Symbolic Regression for Regularization and Model Discovery with Transformer Shallow Recurrent Decoders (T-SHRED)

Alexey Yermakov[1,3*], David Zoro[1*], Mars Gao[2], J. Nathan Kutz[1,3 *]

[1] Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195, United States
[2] Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195, United States
[3] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, United States

Abstract

SHallow REcurrent Decoders (SHRED) have been shown to excel at system identification and forecasting from sparse sensor measurements. Previous uses of SHRED-based models relied on using Recurrent Neural Networks (RNNs) and simple Multi-Layer Perceptrons for the encoder and decoder. Despite the relatively simple backbone of these SHRED models, they are able to predict chaotic dynamical systems on different physical, spatial, and temporal scales directly from data. In this work, we leverage modern deep learning tools to further improve upon SHRED. In particular, we study the effects that using a Transformer for the encoder and a UNet for a decoder has on the performance of SHRED (T-SHRED) on next-step state prediction from a sparse set of randomly placed sensors. We also introduce Multi-Head SINDy Attention (MHSyA) into T-SHRED to perform symbolic regression directly on the latent space as part of the model architecture. We analyze the performance of T-SHRED on three different dynamical systems ranging from the low-data to the high-data regime. We observe that T-SHRED with MHSyA is both interpretable and outperforms other SHRED models on large datasets.

Keywords: SHRED, SINDy, Symbolic Regression, Forecasting, Dynamical Systems, Transformers, Deep learning

1 Introduction

The SHRED neural network architecture is demonstrably useful in scientific and engineering applications. Based on the separation of variables technique for solving partial differential equations, the model can be seen as a joint training of learning the temporal trajectory and spatial field of the input data simultaneously through the encoder and decoder respectively [1]. Several works have used SHRED to perform state space reconstruction from a sparse set of sensors in the spatial dimension [1–3]. These works have shown that a full state reconstruction of the data can be obtained from sparse sensors where the sensors only cover about 0.5% of the original space. Furthermore, the SHRED models are agnostic to the specific system they are modeling. They can perform Go-Pro physics, where dynamics are learned directly from video [2, 3]; they can also learn chaotic fluid dynamics from simulations [2, 3]. These results encourage the further development of SHRED architectures.

While it is clear that the theoretical motivation for SHRED models has led to promising results, previous work has not implemented modern deep learning advancements into the architecture. A significant amount of modern deep learning architectures in every domain use transformers. First used for machine translation [4], they utilize a self-attention mechanism to learn complex patterns from large datasets more efficiently than other models available at the time. Since then, significant empirical evidence has been produced showing that transformer-based models scale exceptionally well with more data in a variety of domains and architectures [5–7].

In this paper, we introduce Transformer SHRED (T-SHRED) which uses a transformer backbone within the SHRED architecture for sparse sensor modeling. We also add symbolic regression to the individual heads of the transformer to force the model to learn interpretable dynamics in the latent space using SINDy. The result, is a Multi-Head SINDy Attention (MHSyA) T-SHRED

*co-first authors

model which we compare with other SHRED architectures. We do a comparative study by evaluating the performance of each architecture in next-step full state prediction from a sparse set of measurements. We also study the effect SINDy loss [3] has on our prediction task for each model.

Next-step state prediction is performed on three different datasets, all coming from physical phenomena on different scales and with different dataset sizes. We show that as the amount of data used in training SHRED increases, transformer based models outperform traditional Recurrent Neural Network (RNN) SHRED architectures on this task.

This task is performed on three different datasets, all coming from physical phenomena on different scales and with different dataset sizes. We show that as the amount of data used in SHRED increases, transformer-based models outperform traditional Recurrent Neural Network (RNN) SHRED architectures on this task. We also demonstrate that MHSyA does not harm the performance of the model.

2 Latent Space Symbolic Regression

Here we outline the use of symbolic regression in the latent space of T-SHRED. First, we motivate the use of symbolic regression into the transformer heads. Then, we demonstrate its effectiveness compared to other SHRED models on various datasets. Finally, we show some examples of dynamics that the heads are learning.

2.1 Motivation

While the widespread adoption of black box models in machine learning has led to significant improvements in predictive performance, they also created a lack of transparency and interpretability, making it difficult for users to understand what the models are learning. In response, the literature has seen an increase in alternative approaches which focus on developing interpretable models by fitting data directly to understandable equations. One of the most notable of these approaches is SINDy (the Sparse Identification of Nonlinear Dynamical systems) [citation], which performs sparse symbolic regression on a library set of functions to find the governing equations of a dynamical system. SINDy has expanded into a subfield where a variety of techniques have been applied to solve domain specific problems [citation]. Motivated by these approaches, we adjust the self-attention mechanism in transformers to force each head of the self-attention mechanics to learn the dynamics of the latent space.

The MHSyA transformer is a parameterized function of the form $f_\theta(\mathbf{x}) = \mathbf{z} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ with learnable parameters θ for $\mathbf{x} \in \mathbb{R}^{n\times d}$. Let $H \in \mathbb{N}$ be the number of heads in the transformer such that $\exists k, H \cdot k = d$. Then, the transformer performs the following operation:

$$Q^{(h)}(\mathbf{x}) = \mathbf{x}W_{h,q}, \quad K^{(h)}(\mathbf{x}) = \mathbf{x}W_{h,k}, \quad V^{(h)}(\mathbf{x}) = \mathbf{x}W_{h,v}, \qquad W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{d\times k} \tag{1}$$

$$S^{(h)\prime} = \text{rowsoftmax}\left(\frac{Q^{(h)}(\mathbf{x})K^{(h)}(\mathbf{x})^T}{\sqrt{k}}\right) V^{(h)}(\mathbf{x}) \tag{2}$$

$$S^{(h)} = \Theta\left(S^{(h)\prime}\right)\Xi^{(h)}, \qquad \Xi^{(h)} \in \mathbf{R}^{k\times m} \tag{3}$$

$$S = \text{concatenate}(S^{(1)}, \ldots, S^{(H)}) \tag{4}$$

$$\mathbf{z} = (S\ W_{ff_1})W_{ff_2}, \qquad W_{ff_2}^T, W_{ff_1} \in \mathbb{R}^{d\times m} \tag{5}$$

$$\tag{6}$$

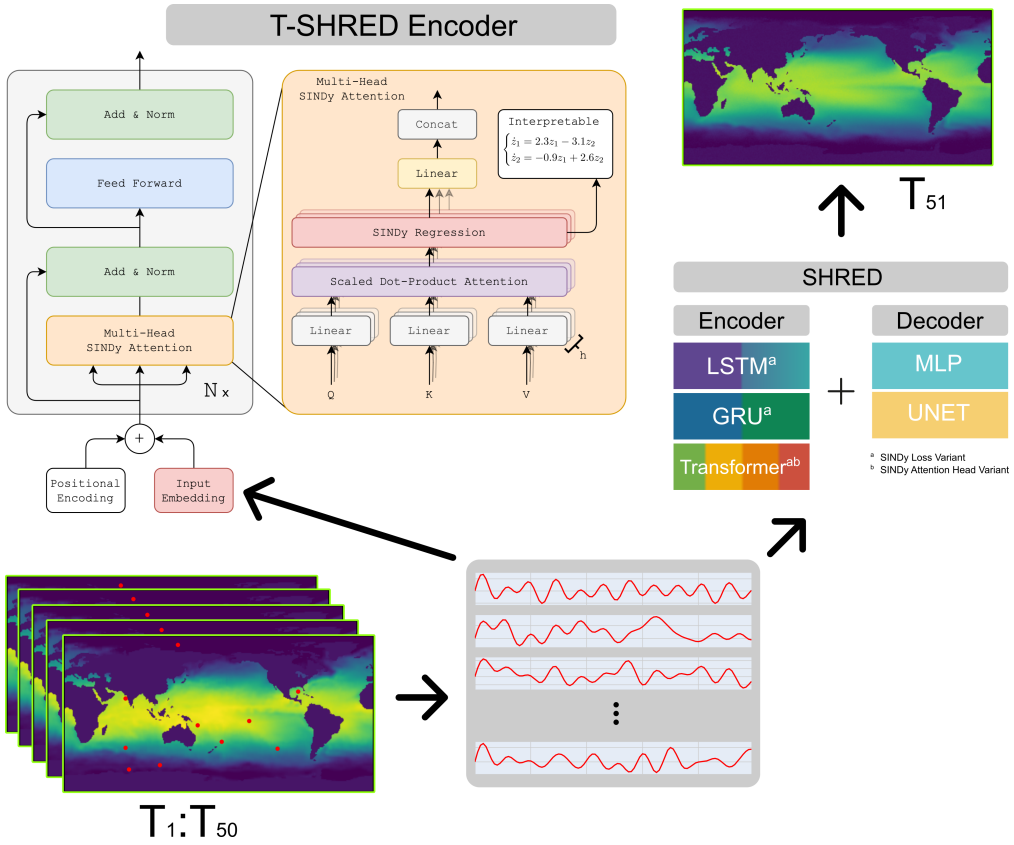
The standard multi-head attention block is modified to do sparse regression on the latent space of each head, as seen in Equation 3. The learnable parameters θ are the weight matrices W as well as the coefficients for the library generated from Θ: $\Xi^{(h)}$.

