Spacecraft Time-Series Anomaly Detection Using Transfer Learning

Sriram Baireddy* Sundip R. Desai† James L. Mathieson†
Richard H. Foster† Moses W. Chan† Mary L. Comer* Edward J. Delp*
* Video and Image Processing Lab (VIPER), Purdue University, West Lafayette, Indiana, USA
† Advanced Technology Center, Lockheed Martin Space, Sunnyvale, California, USA

Abstract

Anomaly detection in telemetry channels is a high priority for spacecraft, especially when considering the harsh environment of space and the magnitude of launch and operation costs. Traditional spacecraft anomaly detection methods are limited in scope and rely on domain experts to correctly determine abnormal behavior. However, with thousands of distinct telemetry channels being transmitted, the amount of data is difficult to monitor manually. Deep learning models can be used to learn the normal behavior of the telemetry channels and flag or label any deviations. The problem is that we have to train a unique model for each channel to ensure best performance. With the large number of channels to monitor, this may not always be possible. We propose using principles of transfer learning to quickly adapt a general pre-trained model to any specific telemetry channel, greatly reducing the number of unique models needed and reducing the training time for each model. We present the results of our approach on the NASA SMAP/MSL dataset to show that we can achieve performance comparable to state-of-the-art anomaly detection methods.

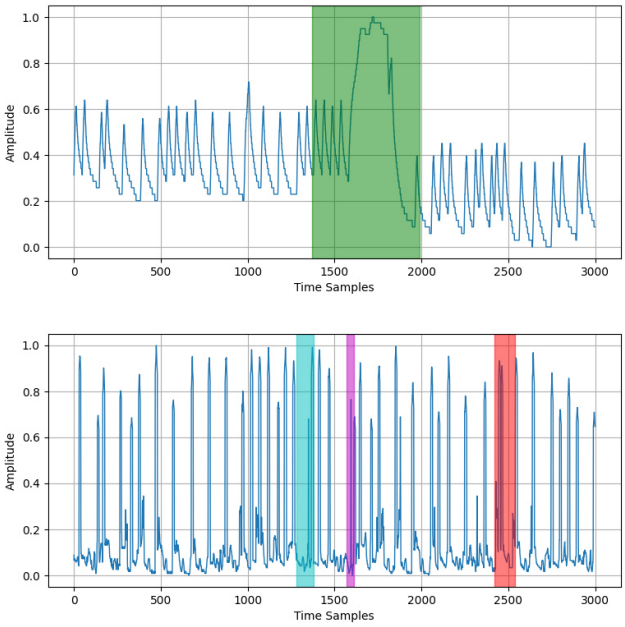

Figure 1: An example of normalized time-series anomalies identified by experts (highlighted in various colors) in two spacecraft telemetry channels.

1. Introduction

The world has numerous satellite constellations in orbit collecting huge amounts of data, such as high-resolution multispectral imagery, as well as internal systems monitoring. These satellites are very expensive, highly complicated systems that operate in the most extreme environment. It is vital to monitor these satellite systems, as well as systems in other spacecraft, for abnormalities that function as precursors to system failure to avoid catastrophic damages and costs. A few examples of anomalous behavior can be seen in Figure 1. Due to the nature of these systems, there are usually thousands of telemetry channels that should be monitored to fully capture and describe any anomalous behavior or system failure. Current system monitoring is performed in a limited scope by domain experts who observe each channel and manually flag sequences they believe to be anomalous [14]. The limiting factor in this scenario is the number of available experts and the time-consuming nature of manual observation.

With the increased interest in machine learning, in particular deep learning, work has been done to show the effectiveness of an automated approach to anomaly detection [3, 19, 22]. The effectiveness of deep learning is usually dependent on the amount of data available for training. Though the limited domain expert knowledge remains

a factor by making labeled anomalous data scarce, we can utilize deep learning in a semi-supervised manner. An efficient semi-supervised anomaly detection approach is to learn the normal and expected behavior of a telemetry channel, so any deviations from this behavior can be flagged in post-processing [2]. This is done effectively by utilizing a recurrent neural network (RNN) as a predictor and a mathematical model of expected prediction errors [14, 20]. However, the large number of channels to monitor hamstrings this strategy for any realistic application; a unique RNN will need to be trained from scratch for every channel for this approach to work optimally.

Thus, we investigate transfer learning, which deals with adapting deep learning models for problems different from their initial task. The nature of the telemetry data recorded by the spacecraft means that there are undoubtedly similarities and correlations between various signal channels, both inter- and intra-subsystem. One way to avoid training thousands of unique deep learning models from scratch would be to use a single anomaly detector for multiple channels, or an entire subsystem, with some finetuning for the predictor to tailor its performance for each channel. For this to have the most impact, our goal is to have a general pretrained predictor model that can adapt to a specific telemetry channel in minimal time.

In this paper, we describe a procedure for training a general predictor model as well as an approach to finetune the model for any specific telemetry channel. After the finetuning is complete, we extract the anomalies by modeling the prediction error using Kernel Density Estimation (KDE) and dynamically thresholding the log-likelihood of the error sequence. We show that it is possible to use knowledge of characteristics and patterns in one spacecraft system to quickly learn information about another spacecraft system and detect anomalous behavior.

