating insect pests and pollinating plants (Bauer and Hoye 2014; Bauer et al. 2017b).

The network of weather surveillance radars known as "NEXRAD" is designed to detect weather across the United States, but it can also be used to detect biological phenomena (Bauer et al. 2017a; Chilson et al. 2012a; Gauthreaux Jr. and Belser 2003; Kelly et al. 2012; Chowdhury et al. 2016; Stepanian and Horton 2015). Weather surveillance radars are particularly useful for finding and tracking roosts because there are already 160 radars scattered across the United States in NEXRAD, and the roosts of some species create a "roost ring" – a distinctive ring of heightened reflectivity on NEXRAD (Kelly and Pletschet 2018). A roost ring forms on radar when a flock of birds leaves its nest for the morning, typically slightly before sunrise. This ring can be used to pinpoint the location of a roost. An example of a roost ring on several fields of NEXRAD is shown in Figure 1.1. Manually searching for roosts on raw radar images is time-consuming, suggesting the need for a technological approach to analyze the raw, non-quality-controlled NEXRAD output (Chilson et al. 2012b). Chilson et al. (2019) found that deep learning is effective for roost detection on 2D, RGB radar images.

Currently, deep learning, specifically convolutional neural networks (CNNs), is the most effective way to automatically locate objects in images (Zhao et al. 2018). Since Girshick et al. (2013) created Region-CNN (R-CNN), a variety of architectures have been created for object detection and segmentation, including Fast R-CNN, Faster R-CNN, Mask R-CNN, the You Only Look Once (YOLO) network, and U-Net (Girshick 2015; Ren et al. 2015; He et al. 2017; Redmon et al. 2015; Ronneberger et al. 2015). These architectures have been effective at segmenting images similar to the radar images in this work's dataset.

This thesis is an extension of Chilson et al. (2019), and it explores the usefulness of CNNs for a) detecting roosts, referred to as the *detection problem*, and b) roughly determining roosts' latitude and longitude locations on level-two NEXRAD images, referred to as the *localization problem*. To solve the detection problem, this work seeks to predict the presence or absence of a roost, while for the localization problem, it attempts to estimate locations of roosts in an image with image segmentation. A shallow CNN architecture similar to the one in Chilson et al. (2019) is used for detection. For localization, this thesis uses a U-Net, a common architecture for image segmentation (Ronneberger et al. 2015).

The dataset consists of purple martin and tree swallow roosts in the Eastern half of the United States. To see if the tool is useful for discovering new roost locations, this thesis applies the localization method to an unlabeled set of purple martin roosts in Indiana from 2018 and 2019. Because the Indiana dataset is unlabeled, statistics must be calculated on a separate, labeled test set. The Indiana dataset is useful for qualitatively determining how well the method may perform on unprocessed data.

The next chapter provides relevant background on bird roosts, NEXRAD radars, and neural networks, as well as a discussion of related works. Chapter 3 describes the dataset used in-depth, while Chapter 4 discusses methods and experimental design, including neural network architectures. Chapter 5 shares and discusses results, while the final chapter provides a conclusion, suggested improvements, and future work.

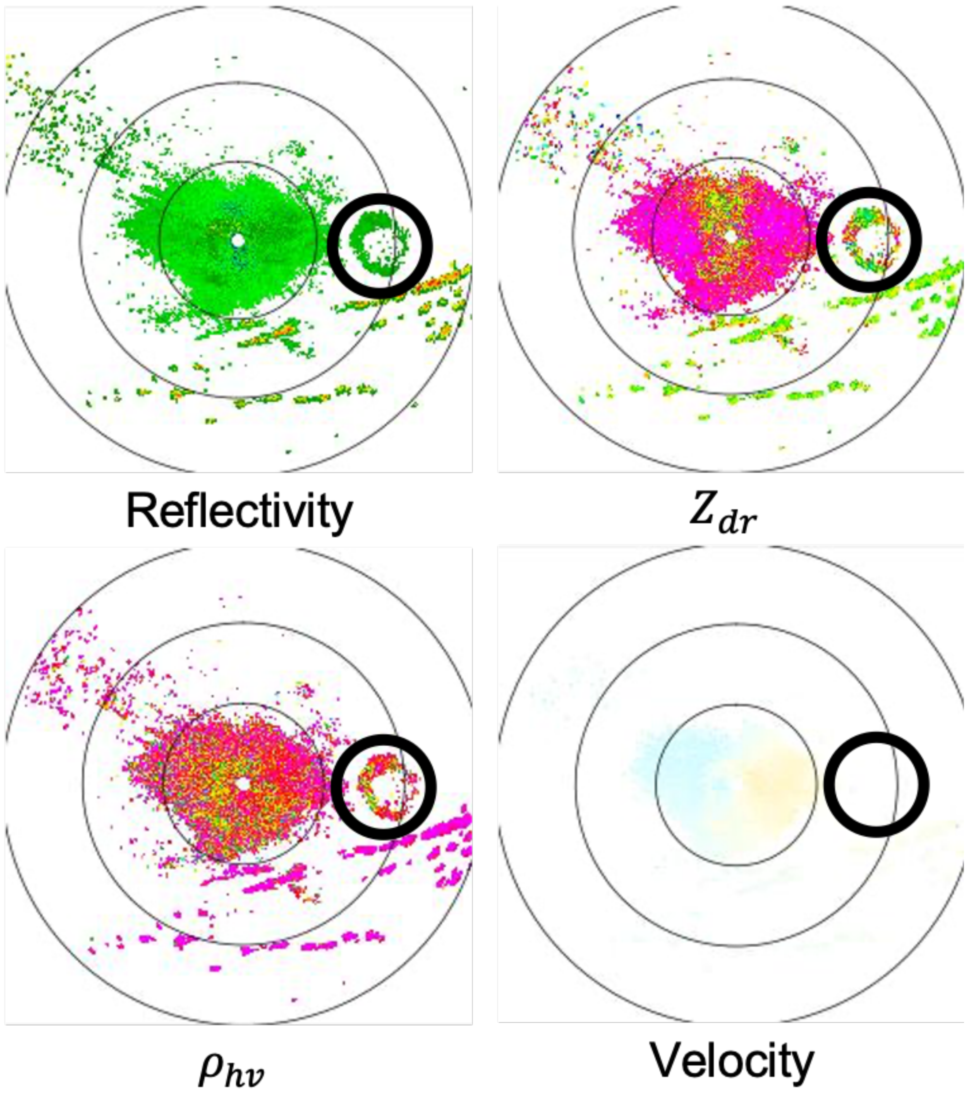

Figure 1.1: An example roost is circled in black on images of four different NEXRAD fields from July 16, 2015. These images were created with the Py-ART library (Helmus and Collis 2016).

Chapter 2

Background

2.1 NEXRAD for Roost Location

Since the early 1990s, NEXRAD has scanned the bottom 10 km of the atmosphere every five to ten minutes (Chilson et al. 2012a). Purple martin roosts are typically visible on level-2 NEXRAD from late June to late August, and roosts have been confirmed at 80 different radars in the eastern half of the United States (Kelly and Pletschet 2018). Most roosts are found about 20 minutes before and 40 minutes after sunrise (Kelly and Pletschet 2018), though this thesis also includes 30 minutes before and after this period, which totals to about 24 radar scans per day. Refer to Chapter 4 for more details.

In addition to birds and weather, radars can detect noise like dust and insects, which are much more common in the lowest parts of the atmosphere. Since radars generate beams that scan upwards at an angle and the Earth curves away, the scans closest to the radar capture much more noise. This means that the center of the radar images in the dataset for this thesis are much noisier than the perimeters of the images.

The NEXRAD images for this thesis come from both single- and dual-polarization radar. Single-polarization radar transmits and receives waves that have horizontal polarization, while dual-polarization radar have both vertical and horizontal polarization. Single-polarization radar includes the reflectivity, Doppler radial velocity,[1] and spectrum width fields, while dual-polarization radar includes these products, along with the differential reflectivity (Z_{DR}), correlation coefficient (ρ_{HV}), and differential phase (ϕ_{DP}) fields. In this thesis, we use only reflectivity, velocity, Z_{DR}, and ρ_{HV} because they have been shown useful for locating roosts (Muller et al. 2015).

Reflectivity measures the intensity of the beam reflected back to the radar after bouncing off a flock of birds (Diehl and Larkin 2005). Velocity is used to determine the direction that the birds are flying, which is important because birds to fly outwardly from their nest, while weather and dust fly in one direction (Gauthreaux Jr. and Belser 2003). ρ_{HV} measures the similarity between the horizontal and vertical polarizations of the radar beam. Birds tend to have a lower ρ_{HV} than airborne water droplets (Van Den Broeke 2013). Z_{DR} is the difference in reflectivity between the horizontal and vertical polarizations, which has been shown useful for determining bird orientation (Stepanian and Horton 2015).

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) are layers of connected nodes that can approximate complicated, nonlinear relationships in data (Rumelhart et al. 1986; Dayhoff and DeLeo 2001). They consist of an input layer, some number of hidden (i.e., middle) layers, and an output layer, as shown in Figure 2.1.

Each node (or blue circle in Figure 2.1) takes inputs, multiplies them by a set of "weights" and adds a "bias," as shown in Figure 2.2. Since the output

[1]For the remainder of this thesis, we refer to "Doppler radial velocity" as simply "velocity."

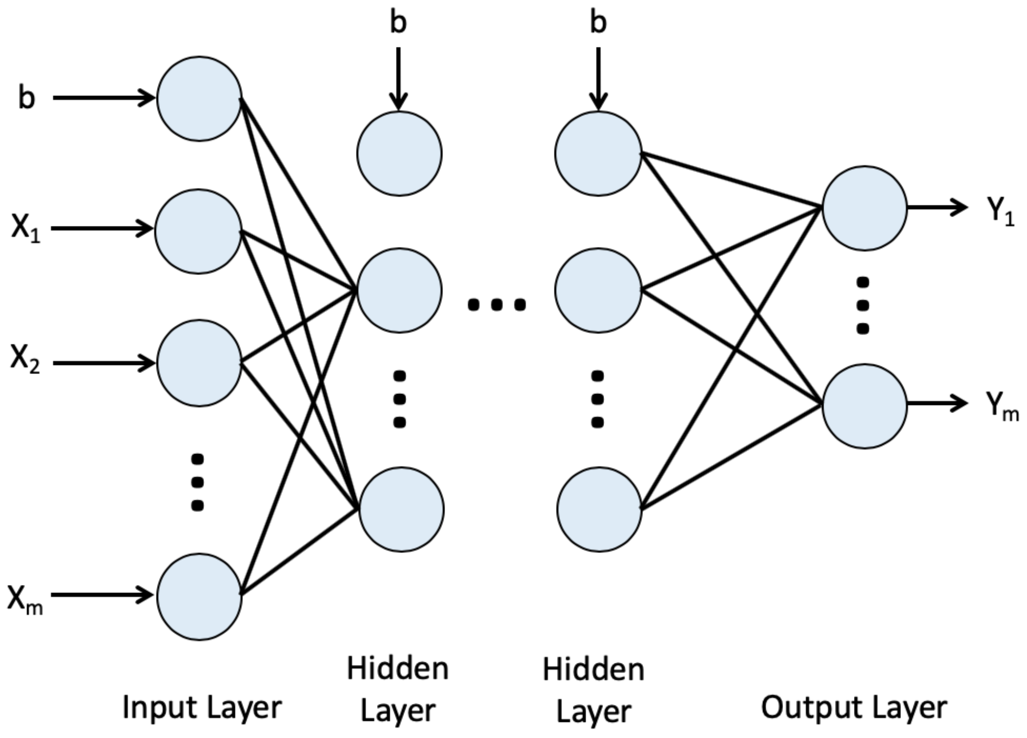

Figure 2.1: An example of a feedforward neural network. X_1 to X_m are the inputs, while Y_1 to Y_m are the outputs, and b is the bias.

of a single neuron can be represented linearly, the weights increase or decrease the effect of an input variable. The weights are calculated and updated via the backpropagation algorithm described in the following section (Lecun 1985; Rumelhart et al. 1986; Werbos 1990).

After summing the weighted inputs and the bias, the output z is passed to an activation function σ. This function decides whether or not the node is "activated" at a given time and introduces nonlinearity into the neural network, allowing it to model a wide range of functions. The bias shifts the activation function to the left or right of the origin to get a better fit for the data (Raudys 1998). There is a variety of possible activation functions, but we shall limit our discussion to the ones used in the networks described in Chapter 4.

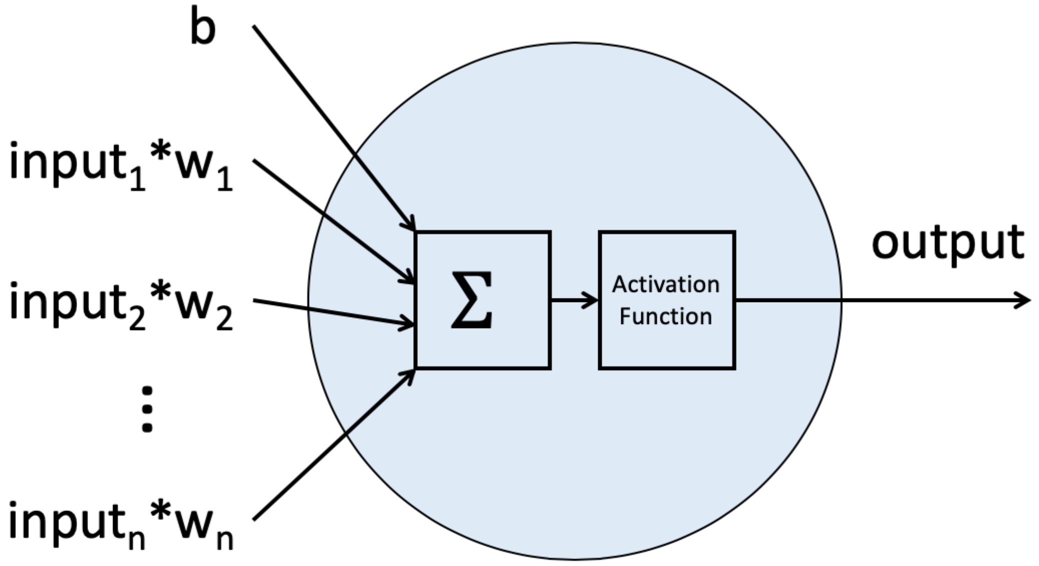

Figure 2.2: An example node. Inputs are multiplied by weights, and the products are summed together, along with the bias. An activation function is applied the output of the sum to obtain the final output.

One of the earliest activation functions is the sigmoid function, which bounds its outputs between zero and one and is typically used for regression. The sigmoid function can be represented by the following equation (Jain et al. 1996):

$$\sigma(z) = \frac{1}{1+e^{-z}}. \tag{2.1}$$

The softmax function, like sigmoid, returns values between zero and one, but the output values always sum to one and are often interpreted as probabilities for classification. For example, in binary classification, softmax may return 0.7 for the most probable class and 0.3 for the other class. The softmax function is represented by the equation

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{m} e^{z_k}}, \tag{2.2}$$

where m is the number of classes, and j is a position in the input z to the activation function (Goodfellow et al. 2016).

