

Inferring Inertial Navigation Errors from SAR Image Distortions Using a Convolutional Neural Network

Teresa White¹ | Jesse Wheeler² | Colton Lindstrom³ | Brennan Bean¹ |
Scott Jenkins⁴ | Randall Christensen⁵ | Kevin R. Moon*¹

¹ Dept. Mathematics & Statistics, Utah State University, Utah, USA

² Dept. Statistics, University of Michigan, Michigan, USA

³ Dept. Electrical & Computer Engineering, Utah State University, Utah, USA

⁴ Sandia National Laboratories, New Mexico, USA

⁵ Blue Origin, Washington, USA

Correspondence

*Kevin R. Moon

Department of Mathematics and Statistics,

Utah State University, Logan,

UT 84322, USA

Email: kevin.moon@usu.edu

Abstract

Unmanned aerial vehicles often rely on the Global Positioning System (GPS) for navigation. GPS signals, however, are very low in power and can be easily jammed or otherwise disrupted. This paper presents a method for estimating the navigation errors present at the beginning of a GPS-denied period using data from a synthetic aperture radar (SAR) system. These errors are estimated by comparing an online-generated SAR image with a reference image obtained *a priori*. The distortions relative to the reference image are exploited by a convolutional neural network to learn the initial navigation 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.

Keywords

CNN, GPS-denied navigation, SAR

1 | INTRODUCTION

Unmanned aerial vehicles (UAVs) are used in 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, the Global Positioning System (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.

GPS signal vulnerability to many forms of interference has motivated research in alternative approaches to localization (Raquet & Martin, 2008; Veth, 2011). 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 to exploit artifacts in SAR imagery to determine navigation errors, either relative to the beginning of the synthetic aperture (Quist & Beard, 2016; Quist et al., 2016; Samczynski,

2012; Samczynski & Kulpa, 2010; Scannapieco et al., 2018) or in a global/absolute reference frame (Bergman et al., 1999; Greco et al., 2012; Hostetler & Andreas, 1983; Kim et al., 2018; Nitti et al., 2015; Nordlund & Gustafsson, 2009; Sjanic & Gustafsson, 2010).

Back-projection-based SAR imaging is a technique for producing radar images during arbitrary flight trajectories (Carrara et al., 1995; Ulaby & Long, 2014; Zaugg & Long, 2015). SAR images are created by sending a series of radar pulses during a portion of the vehicle's trajectory. 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 (CT) directions (Christensen et al., 2019).

In a typical SAR application, an inertial navigation system uses GPS measurements to generate the best estimate of the aircraft trajectory and to 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 that must be solved for to nine initial errors.

There is a strong precedent for applying machine-learning techniques to SAR imagery. The most common application is automatic target recognition (Berisha et al., 2007; Chen & Wang, 2014; Chen et al., 2016; Krishnapuram et al., 2005; Wilmanski et al., 2016; Zhao & Principe, 2001). Other applications include detecting changes in radar images of the same landscape over time (Gong et al., 2016) or auto-focusing blurry radar images (Mason et al., 2017) using deep learning. These latter two applications mainly focus on image reconstruction. In particular, auto-focusing primarily focuses on horizontal velocity errors and is thus not suited for our problem, as it neglects position errors. Hence, despite a wide range of target recognition problems, there is no precedent in the literature for using machine learning to predict initial navigation errors in a regression-based framework, further motivating the need for our analysis.

In another study in the auto-focusing literature, focused SAR images were produced by correcting the phase errors to focus three strong targets (Werness et al., 1990). However, it is currently unclear whether knowing the phase errors is sufficient to infer the entire navigation state. In our approach, we observe that velocity errors in the AT direction induce blurs in the SAR image, which can be interpreted as Doppler or phase errors. While auto-focusing methods typically correct these errors, our goal is not to correct these errors or to produce focused images. Such corrections may result in additional shifts within the image that may make error inference more difficult for the neural network. Instead, the neural network processes uncorrected images and then infers errors in the navigation state.

Our problem shares some similarities with the task of image registration (Hero et al., 2002; Pickup et al., 2007; Tipping & Bishop, 2003; Wyawahare et al., 2009) in that both a reference image and an input image are 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.

We show that neural networks can be used to develop a model of the distortion dynamics that takes as input the distorted image and a reference image and

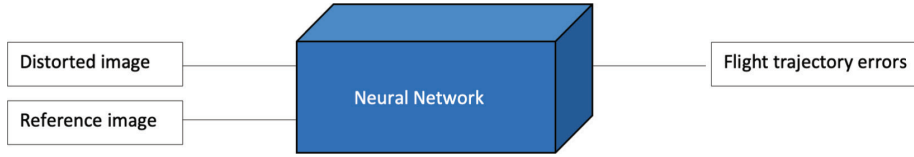


FIGURE 1 Neural network approach to GPS-denied navigation using SAR images
The network takes as input the distorted SAR image and a reference image and predicts the flight trajectory errors. The predicted errors can then be used to obtain the true flight trajectory.

predicts the flight trajectory errors, as shown in Figure 1. Six different scenarios were studied with different combinations of active navigation errors for each data set explored. We applied simple convolutional neural network (CNN) 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.

Section 2 begins by providing the necessary background regarding navigation errors in BPA-SAR images. Section 3 describes the different sets of SAR image data generated for this study and the data preparation for our analysis. Section 4 provides the convolution operation and explores different CNN methods for estimating the initial navigation errors on both simulated and real data. Section 5 presents the performance of the Wide ResNet 50_2 model and provides an analysis of the results. Section 6 discusses future work and provides a conclusion of our study.

2 | NAVIGATION BACKGROUND

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

$$\begin{bmatrix} \dot{\mathbf{p}}^n \\ \dot{\mathbf{v}}^n \\ \dot{q}_b^n \end{bmatrix} = \begin{bmatrix} \mathbf{v}^n \\ T_b^n \mathbf{v}^b + \mathbf{g}^n \\ \frac{1}{2} q_b^n \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b \end{bmatrix} \end{bmatrix} \quad (1)$$

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 aligned in the direction of gravity, and the y-axis determined by the right-hand rule. Thus, the n frame represents an AT, CT, down (D) 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 (EKF). Between these corrections, or in the absence thereof, the navigation errors accumulate over time. For straight and level flights, which are typical in SAR data collection, error growth is modeled to the first order:

$$\begin{bmatrix} \delta \mathbf{p}^n(t) \\ \delta \mathbf{v}^n(t) \\ \delta \boldsymbol{\theta}^n(t) \end{bmatrix} = \Phi(t, t_0) \begin{bmatrix} \delta \mathbf{p}^n(t_0) \\ \delta \mathbf{v}^n(t_0) \\ \delta \boldsymbol{\theta}^n(t_0) \end{bmatrix} \quad (2)$$

Here, $\Phi(t, t_0)$ is the state transition matrix, defined as follows:

$$\Phi(t, t_0) = \begin{bmatrix} I_{3 \times 3} & I_{3 \times 3} \Delta t & (\mathbf{v}^n \times) \frac{\Delta t^2}{2!} \\ 0_{3 \times 3} & I_{3 \times 3} & (\mathbf{v}^n) \times \Delta t \\ 0_{3 \times 3} & 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} \quad (3)$$

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

$$\mathbf{p}^n(t) = \hat{\mathbf{p}}^n(t) + \delta \mathbf{p}^n(t) \quad (4)$$

$$\mathbf{v}^n(t) = \hat{\mathbf{v}}^n(t) + \delta \mathbf{v}^n(t) \quad (5)$$

$$\mathbf{q}_b^n(t) = \begin{bmatrix} 1 \\ -\frac{1}{2} \delta \boldsymbol{\theta}^n(t) \end{bmatrix} \otimes \hat{\mathbf{q}}_b^n(t) \quad (6)$$

Thus, given an initial condition for the navigation errors, the true navigation states can be recovered by propagating the errors using Equation (2) and substituting the result into Equations (4)–(6). Therefore, the task of correcting errors in the estimated trajectory is 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 has been provided by Lindstrom et al. (2022b) and Christensen et al. (2019).

For the simple case of constant position errors, SAR image shifts with respect to an *a priori* SAR map provide 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 (Christensen et al., 2019; Lindstrom et al., 2022b). Previous sensitivity studies discovered a complex and seemingly ambiguous relationship among the nine initial navigation errors and the induced SAR image distortions, as summarized in Table 1 (Christensen et al., 2019). In practical navigation systems, however, the D components of position and velocity errors are easily controlled by a barometric altimeter. The absence of vertical errors, combined with relatively small sensitivity to typical attitude

TABLE 1

Effect of Navigation Errors on BPA-SAR Images

Shifting and blurring distortions are caused by position \mathbf{P}^n , velocity \mathbf{v}^n , and attitude quaternion \mathbf{q}_b^n errors, where n refers to the navigation frame (AT, CT, D coordinate system) and b refers to the body frame.

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

errors, results in SAR image distortions dominated by position and velocity errors in the AT and CT directions. Thus, in this paper, we focus on position and velocity errors.

While image distortions are characterized by shifts and blurs in different directions (see Figure 2), certain combinations of position and velocity 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 learned via neural networks to predict the initial navigation errors.