2. Background

2.1. Time-Series Prediction

A recurrent neural network (RNN) is a deep learning model archetype first introduced by Rumelhart *et al*. in 1986 [25]. It contains connections between hidden layers, allowing the model to retain information about past inputs, enabling time-series modeling. The original model suffers from the *vanishing gradient* problem, making it unable to learn from long sequences [23]. Two modified RNN architectures have since risen in popularity: (1) Long-Short Term Memory (LSTM) models, proposed in [13]; and (2) Gated Recurrent Units (GRUs), proposed in [6]. Both approaches utilize gates to filter what information is kept and what is discarded, allowing learning on longer time-series sequences. Many modern sequence-based processes utilize one of these neural networks.

2.2. Transfer Learning

The standard approach in deep learning is to train for a specific task; if the task or data changes beyond the controlled parameters, the model must be rebuilt. Transfer learning is the process of using knowledge gained from one task to perform another, preferably related, task [32]. Neural networks have been observed to initially capture general information about the training data before making task-specific connections in deeper layers [30]. Therefore, models trained for similar tasks must extract similar information from the data in the early layers. Instead of training the model from scratch every time, we can start with weights inherited from an already trained model. This allows the training process to adjust task-specific weights in deeper layers while already optimized weights need not be relearned. A consequence of this approach is very specific tasks with limited training sets now have viable deep learning solutions, as the large amount of data required to train a normal model is no longer necessary. This can be seen for tasks such as plant counting [1] and endoscopy disease detection [24]. Transfer learning has found applications in temporal tasks such as natural language processing (NLP) as well. For example, Chen *et al*. [5] were able to train NLP models for low-resource languages by using information from other related languages.

3. Anomaly Detection

Depending on the amount of labeled data available, the philosophy of time-series anomaly detection changes. In the rare case where sufficient ground truth information exists, supervised anomaly detection approaches are essentially classification problems. Recent work in supervised anomaly detection has involved transfer learning to account for the lack of labeled data. Wen and Keyes [29] show that modified U-Nets can be used for supervised anomaly detection after pre-training on synthetically generated normal and abnormal time-series data. The effectiveness of a one-nearest-neighbor approach using dynamic time warping for classifying unseen data has been demonstrated by Vecruyssen *et al*. [28].

For spacecraft telemetry channels, the scarcity of labeled anomaly data makes a semi-supervised approach to time-series anomaly detection more promising. The general idea behind the semi-supervised approach is the time-series prediction equivalent of a "one-class" approach [2, 31]. Since anomalies can be varying in nature, an efficient detection approach is to instead learn what the normal behavior of a channel looks like. Then, while inferencing on the data, any large prediction errors can be flagged as anomalous, as that means the data has deviated from the expected value [11].

The simplest form of anomaly detection has been out-of-limit (OOL) approaches. OOL approaches use simple

pre-defined thresholds and the values of the data samples to identify anomalous behavior. Subsequently, more advanced detection techniques based on nearest-neighbors and clustering have been investigated [2]. They are an improvement over OOL approaches, but have disadvantages relating to parameter specification, interpretability, or computational expense [14]. Recently, deep learning approaches using RNNs, such as LSTMs or GRUs, have proven to be very effective at detecting anomalies in time-series data. Malhotra *et al*. [20] showed the utility of LSTMs for detecting abnormal behavior in space shuttle data, as well as other time-series signals like ECG data, which has been corroborated by Chauhan and Vig [3]. They extended their work by using an LSTM-based encoder-decoder structure to work with multi-channel data [19]. Nanduri and Sherry used both LSTMs and GRUs to detect anomalies in aircraft data [22]. This application of RNNs has extended to spacecraft data as well. LSTMs were shown to be efficient in detecting anomalies in satellite and rover data by Hundman *et al*. [14], and the results were improved by an ensemble approach of LSTMs and an SVM proposed by Li *et al*. [18]. Deep learning approaches using generative adversarial networks (GANs) have been shown to be useful for time-series anomaly detection as well. Li *et al*. [17] used GANs consisting of LSTMs to distinguish anomalous data. Geiger *et al*. [9] subsequently showed that similar GANs trained with cycle-consistency loss could be used for unsupervised anomaly detection.

4. Our Approach

As discussed above, deep learning time-series predictors such as RNNs, and more specifically LSTMs, have shown to be very useful for detecting anomalous behavior in time-series data. Since our application scenario deals with thousands of spacecraft telemetry channels to monitor, the optimal structure of an LSTM model may vary for each channel. One approach to this issue is to utilize a more complicated predictor model whose architecture does not necessarily need to be optimized for each telemetry channel [18]. We instead investigate the possibility of training a generalized time-series predictor model with a fixed structure, and then finetuning on a specific channel to learn specific information about its normal behavior, removing the need for architecture optimization by the user. Anomalies can then be identified using the prediction error of the finetuned predictor model during inference. Thus, our anomaly detection approach has three stages: (1) Time-Series Prediction, (2) Transfer Learning, and (3) Anomaly Extraction.

4.1. Time-Series Prediction

Consider the multi-channel time-series $\mathbf{X} \in \mathcal{X}^{m \times n}$ that is comprised of data from a spacecraft system or subsystem, where m is the number of telemetry channels, each of which has n time samples. If we want to predict the next l steps of $\mathbf{X}$ (meaning predicting for all m channels) given the previous p steps, one approach is to divide the problem into m sub-problems and train a unique, optimized network for each channel. The other approach is to treat the m channels as samples from an unknown general distribution $\mathcal{X}$ that represents the behavior of the system as a whole. This means that a single model trained on all of the data could learn to predict the next l steps of any of the m channels provided to it.

