NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPACT OF STOCHASTIC DEPTH ON DETERMINISTIC AND PROBABILISTIC RESNET MODELS FOR WEATHER MODELING

by

Cameron P. Woods

March 2022

Thesis Advisor: Marko Orescanin
Second Reader: Scott Powell

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE IMPACT OF STOCHASTIC DEPTH ON DETERMINISTIC AND PROBABILISTIC RESNET MODELS FOR WEATHER MODELING	**5. FUNDING NUMBERS**
6. AUTHOR(S) Cameron P. Woods	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis builds on the research that uses Bayesian neural networks to generate Global Precipitation Measurement Microwave Imager data collected by LEO satellites from Advanced Baseline Imager data collected by GEO satellites for the purposes of weather modeling. Specifically, this thesis investigates the efficacy of a stochastic depth (SD) implementation in residual networks (ResNet), both deterministic and probabilistic, to reduce long training times associated with Bayesian neural networks while maintaining model accuracy. We show that overall, ResNets fail to perform better with the implementation of SD with the exception of SD ResNet56 S25, utilizing a survivability probability of 0.25. This resulted in an RMSE of 2.863, a 6.83% increase in performance. In our evaluations, SD models did not train faster, with the fastest average time per epoch of 973.69 seconds compared to 960.42 seconds for the base ResNet56. We conclude that SD was unable to provide the expected performance benefits on realistic large-scale satellite data as found in research on smaller datasets.

14. SUBJECT TERMS stochastic depth, SD, residual networks, ResNet, deep neural networks, deterministic models, probabilistic models, Advanced Baseline Imager, ABI, Global Precipitation Measurement, GPM	**15. NUMBER OF PAGES** 63
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

IMPACT OF STOCHASTIC DEPTH ON DETERMINISTIC AND PROBABILISTIC RESNET MODELS FOR WEATHER MODELING

Cameron P. Woods
Lieutenant, United States Navy
BS, United States Naval Academy, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2022

Approved by: Marko Orescanin
Advisor

Scott Powell
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis builds on the research that uses Bayesian neural networks to generate Global Precipitation Measurement Microwave Imager data collected by LEO satellites from Advanced Baseline Imager data collected by GEO satellites for the purposes of weather modeling. Specifically, this thesis investigates the efficacy of a stochastic depth (SD) implementation in residual networks (ResNet), both deterministic and probabilistic, to reduce long training times associated with Bayesian neural networks while maintaining model accuracy. We show that overall, ResNets fail to perform better with the implementation of SD with the exception of SD ResNet56 S25, utilizing a survivability probability of 0.25. This resulted in an RMSE of 2.863, a 6.83% increase in performance. In our evaluations, SD models did not train faster, with the fastest average time per epoch of 973.69 seconds compared to 960.42 seconds for the base ResNet56. We conclude that SD was unable to provide the expected performance benefits on realistic large-scale satellite data as found in research on smaller datasets.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ABI	Advanced Baseline Imager
AI	Artificial Intelligence
BDL	Bayesian Deep Learning
CNN	Convolutional Neural Network
DOD	Department of Defense
GEO	Geostationary
GMI	GPM Microwave Imager
GPM	Global Precipitation Measurement
LEO	Low Earth Orbit
ML	Machine Learning
ResNet	Residual Network
TFP	TensorFlow Probability

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank Dr. Marko Orescanin, Lt Col Pedro Ortiz, and LT Micky Hall for their combined efforts and assistance throughout the course of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Currently, the Navy relies on several different satellites in varying orbits to provide the necessary data that is used for weather prediction models. One of the critical measurements is provided by Passive Microwave (PMW) Low Earth Orbit (LEO) satellites. Data assimilation systems, such as the Joint Effort for Data assimilation Integration (JEDI) system, utilize PMW satellite data in numerical models of the atmosphere and produce the best possible forecast given other observations of the atmosphere. Weather forecasting is an essential piece of the operational planning puzzle, and the Navy recognizes the need to constantly push for advancement in prediction capabilities [1]. Ortiz et al. [2] speculate that providing PMW information to JEDI systems at much higher temporal rates can lead to significantly improved forecasting capability. Hence, this work builds on that research [3] where authors develop synthetic PMW information via Bayesian Deep Learning from readily available ABI data. With the vast amount of data to be processed, the only way forward is through artificial intelligence (AI) and machine learning (ML). Deep Learning and Neural Networks are already being employed in weather data centers to assist in processing and modeling satellite data for weather prediction [4].

One of the most effective ways to improve a machine learning model's accuracy is to increase the amount of data the model has available on which to train [5]. To this end, advanced baseline imager (ABI) data from satellites in a geostationary orbit (GEO) can be used to predict passive microwave (PMW) data that is collected by satellites in a low Earth orbit (LEO). LEO satellites orbit the Earth very quickly, while GEO satellites always stay over the same geographic location, meaning that by comparison, LEO satellite data has a very low temporal resolution (i.e., it's relatively infrequent.) Using AI to artificially predict PMW data will increase the temporal resolution, thus improving the accuracy of the Navy's weather prediction models.

To be effective and useful for fusion with operational models, these predictions need to include not just the estimate but also be able to quantify uncertainty in the prediction. It is also possible to decompose the uncertainty into the aleatoric and epistemic uncertainties as well,

which can be provided by a Bayesian neural network [2], [6]. Aleatoric uncertainty quantifies the uncertainty that is a result of noise in the input data, while epistemic uncertainty is due to the model bias relative to the data. Given enough data epistemic uncertainty is explainable hence the model can be improved. Both are needed for accurate weather prediction, so this thesis will focus on a Bayesian approach. Previously, Orescanin et al. [7] and Ortiz et al. [8] have demonstrated uncertainty quantification and decomposition for the classification of precipitation.

Bayesian neural networks, while accurate, can often suffer in the consumption of computing resources when compared to traditional deterministic models. To that end, this thesis will apply a stochastic depth architecture as employed by Huang et al. [9]. Huang et al. use a stochastic depth approach to effectively reduce training time for their model, without a loss in model fidelity. They tested their model on the CIFAR10 [10], CIFAR100 [10], SVHN [11], and ImageNet [12] datasets, showing both a reduction in error as well as a significant computational speedup when compared to 12 other competing model architectures. [9] Their research, however, applied stochastic depth only to a deterministic Residual Network (ResNet) model, not a Bayesian one, and they were strictly modeling for classification, not regression. This thesis will attempt to repeat their findings utilizing ABI data with a deterministic ResNet model but utilizing regression, not classification, as well as apply stochastic depth to a Bayesian ResNet model for the same PMW data prediction. This builds on the works of Orescanin et al. [7] and Ortiz et al. [2], [8] where it was demonstrated that ResNet style architectures, both deterministic and Bayesian, can be utilized to develop state-of-the-art classification models with satellite data.

1.1 Research Objectives and Contributions

This work aims to build on methods by which Global Precipitation Measurement (GPM) Microwave Imager (GMI) data from LEO satellites can be synthetically created by using Machine Learning (ML) to train on Advanced Baseline Imager (ABI) data from GEO satellites. This approach produces very large supervised learning datasets (more than one million input features) and expected model training and development times are long. Given that, our specific goal is to evaluate stochastic depth ResNet model architecture which could lead to faster training times. Additionally, another goal of the thesis is to understand the quality of uncertainty quantification with stochastic depth ResNet models for regression applications.

GMI data is superior for use in weather modeling, as the lower altitude of the satellite and the nature of the data provides an increased depth and granularity [13], as well as penetrates cloud cover. However, what the GMI data lacks is a high temporal resolution; LEO satellites move quickly across the surface of the Earth, orbiting in the vicinity of every 90 minutes [14]. This, in turn, means that at any given time, a Low Earth Orbit (LEO) satellite does not have much collection time over any specific geographic location, leading to large gaps in GPM Microwave Imager (GMI) data collection and therefore limited effectiveness for weather modeling. ABI data, while not as high quality for modeling purposes, have perfect temporal resolution as Geostationary (GEO) satellites persistently collect over the same geographic location. Using Convolutional Neural Networks (CNNs), the temporal resolution of GMI data can be synthetically increased by training a network on the higher resolution ABI data.

1.2 Organization

This thesis is organized into four additional chapters. Chapter 2 introduces CNNs, Residual Networks (ResNets), and Stochastic Depth. Chapter 3 discusses the dataset and CNN models used in this thesis, the research methodology, and Stochastic Depth implementation details. Chapter 4 examines the study's outcomes, including metrics, observations, efficiency, and accuracy. Chapter 5 discusses the research conclusions, future work, and final observations.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

This chapter explores several vital technical concepts relevant to neural networks, ResNets, and Stochastic Depth utilized throughout this work.

2.1 Convolutional Neural Networks

Neural networks, at their core, work just like the neural connections in our brains [15]. Neural networks utilize many artificial neurons, which are typically referred to as nodes, with each node having an input, a weight, and bias as well as an activation function that produces an output [16]. This collection of elements is called a perceptron [15]. In neural networks, the output of one perceptron is then used as the input of another, as shown in Figure 2.1. All these perceptrons linked together create what is called a layer. A common neural network will have an input layer, an output layer as well as any number of hidden layers in the middle [17], as visible in Figure 2.2.

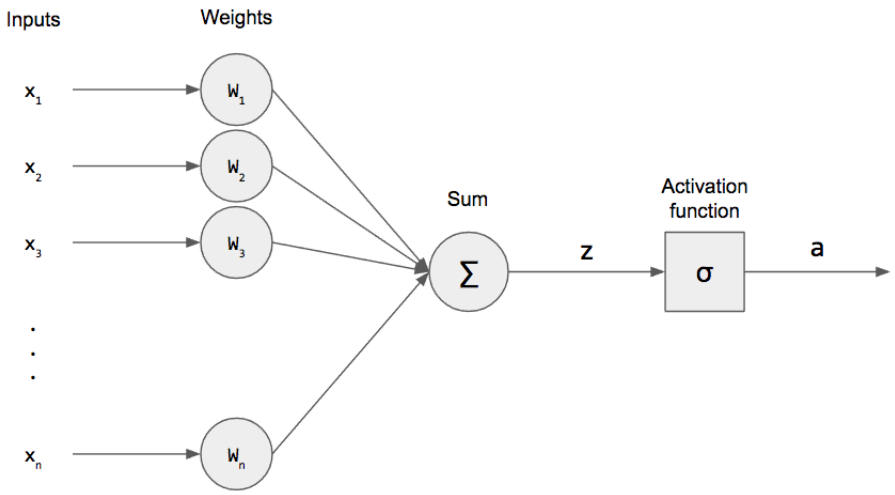

Figure 2.1. Basic Perceptron. Source: [18].