3 David stuff

The initial motivation for this work was to provide an improved solution to the problem of state estimation/identification and next-step full-state prediction from sparse measurements [8]. In consequence, this work introduces Transformer SINDy-SHRED. Transformer SINDy-SHRED is an upgrade from SHRED and SINDy-SHRED. Transformer SINDy-SHRED combines both a transformer encoder and the SINDy attention head. Given that transformers are a well-studied architecture and that they address certain limitations of RNNs, it is clear that Transformer SINDy-SHRED is a natural progression to SHRED and SINDy-SHRED.

This work has several objectives. The first objective is to introduce Transformer SINDy-SHRED because it lays the foundation for the rest of the paper. The second objective is to demonstrate Transformer SINDy-SHRED is an upgrade from SHRED because it establishes a progression in the SHRED concept. The third objective is to demonstrate that Transformer SINDy-SHRED is a solution to the problem of state estimation/identification and next-step full state prediction from sparse measurements because the architecture is an improved solution to the main problem. The fourth objective is to establish the Transformer SINDy-SHRED architecture as comparatively competitive, and generalizable as well as provide baseline metrics for one to extrapolate its performance because it aids interpretability of the work and reproducibility.

This paper has several sections. The background section will explain the SHRED and SINDy-SHRED architecture. Beyond explaining the architectures, it will also provide their limitations, what was done in the prior work, and what else can be improved. The Transformer SHRED section will describe the various models we're testing on; this section will explain the theoretical basis for these models and their architectures and the details required for the rest of the paper. The computational experiments section will provide additional details on the models tested, the methods used to test them, and some preliminary results. The results section will provide the results of the paper and explain what they mean. Afterward, the paper will be concluded, and some brief acknowledgments will be made. In our conclusion, the results will be very well appreciated and reinforce the introduction, conception, and claims made regarding Transformer SINDy-SHRED.



(a)

Figure 1: Illustration of the SHRED architecture. The SHRED architecture takes a time-series of sparse sensor measurements (red) from a state space and outputs the next-step full state prediction. The SHRED architecture consists of an encoder and a decoder, optionally with SINDy Loss during training and SINDy attention for the transformer encoder. The T-SHRED architecture is a SHRED model with a transformer encoder and either an MLP or UNET decoder. The T-SHRED encoder is the transformer architecture modified to include a SINDy Regression layer in the self-attention heads. This forces the model to learn an interpretable model of the dynamics of the latent space. Each head is capable of learning a separate ODE in the latent space.

4 Background

4.1 SHRED

Shallow recurrent decoding networks are a technique that utilizes recurrent neural networks to estimate high dimensional states from limited measurements [3]. Shallow recurrent decoder networks merge an LSTM and a Shallow Decoder Network architecture. The development of SHRED was to address the issue of system identification and forecasting from sparse sensor measurements [8]. The SHRED architecture relies on LSTMs to capture temporal dependencies. The LSTM encoder outputs a latent representation, which is decoded to produce a high dimensional state. The SHRED architecture is based on the separation of variables technique [8]. The separation of variables technique assumes that a solution can be separated as a product of spatial and temporal functions u(x, t) = T(t)* X(x) [8]. The solution reduces the partial differential equation into two separate ordinary differential equations: one equation representing time and the other space. The SHRED architecture can be written as

$$H\{y_i\}_{i=t-k}^{t} = \mathcal{F}(\mathcal{G}(\{y_i\}_{i=t-k}^{t}; W_{RN}); W_{SD}) \tag{7}$$

SHRED has several advantages: It only requires limited sensors as one of them. Second,

SHRED does not require grid-like data collection. However, SHRED does have a limitation that is directly applicable to the motivation for producing this paper. SHRED utilizes RNNs. In consequence, it can have difficulty producing strong results for forecasting or reconstruction due to the vanishing-gradient problem.

4.2 SINDy-SHRED

SINDy-SHRED further exploits the latent space for sparse sensor modeling. Furthermore, SINDy-SHRED enforces model interpretability using a SINDy-based functional class [3]. SINDy-SHRED provides a joint discovery of the coordinates and governing equation [3]. Similarly, SINDy-SHRED accomplishes this by enforcing that the latent state of the recurrent network follows an ODE in the SINDy class of functions [3].

5 Transformer SHRED

5.1 Sequence modeling with transformers

Connection to SINDy-SHRED

Using GRU for sequence modeling with shallow decoder networks

Using GRU for sequence modeling with U-Net

Using transformer for sequence modeling with shallow decoder networks

Using transformer for sequence modeling with U-Net Transformers provide a powerful alternative for sequence modeling [9–12], which is much faster and computationally efficient [4] from parallel processing. The attention mechanism in transformers capture the local features better and reduce the reliance on external information. With positional encoding, the model further incorporates the order of the sequence, which is crucial for time series applications.

Where are transformers currently applied? Transformers are a neural network model that can learn representations of sequences or sets of data points. Transformers have driven advances in natural language processing, computer vision, and spatial-temporal modeling [13]. For illustration, they have become very prevalent and well-known in natural language processing. Transformers are useful because, unlike RNNs, transformers can handle larger input sequences. It is said that transformers have become the predominant infrastructure and state-of-the-art solution for the majority of learning-based machine intelligence tasks in the field of artificial intelligence [14]. The entire transformer architecture is not used. Instead, the transformer encoder is utilized [13].

What composes the architecture of a transformer? What are the benefits of utilizing transformers? Transformers address the limitations of preexisting neural network architectures. They can train faster because they utilize parallel processing. Similarly, transformers address the vanishing gradient problem, which RNNs have to deal with. In consequence, the transformer can utilize longer sequences without reduction in training performance. Similarly, as the objective function is traversed the gradient will not converge too early. Additionally, in biomedical imaging, the transformer can alleviate the problem of scattered target regions and large shape differences [15]. Similarly, the transformer can use a convolutional neural network structure to take advantage of the ability of sample information to extract multiscale local spatial features; this would in turn improve model performance by allowing the model's global and local information achieve a balance [15].

What are the limitations of the transformer architecture? Transformer architectures do have their benefits that make them very clear to utilize them to build upon SHRED-TUN. However, a limitation of transformers it that they hallucinate. It is said that a major contribution of this has to do with function composition [16]. The limitations take place primarily for very large inputs [16].

