12-2021

GPS-Denied Navigation Using Synthetic Aperture Radar Images and Neural Networks

Teresa White
Utah State University

GPS-DENIED NAVIGATION USING SYNTHETIC APERTURE RADAR IMAGES AND NEURAL NETWORKS

by

Teresa White

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Statistics

Approved:

Kevin R. Moon, Ph.D. Major Professor	John R. Stevens, Ph.D. Committee Member
Todd K. Moon, Ph.D. Committee Member	Randy Christensen, Ph.D. Committee Member

D. Richard Cutler, Ph.D.
Interim Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2021

ABSTRACT

GPS-DENIED NAVIGATION USING SYNTHETIC APERTURE RADAR IMAGES AND NEURAL NETWORKS

by

Teresa White, Master of Science

Utah State University, 2021

Major Professor: Kevin R. Moon, Ph.D.
Department: Mathematics and Statistics

Unmanned aerial vehicles (UAV) often rely on GPS for navigation. GPS signals, however, are very low in power and easily jammed or otherwise disrupted. This paper presents a method for determining the navigation errors present at the beginning of a GPS-denied period utilizing data from a synthetic aperture radar (SAR) system. This is accomplished by comparing an online-generated SAR image with a reference image obtained a priori. The distortions relative to the reference image are learned and exploited with a convolutional neural network to recover the initial navigational errors, which can be used to recover the true flight trajectory throughout the synthetic aperture. The proposed neural network approach is able to learn to predict the initial errors on both simulated and real SAR image data.

(79 pages)

ACKNOWLEDGMENTS

First and foremost, I am grateful to God for all the blessings that I received during these years to complete my graduate study in the Department of Mathematics and Statistics at Utah State University. I would like to express my deepest gratitude to Dr. Kevin Moon, my advisor, for his invaluable advice, continuous support, and patience during my MS study. His knowledge and experience have helped me enormously with my research and in general, during my whole academic path here at Utah State University. I would like to thank to my committee members, Dr. John Stevens, Dr. Todd Moon and Dr. Randy Christensen for their support and advice during this process. Also, I want to thank Sandia National Labs for partially supporting this research on grants 201782, 202136, and 202854. The support and resources from the Center for High Performance Computing at the University of Utah are gratefully acknowledged as well.

Additionally, I would like to express my sincere thanks to Professor David Bregenzer for his tremendous support during these two years. It was a great pleasure of working with him. Thanks to his support, I was able to balance my teaching and research work very well. I would like to thank to Dr. Adele Cutler and Dr. Brennan Bean for their advice during my first semester in the program which encouraged me to stay and finish my graduate program. In the same way, I am grateful for Gary Tanner's support. His efficiency and great work in the department has helped me succeed in the different tasks that I needed to complete. I would also like to show gratitude for the support and help from many of the graduate students, my friends, in the department.

Most importantly, none of this could have happened without my family. Special thanks to my parents, Leonardo Arcos Gutierrez and Fortunata Sapa Choccata, for their support and faith in me. I am grateful for the invaluable support of my husband, Jacob White. He supported me throughout this process and encouraged me at times when I thought that it was impossible to continue. I would like to thank my children for their patience and

teamwork, and my sister, Nancy Arcos Sapa, for her advice and support over these years. Their love, support and encouragement have been a great part of this achievement.

Teresa White

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AT	Along Track
BPA	Back-Projection Algorithm
CNN	Convolutional Neural Network
CT	Cross Track
D	Down
EKF	Extended Kalman Filter
GPS	Global Positioning System
MSE	Mean Squared Error
SAR	Synthetic Aperture Radar
UAV	Unmanned Aerial Vehicle

CHAPTER 1

INTRODUCTION

Unmanned aerial vehicles (UAV) have many applications such as search and rescue operations, military applications, agricultural applications, surveillance, and more. In many of these applications, knowledge of the current flight trajectory of the UAV is necessary to carry out the UAV's mission. In normal circumstances, GPS is typically used to obtain this knowledge. However, in abnormal and important circumstances, such as navigating in a hostile airspace, the GPS signal may be denied or otherwise degraded. Here, we propose a deep neural network model that accurately estimates the flight trajectory by comparing a synthetic aperture radar (SAR) image obtained online to a previously-obtained reference image.

1.1 Motivation

GPS signal vulnerability to many forms of interference has motivated research in alternative approaches to localization [1, 2]. Radar is one of many active approaches, which, in contrast to image-based methods, is independent of lighting conditions, operates in any weather condition, and provides an accurate measurement of range. Many approaches have been developed which exploit artifacts in SAR imagery to determine navigation errors, either relative to the beginning of the synthetic aperture [3–7] or in a global/absolute reference frame [8–14].

1.2 Related Work

Back-projection-based SAR imaging is a technique for producing radar images during arbitrary flight trajectories [15–17]. SAR images are created by sending a series of radar pulses during a portion of the vehicle's trajectory as shown in Figure 1.1. The back-projection algorithm (BPA) is then used to synthesize the reflected signals into a two or

three-dimensional SAR image. To obtain an accurate image, back-projection requires an accurate estimate of the position of the radar antenna throughout the synthetic aperture. If the assumed flight trajectory is inaccurate, the resulting SAR image will be distorted by shifts and blurs in the along track (AT) and cross track directions (CT) [18].

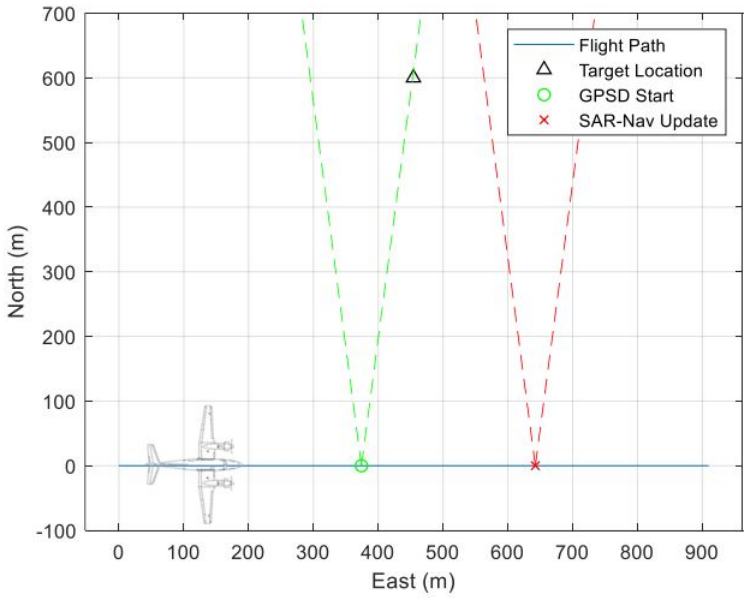

Fig. 1.1: Illustration of the simulated flight path in an eastern direction, with the SAR antenna and its associated beam-width pointing out the left side of the aircraft. The green and red lines denote the beginning and end of the GPS-denied region [18].

In a typical SAR application, an inertial navigation system uses GPS measurements to generate the best estimate of the aircraft trajectory and minimize SAR image distortions. In contrast, we focus on the reverse problem, where the distortions in the SAR image are exploited to estimate the vehicle trajectory throughout the synthetic aperture. To reduce the dimensionality of the estimation problem, we use the well-known error dynamics for inertial navigation systems, effectively reducing the number of parameters to solve for to nine initial errors.

For the simple case of constant horizontal position errors, identification of SAR image shifts with respect to an *a priori* SAR map provides easily exploitable information regarding position errors. When velocity and attitude errors are included, however, the mapping from

initial errors to image distortions is more complex [18, 19].

There is a strong precedent for applying machine learning techniques to SAR imagery. The most common application is automatic target recognition [20–25]. Other applications include the use of deep learning to detect changes in radar images of the same landscape over time [26] or auto-focusing blurry radar images [27], both of which focus primarily on image reconstruction. Despite a wide range of target recognition problems, there is no precedent in the literature to use machine learning to predict initial navigational errors in a regression-based framework, further motivating the need for our analysis.

Our problem shares some similarities with the task of image registration [28–31], in that a reference image and an input image are both considered. However, in image registration, the goal is to align the input image with the reference image. In our setting, we extract information (the true trajectory of the aircraft) from the degree to which the input image and reference image do not match, as well as the manner in which they differ.

1.3 Thesis Contributions

This research shows that neural networks can be used to develop a black-box model of the distortion dynamics: The distorted image and a reference image are input and the true flight trajectory is the output as shown in Figure 1.2.

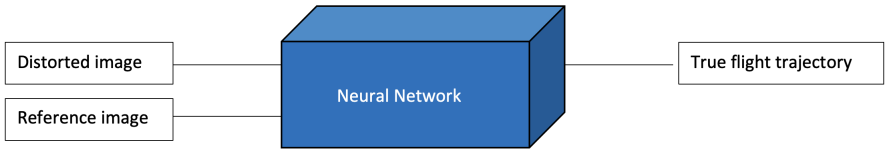

Fig. 1.2: Neural network approach. It takes as input the distorted image and a reference image and outputs the true flight trajectory.

This model uses a specific approach for characterizing the distortions in the input image: stacking the reference image, the distorted image, and the difference between the distorted and reference image as shown in Figure 1.3 so that each is treated as a different

channel in the first convolutional layer. Six different scenarios were studied with different combinations of active navigation errors for each dataset explored.

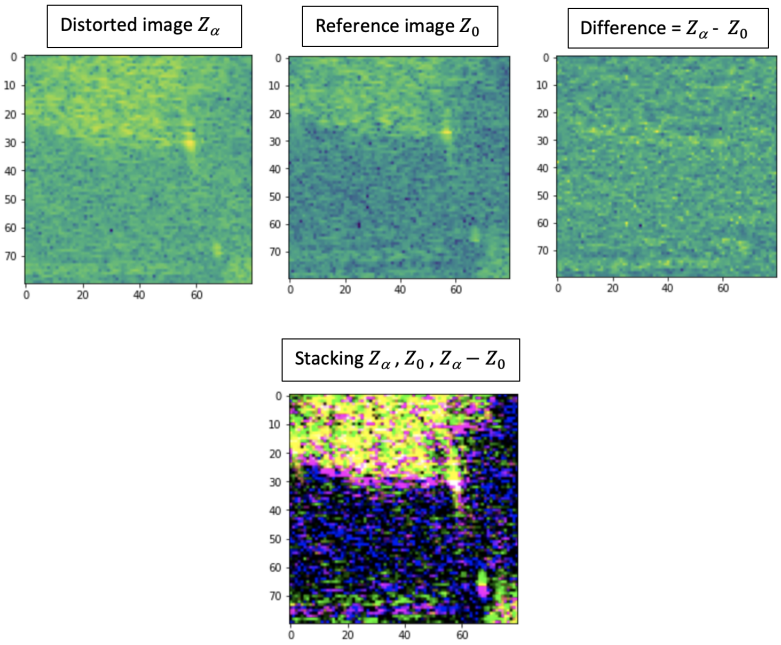

Fig. 1.3: 3-channel Image Input. The input of our model consists of a 3-channel image: the distorted image (Z_α), the reference image (Z_0), and the difference image ($Z_\alpha - Z_0$).

We applied simple convolutional neural network architectures as well as more advanced ResNet and Wide ResNet models to the problem. Our experiments showed that a modified Wide ResNet 50_2 model produced the best results in estimating the flight trajectory using both simulated and real data as summarized in Table 1.1.

Scenario #	Average MSE (Sim Data)	Average MSE (Real Data)
1	0.0442	0.0494
2	0.1809	0.6206
3	0.2792	0.6287
4	0.4315	0.4285
5	0.5425	0.7651
6	0.6396	0.7766

Table 1.1: This table shows the average MSE across all the scenarios for the simulated and real-2-sec data using the Wide ResNet 50_2 model. Due to normalization, an MSE less than 1 indicates that the model is learning useful information. As shown, the Wide ResNet 50_2 model performed well in both simulated and real-2-sec data.

Part of this work has been accepted for publication to the *2021 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) [32]. It was also posted on *arXiv* [33].

1.4 Roadmap

Chapter 2 begins by providing the necessary background regarding navigation errors on BPA-SAR images and convolutional neural networks. **Chapter 3** describes the different sets of SAR image data generated for this study and the data preparation for our analysis. **Chapter 4** explores different methods for estimating the initial navigation errors on both simulated and real data. **Chapter 5** shows the performance of the ResNet models and provides an analysis of the results. **Chapter 6** discusses the future work and provides a conclusion of our study.

CHAPTER 2

BACKGROUND

2.1 Navigation Errors on BPA-SAR Images

In strapdown inertial navigation systems, measurements of specific force $\boldsymbol{\nu^b}$ and angular rate $\boldsymbol{\omega^b}$ are used to propagate the position $\boldsymbol{p}^n$, velocity $\boldsymbol{v}^n$, and attitude quaternion q_b^n of the aircraft, via the following differential equations

$$\begin{bmatrix} \dot{\boldsymbol{p}}^n \\ \dot{\boldsymbol{v}}^n \\ \dot{q}_b^n \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}^n \\ T_b^n \boldsymbol{\nu}^b + \boldsymbol{g}^n \\ \frac{1}{2} q_b^n \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b \end{bmatrix} \end{bmatrix}, \tag{2.1}$$

where b represents a coordinate system attached to the body and n represents a locally-level frame with the x-axis aligned with the nominal vehicle velocity, the z-axis in the direction of gravity, and the y-axis determined by the right-hand-rule. Thus the n frame represents an along-track, cross-track, down coordinate system.

At regular intervals, an external aiding device, such as GPS, provides information to correct the navigation states, typically in the framework of an extended Kalman filter. Between aiding device measurements, or in the absence thereof, the navigation errors accumulate over time. For straight and level flight, typical of SAR data collection, the error growth is modeled to first order:

$$\begin{bmatrix} \delta \boldsymbol{p}^n(t) \\ \delta \boldsymbol{v}^n(t) \\ \delta \boldsymbol{\theta}^n(t) \end{bmatrix} = \Phi(t, t_0) \begin{bmatrix} \delta \boldsymbol{p}^n(t_0) \\ \delta \boldsymbol{v}^n(t_0)) \\ \delta \boldsymbol{\theta}^n(t_0) \end{bmatrix}, \tag{2.2}$$

where $\Phi(t, t_0)$ is the state transition matrix defined as

$$\Phi(t, t_0) = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t & (\boldsymbol{\nu}^n\times)\,\frac{\Delta t^2}{2!} \\ 0_{3\times3} & I_{3\times3} & (\boldsymbol{\nu}^n)\times\Delta t \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix}. \tag{2.3}$$

The relationship between the estimated, true, and error parameters is thus defined by the additive or multiplicative relationships

$$\boldsymbol{p}^n(t) = \hat{\boldsymbol{p}}^n(t) + \delta\boldsymbol{p}^n(t) \tag{2.4}$$

$$\boldsymbol{v}^n(t) = \hat{\boldsymbol{v}}^n(t) + \delta\boldsymbol{v}^n(t) \tag{2.5}$$

$$q_b^n(t) = \begin{bmatrix} 1 \\ -\frac{1}{2}\delta\boldsymbol{\theta}^n(t) \end{bmatrix} \otimes \hat{q}_b^n(t). \tag{2.6}$$

Thus given an initial condition for the navigation errors, the true navigation states can be recovered by propagating the errors using Equation 2.2 and substituting the result into Equations 2.4-2.6. The task of correcting errors in the estimated trajectory is, therefore, reduced to determining the nine errors at the beginning of the time interval of interest. A more detailed discussion of the inertial navigation framework and error modeling is provided in [18, 19].

Error	Shift Direction	Blur Direction
AT Position	AT	None
CT Position	CT	None
D Position	CT	None
AT Velocity	None	AT
CT Velocity	AT	None
D Velocity	AT	None
AT Attitude	None	Small AT
CT Attitude	None	Small AT
D Attitude	None	Small AT

Table 2.1: Effect of navigation errors on BPA-SAR images.

Previous sensitivity studies discovered the complex and seemingly ambiguous relationship between the nine initial navigation errors and the induced SAR image distortions, summarized in Table 2.1 [18]. In practical navigation systems, however, the down (D) component of position and velocity errors are easily controlled with a barometric altimeter. The absence of vertical errors, combined with relatively small sensitivity to typical attitude errors, results in SAR image distortions dominated by position and velocity errors in the AT and CT directions.

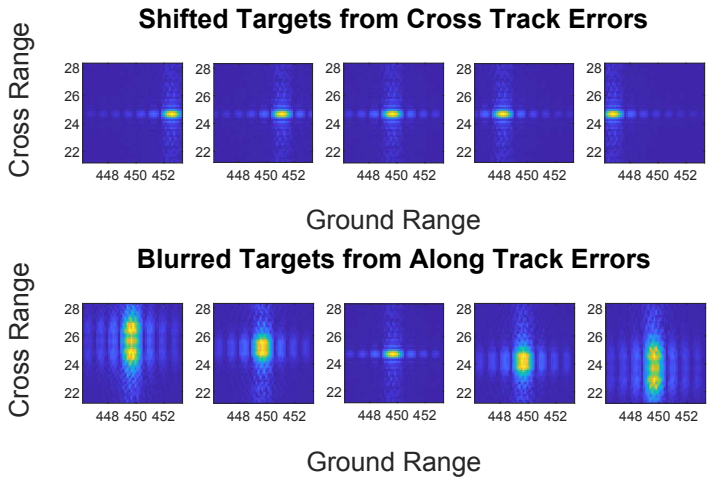

Fig. 2.1: Demonstration of shifting and blurring distortions due to navigation errors. (Top) Shift distortions caused by errors in cross track position. From left to right, cross track errors are -3, -1.5, 0, 1.5, and 3 meters. (Bottom) Blur distortions caused by errors in along track velocity. From left to right, along track errors are -0.4, -0.2, 0, 0.2, and 0.4 meters per second.

While image distortions are characterized by shifts and blurs in different directions (see Figure 2.1), certain combinations of navigation errors can create image distortions that appear similar to the human eye, making it difficult to determine the true source of distortion and thus the true trajectory error. We hypothesize that different error combinations create subtly different image distortions that can be used by neural networks to learn to predict the initial navigation errors.

2.2 Convolutional Neural Networks

In supervised machine learning, the computer learns from observational data to accomplish the task at hand. Beginning in 2006, the discovery of improved techniques for training neural networks combined with increased computational power led to deep learning, wherein deep neural networks have been able to outperform many traditional machine learning approaches in tasks such as image recognition [34].

Inspired by the sophisticated functionality of human brains, a neural network contains a collection of interconnected artificial neurons (nodes). The architecture of a feedforward neural network consists of an input layer of neurons, one or more hidden layers of neurons, and a final layer of output neurons (see Figure 2.2).

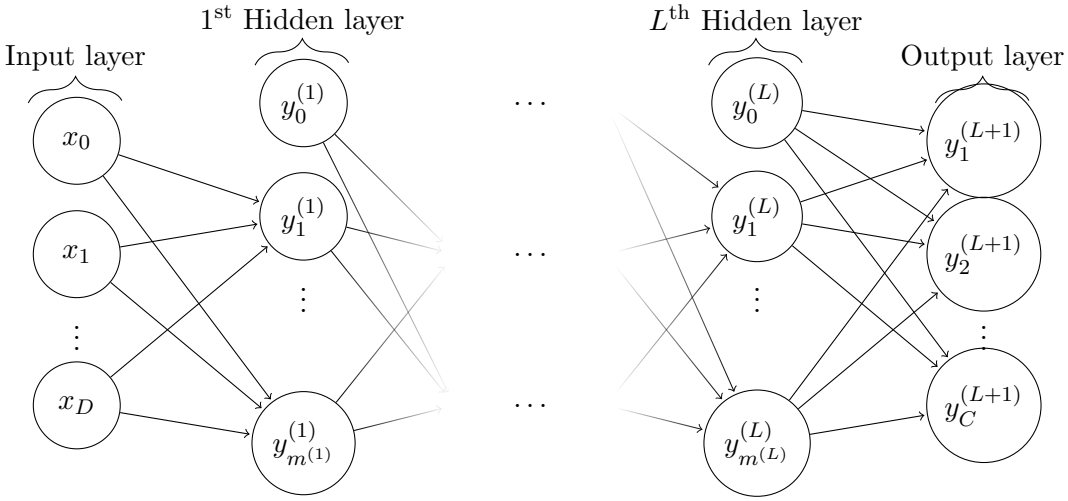

Fig. 2.2: Example of an Artificial Neural Network Architecture [35]. It consists of the input layer with D input units, L hidden layers with $m^{(l)}$ hidden units each, and the output layer with C output units.