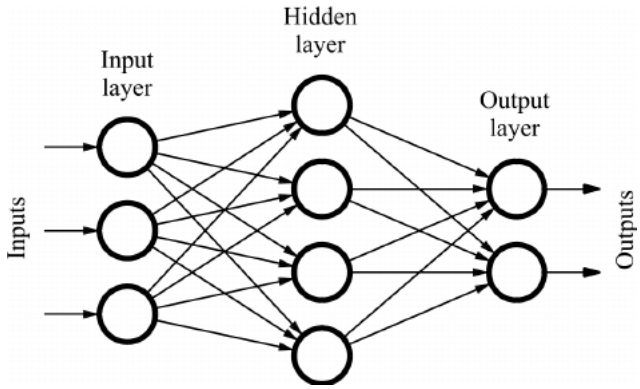

Figure 2.2. Diagram of a Simple Neural Network with Hidden Layers. Source: [19].

Depending on the network architecture there can be any number of nodes in a layer, however, the output layer will typically correlate to the number of classes the network is built to predict [17]. In the case of regression, as is the focus of this thesis, the output of a deterministic neural network is a single neuron with linear activation [15].

CNNs are a specific subset of neural networks and often are used for image recognition tasks, making them ideal for weather prediction modeling explored in this thesis [7], [17]. A traditional neural network falls short in image recognition tasks because there is no connection between nodes that are in the same layer, only between layers. In image recognition, a traditional neural network would have each pixel of an image corresponding to an input in the input layer. If these inputs have no connection to one another, then it can be very easy to lose spatial features in the image. This is because pixels next to or near each other in an image are much more likely to be correlated, and without any logical connection between the nodes in the neural network, this correlation is lost [17].

To achieve this necessary spatial correlation, convolutional neural networks instead will feed an entire patch of pixels (kernel-based approach) onto the input of a single node, thus preserving the spatial correlation between these pixels [17]. This downsampling of the image can be achieved in several ways, a common one being to take the average value of all the pixels in the patch and have that average value passed on as the single input value to the next layer. Another method is to apply a kernel, which is a small array of numbers, and calculate an element-wise product between each element in the input patch of the image

and the kernel, and then sum all those values together to be the output [20]. These kernels are typically 3x3 in size, though 5x5 and 7x7 are also sometimes used. Figure 2.3 shows the architecture of a convolutional neural network, with a 3x3 kernel size [21].

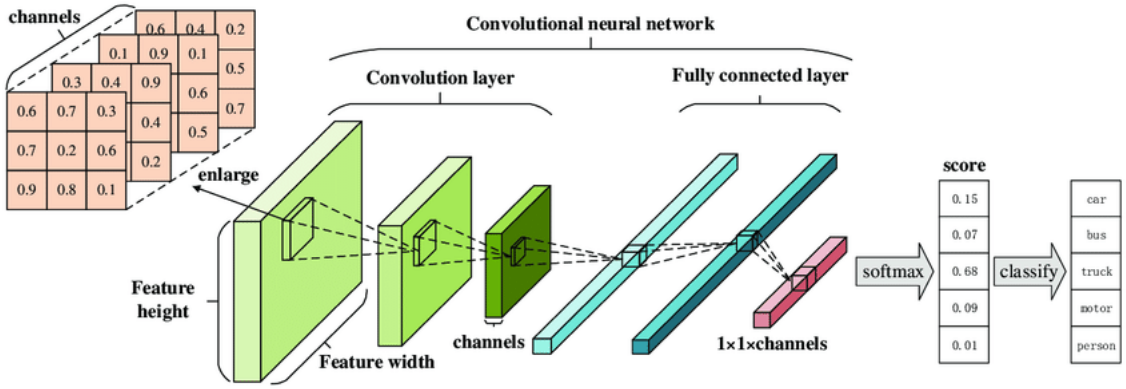

Figure 2.3. Diagram of CNN Architecture. Source: [21].

A hurdle that this method of kernel feature extraction runs into is that the central element of the kernel cannot reach the outermost pixels of the input; the kernel cannot exceed the bounds of the input. To get around this, most CNN architectures will introduce zero padding, which adds the necessary number of zeros around the input image to allow the center of the kernel to reach the edge of the actual image [20]. Without this, the number of convolutions possible on an image would be limited, as the output would be smaller with each successive convolution.

2.2 Deep Residual Networks (ResNet)

The deep residual network, or ResNet, model was first introduced by He et al. [22] to counter a common issue when attempting to create deeper CNNs, effectively increasing the non-linear modeling ability of the architecture. As more layers are added to a CNN in an attempt to capture more image complexity, the performance degrades in what is called the vanishing gradient problem, resulting in increased training error as layer depth is increased [22]. He et al. introduced what are called residual blocks that effectively shortcut connections, aptly named “skip connections." These skips conduct identity mapping, with

their outputs then added to the output of the layers [23]. What this effectively achieves is that if a layer negatively impacts the performance of the model, it then is skipped by regularization (see Figure 2.4).

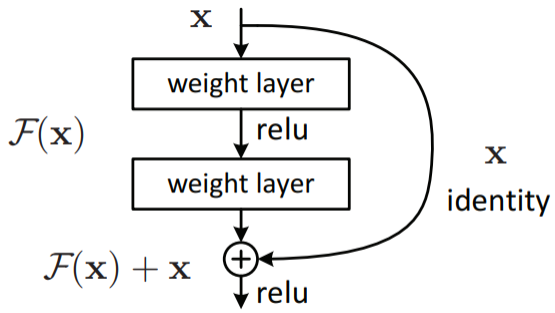

Figure 2.4. Residual Network (ResNet) Residual Block Skip Connection. Source: [23].

$$\mathrm{H}_l = ReLU(f_l(H_{l-1}) + id(H_{l-1})) \tag{2.1}$$

In the Equation 2.1, H_l represents the output of the l^{th} layer, and f_l is the "convolutional transformation from layer $l-1$ to l" [9]. id is the identity transformation, and a ReLU activation function is assumed. Figure 2.5 shows an example of a Residual Block, or ResBlock, where

> ...when the output dimensions of f_l don't match those of H_{l-1}, the authors redefine id() as a linear projection to reduce the dimensions of id(H_{l-1}) to match those of $f_l(H_{l-1})$. The propagation rule in (2.1) allows the network to pass gradients and features (from the input or those learned in earlier layers) back and forth between the layers via the identity transformation id(). [9]

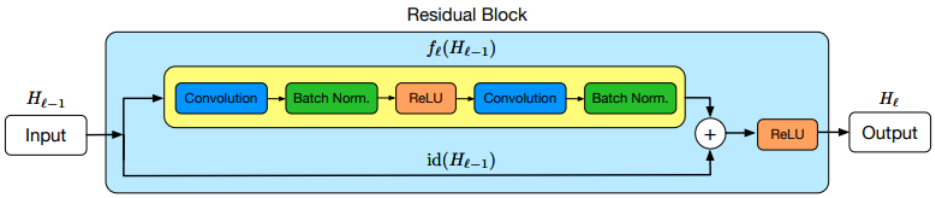

Figure 2.5. Residual Block within a ResNet. Source: [9].

Layers	**Num Params**	**Error(%)**
20	0.27M	8.75
32	0.46M	7.51
44	0.66M	7.17
56	0.85M	6.97
110	1.7M	6.43
1202	19.4M	7.93

Table 2.1. Classification Error of ResNets with Differing Number of Layers on the CIFAR-10 Test Set. Source: [23].

As can be seen in Table 2.1, the performance of ResNet CNNs improves up through 110 layers, something that would not be achievable in a standard CNN. These results from [23] helped us identify what ResNet architecture to select for our testing. We found it prudent to build upon their work, rather than rebuild a new ResNet model from the ground up. As can be seen by the data, ResNet110 performed the best with the lowest classification error, however, ResNet56 was quite close, within 0.2%, and trained considerably faster as it is roughly half the depth of ResNet110. This made ResNet56 the primary candidate for this thesis, though ResNet110 was also tested for comparison.

ResNet models are considered state-of-the-art and have been used in numerous applications, however, they have come to be known particularly well in computer vision and image classification applications [23]. Many leading scientists turn to ResNets for the most reliable results when handling sensitive data, where accuracy is paramount, for example authors Sarwinda et al. [24] utilized ResNets to great effect in the medical imaging classification for the detection of colorectal cancer, and Yang et al. [25] relied on a ResNet architecture for detection of COVID-19 in medical images. Proven success with the ResNet architecture for critical computer vision problems makes it ideal for our weather modeling implementation.

2.3 Stochastic Depth

While ResNets have their strengths, they also have their limitations. In a world of limitless resources and time, one could simply keep adding depth and complexity to their ResNets to eke out more and more performance, however marginal each subsequent improvement was to be. In reality, machine learning is a constant battle of time and resource management balanced against quality results. Toward that end, the implementation of stochastic depth in ResNets has been shown to yield positive results [9]. The purpose of stochastic depth is to effectively reduce the depth of a model during training while retaining its depth during testing. This, in turn, can allow for a considerable reduction in training time, without losing the quality of a sufficiently deep network for testing. When added to residual networks, the implementation of stochastic depth resulted in significantly shorter training time [9] [26].