The Rectified Linear Unit (ReLU) activation function returns zero for values less than or equal to zero and a positive value otherwise:

$$\sigma(z) = max(z, 0). \tag{2.3}$$

ReLU is the standard activation function for hidden layers because it is inexpensive to compute. The function also sets many values to zero, which makes models sparser and more expressive, resulting in less overfitting. The downside of ReLU is that it can sometimes make models too sparse and result in a vanishing gradient (Krizhevsky et al. 2012).

2.2.1 Backpropagation Algorithm

The process of passing data from the input layer to the output layer described in the previous section is called "forward propagation." Introduced in 1985, backpropagation (or the backwards propagation of errors) updates the weights to minimize the loss function (Lecun 1985; Rumelhart et al. 1986; Werbos 1990). The weights are updated via backpropagation from the last layer of the network to the first layer. The data is then passed back through the network via forward propagation, yielding a new (ideally smaller) error value, which is then used to update the weights, described formally by Equation 2.6. The process is repeated until the error no longer decreases.

To calculate the weight update, backpropagation uses gradient descent, an optimization algorithm that finds a local minimum for a differentiable function. In the case of backpropagation, gradient descent uses the loss function. The algorithm first finds the vector that leads to the greatest decrease in the loss function (i.e., the derivative) and then takes a step in the direction of that vector. The size of the step is determined by the learning rate and the magnitude of the gradient. Rather than taking a full derivative, we take the partial derivative with respect to the loss for each weight because the network has more than one input. Because there are multiple layers, we can use the chain rule to pass information backwards through the network (Goodfellow et al. 2016).

Two loss functions L will be discussed in this thesis. n is the number of examples, m is the number of pixels per image, y_i is the true label, and a_i is the predicted label. The binary cross-entropy loss function,

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log(a_i) + (1 - y_i) log(1 - a_i)\right] \tag{2.4}$$

is used for detection. Its prediction a_i for each image is bounded between zero and one, making it useful for classification. The function calculates the average difference between the predicted and actual probability distributions.

The second loss function used is dice loss,

$$L = 1 - \frac{2\sum_{i=1}^{m*n} a_i y_i}{\sum_{i=1}^{m*n} {a_i}^2 + \sum_{i=1}^{m*n} {y_i}^2}. \tag{2.5}$$

Dice loss is more appropriate for image segmentation and uses the dice score coefficient (DSC) (Sudre et al. 2017; Milletari et al. 2016). The numerator of the DSC is the intersection of all the pixels in the predicted segmentation and the ground truth, and the denominator is the total amount of pixels in both the predicted segmentation and the ground truth. Each pixel in the ground truth is either zero or one, and each pixel in the predicted segmentation is also either zero or one. $L = 1 - DSC$ so that the loss will maximize overlap between the set of all predicted pixels and the set of all ground truth pixels.

Formally, backpropagation for the weight w between layer j and layer k can be described by the following three equations:

$$\begin{aligned} \Delta_k &= L_k \sigma'(x_k) \\ w_{jk} &\leftarrow w_{jk} - \alpha y_j \Delta_k \\ \Delta_j &= \sigma'(x_k) \sum_k w_{jk} \Delta_k \end{aligned} \tag{2.6}$$

α represents the learning rate, x the input, y the output, L the loss, and Δ the error gradient for a layer (either k or j) (Goodfellow et al. 2016). The error gradient is a vector used to update weights to minimize error during training.

In this thesis, we use a variation of gradient descent called mini-batch gradient descent. This variation calculates the gradient for a subset of the training data and then applies the weight updates to the network (Li et al. 2014). Mini-batch gradient descent calculates an approximation rather than the true gradient, but it is much faster than standard gradient descent over the entire dataset. Though using a mini-batch is slightly more unstable than using the entire dataset, the gradients produced by the mini-batches will on average point in the direction of the true gradient (Wilson and Martinez 2003).

2.2.2 Deep Learning

Deep learning is a subset of machine learning that uses particularly deep neural networks (Goodfellow et al. 2016). Before the introduction of deep learning in 2012, the performance of neural networks with many layers suffered from the vanishing gradient problem. Because backpropagation uses the chain rule to compute the gradient, the error gradient can be multiplied by very small numbers, drawing it close to zero (Bengio et al. 1994). Because the error gradient is so small, it has little effect on the model during backpropagation.

Fukushima (1980) introduced the concept of convolutional layers – layers of neural networks that apply a mathematical operation called a convolution, which repeatedly multiplies different parts of an input matrix by a small set of weights called a filter. The outputs of these multiplications are collected in a matrix called a feature map, as illustrated in Figure 2.3. Convolution helps to solve the vanishing gradient problem for deep networks by decreasing the network's number of parameters. Because the filter size is typically much smaller than the input matrix, the number of parameters is considerably reduced (Goodfellow et al. 2016). Deep networks can also capture complex, spacial relationships, which traditional neural networks cannot learn. Convolution produces concise, descriptive feature maps because filters (i.e., sets of weights) are small and learned through training. More information about deep learning can be found in Goodfellow et al. (2016).

Networks that use convolutional layers are called convolutional neural networks (CNNs). The networks described in this thesis are all CNNs; further descriptions of the architectures used can be found in Chapter 4.

2.2.3 Coordinate Channels

CNNs are not able to accurately transform points back and forth between Cartesian space and the CNNs' underlying latent space (where the CNN encodes meaningful features). For example, a CNN trying to predict the location of a pixel with the Cartesian coordinates (100, 200) may predict the nearby location

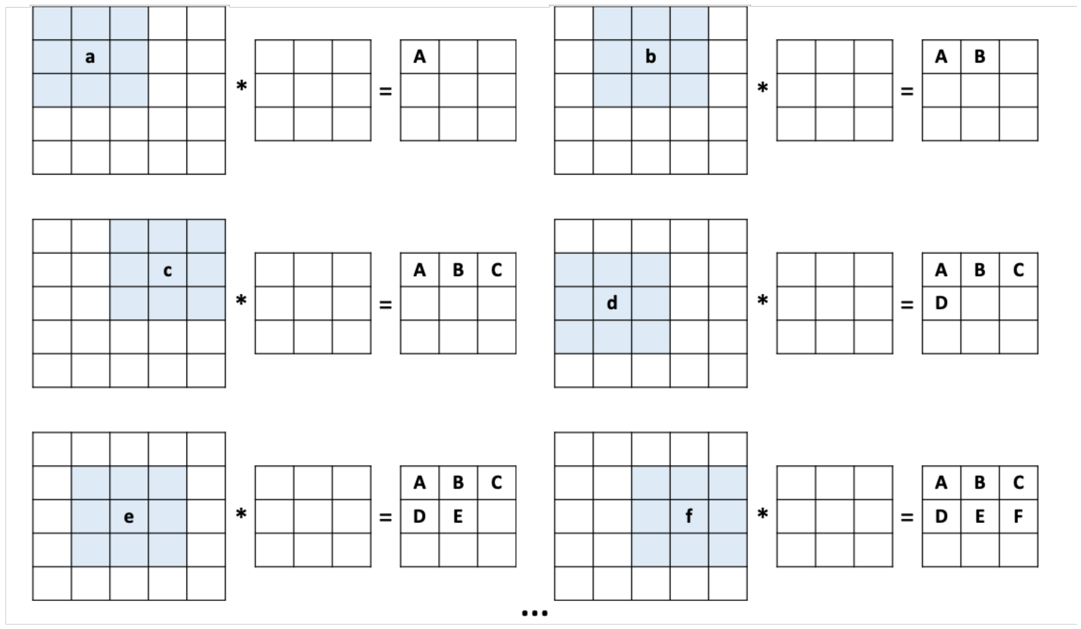

Figure 2.3: The first six steps of the convolution algorithm with a 5 × 5 input matrix and 3 × 3 filter. The shaded parts of the input matrix are multiplied by the filter to produce the 3 × 3 feature map.

(95, 210). This issue can subtly degrade performance on location problems (Liu et al. 2018).

Adding a handcrafted feature that represents Cartesian coordinates helps CNNs to calibrate themselves and improve predictions of locations. Liu et al. (2018) created a custom layer called CoordConv that adds these Cartesian coordinate channels to the convolutional layer, as illustrated in Figure 2.4. CoordConv is 150 times faster than regular convolutional layers, and it uses 10 to 100 fewer parameters. When segmenting grayscale objects on a black background with Faster R-CNN, CoordConv had a 24% improvement on the average intersection-over-union compared to convolution without CoordConv (Liu et al. 2018).

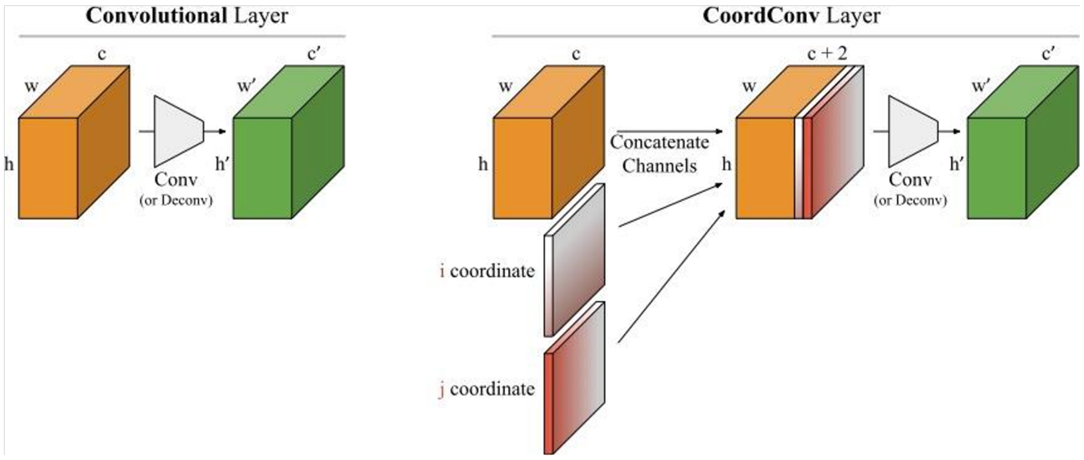

Figure 2.4: The left subfigure shows a normal convolutional layer, while the right subfigure shows a convolutional layer with CoordConv. Before convolution is applied, Cartesian coordinates i and j are concatenated to the input as channels. These extra channels are handcrafted features that aid in localization. Figure from Liu et al. (2018).

One criticism of CoordConv is that it breaks translation equivariance, an assumption of CNNs that states that the location of an object should not affect the CNN's ability to detect it. However, CoordConv does not completely dispense of this property. If the weights of the coordinates are zero (i.e. the network learns that the coordinate channels are not useful), then the CNN retains translation equivariance, but if the network finds coordinate channels useful for the given problem, it dispenses of the equivariance.

2.2.4 Batch Normalization

Batch normalization is a technique that makes CNNs converge faster and makes them more resilient to small changes in hyperparameters. The technique comes from the concept of whitening, which normalizes the outputs of layers, giving them means of zero and variances of one (Ioffe and Szegedy 2015). Because normalizing each output in this way is computationally expensive and is not always differentiable for backpropagation, the variance γ and mean β of the entire training set must be learned. Since γ and β are calculated from the entire training set, they are also more representative than calculations using smaller samples (Ioffe and Szegedy 2015). Algorithm 1 shows how batch normalization is applied to an activation x over a mini-batch. The mini-batch mean and variance are calculated to normalize x. x is then scaled with γ and shifted with β.

Currently, there is some disagreement over why batch normalization increases CNN resilience and speed. Ioffe and Szegedy (2015) asserts that neural networks suffer from internal covariate shift, which is “the change in the distribution of network activations due to the change in network parameters during training.” Because the weights of layers are updated during training, the weights of later layers are affected by those of earlier layers, causing the network’s input distribution to shift. Ioffe and Szegedy (2015) suggests that batch normalization reduces internal covariate shift by normalizing the outputs of each layer. On the other hand, Santurkar et al. (2018) suggests that batch normalization does not necessarily reduce internal covariate shift but rather smooths the optimization landscape, allowing networks to converge faster and avoid local minima.

Algorithm 2.1: Batch Normalization applied to activation x over a mini-batch. Algorithm from (Ioffe and Szegedy 2015)

```
Input  : Values of $x$ over a mini-batch $B = \{x_{1\ldots m}\}$; Parameters to be
         learned: $\gamma, \beta$
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$

  $\mu \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                         // calculate mini-batch mean $\mu$
  $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu B)^2$          // calculate mini-batch variance $\sigma_B^2$
  $\hat{x}_i \leftarrow \frac{(x_i - \mu B)^2}{\sqrt{\sigma_B^2 + \epsilon}}$                    // normalized $x_i$ is $\hat{x}_i$
  Note: $\epsilon$ is a very small constant to avoid dividing by zero.
  $y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$          // scale with $\gamma$ and shift with $\beta$
```

2.2.5 U-Net

The U-Net architecture was first developed by Ronneberger et al. (2015) for biomedical image segmentation but has since been used for other semantic segmentation applications, including agriculture, augmented and virtual reality, and chemical engineering (Zhao et al. 2019; Han et al. 2019; Zhu et al. 2019). The architecture was designed to perform semantic segmentation with a relatively low training time and few labelled images. Segmenting a 512×512 image takes less than one second on an Nvidia Titan GPU (Ronneberger et al. 2015).