Convolutional neural networks (CNN) are a special case of a neural network where some notion of spatial structure is assumed, such as in images. In the case of images, the layers of a CNN have neurons arranged in 3 dimensions (width, height, depth). The most common layers of a CNN are the input layer, convolutional layers, subsampling (or pooling) layers, and fully connected layers:

- Input layer: holds the raw pixel values of the image.
- Convolutional layer: computes the output of neurons that are connected to local regions in the input.
- Subsampling layer: downsamples the output of a convolution layer along the spatial dimensions (width, height).
- Fully connected layer: typically the final layers of a CNN. Maps from the output of the convolutional and subsampling layers to the final output, thus computing the class scores or regression prediction. Here each neuron is connected to all the outputs in the previous layer.

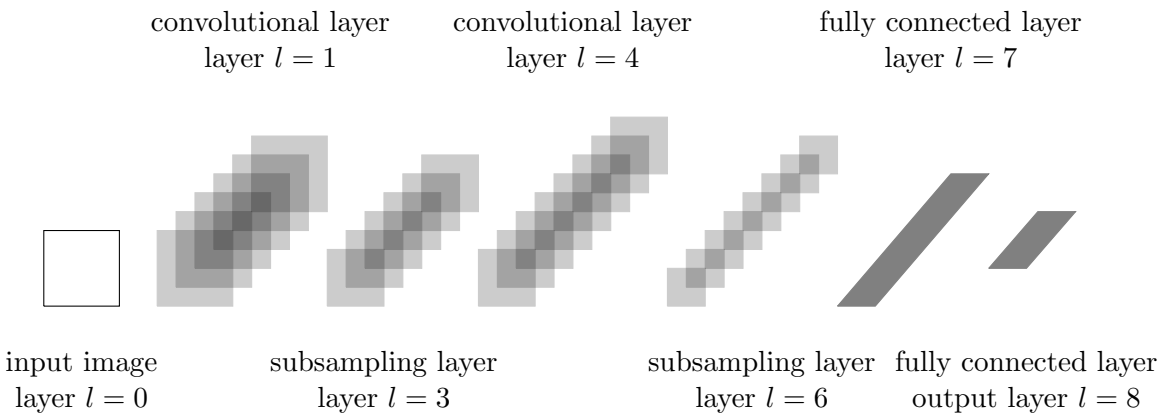

Fig. 2.3: Illustration of a simple convolutional neural network indicating convolutional layers, subsampling layers and fully-connected layers. Some details are omitted such as the number of channels or the input image size, among others. [35].

CNNs achieve very good performance on image recognition tasks [36–38]. We therefore use CNN architectures to learn the initial navigational errors on BPA-SAR Images. In addition, we considered advanced network architectures to obtain the desired performance such as multiple ResNet [36] and Wide ResNet [39] architectures. Each of these architectures is built with a residual block.

In Wide ResNet, a residual block with identity mapping can be represented by the following formula:

$$x_{l+1} = x_l + F(x_l, W_l)$$

where x_{l+1} and x_l are the respective input and output of the $l-$th unit in the network, F is a residual function and W_l are the parameters of the block. A residual network consists of sequentially stacked residual blocks [39].

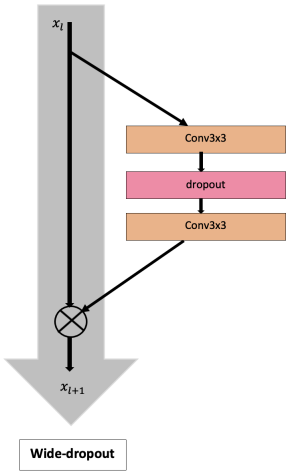

Fig. 2.4: Example of a wide-dropout residual block [39].

This block allows networks to be much deeper (more layers and hence better at dealing with images) while simultaneously mitigating the issue of the vanishing gradient [36]. Each of the ResNet and Wide ResNet architectures have pre-trained models trained on large image datasets.

CHAPTER 3

THE SAR DATA

3.1 Description of the Data

Three different sets of SAR image data were generated for training and testing: simulated data with a 5 second aperture length (sim-5-sec), and two real datasets with 2 (real-2-sec) and 10 second (real-10-sec) aperture lengths, respectively. In each case, the aperture length represents the length of time that radar data is collected for the images.

Simulated data is generated from flight and radar software developed in MATLAB. The software has the capability to generate a flight trajectory, populate the ground with radar targets, and form both truth and distorted (from initial errors) SAR images via the BPA from synthesized radar data [19, 40]. For the real datasets, the radar data is obtained from flight tests of the Space Dynamics Laboratory X-band radar system [41]. A sub-centimeter-accurate, post-processed trajectory is used as truth, and corrupted to synthesize distorted SAR images.

3.2 Data Preprocessing

The brightness of each pixel of a SAR image represents the reflectivity of all targets within that pixel's region. The intensities of the pixels are inherently right-skewed and subject to changes in magnitude based on the geometric conditions. Thus, we preprocessed the images and errors as follows.

Let X be an input image, X' the corresponding reference image, and y_{α_β} the corresponding navigational error vector. We first log-transform each pixel in each image (including the reference images): $L = \log_{10}(X)$ as is common practice in SAR images. We then standardize each pixel in the image to ensure standardized inputs to the neural network by subtracting the mean and dividing by the standard deviation: $Z = \frac{L-\mu_L}{\sigma_L}$, where μ_L

and σ_L are estimated on an image-by-image basis. Each of the navigational errors y_{α_β} are measured on different scales. We therefore standardize the navigational errors as $\boldsymbol{s} = \frac{\boldsymbol{y}-\mu_{\boldsymbol{y}}}{\sigma_{\boldsymbol{y}}}$ to ensure equal consideration of all navigational errors during training. We estimate $\mu_{\boldsymbol{y}}$ and $\sigma_{\boldsymbol{y}}$ from the entire dataset. In practice, the standardization of the navigation parameters would use a mean of 0 (assuming unbiased navigation errors on average) and a standard deviation estimated from an extended Kalman filter (EKF) [42].

3.3 Data Splitting

For each of the simulated and real datasets, six different scenarios were studied, with varying numbers and combinations of active navigation errors, as shown in Table 3.1.

Scenario #	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel
1	x	x				
2				x	x	
3	x	x		x	x	
4	x	x	x			
5				x	x	x
6	x	x	x	x	x	x

Table 3.1: Summary of scenarios and corresponding initial errors. The network attempted to model of the indicated error states. In Scenario 7 (not shown), however, North, East, and Down Attitude errors were also present in the generated data, but the Neural Network only tried to predict North Position and East Position.

Dataset	Training		Validation		Testing	
	# targets	# images	# targets	# images	# targets	# images
Sim-5-sec	134	13500	19	1900	39	3800
Real-2-sec	130	13000	18	1800	37	3700
Real-10-sec	122	12300	17	1700	36	3500

Table 3.2: Data split details. The six datasets of the sim-5-sec, real-2-sec, and real-10-sec data were approximately split into a 70% training, 10% validation, and 20% testing set. As shown, for each of the six scenarios of the real-2-sec data, the training set had 130 unique target locations and 13000 distorted images.

For each of the six scenarios, a total of 192, 185, and 175 unique target locations were considered for the sim-5-sec, real-2-sec, and real-10-sec dataset, respectively. For each

target, we generated 100 distorted images of size 80×80 pixels, paired with the corresponding navigation errors. The datasets were then sorted by targets into training, validation, and testing sets as shown in Table 3.2. We train on a set of targets and validate and test on an entirely different set of targets. Thus, a network's performance on the test data reflects its ability to generalize to new locations.

CHAPTER 4

EXPLORATORY METHODS FOR ESTIMATING INITIAL ERRORS

We applied different methods to the problem such as linear regression, a simple CNN, a fully connected neural network, and several ResNet models to the sim-5-sec and real-2-sec datasets for scenarios 1-3. All of these models attempted to predict the initial navigation errors present in each scenario (e.g. AT Position and CT Position are the only errors present in scenario 1, and the models attempted to predict both error states). These models were constructed, trained, and tested using Python and the Pytorch framework [43].

4.1 Loss Function

We used the average mean squared error (MSE) loss function across all considered initial errors for training and for testing:

$$\text{MSE} = \frac{1}{mn} \sum_{\alpha=1}^{n} \sum_{\beta=1}^{m} \left(s_{\alpha_\beta} - \hat{s}_{\alpha_\beta} \right)^2,$$

where m is the number of error states considered, n is the number of training images in the data, s_{α_β} is the true standardized error of the βth error and the αth image, and $\hat{s}_{\alpha_\beta}$ is the corresponding error estimate by the model.

Due to standardization, the MSE for any prediction can be compared against the value of 1: if the network guesses that the initial error state is 0 or randomly guesses with a mean 0 and a similar standard deviation, then the network will have an estimated MSE of 1. If the network is accurately estimating the relationship between the reference image, distorted image, and initial navigational error states, then the MSE would be less than 1. This is a key benchmark for learning, as it indicates that where an MSE is less than 1, the neural network is learning relevant information for this task.

To prevent overfitting, we used L2 regularization, also known as weight decay. This regularization forces the weights to be reduced towards zero, limiting the risk of overfitting.

4.2 Linear Regression Model

Linear regression is a common predictive model to identify the relationship among variables [44]. In general, given the training data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the linear regression model is represented by the following:

$$\hat{y} = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} \tag{4.1}$$

where $\boldsymbol{x}$ is the feature variable (explanatory variable), $\hat{\boldsymbol{y}}$ is the target variable (response variable), and $\boldsymbol{w}$ are the weights.

For our problem, the linear regression model considers each pixel of the stacked image (stacking the normalized-differenced image, the reference image, and the distorted image) as explanatory variables with the standardized navigation errors $\boldsymbol{s}_\alpha$ of the αth image as the response.

4.2.1 Linear Regression Results

Table 4.1 shows a summary of the results for each scenario (1-3) applying the linear regression model to the sim-5-sec and real-2-sec dataset and its error states (for additional results for this model, see Appendix A subsection A.1).

Scenario #	Dataset	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	MSE (Sim-5-sec)	0.8855	0.6749	N/A	N/A	N/A	N/A	0.7802
	MSE (Real-2-sec)	1.0041	1.0425	N/A	N/A	N/A	N/A	1.0233
2	MSE (Sim-5-sec)	N/A	N/A	N/A	1.0636	0.6502	N/A	0.8569
	MSE (Real-2-sec)	N/A	N/A	N/A	1.0704	1.0560	N/A	1.0632
3	MSE (Sim-5-sec)	1.0534	0.5882	N/A	1.0564	0.4989	N/A	0.7993
	MSE (Real-2-sec)	1.0518	1.0251	N/A	1.0340	0.9738	N/A	1.0212

Table 4.1: Summary of Model Performance for each error state for scenarios 1-3 on the sim-5sec and real-2-sec dataset using the Linear Regression Model. As shown, all the MSE for this model are greater than 1 or almost 1 on the real-2-sec dataset. This indicates the linear model performs poorly and is unable to accurately predict the initial navigation errors.

The results show that the model perfoms well on the sim-5-sec dataset for scenario 1 since the MSE is less than 1 for all the considered errors in that scenario. However, when velocity errors are introduced, the MSE is greater than 1. Along with that, the MSE is greater than 1 or very close to it for all of the scenarios 1-3 on the real-2-sec dataset. These results suggest that a linear model is not sufficiently complex to accurately predict the initial navigation errors in any of these scenarios.

4.3 Simple CNN Model

Since CNNs have been used successfully on image classification and feature extraction [45], we applied a simple CNN model to the sim-5-sec and real-2-sec dataset for scenarios 1-3.

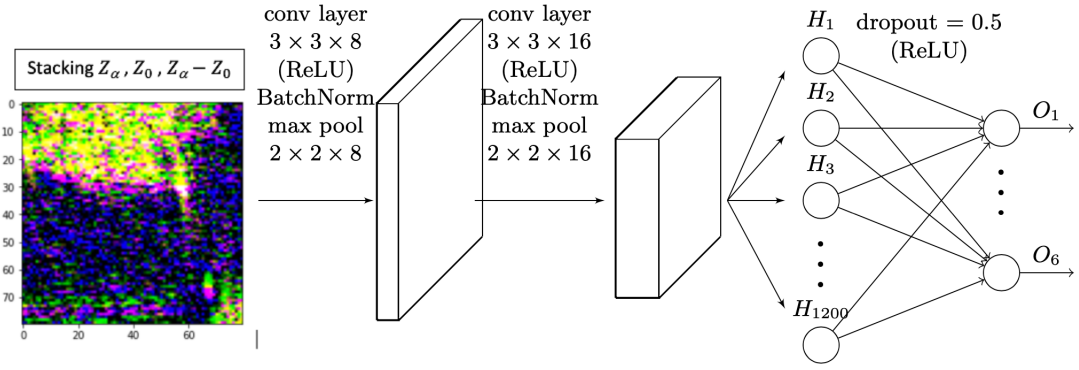

Fig. 4.1: Simple CNN Architecture. This network is a CNN with two convolutional layers followed by a fully connected hidden layer and a fully connected output layer.

We used a linear activation function in the output layer as we are predicting a regression response. We focused on tuning the learning rate, batch size, and number of epochs. Tuning the learning rate and batch size was performed using a grid search over candidate hyper-parameters retaining the model with the lowest average MSE for the validation set, while tuning the number of epochs was performed by tracking the validation accuracy over epochs and retaining the epoch with the lowest validation MSE. Dropout for fully connected layers was fixed at $p = 0.5$ for all fully connected layers during training to limit over-fitting.

4.3.1 Simple CNN Results

The summary of the results by applying this simple CNN model is shown in Table 4.2 (for additional results for this model, see Appendix A subsection A.2). As shown, this model performs well for all the scenarios on the sim-5-sec dataset. These results suggest that the neural network is capable of characterizing both shifts and blurs, in the absence of ambiguity. On the other hand, this model only performs well for scenario 1 on the real-2-sec dataset as the MSE is less than 1 for the considered errors (AT Pos and CT Pos). However, for scenarios 2 and 3, the MSE for some of the considered errors are greater than (or around) 1, suggesting that as the velocity errors are introduced, the performance degrades.

Scenario #	Dataset	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	MSE (Sim-5-sec)	0.0915	0.0539	N/A	N/A	N/A	N/A	0.0727
	MSE (Real-2-sec)	0.0569	0.0414	N/A	N/A	N/A	N/A	0.0492
2	MSE (Sim-5-sec)	N/A	N/A	N/A	0.4748	0.4019	N/A	0.4384
	MSE (Real-2-sec)	N/A	N/A	N/A	1.0949	0.1956	N/A	0.6453
3	MSE (Sim-5-sec)	0.7093	0.3353	N/A	0.5690	0.3385	N/A	0.4880
	MSE (Real-2-sec)	0.9985	0.1972	N/A	0.9850	0.1989	N/A	0.5949

Table 4.2: Summary of Model Performance for each error state for scenarios 1-3 of the sim-5-sec and real-2-sec dataset using a simple CNN Model. This model predicts the initial navigation errors very well on the sim-5-sec dataset (all MSE are less than 1). On the real-2-sec dataset, this model only performs well for scenario 1. For the other scenarios, 2 and 3, the MSE are greater than 1 for some of the error states, which shows the performance decays as velocity errors are present.

4.4 Fully Connected Model

Here we implemented a model that consists of a standard feedforward network with two fully connected hidden layers and a fully connected output layer as shown in Figure 4.2. As in the previous model, we used a linear activation function in the output layer. We focused on tuning the learning rate, batch size, and number of epochs as well. Tuning learning rate and batch size was performed using a grid search over candidate hyper-parameters retaining the model with the lowest average MSE for the validation set, while tuning the number of epochs was performed by tracking the validation accuracy over epochs and retaining the

epoch with the lowest validation MSE. Dropout for fully connected layers was fixed at $p = 0.5$ for all fully connected layers during training to limit over-fitting.

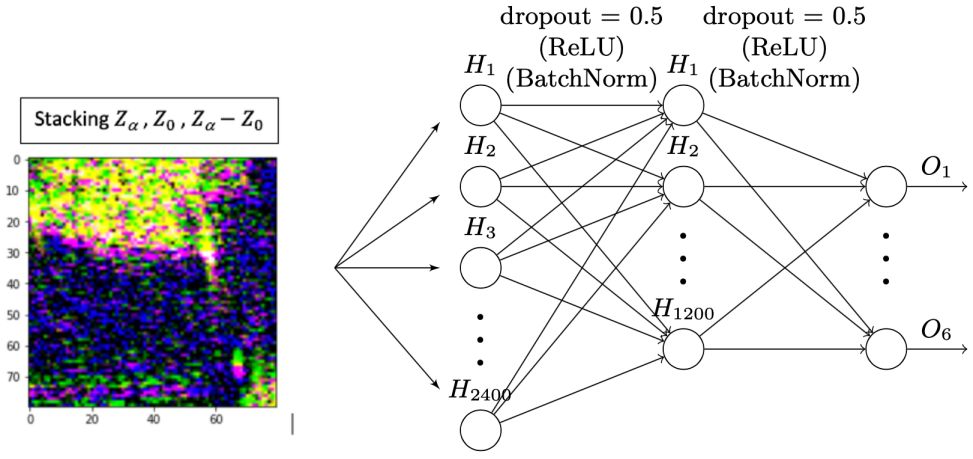

Fig. 4.2: Fully Connected Model. This neural network model consists of two fully connected hidden layers and a fully connected output layer.

4.4.1 Fully Connected Results

Table 4.3 shows a summary of the results using the fully connected model for each scenario (1-3) in the sim-5-sec and real-2-sec dataset and its error states (for additional results for this model, please see Appendix A subsection A.3). As we can see on the table, this model performs well in scenario 1 on the sim-5-sec dataset.

Scenario #	Dataset	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	MSE (Sim-5-sec)	0.5600	0.2336	N/A	N/A	N/A	N/A	0.3968
	MSE (Real-2-sec)	1.0658	1.0762	N/A	N/A	N/A	N/A	1.0710
2	MSE (Sim-5-sec)	N/A	N/A	N/A	1.1543	0.4250	N/A	0.7897
	MSE (Real-2-sec)	N/A	N/A	N/A	1.4362	1.1074	N/A	1.2718
3	MSE (Sim-5-sec)	1.1355	0.3600	N/A	1.1278	0.3501	N/A	0.7434
	MSE (Real-2-sec)	1.4322	0.9967	N/A	1.3948	0.9754	N/A	1.1998

Table 4.3: Summary of Model Performance for each error state for scenarios 1-3 of the sim-5-sec and real-2-sec dataset using the fully connected Model. This model performs poorly on the real-2-sec dataset as all the MSE are around or equal to 1 for scenarios 1-3. On the sim-5-sec dataset, this model only performs well in scenario 1 and showing most of the MSE greater than 1 for scenarios 2 and 3.

However, for scenario 2 and 3, some of the error states present an MSE greater than 1, which indicates the performance degrades as more errors are introduced. On the real-2-sec dataset, the MSE is greater than 1 for all the considered scenarios (1-3). These results suggest that the fully connected model is not learning how to predict the initial navigation errors as desired.