To calculate the extent of the errors (see Table 2), the navigation system was run with GPS updates until the state covariance matrix reached a steady state. Thus, when errors were injected into the data, the errors were calculated by comparing the steady-state covariance matrices of the GPS-available data with the covariance matrices of the GPS-denied data. The GPS-available data were used for this comparison only. These initial errors were then incorporated into the state transition matrix and propagated forward in time. Once enough time had elapsed to form an aperture, an image was formed and saved for processing in the neural network.

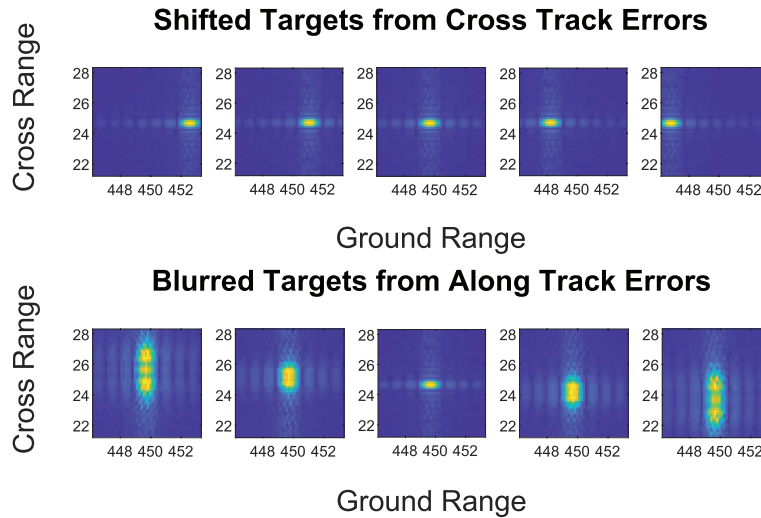


FIGURE 2 Demonstration of shifting and blurring distortions due to navigation errors: (top) shift distortions caused by errors in CT position for CT errors (from left to right) of -3 , -1.5 , 0 , 1.5 , and 3 m; (bottom) blur distortions caused by errors in AT velocity for AT errors (from left to right) of -0.4 , -0.2 , 0 , 0.2 , and 0.4 m/s

TABLE 2
Initial 3σ Values of the State Covariance Matrix
Used to Inject Errors into the Flight Trajectory

Error	3σ Initial Uncertainty
AT Position	0.54 m
CT Position	0.54 m
D Position	1.35 m
AT Velocity	0.051 m/s
CT Velocity	0.051 m/s
D Velocity	0.084 m/s

3 | SAR DATA

We demonstrate our approach on both simulated and real SAR images. Simulated data were 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 (Duersch & Long, 2015; Lindstrom et al., 2022b). We assume that the images are well-focused unless errors have been intentionally injected.

For the real data sets, radar data were obtained from flight tests of the Space Dynamics Laboratory X-band radar system (Jensen et al., 2016). A sub-centimeter-accurate, post-processed trajectory is used as truth. To generate training and test data, the flight trajectory is first distorted with initial errors. An image is then generated using the BPA according to the selected aperture length. For the simulated data, we generated data using a 5-s aperture length (sim-5-s). For the real data, we generated data using 2-s (real-2-s), 6-s (real-6-s), 10-s (real-10-s), and 15-s (real-15-s) aperture lengths. The effects of navigation errors on images differ for different aperture lengths. As the aperture length increases, the initial errors propagate for a longer duration, potentially leading to larger image distortions. This trend is more pronounced for blurring distortions, which we use to our advantage.

For both the real and simulated data sets, the operating frequency is 9.75 GHz with a bandwidth of 500 MHz. For both data sets, the pixel spacing is 30×30 cm. The AT resolution is a function of the radar wavelength, slant range, and synthetic aperture length; thus, changing the aperture length will change the actual AT resolution. Consequently, the best possible AT resolution may differ from the actual pixel size. When back-projection is used to form the images, a grid of points is defined with a user-specified size, and the raw data are projected onto this grid. For convenience in processing, the grid points in our data were held constant despite changes in synthetic aperture length.

For each of the simulated and real data sets, six different scenarios were studied, with varying numbers and combinations of active navigation errors, as shown in Table 3. These scenarios were selected to determine the performance of the neural network both with and without potential error ambiguities. For example, in scenario 1, no error ambiguity is expected, as errors in AT and CT positions result in shifted images in the AT and CT directions, respectively (see Table 1). However, adding the D position error in scenario 4 introduces potential ambiguity in the CT shifts. For each of the six scenarios, a total of 192, 185, 178, 175, and 177 unique target locations were considered for the sim-5-s, real-2-s, real-6-s, real-10-s, and real-15-s data sets, respectively. For each target, we generated 100

TABLE 3

Summary of Scenarios and Corresponding Initial Errors

An “x” indicates the consideration of a navigation error in the scenario.

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 4

Details of Data Splitting

The six scenarios of the sim-5-s, real-2-s, real-6-s, real-10-s, and real-15-s data were split into 70% training, 10% validation, and 20% testing.

Data Set	Training		Validation		Testing	
	targets	images	targets	images	targets	images
Sim-5-s	134	13500	19	1900	39	3800
Real-2-s	130	13000	18	1800	37	3700
Real-6-s	125	12500	18	1700	35	3600
Real-10-s	122	12300	17	1700	36	3500
Real-15-s	124	12300	18	1700	35	3500

distorted images with a size of 80×80 pixels, paired with the corresponding navigation errors. The data sets were then sorted by targets into training, validation, and testing sets, as shown in Table 4. We train on a set of targets and perform validation and testing on an entirely different set of targets. A network's performance on the validation data is used to select appropriate hyperparameters. After the hyperparameters are finalized, the trained network is assessed on the test data. Thus, a network's performance on the test data reflects its ability to generalize to new locations.

The brightness of each pixel of a SAR image represents the reflectivity of all targets within that pixel's region. The pixel intensities are inherently right-skewed and are 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 $y_{\alpha\beta}$ the corresponding navigation error vector. We first log-transform each pixel in each image (including the reference images) using $L = \log_{10}(X)$, 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 image mean and dividing by the image 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 navigation errors $y_{\alpha\beta}$ is measured on a different scale. Therefore, we standardize the navigation errors as $\mathbf{s} = \frac{y - \mu_y}{\sigma_y}$ to ensure equal consideration of all navigation errors during training. We estimate μ_y and σ_y from the entire data set. In practice, the standardization of navigation parameters would use a mean of 0 (assuming unbiased navigation errors on average) and a standard deviation estimated from an EKF (Kalman, 1960).

4 | NEURAL NETWORK METHODS

The parameters (i.e., weights and biases) in a neural network are learned by minimizing a cost or loss function. For our loss function, we used the average mean squared error (MSE) across all considered standardized initial errors for training and testing:

$$\text{MSE} = \frac{1}{mn} \sum_{\alpha=1}^n \sum_{\beta=1}^m (s_{\alpha\beta} - \hat{s}_{\alpha\beta})^2 \quad (7)$$

where m is the number of error states considered, n is the number of training images in the data, $s_{\alpha\beta}$ is the true standardized error of the β -th error and the α -th

image, and $\hat{s}_{\alpha\beta}$ is the corresponding error estimate determined by the neural network. We also used L2 regularization or weight decay to mitigate overfitting.

The loss function of a neural network is typically minimized by using some variant of the stochastic gradient descent (SGD), in which the gradient of the loss function with respect to the parameters is calculated via back-propagation. The Adam optimizer is an adaptive SGD variant that often has superior performance and is easier to tune (Kingma & Ba, 2015). We train our networks using AdamW, a variation on the Adam optimizer that improves the performance by decoupling the weight decay from the gradient-based update (Loshchilov & Hutter, 2017).

Once all of the errors are normalized, the MSE for any prediction can be compared against the value of 1 as a baseline: 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. This comparison serves as a benchmark for learning, where an MSE of less than 1 indicates that the neural network is learning relevant information for this task.

CNNs are a special type of neural network in which some notion of spatial structure is assumed, such as in images. CNNs consist of three common types of layers (O’Shea & Nash, 2015). The first is a convolutional layer, in which neurons are arranged in three dimensions (width, height, and depth). A convolutional layer computes the output of neurons that are connected to local regions in the input by calculating the dot product between their weights and a specific region connected to the input volume. Thus, the output of a convolutional layer consists of the convolution of a filter or kernel with the input image (see Figure 3). The kernel, which consists of a matrix of weights, is used to extract features from the input image. The weights in each kernel are learned during training by choosing values that minimize the loss function. If the input image has three channels, then a kernel with size n has $n \times n \times 3$ weights. In each convolutional layer, multiple kernels are learned, allowing the network to extract multiple features.