The architecture, shown in Figure 2.5, is a roughly symmetrical CNN with 23 convolutional layers. The first half of the U, the "contracting path," downsamples the data while increasing the number of features, while the second half

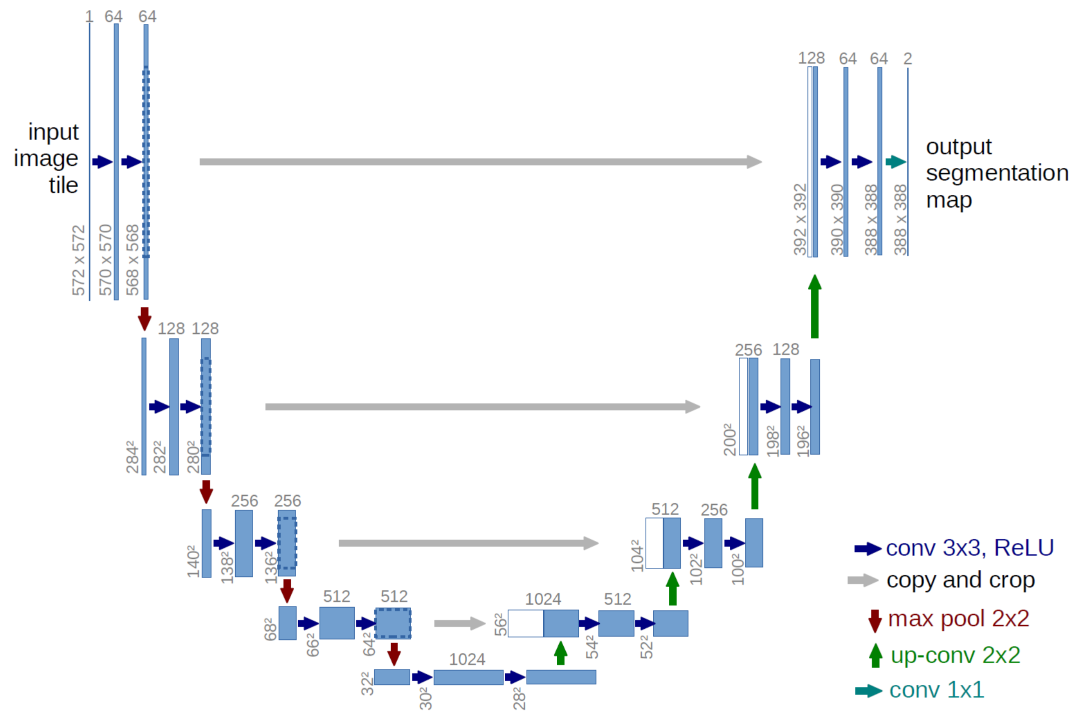

Figure 2.5: U-Net architecture. Figure from Ronneberger et al. (2015).

of the U, the "expansive path," upsamples the data and decreases the feature channels. The final output is a 2D map showing the semantic segmentation of the input image. The U-Net also includes skip connections (shown in gray in Figure 2.5), which transfers spatial information from the first half of the U to the second half since this information may have been lost during the downsampling step. These skip connections have been shown to reduce network training times (Szegedy et al. 2016a; Drozdzal et al. 2016).

2.3 Related Work

This work builds directly on that of Chilson et al. (2019), which seeks to detect the presence or absence of roosts on NEXRAD images with CNNs. This thesis is also very related to Lin et al. (2019), which semantically segments NEXRAD images to find flocks of birds. As such, we discuss these works in detail. For completeness and comparison, we also discuss earlier methods that use radar hardware for bird detection and localization.

2.3.1 Roost Detection

Chilson et al. (2019) detects purple martin and tree swallow roosts on NEXRAD images and attempts to solve the same detection problem as this thesis. The primary difference from this work is that Chilson et al. (2019) focuses only on the presence or absence of a roost, while this work also focuses on the location of the roost in the image. Chilson et al. (2019) also uses a slightly different dataset than the one described in Chapter 3, though some examples are shared between the two sets, particularly in Texas and along the southeastern coast of the United States. Additionally, the detection architecture described in Chapter 4 is slightly different from the architecture in Chilson et al. (2019).

Chilson compared three networks: a simple artificial neural network without convolution, a shallow CNN, and the Inception-v3 network (Szegedy et al. 2016b). Each network classified the images of the radar products separately, and these classifications were then passed to a dense layer for aggregation. Because of the small dataset, Inception-v3 was trained using transfer learning on ImageNet (Deng et al. 2009).

The shallow CNN tested had an area under curve (AUC) of 0.930-0.961 for single-polarization radar and 0.931-0.970 for dual-polarization radar. The Inception-v3 network (Szegedy et al. 2016b) had an AUC of 0.929-0.956 for single-polarization and 0.971-0.990 for dual-polarization (Chilson et al. 2019). The AUC ranges represent a 95% confidence interval. See Figure 2.6 for a full summary of the results. All results were verified using k-fold cross validation with k=5. These detection results were promising, but finding exact locations would be much more useful for ecologists studying bird roosts.

2.3.2 MistNet

Lin et al. (2019) introduces MistNet, a deep CNN that semantically segments biological data on NEXRAD for the purposes of analyzing bird migration patterns. Lin et al. (2019) used a training set of 239,128 dual-polarization NEXRAD scans across CONUS at zero to three hours after sunset. The test set contained 971 dual-polarization and 4,891 single-polarization scans three hours after sunset, which is the peak time for nocturnal migration. All pixel labels, except a subset of 50 used to calculate statistics, were noisy; rather than hand-labeling, all pixels with $\rho_{HV} <= 0.95$ were labels as biology.

MistNet uses a VGG-16 architecture Simonyan and Zisserman (2015) pretrained on ImageNet, followed by a CNN architecture from Shelhamer et al. (2016) that makes a prediction every eight pixels. To better generalize the model to data prior to 2012-2013, the training data was augmented with a downsampled version of each scan that matched the resolution of single-polarization radar. MistNet outputs were also post-processed as follows: pixels were classified as precipitation with a cut off of >0.45 on either a single elevation or an average of all five elevations. Further, the eight pixels surrounding a pixel classified as precipitation were also classified as precipitation.

MistNet obtained an F-score of 97.3 on single-polarization and 97.9 on dual-polarization radar. The true postive rates (TPR) were 98.8 and 99.4 and true negative rates (TNR) were 99.0 and 96.3 for single- and dual-polarization, respectively.

To demonstrate its usefulness for nocturnal bird migration, Lin et al. (2019) analyzed MistNet's output on about 10 million scans from the spring and fall of 1999 to 2018. MistNet outputs were used to estimate cumulative migration pattern during the entire period for CONUS, illustrate the differences among migration patterns during different seasons and nights, and estimate where birds might be in airspace. Results can be found in Figure 5 of Lin et al. (2019).

Lin et al. (2019) differs from this thesis in that we are examining roost rings, not biological data in general. Second, we have a comparably small dataset that has been hand-labeled by ecologists and verified in the field. Since MistNet labels all pixels with $\rho_{HV} <= 0.95$ to biology, we can compare the effectiveness of a small, handle-labeled dataset to that of many noisy labels. Lin et al. (2019) also provided evidence that not only are CNNs capable of locating biological data accurately, but deeper architectures are also necessary for capturing features involved in roost localization.

2.3.3 Radar Methods

Other approaches to finding birds or roosts focus on hardware. Zaugg et al. (2008) modifies a radar to detect individual, flying birds by sensing the movement of their wings. Though the radar can detect single birds within 8 km, it has a fixed radar beam and cannot detect larger birds properly. The Avian Radar Sensor Design, proposed by (Weber et al. 2005), modifies a WSR-88D radar to detect flocks of birds or large, solitary birds within 10 km, though the modification causes the radar to detect weather less effectively. Finally, the Airport Surveillance Radar (ASR-9) detects small flocks that could potentially harm a plane during takeoff or landing (Troxel et al. 2001). ASR-9 takes readings every five seconds at multiple angles and is effective up to 10 km away.

This thesis focuses on implementing a software solution to finding roosts on existing NEXRADs currently used in weather forecasting, rather than using specialized radars. However, these works provide early evidence of radars' capabilities for locating birds and flocks. They also show the limitations of previous methods; hardware solutions are expensive and difficult to scale, especially given their limited ranges of about 10 km.

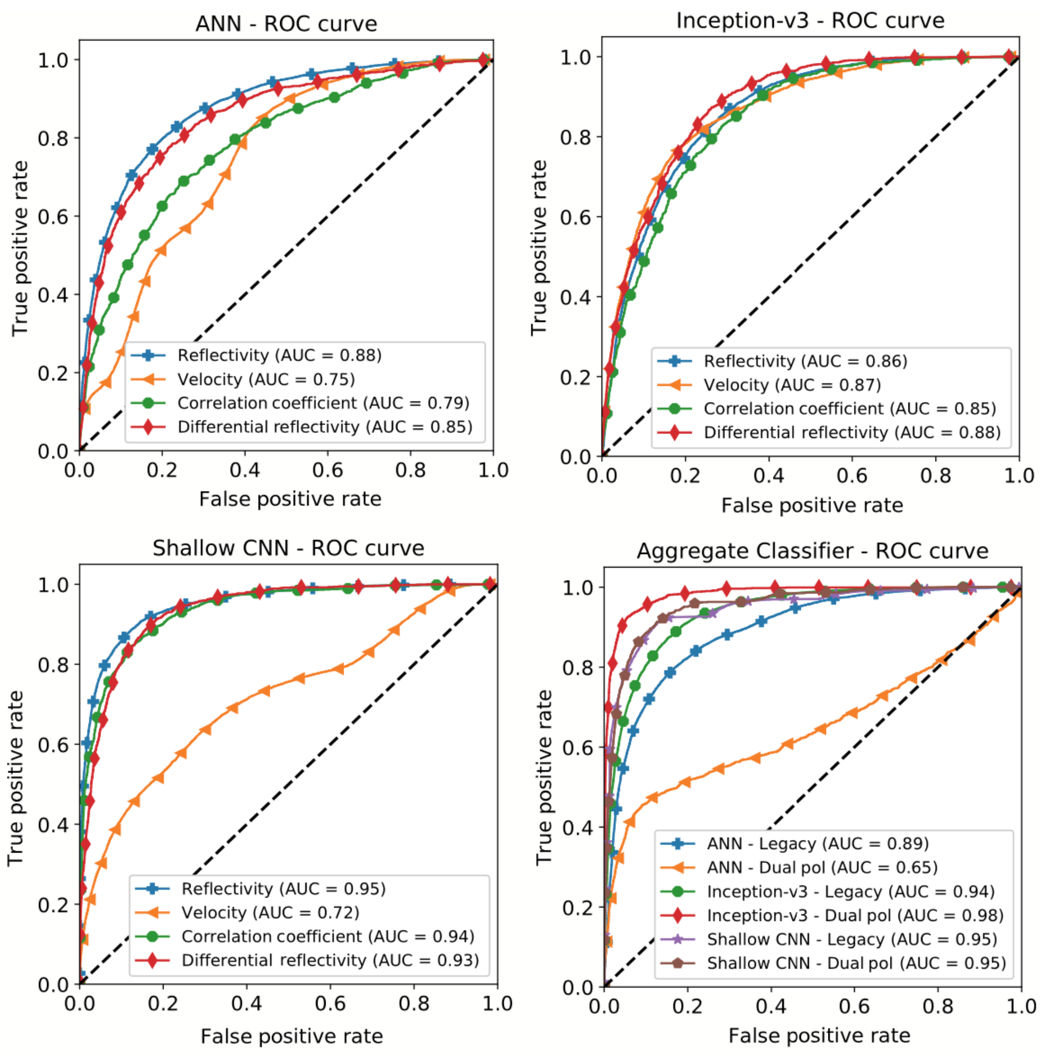

Figure 2.6: The ROC curves of each radar field for a standard (not convolutional) neural network (top left), the Inception-v3 network (top right), and a custom shallow CNN (bottom left). The bottom right ROC curve shows the aggregate results for all three architectures on both dual- and single-polarization data. The figure is from Chilson et al. (2019).

Chapter 3

Dataset

The Purple Martin Conservation Association collected the majority of the data for this thesis showing roost location (Kelly et al. 2012), though Bridge et al. (2016) collected a smaller amount of data. Chilson et al. (2019) labeled all negative examples (or examples that do not contain a roost).

The dataset consists of NEXRAD images in the eastern half of the United States from the years 2009 to 2017. Because all NEXRADs were incrementally updated to dual-polarization in 2012 and 2013, data before that period is only available as single-polarization. Figure 3.1 illustrates the locations of the radars and data and the types of data found in each location.

The input consists of RBG images created with the Py-ART package (Helmus and Collis 2016). The data is of the form $(width \times height \times \# \; of \; channels)$, where the width and height are each 250 pixels and the number of channels is three to represent red, green, and blue. Because this thesis is an extension of Chilson et al. (2019), the data is in the same RGB format since Chilson et al. (2019) originally used color images for transfer learning. Chapter 6 discusses potential changes to this format.

Single-polarization data consists of the reflectivity and velocity radar products, while dual-polarization data consists of reflectivity, velocity, ρ_{HV}, and Z_{DR} products. See Chapter 2 for more information on these radar products. In the dataset for this thesis, there are 11,314 examples with roosts and 10,836 without

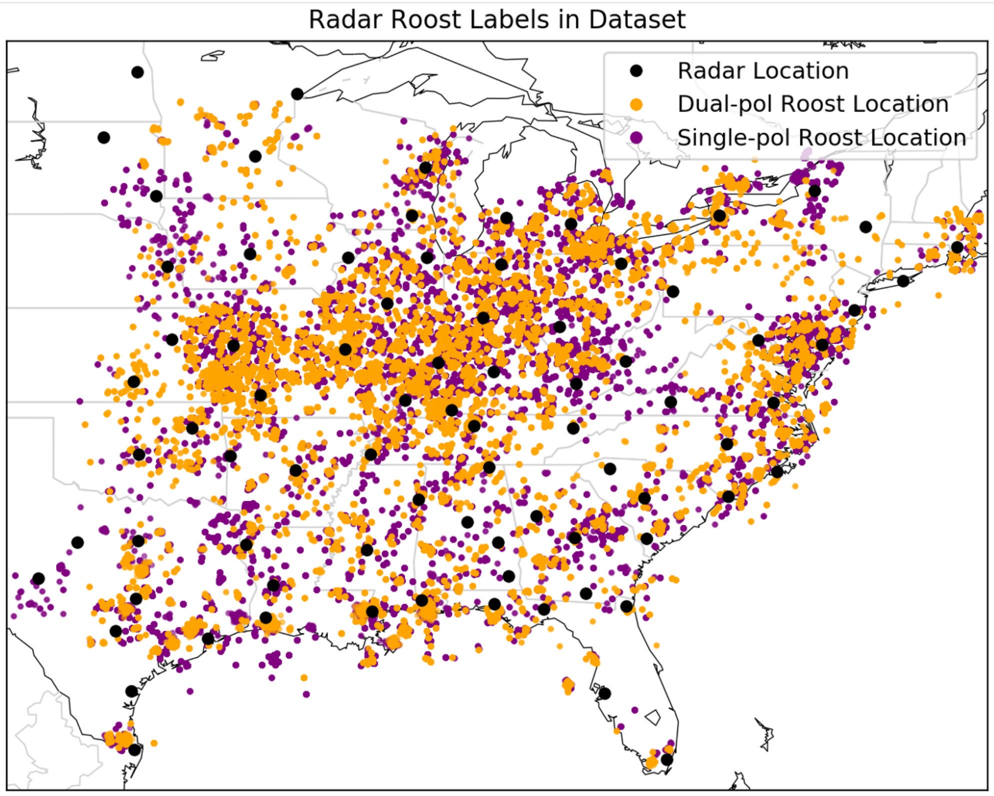

Figure 3.1: Each roost label from the Oklahoma Biological Survey dataset. Only radar locations with associated data are shown.

roosts for the reflectivity and velocity products, and there are 6,383 examples with roosts and 6,464 without roosts for the ρ_{HV} and Z_{DR} products. Table 3.1 summarizes the number and type of examples in the dataset. The detection problem uses both examples with and without roosts, while the localization problem only uses examples with roosts. Labels for the detection problem are simply roost (positive) or no roost (negative), while the labels for localization are created using pixel segmentation. See Chapter 4 for more details.

