# Few-Shot Transfer Learning on Time Series Data with Fisher Task Affinity

Xiaoxi Celia Lyu
*xiaoxi.lyu@duke.edu*

Weixin He
*weixin.he@duke.edu*

Ziyan Chen
*ziyan.chen@duke.edu*

Zhan Shi
*zhan.shi@duke.edu*

## Abstract

Recent work has developed an asymmetric affinity score measuring the similarities between different tasks. With this developed fisher task affinity measurement, this project examines the performance of task affinity and transfer learning on both simulated time series data and the real-world weather data source under different time series deep learning architectures. We found that the fisher task affinity efficiently recovers the intuitive distance between time series datasets, and is stable while changing the neural net architectures. Few-shot transfer learning on limited available data is also successfully implemented.

## 1. Introduction

Given the wide use of time series data, time series forecasting has always been one of the most important research areas. With the fast development of data mining techniques, deep learning models (e.g., neural network) has been successfully applied to the new generation of time series forecasting models[1]. Such methods like MLP, RNN, LSTM, CNN have also been applied to multiple real-world areas including the prediction of stock price, weather, traffic, text generation, etc.

However, inherited from the innate limitation of traditional machine learning, the performance of the forecasting model will be threatened by the difference between the training data and test data: typically the more significant the difference is, the worse the performance on target data will be[7]. However, constrained with the time, budget and all kinds of other limitations, sometimes it is hard to find the matched train data source for the target data. Therefore, transfer learning is of vital importance to alleviate the need for the perfect-matched train dataset, and improve the performance on the target test dataset[10].

Under the scenario of time series forecasting, transfer learning will also be beneficial. Taking weather forecasting as an example, consider the case where we want to build a weather forecasting model for a newly developed area. Collecting the weather data in that particular location from the very start would be expensive and time-consuming. Then, if we can apply the transfer learning to use the model developed from other areas as the starting point for the forecasting task in this new location, we could achieve better performance faster. Another example might be the price prediction for a newly listed stock, transfer learning will be useful if we can apply the models trained for other existing stocks. It can be seen that the transfer learning method would be beneficial in the application of time series data sources.

Moreover, following the example of weather forecasting, it is intuitively important to examine the geographical similarities between the newly developed area and other areas. Generally speaking, the task affinity (i.e., the similarities between different datasets in this task), and their impact on the performance of transfer learning needed to be analyzed. One of the proposed methods of examining task affinity is the Task Affinity Score based one the Fisher Information Matrix[4].

With all the techniques and tools abundantly developed, little applications of transfer learning on the time series data source have been conducted. Therefore, this project focuses on the application of task affinity examination and transfer learning on the time series data source. Several key questions this project examined are listed below:

1. whether the fisher task affinity efficiently recovers the intuitive ordering of the time series datasets: both simulated dataset and real-world weather dataset.

2. whether the fisher task affinity has the expected correlation with the performance on the target dataset.

3. how is the performance of the transfer learning based on the training dataset selected by the fisher task affinity, especially under the few-shot learning with data scarcity setting.

4. whether the task distance and the performance of the transfer learning are stable with different neural net architectures.

The outline of this project report is given below. In section 2, we further discuss the related work to our project. In section 3, we introduce the methodology applied in this project: Task Affinity Score based on the Fisher Information Matrix examined to track the asymmetric affinity between tasks, and two time-series models (RNN and LSTM) used to train the network. In section 4, we showed the results of our methods in both simulated datasets and real-world weather datasets, to verify if the task affinity recoveries the intuitive ordering of the datasets, and whether task distance is stable under different neural net architectures. In section 5, we make the conclusion of the project.

## 2. Related Works

The previous literature has developed multiple deep learning architectures for the time-series datasets that have successfully facilitated decision support with time-series data source[5]. The most commonly used and successfully applied deep learning architectures in the timer series forecasting are feed forward networks, recurrent neural networks (including Elman, long-short term memory, gated recurrent units, and bidirectional networks), and convolutional neural networks[9]. Among all these architectures, recurrent neural networks (RNN) occurred with outstanding performance[2]. For example, in one recent M4 competition, an RNN was able to achieve impressive performance and win the competition[8]. Therefore, this paper trained the time series data source with the method of the standard recurrent neural networks (RNN), and the related Long short-term memory networks.