As an illustration, the issue of function composition can be explained as such. If you have a sentence, "Joe's brother is going for a walk", brother relies on the fact that we are referring to Joe. In the context of spatio-temporal data, a hallucination would be a vastly inaccurate forecasting of the time series that does not match the established ground truth Y. Despite these limitations, the fact that transformers address the problems of RNNs make them a strong proponent for utilization in SHRED-TUN. Why are we using a transformer encoder? The point of this SHRED adaptation (SHRED-TUN) is to be able to utilize the encoder component of the transformer architecture to capture long-term dependencies in the time series data. Theoretically, this would make SHRED more generalizable to different problems and datasets where the temporal connections are longer and more complex. SHRED-TUN is capable of both forecasting and reconstruction. A transformer encoder allows the whole sequence to be processed in parallel, which improves the training time of the model. The original SHRED utilizes an RNN encoder. The typical problems that RNNs face could be resolved using a transformer encoder. The reason for this is that transformers resolve the typical issues RNNs face. This is relevant because

How does this deviate from SHRED and what is the theoretical result? In theory, SHRED-TUN should perform better than SHRED because of its ability to capture long-term dependencies / longer sequences in the data. In this case, better is more generalizable and precise (metric-wise). Moreover, the utilization of a transformer encoder provides more accurate interpretability than SHRED. The reason for the improved interpretability comes from the attention-heads of the model. Moreover, this paper will go through [16] several potential applications of SHRED-TUN that highlight the added interpretability and generalization of the model. Furthermore, implementing a transformer encoder with SHRED allows the model to deal with larger datasets more effectively. In principle, the model can scale; more data reduces the model's chance to overfit. Consequently, the interpretability and reliability of the model improve. Similarly, the model does not have to deal with the vanishing gradient problem that RNNs and LSTMs have to deal with because they can look at the whole sequence once due to their self-attention mechanism. Convolutional neural networks were developed to address certain problems that MLPs faced when dealing with data that possessed many hierarchical features. Additionally, image data for example possess a large number of pixels, which leads to a significant number of parameters in MLPs that make them computationally expensive and prone to overfitting. The CNN is a solution to the limitations of the MLP network when dealing with such data. CNNs have pooling layers that reduce the dimensionality of the data making it less computationally expensive. Similar to how the development of transformers addressed certain limitations facing RNNs, the development of CNNs addressed the limitations of RNNs in handling data that would otherwise produce computationally expensive models. UNETs are an architecture that was originally developed for biomedical imaging. Their architecture utilizes skip connections, which has made them useful for tasks where the inputs and outputs have the same shape. UNets and convolutional decoding have strong evidence in the literature for the ability to extract features and reconstruct spatial information. UNets have been used in a variety of contexts, such as image segmentation and signal/image reconstruction. In this paper, the utilization of UNETs provides the opportunity for SHRED to extract local features better. UNETs rely on convolutional neural networks as an earlier foundation. In consequence, they are still designed to deal with the limitations of MLPs. The original SHRED architecture utilized an MLP decoder. In consequence, while effective, it demonstrates that the utilization of UNETs/CNNs in the SHRED-TUN architecture is reasonable.

What challenges do UNETs overcome? Before UNETs, there were other traditional approaches to image segmentation in medical imaging. UNETs were designed to overcome the limitations of the traditional approaches in image segmentation. For instance, UNETs can segment from input-output pairs without user annotation. User annotation implies sketching image boundaries by hand, which is labor and time-intensive. Additionally, the fact that the UNET operates only on convolutional layers permits it to work on images of any size. The encoding path and decoding path of the architecture permit the UNet to learn both global context and local information. Thus, the UNET architecture is very useful for dealing with reconstruction tasks that require a model to have

the capability to learn different levels of visual information (colors, objects, etc) similar to a spatial-exploration-based CNN. The purpose of utilizing UNETs in SHRED is to help the model learn spatial relationships. UNETs excel at reconstruction in various areas. However, this critical aspect is advantageous for dealing with spatial and temporal data in a hybrid model such as SHRED-TUN (Shallow Recurrent Decoding with Transformer Networks and UNETs). SHRED deals with tasks that require a model to learn both local and global information. Similarly, learning this global features is not sufficient enough. The model must be able to accurately retain the information with minimal loss to properly reconstruct the data. Similarly, a well-learned model can better forecast the spatio-temporal data at the respective time steps necessary.

What is the UNet architecture? UNETs consist of two paths as a basic structure. The first path is the contracting path. The first path provides classification information, and it is also known as the encoder path. It decreases the spatial dimensions of the images while capturing critical information about the image. The contracting path utilizes pooling and convolution layers. The second path is known as the decoder path. The second path consists of up-sampling and concatenations from the contracting path followed by convolutions. The expansion in the second path allows the network to learn localized classification information. Moreover, the upsampling allows the model to recover spatial information. Where are UNETs applied? As stated previously, UNETs have significant applications in medical imaging. This is due to the recent advances in computer vision within the last decade (U-NETS and its variants for medical imaging). UNETs are very strong with localization, and UNETs are useful in preserving fine details. This is helpful in a variety of contexts such as recording EEG signals where fine details are pertinent or reconstructing images extracted from signal measurements. Additionally, U-NETs are very useful to medical imaging because they create highly detailed segmentation with very limited training samples (U-NET and 6 its variants for medical image segmentation: theory and applications). Another benefit is that U-NETs are faster to train than other segmentation methods (U-NET and its variants for medical image segmentation: Theory and applications). Consequently, UNETs/CNNs seemed to be an adequate choice for our decoder network.

How do UNETs work and why use a UNET decoder? UNETs can be seen as an encoder network followed by a decoder network in simple terms. The use of a UNET decoder is beneficial. The reason why a UNET decoder is helpful is that it can recover the spatial relationships from the data rigorously. For emphasis, the decoder would be useful in capturing the intricacies of the local and global context of the spatiotemporal data. By using a UNET decoder, SHRED can deal with more complex data and problems. Similarly, SHRED can improve its reconstruction capacity significantly. This is essential because the original uses of SHRED are related to reconstruction and forecasting problems. SHRED-TUN as an extension of SHRED should be able to deal with the problems that SHRED did, but it should be able to do so with significantly more precision. Additionally, the ability of UNETs to enhance the precision of segmentation or reconstruction tasks with less data makes it an ideal choice for our network. SHRED-TUN has many potential uses. This can be seen in the applications section of this paper. The combination of utilizing transformers and UNETs presents an excellent opportunity for better interpretability and generalizability to many tasks. SHRED-TUN utilizes a transformer encoder and UNET decoder. The UNET decoder is a feature that utilizes hyperparameter tuning. In other words, it could be a simple CNN; it depends on the dataset. In the case of this paper, the UNET-Decoder is a width-based CNN where the feature maps increase in size at each layer. Additionally, the decoder utilizes same padding across each layer. SHRED-TUN takes advantage of the transformers resolution of problems regarding RNNs. Furthermore, SHRED-TUN utilizes the ability of CNNs and UNETs to extract spatial features. SHRED-TUN also deviates from SHRED in that convolutional neural networks were specifically designed to deal with spatial data such as images and such. Thus, the SHRED-TUN model theoretically should be an improvement from SHRED when dealing with data of that context. Moreover, the one-dimensional component of the convolutional layers are utilized in the process are effective at learning temporal relationships in the data. In this case, their goal is to decode the spatio-temporal relationships being encoded by the convolutional neural network.

Since the transformer deals with the vanishing gradient problem, SHRED-TUN can utilize long sequences and more data to provide more effective generalizations to many problems.