CHAPTER 5

RESNET ARCHITECTURES FOR ESTIMATING INITIAL ERRORS

As shown in section 4.2, the linear regression model didn't perform well for any of the three scenarios on the real-2-sec dataset. On the other hand, the results in subsection 4.3.1 show that a basic CNN model is able to learn the initial navigation errors in some scenarios. Thus, we studied more advanced CNN architectures.

5.1 ResNet Models considered

CNNs were chosen because they have been successful on various problems involving images. The convolutional filters that are part of CNNs allow for the network to learn short and long range dependencies between image pixels. Beyond the previous neural network models tested, more advanced network architectures were considered:

- ResNet 34 [36]
- ResNet 50 [36]
- ResNet 101 [36]
- ResNet 152 [36]
- Wide ResNet 50_2 [39]
- Wide ResNet 101_2 [39]

Note that each of these architectures is built with the same basic idea: the residual block as shown in Figure 2.4. This block allows networks to be much deeper (more layers and hence better at dealing with images) while simultaneously mitigating the issue of the vanishing gradient [36]. The residual block technique is still considered a state-of-the-art way of dealing with image data.

5.2 Proposed Method based on ResNet Models

For all the ResNet approaches, our input consists of a 3-channel image with the distorted image, the reference image, and the difference image as each of the channels. This 3-channel image input is fed into a randomly initialized convolutional layer followed by the considered ResNet architecture. The final layer of ResNet is replaced with a fully connected layer with the same number of outputs as error states as shown in Figure 5.1.

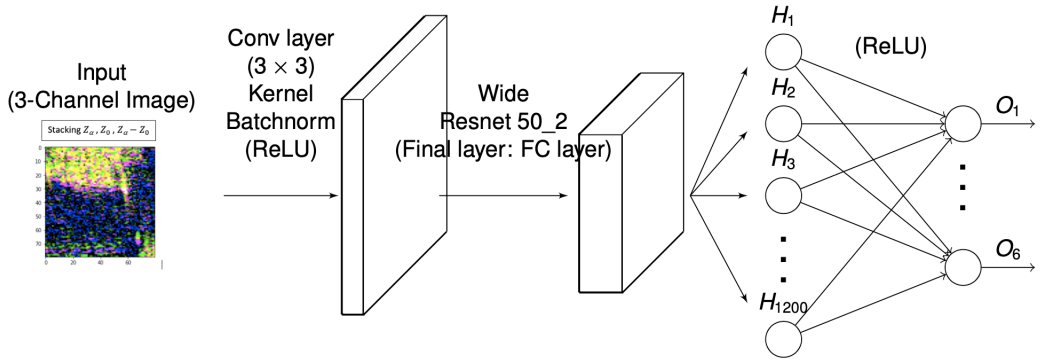

Fig. 5.1: Architecture of the Wide ResNet 50_2 approach. This architecture illustrates an example of the features used in our proposed method based on ResNet models. The input consists of a 3-channel image with the distorted image, the reference image, and the difference image, which is fed into a randomly initialized convolutional layer followed by the considered ResNet architecture (e.g. Wide ResNet 50_2, ResNet-34). The final layer of ResNet is replaced with a fully connected layer with the same number of outputs as error states.

Each of the ResNet architectures listed above have pre-trained models available in the Pytorch framework. In the initial attempts at estimating initial navigational errors, we explored whether starting with pre-trained weights or randomly initializing the model's weights would produce the best results. Unsurprisingly, in all cases, the model with pre-trained weights outperformed models that had weights that were trained using only the reference chips and distorted images. We further discovered that starting with pre-trained weights but then allowing these weights to be modified specifically for the problem at hand produced better results than "freezing" the already pre-trained weights. This is remarkable

considering the datasets used for pretraining contained natural images, which differ considerably from SAR images. This suggests that the pretrained network has learned image features (e.g. lines and other shapes) that are also present in SAR images.

5.3 Selecting a ResNet Model based on performance

To compare the performance between the different ResNet models, we applied our proposed method to the sim-5-sec and real-2-sec dataset for scenarios 1-3.

Scenario #	Model (sim-5-sec data)	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	ResNet-34	0.0846	0.0313	N/A	N/A	N/A	N/A	0.0579
	ResNet-50	**0.0547**	0.0450	N/A	N/A	N/A	N/A	0.0498
	ResNet-101	0.0575	0.0387	N/A	N/A	N/A	N/A	0.0481
	ResNet-152	0.0570	0.0378	N/A	N/A	N/A	N/A	0.0474
	Wide ResNet 50_2	0.0594	**0.0289**	N/A	N/A	N/A	N/A	**0.0442**
	Wide ResNet 101_2	0.0651	0.0335	N/A	N/A	N/A	N/A	0.0493
2	ResNet-34	N/A	N/A	N/A	**0.2068**	0.1463	N/A	**0.1766**
	ResNet-50	N/A	N/A	N/A	0.2544	**0.1137**	N/A	0.1841
	ResNet-101	N/A	N/A	N/A	0.2529	0.2044	N/A	0.2286
	ResNet-152	N/A	N/A	N/A	0.2782	0.2710	N/A	0.2746
	Wide ResNet 50_2	N/A	N/A	N/A	0.2456	0.1162	N/A	0.1809
	Wide ResNet 101_2	N/A	N/A	N/A	0.3085	0.1886	N/A	0.2485
3	ResNet-34	0.7824	0.2671	N/A	0.6088	0.1737	N/A	0.4580
	ResNet-50	0.6140	0.2689	N/A	0.3003	**0.1151**	N/A	0.3246
	ResNet-101	0.5772	0.2552	N/A	0.2670	0.1290	N/A	0.3071
	ResNet-152	0.5722	0.2360	N/A	0.2959	0.1333	N/A	0.3094
	Wide ResNet 50_2	**0.5229**	**0.2237**	N/A	**0.2442**	0.1259	N/A	**0.2792**
	Wide ResNet 101_2	0.5972	0.2763	N/A	0.3004	0.1716	N/A	0.3364

Table 5.1: Summary of Model Performance for each error state for scenarios 1-3 on the sim-5-sec dataset using the ResNet and Wide ResNet Models. The bold numbers represent the smallest MSE of the considered error state for each scenario to indicate which model is doing the best. This shows that the Wide ResNet 50_2 model outperforms the other models more frequently than any other.

As seen in Table 5.1, all of the ResNet models performed very well in all the scenarios ($MSE < 1$) on the simulated data. Particularly the Wide ResNet 50_2 model outperformed the other models more frequently than any other (e.g. the Wide ResNet 50_2 model has the lowest MSE in most of the error states in scenario 3).

Scenario #	Model(real-2-sec data)	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	ResNet-34	0.0705	**0.0351**	N/A	N/A	N/A	N/A	0.0528
	ResNet-50	0.0586	0.0520	N/A	N/A	N/A	N/A	0.0553
	ResNet-101	0.0612	0.0465	N/A	N/A	N/A	N/A	0.0538
	ResNet-152	0.0925	0.0500	N/A	N/A	N/A	N/A	0.0713
	Wide ResNet 50_2	**0.0563**	0.0425	N/A	N/A	N/A	N/A	**0.0494**
	Wide ResNet 101_2	0.0618	0.0490	N/A	N/A	N/A	N/A	0.0554
2	ResNet-34	N/A	N/A	N/A	1.2545	0.0859	N/A	0.6702
	ResNet-50	N/A	N/A	N/A	1.1395	**0.0810**	N/A	**0.6103**
	ResNet-101	N/A	N/A	N/A	1.1430	0.1190	N/A	0.6310
	ResNet-152	N/A	N/A	N/A	1.1420	0.8100	N/A	0.6115
	Wide ResNet 50_2	N/A	N/A	N/A	**1.0729**	0.1683	N/A	0.6206
	Wide ResNet 101_2	N/A	N/A	N/A	1.1548	0.1196	N/A	0.6372
3	ResNet-34	1.0988	0.2000	N/A	1.1107	**0.1073**	N/A	0.6292
	ResNet-50	1.1881	0.1850	N/A	1.1508	0.1179	N/A	0.6604
	ResNet-101	1.1558	0.2171	N/A	1.1049	0.1256	N/A	0.6508
	ResNet-152	1.1251	0.2235	N/A	1.1408	0.1171	N/A	0.6516
	Wide ResNet 50_2	**1.0657**	0.2340	N/A	**1.0682**	0.1468	N/A	**0.6287**
	Wide ResNet 101_2	1.1933	**0.1725**	N/A	1.1244	0.1234	N/A	0.6534

Table 5.2: Summary of Model Performance for each error state for scenarios 1-3 on the real-2-sec dataset using the ResNet and Wide ResNet Models. The bold numbers represent the smallest MSE of the considered error state for each scenario to indicate which model is doing the best. As on the simulated data, the Wide ResNet 50_2 model outperforms the other models more frequently than any other on the real-2-sec data as well.

Table 5.2 shows the results obtained applying the proposed approach to the real-2-sec data. The outcomes obtained suggested that all of the considered ResNet architectures had similar performance for scenarios 1-3. In scenario 1, all the MSE of the ResNet models were less than 1 but in scenarios 2 and 3, some of the error states got a $MSE > 1$. Notably the Wide ResNet 50_2 model outperformed the other models more frequently than any other on both the sim-5-sec and real-2-sec data.

Hence, after testing these architectures, we settled on the Wide ResNet 50_2 network because it outperformed all other models and because its wide nature enables faster training and therefore more time can be devoted to trying specific hyper-parameters (for additional results, see Appendix A subsection A.4).

5.4 Wide ResNet 50_2 Application on the simulated and real-2-sec data

To continue evaluating the performance of our proposed method based on the Wide ResNet 50_2 model, we trained on the sim-5-sec and real-2-sec data for scenarios 1-6 using the split described in Table 3.2.

Scenario #	Dataset	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	MSE (Sim-5-sec)	0.0594	0.0289	N/A	N/A	N/A	N/A	0.0442
	MSE (Real-2-sec)	0.0563	0.0425	N/A	N/A	N/A	N/A	0.0494
2	MSE (Sim-5-sec)	N/A	N/A	N/A	0.2456	0.1162	N/A	0.1809
	MSE (Real-2-sec)	N/A	N/A	N/A	1.0729	0.1683	N/A	0.6206
3	MSE (Sim-5-sec)	0.5229	0.2237	N/A	0.2442	0.1259	N/A	0.2792
	MSE (Real-2-sec)	**1.0657**	0.2340	N/A	**1.0682**	0.1468	N/A	0.6287
4	MSE (Sim-5-sec)	0.0941	0.9204	0.2800	N/A	N/A	N/A	0.4315
	MSE (Real-2-sec)	0.1020	0.8102	0.3734	N/A	N/A	N/A	0.4285
5	MSE (Sim-5-sec)	N/A	N/A	N/A	0.2834	0.7982	0.5460	0.5425
	MSE (Real-2-sec)	N/A	N/A	N/A	**1.0699**	0.6459	0.5795	0.7651
6	MSE (Sim-5-sec)	0.6321	0.9139	0.5245	0.3677	0.8661	0.5331	0.6396
	MSE (Real-2-sec)	**1.0846**	0.7509	0.5353	**1.0868**	0.6039	0.5984	0.7766

Table 5.3: Summary of Model Performance for each error state for scenarios 1-6 of the sim-5-sec and real-2-sec datasets. The network is able to learn many of the errors for both simulated and real data ($MSE < 1$), but it still has some difficulties in resolving error sources in certain scenarios shown in bold numbers ($MSE > 1$).

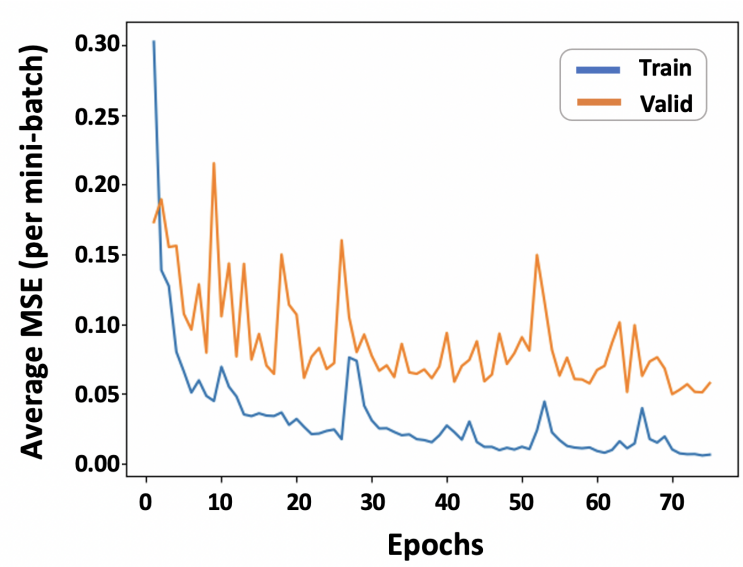

Fig. 5.2: Training and validation MSE as a function of training epoch for scenario 1 on the real-2-sec dataset using the proposed method. The gap between the errors is small.

Table 5.3 lists the results obtained in all six scenarios for both the sim-5-sec and the real-2-sec dataset. For all scenarios in the simulated dataset, the MSE is less than 1 for all considered errors. In particular, the simulated dataset performs well for scenarios 1 and 2, where no ambiguity exists. These results suggest that the neural network is capable of characterizing both shifts and blurs in the absence of ambiguity. However, as more errors are introduced, the performance degrades, suggesting difficulties in resolving ambiguous error sources.

In the real dataset, the MSE is below 1 for all errors in scenarios 1 and 4. The network performs particularly well in scenario 1 with a low MSE (see also Figure 5.2). It is noteworthy that for scenario 4, ambiguity exists in CT shifts, due to either CT or D position errors. The network, however, is able to identify subtleties in the distorted SAR images and properly attribute the effect to the corresponding error.

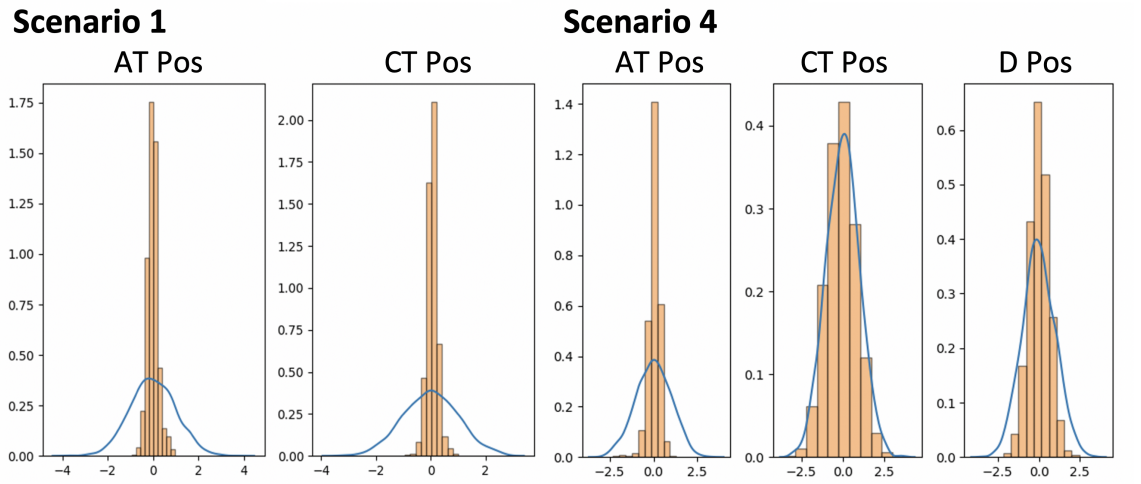

Fig. 5.3: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenarios 1 and 4. In both cases, the error distributions are more concentrated around zero, indicating that the network has improved our ability to estimate navigation errors. No outliers have been introduced by the network.

Figure 5.3 shows the distribution of the error states before and after estimation of the navigation errors for scenarios 1 and 4. The resulting error distributions are more concentrated around zero, indicating that the neural network is able to improve our knowledge

of the initial error states. Furthermore, no outliers are introduced by the neural network. However, as was the case for the simulated dataset, increasing numbers of ambiguous error sources results in degraded overall performance, with an MSE near 1.0 in cases where little information is learned by the network.

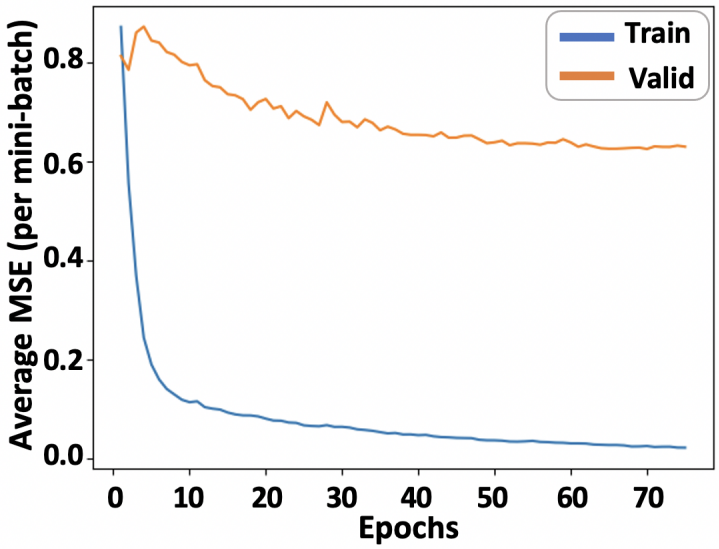

Fig. 5.4: Training and validation MSE as a function of training epoch for scenario 2 for the real-2-sec dataset using the propodes method. There is a large gap between the training and validation error, suggesting the network may be overfitting. Compare with Figure 5.2, where little overfitting is occurring in scenario 1.

For some of these errors, the network may also be overfitting. Figure 5.4 shows the training and validation MSE as a function of training epoch for scenario 2. Here, we see that the average training error is very low, indicating that the network is able to accurately predict both errors on the training set. However, the validation MSE is considerably higher, suggesting that overfitting is occurring. Increasing the amount of training data, especially adding more targets, will likely reduce the tendency to overfit.

5.5 Transfer Learning Technique with Simulated Data

Since the network was having difficulties in resolving some error sources in certain scenarios of the real-2-sec dataset (see Table 5.3), we hypothesized that transfer learning would help in improving the performance in those scenarios.

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [46]. For example, using the pretrained Wide ResNet 50_2 is a form of transfer learning. The goal of using transfer learning is to fully learn the target task given the transferred knowledge instead of learning from scratch. It can often speed up the training and improve the performance of the model.

To do transfer learning in this approach, we pretrained the network (which was already pretrained on other image data) on the sim-5-sec data and then trained on the real-2-sec data. We then applied this technique to the real-2-sec data for scenario 2 to evaluate and compare the performance.

Dataset, Scenario 2	AT Vel	CT Vel	Average
MSE (Real-2-sec)	1.0729	0.1683	0.6206
MSE (Transfer Learning - 2 sec)	1.0429	0.0849	0.5639

Table 5.4: Summary of Model Performance for the error states present in scenario 2. This table shows: 1) the results of applying the proposed model on the real-2-sec; 2) the results of pretraining on simulated data and training on real-2-sec data. Improvements from transfer learning are negligible.

Table 5.4 shows the performance for scenario 2, showing negligible improvement in both the AT and CT velocity MSE, suggesting that the simulated and real data are not similar enough to justify transfer learning (for additional results, see Appendix C subsection C.1).

5.6 Increase of the aperture length on the real data

Most of the errors that the previous networks struggled with are associated with image blurring. Blurring errors are more apparent with longer flight trajectories. This fact motivates an increase in the aperture length to emphasize the blurring distortion in the images.