Meanwhile, transfer learning and task affinity have also been widely studied. Multiple methods of measuring the similarity between tasks have been proposed in the past literature. According to the literature review in the paper by Cat, Juncheng, Mohammadreza, and Vahid[4]: many approaches in transfer learning are based on the assumption that similar tasks often share similar architectures. Therefore, those measures are more focused on the implementation of transfer learning on how trained weights could be transferred to the target task, instead of providing a measure that identifies the tasks themselves. Fisher affinity measure, on the other hand, not only allows the identification of the related tasks to the target task but also requests only a few data samples. The paper has conducted experiments with this fisher task affinity on multiple computer vision datasets. However, there is no experiment conducted on the time series data source with time series modeling architecture.

. Final project for ECE685D Introduction to Deep Learning, Duke University, Spring 2022. All authors contributed equally to this work.

While there is sufficient literature on both machine learning techniques in the time series data source and the task affinity examination for the transfer learning, applications of the transfer learning on the time series dataset have been rarely done. Here, we applied the fisher task affinity measure on the time series data source to examine the performance of the fisher task affinity measure and the transfer learning on the down-sampling datasets.

## 3. Methodology

### 3.1. Task Affinity Score

An asymmetric affinity score is a score to represent the complexity of utilizing the knowledge of one task for learning another one. In other words, task affinity score measures the similarity from a source task to a target task. In this section, we introduce how to get the task affinity score from a mathematical point of view. Firstly, the Fisher Information Matrix should be defined. Based on that, the Task Affinity Score can be calculated. Noted that all the contents in this section are cited from the paper by Cat, Juncheng, Mohammadreza, and Vahid [4].

**Fisher Information Matrix.** For a neural network $N_\theta$ with weights $\theta$, data $X$, and the negative log-likelihood loss function $L(\theta) := L(\theta, X)$, the Fisher Information matrix is defined as:

$$F(\theta) = \mathbb{E}[\nabla_\theta L(\theta) \nabla_\theta L(\theta)^T] = -\mathbb{E}[\mathbf{H}(L(\theta))] \tag{1}$$

where $\mathbf{H}$ is the Hessian matrix, i.e., $\mathbf{H}(L(\theta)) = \nabla^2_\theta L(\theta)$, and expectation is taken w.r.t the data.
In practice, we use the empirical Fisher Information matrix computed as follows:

$$\hat{F}(\theta) = \frac{1}{|X|} \sum_{i \in X} \nabla_\theta L^i(\theta) \nabla_\theta L^i(\theta)^T \tag{2}$$

where $L_i(\theta)$ is the loss value for the $i^{th}$ data point in $X$.

**Task Affinity Score (TAS).** Let $X_a$ be the source dataset $a$, $X_b$ be the source target $b$, $N_{\theta_a}$ be the network trained with dataset $a$. Let $F_{a,a}$ be the Fisher Information matrix of $N_{\theta_a}$ with the source dataset $a$, and $F_{a,b}$ be the Fisher Information matrix of $N_{\theta_a}$ with the target dataset $b$. Then we define the TAS from the source dataset $a$ to the target dataset $b$ based on Fréchet distance as follows:

$$s[a,b] := \frac{1}{\sqrt{2}} Trace(F_{a,a} + F_{a,b} - 2(F_{a,a}F_{a,b})^{\frac{1}{2}})^{\frac{1}{2}} \tag{3}$$

Here, we use the diagonal approximation of the Fisher Information matrix since computing the full Fisher matrix is prohibitive in the huge space of neural network parameters. We also normalize these matrices to have unit trace. As a result, the TAS in equation (3) can be simplified by the following formula:

$$\begin{aligned} s[a,b] &:= \frac{1}{\sqrt{2}} ||F^{\frac{1}{2}}_{a,a} - F^{\frac{1}{2}}_{a,b}||_F \\ &= \frac{1}{\sqrt{2}} [\sum_i ((F^{ii}_{a,a})^{\frac{1}{2}} - (F^{ii}_{a,b})^{\frac{1}{2}})^2]^{\frac{1}{2}} \end{aligned} \tag{4}$$

where $F^{ii}$ denotes the $i^{th}$ diagonal entry of the Fisher Information matrix. The TAS ranges from 0 to 1, with the score $s = 0$ denotes a perfect similarity and the score $s = 1$ indicates a perfect dissimilarity.