As shown by Huang et al. [9], the way stochastic depth achieves this depth reduction is by randomly dropping entire ResBlocks during the training of the model. Similar to (1), let b_l be a Bernoulli random variable, indicating whether the l^{th} ResBlock is active ($b_l = 1$) or inactive ($b_l = 0$). From this, a slightly modified output equation of

$$\mathrm{H}_l = ReLU(b_l f_l(H_{l-1}) + id(H_{l-1})). \tag{2.2}$$

When $b_l = 1$, Equation (2.2) reduces down to the normal ResNet output equation seen in (2.1). However, when $b_l = 0$, the ResBlock instead becomes just the identity function

$$\mathrm{H}_l = id(H_{l-1}). \tag{2.3}$$

The value of the Bernoulli random variable b_l is dictated by what's called the survival probability p_l, which aptly named, indicates the probability that a ResBlock is dropped. While Huang et al. recommended a $p_l = 0.5$, a range of probabilities were tested for this thesis [9].

2.3.1 Original Implementation

It is important to discuss Huang et al.'s original implementation of stochastic depth to better understand both their results as well as our results in comparison. Huang et al. used a slightly modified ResNet110 architecture [9], building off of the same architecture from He et al. [23] that we did for this thesis. The only modification made for their testing of the ResNet110 with stochastic depth was that they included a dense layer with 100 with a softmax activation, and this was only for specifically testing on the CIFAR-100 dataset [9].

Original Datasets

Huang et al. tested their original stochastic depth implementation against a collection of widely-used datasets, namely CIFAR-10, CIFAR-100, SVHN, and ImageNet, as listed in Chapter 1. CIFAR-10 is a dataset of 32x32 color images, divided into 10 classes of "natural scene objects [9]." The dataset contains 50,000 training images and 10,000 testing images. For their testing, Huang et al. retained 5,000 of the training images for validation, with the remaining 45,000 used for training. CIFAR-100 is very similar to CIFAR-10, with it comprised of 32x32 color images, with 50,000 training images and 10,000 testing images. The primary difference between the two datasets is while CIFAR-10 has 10 classes, CIFAR-100 is comprised of 100 classes, evenly represented in the dataset with 600 images per class [9].

SVHN, like Canadian Institute for Advanced Research (CIFAR)-10 and CIFAR-100, is a dataset of 32x32 color images. This dataset specifically is entirely comprised of cropped house numbers from Google Street View [9], the objective being to classify the digit at the center of the image. SVHN is larger than CIFAR-10 and CIFAR-100, with 73,257 digits in the training set, 26,032 in the testing set, with an additional 531,131 "easier" digits for additional training if necessary [9].

The final dataset used by Huang et al., the ILSVRC 2012 classification dataset, or ImageNet, is a much larger, and the authors deliberately set it apart from the first three in their analysis [9]. ImageNet contains 1000 unique classes, with 1.2 million training images, 50,000 validation images, and 100,000 testing images. For ImageNet, Huang et al. changed their testing architecture, moving to a 152-layer ResNet [9], following He et al [23].

Original Results

We will first discuss their training time results for the first datasets, as the author does not include results for ImageNet, due to complications with the immense size of the datset, which we will discuss later in this section.

	CIFAR10+	CIFAR100+	SVHN
Constant Depth	20h 42m	20h 51m	33h 43m
Stochastic Depth	15h 7m	15h 20m	25h 33m

Figure 2.6. Huang et al. Training Time Results. Source: [9]

As seen in Figure 2.6, Huang et. al saw a significant speed-up in training time with the addition of stochastic depth to their ResNets across all three of the relatively similar datasets.

	CIFAR10+	CIFAR100+	SVHN	ImageNet
Maxout [21]	9.38	-	2.47	-
DropConnect [20]	9.32	-	1.94	-
Net in Net [24]	8.81	-	2.35	-
Deeply Supervised [13]	7.97	-	1.92	33.70
Frac. Pool [25]	-	27.62	-	-
All-CNN [6]	7.25	-	-	41.20
Learning Activation [26]	7.51	30.83	-	-
R-CNN [27]	7.09	-	1.77	-
Scalable BO [28]	6.37	27.40	1.77	-
Highway Network [29]	7.60	32.24	-	-
Gen. Pool [30]	6.05	-	1.69	28.02
ResNet with constant depth	6.41	27.76	1.80	21.78
ResNet with stochastic depth	5.25	24.98	1.75	21.98

Figure 2.7. Huang et al. Model Performance Results. Source: [9]

Figure 2.7 shows Huang et al.'s overall model performance results, with the test error (%) used as the metric for comparison [9]. As can be seen, ResNets with and without stochastic depth are highlighted, with several other competitive benchmark models included as well. A conventional ResNet already performs quite well when compared to many of these benchmarks, however, the inclusion of stochastic depth generates the best performance without exception for CIFAR-10, CIFAR-100, and SVHN, with 5.25%, 24.98%, and 1.75% test error, respectively. These results do not carry over to ImageNet however, failing to beat the conventional ResNet's results of 21.78%, though still much better than the other benchmark results available at 21.98%.

The authors posit that the size and complexity of ImageNet may require a much deeper model than the 152-layer ResNet they used and that unfortunately computing resource availability ended up being a bottleneck for their efforts with ImageNet [9].

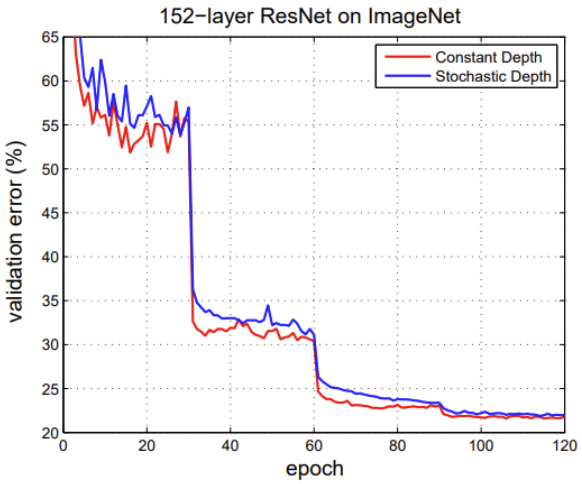

Figure 2.8. Huang et al. ImageNet Results. Source: [9]

They maintained that while stochastic depth showed no improvement over the conventional ResNet with ImageNet, as seen in depth in Figure 2.8, stochastic depth may still yield positive results on ImageNet with deeper architectures, especially as GPU capacity continues to grow and more computing resources become available to the community at large [9].

2.4 Bayesian Deep Learning

This thesis will also explore stochastic depth in probabilistic ResNet models and deterministic models. Probabilistic modeling is achieved namely through Bayesian Deep Learning and this is a novel and unique evaluation. A probabilistic approach through Bayesian Deep Learning allows the model to not only provide a prediction on the data but also uncertainty information about that prediction. To create this conversion from a deterministic model to a probabilistic one, the weights of the CNN, which we will call θ, are instead replaced with a distribution over weights. A core principle of Bayesian statistical analysis is “prior beliefs influence posterior beliefs" [27]. Applying Bayes theorem to our distributions becomes

$$P(\mathbf{w}|D) = \frac{P(D|\mathbf{w}) \cdot P(\mathbf{w})}{P(D)} = \frac{P(D|\mathbf{w}) \cdot P(\mathbf{w})}{\int P(D|\mathbf{w}) \cdot P(\mathbf{w})\, d\mathbf{w}} \tag{2.4}$$

In this equation, $P(D|\mathbf{w})$ represents what is called the likelihood; $P(\mathbf{w})$ is the prior belief,

which is what is updated as the model trains due to the introduction of D. $P(\mathbf{w}|\mathrm{D})$ is the posterior belief, which encodes the epistemic uncertainty, which is the uncertainty due to the lack of data available. $P(\mathrm{D}|\mathbf{w}) \cdot P(\mathbf{w}) = P(\mathrm{D}, \mathbf{w})$, which is the joint probability of D and $\mathbf{w}$. These two combined account for both uncertainty types necessary for probabilistic modeling [27].

2.4.1 Flipout

As developed by Wen et al. [28], Flipout is a regularization technique for probabilistic modeling. Regularization is a key element of neural network architectures, and "[the] most widely used regularization methods are based on randomly perturbing a network's computations" [28]. The authors go on to expand that typically regularization methods will perturb a network's activations, while others perturb weights, as is the case with Bayesian neural networks. Perturbing weights, however, comes with a cost: resources. There are (typically) many more weights than there are units in a neural network, so to compute the weight perturbation for every single element in a mini-batch becomes incredibly costly on computational resources [28]. As a result, methods that are regularized by weights will usually only use a single sample per mini-batch. [28].

In contrast, when perturbing activations, "the training algorithm [is able] to see orders of magnitude more perturbations in a given amount of time, and the variance of the stochastic gradients decays as $1/N$, where N is the mini-batch size" [28]. This variance rate of decay is ideal, and Bayesian neural networks are locked out of this efficiency due to the use of weight perturbation. To combat this issue, Wen et al. developed the Flipout method, which they describe as

an efficient method for decorrelating the gradients between different examples without biasing the gradient estimates. Flipout applies to any perturbation distribution that factorizes by weight and is symmetric around 0–including DropConnect, multiplicative Gaussian perturbations, evolution strategies, and variational Bayesian neural nets–and to many architectures, including fully connected nets, convolutional nets, and RNNs. [28]

2.4.2 Uncertainty Estimate with Variational Inference

The purpose of employing a probabilistic model is to provide uncertainty about the results, as well as the results themselves, as discussed in Section 2.4. To achieve this, first, we must make inferences on our model. We make a prediction, which we will call $\hat{y}_n$, for every new input, which we will call x_n. We then generate T predictions for every x_n by using the Monte Carlo integration with T samples of the weight distribution [29] [30]. For this thesis, we arbitrarily chose $T = 100$, and therefore made 100 predictions which we will call $\hat{y}_t$ for every input and from them calculated $\hat{y}_n$ via the following equation:

$$\hat{y}_n = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t(x_n, \mathbf{w}_t) \tag{2.5}$$

To then generate variance, which can then provide a measure of uncertainty, we have to understand that every $\mathbf{w}_t$ is different because the model weights are distributions. This makes $\hat{y}_n$ an average predicted value across that distribution. To calculate variance of $\hat{y}_n$, we use the same initial T predictions [31]:

$$Var(\hat{y}_n) = \frac{1}{T} \sum_{t=1}^{T} (\hat{y}_t(x_n, \mathbf{w}_t) - \hat{y}_n)^2 \tag{2.6}$$