5.6.1 Real data with a 10-second synthetic aperture length

Dataset, Scenario 2	AT Vel	CT Vel	Average
MSE (Real-2-sec)	1.0729	0.1683	0.6206
MSE (Real-10-sec)	0.7895	0.0812	0.4354
MSE (Transfer Learning - 2 sec)	1.0429	0.0849	0.5639
MSE (Transfer Learning - 10 sec)	0.9004	0.0851	0.4927

Table 5.5: Summary of Model Performance for the error states present in scenario 2. This table shows: 1) the results of applying the proposed model on the real-2-sec and real-10-sec data; 2) the results of pretraining on simulated data and training on real-2-sec (and real-10-sec) datasets. Going from 2 seconds to 10 seconds improves the performance. Improvements from transfer learning are better when using the 10-second aperture length ($MSE < 1$).

We analyzed the real data with a 10-second synthetic aperture length in scenario 2. Table 5.5 shows that for both errors AT Vel and CT Vel the $MSE < 1$ on the real-10-sec data. Also, when we applied the transfer learning technique to this real dataset, a slight improvement was shown in the AT Vel MSE.

Table 5.6 shows the results for all six scenarios. In scenario 2 of the real-2-sec dataset the network was not able to characterize blur distortions for the 2-second synthetic aperture despite the lack of error ambiguity. Comparing those results to the results obtained using the 10-second aperture, in the latter case the MSE for the North velocity (AT Vel) error decreased by more than 25%, suggesting that the neural network successfully characterized the effect of blurs for the larger apertures. Notably, the neural network also improves its estimation of the East velocity (CT Vel) error by a factor of 2.

Scenario #	Dataset	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	MSE (Sim-5-sec)	0.0594	0.0289	N/A	N/A	N/A	N/A	0.0442
	MSE (Real-2-sec)	0.0563	0.0425	N/A	N/A	N/A	N/A	0.0494
	MSE (Real-10-sec)	0.1808	0.1338	N/A	N/A	N/A	N/A	0.1573
2	MSE (Sim-5-sec)	N/A	N/A	N/A	0.2456	0.1162	N/A	0.1809
	MSE (Real-2-sec)	N/A	N/A	N/A	1.0729	0.1683	N/A	0.6206
	MSE (Real-10-sec)	N/A	N/A	N/A	0.7895	0.0812	N/A	0.4354
3	MSE (Sim-5-sec)	0.5229	0.2237	N/A	0.2442	0.1259	N/A	0.2792
	MSE (Real-2-sec)	1.0657	0.2340	N/A	1.0682	0.1468	N/A	0.6287
	MSE (Real-10-sec)	0.8864	0.2924	N/A	0.7894	0.1319	N/A	0.5250
4	MSE (Sim-5-sec)	0.0941	0.9204	0.2800	N/A	N/A	N/A	0.4315
	MSE (Real-2-sec)	0.1020	0.8102	0.3734	N/A	N/A	N/A	0.4285
	MSE (Real-10-sec)	0.2694	0.8165	0.4186	N/A	N/A	N/A	0.5015
5	MSE (Sim-5-sec)	N/A	N/A	N/A	0.2834	0.7982	0.5460	0.5425
	MSE (Real-2-sec)	N/A	N/A	N/A	1.0699	0.6459	0.5795	0.7651
	MSE (Real-10-sec)	N/A	N/A	N/A	0.9072	0.6453	0.5781	0.7102
6	MSE (Sim-5-sec)	0.6321	0.9139	0.5245	0.3677	0.8661	0.5331	0.6396
	MSE (Real-2-sec)	1.0846	0.7509	0.5353	1.0868	0.6039	0.5984	0.7766
	MSE (Real-10-sec)	0.9875	0.7382	0.5102	0.9447	0.6233	0.5113	0.7192

Table 5.6: Summary of Model Performance for each error state for scenarios 1-6 of the sim-5-sec, the real-2-sec and the real-10-sec datasets. In all of the scenarios the $MSE < 1$ for the real-10-sec dataset. This shows that increasing the aperture length improves performance with blur-related errors.

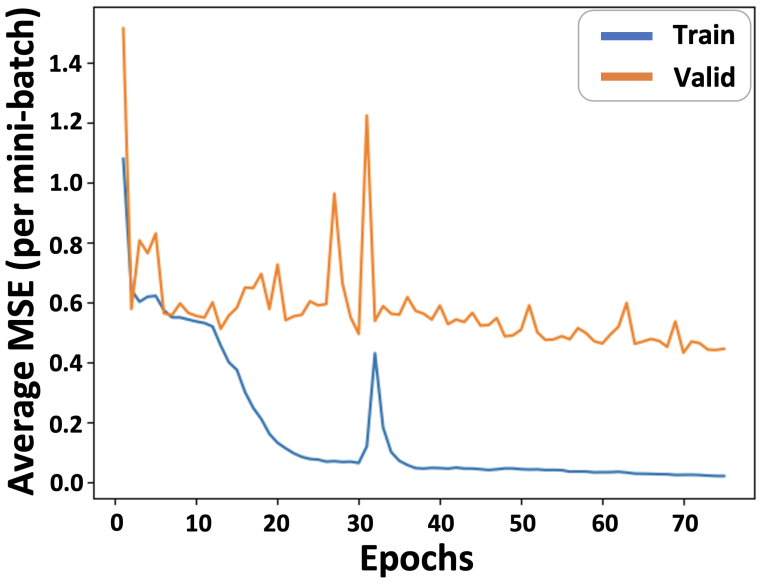

Fig. 5.5: Training and validation MSE as a function of training epoch for scenario 2 for the real-10-sec dataset using the proposed method. Compared to Figure 5.4, here the gap between the training and validation error is smaller, which shows that the network may be overfitting less in scenario 2 for the real-10-sec data.

Further, in Figure 5.5, we can see that the gap between the the training and validation error for scenario 2 is smaller than the gap shown on the figure for the real-2-sec data (Figure 5.4), this suggests that the overfitting occurs less on the real-10-sec data.

In fact, the MSE for all scenarios on the real-10-sec dataset is less than 1. This shows that the neural network performs better when the errors are more apparent. These results identify a trend of improved performance on real datasets with increasing aperture length (for additional results, see Appendix B subsection B.2).

5.6.2 Real data with a 6-second and 15-second synthetic aperture length

As demonstrated in Table 5.6, as the aperture length increases, the network performs better in some settings. To examine this further, we also tested our neural network approach using the real datasets with an aperture length of 6-seconds and 15-seconds for scenarios 1-3.

Scenario #	Dataset	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Average
1	MSE (Sim-5-sec)	0.0594	0.0289	N/A	N/A	N/A	N/A	0.0442
	MSE (Real-2-sec)	0.0563	0.0425	N/A	N/A	N/A	N/A	0.0494
	MSE (Real-6-sec)	0.1705	0.1326	N/A	N/A	N/A	N/A	0.1516
	MSE (Real-10-sec)	0.1808	0.1338	N/A	N/A	N/A	N/A	0.1573
	MSE (Real-15-sec)	0.1967	0.1296	N/A	N/A	N/A	N/A	0.1632
2	MSE (Sim-5-sec)	N/A	N/A	N/A	0.2456	0.1162	N/A	0.1809
	MSE (Real-2-sec)	N/A	N/A	N/A	1.0729	0.1683	N/A	0.6206
	MSE (Real-6-sec)	N/A	N/A	N/A	1.0330	0.1389	N/A	0.5859
	MSE (Real-10-sec)	N/A	N/A	N/A	0.7895	0.0812	N/A	0.4354
	MSE (Real-15-sec)	N/A	N/A	N/A	0.7933	0.0809	N/A	0.4371
3	MSE (Sim-5-sec)	0.5229	0.2237	N/A	0.2442	0.1259	N/A	0.2792
	MSE (Real-2-sec)	1.0657	0.2340	N/A	1.0682	0.1468	N/A	0.6287
	MSE (Real-6-sec)	1.0193	0.2177	N/A	0.9918	0.1995	N/A	0.6071
	MSE (Real-10-sec)	0.8864	0.2924	N/A	0.7894	0.1319	N/A	0.5250
	MSE (Real-15-sec)	0.7787	0.2386	N/A	0.7755	0.1276	N/A	0.4801

Table 5.7: Summary of Model Performance for each error state for scenarios 1-3 of the sim-5-sec, the real-2-sec, real-6-sec, real-10-sec and real-15-sec datasets. The network is able to learn many of the errors for both simulated and real data ($MSE < 1$). In particular, the MSE is less than 1 for all the scenarios for the real-10-sec and real-15-sec datasets.

Table 5.7 shows the experimental results on these real datasets and compares them with the previous results. For instance, in scenario 2 and 3, the performance improves

slightly going from 2 seconds to 6 seconds of aperture length of the real data.

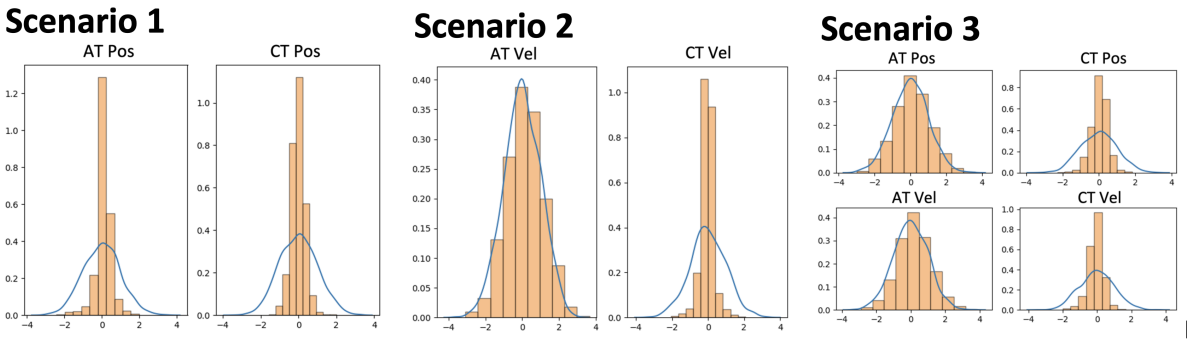

Fig. 5.6: Distribution of Error States before (blue line) and after (histogram) estimation for the real-6-sec dataset for scenarios 1-3. Our proposed method performed well, in particular in scenario 1.

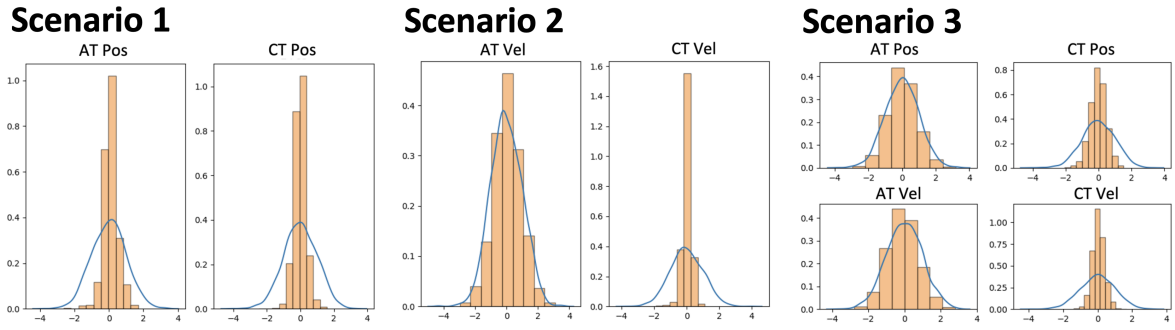

Fig. 5.7: Distribution of Error States before (blue line) and after (histogram) estimation for the real-15-sec dataset for scenarios 1-3. Our proposed method performed very well in all the scenarios (1-3).

Comparing Figure 5.6 and Figure 5.7, we can see that as the aperture length increases, the network performs even better in scenarios 2 and 3. Going from 6 seconds to 15 seconds of aperture length, the AT Pos error MSE decreased by more than 20% in scenario 3. Furthermore, the AT Vel error MSE decreased by more than 20% as well in scenarios 2 and 3, suggesting again that the neural network successfully characterized the effect of blurs for the larger apertures.

In fact, in all of the scenarios for the real-10-sec and real-15-sec datasets, the MSE is

less than 1. These results confirm our hypothesis: As the synthetic aperture length of the real data increases, the performance of our proposed method improves in certain scenarios (for additional results, see Appendix B subsection B.1, subsection B.2, and subsection B.3).

CHAPTER 6

FUTURE WORK AND CONCLUSION

We used a convolutional neural network to estimate position and velocity errors at the beginning of a SAR data collection period, by comparison of a distorted SAR image to an *a priori* SAR reference image. Performance was assessed on both simulated and real SAR data. In general, the network performs well in the absence of ambiguous error sources, reducing the MSE of the active navigation errors. As the number of error sources increases, the performance degrades due to the inability to separate errors which result in the same image distortion. In the case of the real data, however, the network successfully separated CT shifts caused by CT and D position errors. This exciting result suggests that the network is able to identify subtleties in the image distortions that are not readily apparent to the human eye. Another key finding in the testing of the network on real data is the sensitivity of estimation performance on the length of the synthetic aperture. As the length increases, the network successfully characterizes blurring in real images and appropriately attributes it to the corresponding error source.

Future work includes further investigation of the effect of the aperture length on the network's ability to learn. Furthermore, the sensitivity of distortions to the vehicle/target geometry may be exploited to reduce ambiguities and improve performance. Future approaches may, therefore, augment the training data with information about the viewing geometry, including the estimated vehicle position at the beginning/end of the synthetic aperture, the location of known targets, or the time history of barometric altimeter measurements. Finally, including more training data in future iterations will likely help mitigate over-fitting.

REFERENCES

[1] J. Raquet and R. K. Martin, "Non-GNSS radio frequency navigation," in *ICASSP*, Mar. 2008, pp. 5308–5311.

[2] M. J. Veth, "Navigation Using Images, A Survey of Techniques," *Navigation*, vol. 58, no. 2, pp. 127–139, 2011.

[3] E. B. Quist, P. C. Niedfeldt, and R. W. Beard, "Radar odometry with recursive-RANSAC," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 4, pp. 1618–1630, Aug. 2016.

[4] E. B. Quist and R. W. Beard, "Radar odometry on fixed-wing small unmanned aircraft," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 1, pp. 396–410, Feb. 2016.

[5] A. F. Scannapieco, A. Renga, G. Fasano, and A. Moccia, "Experimental Analysis of Radar Odometry by Commercial Ultralight Radar Sensor for Miniaturized UAS," *J. Intell. Robot. Syst*, vol. 90, no. 3-4, pp. 485–503, June 2018.

[6] P. Samczynski and K.S. Kulpa, "Coherent MapDrift Technique," *IEEE Trans. Geosci. Remote Sens*, vol. 48, no. 3, pp. 1505–1517, Mar. 2010.

[7] P. Samczynski, "Superconvergent Velocity Estimator for an Autofocus Coherent MapDrift Technique," *IEEE Geosci. Remote Sense. Lett*, vol. 9, no. 2, pp. 204–208, Mar. 2012.

[8] L. Hostetler and R. Andreas, "Nonlinear Kalman filtering techniques for terrain-aided navigation," *IEEE Trans. Automat. Contr*, vol. 28, no. 3, pp. 315–323, Mar. 1983.

[9] N. Bergman, L. Ljung, and F. Gustafsson, "Terrain navigation using Bayesian statistics," *IEEE Control Syst. Mag*, vol. 19, no. 3, pp. 33–40, June 1999.

[10] P.-J. Nordlund and F. Gustafsson, "Marginalized Particle Filter for Accurate and Reliable Terrain-Aided Navigation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 45, no. 4, pp. 1385–1399, Oct. 2009.

[11] Y Kim, J Park, and H Bang, "Terrain-Referenced Navigation using an Interferometric Radar Altimeter: IRA-Based TRN," *Navigation*, vol. 65, no. 2, pp. 157–167, June 2018.

[12] Z Sjanic and F Gustafsson, "Simultaneous navigation and SAR auto-focusing," in *FUSION*, July 2010, pp. 1–7.

[13] M. Greco, S. Querry, G. Pinelli, K. Kulpa, P. Samczynski, D. Gromek, A. Gromek, M. Malanowski, B. Querry, and A. Bonsignore, "SAR-based augmented integrity navigation architecture," in *IRS*, May 2012, pp. 225–229.

[14] D Nitti, F Bovenga, M Chiaradia, M Greco, and G Pinelli, "Feasibility of Using Synthetic Aperture Radar to Aid UAV Navigation," *Sensors*, vol. 15, no. 8, pp. 18334–18359, July 2015.

[15] E. C. Zaugg and D. G. Long, "Generalized Frequency Scaling and Backprojection for LFM-CW SAR Processing," *IEEE Trans. Geosci. Remote Sens*, vol. 53, no. 7, pp. 3600–3614, July 2015.

[16] W. G. Carrara, R. S. Goodman, and R.d M. Majewski, *Spotlight SAR: Signal processing algorithms*, Artech House, Boston, 1995.

[17] F Ulaby and D Long, *Microwave Radar and Radiometric Remote Sensing*, University of Michigan Press, Ann Arbor, 2014.

[18] R. S. Christensen, J. Gunther, and D. Long, "Toward gps-denied navigation utilizing back projection-based synthetic aperture radar imagery," in *Proceedings of the ION 2019 Pacific PNT Meeting*, 2019, pp. 108–119.

[19] C. Lindstrom, R. Christensen, and J. Gunther, "Sensitivity of bpa sar image formation to initial position, velocity, and attitude navigation errors," *arXiv*, vol. abs/2009.10210, 2020.

[20] V Berisha, N Shah, D Waagen, H Schmitt, S Bellofiore, A Spanias, and D Cochran, "Sparse manifold learning with applications to sar image classification," in *ICASSP*. IEEE, 2007, vol. 3, pp. III–1089.

[21] S Chen and H Wang, "Sar target recognition based on deep learning," in *DSAA*. IEEE, 2014, pp. 541–547.

[22] S Chen, H Wang, F Xu, and Y.-Q. Jin, "Target classification using the deep convolutional networks for sar images," *IEEE Trans. Geosci. Remote Sens*, vol. 54, no. 8, pp. 4806–4817, 2016.

[23] M Wilmanski, C Kreucher, and J Lauer, "Modern approaches in deep learning for sar atr," in *Algorithms for SAR imagery*, 2016, vol. 9843, p. 98430N.

[24] Q Zhao and J. C. Principe, "Support vector machines for sar automatic target recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 2, pp. 643–654, 2001.

[25] B Krishnapuram, D Williams, Y Xue, L Carin, M Figueiredo, and A. J. Hartemink, "On semi-supervised classification," in *NeurIPS*, 2005, pp. 721–728.

[26] M Gong, J Zhao, J Liu, Q Miao, and L Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 27, no. 1, pp. 125–138, 2016.

[27] E Mason, B Yonel, and B Yazici, "Deep learning for radar," in *RadarConf*. IEEE, 2017, pp. 1703–1708.

[28] A. O. Hero, B. Ma, O. JJ. Michel, and J. Gorman, "Applications of entropic spanning graphs," *IEEE Signal Process. Mag*, vol. 19, no. 5, pp. 85–95, 2002.

[29] M. E. Tipping and C. M. Bishop, "Bayesian image super-resolution," in *NeurIPS*, 2003, pp. 1303–1310.

[30] L. C. Pickup, D. P. Capel, S. J. Roberts, and A. Zisserman, "Bayesian image super-resolution, continued," in *NeurIPSs*, 2007, pp. 1089–1096.

[31] M. V. Wyawahare, P. M. Patil, H. K. Abhyankar, et al., "Image registration techniques: an overview," *Int. J. Signal Process. Image Process. and Patt. Recog*, vol. 2, no. 3, pp. 11–28, 2009.

[32] Teresa White, Jesse Wheeler, Colton Lindstrom, Randall Christensen, and Kevin R. Moon, "Gps-denied navigation using sar images and neural networks," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 2395–2399.

[33] Teresa White, Jesse Wheeler, Colton Lindstrom, Randall Christensen, and Kevin R. Moon, "Gps-denied navigation using sar images and neural networks," 2020.

[34] Michael Nielsen, "Neural networks and deep learning," http://neuralnetworksanddeeplearning.com/, 2015.