This problem can be formalized as follows, using notation from the sequence-to-sequence modeling literature [21]. We denote the ith time-series of $\mathbf{X}$ as $X(i)$ and its value at time t as $X_t(i)$. Additionally, the sequence $(X_a(i), X_{a+1}(i), \cdots, X_b(i))$ is written in shorthand as $X_a^b(i)$. We want to learn a mapping function $f : \mathcal{X}^p \rightarrow \mathcal{X}^l$ by training a predictor model, where p is the number of previous samples provided to the model and l is the number of future samples predicted by the model. This means that the training set $\mathbf{Z}$ we use given time-series $\mathbf{X}$ can be defined as:

$$\mathbf{Z} = \{(X_{t-p}^{t-1}(i), X_t^{t+l-1}(i)) : 1 \leq i \leq m, p \leq t \leq n\}. \quad (1)$$

Using this set of sequence pairs $\mathbf{Z}$, we can train a predictor model to learn and predict the time-series $\mathbf{X}$. To ensure that the more varied nature of the data is captured, we utilize a more layered LSTM-based model than what has been used in other approaches [14], as shown in Figure 2. We use three layers of LSTMs, two with 128 units and one with 64 units, followed by three fully-connected (FC) layers that have 32, 8, and l units. The number of units in the last FC layer is controlled by the empirically determined best value of l, which we have determined to be 4. We have also determined the best value of p to be 30. These values have been determined to maximize anomaly detection performance.

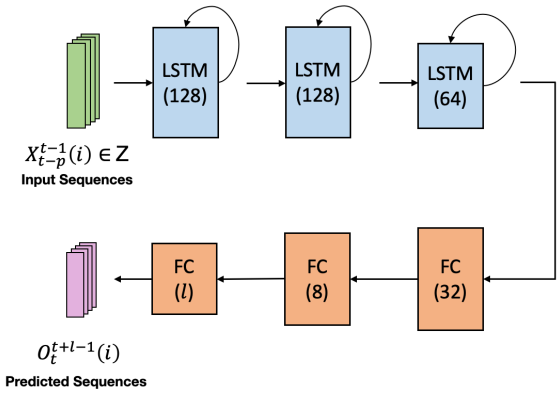

Figure 2: The architecture of our predictor model.

4.2. Transfer Learning

Suppose we have trained a neural network to replicate the mapping function $f : \mathcal{X}^p \rightarrow \mathcal{X}^l$, given $\mathbf{X}$. We could then finetune the network on each channel $X(i)$ in $\mathbf{X}$ to learn m different mapping functions f_i. However, if the model has learned the general mapping function f, finetuning it to learn a mapping f_i is trivial. Let us consider another multi-channel spacecraft telemetry time-series $\mathbf{Y} \in \mathcal{Y}^{u \times v}$. We will instead investigate the efficacy of applying and adapting the neural network to the data in $\mathbf{Y}$.

In essence, we are evaluating if knowledge of characteristics and patterns in one spacecraft system (source) can be used effectively to quickly learn information about a different, independent spacecraft system (target). The importance of the source dataset has been explored for time-series classification [8], with the conclusion being the source and target domains needed to be similar. Now, we investigate the feasibility of transferring between similar domains for time-series prediction.

Ideally, the mapping function f learned from $\mathbf{Z}$ would be the ideal mapping function $F : \mathbb{R}^p \rightarrow \mathbb{R}^l$, meaning it could be used to map $\mathcal{Y}^p \rightarrow \mathcal{Y}^l$ as well. However, this is usually not the case, so we use transfer learning principles to adapt the neural network to the data in $\mathbf{Y}$. Given $\mathbf{Y}$, we can, as in Equation 1, define u different training sets, one for each channel, to finetune the model trained on $\mathbf{Z}$:

$$\mathbf{W}_i = \{(Y_{t-p}^{t-1}(i), Y_t^{t+l-1}(i)) : 1 \leq i \leq u, p \leq t \leq v\}. \quad (2)$$

Let us look at the ith channel of $\mathbf{Y}$, $Y(i)$. As is the nature of finetuning, we inherit the trained weights from the original model trained on $\mathbf{Z}$ and update them by training on $\mathbf{W}_i$. We can control which parts of the network are updated during this finetuning process by freezing the parameters of certain layers at their starting values. This is usually done on the lower, or earlier, layers of the network [30]. Thus, we will also investigate the impact of freezing the LSTM layers of our network on its final anomaly detection performance.

The benefit of finetuning is that, if used appropriately, the neural network minimizes its loss function much quicker, meaning less resources are expended to generate a usable model, with the trade-off of a potential lower performance. When dealing with thousands of telemetry channels to monitor, the impact of lowering training time by a significant factor cannot be overstated.