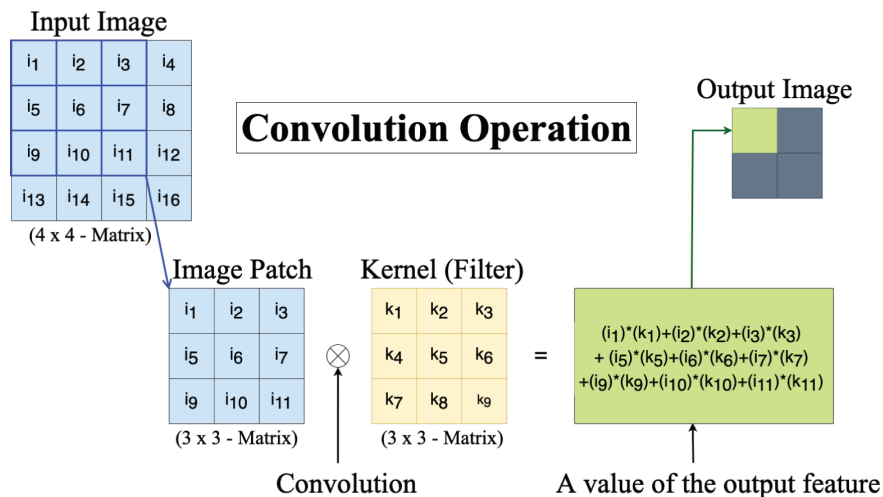


FIGURE 3 Example of a convolution operation

In this operation, the kernel slides across the *input image*, performing an element-wise multiplication with the input element (*image patch*) that overlaps with the *kernel*. The results are summed into a single *output* pixel in the current location (Dumoulin & Visin, 2016). This example shows a 3 × 3 kernel (filter) applied to the top-left corner of a 4 × 4 input image. In a CNN, the weights in each kernel are learned during training. This process is repeated by shifting the kernel by a fixed number of pixels to obtain the full output image.

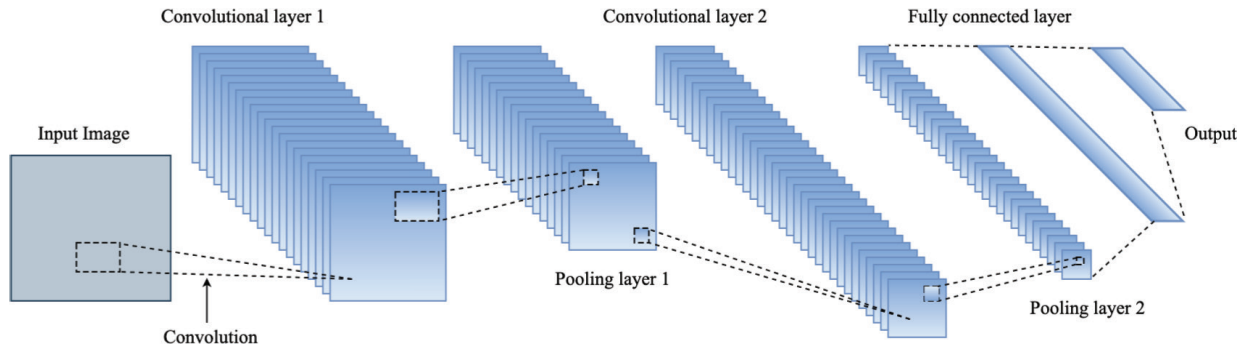


FIGURE 4 Overview of a simple CNN architecture
 This example illustrates the source image, followed by the most common building blocks of a CNN, such as convolutional layers, pooling layers, fully connected layers, and the output layer.

Convolutional layers are typically followed by a pooling layer, which downsamples the output of a convolutional layer to reduce the number of parameters. The most common pooling layer is a max-pooling layer, which returns the maximum value inside a subregion of the image. CNNs typically consist of multiple convolutional and pooling layers and then end with one or more fully connected layers, as shown in Figure 4.

The use of the kernel via the convolution operation leads to shared weights that can learn features that may be present in any part of the image. The weights are shared, which reduces the total number of weights when compared to a fully connected neural network, in which each pixel of the image is treated as a node in the neuron. This approach leads to faster training times, allowing for improved learning.

In particular, deep CNNs with many layers tend to perform well on image recognition tasks (He et al., 2016; Zagoruyko & Komodakis, 2016), as they can learn higher levels of abstraction in the data (Nielsen, 2015). Deep networks generally have a greater capacity to learn complex functions compared with shallow networks. However, deep networks are more difficult to train because of vanishing gradient problems and other issues, requiring longer training times and very large data sets to be successful. To overcome this obstacle, many researchers use transfer learning. Transfer learning is a technique that can help to improve learning in a new task through the transfer of knowledge from another related task (Pan & Yang, 2009). The goal of using transfer learning is to fully learn the target task given the transferred knowledge instead of learning from scratch. This approach can often speed up the training and improve the performance of the model, especially when the sample size in the target task is small (Bar et al., 2015; Tajbakhsh et al., 2016).

A common approach to transfer learning with neural networks is to begin the learning process with a network that has been pretrained on a large data set. In this case, most of the architecture of the network for the target task is predefined based on the pretrained network. The layers close to the inputs and outputs are typically modified to conform to the target task. The weights in each of the predefined layers are then typically initialized using the weights learned by the pretrained network. Weights for modified or added layers are randomly initialized; then, all of the weights in the network are updated by training on data from the target task.

For our SAR navigation problem, we apply transfer learning by using existing deep CNN architectures that have been pretrained on large data sets (e.g., ImageNet (Deng et al., 2009)). Specifically, we considered multiple ResNet (He et al., 2016)

and Wide ResNet (Zagoruyko & Komodakis, 2016) architectures pretrained on large image data sets. The ResNet architectures are CNNs that use a series of layers referred to as the “residual block.” This block allows networks to be much deeper while simultaneously mitigating the issue of a vanishing gradient, allowing us to obtain the capacity of deeper networks (He et al., 2016). For this reason, ResNet models have performed well on multiple image recognition tasks when used for transfer learning (Du et al., 2018; Liu et al., 2019; Ni & Binke, 2020).

We modified these ResNet networks to handle our input images, which consist of three-channel images with the distorted image, reference image, and difference image utilized as each of the channels, as shown in Figure 5. The three-channel image is fed into a randomly initialized convolutional layer followed by the ResNet architecture. We adapted the final layer of ResNet to our problem by replacing it with a fully connected layer with the same number of outputs as the error states. Figure 6 shows an example of the overall architecture using Wide ResNet 50_2.

In transfer learning, the weights of the pretrained network can either be fine-tuned (with the pretrained weights used for initialization), frozen, or randomly initialized. In our experiments, we found that fine-tuning all of the ResNet weights resulted in better performance than “freezing” any of the weights or randomly initializing them. This result is somewhat remarkable, considering that the data

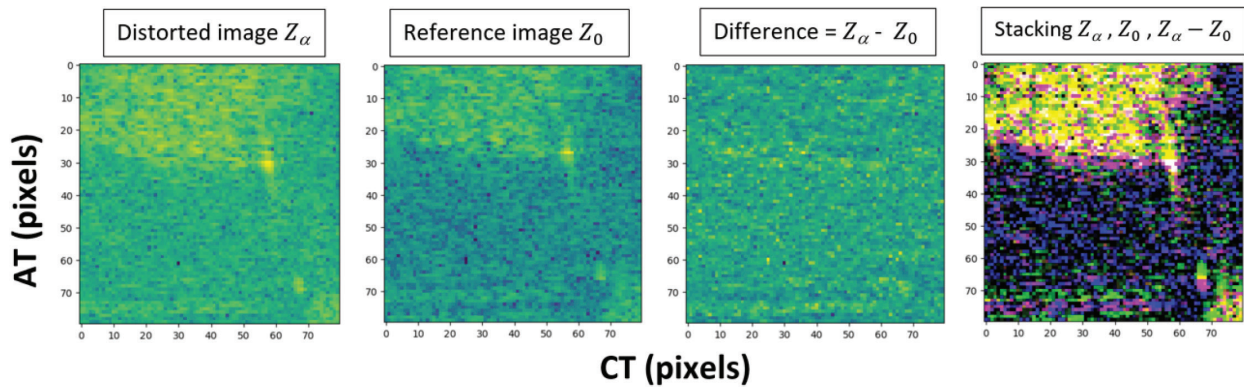


FIGURE 5 A three-channel image input

The input of our model consists of a three-channel image: the distorted image (Z_α), the reference image (Z_0), and the difference image ($Z_\alpha - Z_0$). The proposed model is trained by using the stacked image (image input), which is treated as a different channel in the first convolutional layer.