[35] David Stutz, "Learning shape completion from bounding boxes with cad shape priors," http://davidstutz.de/, September 2017.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012, pp. 1097–1105.

[38] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep CNN for LVCSR," in *ICASSP*. IEEE, 2013, pp. 8614–8618.

[39] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv*, vol. abs/1605.07146, 2016.

[40] M. I. Duersch and D. G. Long, "Analysis of time-domain back-projection for stripmap SAR," *International Journal of Remote Sensing*, vol. 36, no. 8, pp. 2010–2036, Apr. 2015.

[41] Mark Jensen, Chad Knight, and Brent Haslem, "FlexSAR, a high quality, flexible, cost effective, prototype SAR system," in *Radar Sensor Technology XX*. 2016, vol. 9829, SPIE.

[42] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng*, vol. 82, no. 1, pp. 35–45, 1960.

[43] A Paszke, S Gross, S Chintala, G Chanan, E Yang, Z DeVito, Z Lin, A Desmaison, L Antiga, and A Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[44] Alan J. Lee George A. F. Seber, "Linear regression analysis," Wiley Series in Probability and Statistics, 2012.

[45] Y LeCun, Léon Bottou, Y Bengio, P Haffner, et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] Shavlik J. Torrey L., "Transfer learning," in *In E. Olivas, J. Guerrero, M. Martinez-Sober, J. Magdalena-Benedito, A. Serrano López (Ed.), Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 2010, pp. pp. 242–264.

APPENDICES

APPENDIX A

Additional Results from chapter 5

A.1 Linear Regression Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

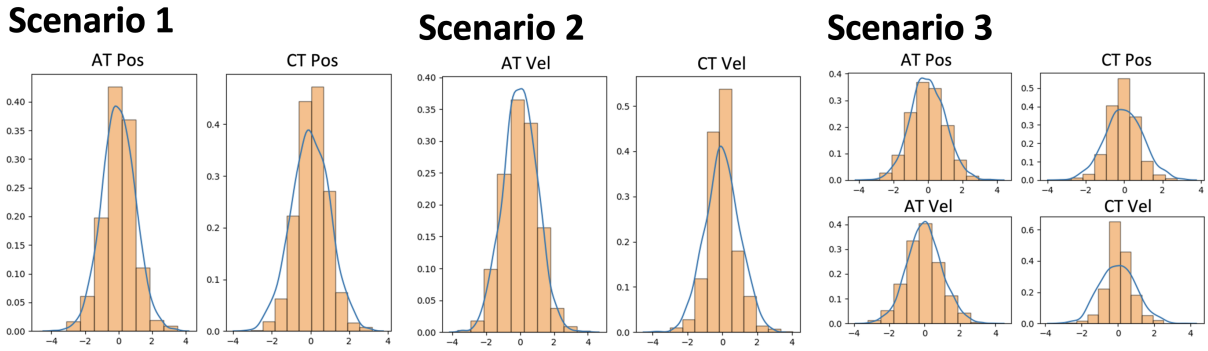

Fig. A.1: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the Linear Regression Model. This model performed better in scenario 1 on the simulated data but not on the other scenarios.

- **Real-2-sec Data for scenarios 1-3:**

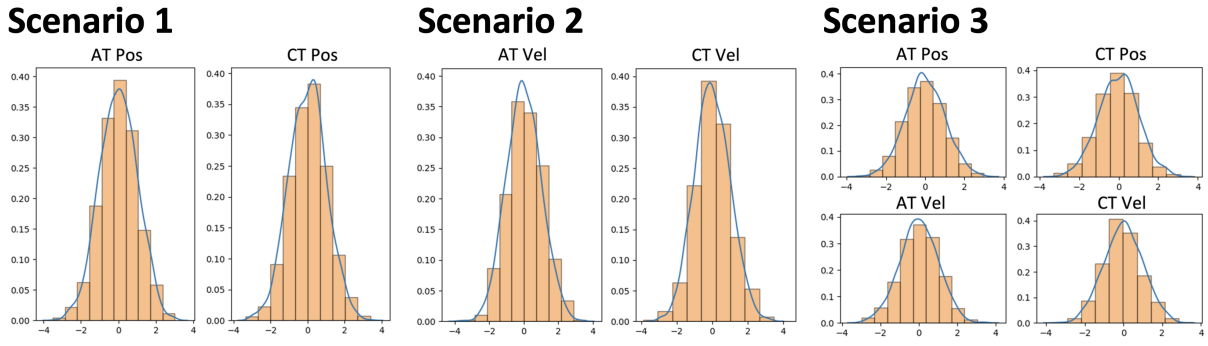

Fig. A.2: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the Linear Regression Model. This model didn't perform well in any scenario.

A.2 Simple CNN Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

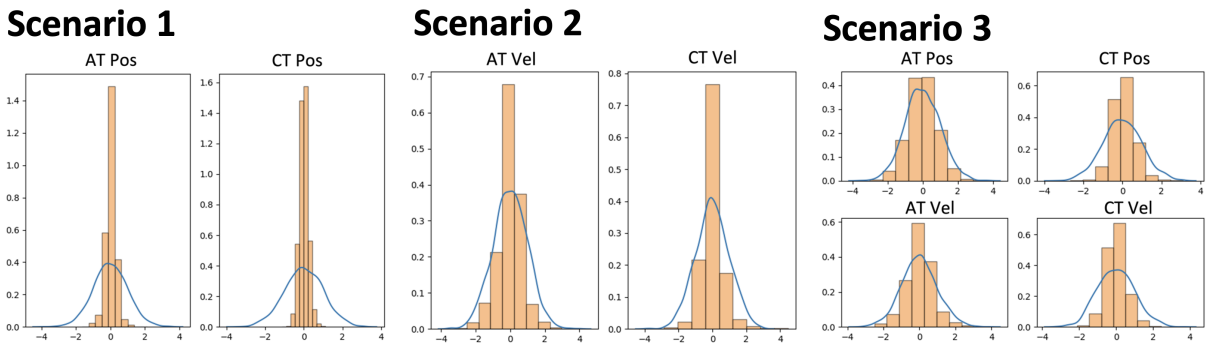

Fig. A.3: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the simple CNN Model. As shown, this model performed well in all the scenarios.

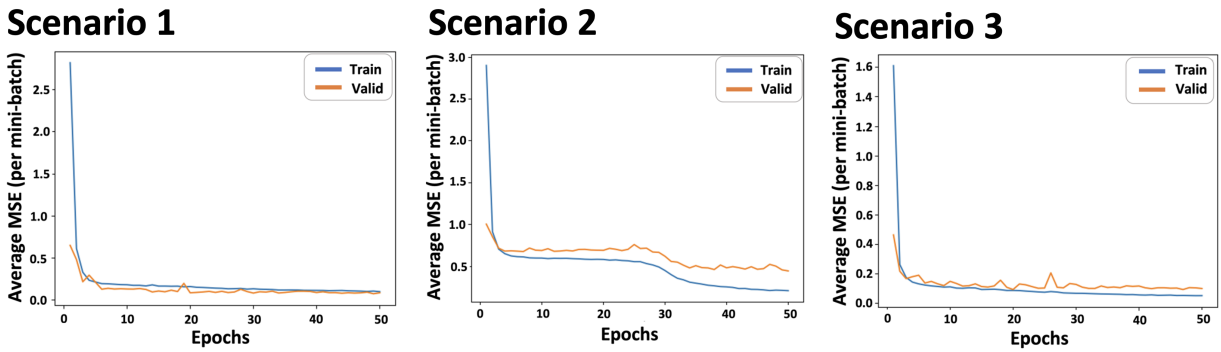

Fig. A.4: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the simple CNN Model. There is a very small gap between the training and validation error in scenarios 1 and 3. It is a little bit bigger in scenario 2.

- **Real-2-sec Data for scenarios 1-3:**

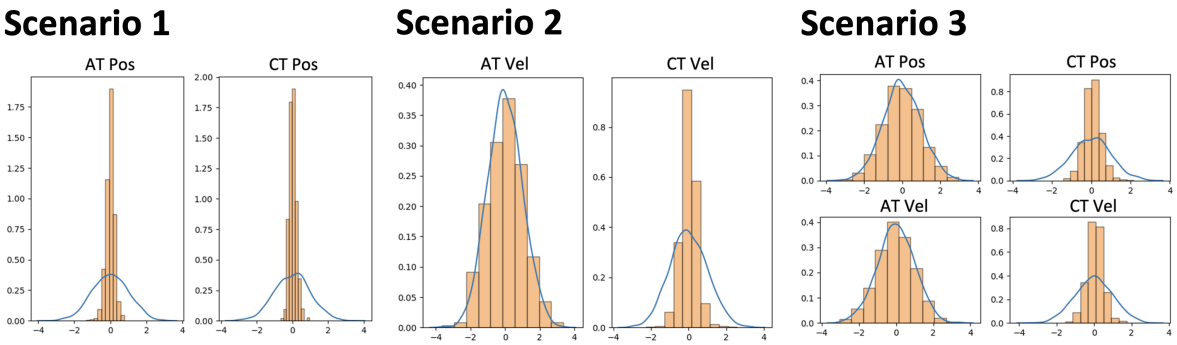

Fig. A.5: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the simple CNN Model. As shown, this model performed well, in particular, in scenario 1.

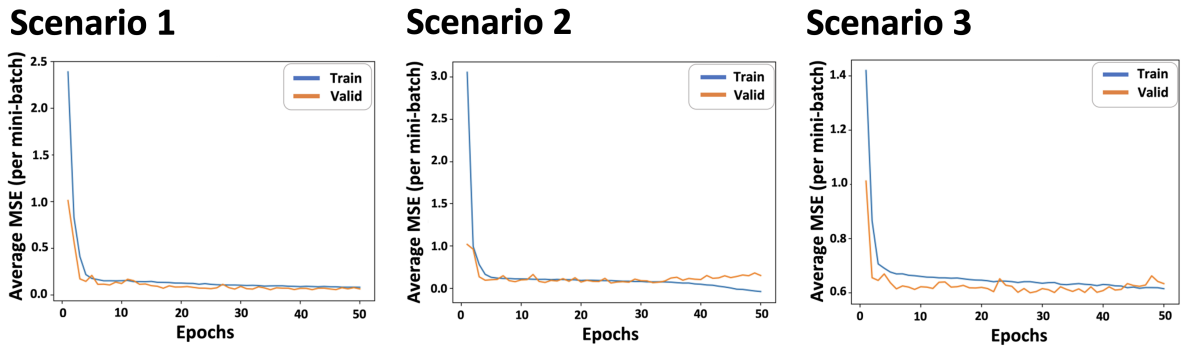

Fig. A.6: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the simple CNN Model. In all scenarios, there is a very small gap between the training and validation error.

A.3 Fully Connected Model Results

- **Sim-5-sec Data for scenarios 1-3:**

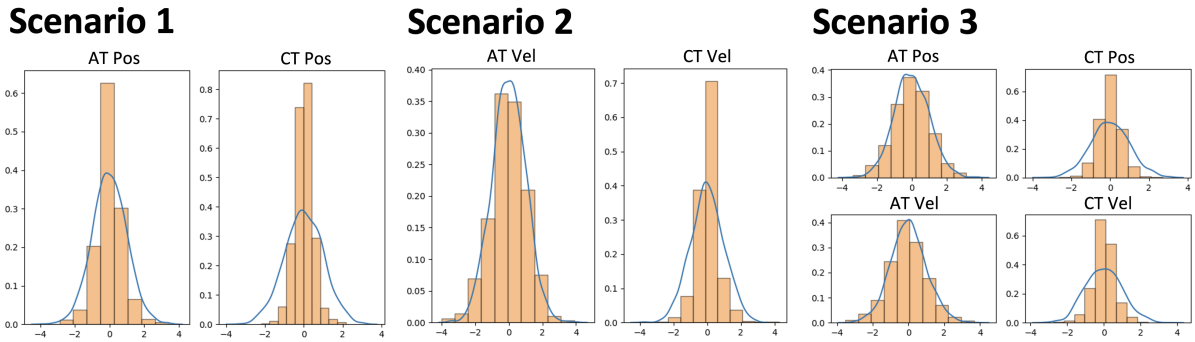

Fig. A.7: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the Fully Connected Model. As shown, this model performed well, in particular, in scenario 1.

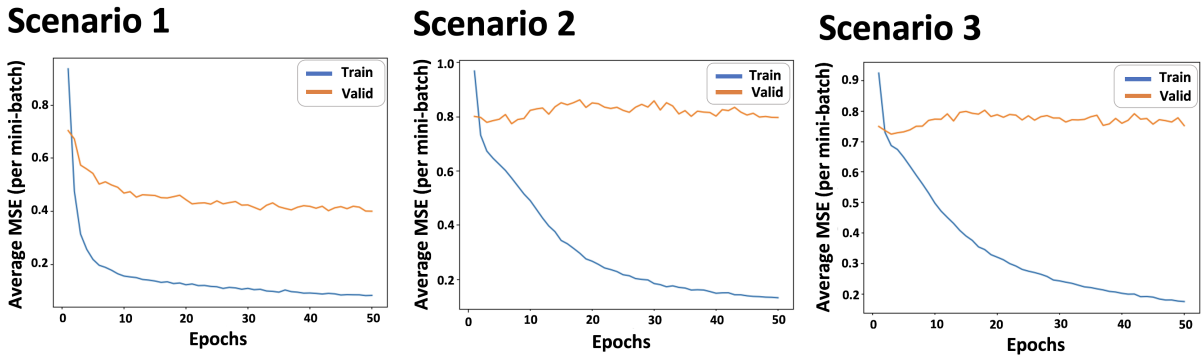

Fig. A.8: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the Fully Connected Model. There is a big gap between the training and validation error for scenario 2 and 3. The gap is smaller in scenario 1.

- **Real-2-sec Data for scenarios 1-3:**

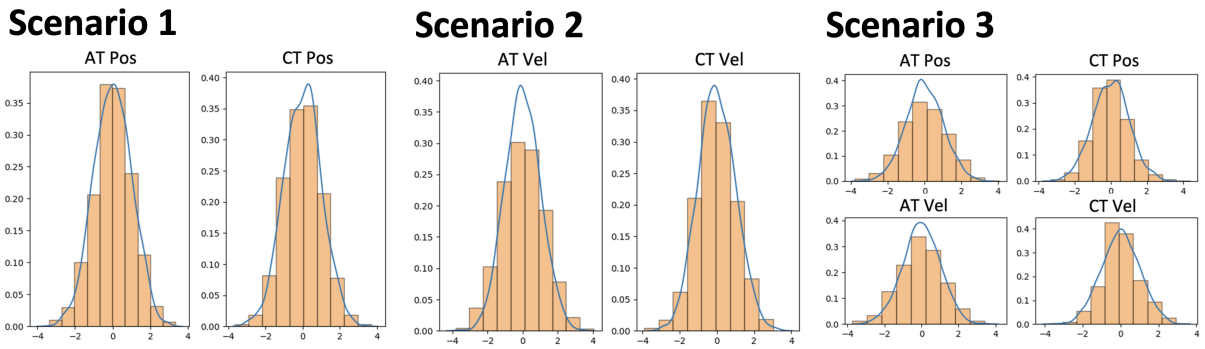

Fig. A.9: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the Fully Connected Model. As shown, this model didn't perform well in any of the scenarios.

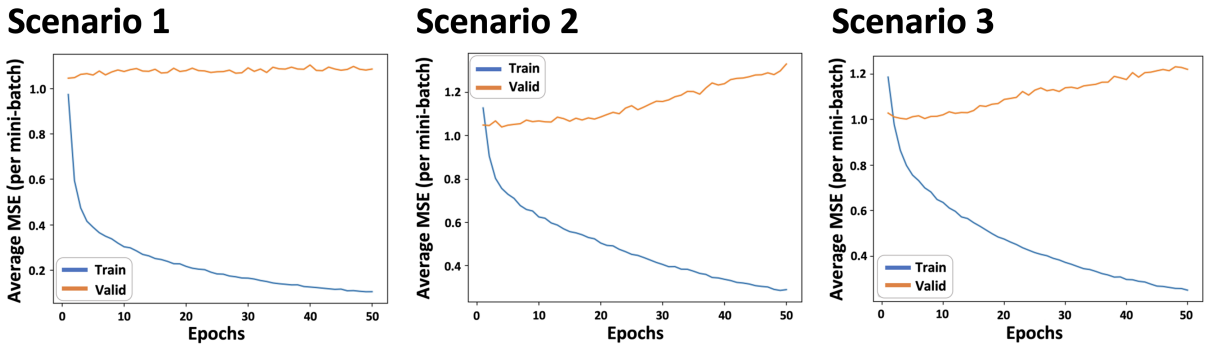

Fig. A.10: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the Fully Connected Model. In all scenarios, there is a big gap between the training and validation error suggesting overfitting.

A.4 ResNet Models

ResNet-34 Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

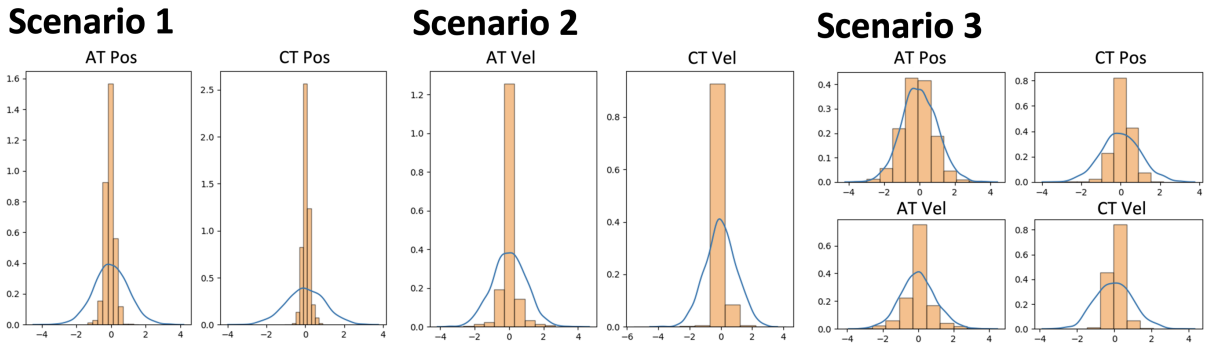

Fig. A.11: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the ResNet-34 Model. As shown, this model performed well in all the scenarios.

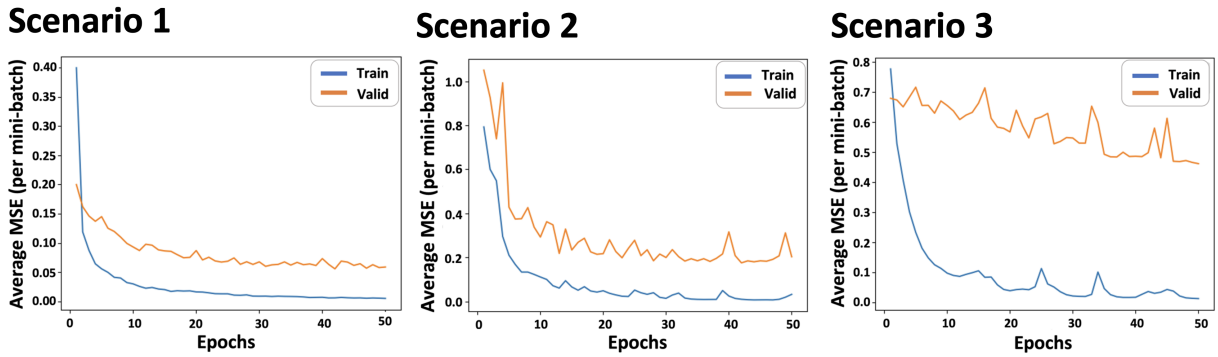

Fig. A.12: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the ResNet-34 Model. There is a small gap between the training and validation error in scenarios 1 and 2 but a big gap in scenario 3.