The public NEXRAD data was downloaded from Amazon Web Services (AWS), and the Oklahoma Biological Survey collected the original data labels (Amazon Web Services 2020). Labels include the latitudes, longitudes, and

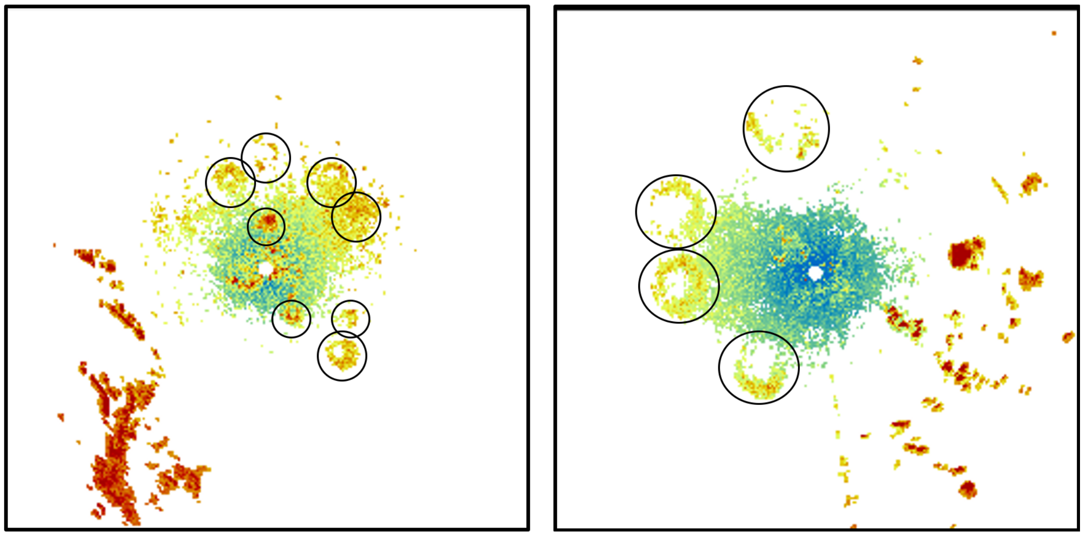

Figure 3.2: Examples of radar reflectivity images containing multiple roost rings, which are circled in black. The left subfigure shows eight roost rings on the KLSX radar on August 1, 2012 at 10:53:35 GMT, and the right subfigure shows four roost rings on the KBMX radar on July 15, 2011 at 11:19:14 GMT. Radar images were created with Py-Art (Helmus and Collis 2016).

sometimes radii of the roosts, along with associated metadata, such as timestamps, radars, and ID numbers. A single radar image could have one or multiple roost locations, as shown in Figure 3.2.

The original dataset had been rounded to anywhere between one and six decimal places, which is not precise enough to correctly predict locations since labelled locations can be off by tens of kilometers. To remedy this issue, most of the data was relabeled using a web application created using JavaScript and Electron (Electron 2020). See Figure 3.3 for details on the interface.

3.1 Data Augmentation

Data augmentation can greatly increase the effective dataset size when hand-labelled data is scarce (Dieleman et al. 2015). As an added benefit, augmentation also has an implicit regularizing effect on models (Hernández-García and

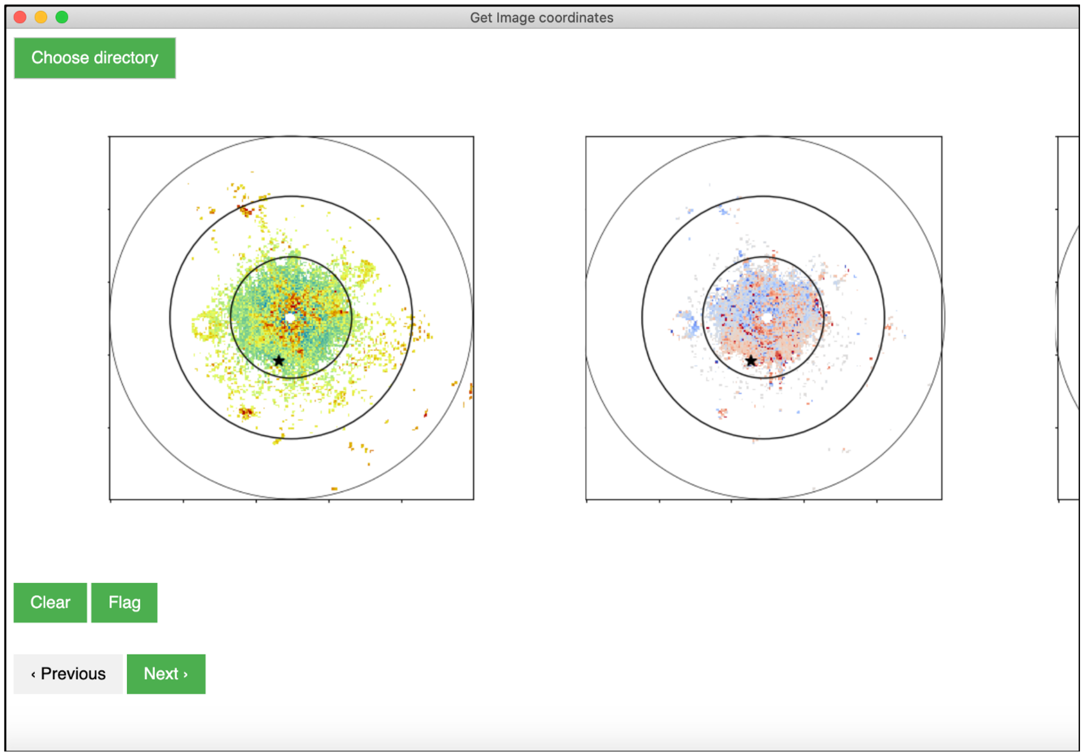

Figure 3.3: This simple Electron app written in JavaScript was used to relabel roost locations in the dataset. Users click on spots where roosts are likely present and have the option of flagging spots that may or may not have a roost. The small black star represents the label from the original dataset. Radar images were created in Py-ART (Helmus and Collis 2016). Code for the Electron app can be found at https://github.com/kteavery/record-long-lat.

König 2018). Augmentation is accomplished by applying transformations to data, such as rotating, translating, scaling, flipping, adding noise, adjusting brightness, and changing color. When applying these transformations, the final label must be preserved (Dieleman et al. 2015). For example, distorting the roost ring into a physically unrealistic shape would be an invalid transformation.

Because the dataset is small, we augmented the data by first flipping, then rotating by 45°, and finally adding salt-and-pepper noise. By stacking these augmentation techniques, the dataset size was increased by 32 times, as illustrated in Figure 3.4 and Table 3.1. Although flipping and rotating preserve the

	Roost	No Roost
Single-Pol Available (No Augmentation)	11,314	10,836
Dual-Pol Available (No Augmentation)	6,383	6,464
Single-Pol Available (With Augmentation)	362,048	346,752
Dual-Pol Available (With Augmentation)	204,256	206,848

Table 3.1: The number and distribution of labels before and after data augmentation.

physical realism of the bird roosts, these transformations would not be acceptable for identifying weather on NEXRAD images since the weather would come from unrealistic directions.

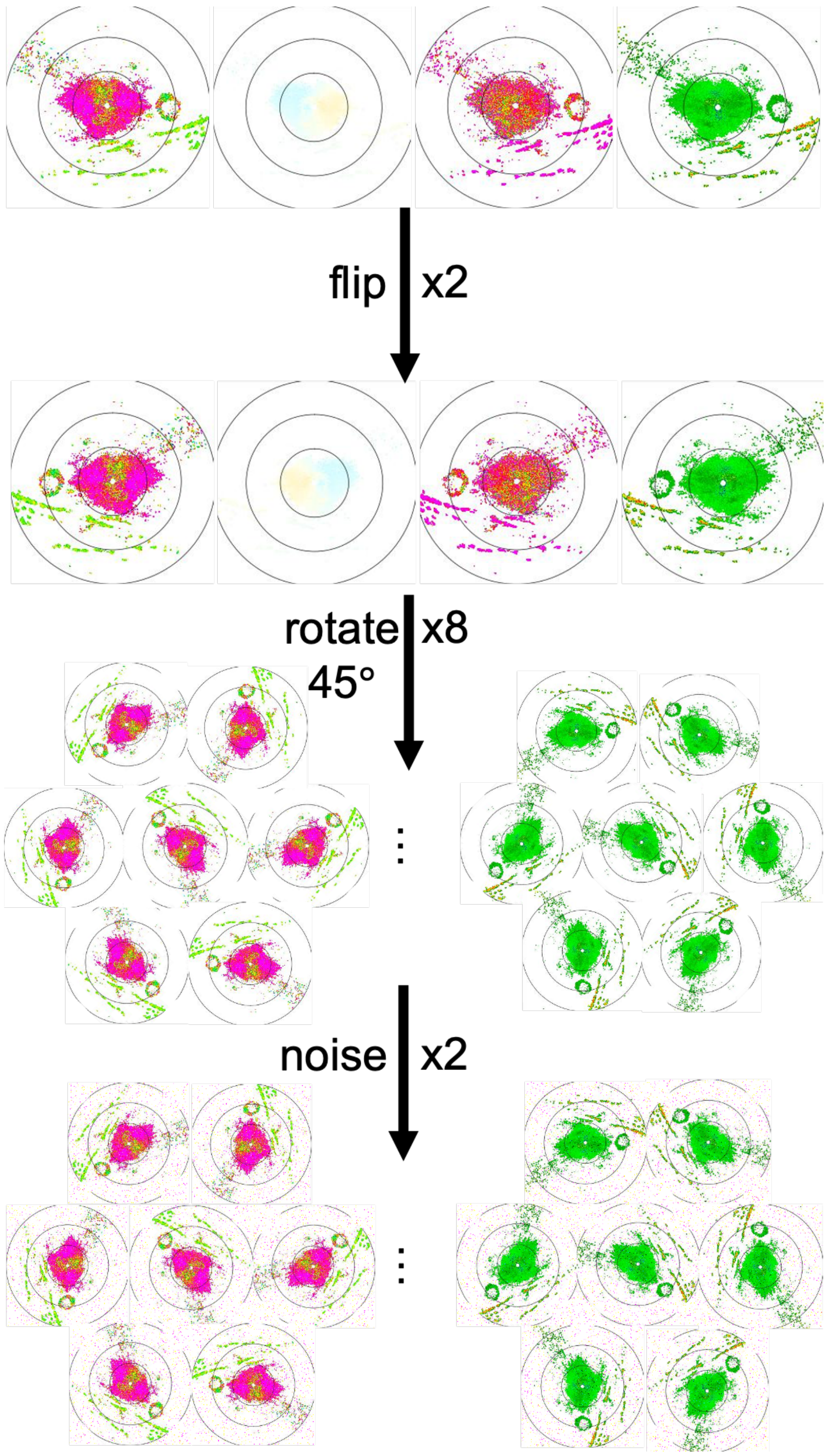

Figure 3.4: This figure shows the result of augmenting the radar images shown in Figure 1.1 using the augmentation procedure described. The rotation and noise steps only show the reflectivity and Z_{dr} products for conciseness. Radar images were created with Py-ART (Helmus and Collis 2016).

Chapter 4

Methods

This chapter describes the implementation details of both bird roost detection and localization problems described in Chapter 1. Detection is examined both as a preliminary question to verify the correctness of our data processing and machine learning methodology, as well as a comparison to Chilson et al. (2019). The code for this project is publicly available on GitHub.[1]

4.1 Network Architectures

Several different architectures were examined during the course of the experiments, including a custom, shallow CNN to detect the presence of roosts and a U-Net to segment images and thus locate roosts. These architectures are applied separately to each radar field: reflectivity, velocity, ρ_{HV}, and Z_{DR}. The outputs of the networks for each field are then passed through two dense layers (for the shallow CNN) or two convolutional layers (for the U-Net) to combine their outputs, as shown in Figure 4.1.

4.1.1 Shallow CNN

The architecture of the custom, shallow CNN developed is shown in Figure 4.2. This architecture is based on the custom architecture described by Chilson et al.

[1]https://github.com/kteavery/BirdRoostLocation

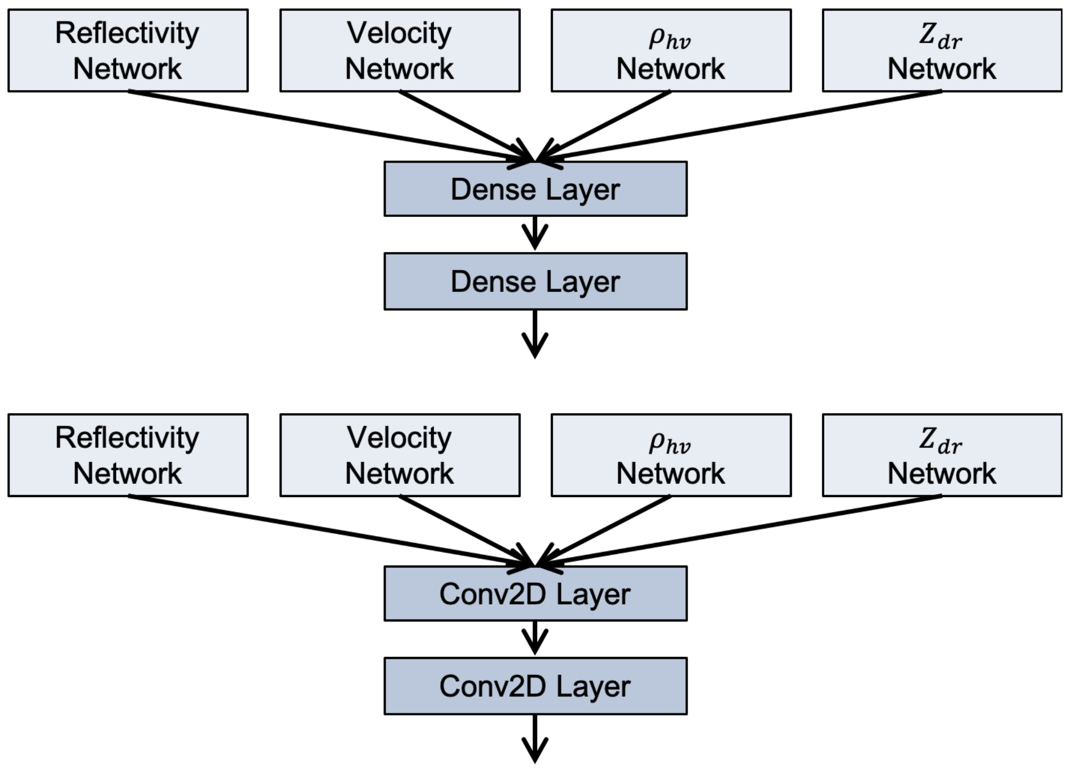

Figure 4.1: All four radar fields are trained separately and combined with either (top) two dense layers or (bottom) two convolutional layers.