### 3.2. Time Series Model Architectures.

In this section, we introduce two architectures we used to get the trained network $N_\theta$ in this project.

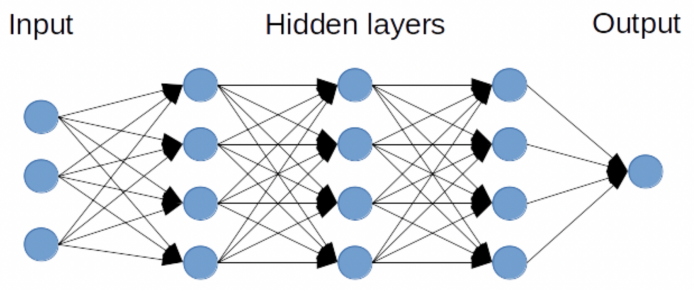


Figure 1. A feed forward neural network

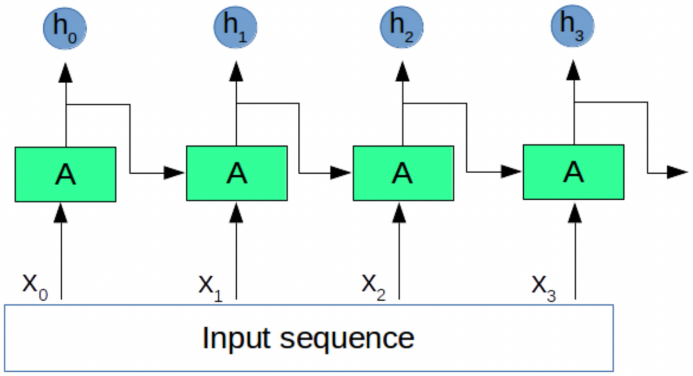


Figure 2. An RNN network

**RNN.** An RNN (recurrent neural network) architecture was used in this project. In simple terms, an RNN network is a neural network that allows previous outputs to be used as inputs. Comparing with the feed-forward network in Figure 1, the modular structure of RNN is slightly more complicated. Specifically, each module in the RNN is a layer that is connected to both the input at the current layer $i$ and the output of the previous layer $i-1$, where $i > 0$. In this way, an RNN structure in Figure 2 allows the network to keep both the current and the previous context in mind when processing the input sequence, which helps achieve better accuracy in come cases [6].

**LSTM.** An LSTM (long short term memory) architecture was also used in this project, which is an extended form of RNN developed to extend the memory to learn long-term dependencies in the input sequence [3]. Similar to an RNN network, the LSTM network allows previous outputs to be used as inputs. Additionally, the LSTM architecture can remember information for long periods of time. Similar to RNN networks, the LSTM also has a chain-like structure, but the repeating module has a different structure from the RNN. The LSTM in Figure 3 has four inner layers of units (i.e., forget gate layer, input gate layer, new candidate gate layer, output gate layer), but the RNN just has one. Using the learnable parameters of these four inner layers, the LSTM can learn what parts should be kept in mind when producing activations.

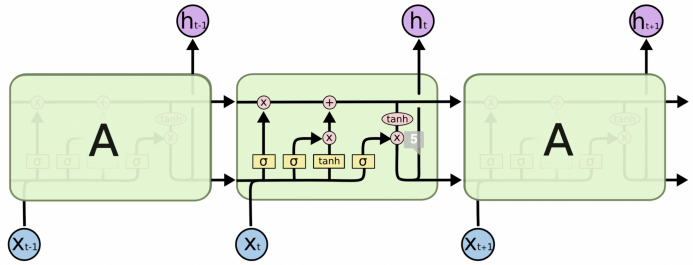


Figure 3. The repeating module in an LSTM contains four inner layers

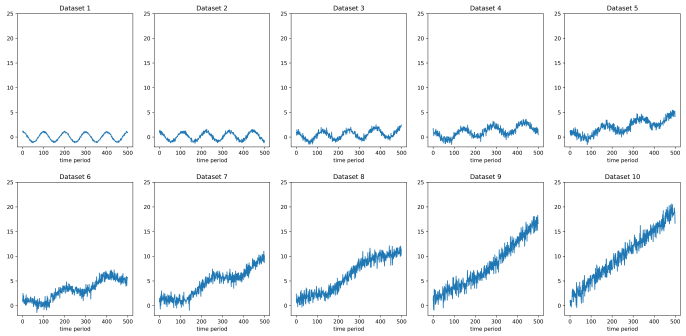


Figure 4. Synthetic data (first 500 observations only)