4.3. Anomaly Extraction

The final stage of our anomaly detection system is anomaly extraction. At this point, the predictor models have been trained and finetuned on data that is assumed to be normal and not anomalous. Given an input sequence $Y_{t-p}^{t-1}(i)$ and our finetuned predictor model, we can output a predicted sequence for the next l values $(O_t, O_{t+1}, \cdots, O_{t+l-1})$, written in shorthand as O_t^{t+l-1}. The prediction error sequence in this scenario can be written as $E_t^{t+l-1}(i)$ or $(E_t(i), E_{t+1}(i), \cdots, E_{t+l-1}(i))$, where $E_j(i) = |O_j - Y_j(i)|$ and $t \leq j \leq t + l - 1$. Because our predictor model is based on the previous p values, abrupt changes in the input can lead to brief prediction errors independent of anomalous behavior. To account for this, we smooth the prediction error sequence $E_t^{t+l-1}(i)$ using a Gaussian filter, generating a smoothed prediction error sequence $G_t^{t+l-1}(i)$.

As shown by Li *et al.* [18], instead of directly thresholding this smoothed error sequence $G_t^{t+l-1}(i)$ to find anomalies, we first estimate the prediction error distribution and calculate the log-likelihood of each value in the prediction error sequence. Then we threshold the log-likelihood sequence to determine the anomalies.

Ideally, the prediction errors would fit a Gaussian-like probability distribution function. This is rarely true, so we use Kernel Density Estimation (KDE) to estimate the prediction error probability distribution function [26]. Let the unknown prediction error distribution for $Y(i)$ have a probability density function e_i and the collection of all smoothed prediction error values for $Y(i)$ be $G(i)$. We can then find an estimate probability density function $\hat{e}_i$:

$$\hat{e}_i^h(z) = \frac{1}{Nh} \sum_{j=1}^{N} K(\frac{z - G_j(i)}{h}), \quad (3)$$

where:

K is the kernel function; it needs to integrate to 1,
h is the bandwidth that affects degree of smoothing,
N is the total number of smoothed error values in $G(i)$.

The bandwidth h has a significant impact on the KDE process [4], and we have empirically determined 0.15 to be the best value to maximize anomaly detection. As the selection of the kernel function K is not as impactful, we use the Gaussian kernel for our experiments.

After the error probability distribution function is determined, we can estimate the log-likelihood $G_j(i)$ in $G(i)$, resulting in a log-likelihood sequence we will represent as $L(i)$. Note that, assuming our predictor model is trained correctly, anomalous behavior results in unexpected prediction error values, which meaning that the probability of those prediction errors occuring is low. These low probabilites translate to large negative log-likelihood values. This means we can threshold $L(i)$ to extract the anomalies. Simple thresholding across the entire sequence is not representative of local and/or contextual anomalies. Thus, we use the Dynamic Thresholding (DT) algorithm proposed by Li *et al.* [18].

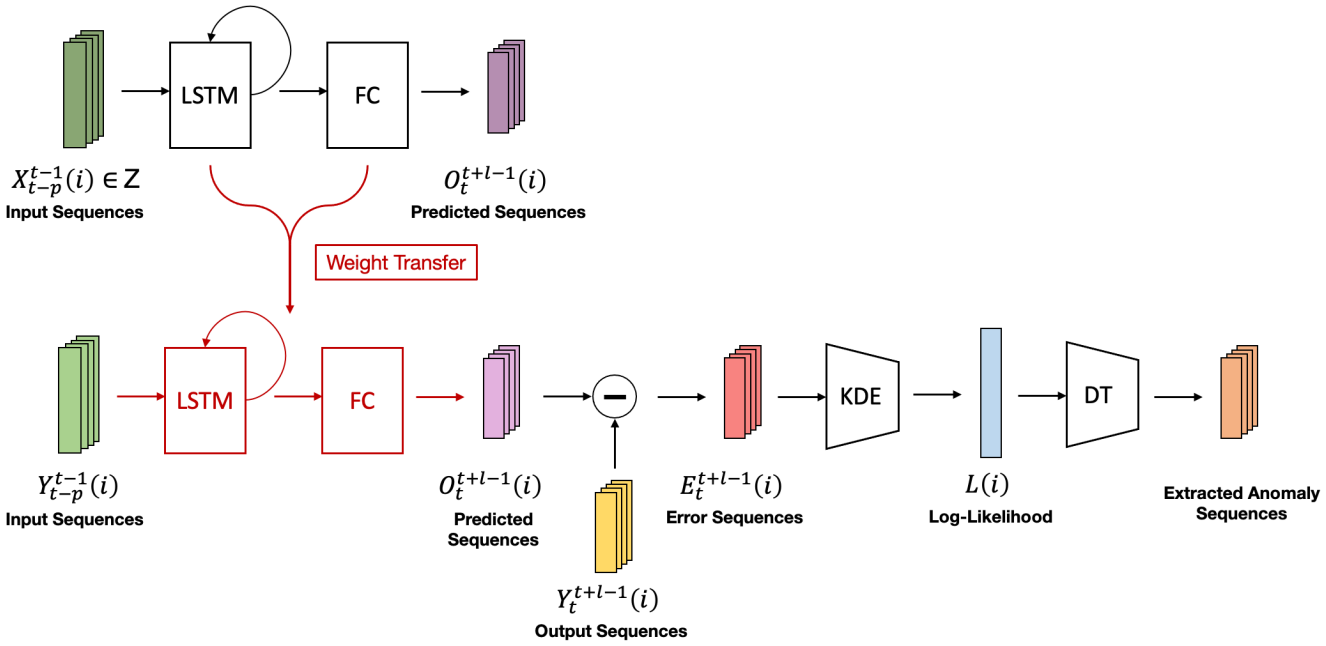

Figure 3: Our anomaly detection system, where KDE is Kernel Density Estimation and DT is Dynamic Thresholding. The weights of the general predictor model trained on $X(i)$ are transferred to the channel-specific models finetuned on $Y(i)$.