- **Real-2-sec Data for scenarios 1-3:**

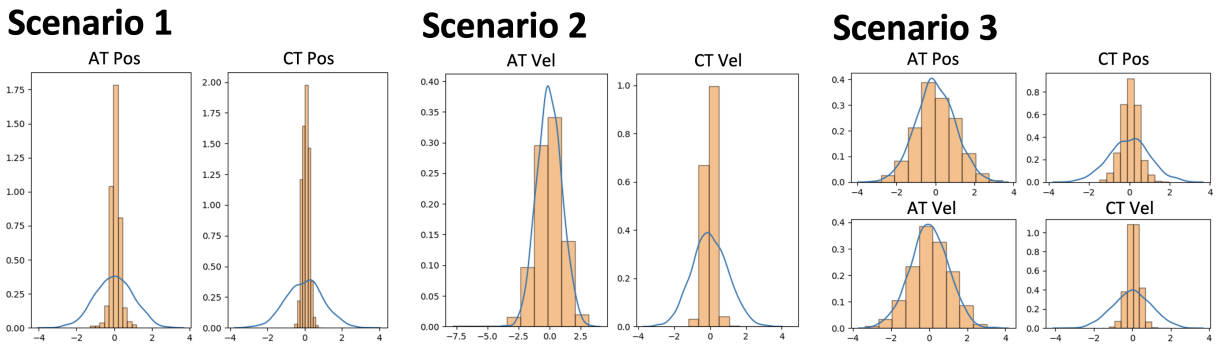

Fig. A.13: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the ResNet-34 Model. This model performs well, in particular, in scenario 1.

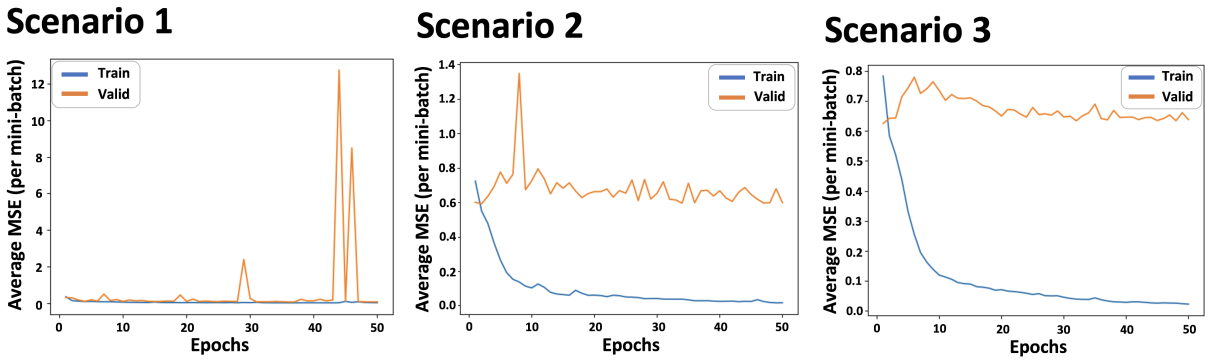

Fig. A.14: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the ResNet-34 Model. There is a small gap between the training and validation error in scenario 1 but the gap increases in scenario 2 and 3.

ResNet-50 Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

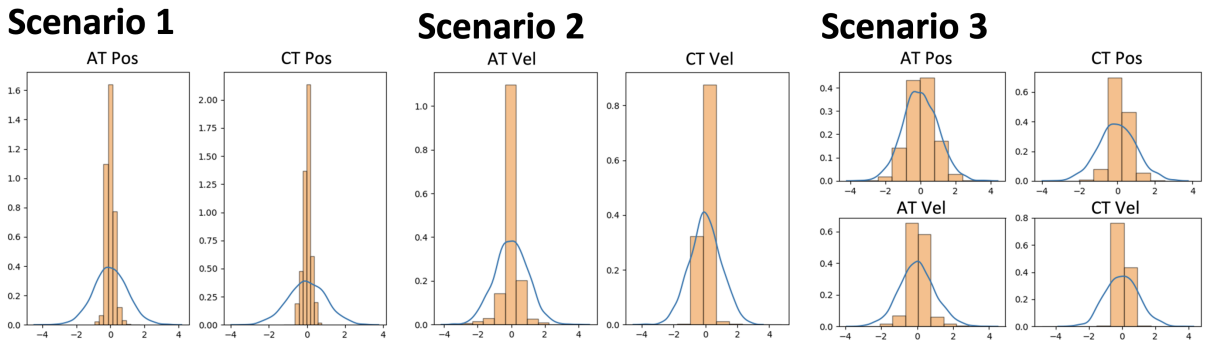

Fig. A.15: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the ResNet-50 Model. As shown, this model performed well in all the scenarios.

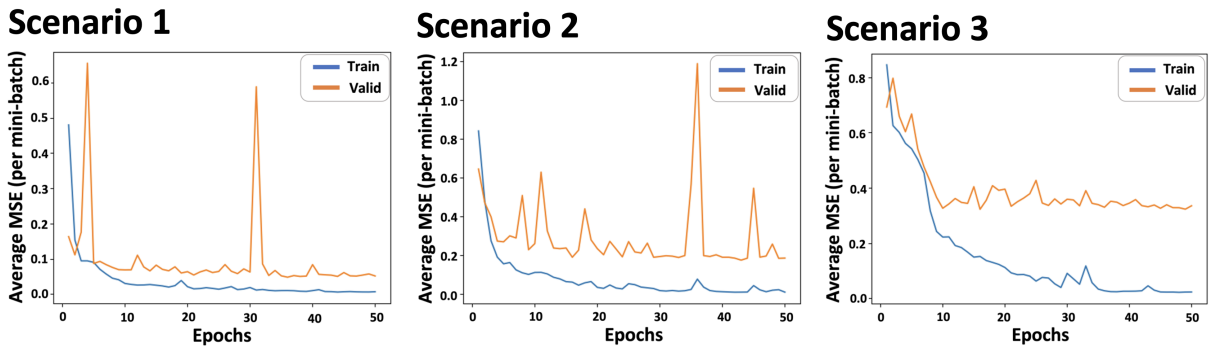

Fig. A.16: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the ResNet-50 Model. There is a small gap between the training and validation error in scenarios 1 and 2.

- **Real-2-sec Data for scenarios 1-3:**

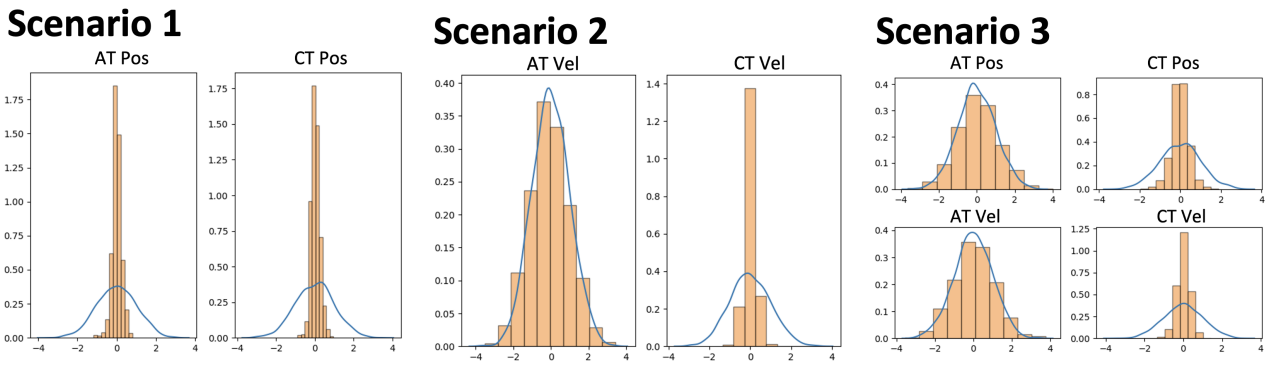

Fig. A.17: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the ResNet-50 Model. The model performs well in scenario 1.

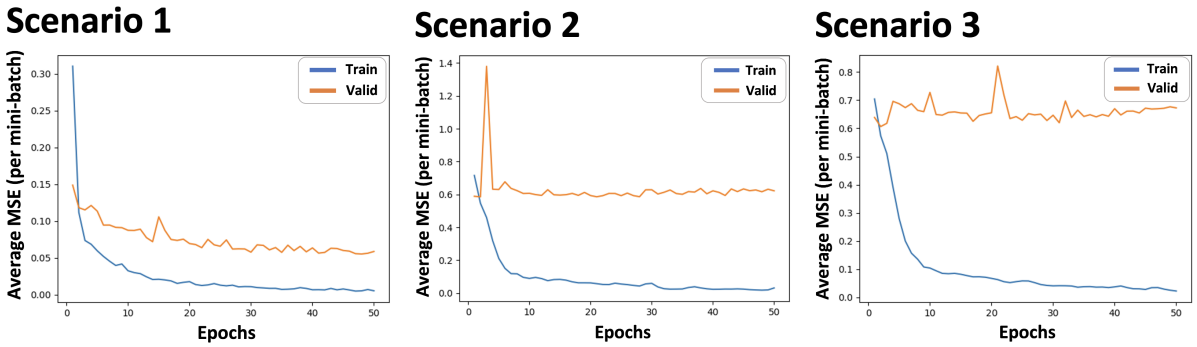

Fig. A.18: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the ResNet-50 Model. There is a bigl gap between the training and validation error in scenario 3.

ResNet-101 Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

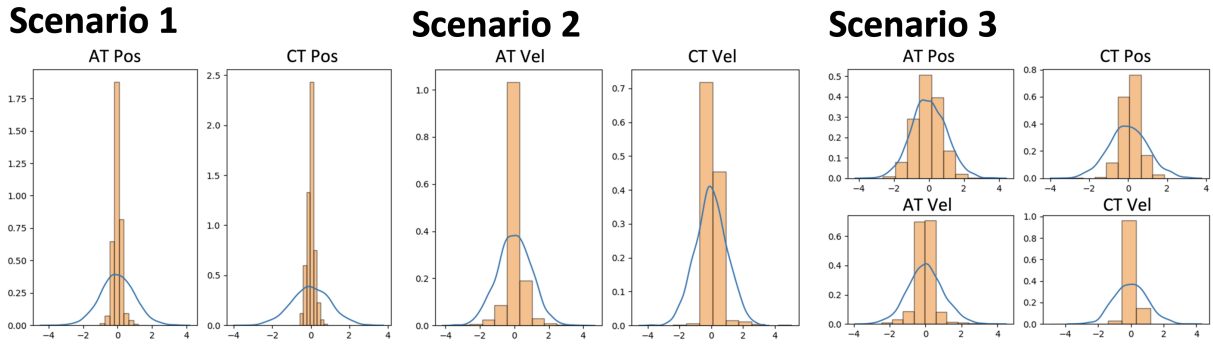

Fig. A.19: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the ResNet-101 Model. As shown, this model performed well in all the scenarios.

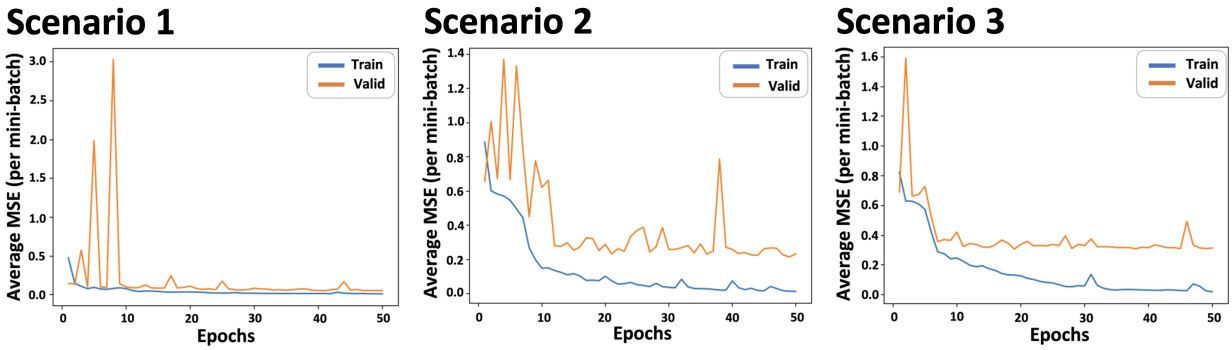

Fig. A.20: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the ResNet-101 Model. There is a very small gap between the training and validation error in scenarios 1-3.

- **Real-2-sec Data for scenarios 1-3:**

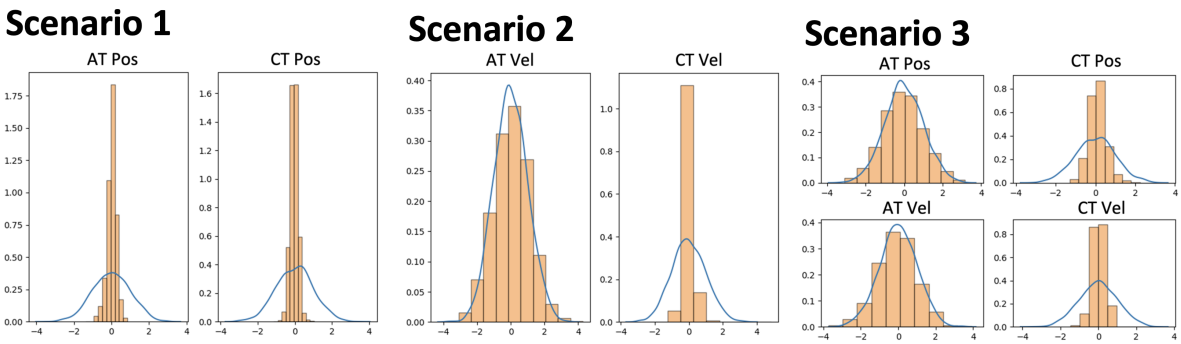

Fig. A.21: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the ResNet-101 Model. This model performs well in scenario 1.

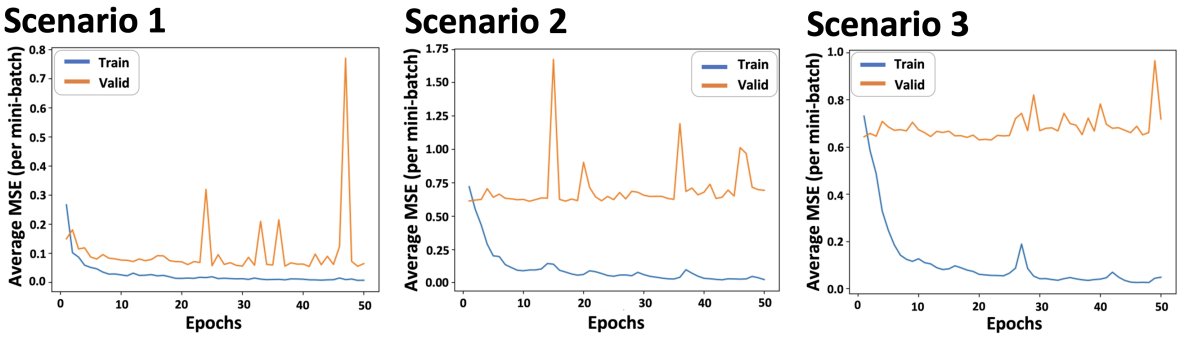

Fig. A.22: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the ResNet-101 Model. There is a small gap between the training and validation error in scenario 1 but not in scenario 2 and 3.

ResNet-152 Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

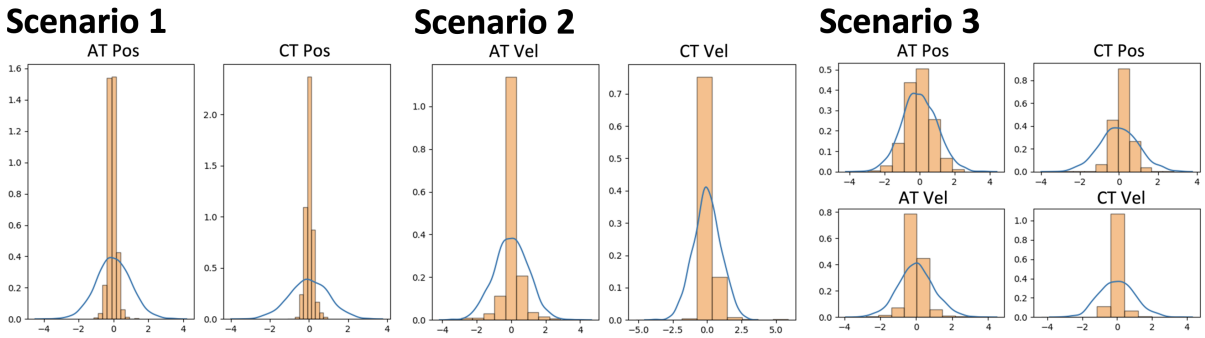

Fig. A.23: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the ResNet-152 Model. As shown, this model performed well in all the scenarios.

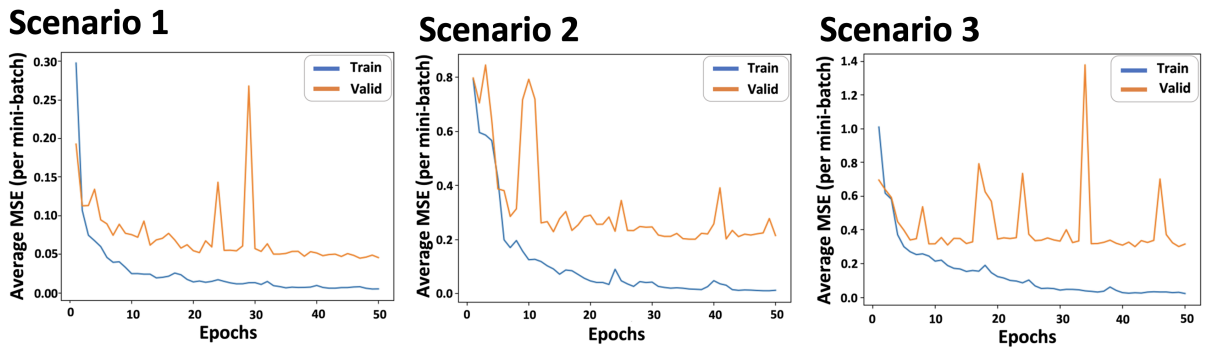

Fig. A.24: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the ResNet-152 Model. There is a small gap between the training and validation error in scenarios 1-3.

- **Real-2-sec Data for scenarios 1-3:**

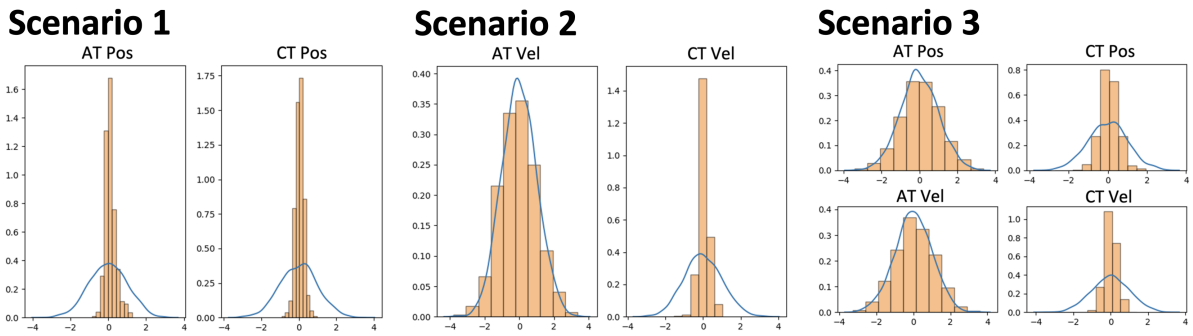

Fig. A.25: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the ResNet-152 Model. This model performed well in scenario 1.