SHRED-TUN has proven very precise in a myriad of problems/contexts. In similar consideration, the capability of SHRED- TUN to resolve the vanishing gradient problem seen in SHRED, allows additional information to be retained in the model. Thus, the reconstruction precision is improved. Since UNETs/CNNs can do more with less data and transformers allow one to increase the amount of data passed to SHRED with less consequences. The combination of these two powerful tools provides a relevant jump from SHRED. Since SHRED-TUN utilizes a transformer encoder, it produces greater interpretability than SHRED. The significance of this is that it could produce better insights into the patterns for the data it is trained on. SHRED-TUN can be applied in reconstruction tasks. For instance, the utilization of a UNET in the SHRED-TUN architecture gives it the possibility to effectively reconstruct spatio-temporal systems that require the model to learn both global and local information. A possible application of SHRED-TUN would be a biomedical application given that it utilizes a UNET decoder. SHRED-TUN could be utilized to help decode MEG signals. Moreover, SHRED-TUN could be utilized to reconstruct relevant features coming from biomedical signals or images. Similarly, SHRED-TUN can be used to solve various problems in physics and engineering. An example of this would be in plasma physics. This work will demonstrate that SHRED-TUN can be effectively utilized in forecasting and reconstructing plasma signals via multiple sensors. Similarly, SHRED-TUN performs relatively well when reconstructing plasma signals as well. In consequence, this is a promising application area for SHRED-TUN.

For a formulaic representation of the SHRED-TUN architecture referenced in this paper, $\mathrm{X} \in \mathbb{R}^{B \times L \times d_{model}}$ (batch-size, sequence-length, feature-dim) represents the input to the model. Y represents the output of the model - multi-variable function -, and $\mathrm{Z} \in \mathbb{R}^{B \times H_{size}}$ (batch-size, hidden-size) represents the latent dimensions of the encoder. Consequently, the equation for the model assuming four layers can be represented below with two cases - ReLU activation and non-activation. However, we will use represent the equation with four layers.

$$Z^{(1)} = TransformerEncoder(X^{(0)}) \tag{8}$$

$$Y = \sigma\left(W^{(4)} \cdot \sigma\left(W^{(3)} \cdot \sigma\left(W^{(2)} \cdot \sigma(Z^{(1)}) + b^{(1)}\right) + b^{(2)}\right) + b^{(3)}\right) \tag{9}$$

In this work, the metric that was used to measure the performance of SHRED-TUN in both reconstruction and forecasting was root-mean-squared-error. This metric allows one to better extrapolate the performance of SHRED-TUN relative to the ground truth. SHRED-TUN, however, does have a limitation. Given the difficulty transformers have with function composition, SHRED-TUN is limited in how effectively it can forecast signals. However, SHRED-TUN as the results suggest performs better in reconstruction tasks than forecasting tasks. Beyond this, the architecture builds upon the SHRED concept and does an excellent job at reconstruction and forecasting nonetheless. With this in mind, it is critical to benchmark SHRED-TUN with respect to other architectures. In future work, the limitation of SHRED-TUN may be addressed. If not, latter work will produce an architecture that extends upon this architecture. Consequently, the performance of the SHRED concept will continue to improve. With that, the potential of Shallow REcurrent Decoding (SHRED) continues to be validated. Additionally, future work may focus on testing SHRED-TUN with additional datasets. Given that this study will test SHRED-TUN on a limited number of datasets. To truly establish an objective of this paper: generalizability. It will be beneficial to continue this work by testing the architecture on additional datasets.

6 SINDy-Transformer SHRED

6.1 Attention as dynamical system modeling

6.2 Symbolic regression as a regularization via SINDy

SINDy attention head

SINDy regularization

7 Computational Experiments

This section explores three different physical systems that we train the SHRED models on to perform next-step state prediction. These dynamical systems come from different inherently different domains and physical scales, ranging from the low-data to the high-data regime as well as from particle physics to global weather phenomena.

For each dataset, we randomly select 50 sensor locations in the state space that persist across time as input to the SHRED models. We then train 16 models which are combinations of encoders and decoders. We perform hyperparameter tuning by modifying the number of layers for the encoders and the model learning rate, increasing the total number of trained models to 128 per dataset. We train each model with a look-back window of 50 for 100 epochs and save the model with the lowest validation loss obtained after at least 10 epochs.

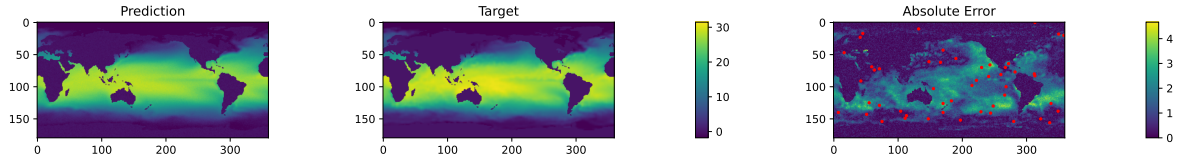
We use the following combinations of numbers of encoder n_e and n_d decoder layers: $n_e \in \{1, 2, 3, 4\}$ and $n_d = 1$. We also train each model with a learning rate lr of $lr \in \{10^{-2}, 10^{-3}\}$.

7.1 Sea Surface Temperature

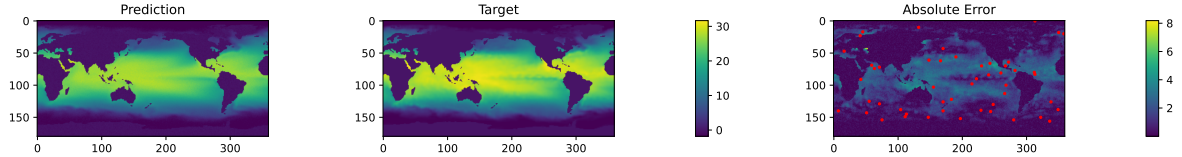
The first dataset we consider is the Sea-Surface Temperature dataset (SST) dataset: a collection of 1,400 weekly snapshots of the weekly mean sea surface temperature collected from 1992 to 2019 by NOAA [17]. The data comes in a 180×360 grid with 44,219 of the 68,400 locations containing sea surface temperature information. The dataset's total size is 179MB.

The dataset is converted into a training, validation, and test set by taking the first 80% of time-steps as the training set, the next 10% as the validation set, and the remaining 10% as the test set. The data is also normalized by min-max scaling the full dataset onto the range [0,1].

Using the hyperparameters listed in section 7, the top 12 best performing models are shown in Figure 3. Figure 2 shows the output of the best model on the test set.



(a) Predicted vs True Sea-Surface Temperature for the first next-step prediction in the test dataset.



(b) Predicted vs True Sea-Surface Temperature for the last next-step prediction in the test dataset. Sensor locations are shown in red.

Figure 2: Predicted vs True Sea-Surface Temperature for the best performing model on SST (GRU encoder with one layer, UNET decoder with one layer). The top figure shows the next-step state prediction in the test dataset generated by passing first window from the test set and is compared with the true output as well as the absolute error. The bottom figure shows the next-step state prediction in the test dataset generated by passing the last window from the test set and is compared with the true output as well as the absolute error.

7.2 Complex Plasma Physics

The second dataset we consider is the plasma dataset. David: please fill out information about the dataset here. The data comes as a time-series of 65,792 time-steps where each time-step is represented as a 2,000 dimensional state vector. The dataset's total size is 785MB.

Using the hyperparameters listed above, the top 12 best performing models are shown in Figure 3

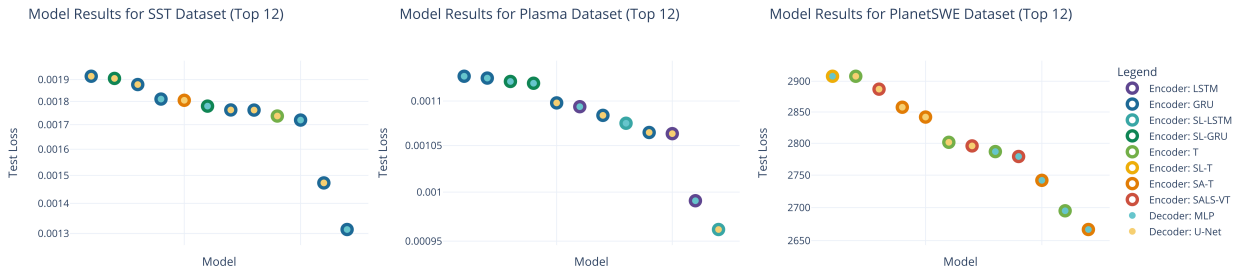


Figure 3: Top 12 best performing models on each dataset. SL-LSTM representes a SINDy-Loss LSTM, SL-GRU represents a SINDy-Loss GRU, T represents a Vanilla Transformer, SL-T represents a SINDy-Loss Transformer, SA-T represents a Sindy-Attention Transformer, SASL-T a SINDy-Attention SINDy-Loss Transformer.