(2019) with a few modifications that were determined via a grid search. First, we add CoordConv layers, as described in Chapter 2. We then change the order in which the batch normalization layers are applied. In Chilson et al. (2019) and in the original Ioffe and Szegedy (2015) paper, batch normalization is applied before the activation layer, but this is changed to after the activation layer so that the input of the following layer will be normalized. Here the hidden layer refers to the unit consisting of the convolutional layer, activation layer, and any other operations. Finally, we add three more hidden layers to capture more complex features. A summary of the hyperparameters for both the custom CNN in this thesis and the CNN in Chilson et al. (2019) is shown in Table 4.1.

Chilson et al. (2019)

type	BN	filters	nodes	kernel	pool	stride	activation
Conv	Y	32		5 x 5		1	ReLU
Max Pool					2 x 2	2	
Conv	Y	64		5 x 5		1	ReLU
Max Pool					2 x 2	2	
Dense	Y		500				ReLU
Dense	N		2				softmax

Updated CNN

type	BN	CC	filters	nodes	kernel	pool	stride	activation
Conv	Y	Y	8		5 x 5		1	ReLU
Max Pool						2 x 2	2	
Conv	Y	Y	8		5 x 5		1	ReLU
Max Pool						2 x 2	2	
Conv	Y	Y	16		5 x 5		1	ReLU
Max Pool						2 x 2	2	
Conv	Y	Y	32		5 x 5		1	ReLU
Max Pool						2 x 2	2	
Conv	Y	Y	64		5 x 5		1	ReLU
Max Pool						2 x 2	2	
Dense	Y	N		500				ReLU
Dense	N	N		2				softmax

Table 4.1: A comparison of the architecture and hyperparameters of Chilson et al. (2019)'s shallow CNN (top) and this work's CNN for detection (bottom). BN is an abbreviation for batch normalization, and CC is an abbreviation for CoordConv.

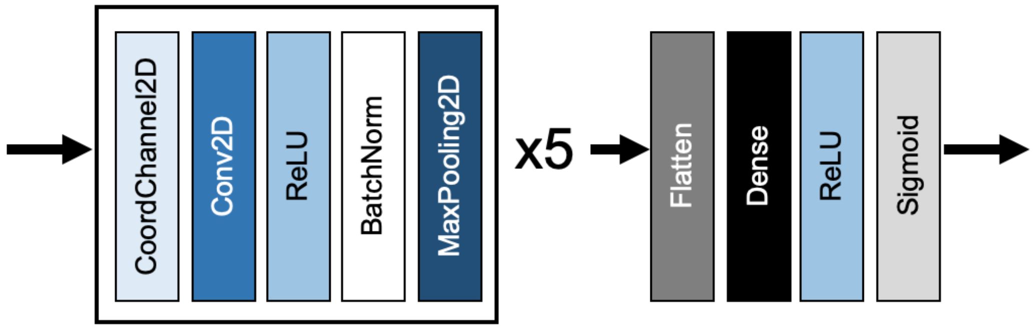

Figure 4.2: The custom CNN includes five consecutive units of the boxed layers shown above, followed by the flatten, dense, ReLU, and sigmoid layers. This architecture was trained on each of the radar fields for the detection problem.

4.1.2 U-Net

The U-Net architecture is the same as described in Ronneberger et al. (2015) and shown in Figure 2.5. Because the architecture is used for localization, the ground truth is a 240 × 240 matrix with a filled circle around each roost. The radius of the circle is either listed in the label (11.73% of cases) or set to the average radius of 28 km if no radius is listed (88.27% of cases). Roosts locations were also interpolated based on data at the same radar from other days of the month. For example, a roost visible on July 6 and July 8 but not on July 7 would be labeled as present on all three days.

When labeling the images using the interface in Figure 3.3, clicking on a roost in an image records the coordinates of the roost. If several records are created within 20 pixels of each other, the records are averaged to create one circle on the ground truth mask. Records greater than 20 pixels from each other are considered separate roosts, although the circles created from the records sometimes overlap on the ground truth masks. An example of this is shown

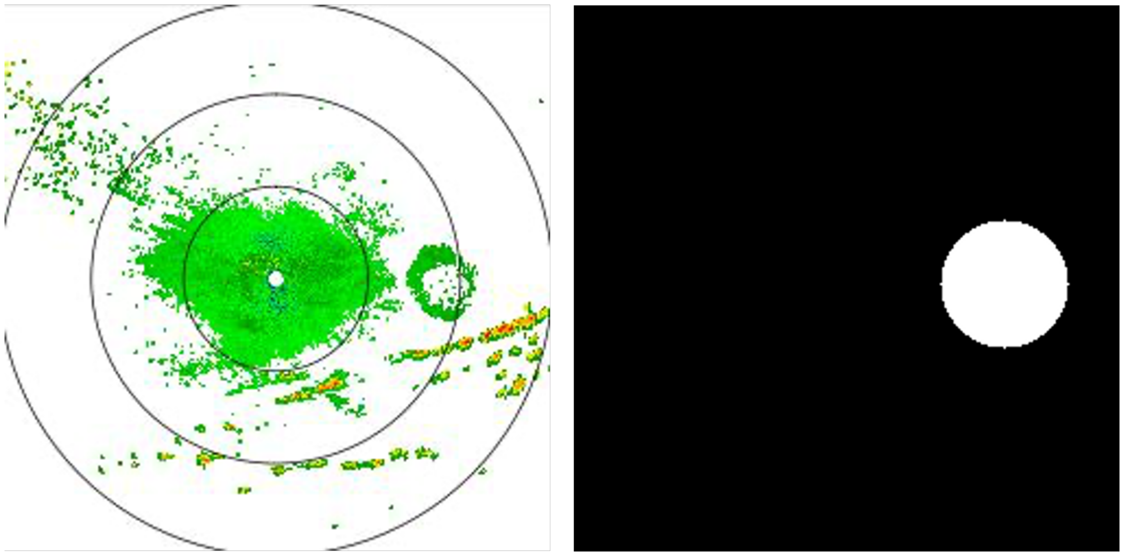

Figure 4.3: A 240 × 240 pixel mask showing a July 16, 2015 reflectivity image with one roost and its corresponding mask. The left image was created with Py-ART (Helmus and Collis 2016).

in Figure 4.4. A discussion of potential changes to the labeling method is in Chapter 6.

Figure 4.3 shows a simple example of a mask with one roost, while Figure 4.4 shows a more complex case. All reflectivity images shown in Figure 4.4 are from the KGRX radar in June 2014, but different roosts are more visible at different times. In the top right reflectivity image, the roosts are obscured by weather, and various roosts are visible in the other three images.

4.2 Training

Training took place on a Microsoft Azure machine with 12 CPUs and 112 GB of RAM. A complete run of the neural net for detection takes about five to six hours, while localization takes about four days.

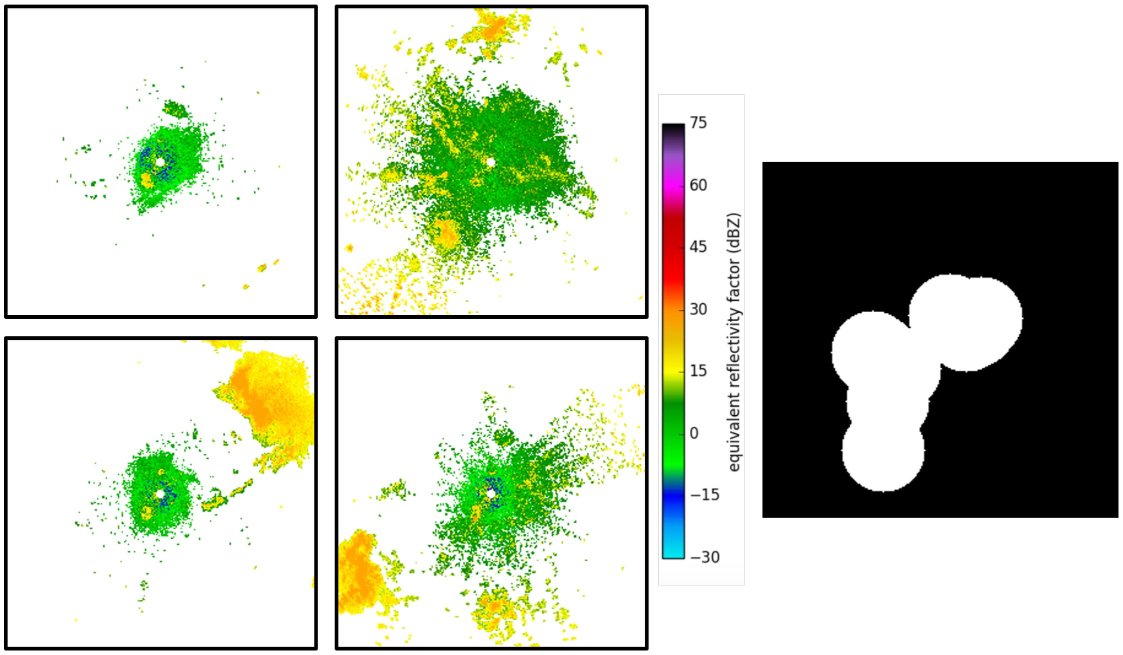

Figure 4.4: A mask showing the locations of roosts on the KGRX radar in June 2014. The exact days and times are as follows: top left – July 8, 2014 at 11:16:58 GMT, top right – July 30, 2014 at 11:44:22 GMT, bottom left – July 31, 2014 at 11:35:27 GMT, bottom right – July 11, 2014 at 11:19:24 GMT. Radar images were created with Py-ART (Helmus and Collis 2016).

4.3 Prediction

To further validate our results, we examine the results of our method on the three NEXRADs in Indiana: KIND, KIWX, and KVWX. We examine the years 2018 and 2019 from late June to late August starting an hour before sunrise, which totals to 3364 images for each radar field.

4.4 Metrics

For the detection problem, true positives (TP) are correct predictions of a roost in an image, while false positives (FP) are incorrect predictions of a roost when one does not exist. Conversely, true negatives (TN) are correct predictions

		Actual Label	
		Roost	Background / No Roost
Predicted Label	Roost	TP	FP
	Background / No Roost	FN	TN

Table 4.2: A binary contingency table for whether or not (in the case of detection) a NEXRAD image contains a roost or (in the case of localization) the correct location is identified on a NEXRAD image.

that a roost does not exist in an image, and false negatives (FN) are incorrect predictions that a roost does not exist. See Table 4.2.

To determine these metrics for the localization problem, we must discretize the predictions and labels. In general, a "positive" label would be the circle of pixels that are marked with a roost, while a "negative" label would be background or the pixels that are not marked with a roost. A pixel of a prediction is marked as correct (true) if it is the same as a corresponding pixel of the truth mask. If the pixel is different from the truth mask, the pixel is labeled as false. Therefore, a whole prediction image could be some part true and some part false. The dice score coefficient measures the proportion of overlap between predictions and ground truth and is defined in Chapter 2, along with dice loss.

Equations 4.1, 4.2, and 4.3 are used to evaluate the effectiveness of the given method for either detection or localization. ACC stands for accuracy, TPR stands for true positive rate, and TNR stands for true negative rate:

$$ACC = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.1}$$

$$TPR = \frac{TP}{TP + FN} \tag{4.2}$$

$$TNR = \frac{TN}{FP + TN} \tag{4.3}$$

Receiver-operating characteristic (ROC) curves plot the true positive rate versus the false positive rate. Because both rates are between zero and one, the line y=x represents random guessing, and curves above that line represent better than random performance. The area under the curve (AUC) of the ROC curve is a useful summary of the plot and a metric of performance that accounts for true and false positives and negatives. An AUC of 0.5 is random, while 1.0 is perfect (Fawcett 2006).

K-fold cross validation is an evaluation method that gives a more realistic estimate of a model's skill. The dataset is split evenly into k groups. The first group is used as the test set, the second as the validation set, and the remaining groups as the training set. This process is repeated for each combination of the groups. For example, the second group might be the test set, the fifth group the validation set, and the remaining the training set. The experiments are run on each combination of training, test, and validation sets. For the models in this thesis, k is equal to five.

Chapter 5

Results and Discussion

5.1 Shallow CNN

The classification architecture was moderately successful in detecting roosts. The binary cross entropy loss for all four products and the aggregate classifier of the products is shown in Figures 5.1 and 5.3. The accuracy, true positive and negative rates, and the area under the ROC curve are summarized in Table 5.1. The learning curves are shown in Figures 5.2 and 5.4, and ROC curves are shown in Figure 5.5.

ρ_{HV} and Z_{DR}, the radar products that became available after NEXRAD was updated in 2012-2013, were slightly more unstable than reflectivity and velocity, possibly because there was less data available for those fields. Z_{DR} was the most predictive field with an AUC of about 0.6675 and an accuracy of about 61.78%. The aggregate detection performed comparably to its Z_{DR} input, but it would likely perform much worse on pre-update data, which would only include reflectivity and velocity. A different aggregate architecture may be able to combine the four fields more effectively and rely more heavily on the more accurate Z_{DR} field.

This detection experiment corrects Chilson et al. (2019)'s results. The shallow architecture used in this work was also based on Chilson et al. (2019)'s shallow architecture, but it was further refined, as described in Chapter 4.

	BCE	ACC
Reflectivity	0.7110 ± 0.0193	0.5784 ± 0.0165
Velocity	0.6963 ± 0.0653	0.5673 ± 0.0357
ρ_{HV}	0.6330 ± 0.1326	0.5974 ± 0.0248
Z_{DR}	0.5439 ± 0.1952	0.6178 ± 0.0139
Aggregate	0.6476 ± 0.0798	0.5947 ± 0.0231

	TPR	TNR	AUC
Reflectivity	0.5784 ± 0.0165	0.5784 ± 0.0165	0.6089 ± 0.0232
Velocity	0.5673 ± 0.0357	0.5673 ± 0.0357	0.5905 ± 0.0517
ρ_{HV}	0.5974 ± 0.0248	0.5974 ± 0.0248	0.6415 ± 0.0294
Z_{DR}	0.6178 ± 0.0139	0.6178 ± 0.0139	0.6675 ± 0.0163
Aggregate	0.5947 ± 0.0231	0.5947 ± 0.0231	0.6268 ± 0.0346

Table 5.1: Results summary for the detection architecture. The abbreviations are as follows: BCE = binary cross entropy loss, ACC = accuracy, TPR = true positive rate, TNR = true negative rate, AUC = area under a ROC curve.

Chilson et al. (2019) has data contamination, which makes the the results optimistic. The data were split by year, rather than by roost. This would normally make sense in meteorological contexts, but since roost locations persist over months and years, splitting by roost leads to more independence among splits. Chilson et al. (2019)'s results are summarized in Section 2.3.1. See the Results section of Chilson et al. (2019) for more details.

In contrast to Chilson et al. (2019), Z_{DR} performed much better than the reflectivity field, and Z_{DR} performed better after modifications to Chilson et al. (2019)'s original shallow architecture. Unlike Chilson et al. (2019)'s results, velocity performed only slightly worse than other fields.