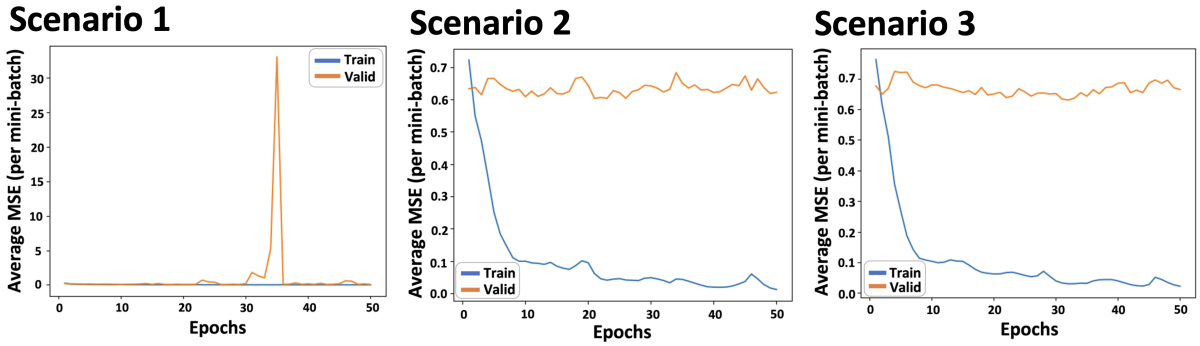

Fig. A.26: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the ResNet-152 Model. There is a big gap between the training and validation error in scenarios 2 and 3.

Wide-ResNet-50_2 Model Additional Results

- **Sim-5-sec Data for scenarios 1-6:**

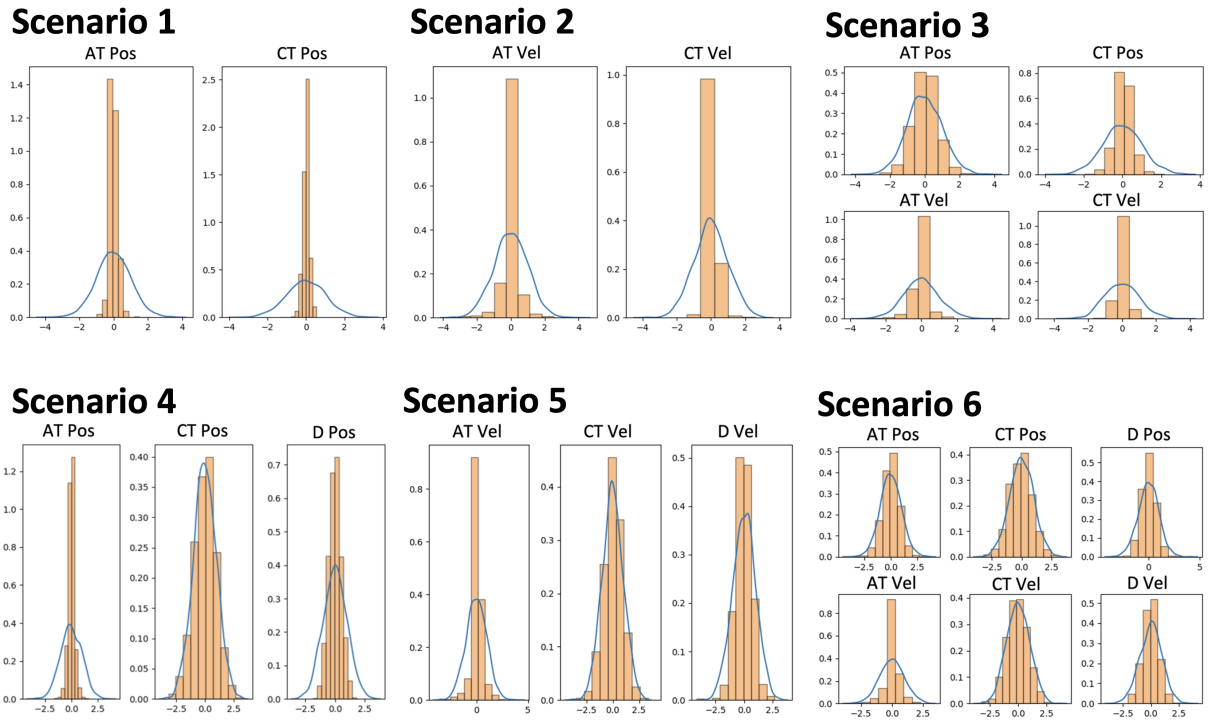

Fig. A.27: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-6 using the Wide-ResNet-50_2 Model. As shown, this model performed well in all the scenarios.

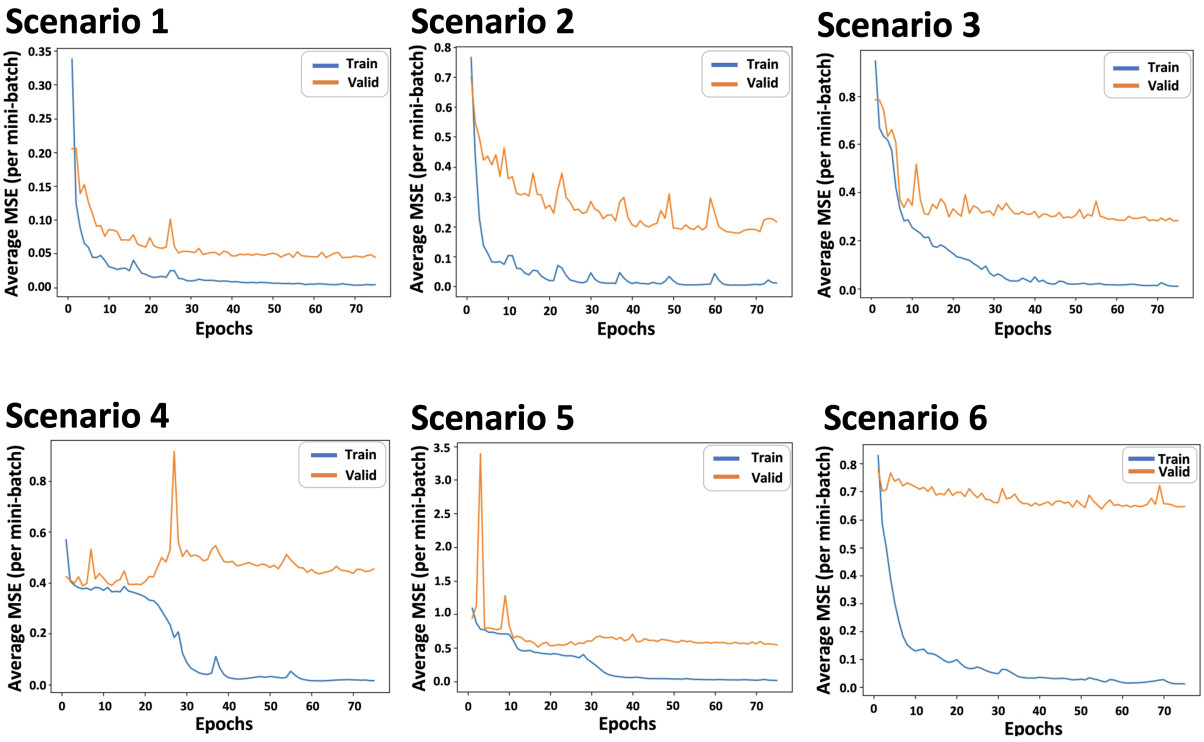

Fig. A.28: Training and validation MSE as a function of training epoch for scenario 1-6 for the sim-5-sec dataset using the Wide-ResNet-50_2 Model. There is a gap between the training and validation error in all the scenarios, in particular in scenario 6.

- **Real-2-sec Data for scenarios 1-6:**

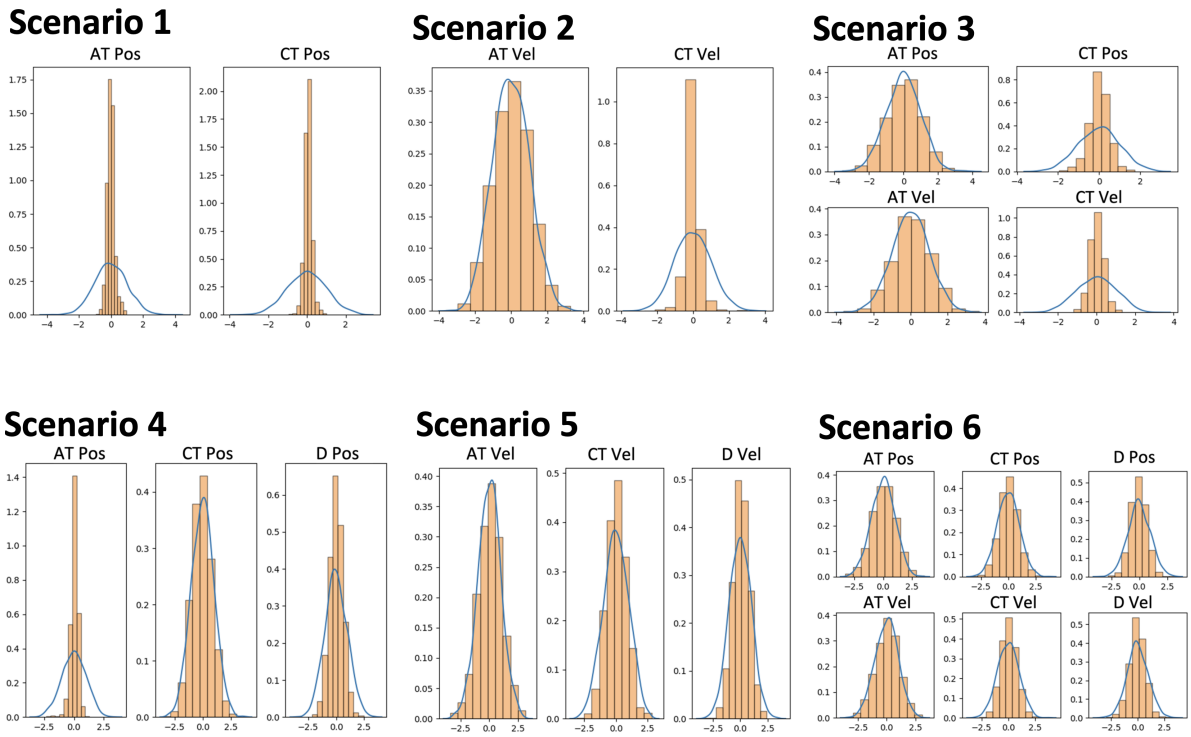

Fig. A.29: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-6 using the Wide-ResNet-50_2 Model. The model performs well in scenario 1 ($MSE < 1$).

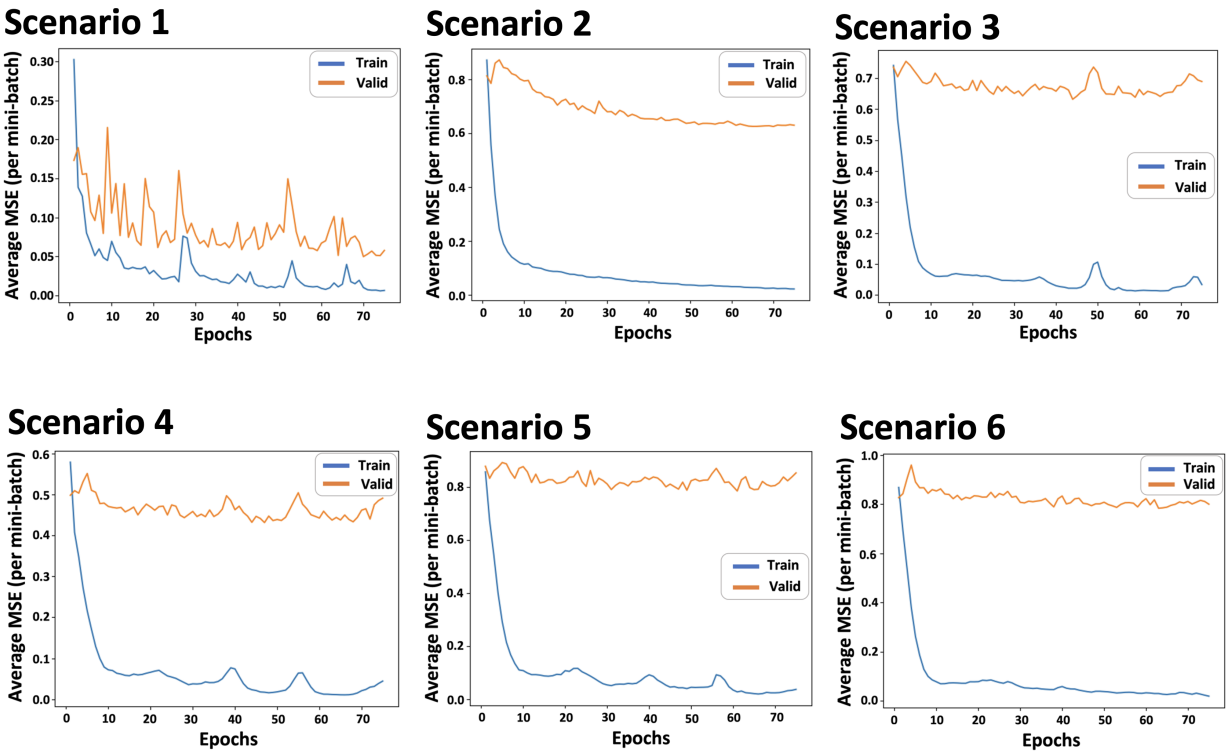

Fig. A.30: Training and validation MSE as a function of training epoch for scenario 1-6 for the real-2-sec dataset using the Wide-ResNet-50_2 Model. There is a big gap between the training and validation error in most of the scenarios.

Wide-ResNet-101_2 Model Additional Results

- **Sim-5-sec Data for scenarios 1-3:**

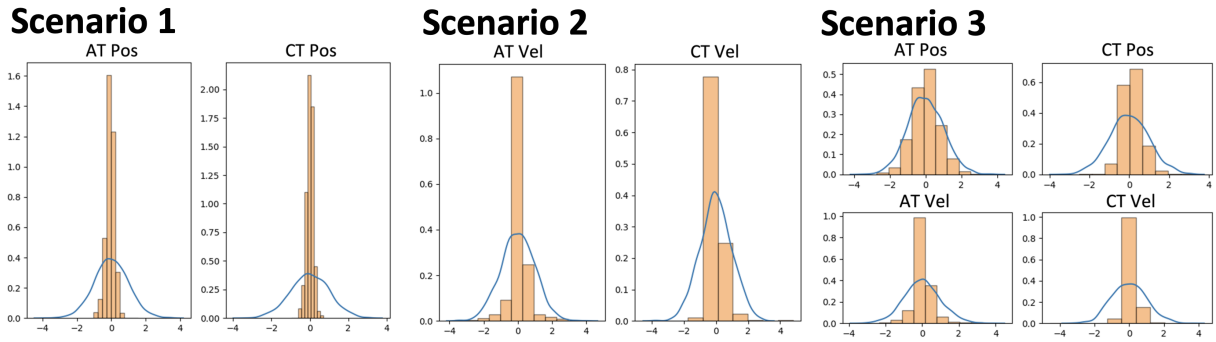

Fig. A.31: Distribution of Error States before (blue line) and after (histogram) estimation for the sim-5-sec dataset for scenario 1-3 using the Wide-ResNet-101_2 Model. As shown, this model performed well in all the scenarios.

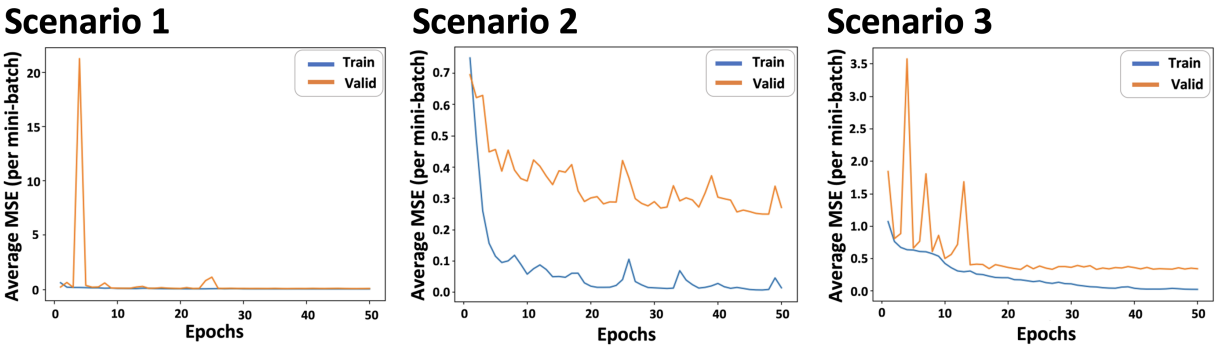

Fig. A.32: Training and validation MSE as a function of training epoch for scenario 1-3 for the sim-5-sec dataset using the Wide-ResNet-101_2 Model. There is a very small gap between the training and validation error in scenarios 1 and 3.

- **Real-2-sec Data for scenarios 1-3:**

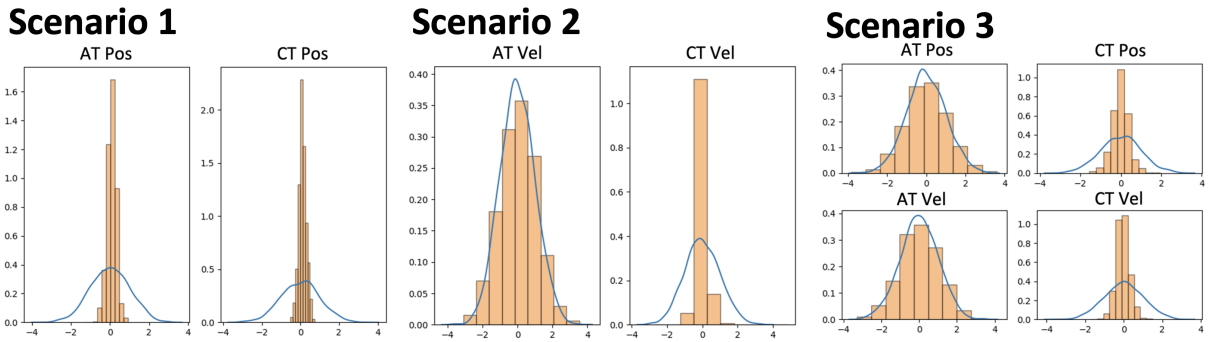

Fig. A.33: Distribution of Error States before (blue line) and after (histogram) estimation for the real-2-sec dataset for scenario 1-3 using the Wide-ResNet-101_2 Model. This model performs well in scenario 1.

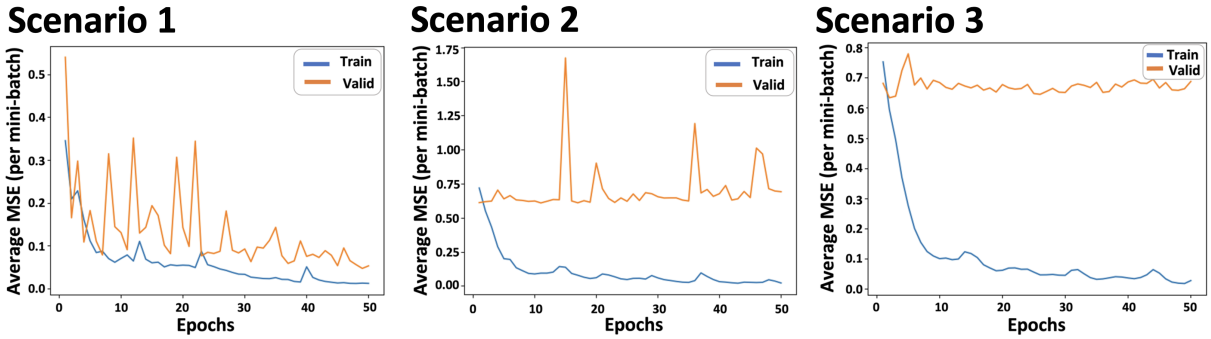

Fig. A.34: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-2-sec dataset using the Wide-ResNet-101_2 Model. There is a big gap between the training and validation error in scenario 3.