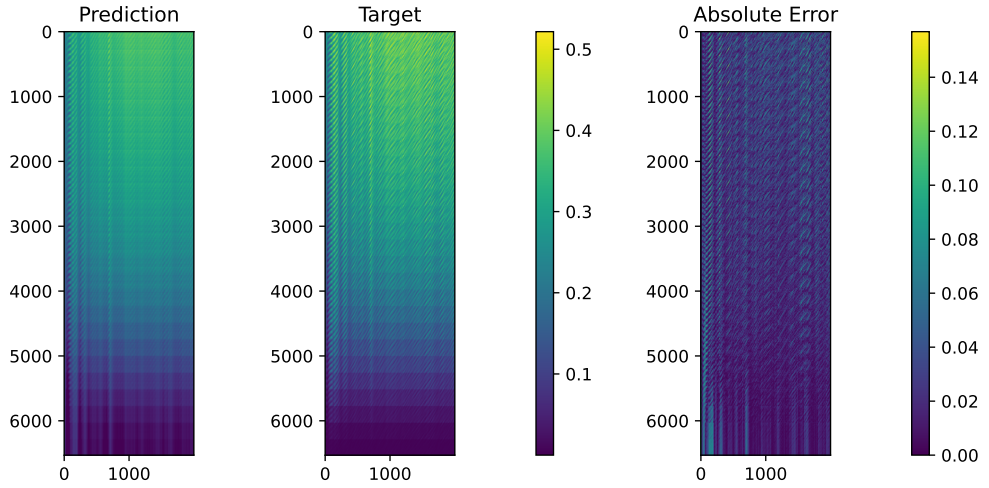


Figure 4: Predicted vs True data for the best performing model on the Plasma dataset (LSTM encoder with three layers, MLP decoder with one layer). The left two figures are the predicted and expected states across the full test dataset (not shown is the 50 time-step input window from the test set). The third figure shows the absolute error between the predicted and target states.

7.3 Shallow Water Equations

The third example comes from The Well [18], which a comprehensive collection of numerical physics simulations spanning a diverse set of domains. We use a subset of the `planetswe` datset from The Well to train our SHRED-based models to observe the performance of T-SHRED in the high-data regime.

The `planetswe` dataset was generated from simulating the rotating forced hyperviscous spherical shallow water equations:

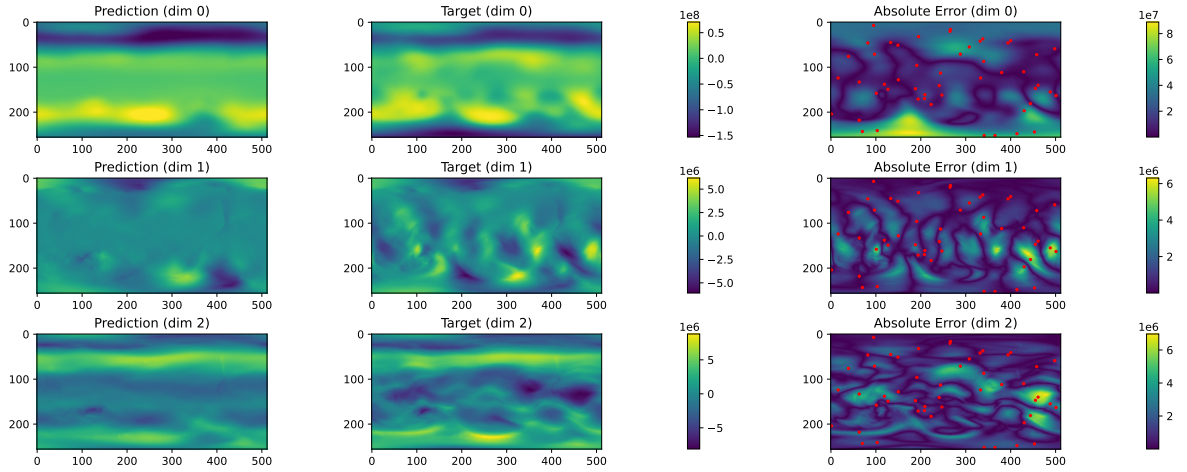
$$\frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u} - g\nabla h - \nu \nabla^4 \boldsymbol{u} - 2\Omega \times \boldsymbol{u} \tag{10}$$

$$\frac{\partial h}{\partial t} = -H\nabla \cdot \boldsymbol{u} - \nabla \cdot (h\boldsymbol{u}) - \nu \nabla^4 h + F \tag{11}$$

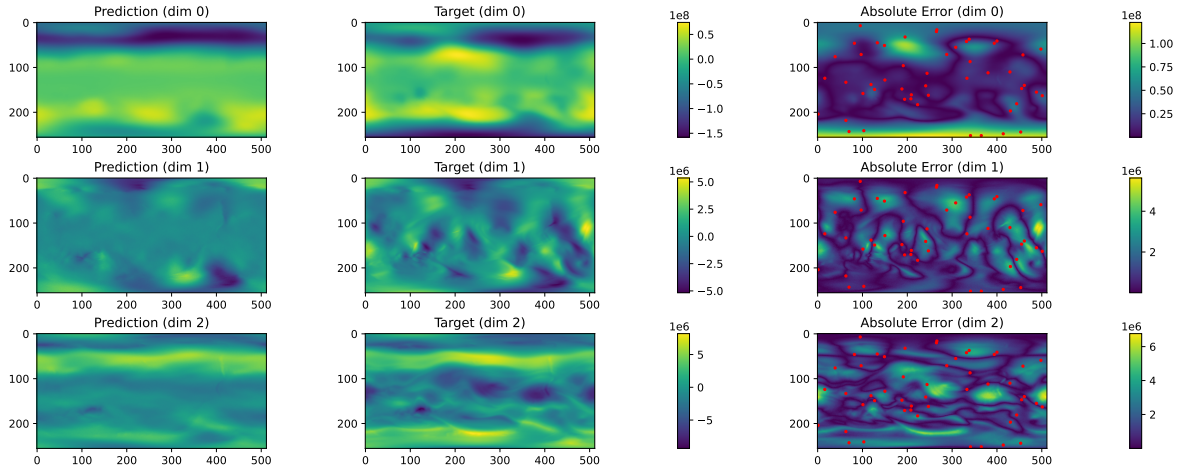
Where $\boldsymbol{u}$ is the vector-valued velocity field, h is the surface height, $\nu = 1.76 \times 10^{-10}$ was used for simulation stability, and F is a time-dependent forcing term.

To generate our training, testing, and validation data, we first take out the first 10 of the 120 total tracks in the dataset. We then split each of the 10 by taking the first 80% of the data as training data, the next 10% of the data as validation data, and the remaining 10% as the testing data. This results in a total dataset size of 15.5 GB.

The top 12 best performing models trained on `planetswe` are shown in Figure 3.



(a) Predicted vs True PlanetSWE for track 0 for the first next-step prediction in the test dataset. Sensor locations are shown in red.



(b) Predicted vs True PlanetSWE for track 0 for the last next-step prediction in the test dataset.Sensor locations are shown in red.