The main focus here is the search for the best threshold to use. To determine the best threshold T_b, we use the scoring function s for a given threshold T:

$$s(T) = \frac{P_{\text{vd}}}{1 + P_{\text{anom}} + w \cdot N_{\text{anom}}^2}, \tag{4}$$

where:

- P_{vd} is the percentage decrease of the variance of $G(i)$ when the anomalies identified by threshold T are removed,
- P_{anom} is the percentage of points that are anomalies identified by threshold T,
- N_{anom} is the number of anomaly sequences,
- w is a weight that controls the number of anomaly sequences.

All smoothed error points $G_j(i)$ with log-likelihoods $L_j(i)$ below threshold T are considered anomalous, as a lower log-likelihood means that the associated error did not fit well into the error probability distribution. The anomaly sequences, counted by N_{anom}, are found by using a dilation operation to connect anomaly points that are close to each other. This process can sometimes lead to extracting anomaly sequences of different lengths from the ground truth. We search for T_b by steps, starting from the maximum of $L(i)$ and ending at the minimum. The optimization goal is to find T that minimizes $s(T)$. The number of steps is a parameter that can be adjusted; we have used 25 steps in our experiments.

After evaluating discarded error sequences whose log-likelihood values are close to T_b, the collection of extracted anomalous sequences in time-series $Y(i)$ is finalized. Our entire anomaly detection system is shown in Figure 3.

5. Experiments

The architecture of the neural network model we use in our experiments is shown in Figure 2. First, there are three layers of LSTMs, two with 128 units and one with 64 units. They are followed by three FC layers that have 32, 8, and 4 neurons. The LSTMs use the hyperbolic tangent function as their activation function, and we use the sigmoid function for the FC layers to enforce non-linearity. Dropout [12] is used as a regularization technique to reduce overfitting by randomly zeroing units with a probability we set as 0.3. The Adam optimizer [15] was used to train the model, while mean squared error (MSE) loss was used as the loss function. We train our model for 200 epochs with an initial learning rate 0.001 that decays linearly to zero by the final epoch. To reduce the overall training time, an early stopping was applied when the validation loss did not decrease for more than 10 epochs. When finetuning the model, the initial learning rate was instead 0.0005. We have also determined empirically the values for length of input sequence $p = 30$ and length of output sequence $l = 4$.

As discussed before, in Equation 3 the kernel function K used is the Gaussian kernel, and we empirically determined the best value of bandwidth $h = 0.15$. We also determined the best value for $w = 0.5$ in the threshold scoring function $s(T)$, shown in Equation 4.

5.1. The MRO Experiment

We use a Mars Reconaissance Orbiter (MRO) [10] dataset, which consists of thousands of channels that encompass all spacecraft subsystems. Each channel is sampled only when there is a signal change, which means time between samples is irregular. Additionally, the data is completely unlabeled as locations of anomalous behavior are unknown. Since the data ends for each channel at a system reboot, we make the assumption that anomalous behavior occurs in proximity to the reboot. Therefore, our MRO training and validation data consists of the first half of every channel, split with a ratio of $4:1$. We normalize each channel to have values between 0 and 1 so the models are focused on learning the patterns of the data.

Recall that our approach involves training a general time-series predictor on one multi-channel spacecraft time-series $\mathbf{X}$ and finetuning for channels $Y(i)$ from another spacecraft time-series $\mathbf{Y}$. Since the MRO dataset consists of many unlabeled telemetry channels, it is ideal for training and evaluating our general time-series predictor. Let us take a subset of the MRO dataset containing $m = 50$ channels as $\mathbf{X}$. These channels were expertly identified as containing variability in amplitude, phase, and other signal characteristics to help generate a representative time-series dataset that a general predictor model could learn from. We train a general predictor model with the architecture discussed in Section 4.1 using the training set $\mathbf{Z}$, defined in Equation 1. We will refer to this trained general time-series predictor model as *MRONet*. We also train models with the same architecture on each channel $X(i)$ separately, using a training set similar to W_i in Equation 2, except with $X(i)$ instead of $Y(i)$. To evaluate the performance of MRONet, we can compare its average prediction RMSE for one time step ahead across all $m = 50$ channels with the average of the same for the $m = 50$ channel-specific models, as shown in Table 1.

Model	RMSE
MRONet	8.1%
Channel-Specific LSTMs	6.3%

Table 1: The average prediction error of MRONet vs. Channel-Specific LSTMs on the MRO data.

While the overall performance of MRONet has room for improvement, the prediction error is still relatively close to the one achieved by the channel-specific models, meaning, in this case, a small amount of channel-specific information is lost when gaining the capacity for a general prediction approach. This means MRONet must be capturing the general behavior of all $m = 50$ channels, making it a good candidate for our transfer learning experiment. The theoretical work done by Mariet and Kuznetsov [21] suggests that a potential area of improvement is to increase the number of channels m used during training. We can also explore other model structure archetypes, like an encoder-decoder, but must prioritize the ease of applying transfer learning principles to the model for the ultimate goal of anomaly detection.