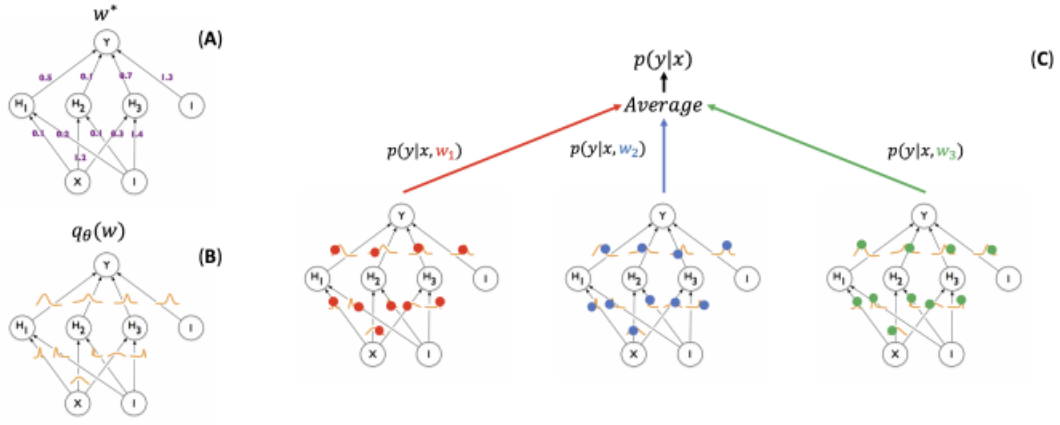

Figure 2.9. Visual Representation of Monte Carlo Sampling. Source: [32]

Figure 2.6 shows a visual representation of the Monte Carlo sampling from McClure et al's paper on Bayesian deep learning [32]. Image (A) in the top left shows a traditional neural network, with one weight per connection. (B) represents a Bayesian neural network, with posterior distributions instead of constant weights. The majority of Figure 2.6 shows how Monte Carlo sampling is constructed (C), whereby

> [the] process of training starts with an assigned prior distribution for each weight, and returns an approximate posterior distribution. At test time (C, right), a weight sample $\mathbf{w}_1$ (red) is drawn from the posterior distribution of the weights, and the resulting network is used to generate a prediction $p(y|x, \mathbf{w}_1)$ for an example x. The same can be done for samples $\mathbf{w}_2$ (blue) and $\mathbf{w}_3$ (green), yielding predictions $p(y|x, \mathbf{w}_2)$ and $p(y|x, \mathbf{w}_3)$, respectively. The three networks are treated as an ensemble and their predictions averaged [32]

It is important to note, as will be seen in Chapter 4, that variance calculation is only relevant for probabilistic models. When making the $T = 100$ predictions on a deterministic model, those 100 predictions will all be identical, and thus the variance will be 0. This makes sense, as we are using variance as an indicator of uncertainty, which deterministic models do not provide.

CHAPTER 3:
Methodology

This chapter discusses the GMI and ABI datasets, the CNN models used in this thesis, and the research methodology.

3.1 Datasets

This thesis is built around GMI and ABI datasets. GMI data, coming from LEO satellites, is preferred for weather modeling due to the better fidelity afforded by GMI's ability to penetrate cloud cover compared to a GEO satellite. PMW satellites collect 13 channels of microwave data, through a range of frequencies from 10.6 GHz to 183 GHz, as seen in Table 3.1. Different channels are better equipped to capture various weather events than others, e.g., the lower channels (1-5) are more sensitive to moderate to heavy rainfall, while the upper bands are more sensitive to water vapor in the atmosphere. The best weather modeling comes from utilizing information from across the various channels in order to leverage the inherent strength of each frequency range [33]. It is this GMI data that we aim to synthetically produce to increase the inherently poor temporal resolution of data collected by LEO satellites.

Table 3.1. GMI Technical Summary

Band	GHz	Polarization
1	10.6	Vertical
2	10.6	Horizontal
3	18.7	Vertical
4	18.7	Horizontal
5	23	Vertical
6	37	Vertical
7	37	Horizontal
8	89	Vertical
9	89	Horizontal
10	166	Vertical
11	166	Horizontal
12	183±2	Vertical
13	183±7	Vertical

To account for the poor temporal resolution of GMI data, we rely on ABI data collected from GEO satellites, specifically GOES-16 and GOES-17, as GEO satellites have terrific temporal resolution because they are constantly collecting from the same patch of Earth, depending on where their orbit has been placed around the equator. ABI data collected from GOES satellites is split into 16 bands, as seen in Table 3.2. Similar to GMI data, different bands are more equipped to capture specific atmospheric conditions over others, so it is important to leverage a range of bands.

Table 3.2. ABI Technical Summary

ABI Band	**Central Wavelength(μm)**	**Type**	**Instantaneous Geometric Field of View (km)**
1	0.47	Visible	1
2	0.64	Visible	0.5
3	0.86	Near-Infrared	1
4	1.37	Near-Infrared	2
5	1.6	Near-Infrared	1
6	2.2	Near-Infrared	2
7	3.9	Infrared	2
8	6.2	Infrared	2
9	6.9	Infrared	2
10	7.3	Infrared	2
11	8.4	Infrared	2
12	9.6	Infrared	2
13	10.3	Infrared	2
14	11.2	Infrared	2
15	12.3	Infrared	2
16	13.3	Infrared	2

Figure 3.1 shows a visualization of the 16 ABI input channels [34]. While we are initially ingesting 16 channels, we chose not to utilize channels 1-6, and instead exclusively use infrared data. Using infrared data meant that we could use the data to run our models across a 24-hour timeperiod, without needing to take into account the lack of visible and near-infrared data during night time, it was an unnecessary level of complexity that was better avoided than worked around.

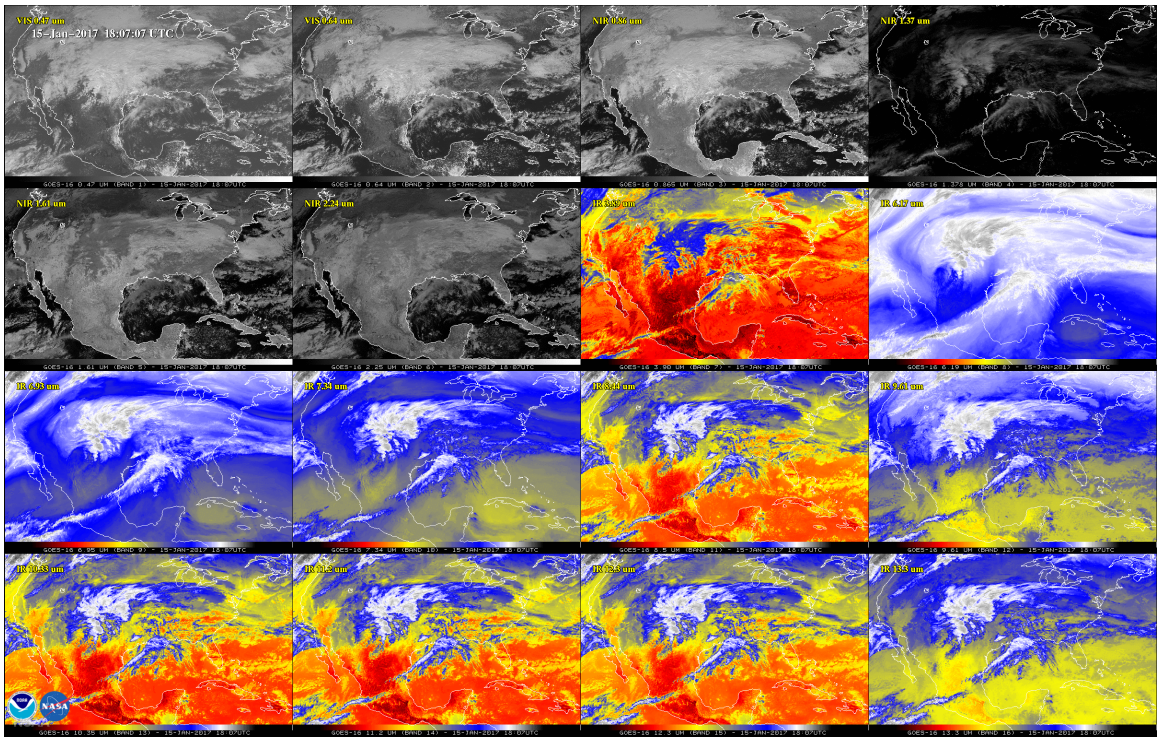

Figure 3.1. Input Channel Visualization. Source: [34].

This research was further restricted to data collected over water due to inherent limitations in GMI data. The GMI sensors on LEO satellites are heavily influenced by the emissivity of ground surfaces, which is very difficult to account for without information about specifics such as soil moisture content, soil composition, etc. We concluded that it was outside of the scope to attempt modeling over both land and sea, and instead focused on just data over sea. This was reinforced by the Navy being the primary target of this research, and modeling capabilities for only sea-covered sections of the planet are more relevant to the Navy's needs, and therefore a good place to begin with our research.

3.1.1 Initial Dataset

Given the initial GMI and ABI observations, they were first collocated by date and geolocation. We then generated patches of the first 11 bands of ABI data of size 39x39, and labelled them with the collocated GMI center pixel. These label-feature pairs included other relevant information as well, namely the latitude and longitude, viewing angle, and a flag

matrix of the same size as the ABI patch that indicated if any of the patch was over land or over ocean. As discussed in the previous section, any sections that were centered over land were discarded as well as any records that did not allow the inclusion of a full 39x39 patch of ABI data. This is important as the data we are working with, when viewed as a picture of the earth they cover, are not neat lines, but rather curved swaths across the earth, tracing the path of the satellite as it orbits. This leads to a number of data points along the edges of these swaths that cannot be used, and were therefore discarded.