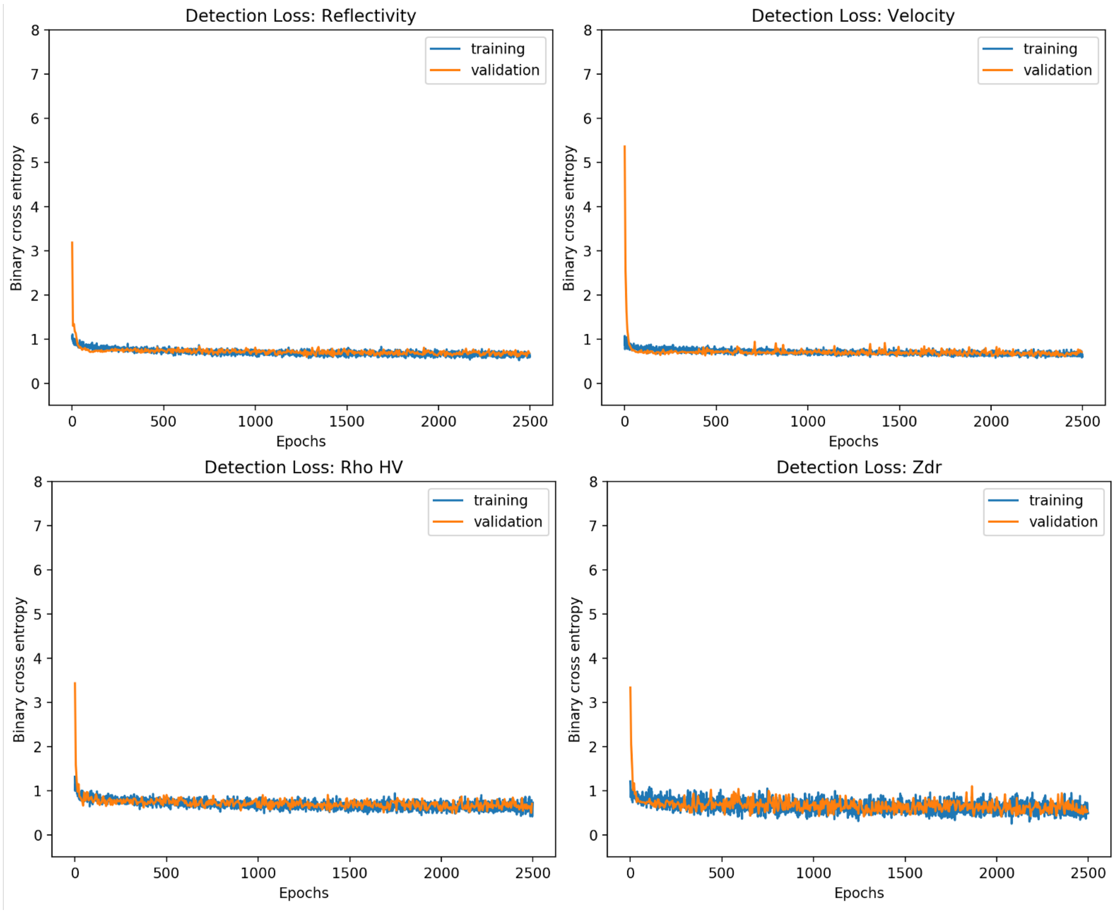

Figure 5.1: The detection loss curves for each radar field for 2500 epochs. The plots are averages of five runs.

5.2 U-Net

The U-Net was somewhat successful in locating roosts. The dice loss for each radar product is shown in Figure 5.6, while the dice loss for the aggregate model is shown in Figure 5.7. Table 5.2 summarizes accuracy, true positive rates, and true negative rates for the localization models as described in Section 4.4.

Because there are many more background than roost pixels, the dataset is imbalanced, and the accuracy metric is biased towards the larger background label. Reflectivity, ρ_{HV}, and Z_{DR} all do well at predicting background with a true negative rate in the high 0.8s, but they do poorly when predicting roost

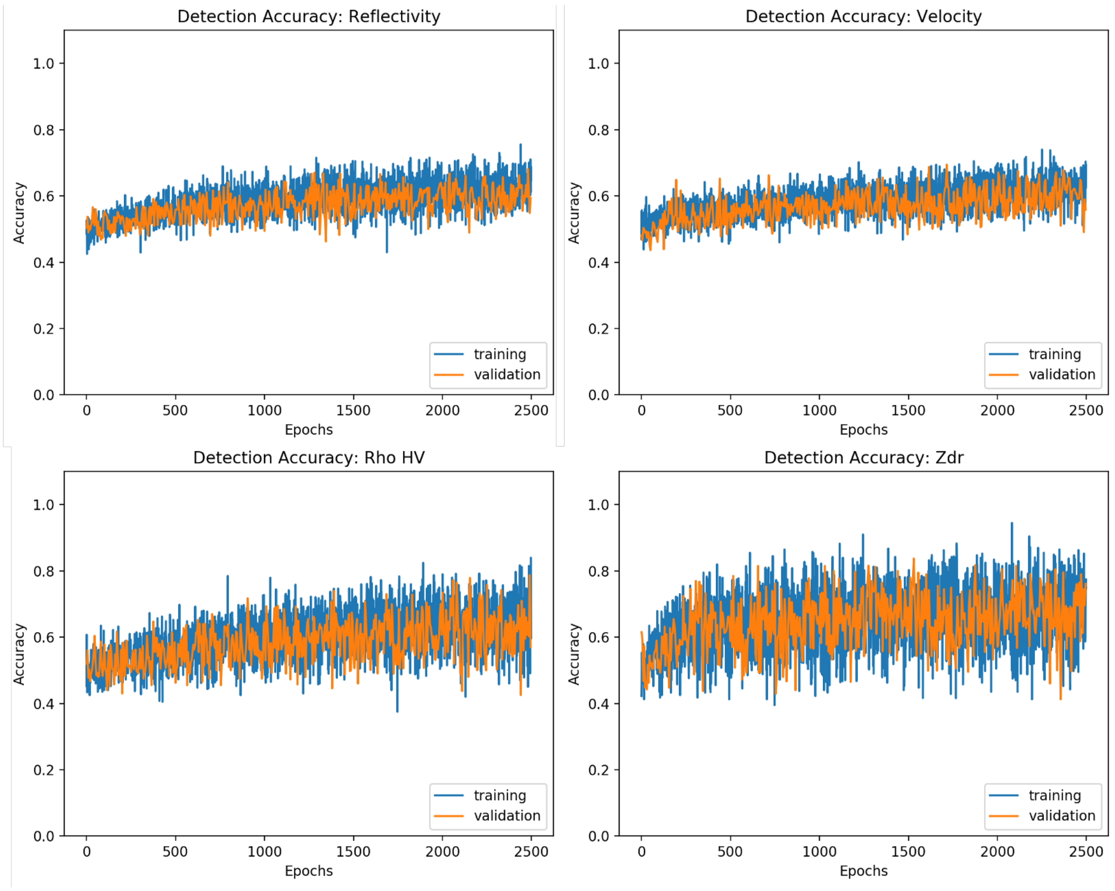

Figure 5.2: The detection learning curves for each radar field for 2500 epochs. The plots are averages of five runs.

locations. Velocity performs roughly equally with true positive and true negative rates at around 0.75 each. The aggregate model does the best at predicting roost locations with a true positive rate around 0.8, but it has a significantly worse true negative rate at around 0.6.

Examples of localization predictions are shown in Figure 5.8. Interestingly, because roosts tend to show up more often in the center of radar scans, the U-Net tends to predict that the roosts will be in the center. If the ground truth shows a roost on one side, as in the case of the bottom panel, the prediction skews towards one side to capture the roost. The predictions also tend to predict roosts in one large, contiguous area, rather than in discrete circles. The area is

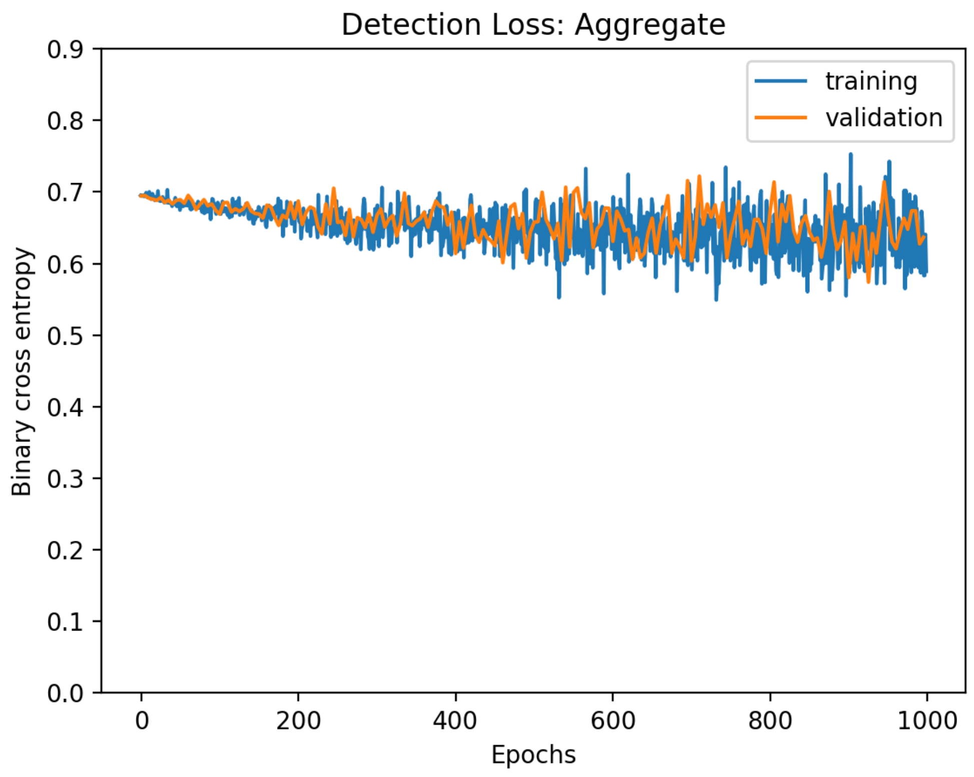

Figure 5.3: Detection loss curve for the aggregate classifier over 1000 epochs. The plot is an average of five runs.

generally the size of the roost or group of roosts, as in the top panel, though it can be larger, as in the bottom panel. Smaller roost labels in the ground truth are likely to be ignored if they are near a large group of labels, as in the case of the leftmost roost label in the top panel.

5.3 Predictions in Indiana

See Chapter 4 for details on the methodology used in this section. Because there is no labelled data for Indiana radars in 2019, we cannot calculate accuracy or true positive and true negative rates. However, we can observe the U-Net's performance on samples, as shown in Figure 5.9.

	Dice Loss	ACC
Reflectivity	0.8326 ± 0.0144	0.8326 ± 0.0144
Velocity	0.8005 ± 0.0485	0.8005 ± 0.0485
ρ_{HV}	0.8172 ± 0.0426	0.8172 ± 0.0426
Z_{DR}	0.8291 ± 0.0228	0.8291 ± 0.0200
Aggregate	2.5055 ± 0.0050	0.6264 ± 0.0017

	TPR	TNR
Reflectivity	0.6178 ± 0.0230	0.8522 ± 0.0172
Velocity	0.6478 ± 0.1139	0.8317 ± 0.0703
ρ_{HV}	0.6114 ± 0.1097	0.8368 ± 0.0568
Z_{DR}	0.6186 ± 0.0389	0.8475 ± 0.0248
Aggregate	0.8031 ± 0.0047	0.6103 ± 0.0024

Table 5.2: Results summary for the localization architecture. The abbreviations are as follows: ACC = accuracy, TPR = true positive rate, TNR = true negative rate. The results for the four products are averages of five trials, while the results for the aggregate are averages of two trials.

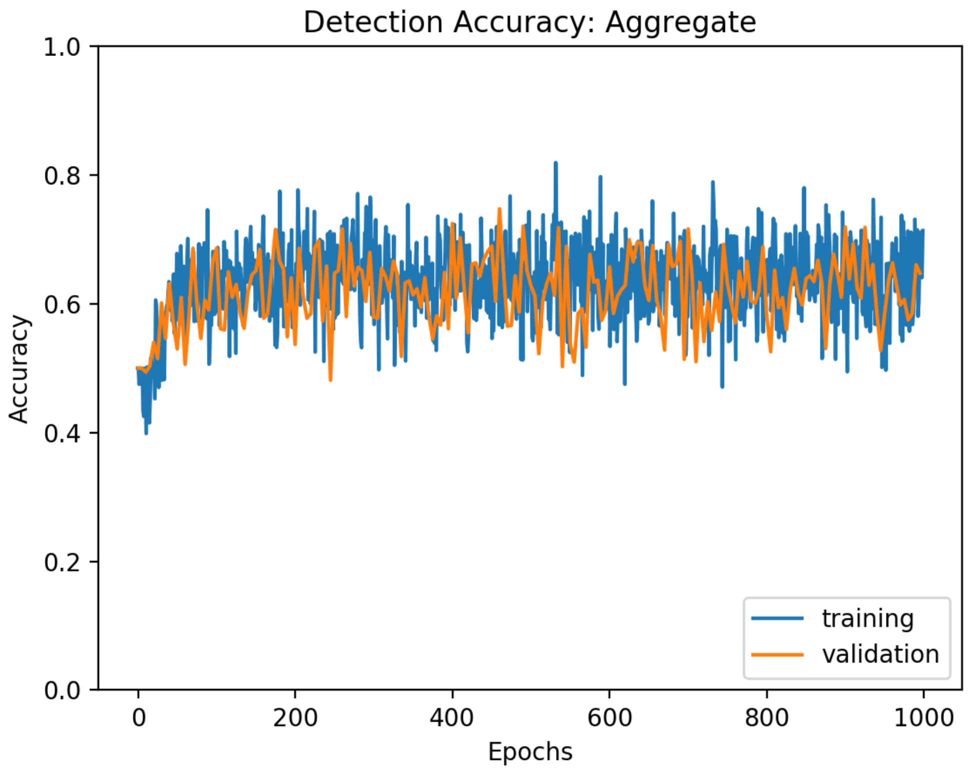

Figure 5.4: The detection learning curve for the aggregate classifier over 1000 epochs. The plot is an average of five runs.

The raw, unlabelled data contains primarily negative examples since most images will not contain a roost. To find the images that contain roosts, the detection architecture is applied first. The positive images identified by the detection architecture are then given to the U-Net for semantic segmentation. Identifying positive images with the detection architecture is necessary because U-Net was trained only on positive examples.