5.2. Anomaly Detection Experiment

Hundman *et al.* [14] provided a publicly available spacecraft telemetry dataset that contains expert-labeled anomalous data. The dataset consists of telemetry data from the NASA Soil Moisture Active Passive (SMAP) satellite as well as the NASA Mars Science Laboratory (MSL) rover, Curiosity. It contains 54 and 27 channels of data for SMAP and MSL, respectively. This data has also already been split into training and testing sets. We further split the training data for each channel in a ratio of $4:1$ to obtain our training and validation data, respectively. We normalize this data to have values between 0 and 1 as well. Let us take this SMAP/MSL dataset with $u = 81$ channels as $\mathbf{Y}$.

Instead of training $u = 81$ channel-specific models from scratch, we can use the general time-series predictor model MRONet as discussed in Section 4.2. We explore two options for the MRONet finetuning process: (1) finetune the entire model (*MRONet-FT*) and (2) freeze the LSTM layers and finetune only the FC layers (*MRONet-FC*). Both approaches result in $u = 81$ channel-specific models that have been finetuned on the training sets $\mathbf{W_i}$, defined in Equation 2. These models are then inferenced on the provided test data, and the anomaly extraction process detailed in Section 4.3 is used to detect anomaly sequences.

As we are evaluating both performance and efficiency, we look at both the anomalies detected as well as the training time taken. Below, we define our anomaly detection performance criteria, identical to the rules used by Hundman *et al.* [14]:

1. We record a true positive (TP) if any detected anomaly sequence overlaps with a true anomaly sequence. If multiple detected anomalies overlap with the same true anomaly sequence, only one true positive is recorded.

2. We record a false positive (FP) if a detected anomaly sequence has no overlap with a true anomaly sequence.

3. We record a false negative (FN) if a true anomaly sequence has no overlap with a detected anomaly sequence.

Method	SMAP		MSL		Total	
	Recall	Precision	Recall	Precision	Recall	Precision
MRONet	0.833	0.487	0.806	0.521	0.825	0.497
MRONet-FC	0.712	0.770	0.742	0.561	0.722	0.686
MRONet-FT	0.773	0.797	0.774	0.706	0.774	0.765
Channel-Specific LSTMs	0.833	0.809	0.839	0.765	0.835	0.794
TadGAN [9]	0.835	0.523	0.694	0.490	0.786	0.513
Hundman *et al*. [14]	0.855	0.855	0.694	0.926	0.800	0.875
StackedPredictor [18]	0.913	0.900	0.861	0.816	0.895	0.870

Table 2: The recall and precision of anomaly detection for our models on the SMAP/MSL dataset provided by Hundman *et al*. [14]. We also show the results of three other approaches from the literature: the results of an LSTM GAN-based approach called "TadGAN" [9], the LSTM model from Hundman *et al*. [14], and a combination of LSTMs and an SVM called "StackedPredictor" [18]

In the future, a more sophisticated approach can be employed, where the degree of overlap (or lack thereof) is factored into the rules, enabling evaluation of "close" misses or "lucky" detections [16]. However, with this relatively simpler criteria, we now define our primary metrics for evaluating anomaly detection performance:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{6}$$

A high precision means that the model is not prone to flagging false positive sequences. This can be made a priority because new adopters of technology are often deterred by misleading results that require attention [14]. Recall can also be prioritized, as a high recall means the number of true anomalies going undetected is low. Depending on the operating situation, a prioritization decision has to be made. We show the precision and recall values obtained by our models on the SMAP/MSL dataset in Table 2.

In terms of degree of finetuning on the SMAP/MSL dataset for the versions of the MRONet model, we can order them as MRONet (least/none) $\rightarrow$ MRONet-FC $\rightarrow$ MRONet-FT (most). We can see this progression in similar ways for SMAP and MSL. MRONet flags many sequences in the data, meaning it identifies a lot of true anomalies, but also detects a large number of false alarms. This is expected, as the model has not seen any of the data in the SMAP/MSL dataset. As more of MRONet is finetuned on the data, the overall performance improves. While the recall drops slightly as the model misses some anomalies, the precision greatly improves with the degree of finetuning as the model learns more about the specific channel data. This means a small number of true anomalies are missed to drastically drop the number of false anomalies, which is an acceptable outcome. We also note that freezing the LSTM layers seems to limit the MRONet model performance in terms of anomaly detection. Since the LSTMs capture patterns in the input data, this limitation is expected and is compliant with transfer learning literature [30]. However, freezing certain LSTM layers while leaving others for finetuning could be a viable strategy if time is an important constraint.

We can see two anomaly detection results from MRONet-FT in Figures 4 and 5. Figure 4 shows the anomaly detected by MRONet-FT on channel A-4 from the SMAP data. The top row shows the ground truth, or where the anomaly is actually located. The bottom row shows the detected anomaly sequence, which we can see encompasses the area the ground truth is actually located. In Figure 5, we see the results on channel F-7 from the MSL data. The layout of the figure is the same, with the different colors representing different anomalies, and we can see that the true anomalies are all detected. However, the model also flags a normal sequence early in the data. The difference in length of some of the detected and actual anomalies indicates a discrepancy in the specific samples identified as part of the anomalous sequence (as discussed in Section 4.3).