The entire dataset spanned 6 months, from January 2020 to June 2020. For training purposes, we primarily used data from January and February only, as utilizing the entire dataset would have been an unnecessary strain on computing resources given the quantity of data available. The data was further broken up into training, testing, and validation datasets. Three entire days were chosen at random from both January and February to be a validation dataset, and 3 additional days were then used as a testing dataset, as seen in the table below. The rest of the data then became the training dataset.

Table 3.3. Validation and Test Days by Month

Month	**Validation Days**	**Test Days**
January	6, 13, 19	8, 21, 26
February	4, 11, 27	7, 16, 24

3.1.2 Data Sampling

To account for naturally occurring imbalances in both the GMI and ABI data, we conducted a measure of data sampling to help prevent bias in the model towards the data that occurs the most frequently. This, in turn, helped create a more balanced dataset, boosting the amount of GMI data that was the least frequently present in the dataset. For each GMI band, a randomly selected 65% of the most frequently occurring temperature labels were ultimately included in the training set.

3.2 Experiment Methodology

The aim of this thesis primarily was to test the introduction of stochastic depth compared to both deterministic and probabilistic models training on the weather data discussed above. To that end, ResNet V2 [22] models, one with 56 layers, and one with 110 layers were used, both with and without stochastic depth introduced. Survival probabilities as discussed in Chapter 2.3 were also tested on the ResNet56 with stochastic depth, examining a survivability of 0.25, 0.5, and 0.75. In their original paper, Huang et al. [9] did a test of survival probabilities ranging from 0.1 to 1.0, and concluded that a probability of 0.5 was optimal [9]. However, they also concluded that it is still a hyper parameter worth investigating, which is why we also included a test of 0.25 and 0.75. The probabilistic model testing used a Bayesian ResNet with an identical configuration as the deterministic ResNet56 used above, implementing Flipout layers [28]. Lastly, in an attempt to test stochastic depth implementation under varying conditions, a test was done comparing the deterministic ResNet56 with and without stochastic depth, but specifically trained with only 1 GPU each, with a batch size of 1024. For all other testing, training parameters were identical, to ensure no change from run to run so as to guarantee accurate compute time results as well as model results. To implement stochastic depth, the Stochastic Depth layer within Tensorflow Addons [35] was used, as it was explicitly built as an implementation of stochastic depth as laid out by Huang et al [9].

3.3 Loss Functions Evaluated

3.3.1 Deterministic Loss Function

Mean squared error (MSE) is perhaps one of the simplest and most common loss functions, and we employed it here to calculate loss and train the weights of our deterministic models accordingly:

$$\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y_i - \hat{y}_i)^2 \tag{3.1}$$

where N is the number of predictions, $\hat{y}$ is the predicted GMI data and y is the correct label for GMI data against which we are testing. MSE takes the difference between our predicted values and the real values for each prediction, squares them, and then averages all of these new values to one mean squared error, representing how close or far the model's prediction

was against the real value.

3.3.2 Bayesian Loss Function

For Bayesian models, the approach to loss is a bit more complex. First we must return to Equation 2.4, the posterior distribution, P(**w**|D), over the model's parameters. The problem with this equation as written is that the denominator frequently has no closed form solution [36], and we must instead find an approximation for P(**w**|D). To help get around this problem, we will rely on variational inference, which can be used to approximate the posterior distribution. This is achieved through an optimization problem, which identifies the parameters, θ, of a distribution within a larger set of distributions, $q_\theta(\mathbf{w}) \in Q$, with the smallest Kullback-Leibler divergence (KL) [37] from our target distribution, P(**w**|D).

$$KL(q_\theta(\mathbf{w}) \mid\mid P(\mathbf{w} \mid \mathrm{D})) = \int q_\theta(\mathbf{w}) \log \frac{q_\theta(\mathbf{w})}{P(\mathbf{w} \mid \mathrm{D})} d\mathbf{w} \tag{3.2}$$

This is a step in the right direction; however, we are still left with P(**w**|D), which we need to approximate in order to make this equation practically solvable. To do this, we will utilize the evidence lower bound (ELBO), which can serve as a computationally tractable substitute for this optimization problem [38]. The equation then becomes:

$$KL(q_\theta(\mathbf{w}) \mid\mid p(\mathbf{w} \mid \mathrm{D})) = \log\ p(\mathrm{D}) - \underbrace{\int q_\theta(\mathbf{w}) \log \frac{p(\mathbf{w})p(\mathrm{D} \mid \mathbf{w})}{q_\theta(\mathbf{w})} d\mathbf{w}}_{\text{Evidence Lower Bound (ELBO)}} \tag{3.3}$$

To solve the optimization problem from here, we then maximize the ELBO, which results in the following optimization problem [38]:

$$\theta^* = \arg\max_\theta \int q_\theta(\mathbf{w}) \log \frac{p(\mathbf{w})p(\mathrm{D} \mid \mathbf{w})}{q_\theta(\mathbf{w})} d\mathbf{w} \tag{3.4}$$

3.4 Metrics

To evaluate model performance, we measured Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), as benchmarks for predictive error.

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\mathrm{MSE}(y, \hat{y})} = \sqrt{\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (y_i - \hat{y}_i)^2} \tag{3.5}$$

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} |y_i - \hat{y}_i| \tag{3.6}$$

In the above equations, y is the set of GMI label data and $\hat{y}$ is the set of average predicted GMI data. The inclusion of Mean Absolute Percentage Error (MAPE) introduces a scaled comparison of the error, making it a useful metric to truly compare model to model.

$$\mathrm{MAPE}(y, \hat{y}) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \frac{|y_i - \hat{y}_i|}{max(\epsilon, |y_i|)} \tag{3.7}$$

As above, y is the set of GMI label data, and $\hat{y}$ is the set of predicted GMI data, while ϵ is an arbitrarily small positive number; approximately 2e-16 as dictated by the Scikit-learn library, which was used to calculate all error metrics for this work. [39]

As for comparing the computational performance of each trained model, the primary metrics used were time/epoch and time/step. As different bands had different amounts of data in them, time/epoch serves only as a general birds eye view of the performance, because not all the epochs for every run were of identical size. Time/step on the other hand, allows us to accurately compare the computational time from model to model, across various bands. To ensure that we conduct fair benchmarking we utilized the same hardware configuration for the tests and an identical training schedule. Every model we ran used the Adam optimizer [40], with an initial learning rate of 0.001. All training was set to run for 500 epochs. We monitored validation loss while training, and if after 5 epochs the validation loss had not improved, we reduced the learning rate by a factor of 4. All models were trained on 4 NVIDA RTX 8000 48GB GPUs, with a batch size of 2048. We utilized an early stopping patience of 50, meaning that if after 50 epochs the validation loss had not improved, the model finished early.

CHAPTER 4:
Results

This chapter presents the results of testing stochastic depth implementation in both deterministic and probabilistic models, as discussed in Chapter 2. Results are presented in two main sections. The first section will be an examination of how the various models performed, both with and without the implementation of stochastic depth. Model performance is paramount, as it does not matter how quickly a model can train if the predictions it delivers are sub-par. This model performance will be determined by examining the primary metrics discussed in Chapter 3.3. The second section will look at the computing performance of the various models, to discover if the addition of stochastic depth provided any meaningful advantage in training speed over the models without stochastic depth.

4.1 Model Computational Performance During Training

One of the primary benefits of stochastic depth that we were looking to verify is the theoretical reduction in training time, without a sacrifice to model performance. To that end, while training models for weather modeling, we recorded metrics about resource consumption to compare across models, both with and without stochastic depth.

Model	GMI Band	Total Time (Hours)	Total Number of Epochs	Average Time Epoch(Seconds)	Average Time/Step (Ms)	Best Epoch	Best RMSE During Training
ResNet56	9	24.27	91	960.42	390.49	88	1.26
SD ResNet56	9	36.78	136	973.69	395.2	131	1.75
ResNet56	10	39.32	99	1430.17	367.49	91	1.17
SD ResNet56	10	43.83	102	1546.97	397.53	98	1.89
ResNet110	9	93.94	199	1699.5	693.81	163	1.09
SD ResNet110	9	49.81	97	1848.75	754.52	95	1.75
SD ResNet56 S25	9	45.42	166	985.19	400.63	159	2.16
SD ResNet56 S75	9	35.65	130	987.32	402.12	118	1.45
ResNet56 Flip.	9	87.12	170	1844.96	753.18	160	0.78
SD ResNet56 Flip.	9	64.05	123	1874.59	765.15	121	1.574

Table 4.1. Model Performance During Training. SD - Stochastic Depth, S25/S75 - Survivability, Flip - Flipout

Table 4.1 shows the metrics for model training time and relative performance, where models with SD denotes stochastic depth variants, and S25 S75 denote a modified survivability probability of 0.25 and 0.75 respectively, over the baseline 0.5. We mostly tested only on GMI band 9 to maintain consistency across all models, however we tested two on band 10 as well, as a means to compare across two bands as well. Band 9 was chosen for most of the models as it contains less data, therefore all training runs would be moderately quicker than if we had primarily used Band 10.

As can be seen by the Total Time column, Table 4.1, there were mixed results for total training time across the various models. For the ResNet56 testing, the stochastic depth implementation ran for longer with more epochs than the base ResNet56 models on both bands 9 and 10. ResNet110 however shows a sharp decrease in training time for the stochastic depth iteration, which initially would lead to the assumption that the implementation of stochastic depth helped decrease training time. However, the stochastic depth implementation ran for only 97 epochs, compared to the base ResNet110 running for 199. This is likely because of the adaptive training model, whereby the training was stopped early if the model was not improving. It would be a worthwhile course for future research to allow both models to run for the full 500 epochs and compare the results, to remove this variable from comparison analysis.

Furthermore, the stochastic depth implementation had a slower average time per epoch than the base ResNet110 model, with 1848.75 seconds compared to 1699.5 seconds for the base ResNet110. We included the ResNet110 in testing to see if the addition of stochastic depth to the model yielded a more visible training speed up when the depth of the model is drastically increased. These results would indicate that a meaningful training speed up did not occur. This trend is consistent throughout our testing, with all stochastic depth ResNets running slower per epoch when compared to their base ResNet counterparts, regardless of how many epochs the models ended up running.