Because the detection architecture has an accuracy of only around 60%, it misclassified many of the unlabeled images. Those images, which primarily contained no roosts, were then passed to the U-Net for segmentation. Of the examined output, the predictions for the 2019 Indiana dataset were almost all

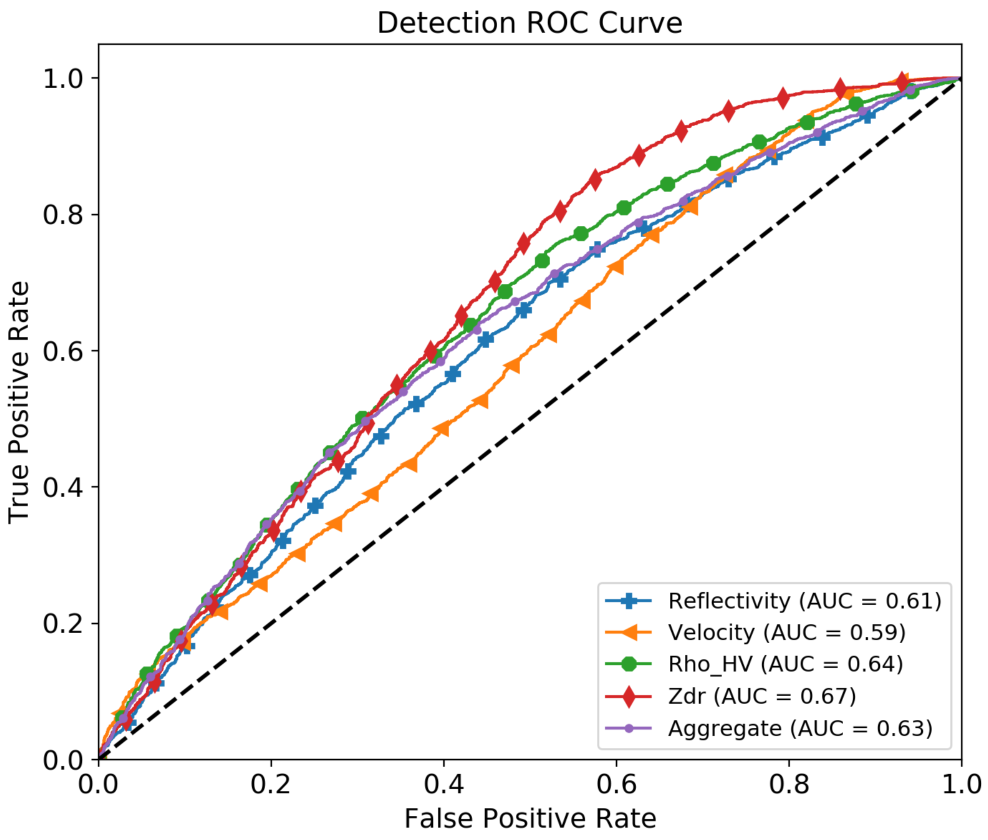

Figure 5.5: The ROC curves for detection.

background since the input data contained few roosts. Examples with roosts tended to be misidentified by the detection architecture and therefore were not segmented by the U-Net.

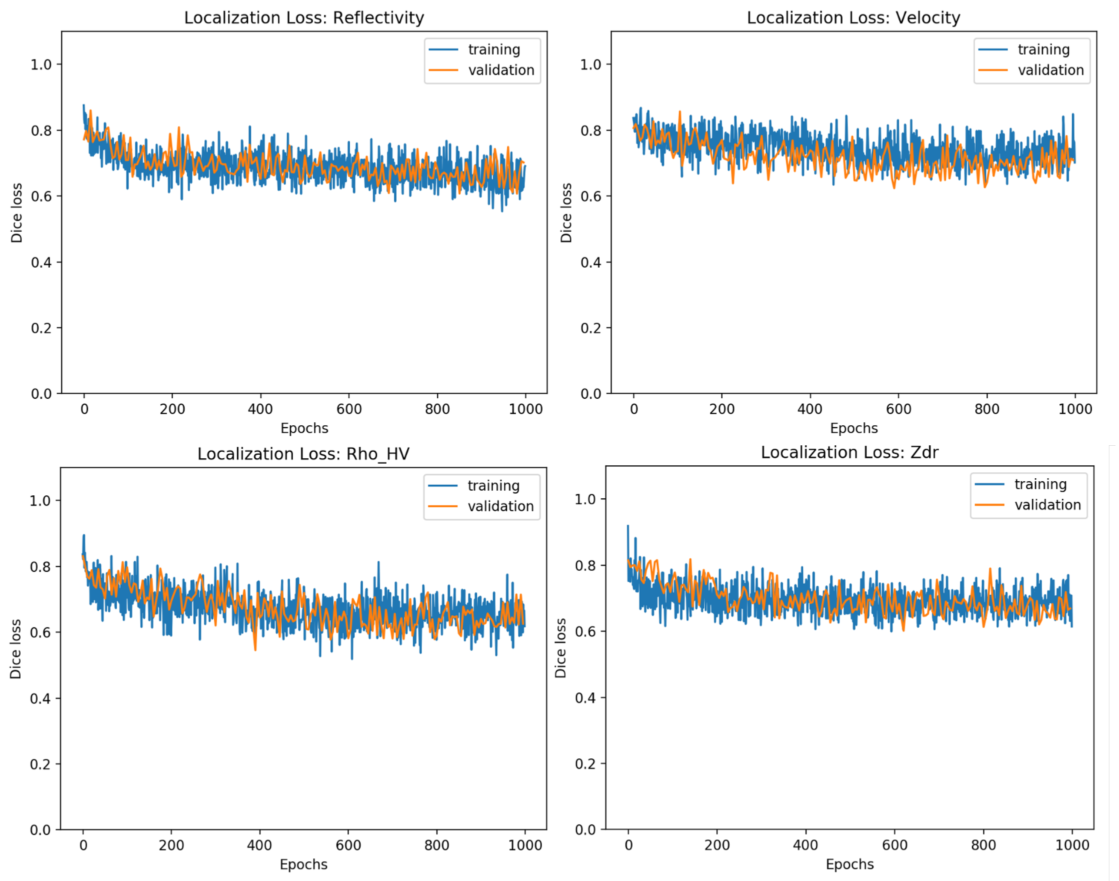

Figure 5.6: The dice loss curves for each radar field for 1000 epochs. The plots are averages of two runs.

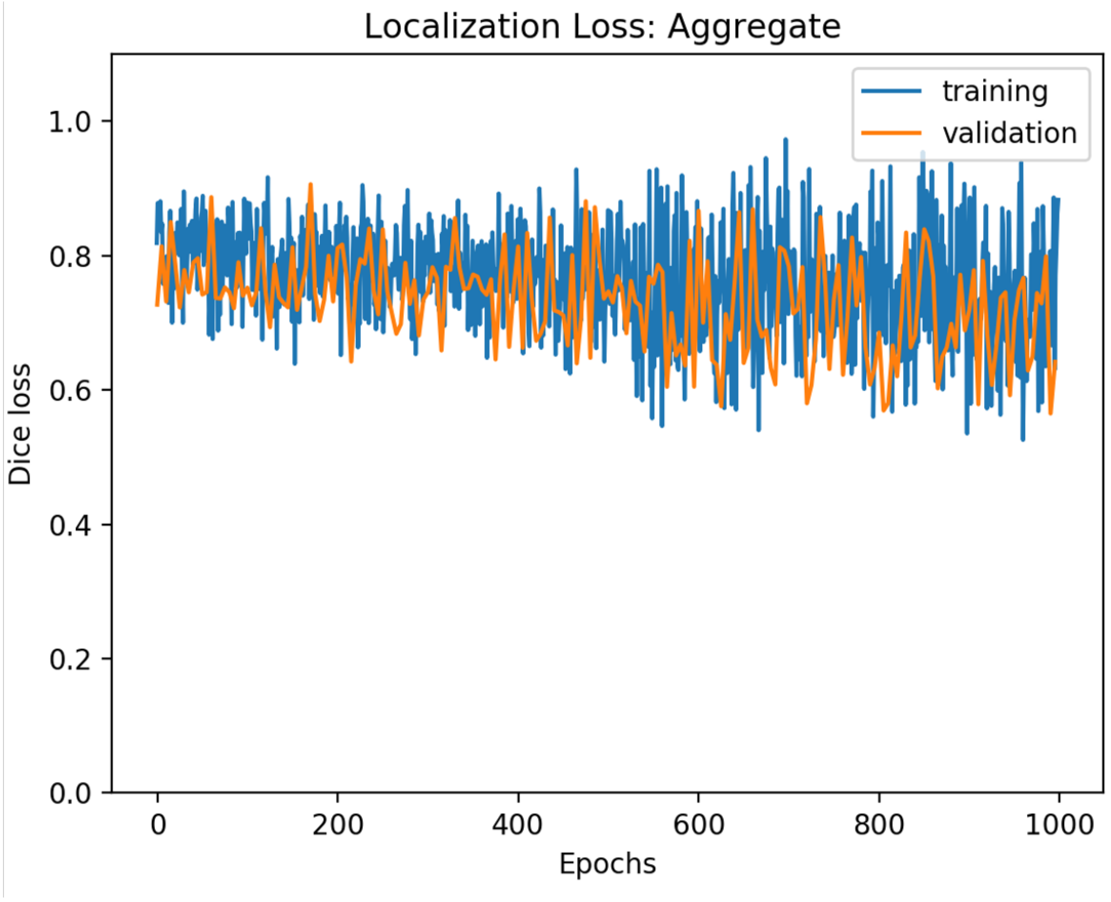

Figure 5.7: The dice loss curve for the aggregate model for 1000 epochs. The plot is an average of two runs.

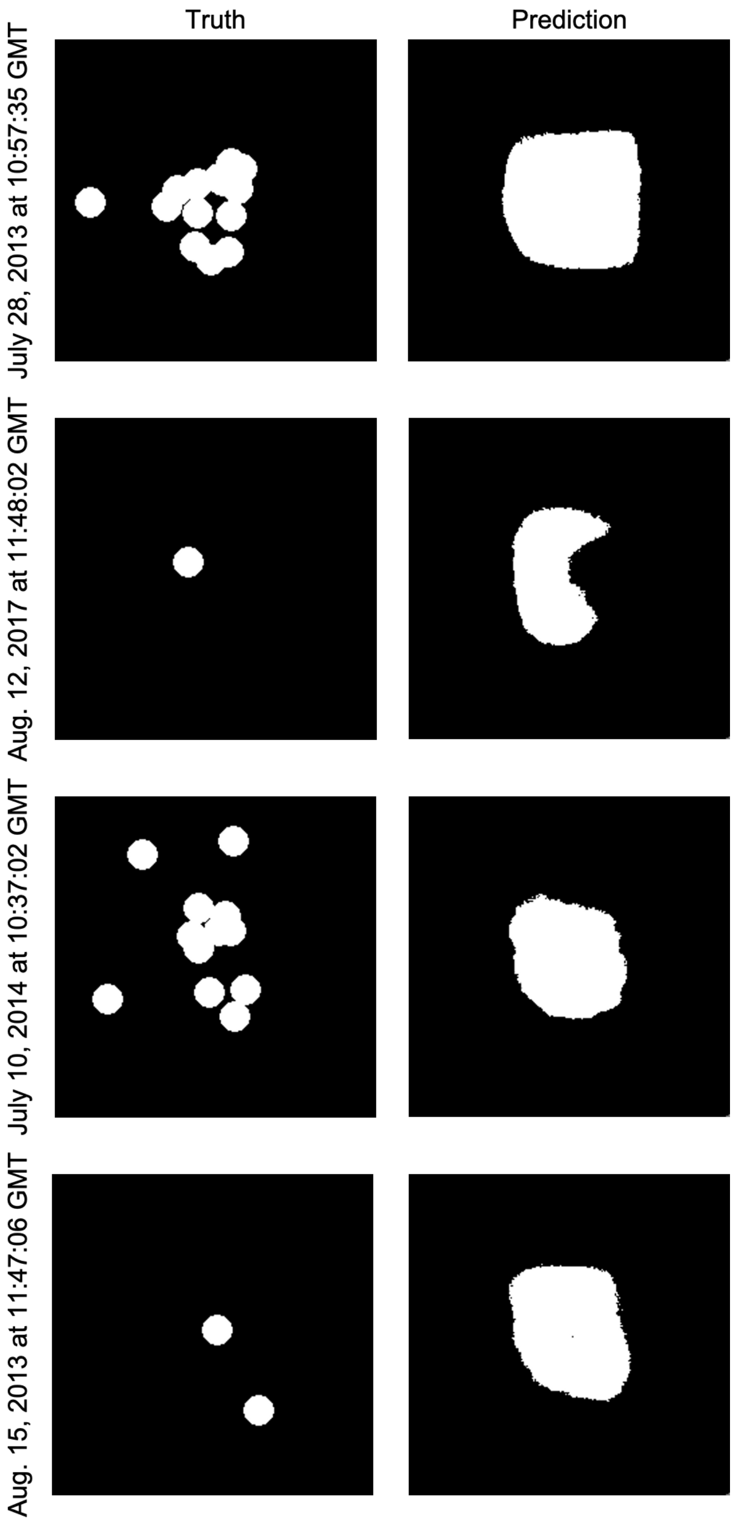

Figure 5.8: Localization prediction examples using the aggregate U-Net.

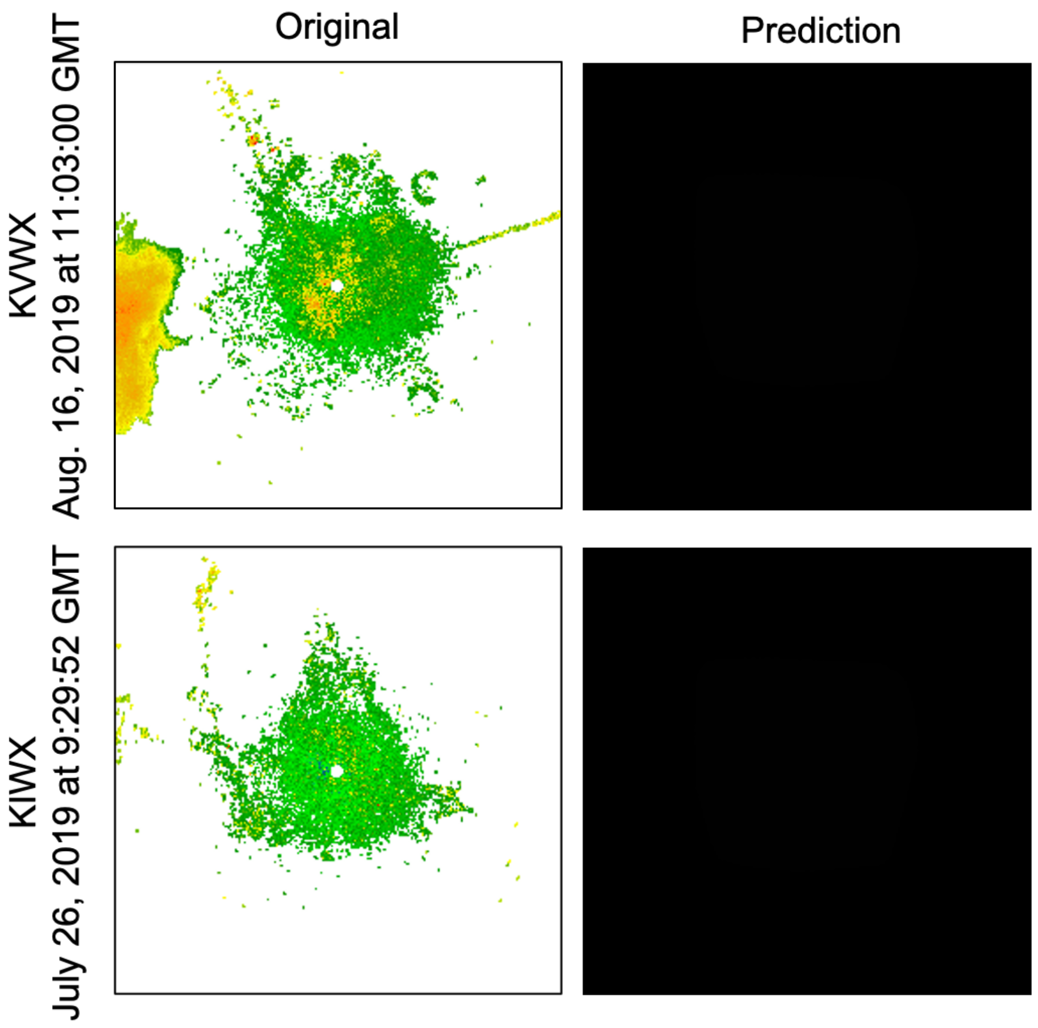

Figure 5.9: Localization prediction examples on the 2019 Indiana dataset. Examples with roosts tended to be misidentified, as shown in the top row, while examples without roosts were more likely to be identified correctly, as shown in the bottom row. As in Figure 5.8, black pixels in the prediction represent background, and white pixels represent a roost. Therefore, all black predictions mean that no roost is predicted in the image.