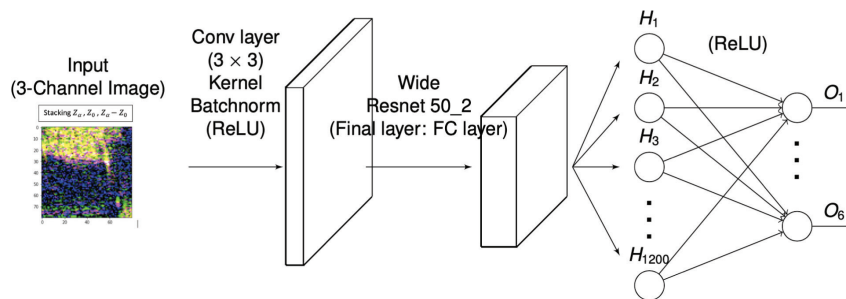


FIGURE 6 Full architecture for using transfer learning with Wide ResNet 50_2

The input consists of a three-channel image with the distorted image, reference image, and difference image (see Figure 5), which is fed into a randomly initialized convolutional layer followed by the considered ResNet architecture (e.g., Wide ResNet 50_2, ResNet-34). We adapted the final layer of ResNet to our problem by replacing it with a fully connected layer with the same number of outputs as error states. FC: fully connected; ReLU: rectified linear unit

sets used for pretraining these networks contain natural images, which differ considerably from SAR images. This finding suggests that the pretrained network has learned general image features (e.g., lines and other shapes) that are also present in SAR images, which is consistent with other findings in the literature (Tajbakhsh et al., 2016; Yosinski et al., 2014).

We selected the Wide ResNet 50_2 network for our final model because it outperformed all other models (see Table 11 in Appendix B) and because its wide (and relatively shallow) nature enables faster training compared with other ResNet architectures. We considered two approaches for training the model with the real data. In the first approach, we train the model using only the training set from the data described in Table 4. In the second approach, we use transfer learning by pretraining the network (which is already pretrained on other images) on the training set of simulated data; using the resulting parameters as a starting point, we then train the model using the training set from the real data. The second approach is attempted to determine whether pretraining using simulated data can improve the performance on real data; the effects of this approach are discussed further in the next section.

5 | EXPERIMENTAL RESULTS

Here, we present our experimental results obtained using the Wide ResNet 50_2 network. Results using other models are presented in Appendix A and B. We applied our model to simulated and real data, and the results were quantified in terms of the MSE (see Equation (7)) obtained on the test data for each scenario.

5.1 | Simulated Data

Table 5 lists the results obtained in all six scenarios for the sim-5-s data set using the selected model, Wide ResNet 50_2. For all scenarios in the simulated data set, the test MSE is less than 1 for all errors considered. In particular, the simulated data set 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, for some of the errors considered, the test MSE is close to unity. This result indicates that as more errors are introduced, the performance degrades, suggesting difficulties in resolving ambiguous error sources.

TABLE 5

MSE for Each Error State for Scenarios 1–6 of the Sim-5-s Data Set

A test MSE value of less than 1 is obtained for all scenarios, showing that the network is able to learn most of the considered errors to some degree.

Scenario	Test MSE (Sim-5-s)						
	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Avg.
1	0.059	0.029	N/A	N/A	N/A	N/A	0.044
2	N/A	N/A	N/A	0.246	0.116	N/A	0.181
3	0.523	0.224	N/A	0.244	0.126	N/A	0.279
4	0.094	0.920	0.280	N/A	N/A	N/A	0.432
5	N/A	N/A	N/A	0.283	0.798	0.546	0.543
6	0.632	0.914	0.525	0.368	0.866	0.533	0.640

5.2 | Real Data

Table 6 shows the results obtained for the real data sets using the proposed model. For scenarios 1 and 4, the best performance was obtained for a 2-s aperture. The network performs particularly well in scenario 1, with a low test MSE (see also Figure 7). We note that ambiguity exists in CT shifts for scenario 4, due to either CT or D position errors. The network, however, is able to identify some subtleties in the distorted SAR images and at least partially attribute the effect to the corresponding error. Figure 8 shows the distribution of the error states before and after estimation of the navigation errors for scenarios 1 and 4. The resulting error distributions after estimation 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 data set, increasing the number of ambiguous error sources results in degraded overall performance, with a test MSE near unity for cases in which little information is learned by the network.

TABLE 6

Model Performance for Each Error State for Scenarios 1–6 of the Real Data Sets for a 2-s, 6-s, 10-s, or 15-s Aperture Length Using the Wide ResNet 50_2 Model
 The network is able to learn many of the errors for the real data sets (MSE < 1). The bold numbers represent the smallest test MSE of the considered error state for each scenario to indicate which data set achieves the best performance.

Scenario/Data Set		Test MSE						
		AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Avg.
1	Real-2-s	0.056	0.043	N/A	N/A	N/A	N/A	0.049
	Real-6-s	0.171	0.133	N/A	N/A	N/A	N/A	0.152
	Real-10-s	0.181	0.134	N/A	N/A	N/A	N/A	0.157
	Real-15-s	0.197	0.130	N/A	N/A	N/A	N/A	0.163
2	Real-2-s	N/A	N/A	N/A	1.073	0.168	N/A	0.621
	Real-6-s	N/A	N/A	N/A	1.033	0.139	N/A	0.589
	Real-10-s	N/A	N/A	N/A	0.790	0.081	N/A	0.435
	Real-15-s	N/A	N/A	N/A	0.793	0.081	N/A	0.437
3	Real-2-s	1.066	0.234	N/A	1.068	0.147	N/A	0.629
	Real-6-s	1.019	0.218	N/A	0.992	0.200	N/A	0.607
	Real-10-s	0.886	0.292	N/A	0.789	0.132	N/A	0.525
	Real-15-s	0.779	0.239	N/A	0.776	0.128	N/A	0.480
4	Real-2-s	0.102	0.810	0.373	N/A	N/A	N/A	0.429
	Real-6-s	0.211	0.874	0.422	N/A	N/A	N/A	0.502
	Real-10-s	0.269	0.817	0.419	N/A	N/A	N/A	0.502
	Real-15-s	0.314	0.861	0.424	N/A	N/A	N/A	0.533
5	Real-2-s	N/A	N/A	N/A	1.070	0.646	0.580	0.765
	Real-6-s	N/A	N/A	N/A	1.047	0.720	0.576	0.781
	Real-10-s	N/A	N/A	N/A	0.907	0.645	0.578	0.710
	Real-15-s	N/A	N/A	N/A	0.971	0.702	0.568	0.747
6	Real-2-s	1.085	0.751	0.535	1.087	0.604	0.598	0.777
	Real-6-s	1.042	0.796	0.496	1.024	0.712	0.527	0.766
	Real-10-s	0.988	0.738	0.510	0.945	0.623	0.511	0.719
	Real-15-s	0.957	0.814	0.516	0.920	0.709	0.533	0.742

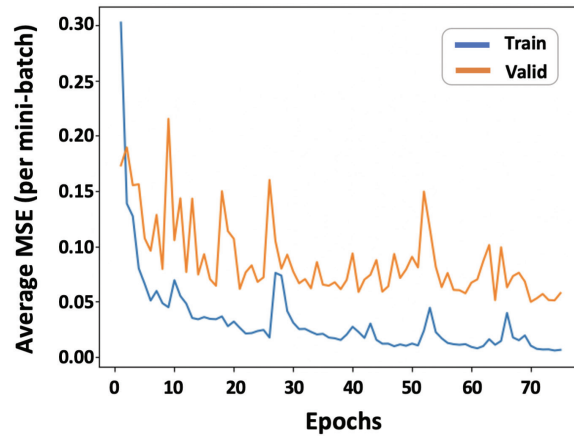


FIGURE 7 Training and validation MSE as a function of training epoch for scenario 1 for the real-2-s data set
The gap between the errors is small, suggesting that little overfitting is occurring.

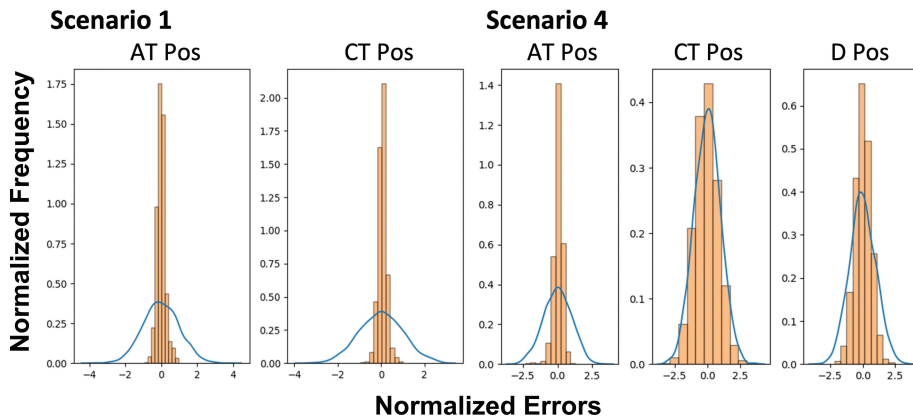


FIGURE 8 Distribution of error states before (blue line) and after (histogram) estimation for the real-2-s data set for scenarios 1 and 4
In both cases, the error distributions are more concentrated around zero, indicating that the network has improved our ability to estimate navigation errors. No outliers have been introduced by the network.

For some of these errors, the network may also be overfitting. Overfitting typically occurs when the gap between the training error and validation/test error is large. Figure 9 shows the training and validation MSE as a function of training epoch for scenario 2 for the real-2-s data set. Here, 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.

Most of the errors that caused issues for the network on the real-2-s data are associated with image blurring. For example, as shown in scenario 2 for the real-2-s data set, the network was not able to characterize blur distortions despite the lack of error ambiguity. Because blurring errors are more apparent with longer flight trajectories, we hypothesized that increasing the aperture length to emphasize the blurring distortion in the images would improve the performance.