The absence of a training speed increase is itself disappointing; however, it is only part of the puzzle. Not only did the stochastic depth models not train faster, but nearly all of them yielded a worse RMSE during training than their base ResNet counterparts, in some cases much worse, particularly the Flipout stochastic depth model, with an RMSE of 1.574 compared to the base Flipout ResNet56, with an RMSE of 0.78.

Model	GMI Band	Total Time (Hours)	Total Number of Epochs	Average Time Epoch(Seconds)	Average Time/Step (Ms)	Best Epoch	Best RMSE During Training
ResNet56	9	24.27	91	960.42	390.49	88	1.26
SD ResNet56	9	36.78	136	973.69	395.2	131	1.75
SD ResNet56 S25	9	45.42	166	985.19	400.63	159	2.16
SD ResNet56 S75	9	35.65	130	987.32	402.12	118	1.45

Table 4.2. Stochastic Depth Variations.

As discussed in Chapter 3, we tested a few different iterations of stochastic depth on ResNet56, as highlighted in Table 4.2. The standard stochastic depth ResNet56 had a default survival probability of 0.5, as outlined by Huang et al [9]. We also tested a survival probability of 0.25 and 0.75, represented by S25 and S75 in Table 4.2, respectively. While their performance was very similar across testing, it is worth noting that the S75 model, while slightly slower than the recommended 0.5 survivability model, had a lower RMSE of 1.45 during training, which is a 20% improvement over the base SD ResNet56.

It is important to note that these RMSE values are during training, not testing, and are not the most accurate metric to use when comparing models. We will discuss training results

with the model predictive performance in Section 4.2.

4.2 Model Predictive Performance During Testing

Model	RMSE	MAE	MAPE	MedAbsErr	MaxErr	AvgVar
ResNet56	3.073	1.938	0.008	1.430	138.576	0
SD ResNet56	3.078	1.953	0.008	1.465	142.205	0
ResNet110	2.931	1.763	0.007	1.282	136.345	0
SD ResNet110	3.036	1.879	0.007	1.352	141.211	0
SD ResNet56 S25	2.863	1.754	0.007	1.263	139.727	0
SD ResNet56 S75	3.298	2.188	0.009	1.706	135.749	0
ResNet56 Flip.	2.875	1.709	0.007	1.215	144.393	0.187
SD ResNet56 Flip.	2.999	1.887	0.007	1.408	140.136	0.048

Table 4.3. 183 GHz Prediction Results for January - 720500 Samples.

As can be seen in Table 4.3, with one exception, the stochastic depth variants of the ResNet models were unable to outperform their standard counterparts when comparing RMSE. The stochastic depth variants were very close in performance, however with no added speed-up benefit during training, one would hope to at least see improvements in the models' predictive performance. The one exception was SD ResNet56 S25, which had the lowest RMSE of all the models (including the probabilistic ones), with an RMSE of 2.863. This narrowly beats out the standard ResNet56, with an RMSE of 3.073, and strongly outperformed the S75 model, with an RMSE of 3.293.

This spread between the three survivabilities for the stochastic depth ResNet56s, 0.25, 0.5 (standard), and 0.75 carries on to MAE as well, with each model having an MAE of 1.754, 1.953 (at which point the standard ResNet56 performed better) and 2.188, respectively. When comparing all models by MAE however, the base probabilisitic model, ResNet56 Flipout, performed the best out of all of the models tested, with an MAE of 1.709. Looking at the MAPE results for all the models, we can see just how close all of these iterations were in practicality. With the exception of SD ResNet56 S25 and S75, each stochastic

depth model was within at least 1/1000 of each other's MAPE. SD ResNet56 S25 shows improvement here over both the base ResNet56 and SD ResNet56, as it did with RMSE, with a MAPE of 0.007 compared to 0.008. This trend continues when looking at the median absolute error (MedAbsErr), where all non-SD models had a lower MedAbsErr compared to their SD counterparts, with the exception of SD ResNet56 S25, which had a MedAbsErr of 1.263 compared to ResNet56's 1.430, an 11.67% decrease. Max error (MaxErr) is not as consistent as the previous results. SD ResNet56 failed to outperform ResNet56, however SD ResNet56 S25, which until now had been marginally beating out the base ResNet56, now fales, with a higher MaxErr of 139.72. Surprisingly, SD ResNet56 S75 did better than both, with a MaxErr of just 135.749, the lowest of all models tested. The MaxErr for the probabilistic models is also inconsistent with previous results, with SD ResNet56 Flip. having a lower MaxErr of 140.136 compared to ResNet56 Flip.'s 144.393. The average variance for most of the models is 0, because they are deterministic models, as discussed in Chapter 2. The results for the probabilistic models, however, is quite interesting, with SD ResNet56 Flip. returning a considerably lower variance of just 0.048 compared to ResNet56 Flip.'s 0.187.

4.2.1 Spatial Mapping of Results

Figures 4.1 through 4.8 show spatial representations of all the models tested, mapping a GMI swath of the labels and predictions from data collected on 08 January. For the labels and predictions, Figures (a) and (b), respectively, the color values equate to Tb (Temperature Brightness) in Kelvin (K), and for (c) and (d) where applicable, the colors are in Kelvin (K), and K^2 respectively. Figures 4.7 and 4.8 have (d) variance graphs, while the other ones do not, because only those two models were probabilistic, and thus all other models have a variance of 0, as seen in Table 4.3.

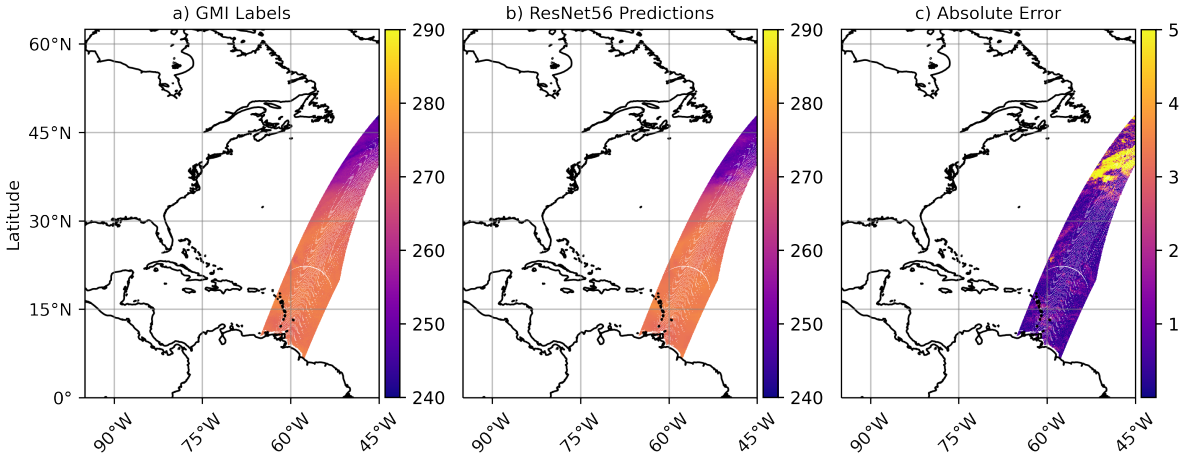

Figure 4.1. GMI Swath at 183 GHz on 08 January for ResNet56.

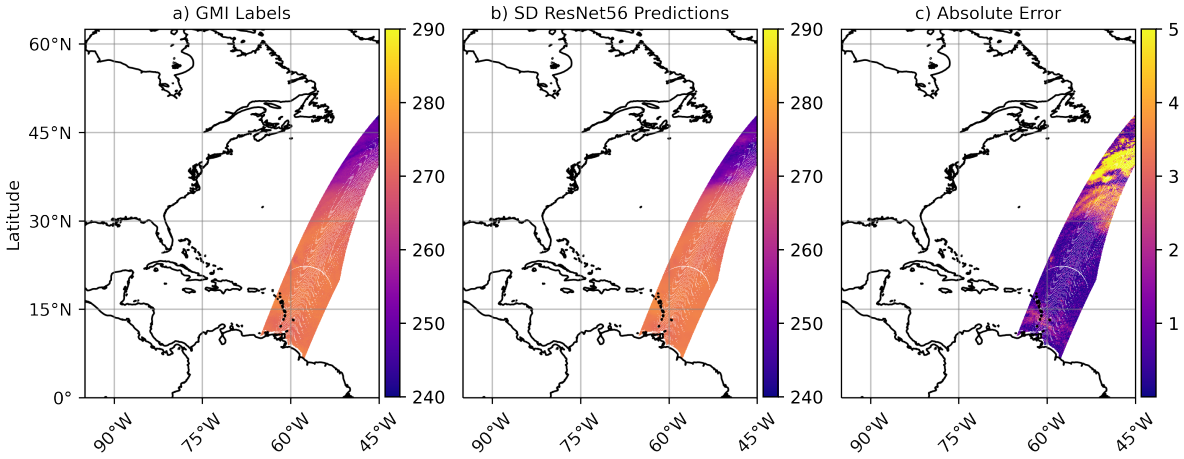

Figure 4.2. GMI Swath at 183 GHz on 08 January for SD ResNet56.

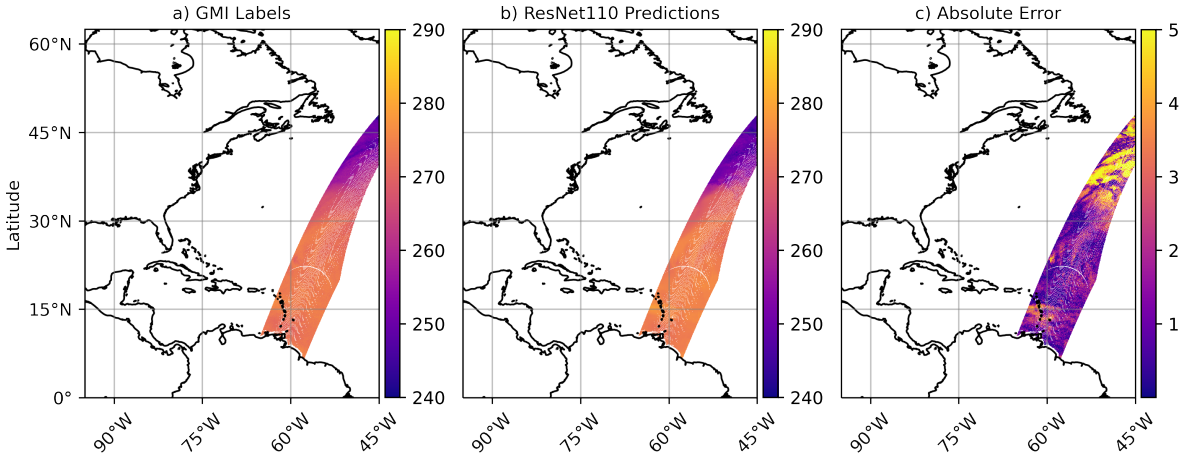

Figure 4.3. GMI Swath at 183 GHz on 08 January for ResNet110.