Chapter 6

Conclusion

Overall, detecting and predicting roosts using the methods shown are somewhat effective, though these methods need additional refining. The detection method corrected Chilson et al. (2019)'s work, which had some data contamination, and the models achieved accuracies of around 60%. Predicting roost location with U-Net achieved a true positive rate of around 80% with the aggregate model and a true negative rate of about 85% with the Z_{DR} model.

Observing predictions reveals the kinds of flaws that the models face. For example, the models tend to predict roosts in the in center of the image, and they predict large groups of roosts as one large, contiguous area. Predictions on the Indiana dataset reveal that the two networks are not yet effective enough to be used in conjunction. The detection network has only a 60% accuracy and is therefore not accurate enough to provide input for the localization network.

6.1 Future Work

This project did not make use of the data's temporal aspects. Network architectures, such as recurrent neural networks or long-short term memory networks, would take temporal features into account. However, the dataset used into this project is not large enough to properly train such networks since the networks would require combining timestamps into a single example, which would drastically reduce the amount of data.

Improving the mask creation method could make the localization method more effective. The average radius size of 28.0 km seems large for many roosts in the dataset, though this likely represents the roost rings when they reach their maximum size before disappearing from view. Also, labeling each timestamp individually may be more effective than interpolating roost locations based on other days of the month.

Now that the data and splits used in Chilson et al. (2019)'s original paper has been corrected, it would be interesting to redo an architecture search to see if another architecture would be more effective for detection. A more sophisticated architecture may capture complex features more effectively. Further, combining the field outputs into an aggregate classification may be more effective with a different architecture, rather than just two dense or convolutional layers.

Finally, the dataset should be converted to grayscale to save on computational costs. The dataset is currently in color because it is an extension of Chilson et al. (2019), which used color images for transfer learning.

Reference List

Amazon Web Services, 2020: .
URL `https://aws.amazon.com/public-datasets/nexrad/`, Accessed on 7-29-2020.

Bauer, S., J. W. Chapman, D. R. Reynolds, J. A. Alves, A. M. Dokter, M. M. Menz, N. Sapir, M. Ciach, L. B. Pettersson, J. F. Kelly, et al., 2017a: From agricultural benefits to aviation safety: Realizing the potential of continent-wide radar networks. *BioScience*, **67**, 912–918.

Bauer, S., J. W. Chapman, D. R. Reynolds, J. A. Alves, A. M. Dokter, M. M. H. Menz, N. Sapir, M. Ciach, L. B. Pettersson, J. F. Kelly, H. Leijnse, and J. Shamoun-Baranes, 2017b: From Agricultural Benefits to Aviation Safety: Realizing the Potential of Continent-Wide Radar Networks. *BioScience*, **67**, 912–918.
URL `https://doi.org/10.1093/biosci/bix074`

Bauer, S. and B. J. Hoye, 2014: Migratory animals couple biodiversity and ecosystem functioning worldwide. *Science*, **344**, 1242552.

Bengio, Y., P. Simard, and P. Frasconi, 1994: Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**, 157–166.

Bridge, E. S., S. M. Pletschet, T. Fagin, P. B. Chilson, K. G. Horton, K. R. Broadfoot, and J. F. Kelly, 2016: Persistence and habitat associations of purple martin roosts quantified via weather surveillance radar. *Landscape ecology*, **31**, 43–53.

Chilson, C., K. Avery, A. McGovern, E. Bridge, D. Sheldon, and J. Kelly, 2019: Automated detection of bird roosts using nexrad radar data and convolutional neural networks. *Remote Sensing in Ecology and Conservation*, **5**, 20–32.
URL `https://zslpublications.onlinelibrary.wiley.com/doi/abs/10.1002/rse2.92`

Chilson, P. B., E. Bridge, W. F. Frick, J. W. Chapman, and J. F. Kelly, 2012a: Radar aeroecology: exploring the movements of aerial fauna through radio-wave remote sensing.

—, 2012b: Radar aeroecology: exploring the movements of aerial fauna through radio-wave remote sensing. *Biology Letters*, **8**, 698–701.
URL `https://royalsocietypublishing.org/doi/abs/10.1098/rsbl.2012.0384`

Chowdhury, A., D. Sheldon, S. Maji, and E. Learned-Miller, 2016: Distinguishing weather phenomena from bird migration patterns in radar imagery. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 10–17.

Dayhoff, J. E. and J. M. DeLeo, 2001: Artificial neural networks. *Cancer*, **91**, 1615–1635.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, 2009: Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 248–255.

Diehl, R. H. and R. P. Larkin, 2005: Introduction to the WSR-88D (NEXRAD) for ornithological research. Ralph, C. John; Rich, Terrell D., editors 2005. *Bird Conservation Implementation and Integration in the Americas: Proceedings of the Third International Partners in Flight Conference.* 2002 March 20-24; Asilomar, California, Volume 2 Gen. Tech. Rep. PSW-GTR-191. Albany, CA: U.S. Dept. of Agriculture, Forest Service, Pacific Southwest Research Station: p. 876-888, **191**.

Dieleman, S., K. W. Willett, and J. Dambre, 2015: Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, **450**, 1441–1459.
URL `http://dx.doi.org/10.1093/mnras/stv632`

Drozdzal, M., E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, 2016: The importance of skip connections in biomedical image segmentation. *CoRR*, **abs/1608.04117**.
URL `http://arxiv.org/abs/1608.04117`, Accessed on 7-27-2020.

Electron, 2020: Electron.
URL `https://github.com/electron/electron`, Accessed on 7-29-2020.

Fawcett, T., 2006: An introduction to ROC analysis. *Pattern Recognition Letters*, **27**, 861–874.

Fukushima, K., 1980: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**, 193–202.

Gauthreaux Jr., S. A. and C. G. Belser, 2003: Radar ornithology and biological conservation. *The Auk*, **120**, 266–277.

Girshick, R. B., 2015: Fast R-CNN. *CoRR*, **abs/1504.08083**.
URL `http://arxiv.org/abs/1504.08083`, Accessed on 7-27-2020.

Girshick, R. B., J. Donahue, T. Darrell, and J. Malik, 2013: Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, **abs/1311.2524**.
URL `http://arxiv.org/abs/1311.2524`, Accessed on 7-27-2020.

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

Han, S. Y., Y. Kim, S. H. Lee, and N. I. Cho, 2019: Pupil center detection based on the unet for the user interaction in VR and AR environments. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, IEEE, 958–959.

He, K., G. Gkioxari, P. Dollár, and R. B. Girshick, 2017: Mask R-CNN. *CoRR*, **abs/1703.06870**.
URL `http://arxiv.org/abs/1703.06870`, Accessed on 7-27-2020.

Helmus, J. and S. Collis, 2016: The python arm radar toolkit (py-art), a library for working with weather radar data in the python programming language. *Journal of Open Research Software*, **4**.

Hernández-García, A. and P. König, 2018: Data augmentation instead of explicit regularization. *CoRR*, **abs/1806.03852**.
URL `http://arxiv.org/abs/1806.03852`, Accessed on 7-27-2020.

Ioffe, S. and C. Szegedy, 2015: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, **abs/1502.03167**.
URL `http://arxiv.org/abs/1502.03167`, Accessed on 7-27-2020.

Jain, A. K., J. Mao, and K. M. Mohiuddin, 1996: Artificial neural networks: A tutorial. *Computer*, **29**, 31–44.

Johnson, G. D., W. P. Erickson, M. D. Strickland, M. F. Shepherd, D. A. Shepherd, and S. A. Sarappo, 2002: Collision mortality of local and migrant birds at a large-scale wind-power development on Buffalo Ridge, Minnesota. *Wildlife Society Bulletin*, 879–887.

Kelly, J. F. and S. M. Pletschet, 2018: Accuracy of swallow roost locations assigned using weather surveillance radar. *Remote Sensing in Ecology and Conservation*, **4**, 166–172.
URL `https://zslpublications.onlinelibrary.wiley.com/doi/abs/10.1002/rse2.66`

Kelly, J. F., J. R. Shipley, P. B. Chilson, K. W. Howard, W. F. Frick, and T. H. Kunz, 2012: Quantifying animal phenology in the aerosphere at a continental scale using NEXRAD weather radars. *Ecosphere*, **3**, 1–9.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012: Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105.

Lecun, Y., 1985: Une procedure d'apprentissage pour reseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). *Proceedings of Cognitiva 85, Paris, France*, 599–604.

Li, M., T. Zhang, Y. Chen, and A. J. Smola, 2014: Efficient mini-batch training for stochastic optimization. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 661–670.

Lin, T.-Y., K. Winner, G. Bernstein, A. Mittal, A. M. Dokter, K. G. Horton, C. Nilsson, B. M. Van Doren, A. Farnsworth, F. A. La Sorte, S. Maji, and D. Sheldon, 2019: Mistnet: Measuring historical bird migration in the us using archived weather radar data and convolutional neural networks. *Methods in Ecology and Evolution*, **10**, 1908–1922.
URL `https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13280`

Liu, R., J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, 2018: An intriguing failing of convolutional neural networks and the coordconv solution. *CoRR*, **abs/1807.03247**.
URL `http://arxiv.org/abs/1807.03247`, Accessed on 7-27-2020.

Milletari, F., N. Navab, and S. Ahmadi, 2016: V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR*, **abs/1606.04797**.
URL `http://arxiv.org/abs/1606.04797`, Accessed on 7-27-2020.

Muller, B. M., F. R. Mosher, C. G. Herbster, and A. T. Brickhouse, 2015: Aviation bird hazard in NEXRAD dual polarization weather radar confirmed by visual observations. *International Journal of Aviation, Aeronautics, and Aerospace*, **2**, 6.

Raudys, Š., 1998: Evolution and generalization of a single neurone: I. single-layer perceptron as seven statistical classifiers. *Neural Networks*, **11**, 283–296.

Redmon, J., S. K. Divvala, R. B. Girshick, and A. Farhadi, 2015: You only look once: Unified, real-time object detection. *CoRR*, **abs/1506.02640**.
URL `http://arxiv.org/abs/1506.02640`, Accessed on 7-27-2020.

Ren, S., K. He, R. B. Girshick, and J. Sun, 2015: Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, **abs/1506.01497**.
URL `http://arxiv.org/abs/1506.01497`, Accessed on 7-27-2020.

Ronneberger, O., P. Fischer, and T. Brox, 2015: U-net: Convolutional networks for biomedical image segmentation. *CoRR*, **abs/1505.04597**.
URL `http://arxiv.org/abs/1505.04597`, Accessed on 7-27-2020.

Rumelhart, D. E., G. E. Hintont, and R. J. Williams, 1986: Learning representations by back-propagating errors. *NATURE*, **323**, 9.

Santurkar, S., D. Tsipras, A. Ilyas, and A. Madry, 2018: How does batch normalization help optimization? *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, NIPS'18, 2488–2498.

Shelhamer, E., J. Long, and T. Darrell, 2016: Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**, 1–1.

Simonyan, K. and A. Zisserman, 2015: Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, eds.
URL `http://arxiv.org/abs/1409.1556`, Accessed on 7-27-2020.

Stepanian, P. M. and K. G. Horton, 2015: Extracting migrant flight orientation profiles using polarimetric radar. *IEEE Transactions on Geoscience and Remote Sensing*, **53**, 6518–6528.

Sudre, C. H., W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, 2017: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *CoRR*, **abs/1707.03237**.
URL `http://arxiv.org/abs/1707.03237`, Accessed on 7-27-2020.

Szegedy, C., S. Ioffe, and V. Vanhoucke, 2016a: Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, **abs/1602.07261**.
URL `http://arxiv.org/abs/1602.07261`, Accessed on 7-27-2020.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, 2016b: Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Troxel, S., M. Isaminger, B. Karl, M. Weber, and A. Levy, 2001: Designing a terminal area bird detection and monitoring system based on asr-9 data. *2001 Bird Strike Committee-USA/Canada, Third Joint Annual Meeting, Calgary, AB*, 25.

Van Den Broeke, M. S., 2013: Polarimetric radar observations of biological scatterers in hurricanes irene (2011) and sandy (2012). *Journal of Atmospheric and Oceanic Technology*, **30**, 2754–2767.

Weber, P., T. J. Nohara, and S. Gauthreaux Jr., 2005: Affordable, real-time, 3-d avian radar networks for centralized north american bird advisory systems. *2005 Bird Strike Committee-USA/Canada 7th Annual Meeting, Vancouver, BC*, 7.

Werbos, P. J., 1990: Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, **78**, 1550–1560.

Wilson, D. R. and T. R. Martinez, 2003: The general inefficiency of batch training for gradient descent learning. *Neural Networks*, **16**, 1429–1451.

Zaugg, S., G. Saporta, E. van Loon, H. Schmaljohann, and F. Liechti, 2008: Automatic identification of bird targets with radar via patterns produced by wing flapping. *Journal of The Royal Society Interface*, **5**, 1041–1053. URL `https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2007.1349`

Zhao, H., C. Yuan, P. Song, J. Ding, C.-L. Lin, Liang, and M. Zhang, 2019: Use of unmanned aerial vehicle imagery and deep learning unet to extract rice lodging. *Sensors*, **19**, 3859.

Zhao, Z., P. Zheng, S. Xu, and X. Wu, 2018: Object detection with deep learning: A review. *CoRR*, **abs/1807.05511**. URL `http://arxiv.org/abs/1807.05511`, Accessed on 7-27-2020.

Zhu, W., Y. Ma, M. G. Benton, J. A. Romagnoli, and Y. Zhan, 2019: Deep learning for pyrolysis reactor monitoring: From thermal imaging toward smart monitoring system. *AIChE Journal*, **65**, 582–591. URL `https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.16452`