Indeed, we found that as the aperture length increased from 2 s to 10 s in scenario 2, the MSE for the AT velocity error decreased by more than 25%, suggesting

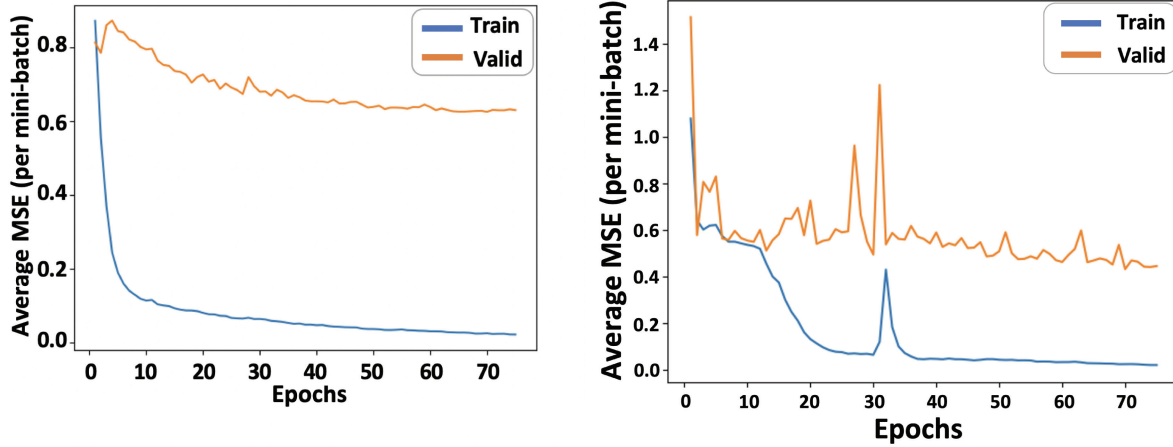


FIGURE 9 Training and validation MSE as a function of training epoch for scenario 2 for the real-2-s (left) and real-10-s (right) data sets

There is a large gap between the training and validation error in the 2-s data, suggesting that the network may be overfitting. In contrast, the gap is smaller for the 10-s data, suggesting that less overfitting occurs for the real-10-s data. The large variations in losses are due to the inherent randomness of the SGD.

that the neural network successfully characterized the effect of blurs for the larger apertures. Notably, the neural network also improved its estimation of the CT velocity error by a factor of 2.

Further, in Figure 9, we see that the gap between the training and validation error for scenario 2 for the real-10-s data is smaller than the gap shown on the figure for the real-2-s data, suggesting that less overfitting occurs on the real-10-s data. In fact, Table 6 shows that the MSE for all scenarios on the real-10-s data set is less than 1, and the best performance in scenario 6, in which all errors are introduced, is achieved using a 10-s aperture. This result indicates that the neural network performs better when the blurring errors are more apparent with a larger aperture length.

To evaluate this trend further, we tested our proposed method using the real data sets with aperture lengths of 6 s and 15 s for scenarios 1–6. Table 6 demonstrates that as the aperture length increases, the network performs better in some settings. For instance, in scenarios 2 and 3, the performance improves slightly as the aperture length increases from 2 s to 6 s for the real data.

As the aperture length increases further, the network performs even better in scenarios 2 and 3. As the aperture length increased from 6 s to 15 s, the MSE of the AT position error decreased by more than 20% in scenario 3. Furthermore, the MSE of the AT velocity error decreased by more than 20% in scenarios 2 and 3, suggesting again that the neural network successfully characterized the effect of blurs for the larger apertures.

A larger aperture does not lead to uniform improvement in all scenarios. For scenarios 1 and 4, in which only positional errors were included, the best MSE performance was obtained for a 2-s aperture length. However, for all other scenarios in which velocity errors are introduced, either the 10-s or 15-s aperture length performed the best. Regardless, in all of the scenarios for the real-10-s and real-15-s data sets, the MSE is less than 1, which is not the case for the 2-s aperture data. These results confirm our hypothesis: As the synthetic aperture length of the real data increases (up to a point), the performance of our proposed method improves when velocity errors are included, which can induce blurring errors and other ambiguities.

We can compare our results here with those obtained by Lindstrom et al. (2022a). Lindstrom et al. (2022a) used an EKF combined with range–Doppler images that were compared with a reference image. Although the images created for this approach are different, Lindstrom et al. (2022a) used the same raw data employed in this work, with similar error statistics. Thus, we can compare our results to those obtained by Lindstrom et al. (2022a) and account for any differences in the statistics by normalizing the errors.

Because Lindstrom et al. (2022a) used altimeter measurements to control the vertical errors, their results are most analogous to our scenario 3, where we consider the AT and CT positions and velocities. We first consider the initial errors of the real data reported by Lindstrom et al. (2022a). The normalized AT position initial error is approximately 0.39, whereas the normalized CT position initial error is greater than 1. The normalized AT velocity initial error is approximately 1, and the normalized CT velocity initial error is approximately 0. The AT position and CT velocity errors compare favorably with our approach, although the CT position and AT velocity errors are worse. However, as time goes on, the AT errors tend to converge to zero whereas the CT errors do not achieve a steady state. This trend seems to contrast with our results, which showed CT errors that were lower than AT errors.

Overall, the comparison here does not show a clear winner between the two approaches. Combining our machine-learning approach with the EKF approach of Lindstrom et al. (2022a) may lead to overall improved results. We leave this for future work.

5.3 | Pretraining Using Simulated Data

As an alternative to increasing the aperture length, we hypothesized that transfer learning using simulated data could improve the network’s performance on the real-2-s data set. To apply transfer learning in this approach, we pretrained the proposed network on the sim-5-s data and then trained on the real-2-s data. We then applied this technique to the real-2-s data for scenario 2 to evaluate and compare the performance.

Table 7 displays the performance for scenario 2, showing negligible improvement in both the AT and CT velocity MSE. Similar results were obtained when we applied the transfer learning technique to the real-10-s data set. This finding suggests that the simulated and real data are not similar enough to justify this approach for transfer learning. Generating more realistic simulated data could improve the performance.

TABLE 7

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-s and real-10-s data and (2) the results of using transfer learning, pretraining on simulated data, and training on the real-2-s or real-10-s data set. The Wide ResNet 50_2 model approach was used in both cases. Increasing the aperture length from 2 s to 10 s improves the performance. Improvements from transfer learning are negligible.

Data Set, Scenario 2	AT Vel	CT Vel
Test MSE (Real-2-s)	1.073	0.168
Test MSE (Real-10-s)	0.790	0.081
Test MSE (Transfer Learning - 2 s)	1.043	0.085
Test MSE (Transfer Learning - 10 s)	0.900	0.085

6 | CONCLUSION

We trained a CNN to estimate position and velocity errors at the beginning of a SAR data collection period by comparing a distorted SAR image against 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 generally degrades because of the network's inability to fully distinguish between errors that result in the same image distortion. Nevertheless, the network performs better than random guessing in all tested error scenarios with an appropriate aperture length. This result suggests that the network can identify some subtleties in the image distortions that are not readily apparent to the human eye.

This study acknowledges the presence of navigation errors at the onset of GPS-denied periods. These errors induce distortions between online-generated and reference SAR images. These image distortions may encompass various types of imaging errors, including Doppler errors and higher-order phase errors. The proposed neural network does not directly analyze or quantify these imaging errors, but instead learns their effect from the images and uses that information to infer the navigation errors.

Another key finding from our results is that the estimation performance depends on the synthetic aperture length. As the length increases, the network successfully characterizes blurring in real images and appropriately attributes it to the corresponding error source. Future work should investigate and quantify this effect in more detail. We also note that our work uses log-magnitude SAR images as inputs. It is possible that this approach may neglect potentially useful information in the discarded phase. Future work will explore using complex images as input instead. Future work will also investigate how variations in traditional radar characteristics affect the performance of the neural network. Such characteristics include signal-to-noise ratio, image content (e.g., images with many strong targets versus images with low-contrast clutter), pixel resolution, and carrier frequency.

The inclusion of auxiliary information about viewing geometry may also improve model predictions and should be explored further. For example, the sensitivity of distortions to the vehicle/target geometry may be exploited to reduce ambiguities and improve performance. Possible approaches include augmenting 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 velocimeter measurements such as Doppler lidar or Doppler radar measurements. Other auxiliary measurements may also prove useful in improving model predictions, such as barometric altimeter or velocimeter measurements. For this approach, a velocity sensor could be used to eliminate the AT, CT, and D velocity errors. Using a barometric altimeter to reduce the D position errors would leave us with only AT and CT position errors, which were more easily learned in our experiments.

Combining our machine-learning approach with the information obtained on the phase errors from the auto-focusing method reported by Werness et al. (1990) could also be fruitful. Additionally, combining our approach with the iterative EKF method of Lindstrom et al. (2022a) may also lead to improved results. Finally, including more training data in future iterations will likely help to mitigate overfitting. In particular, data augmentation techniques may improve our method's robustness to changes in the reference image. For example, we can expand the training set by adding noise to or perturbing the reference images.