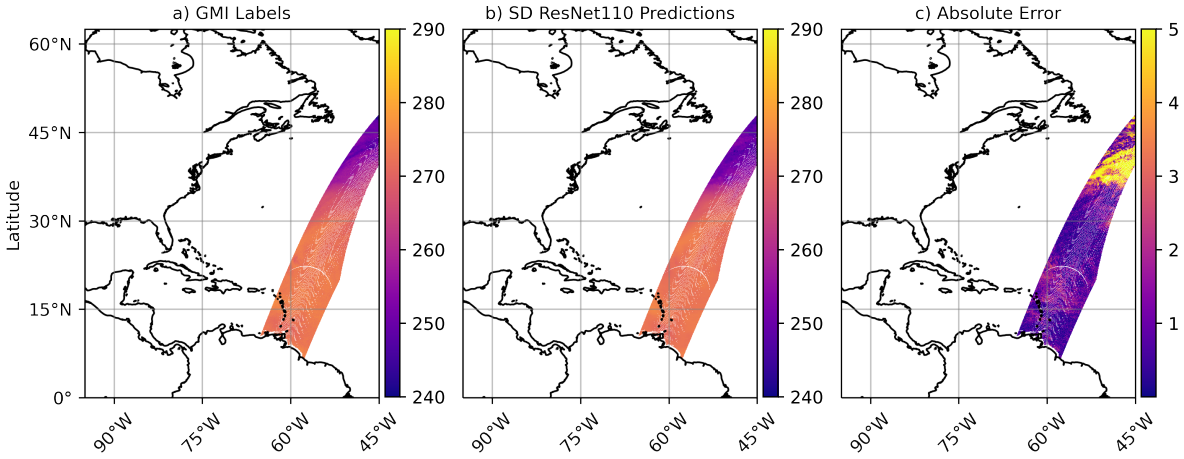

Figure 4.4. GMI Swath at 183 GHz on 08 January for SD ResNet110.

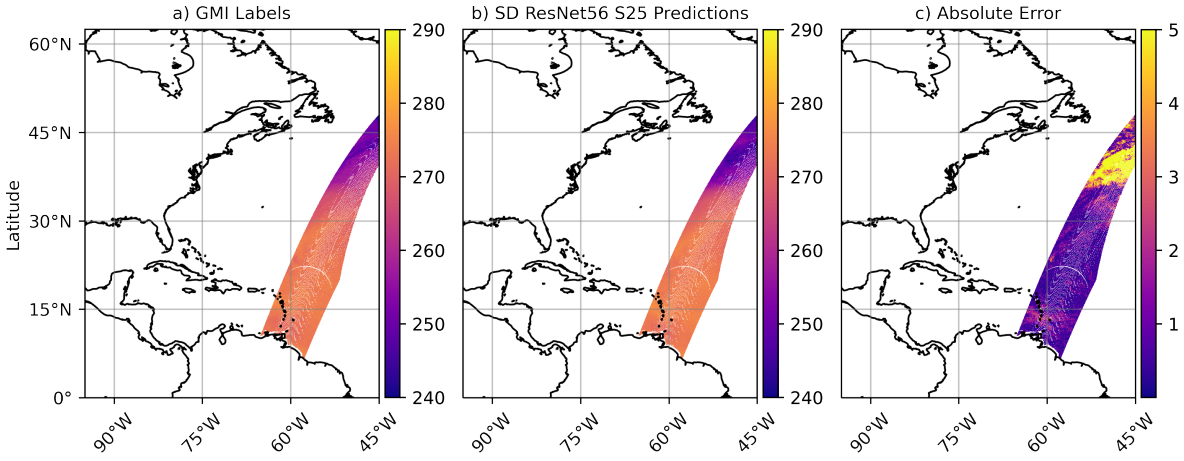

Figure 4.5. GMI Swath at 183 GHz on 08 January for SD ResNet56 S25.

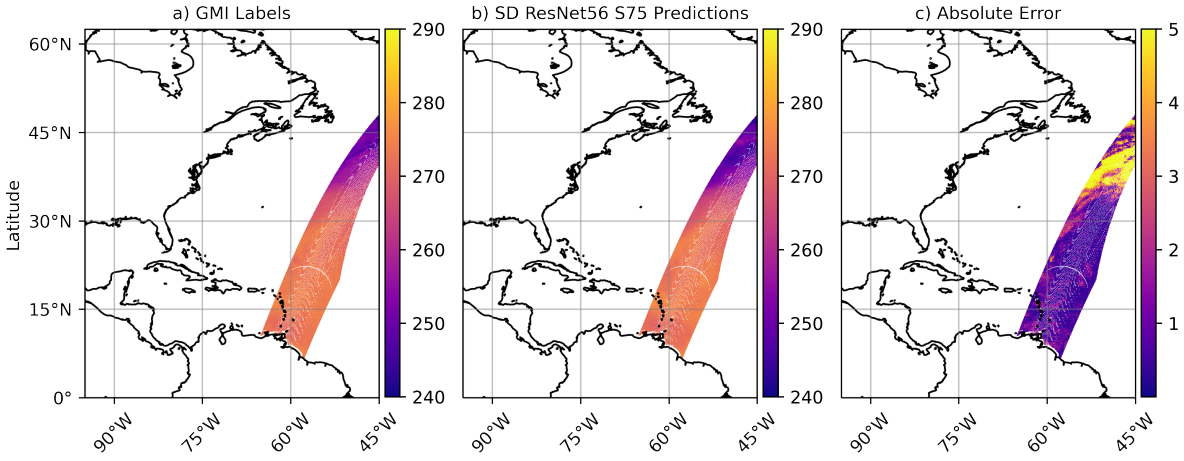

Figure 4.6. GMI Swath at 183 GHz on 08 January for SD ResNet56 S75.

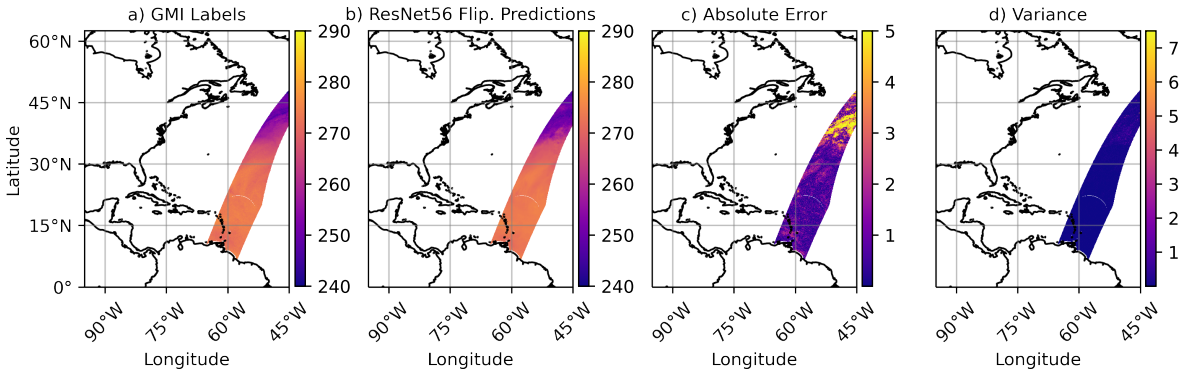

Figure 4.7. GMI Swath at 183 GHz on 08 January for ResNet56 Flip.

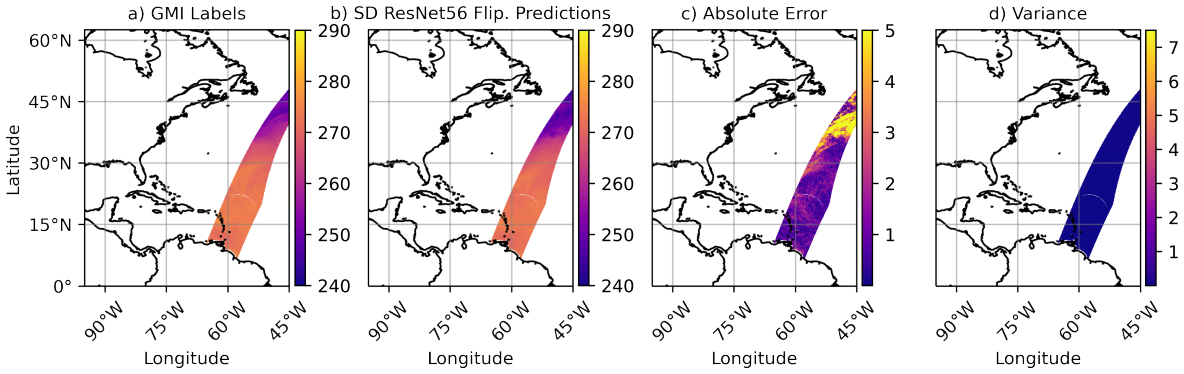

Figure 4.8. GMI Swath at 183 GHz on 08 January for SD ResNet56 Flip.

When looking at the spatial mapping figures, it can be difficult to visually gauge the results of the models vs. the labels, as well as comparing them against each other; however, when looking at the absolute errors, the difference becomes more stark. For example, in Figure 4.2, there is visibly higher absolute error in the vicinity of 35°N 50°W in the swath.