Overall, it seems that finetuning MRONet allows for comparable anomaly detection performance to equivalent channel-specific models trained from scratch, as well as other approaches. The strength of the finetuning approach comes from the time taken to complete its training satisfactorily. We report the average training time taken to achieve the best model for each channel of the SMAP/MSL dataset in Table 3. Recall that an early stopping criteria was used for all models to reduce overall training time.

From Table 3, we can see that the MRONet anomaly detection performances in Table 2 are achieved in a fraction of the time taken to train equivalent channel-specific models from scratch, let alone the time taken by the high-performing StackedPredictor [18]. This shows us that it is indeed possible to use knowledge about one spacecraft sys-

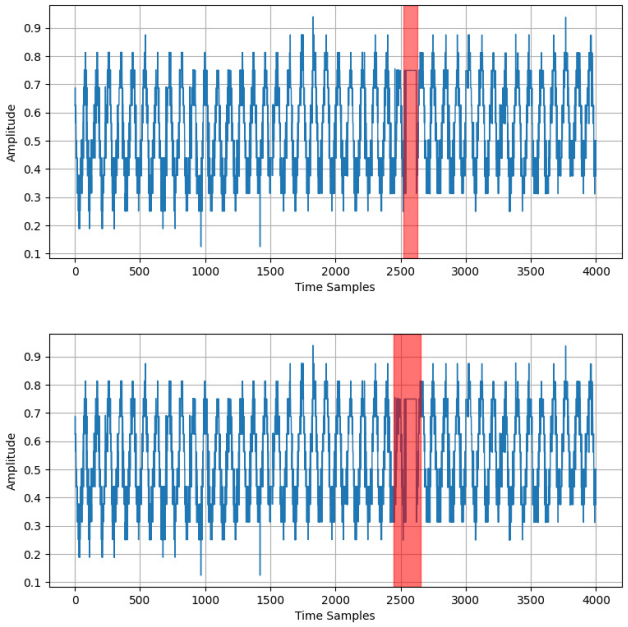

Figure 4: The result of MRONet-FT on channel A-4 from the SMAP data. The top row shows the actual anomaly location and the bottom row shows the detected anomaly sequence.

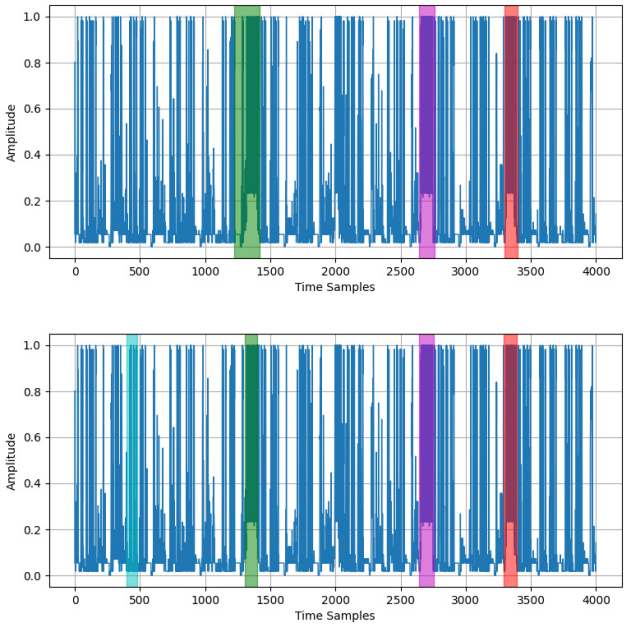

Figure 5: The result of MRONet-FT on channel F-7 from the MSL data. The top row shows the actual anomaly locations and the bottom row shows the detected anomaly sequences, including a false positive.

Model	Time (seconds)
MRONet-FC	13.35
MRONet-FT	34.41
Channel-Specific LSTMs	71.45
StackedPredictor [18]	427.5

Table 3: The average training time to achieve the best model for each of the $u = 81$ channels of the SMAP/MSL dataset.

tem to quickly learn information about another spacecraft system. While performance should always be the priority, this tradeoff between performance, adaptability, and training time allows for some possible applications. For example, an ensemble of MRONet models can be trained quickly to further improve anomaly detection performance, as ensembles tend to be more robust [7]. Another option is to use an MRONet model to quickly adapt to unseen telemetry data onboard a spacecraft in real-time, or at least until a more permanent solution is made available.

6. Conclusions

It is vital to observe spacecraft systems for anomalous behavior to monitor for potential issues. With thousands of channels to evaluate, optimizing unique models for each channel can be tricky and time-consuming. In this paper, we present an approach to train a general time-series predictor model and quickly adapt it to detect anomalies in any specific telemetry channel, dramatically reducing the number of truly unique models. The anomaly detection performance of our finetuned models is comparable to that of similar models trained from scratch on the same data, as well as previous approaches to anomaly detection. We also show that it is feasible to take knowledge learned from one spacecraft system and use it to quickly learn information about another system, as our finetuned models reach their peak performance in a fraction of the time taken by normal approaches. Future work includes establishing a measure of dataset similarity for anomaly detection and prediction, as well as investigating attention and Transformers [27] as an option to further highlight patterns in time-series data. We will also explore the feasibility of using these models in an online learning scenario.

7. Acknowledgements

This material is based on research sponsored by Lockheed Martin Space. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Lockheed Martin Space.