ACKNOWLEDGMENTS

This research was partially supported by Sandia National Labs via grants 201782, 202136, and 202854 and partially supported by the National Science Foundation under grant 2212325. Raw SAR and navigation data were provided by the Space Dynamics Laboratory X-band radar system at Utah State University. Support and resources from the Center for High-Performance Computing at the University of Utah are gratefully acknowledged.

This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC, under contract No. DE-NA0003525 with the United States Department of Energy (DOE). The employee owns all rights, titles, and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<https://www.energy.gov/downloads/doe-public-access-plan>).

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the United States DOE or the United States Government.

This paper is an extended version of our paper published in the 2021 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (White et al., 2021).

CONFLICT OF INTEREST

The authors claim no conflicts of interest.

REFERENCES

- Bar, Y., Diamant, I., Wolf, L., Lieberman, S., Konen, E., & Greenspan, H. (2015). Chest pathology detection using deep learning with non-medical training. *Proc. of the 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, Brooklyn, NY, 294–297. <https://doi.org/10.1109/ISBI.2015.7163871>
- Bergman, N., Ljung, L., & Gustafsson, F. (1999). Terrain navigation using Bayesian statistics. *IEEE Control Systems Magazine*, 19(3), 33–40. <https://doi.org/10.1109/37.768538>
- Berisha, V., Shah, N., Waagen, D., Schmitt, H., Bellofiore, S., Spanias, A., & Cochran, D. (2007). Sparse manifold learning with applications to SAR image classification. *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '07)*, Honolulu, HI, III–1089. <https://doi.org/10.1109/ICASSP.2007.366873>
- Carrara, W. G., Goodman, R. S., & Majewski, R. M. (1995). *Spotlight SAR: Signal processing algorithms*. Artech House. <https://us.artechhouse.com/Spotlight-Synthetic-Aperture-Radar-Signal-Processing-Algorithms-P523.aspx>
- Chen, S., & Wang, H. (2014). SAR target recognition based on deep learning. *Proc. of the 2014 International Conference on Data Science and Advanced Analytics (DSAA)*, Shanghai, China, 541–547. <https://doi.org/10.1109/DSAA.2014.7058124>
- Chen, S., Wang, H., Xu, F., & Jin, Y.-Q. (2016). Target classification using the deep convolutional networks for SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8), 4806–4817. <https://doi.org/10.1109/TGRS.2016.2551720>
- Christensen, R. S., Gunther, J., & Long, D. (2019). Toward GPS-denied navigation utilizing back projection-based synthetic aperture radar imagery. *Proc. of the ION 2019 Pacific PNT Meeting*, Honolulu, HI, 108–119. <https://doi.org/10.33012/2019.16797>
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *Proc of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, 248–255. <https://doi.org/10.1109/CVPR.2009.5206848>
- Du, H., He, Y., & Jin, T. (2018). Transfer learning for human activities classification using micro-doppler spectrograms. *Proc. of the 2018 IEEE International Conference on Computational Electromagnetics (ICCEM)*, Chengdu, China, 1–3. <https://doi.org/10.1109/ICPEM.2018.8496654>

- Duersch, M. I., & Long, D. G. (2015). Analysis of time-domain back-projection for stripmap SAR. *International Journal of Remote Sensing*, 36(8), 2010–2036. <https://doi.org/10.1080/01431161.2015.1030044>
- Dumoulin, V., & Visin, F. (2016). *A guide to convolution arithmetic for deep learning*. arXiv. <https://doi.org/10.48550/ARXIV.1603.07285>
- Gong, M., Zhao, J., Liu, J., Miao, Q., & Jiao, L. (2016). Change detection in synthetic aperture radar images based on deep neural networks. *IEEE Transactions Neural Networks and Learning Systems*, 27(1), 125–138. <https://doi.org/10.1109/TNNLS.2015.2435783>
- Greco, M., Querry, S., Pinelli, G., Kulpa, K., Samczynski, P., Gromek, D., Gromek, A., Malanowski, M., Querry, B., & Bonsignore, A. (2012). SAR-based augmented integrity navigation architecture. *Proc. of the 13th International Radar Symposium*, Warsaw, Poland, 225–229. <https://doi.org/10.1109/IRS.2012.6233320>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Computer Vision Pattern Recognition*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Hero, A. O., Ma, B., Michel, O. J., & Gorman, J. (2002). Applications of entropic spanning graphs. *IEEE Signal Processing Magazine*, 19(5), 85–95. <https://doi.org/10.1109/MSP.2002.1028355>
- Hostetler, L., & Andreas, R. (1983). Nonlinear Kalman filtering techniques for terrain-aided navigation. *IEEE Transactions on Automatic Control*, 28(3), 315–323. <https://doi.org/10.1109/TAC.1983.1103232>
- Jensen, M., Knight, C., & Haslem, B. (2016). FlexSAR, a high quality, flexible, cost effective, prototype SAR system. *Radar Sensor Technology XX*, 9829. <https://doi.org/10.1117/12.2224951>
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 35–45. <https://doi.org/10.1115/1.3662552>
- Kim, Y., Park, J., & Bang, H. (2018). Terrain-referenced navigation using an interferometric radar altimeter: IRA-based TRN. *NAVIGATION*, 65(2), 157–167. <https://doi.org/10.1002/navi.233>
- Kingma, D., & Ba, L. (2015). Adam: A method for stochastic optimization. *Proc. of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA. <https://doi.org/10.48550/arXiv.1412.6980>
- Krishnapuram, B., Williams, D., Xue, Y., Carin, L., Figueiredo, M., & Hartemink, A. J. (2005). On semi-supervised classification. *Proc. of the Advances in Neural Information Processing Systems 17 (NIPS 2004)*, British Columbia, Canada, 721–728. https://papers.nips.cc/paper_files/paper/2004/hash/7aee26c309def8c5a2a076eb250b8f36-Abstract.html
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>
- Lindstrom, C., Christensen, R., Gunther, J., & Jenkins, S. (2022a). GPS-denied navigation aided by synthetic aperture radar using the range-doppler algorithm. *NAVIGATION*, 69(3). <https://doi.org/10.33012/navi.533>
- Lindstrom, C., Christensen, R., Gunther, J., & Jenkins, S. (2022b). Sensitivity of back-projection algorithm (BPA) synthetic aperture radar (SAR) image formation to initial position, velocity, and attitude navigation errors. *IET Radar, Sonar and Navigation*, 16(2), 364–378. <https://doi.org/10.1049/rsn2.12189>
- Liu, S., Tian, G., & Xu, Y. (2019). A novel scene classification model combining ResNet based transfer learning and data augmentation with a filter. *Neurocomputing*, 338, 191–206. <https://doi.org/10.1016/j.neucom.2019.01.090>
- Loshchilov, I., & Hutter, F. (2017). *Decoupled weight decay regularization*. arXiv. <https://doi.org/10.48550/ARXIV.1711.05101>
- Mason, E., Yonel, B., & Yazici, B. (2017). Deep learning for radar. *Proc. of the 2017 IEEE Radar Conference (RadarConf)*, Seattle, WA, 1703–1708. <https://doi.org/10.1109/RADAR.2017.7944481>
- Ni, Z., & Binke, H. (2020). Human identification based on natural gait micro-doppler signatures using deep transfer learning. *IET Radar, Sonar & Navigation*, 14(10), 1640–1646. <https://doi.org/10.1049/iet-rsn.2020.0183>
- Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). Determination Press. <http://neuralnetworksanddeeplearning.com/>
- Nitti, D., Bovenga, F., Chiaradia, M., Greco, M., & Pinelli, G. (2015). Feasibility of using synthetic aperture radar to aid UAV navigation. *Sensors*, 15(8), 18334–18359. <https://doi.org/10.3390/s150818334>
- Nordlund, P.-J., & Gustafsson, F. (2009). Marginalized particle filter for accurate and reliable terrain-aided navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 45(4), 1385–1399. <https://doi.org/10.1109/TAES.2009.5310306>
- O’Shea, K., & Nash, R. (2015). *An introduction to convolutional neural networks*. arXiv. <https://doi.org/10.48550/ARXIV.1511.08458>
- Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359. <https://doi.org/10.1109/TKDE.2009.191>
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. *Neural Information Processing Systems Workshop on Autodifferentiation*. <https://openreview.net/forum?id=BJJsrnmcZ>
- Pickup, L. C., Capel, D. P., Roberts, S. J., & Zisserman, A. (2007). Bayesian image super-resolution, continued. *Neural Information Processing Systems*, 1089–1096. <https://doi.org/10.7551/mitpress/7503.003.0141>