Figure 5: Predicted vs True PlanetSWE for track 0 for the first next-step prediction in the test dataset and the last next-step prediction in the test dataset.

8 Results

The experimental results clearly demonstrate that T-SHRED benefits from larger datasets. This aligns with current deep learning literature that use large datasets in their transformer-based architectures to learn complex dependencies in the dataset. Particularly, we observe in Figure 3 RNNs outperform transformers for the low-data regimes of the SST and plasma dataset, with GRUs excelling on SST and LSTMs on the plasma dataset. In contrast, from Figure 3 we see that transformers perform best on the `planetswe` dataset, comprising all 12 of the lowest test loss models.

Interestingly, while the plasma dataset didn't have any transformers rank in the best 12 lowest test loss models, transformers scored third and fourth in the SST dataset. We also observe that there is no specific transformer configuration that performs better than the rest consistently. The vanilla transformer has just as much of a presence in the high-data regime of `planetswe` in Figure 3 as with the other models, demonstrating that the core architecture of the transformer is sufficient to learn the underlying dynamics of the dataset. While we do not observe any benefit in using a SINDy attention head or SINDy loss for the final model performance, we do not that the best performing model on `planetswe` was a SINDy attention transformer paired with a UNET.

We also do not observe any benefit of using either an MLP or a UNET for the SST or plasma datasets; however, it is clear that the top five best performing models on `planetswe` use UNETs, suggesting their enhanced performance on larger datasets.

9 Conclusion

10 Acknowledgements

11 Appendix

Transformers provide a powerful alternative for sequence modeling. Classical models like RNNs have a limitation in that they have short-term [15] dependencies associated with exploding and vanishing gradients. For this reason, LSTMs were developed. LSTMs still struggle with the problem of sequential processing. In other words, LSTMs process input one sample at a time. This challenge was addressed with bidirectional LSTMs; however, the limitation produced was contextual loss. Transformers address the concerns regarding sequential processing and contextual loss [14]. By resolving the limitations of RNNs, transformers are said to be more efficient and faster. In addition, the attention mechanism of transformers allows for parallel processing. Parallel processing is a feature of transformers, whereas RNNs have to process data sequentially.

How do transformers work? The attention mechanism in transformers allows them to process input sequences in parallel. The attention mechanism was introduced as an approach in computer vision tasks to emphasize certain parts based on their contextual relevance in the application. Incorporating the self-attention mechanism in the transformer model better captured local features and reduced the reliance on external information. Additionally, transformers possess a positional encoding. Positional encoding provides information as to the order of sequences within the model. In the context of words, the model can better gauge word order. The reason this is necessary is by processing the data in parallel. The model is not able to gauge the order of the words. Positional encoding addresses this by encoding the position of each word within the sequence.

Where are transformers currently applied? Transformers are a neural network model that can learn representations of sequences or sets of data points. Transformers have driven advances in natural language processing, computer vision, and spatial-temporal modeling [13]. For illustration, they have become very prevalent and well-known in natural language processing. Transformers are useful because, unlike RNNs, transformers can handle larger input sequences. It is said that transformers have become the predominant infrastructure and state-of-the-art solution for the majority of learning-based machine intelligence tasks in the field of artificial intelligence [14]. The entire transformer architecture is not used. Instead, the transformer encoder is utilized [13].

What composes the architecture of a transformer? What are the benefits of utilizing transformers? Transformers address the limitations of preexisting neural network architectures. They can train faster because they utilize parallel processing. Similarly, transformers address the vanishing gradient problem, which RNNs have to deal with. In consequence, the transformer can utilize longer sequences without reduction in training performance. Similarly, as the objective function is traversed the gradient will not converge too early. Additionally, in biomedical imaging, the transformer can alleviate the problem of scattered target regions and large shape differences [15]. Similarly, the transformer can use a convolutional neural network structure to take advantage of the ability of sample information to extract multiscale local spatial features; this would in turn improve model performance by allowing the model's global and local information achieve a balance [15].

What are the limitations of the transformer architecture? Transformer architectures do have their benefits that make them very clear to utilize them to build upon SHRED-TUN. However, a limitation of transformers it that they hallucinate. It is said that a major contribution of this has to do with function composition [16]. The limitations take place primarily for very large inputs [16]. As an illustration, the issue of function composition can be explained as such. If you have a sentence, "Joe's brother is going for a walk", brother relies on the fact that we are referring to Joe.

In the context of spatio-temporal data, a hallucination would be a vastly inaccurate forecasting of the time series that does not match the established ground truth Y. Despite these limitations, the fact that transformers address the problems of RNNs make them a strong proponent for utilization in SHRED-TUN. Why are we using a transformer encoder? The point of this SHRED adaptation (SHRED-TUN) is to be able to utilize the encoder component of the transformer architecture to capture long-term dependencies in the time series data. Theoretically, this would make SHRED more generalizable to different problems and datasets where the temporal connections are longer and more complex. SHRED-TUN is capable of both forecasting and reconstruction. A transformer encoder allows the whole sequence to be processed in parallel, which improves the training time of the model. The original SHRED utilizes an RNN encoder. The typical problems that RNNs face could be resolved using a transformer encoder. The reason for this is that transformers resolve the typical issues RNNs face. This is relevant because

How does this deviate from SHRED and what is the theoretical result? In theory, SHRED-TUN should perform better than SHRED because of its ability to capture long-term dependencies / longer sequences in the data. In this case, better is more generalizable and precise (metric-wise). Moreover, the utilization of a transformer encoder provides more accurate interpretability than SHRED. The reason for the improved interpretability comes from the attention-heads of the model. Moreover, this paper will go through [16] several potential applications of SHRED-TUN that highlight the added interpretability and generalization of the model. Furthermore, implementing a transformer encoder with SHRED allows the model to deal with larger datasets more effectively. In principle, the model can scale; more data reduces the model's chance to overfit. Consequently, the interpretability and reliability of the model improve. Similarly, the model does not have to deal with the vanishing gradient problem that RNNs and LSTMs have to deal with because they can look at the whole sequence once due to their self-attention mechanism. Convolutional neural networks were developed to address certain problems that MLPs faced when dealing with data that possessed many hierarchical features. Additionally, image data for example possess a large number of pixels, which leads to a significant number of parameters in MLPs that make them computationally expensive and prone to overfitting. The CNN is a solution to the limitations of the MLP network when dealing with such data. CNNs have pooling layers that reduce the dimensionality of the data making it less computationally expensive. Similar to how the development of transformers addressed certain limitations facing RNNs, the development of CNNs addressed the limitations of RNNs in handling data that would otherwise produce computationally expensive models. UNETs are an architecture that was originally developed for biomedical imaging. Their architecture utilizes skip connections, which has made them useful for tasks where the inputs and outputs have the same shape. UNets and convolutional decoding have strong evidence in the literature for the ability to extract features and reconstruct spatial information. UNets have been used in a variety of contexts, such as image segmentation and signal/image reconstruction. In this paper, the utilization of UNETs provides the opportunity for SHRED to extract local features better. UNETs rely on convolutional neural networks as an earlier foundation. In consequence, they are still designed to deal with the limitations of MLPs. The original SHRED architecture utilized an MLP decoder. In consequence, while effective, it demonstrates that the utilization of UNETs/CNNs in the SHRED-TUN architecture is reasonable.