The results between ResNet110 and SD ResNet110 are harder to visibly gauge, as it looks like around 35°N 50°W there are more sections of lower error in ResNet110, while around 20°N 50°W in the swath SD ResNet110 has slightly more success.

The results do not visually appear to be any different with the probabilistic models, Figures 4.7 and 4.8. They both have very similar results, with what appears to be a slightly higher scattering of higher absolute error with SD ResNet56 Flip., which is consistent with the slightly worse performance seen in Table 4.3

CHAPTER 5:
Conclusion

Weather modeling is both a complex task, and the practicality of the results of weather modeling depends not only on the quality of the prediction, but also the timeliness, particularly when the models are intended for operational planning by the U.S. Navy. Weather modeling, and the weather reports generated from them are a necessity for good operational planning; no operation the Navy conducts is free from the influence of weather conditions, be that sea state levels for ships and submarines underway, to wind and air temperature predictions for flight operations. Better weather forecasts means removing as much of the unknown as possible when planning, and by extension, keeping U.S. Navy sailors safer. Ortiz et al. [2] speculate that providing synthetic GMI data to Navy data assimilation systems can increase weather forecasting performance. To produce synthetic GMI data Ortiz et al. [2] use Bayesian ResNet architectures. To that end, an exploration of stochastic depth seemed fruitful as an expansion of that work. Bayesian CNNs have tremendous value as predictive tools. They provide uncertainty which in turn can help generate more accurate results compared to deterministic counterparts. However, this makes them much more computationally expensive, which is where stochastic depth's benefits come in. We sought to utilize ResNets, a proven successful architecture for computer vision applications, with the inclusion of stochastic depth, as a means to potentially drastically reduce training time for these weather modeling scenarios, without any impact to performance. Unfortunately, we were not able to re-create the results we expected, and as those described by Huang et al [9].

We saw no improvement to computing time or model performance, while Huang et al. [9] reported significant improvements in both, with a speed-up of 26.97% in computing time for CIFAR-10, 26.45% for CIFAR-100, and 24.22% for SVHN. Their stochastic depth ResNet model also showed the lowest test error percentage out of twelve other competing models, to include a matching ResNet architecture without stochastic depth.

However, Huang et al. [9] achieved similar results with a large scale dataset such as ImageNet that is comparable in size to the satellite dataset used in this thesis. Specifically, the authors report that they achieved lower performance compared to the non-SD architecture and that

they do not provide any measure of speed-up in training. They do comment that they faced issues verifying a speed-up with an SD architecture and ImageNet dataset.

Similarly, the finding of this thesis confirms that SD architectures with large scale datasets do not result in a speed-up in training. We speculate that the reason for that is the exaggerated I/O due to our data being read from storage rather than being in-memory as it is with toy-datasets.

Another possibility is the training conditions. It is possible that the computation nodes we used for this research are effectively too powerful for us to notice the training benefits afforded by stochastic depth. Huang et al. [9] does not specify hardware configuration for their runs.

Additionally, this thesis demonstrated for the first time in literature the effect of utilizing a stochastic depth layer with a Bayesian ResNet architecture. The overall performance is comparable to the Bayesian ResNet without the SD. We confirmed that the SD Bayesian ResNet model can produce meaningful uncertainties but have not analyzed that aspect further.

5.1 Future Work

There are a number of avenues for future work in this exploration of stochastic depth. We do not believe that this research definitively concludes that stochastic depth does not work. The next step to test stochastic depth implementation would be to repeat the experiments conducted by Huang et al. [9] on CIFAR-10, CIFAR-100, SVHN, and ImageNet, but just utilizing the TensorFlow Addons API for stochastic depth implementation. This should illuminate whether or not the API is correctly implementing stochastic depth as Huang et al. [9] did. If we are unable to repeat their findings with our ResNet models as we currently have them configured, the following course of action would be to build a ResNet model from scratch, designing custom stochastic depth implementation, rather than using the TensorFlow Addons API, and then attempting to repeat Huang et al.'s [9] findings. This would guarantee that stochastic depth is working correctly, dropping the ResBlocks in training and then testing against the full depth of the model. This would in turn provide concrete results when directly comparing models with and without stochastic depth for weather modeling.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] The forecast is: Better weather predictions for the Navy. *News: The Forecast Is: Better Weather Predictions for the Navy - Office of Naval Research*. [Online]. Available: https://www.onr.navy.mil/en/Media-Center/Press-Releases/2013/NAVGEM-Weather-Prediction-Navy-Ferek

[2] P. Ortiz, M. Orescanin, S. W. Powell, M. Hall, and V. Petković, "Uncertainty quantification through Bayesian deep learning for passive microwave brightness temperature prediction," unpublished.

[3] M. Hall, "Emulating passive microwave observations with patch-to-pixel convolutional neural networks," unpublished.

[4] T. Hall, H. E. Brooks, and C. A. Doswell. Precipitation forecasting using a neural network. *Neural Net QPF paper*. [Online]. Available: https://www.nssl.noaa.gov/users/brooks/public_html/hall/neural.html

[5] V. Chawla. (2021, Jun). Is more data always better for building analytics models? *Analytics India Magazine*. [Online]. Available: https://analyticsindiamag.com/is-more-data-always-better-for-building-analytics-models/

[6] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, vol. 30.

[7] M. Orescanin, V. Petković, S. W. Powell, B. R. Marsh, and S. C. Heslin, "Bayesian deep learning for passive microwave precipitation type detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.

[8] P. Ortiz, M. Orescanin, V. Petković, S. W. Powell, and B. Marsh, "Decomposing satellite-based classification uncertainties in large earth science datasets," *IEEE Transactions on Geoscience and Remote Sensing*, 2022.

[9] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," *CoRR*, vol. abs/1603.09382, 2016. Available: http://arxiv.org/abs/1603.09382

[10] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

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

[13] V. Petković, M. Orescanin, P. Kirstetter, C. Kummerow, and R. Ferraro, "Enhancing PMW satellite precipitation estimation: Detecting convective class," *Journal of Atmospheric and Oceanic Technology*, vol. 36, no. 12, pp. 2349–2363, 2019.

[14] T. G. Roberts. (2020, Oct). Popular orbits 101. *Aerospace Security*. [Online]. Available: https://aerospace.csis.org/aerospace101/popular-orbits-101/#:~:text=Satellites%20in%20LEO%20typically%20take,including%20Earth%20observation%20and%20reconnaissance.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[16] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications Co, 2018, oCLC: ocn982650571.

[17] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational Vision Science Technology*, vol. 9, no. 2, pp. 14–14, 02 2020. Available: https://doi.org/10.1167/tvst.9.2.14

[18] M. Deshpande. (2022, Jan). Perceptrons: The First Neural Networks. *Python Machine Learning*. [Online]. Available: https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/

[19] R. Quiza and J. Davim, *Computational Methods and Optimization*. NY, USA: Springer-Verlag, 01 2011, ch. 2, pp. 177–208.

[20] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug 2018. Available: https://doi.org/10.1007/s13244-018-0639-9

[21] X. Kang, B. Song, and F. Sun, "A deep similarity metric method based on incomplete data for traffic anomaly detection in iot," *Applied Sciences*, vol. 9, p. 135, 01 2019.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[24] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep learning in image classification using residual network (resnet) variants for detection of colorectal cancer," *Procedia Computer Science*, vol. 179, pp. 423–431, 2021, 5th International Conference on Computer Science and Computational Intelligence 2020. Available: https://www.sciencedirect.com/science/article/pii/S1877050921000284

[25] D. Yang, C. Martinez, L. Visuña, H. Khandhar, C. Bhatt, and J. Carretero, "Detection and analysis of covid-19 in medical images using deep learning techniques," *Scientific Reports*, vol. 11, no. 1, p. 19638, Oct 2021. Available: https://doi.org/10.1038/s41598-021-99015-3

[26] D. Chen, W. Zhang, X. Xu, and X. Xing, "Deep networks with stochastic depth for acoustic modelling," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016, pp. 1–4.

[27] L. V. Jospin, W. L. Buntine, F. Boussaïd, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks - a tutorial for deep learning users," *CoRR*, vol. abs/2007.06823, 2020. Available: https://arxiv.org/abs/2007.06823

[28] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *International Conference on Learning Representations*, 2018.

[29] R. Feng, N. Balling, D. Grana, J. S. Dramsch, and T. M. Hansen, "Bayesian convolutional neural networks for seismic facies classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 10, pp. 8933–8940, 2021.

[30] A. Filos, S. Farquhar, A. N. Gomez, T. G. J. Rudner, Z. Kenton, L. Smith, M. Alizadeh, A. de Kroon, and Y. Gal. (2019). A Systematic Comparison of Bayesian Deep Learning Robustness in Diabetic Retinopathy Tasks. [Online].

[31] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, p. 357–362, Sep 2020.

[32] P. McClure, N. Rho, J. A. Lee, J. R. Kaczmarzyk, C. Y. Zheng, S. S. Ghosh, D. Nielson, A. G. Thomas, P. A. Bandettini, and F. Pereira, "Knowing what you know in

brain segmentation using deep neural networks," *CoRR*, vol. abs/1812.01719, 2018. Available: http://arxiv.org/abs/1812.01719

[33] Gpm Microwave Imager (GMI). *NASA*. [Online]. Available: https://gpm.nasa.gov/missions/GPM/GMI

[34] Input Channel Visualization. *NASA*. [Online]. Available: https://www.nasa.gov/sites/default/files/thumbnails/image/3252.png

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. [Online]. Available: https://www.tensorflow.org/. Software available from tensorflow.org.

[36] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, p. 859–877, Apr 2017. Available: http://dx.doi.org/10.1080/01621459.2017.1285773

[37] T. Ganegedara. (2019, Jun). Intuitive guide to understanding KL divergence. *Medium*. [Online]. Available: https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-kl-divergence-2b382ca2b2a8

[38] D. Oliver, B. Sick, and E. Murina, *Probabilistic Deep Learning: With Python, Keras, and Tensorflow Probability*. Manning, 2020.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and [U+FFFD] Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. Available: http://arxiv.org/abs/1412.6980

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California