- Quist, E. B., & Beard, R. W. (2016). Radar odometry on fixed-wing small unmanned aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, 52(1), 396–410. <https://doi.org/10.1109/TAES.2015.140186>
- Quist, E. B., Niedfeldt, P. C., & Beard, R. W. (2016). Radar odometry with recursive-RANSAC. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4), 1618–1630. <https://doi.org/10.1109/TAES.2016.140829>
- Raquet, J., & Martin, R. K. (2008). Non-GNSS radio frequency navigation. *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, NV, 5308–5311. <https://doi.org/10.1109/ICASSP.2008.4518858>
- Samczynski, P. (2012). Superconvergent velocity estimator for an autofocus coherent mapDrift technique. *IEEE Geoscience and Remote Sensing Letters*, 9(2), 204–208. <https://doi.org/10.1109/LGRS.2011.2163700>
- Samczynski, P., & Kulpa, K. (2010). Coherent mapDrift technique. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3), 1505–1517. <https://doi.org/10.1109/TGRS.2009.2032241>
- Scannapieco, A. F., Renga, A., Fasano, G., & Moccia, A. (2018). Experimental analysis of radar odometry by commercial ultralight radar sensor for miniaturized UAS. *Journal of Intelligent and Robotic Systems*, 90(3-4), 485–503. <https://doi.org/10.1007/s10846-017-0688-1>
- Sjanic, Z., & Gustafsson, F. (2010). Simultaneous navigation and SAR auto-focusing. *Proc. of the International Conference on Information Fusion*, Edinburgh, UK, 1–7. <https://doi.org/10.1109/ICIF.2010.5711931>
- Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5), 1299–1312. <https://doi.org/10.1109/TMI.2016.2535302>
- Tipping, M. E., & Bishop, C. M. (2003). Bayesian image super-resolution. *Proc. of the Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Vancouver, Canada, 1303–1310. <http://papers.nips.cc/paper/2315-bayesian-image-super-resolution>
- Ulaby, F., & Long, D. (2014). *Microwave radar and radiometric remote sensing*. University of Michigan Press. <https://ieeexplore.ieee.org/document/9100160>
- Veth, M. J. (2011). Navigation using images, a survey of techniques. *NAVIGATION*, 58(2), 127–139. <https://doi.org/10.1002/j.2161-4296.2011.tb01796.x>
- Werness, S. A., Carrara, W. G., Joyce, L., & Franczak, D. B. (1990). Moving target imaging algorithm for SAR data. *IEEE Transactions on Aerospace and Electronic Systems*, 26(1), 57–67. <https://doi.org/10.1109/7.53413>
- White, T., Wheeler, J., Lindstrom, C., Christensen, R., & Moon, K. R. (2021). GPS-denied navigation using SAR images and neural networks. *Proc. of the ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Ontario, 2395–2399. <https://doi.org/10.1109/ICASSP39728.2021.9414421>
- Wilmanski, M., Kreucher, C., & Lauer, J. (2016). Modern approaches in deep learning for SAR ATR. *Society of Photo-Optical Instrumentation Engineers (SPIE) Defense + Security*, 9843, 98430N. <https://doi.org/10.1117/12.2220290>
- Wyawahare, M. V., Patil, P. M., Abhyankar, H. K., et al. (2009). Image registration techniques: An overview. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(3), 11–28. <https://www.academia.edu/download/49405929/2.pdf>
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Proc. of the Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Montreal, Canada, 3320–3328. <https://doi.org/10.48550/arXiv.1411.1792>
- Zagoruyko, S., & Komodakis, N. (2016). *Wide residual networks*. arXiv. <http://arxiv.org/abs/1605.07146>
- Zaugg, E. C., & Long, D. G. (2015). Generalized frequency scaling and backprojection for LFM-CW SAR processing. *IEEE Transactions on Geoscience and Remote Sensing*, 53(7), 3600–3614. <https://doi.org/10.1109/TGRS.2014.2380154>
- Zhao, Q., & Principe, J. C. (2001). Support vector machines for SAR automatic target recognition. *IEEE Transactions on Aerospace and Electronic Systems*, 37(2), 643–654. <https://doi.org/10.1109/7.937475>

How to cite this article: White, T., Wheeler, J., Lindstrom, C., Bean, B., Jenkins, S., Christensen, R., & Moon, K. R. (2025). Inferring inertial navigation errors from SAR image distortions using a convolutional neural network. *NAVIGATION*, 72(4). <https://doi.org/10.33012/navi.727>

APPENDICES

Here, we show the full results obtained using various neural network models. All of these models take as input the distorted image and a reference image (as shown in Figure 5) and attempt to predict the flight trajectory errors. 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 (Paszke et al., 2017). We also used the MSE loss function in model tuning and training. Below, we show that the ResNet models (advanced CNN-based architectures) outperform other simple neural network architectures.

A | RESULTS USING SIMPLE MODELS

In the process of exploring several methods for estimating the initial errors, we applied different methods to the problem, such as a simple CNN (S-CNN) and a fully connected (FC) neural network to the sim-5-s and real-2-s data sets for scenarios 1–3. These simple neural networks were not pretrained. The FC model consisted of two fully connected hidden layers and a fully connected output layer, as shown in Figure 10. We used a linear activation function in the output layer. We focused on tuning the learning rate, batch size, and number of epochs. The learning rate and batch size were tuned using a grid search over candidate hyperparameters, retaining the model with the lowest average MSE for the validation set. The number of epochs was tuned by tracking the validation accuracy over epochs and retaining the epoch with the lowest validation MSE. Dropout was fixed at $p = 0.5$ for all fully connected layers during training.

As CNNs have been used successfully in image classification and feature extraction (LeCun et al., 1998), we also applied an S-CNN model to the sim-5-s and real-2-s data sets for scenarios 1–3. This S-CNN model consists of two convolutional layers followed by a fully connected hidden layer and a fully connected output layer, as shown in Figure 11.

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. The learning rate and batch size were tuned by using a grid search over candidate hyperparameters, retaining the model with the lowest average MSE

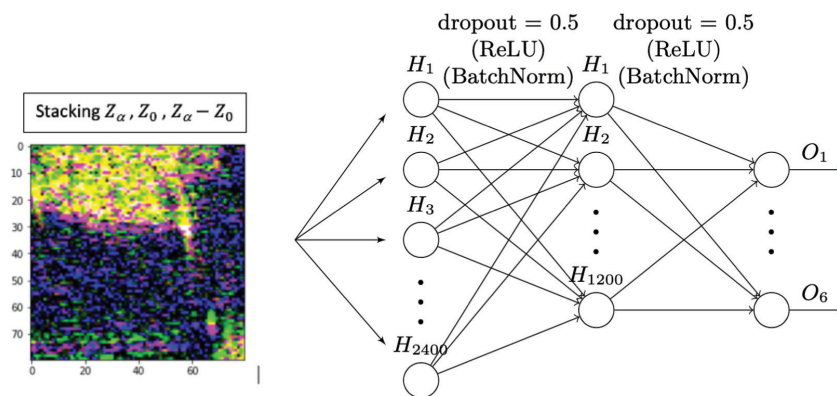


FIGURE 10 FC model

This neural network model consists of two fully connected hidden layers and a fully connected output layer.

for the validation set. The number of epochs was tuned 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 overfitting.

Table 8 shows a summary of the performance for each error state for scenarios 1–3 of the sim-5-s data using the S-CNN and FC model. The S-CNN model performs well, showing an MSE of less than 1 in each scenario. These results suggest that the neural network is capable of characterizing both shifts and blurs in simulated data, in the absence of ambiguity. The FC model performs well in scenario 1 on the sim-5-s data; however, for scenarios 2 and 3, some of the error states present an MSE greater than 1, which indicates that the performance degrades as more errors are introduced.

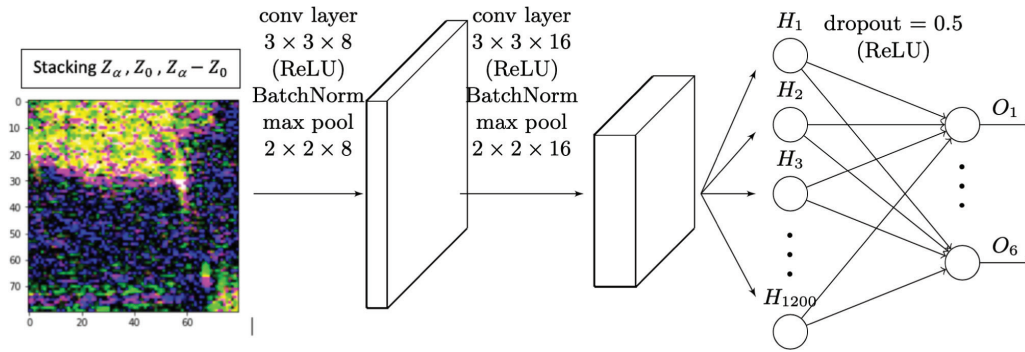


FIGURE 11 S-CNN architecture

This network is a CNN with two convolutional layers followed by a fully connected hidden layer and a fully connected output layer.