What challenges do UNETs overcome? Before UNETs, there were other traditional approaches to image segmentation in medical imaging. UNETs were designed to overcome the limitations of the traditional approaches in image segmentation. For instance, UNETs can segment from input-output pairs without user annotation. User annotation implies sketching image boundaries by hand, which is labor and time-intensive. Additionally, the fact that the UNET operates only on convolutional layers permits it to work on images of any size. The encoding path and decoding path of the architecture permit the UNet to learn both global context and local information. Thus, the UNET architecture is very useful for dealing with reconstruction tasks that require a model to have the capability to learn different levels of visual information (colors, objects, etc) similar to a spatial-exploration-based CNN. The purpose of utilizing UNETs in SHRED is to help the model learn

spatial relationships. UNETs excel at reconstruction in various areas. However, this critical aspect is advantageous for dealing with spatial and temporal data in a hybrid model such as SHRED-TUN (Shallow Recurrent Decoding with Transformer Networks and UNETs). SHRED deals with tasks that require a model to learn both local and global information. Similarly, learning this global features is not sufficient enough. The model must be able to accurately retain the information with minimal loss to properly reconstruct the data. Similarly, a well-learned model can better forecast the spatio-temporal data at the respective time steps necessary.

What is the UNet architecture? UNETs consist of two paths as a basic structure. The first path is the contracting path. The first path provides classification information, and it is also known as the encoder path. It decreases the spatial dimensions of the images while capturing critical information about the image. The contracting path utilizes pooling and convolution layers. The second path is known as the decoder path. The second path consists of up-sampling and concatenations from the contracting path followed by convolutions. The expansion in the second path allows the network to learn localized classification information. Moreover, the upsampling allows the model to recover spatial information. Where are UNETs applied? As stated previously, UNETs have significant applications in medical imaging. This is due to the recent advances in computer vision within the last decade (U-NETS and its variants for medical imaging). UNETs are very strong with localization, and UNETs are useful in preserving fine details. This is helpful in a variety of contexts such as recording EEG signals where fine details are pertinent or reconstructing images extracted from signal measurements. Additionally, U-NETs are very useful to medical imaging because they create highly detailed segmentation with very limited training samples (U-NET and 6 its variants for medical image segmentation: theory and applications). Another benefit is that U-NETs are faster to train than other segmentation methods (U-NET and its variants for medical image segmentation: Theory and applications). Consequently, UNETs/CNNs seemed to be an adequate choice for our decoder network.

How do UNETs work and why use a UNET decoder? UNETs can be seen as an encoder network followed by a decoder network in simple terms. The use of a UNET decoder is beneficial. The reason why a UNET decoder is helpful is that it can recover the spatial relationships from the data rigorously. For emphasis, the decoder would be useful in capturing the intricacies of the local and global context of the spatiotemporal data. By using a UNET decoder, SHRED can deal with more complex data and problems. Similarly, SHRED can improve its reconstruction capacity significantly. This is essential because the original uses of SHRED are related to reconstruction and forecasting problems. SHRED-TUN as an extension of SHRED should be able to deal with the problems that SHRED did, but it should be able to do so with significantly more precision. Additionally, the ability of UNETs to enhance the precision of segmentation or reconstruction tasks with less data makes it an ideal choice for our network. SHRED-TUN has many potential uses. This can be seen in the applications section of this paper. The combination of utilizing transformers and UNETs presents an excellent opportunity for better interpretability and generalizability to many tasks. SHRED-TUN utilizes a transformer encoder and UNET decoder. The UNET decoder is a feature that utilizes hyperparameter tuning. In other words, it could be a simple CNN; it depends on the dataset. In the case of this paper, the UNET-Decoder is a width-based CNN where the feature maps increase in size at each layer. Additionally, the decoder utilizes same padding across each layer. SHRED-TUN takes advantage of the transformers resolution of problems regarding RNNs. Furthermore, SHRED-TUN utilizes the ability of CNNs and UNETs to extract spatial features. SHRED-TUN also deviates from SHRED in that convolutional neural networks were specifically designed to deal with spatial data such as images and such. Thus, the SHRED-TUN model theoretically should be an improvement from SHRED when dealing with data of that context. Moreover, the one-dimensional component of the convolutional layers are utilized in the process are effective at learning temporal relationships in the data. In this case, their goal is to decode the spatio-temporal relationships being encoded by the convolutional neural network. Since the transformer deals with the vanishing gradient problem, SHRED-TUN can utilize long sequences and more data to provide more effective generalizations to many problems.

SHRED-TUN has proven very precise in a myriad of problems/contexts. In similar consideration, the capability of SHRED- TUN to resolve the vanishing gradient problem seen in SHRED, allows additional information to be retained in the model. Thus, the reconstruction precision is improved. Since UNETs/CNNs can do more with less data and transformers allow one to increase the amount of data passed to SHRED with less consequences. The combination of these two powerful tools provides a relevant jump from SHRED. Since SHRED-TUN utilizes a transformer encoder, it produces greater interpretability than SHRED. The significance of this is that it could produce better insights into the patterns for the data it is trained on. SHRED-TUN can be applied in reconstruction tasks. For instance, the utilization of a UNET in the SHRED-TUN architecture gives it the possibility to effectively reconstruct spatio-temporal systems that require the model to learn both global and local information. A possible application of SHRED-TUN would be a biomedical application given that it utilizes a UNET decoder. SHRED-TUN could be utilized to help decode MEG signals. Moreover, SHRED-TUN could be utilized to reconstruct relevant features coming from biomedical signals or images. Similarly, SHRED-TUN can be used to solve various problems in physics and engineering. An example of this would be in plasma physics. This work will demonstrate that SHRED-TUN can be effectively utilized in forecasting and reconstructing plasma signals via multiple sensors. Similarly, SHRED-TUN performs relatively well when reconstructing plasma signals as well. In consequence, this is a promising application area for SHRED-TUN.

For a formulaic representation of the SHRED-TUN architecture referenced in this paper, $\mathrm{X} \in \mathbb{R}^{B \times L \times d_{model}}$ (batch-size, sequence-length, feature-dim) represents the input to the model. Y represents the output of the model - multi-variable function -, and $\mathrm{Z} \in \mathbb{R}^{B \times H_{size}}$ (batch-size, hidden-size) represents the latent dimensions of the encoder. Consequently, the equation for the model assuming four layers can be represented below with two cases - ReLU activation and non-activation. However, we will use represent the equation with four layers.

$$Z^{(1)} = TransformerEncoder(X^{(0)}) \tag{12}$$

$$Y = \sigma\left(W^{(4)} \cdot \sigma\left(W^{(3)} \cdot \sigma\left(W^{(2)} \cdot \sigma(Z^{(1)}) + b^{(1)}\right) + b^{(2)}\right) + b^{(3)}\right) \tag{13}$$

In this work, the metric that was used to measure the performance of SHRED-TUN in both reconstruction and forecasting was root-mean-squared-error. This metric allows one to better extrapolate the performance of SHRED-TUN relative to the ground truth. SHRED-TUN, however, does have a limitation. Given the difficulty transformers have with function composition, SHRED-TUN is limited in how effectively it can forecast signals. However, SHRED-TUN as the results suggest performs better in reconstruction tasks than forecasting tasks. Beyond this, the architecture builds upon the SHRED concept and does an excellent job at reconstruction and forecasting nonetheless. With this in mind, it is critical to benchmark SHRED-TUN with respect to other architectures. In future work, the limitation of SHRED-TUN may be addressed. If not, latter work will produce an architecture that extends upon this architecture. Consequently, the performance of the SHRED concept will continue to improve. With that, the potential of Shallow REcurrent Decoding (SHRED) continues to be validated. Additionally, future work may focus on testing SHRED-TUN with additional datasets. Given that this study will test SHRED-TUN on a limited number of datasets. To truly establish an objective of this paper: generalizability. It will be beneficial to continue this work by testing the architecture on additional datasets.

Alternative introduction (copied here as a reference)

The SHRED neural network architecture is demonstrably useful in scientific and engineering applications. Based on the separation of variables technique, the model can be seen as a joint training of learning the temporal trajectory and spatial field of the input data simultaneously through the encoder and decoder respectively [1]. Several works have used SHRED to perform state space reconstruction from a sparse set of sensors in the spatial dimension [1–3]. These works have shown that a full state reconstruction of the data can be obtained from sparse sensors

where the sensors only cover about 0.5% of the original space. Furthermore, the SHRED models are agnostic to the specific system they are modeling. They can perform Go-Pro physics, where dynamics are learned directly from video [2, 3]; they can also learn chaotic fluid dynamics from simulations [2, 3]. These results encourage the further development of SHRED architectures.