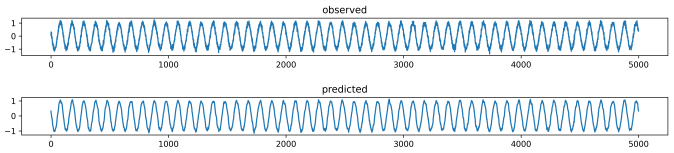


Figure 5. Dataset 0 and its predicted value using RNN (first 5000 periods only)

# 4. Experimental results

## 4.1. Results on Simulated Data

In this section, we present the results in two experiments with a synthetic time series dataset.

**4.1.1. Synthetic Data Simulation.** In this simulation study, we construct 10 time series data, with each having 10,000 observations. These ten series are constructed in a way that

$$\boldsymbol{series}_j = \boldsymbol{cos}_j + \boldsymbol{\epsilon}_j + \boldsymbol{trend}_j$$

for $j \in \{0, 1, \cdots, 9\}$ where

- time periods: $\boldsymbol{t} = \{0, 1, 2, \cdots, 9999\}$
- cosine frequency: $\gamma_j = -.1 \times j + 1$
- cosine: $\boldsymbol{cos}_j = cos(2\pi\gamma_j \times \boldsymbol{t})$
- time trend: $\boldsymbol{trend}_j = .05 \times \boldsymbol{t} \times j^2$
- noise: $\boldsymbol{\epsilon}_j \sim N(0, \sigma_j^2)$ where $\sigma_j = .1(j+1)$

In this study, $\boldsymbol{series}_0$ (dataset 0) is treated as the source dataset. By our construction, from dataset 0 to 9, the frequency of the cosine function decreases, the variance of the noise increases, and the scale of the time trend increases. Correspondingly, the intuitive difference between $\boldsymbol{series}_0$ and $\boldsymbol{series}_j$ increases from $j = 1$ to $j = 9$. Therefore, the index of the datasets $0 : 9$ represents the intuitive order of data, with $\boldsymbol{series}_1$ being closest to $\boldsymbol{series}_0$ while $\boldsymbol{series}_9$ is the furthest. Figure 4 plots the first 500 observations of these 10 time series datasets. We split each series into a train set and a test set with a 9-1 ratio, using windows with size 20 and stride equal to 1.

**4.1.2. Experiment 1: Fisher Task Affinity.**

**Fisher distance and the intuitive order.** In experiment 1, since the synthetic dataset 0 is treated as the source dataset, we first train a model to fit this source data. In the RNN network, both the input and the output dimensions are 1, and the dimension of the two hidden layers is 40. And we use the learning rate as 0.001, the optimizer as Adam, and a scheduler as MultiStepLR with 50 as milestone epoch and 0.1 gamma factor. The true and predicted dataset 0 in the first 5,000 periods are shown in Figure 5.

The fisher distance from source dataset 0 to synthetic dataset 1 to 9 are then evaluated based on the RNN model trained on dataset 0. As expected, as the initialization order of the synthetic dataset increases, its fisher distance to the source dataset 0 steadily increases. The result is shown in Figure 6.