TABLE 8

Summary of Model Performance for Each Error State for Scenarios 1–3 for the Sim-5-s Data Using the S-CNN and FC Models

Scenario/Model		Test MSE (Sim-5-s)						
		AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Avg.
1	S-CNN	0.092	0.054	N/A	N/A	N/A	N/A	0.073
	FC	0.560	0.234	N/A	N/A	N/A	N/A	0.397
2	S-CNN	N/A	N/A	N/A	0.475	0.402	N/A	0.438
	FC	N/A	N/A	N/A	1.154	0.425	N/A	0.790
3	S-CNN	0.709	0.335	N/A	0.569	0.339	N/A	0.488
	FC	1.136	0.360	N/A	1.128	0.350	N/A	0.743

TABLE 9

Summary of Model Performance for Each Error State for Scenarios 1–3 for Real-2-s Data Using the S-CNN and FC Models

Scenario/Model		Test MSE (Real-2-s)						
		AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Avg.
1	S-CNN	0.057	0.041	N/A	N/A	N/A	N/A	0.049
	FC	1.066	1.076	N/A	N/A	N/A	N/A	1.071
2	S-CNN	N/A	N/A	N/A	1.095	0.196	N/A	0.645
	FC	N/A	N/A	N/A	1.436	1.107	N/A	1.272
3	S-CNN	0.999	0.197	N/A	0.985	0.199	N/A	0.595
	FC	1.432	0.997	N/A	1.395	0.975	N/A	1.200

Table 9 shows a summary of the performance for each error state for scenarios 1–3 of the real-2-s data using the S-CNN and FC models. The S-CNN model performs well for scenario 1, as the test MSE is less than 1 for the considered errors (AT position and CT position). However, for scenarios 2 and 3, the test MSE for some of the considered errors is greater than (or close to) 1, suggesting that the performance degrades as velocity errors are introduced. For the FC model, the test MSE is greater than 1 for all considered scenarios (1–3). These results suggest that the FC model is not learning how to predict the initial navigation errors as desired.

B | RESULTS USING RESNET MODELS

CNNs were chosen for this problem because they have been successful in various problems involving images. The convolutional filters that are part of CNNs allow the network to learn short- and long-range dependencies between image pixels. Because a basic CNN model is able to learn the initial navigation errors in some scenarios, as shown in Appendix A, more advanced CNN architectures were considered: ResNet 34 (He et al., 2016), ResNet 50 (He et al., 2016), ResNet 101 (He et al., 2016), ResNet 152 (He et al., 2016), Wide ResNet 50_2 (Zagoruyko & Komodakis, 2016), and Wide ResNet 101_2 (Zagoruyko & Komodakis, 2016).

For all of the ResNet approaches, as shown in Figure 12, our input consists of a three-channel image with the distorted image, reference image, and difference image as each of the channels, as shown in Figure 5. This three-channel image input is fed into a randomly initialized convolutional layer followed by the considered

TABLE 10

Summary of Model Performance for Each Error State for Scenarios 1–3 on the Sim-5-s Data Set Using the ResNet (RN) and Wide ResNet (WRN) Models

The bold numbers represent the smallest test MSE of the considered error state for each scenario to indicate which model achieves the best performance. The results show that the Wide ResNet 50_2 model achieves the best performance more frequently than any other model.

Scenario/Model	Test MSE (Sim-5-s)							
	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Avg.	
1	RN-34	0.085	0.031	N/A	N/A	N/A	N/A	0.058
	RN-50	0.055	0.045	N/A	N/A	N/A	N/A	0.050
	RN-101	0.058	0.039	N/A	N/A	N/A	N/A	0.048
	RN-152	0.057	0.038	N/A	N/A	N/A	N/A	0.047
	WRN 50_2	0.059	0.029	N/A	N/A	N/A	N/A	0.044
	WRN 101_2	0.065	0.034	N/A	N/A	N/A	N/A	0.049
2	RN-34	N/A	N/A	N/A	0.207	0.146	N/A	0.177
	RN-50	N/A	N/A	N/A	0.2544	0.114	N/A	0.184
	RN-101	N/A	N/A	N/A	0.253	0.204	N/A	0.229
	RN-152	N/A	N/A	N/A	0.278	0.271	N/A	0.275
	WRN 50_2	N/A	N/A	N/A	0.246	0.116	N/A	0.181
	WRN 101_2	N/A	N/A	N/A	0.309	0.189	N/A	0.249
3	RN-34	0.782	0.267	N/A	0.609	0.174	N/A	0.458
	RN-50	0.614	0.269	N/A	0.300	0.115	N/A	0.325
	RN-101	0.577	0.255	N/A	0.267	0.129	N/A	0.307
	RN-152	0.572	0.236	N/A	0.296	0.133	N/A	0.309
	WRN 50_2	0.523	0.224	N/A	0.244	0.126	N/A	0.279
	WRN 101_2	0.597	0.276	N/A	0.300	0.172	N/A	0.336

ResNet architecture. The final layer of ResNet is replaced with a fully connected layer with the same number of outputs as the error states. For each of the ResNet architectures, pretrained models are available in the PyTorch framework.

As shown in Table 10, all of the ResNet models performed very well in all scenarios ($MSE < 1$) on the simulated data. In particular, the Wide ResNet 50_2 model

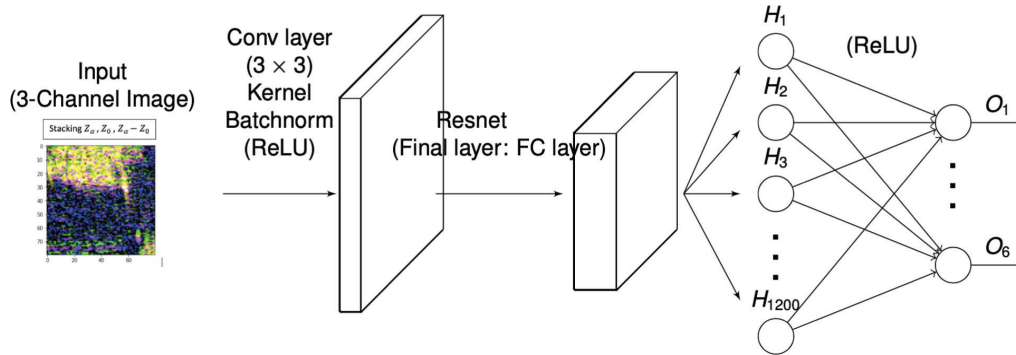


FIGURE 12 Architecture of the ResNet model approach

These architectures illustrate an example of the features used in the approach using ResNet models. The input consists of a three-channel image with the distorted image, reference image, and 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, ResNet 101). The final layer of ResNet is replaced with a fully connected layer with the same number of outputs as the error states.

TABLE 11

Summary of Model Performance for Each Error State for Scenarios 1–3 on the Real-2-s Data Set Using the ResNet (RN) and Wide ResNet (WRN) Models

The bold numbers represent the smallest test MSE of the considered error state for each scenario to indicate which model achieves the best performance. As observed for the simulated data, the Wide ResNet 50_2 model exhibited superior performance more frequently than any other model on the real-2-s data.

Scenario/Model	Test MSE (Real-2-s)							
	AT Pos	CT Pos	D Pos	AT Vel	CT Vel	D Vel	Avg.	
1	RN-34	0.071	0.035	N/A	N/A	N/A	N/A	0.053
	RN-50	0.059	0.052	N/A	N/A	N/A	N/A	0.055
	RN-101	0.061	0.047	N/A	N/A	N/A	N/A	0.054
	RN-152	0.093	0.050	N/A	N/A	N/A	N/A	0.071
	WRN 50_2	0.056	0.043	N/A	N/A	N/A	N/A	0.049
	WRN 101_2	0.062	0.049	N/A	N/A	N/A	N/A	0.055
2	RN-34	N/A	N/A	N/A	1.255	0.086	N/A	0.670
	RN-50	N/A	N/A	N/A	1.140	0.081	N/A	0.610
	RN-101	N/A	N/A	N/A	1.143	0.119	N/A	0.631
	RN-152	N/A	N/A	N/A	1.142	0.810	N/A	0.612
	WRN 50_2	N/A	N/A	N/A	1.073	0.168	N/A	0.621
	WRN 101_2	N/A	N/A	N/A	1.155	0.120	N/A	0.637
3	RN-34	1.099	0.200	N/A	1.111	0.107	N/A	0.629
	RN-50	1.188	0.185	N/A	1.151	0.118	N/A	0.660
	RN-101	1.156	0.217	N/A	1.105	0.126	N/A	0.651
	RN-152	1.125	0.224	N/A	1.141	0.117	N/A	0.652
	WRN 50_2	1.066	0.234	N/A	1.068	0.147	N/A	0.629
	WRN 101_2	1.193	0.173	N/A	1.124	0.123	N/A	0.653

achieved superior performance more frequently than any other model, except in scenario 2, where the Wide ResNet 50_2 model achieved the second-best performance on average.

Table 11 shows the results obtained by applying the proposed approach to the real-2-s data. The outcomes obtained suggest that all of the considered ResNet architectures had a similar performance for scenarios 1–3. In scenario 1, the test MSEs of the ResNet models were less than 1; however, in scenarios 2 and 3, some of the error states obtained a test MSE greater than 1. Notably, the Wide ResNet 50_2 model exhibited superior performance more frequently than any other model on both the sim-5-s and real-2-s data.

Hence, after testing these architectures, we selected the Wide ResNet 50_2 network because it outperformed all other models and because its wide nature enables faster training, allowing more time to be devoted to testing specific hyperparameters.