While it is clear that the theoretical motivation for SHRED models has led to promising results, previous work has not implemented modern deep learning advancements into the architecture. A significant amount of modern deep learning architectures in every domain use transformers. First used for machine translation [4], they utilize a self-attention mechanism to learn complex patterns from large datasets more efficiently than other models available at the time. Since then, significant empirical evidence has been produced showing that transformer-based models scale exceptionally well with more data in a variety of domains and architectures [5–7].

In this paper, we introduce Transformer SHRED (T-SHRED) which uses a transformer backbone within the SHRED architecture for sparse sensor modeling. The incorporation of a transformer into the SHRED architecture should outperform previous SHRED models by utilizing recent developments in deep learning. We perform an empirical analysis of T-SHRED with the other SHRED architectures by combining different encoders and decoders and evaluating their performance in next-step full state prediction from a sparse set of measurements. We also study the effect SINDy loss [3] has on our prediction task as well as attaching SINDy onto the individual heads of the transformer.

The next-step state prediction task is performed on three different datasets, all coming from physical phenomena on different scales and with different dataset sizes. We show that as the amount of data used in trainingz SHRED increases, transformer based models outperform traditional Recurrent Neural Network (RNN) SHRED architectures on this task. Piece of the Introduction The SHRED neural network architecture is demonstrably useful in scientific and engineering applications. Based on the separation of variables technique, the model can be seen as a joint training of learning the temporal trajectory and spatial field of the input data simultaneously through the encoder and decoder respectively [1]. Several works have used SHRED to perform state space reconstruction from a sparse set of sensors in the spatial dimension [1–3]. These works have shown that a full state reconstruction of the data can be obtained from sparse sensors where the sensors only cover about 0.5% of the original space. Furthermore, the SHRED models are agnostic to the specific system they are modeling. They can perform Go-Pro physics, where dynamics are learned directly from video [2, 3]; they can also learn chaotic fluid dynamics from simulations [2, 3]. These results encourage the further development of SHRED architectures.

While it is clear that the theoretical motivation for SHRED models has led to promising results, previous work has not implemented modern deep learning advancements into the architecture. A significant amount of modern deep learning architectures in every domain use transformers. First used for machine translation [4], they utilize a self-attention mechanism to learn complex patterns from large datasets more efficiently than other models available at the time. Since then, significant empirical evidence has been produced showing that transformer-based models scale exceptionally well with more data in a variety of domains and architectures [5–7].

In this paper, we introduce Transformer SHRED (T-SHRED) which uses a transformer backbone within the SHRED architecture for sparse sensor modeling. We perform an empirical analysis of different SHRED architectures by combining different encoders and decoders and evaluating their performance in next-step full state prediction from a sparse set of measurements. We also study the effect SINDy loss [3] has on our prediction task as well as attaching SINDy onto the individual heads of the transformer.

The next-step state prediction task is performed on three different datasets, all coming from physical phenomena on different scales and with different dataset sizes. We show that as the amount of data used in training SHRED increases, transformer based models outperform traditional Recurrent Neural Network (RNN) SHRED architectures on this task.

Alternative Abstract (copied here as a reference)

Shallow recurrent decoder networks and SINDy-SHRED have succeeded in state estimation

and identification with datasets from a myriad of historically challenging tasks. These tasks, in question, pertain to state estimation and identification as well as next-step full state prediction from sparse sensor measurements. Similarly, shallow recurrent decoding as a tool to address the problem of state estimation and identification while providing significant results with limited data has proven successful. State estimation and identification enable further control of complex systems that impact many areas of society that possess functions that are obstructed from view. Thus, these reasons provide the drive to continue pursuing the SHRED concept in SINDy-transformer. The success of the SHRED concept, especially SINDy-SHRED, indicates that it is worth trying to implement the SINDy-Transformer. SHRED provides a baseline for comparison of the performance of the SINDy-Transformer and its theoretical basis. By comparing SINDy-Transformer to historically high-performing models, one can gauge its relevance, functionality, and impact more easily. Additionally, SINDy-SHRED and SHRED utilize RNNs for the encoder. In consequence, this implies it still deals with the vanishing gradient problem. By utilizing a transformer encoder, it is believed that SINDy-Transformer can improve upon the performance of SINDy-SHRED and SHRED, as well as be able to account for situations where there is vast data. In theory, SINDy-Transformer should be able to perform better than SINDy-SHRED and SHRED because of its ability to address the vanishing gradient problem; it should perform better because a large dataset allows the model to learn more complex patterns and generalize better. Similarly, the self-attention capacity of the transformer allows it to process data in parallel and capture long-range spatiotemporal dependencies, which is critical for systems that possess memory. The results in this paper were achieved through experimentation with the SINDy-Transformer indicate that this direction is very promising.

References

References

[1] J. P. Williams, O. Zahn, and J. N. Kutz, *Sensing with shallow recurrent decoder networks*, Proceedings of the Royal Society A, 480 (2024), p. 20240054.

[2] M. Tomasetto, J. P. Williams, F. Braghin, A. Manzoni, and J. N. Kutz, *Reduced order modeling with shallow recurrent decoder networks*, 2025.

[3] M. L. Gao, J. P. Williams, and J. N. Kutz, *Sparse identification of nonlinear dynamics and koopman operators with shallow recurrent decoder networks*, 2025.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, Advances in neural information processing systems, 30 (2017).

[5] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, *Scaling vision transformers*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2022, pp. 12104–12113.

[6] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, *Scaling laws for neural language models*, arXiv preprint arXiv:2001.08361, (2020).

[7] Z. Liang, H. He, C. Yang, and B. Dai, *Scaling laws for diffusion transformers*, arXiv preprint arXiv:2410.08184, (2024).

[8] J. P. Williams, O. Zahn, and J. N. Kutz, *Sensing with shallow recurrent decoder networks*, 2024.

[9] J. L. Elman, *Finding structure in time*, Cognitive science, 14 (1990), pp. 179–211.

[10] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.

[11] M. Schuster and K. K. Paliwal, *Bidirectional recurrent neural networks*, IEEE transactions on Signal Processing, 45 (1997), pp. 2673–2681.

[12] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, *xlstm: Extended long short-term memory*, arXiv preprint arXiv:2405.04517, (2024).

[13] R. E. Turner, *An introduction to transformers*, 2024.

[14] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz, *A comprehensive survey on applications of transformers for deep learning tasks*, 2023.

[15] D. Mienye, T. Swart, and G. Obaido, *Recurrent neural networks: A comprehensive review of architectures, variants, and applications*, Information, 15 (2024), p. 517.

[16] B. Peng, S. Narayanan, and C. Papadimitriou, *On limitations of the transformer architecture*, 2024.

[17] R. W. Reynolds, N. A. Rayner, T. M. Smith, D. C. Stokes, and W. Wang, *An improved in situ and satellite sst analysis for climate*, Journal of climate, 15 (2002), pp. 1609–1625.

[18] R. Ohana, M. McCabe, L. Meyer, R. Morel, F. J. Agocs, M. Beneitez, M. Berger, B. Burkhart, K. Burns, S. B. Dalziel, D. B. Fielding, D. Fortunato, J. A. Goldberg, K. Hirashima, Y.-F. Jiang, R. R. Kerswell, S. Maddu, J. Miller, P. Mukhopadhyay, S. S. Nixon, J. Shen, R. Watteaux, B. R.-S. Blancard, F. Rozet, L. H. Parker, M. Cranmer, and S. Ho, *The well: a large-scale collection of diverse physics simulations for machine learning*, 2025.