APPENDIX B

Additional Results from the implementation of the Wide ResNet 50_2 Model with different aperture lengths

B.1 Implementing the Wide ResNet 50_2 Model on the real-6-sec data for scenarios 1-3

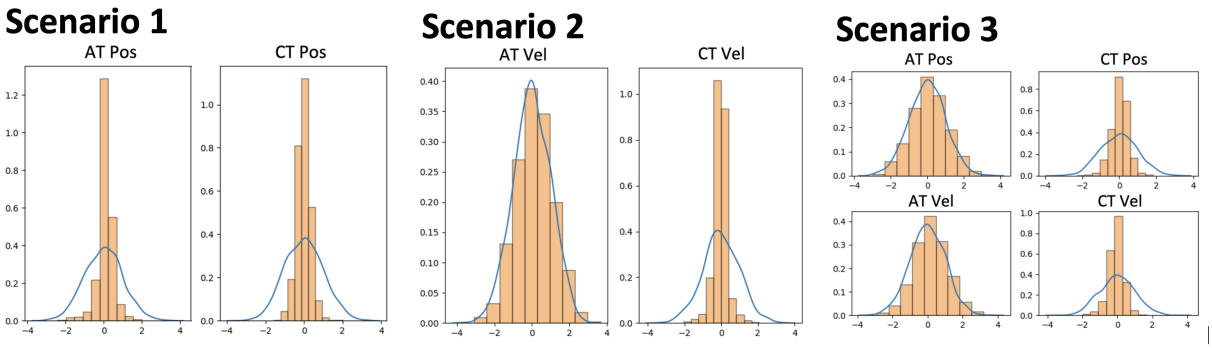

Fig. B.1: Distribution of Error States before (blue line) and after (histogram) estimation for the real-6-sec dataset for scenario 1-3 using the Wide-ResNet-50_2 Model. This model performs well in all scenario 1.

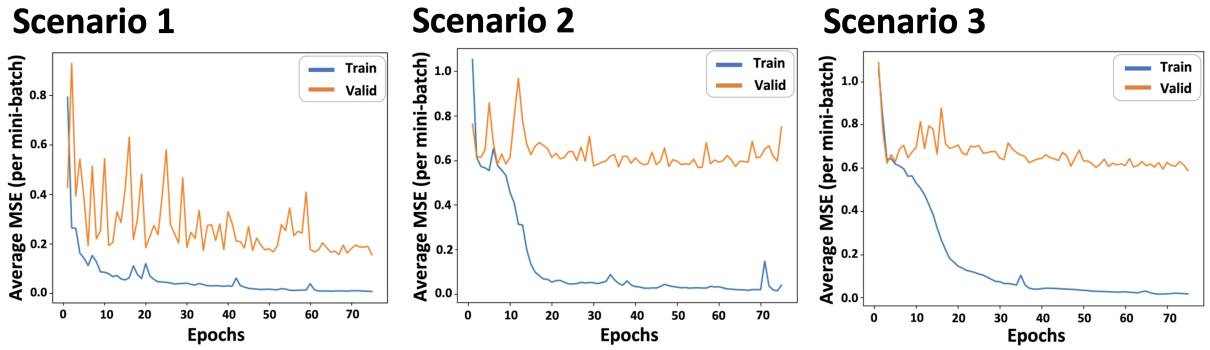

Fig. B.2: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-6-sec dataset using the Wide-ResNet-50_2 Model. There is a big gap between the training and validation error in scenario 2 and 3, suggesting overfitting.

B.2 Implementing the Wide ResNet 50_2 Model on the real-10-sec data for scenarios 1-6

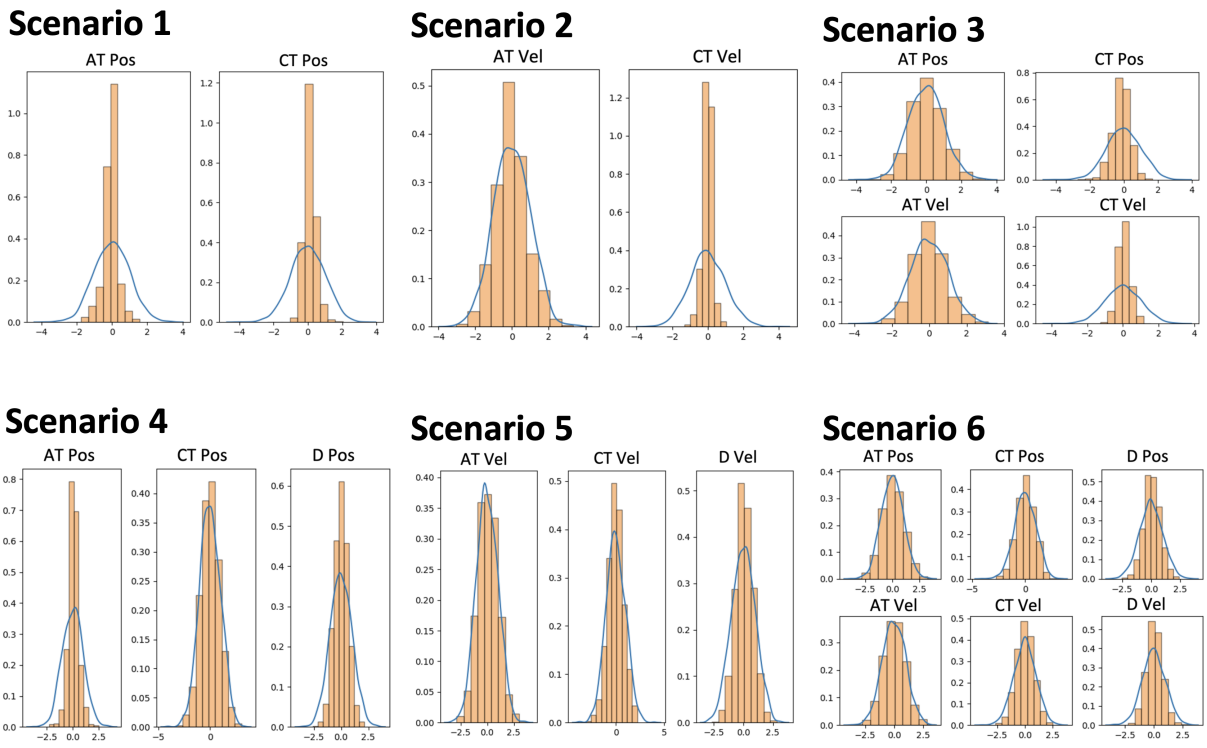

Fig. B.3: Distribution of Error States before (blue line) and after (histogram) estimation for the real-10-sec dataset for scenario 1-6 using the Wide-ResNet-50_2 Model. This model performs very well in all scenarios.

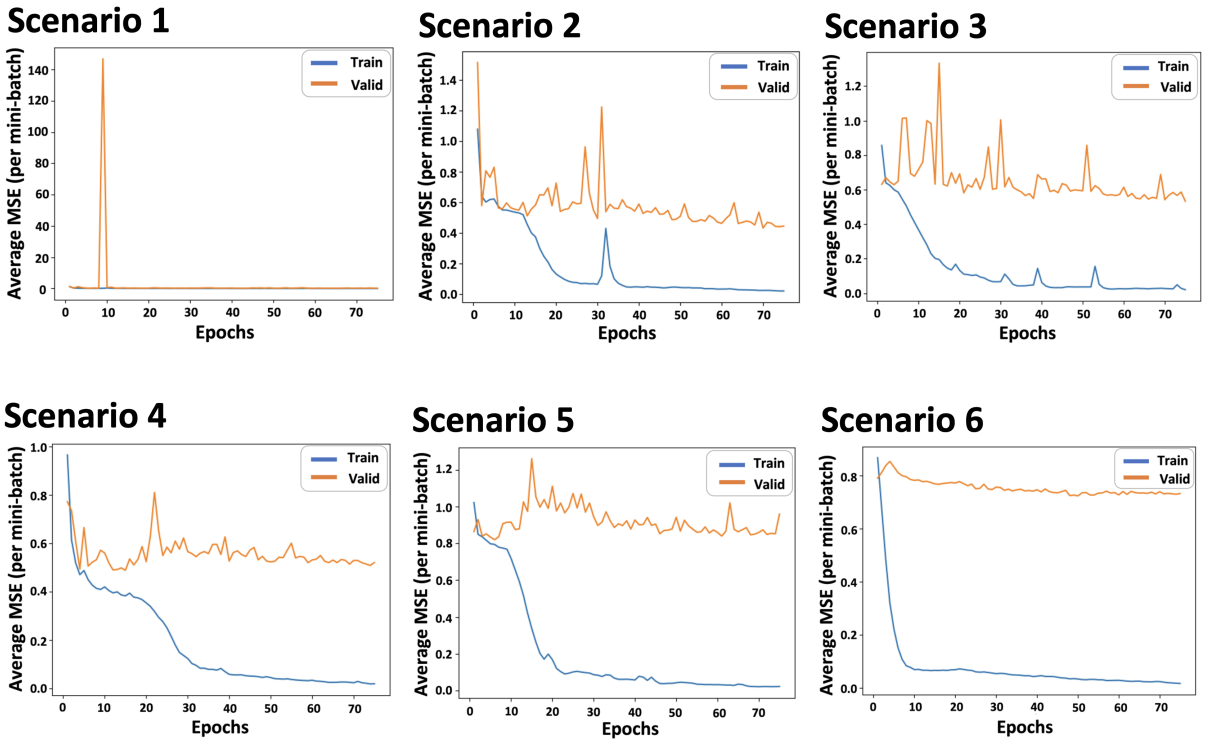

Fig. B.4: Training and validation MSE as a function of training epoch for scenario 1-6 for the real-10-sec dataset using the Wide-ResNet-50_2 Model. There is a big gap between the training and validation error in most of the scenarios, in particular in scenario 5 and 6.

B.3 Implementing the Wide ResNet 50_2 Model on the real-15-sec data for scenarios 1-3

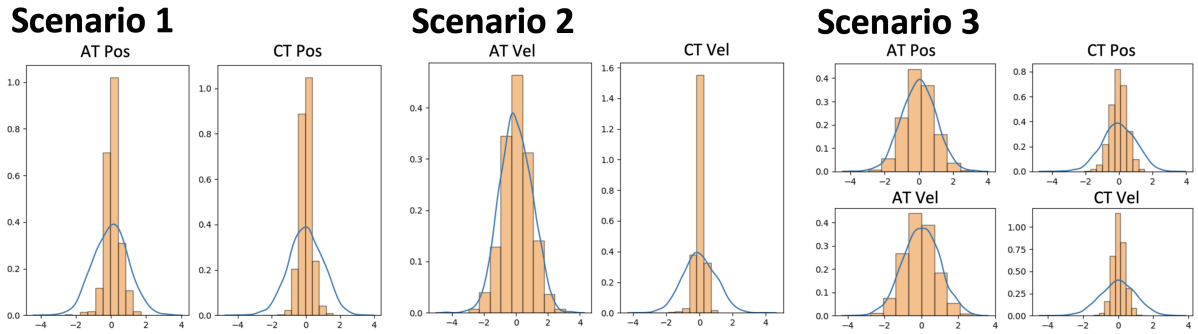

Fig. B.5: Distribution of Error States before (blue line) and after (histogram) estimation for the real-15-sec dataset for scenario 1-3 using the Wide-ResNet-50_2 Model. This model performs very well in all scenarios.

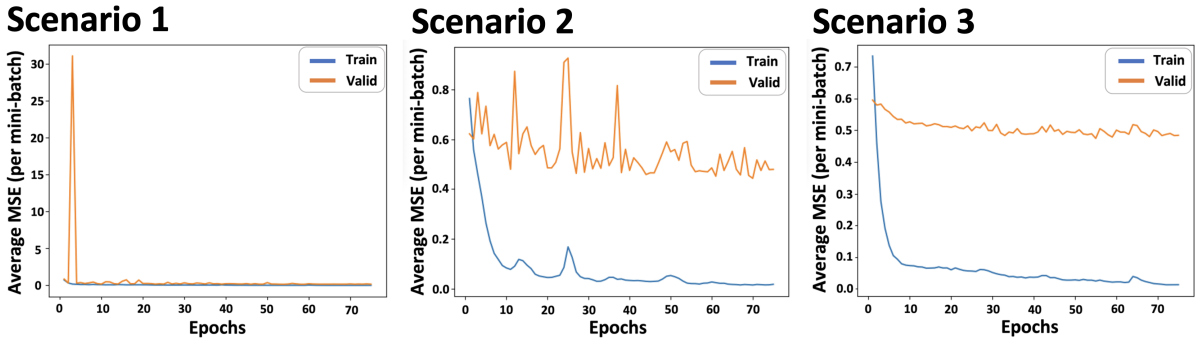

Fig. B.6: Training and validation MSE as a function of training epoch for scenario 1-3 for the real-15-sec dataset using the Wide-ResNet-50_2 Model. There is a big gap between the training and validation error in scenario 2 and 3.

APPENDIX C

Using Transfer Learning

Results obtained pretraining the sim-5-sec data and training the real-2-sec or real-10sec datset for scenario 2. AT Velocity and CT Velocity are the only errors present in this scenario.

C.1 Transfer Learning using the real-2-sec dataset:

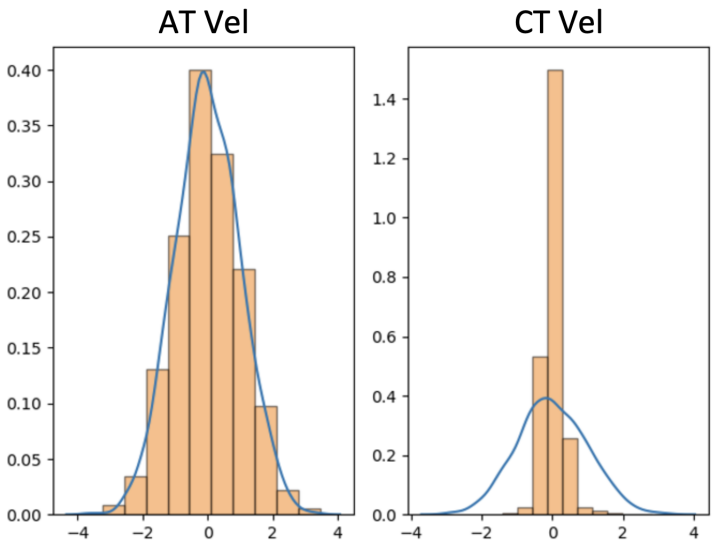

Fig. C.1: Distribution of Error States before (blue line) and after (histogram) estimation for scenario 2 using transfer learning using the real-2-sec dataset. This approach performed well for the CT Velocity error state.

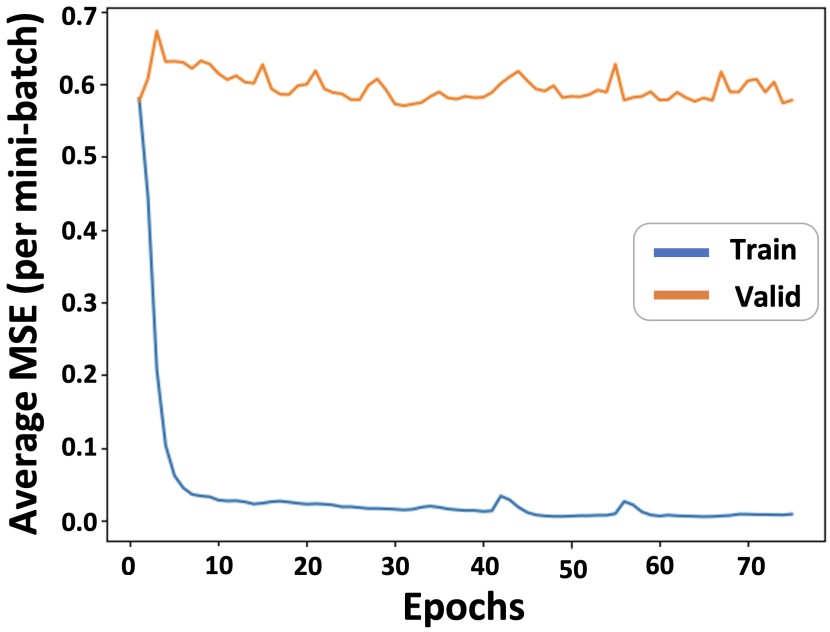

Fig. C.2: Training and validation MSE as a function of training epoch for scenario 2 using transfer learning using the real-2-sec dataset. There is a big gap between the training and validation error, suggesting overfitting.

C.2 Transfer Learning using the real-10-sec dataset:

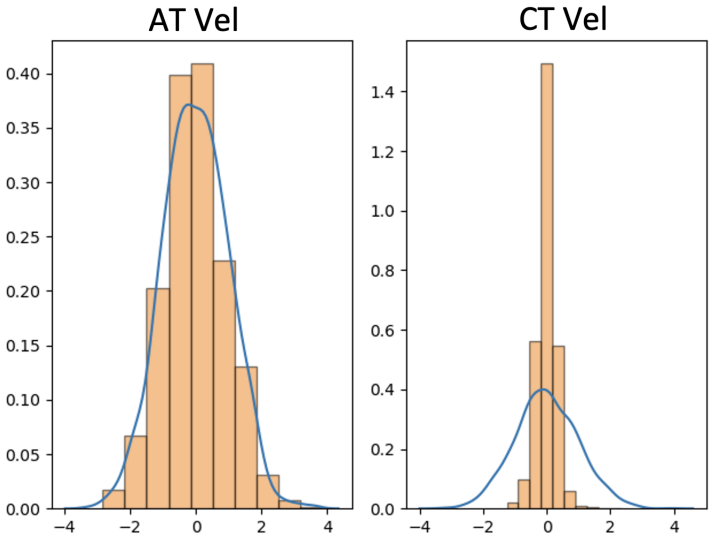

Fig. C.3: Distribution of Error States before (blue line) and after (histogram) estimation for scenario 2 using transfer learning using the real-10-sec dataset. Increasing to a 10-second aperture length helped a little bit in the performance of the AT Velocity error state.

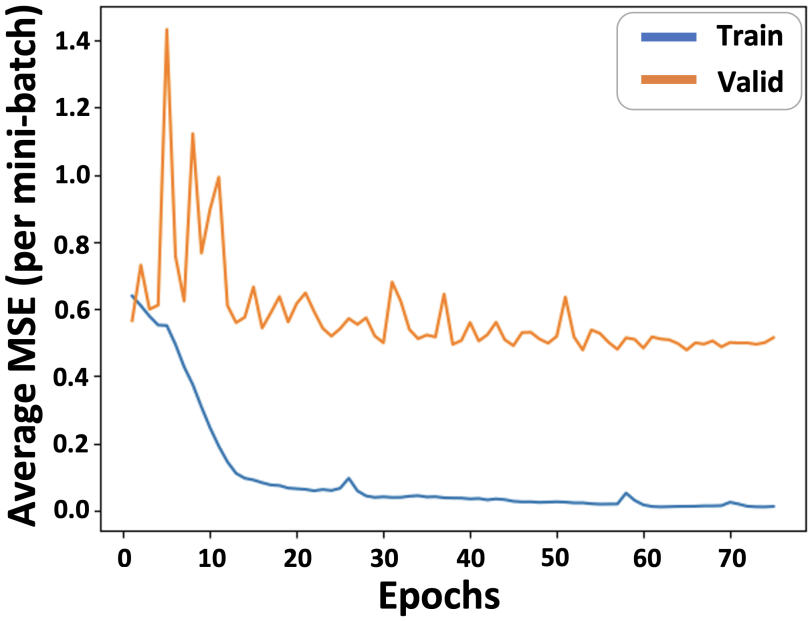

Fig. C.4: Training and validation MSE as a function of training epoch for scenario 2 using transfer learning using the real-10-sec dataset. The gap between the training and validation error is smaller than in Figure C.2.