References

[1] Enyu Cai, Sriram Baireddy, Changye Yang, Melba Crawford, and Edward J. Delp. Deep Transfer Learning for Plant Center Localization. *Proceedings of the IEEE Conference for Computer Vision and Pattern Recognition Workshops*, pages 277–284, June 2020. Seattle, WA. 2

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3), July 2009. 2, 3

[3] Sucheta Chauhan and Lovekesh Vig. Anomaly Detection in ECG Time Signals via Deep Long-Short Term Memory Networks. *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics*, pages 1–7, October 2015. Paris, France. 1, 3

[4] Su Chen. Optimal Bandwidth Selection for Kernel Density Functionals Estimation. *Journal of Probability and Statistics*, 2015:1–21, 2015. 4

[5] Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. Multi-Source Cross-Lingual Model Transfer: Learning What to Share. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, July 2019. 2

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, October 2014. Doha, Qatar. 2

[7] Thomas G. Dietterich. Ensemble Methods in Machine Learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, June 2000. Cagliari, Italy. 8

[8] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer Learning for Time Series Classification. *Proceedings of the IEEE International Conference on Big Data*, pages 1367–1376, December 2018. Seattle, WA. 4

[9] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks. *arXiv preprint arXiv:2009.07769*, November 2020. 3, 7

[10] James Graf, Richard Zurek, Ross Jones, Howard Eisen, M. Dan Johnston, Ben Jai, and Bill Mateer. An Overview of the Mars Reconnaissance Orbiter Mission. *Proceedings of the IEEE Aerospace Conference*, pages 1–171–1–180, March 2002. Big Sky, MT. 6

[11] Paul Hayton, Simukai Utete, Dennis King, Steve King, Paul Anuzis, and Lionel Tarrasenko. Static and Dynamic Novelty Detection Methods for Jet Engine Health Monitoring. *Philosophical Transactions of the Royal Society A*, 365:493–514, 2006. 2

[12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580*, July 2012. 5

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. 2

[14] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 387–395, August 2018. London, United Kingdom. 1, 2, 3, 6, 7

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference for Learning Representations*, May 2015. San Diego, California. 5

[16] Alexander Lavin and Subutai Ahmad. Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. *Proceedings of the 14th IEEE International Conference on Machine Learning and Applications*, pages 38–44, December 2015. Miami, FL. 7

[17] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. *Proceedings of the 28th International Conference on Artificial Neural Networks*, pages 703–716, September 2019. Munich, Germany. 3

[18] Tianyu Li, Sundip R. Desai, James L. Mathieson, Richard H. Foster, Moses W. Chan, Mary L. Comer, and Edward J. Delp. A Stacked Predictor and Dynamic Thresholding Algorithm for Anomaly Detection in Spacecraft. *Proceedings of the IEEE Military Communications Conference*, pages 165–170, November 2019. Norfolk, VA. 3, 4, 7, 8

[19] Pankaj Malhotra, Anushka Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. LSTM-Based Encoder-Decoder for Multi-Sensor Anomaly Detection. *arXiv preprint arXiv:1607.00148*, July 2016. 1, 3

[20] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long Short Term Memory Networks for Anomaly Detection in Time Series. *Proceedings of the 23rd European Symposium on Artificial Neural Networks*, April 2015. Bruges, Belgium. 2, 3

[21] Zelda Mariet and Vitaly Kuznetsov. Foundations of Sequence-to-Sequence Modeling for Time Series. *Proceedings of the 22nd International Conference on Artifical Intelligence and Statistics*, pages 408–417, April 2019. Naha, Japan. 3, 6

[22] Anvard Nanduri and Lance Sherry. Anomaly Detection in Aircraft Data Using Recurrent Neural Networks (RNN). *Proceedings of Integrated Communications Navigation and Surveillance*, pages 5C2–1–5C2–8, April 2016. Herndon, VA. 1, 3

[23] Razcan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages 1310—-1318, June 2013. Atlanta, GA. 2

[24] Shahadate Rezvy, Tahmina Zebin, Barbara Braden, Wei Pang, Stephen Taylor, and Xiaohong W. Gao. Transfer

Learning For Endoscopy Disease Detection & Segmentation With Mask-RCNN Benchmark Architecture. *Proceedings of the 2nd International Workshop and Challenge on Computer Vision in Endoscopy*, pages 68–72, April 2020. Iowa City, IA. 2

[25] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323(6088):533–536, October 1986. 2

[26] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*, volume 26. CRC Press, Boca Raton, FL, 1986. 4

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, December 2017. Long Beach, CA. 8

[28] Vincent Vecruyssen, Wannes Meert, and Jesse Davis. Transfer Learning for Time Series Anomaly Detection. *Proceedings of the European Conference on Machine Learning & Principles of Knowledge Discovery in Databases, Workshop and Tutorial on Interactive Adaptive Learning*, pages 27–37, September 2017. Skopje, Macedonia. 2

[29] Tailai Wen and Roy Keyes. Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning. *arXiv preprint arXiv:1905.13628*, May 2019. 2

[30] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable Are Features in Deep Neural Networks? *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, December 2014. Montréal, Canada. 2, 4, 7

[31] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. One-Class Adversarial Nets for Fraud Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1286–1293, July 2019. Honolulu, HI. 2

[32] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *arXiv preprint arXiv:1911.02685*, June 2020. 2