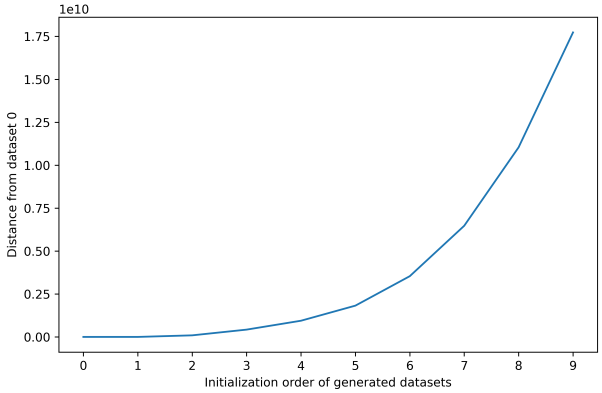


Figure 6. Fisher distance from the synthetic source dataset 0 using RNN

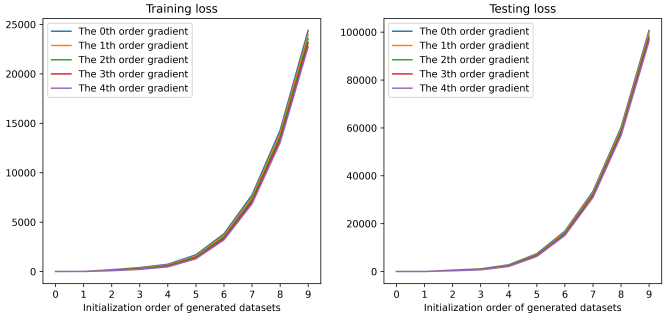


Figure 7. Gradient update on synthetic datasets using RNN

**Fisher distance and loss evaluation.** We want to see whether the recovered fisher task affinity has the expected correlation with the source model's performance on the target dataset. To do this, after we obtain the base model trained with the source dataset 0, we perform a gradient update, using the trainset of each of the remaining 9 series separately. For each of the updated models, we then evaluated the loss on the testset correspondingly. Each gradient update step is implemented with the entire trainset, and we performed 0-4 steps of gradient updates. Figure 7 reports the train and test loss in each step during the updates (please zoom in). Ideally, for the same order of gradient update, as the initialization order of the synthetic dataset increases, both the training loss and the testing loss should steadily increase as the fisher distance from source dataset 0 increases; and for the same dataset, as the order of gradient increases, both the training loss and the testing loss should steadily decrease as the network converges to a better solution. The result presents in Figure 7 meets our expectations.

**Fisher distance stability.** We want to check whether results in Figure 6 and Figure 7 are invariant to the architecture of our neural network. To this end, we re-calculate the fisher distance of each series to the source dataset 0 and re-evaluate its correlation with model performance using an LSTM model. Results shown in Figure 8 and Figure 9 show that the behavior of the fisher distances are invariant to our model architecture selection. As a result, in the remaining part of this study, we will stick to the CNN model.

Apart from checking the stability in terms of model selection, we want to examine whether the behavior of the distances are consistent regardless of the choice of the source dataset. In other words, we will present similar results in Figure 6, by replacing the source dataset 0 with each of the dataset $i$ separately. Results are shown in Figure 10. This figure shows that the fisher distance is robust in each graph, as the position of the lowest point coincides with the index of the dataset used for training. However, the fisher distance cannot always perfectly represent the intuitive order, as

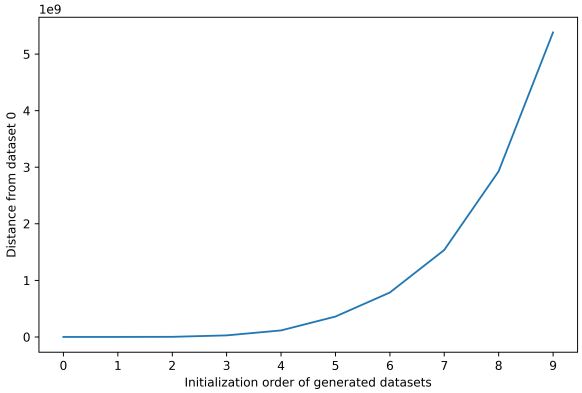


Figure 8. Fisher distance from source dataset 0 using LSTM

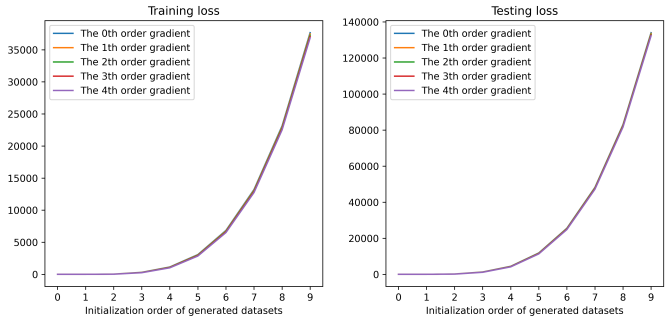


Figure 9. Gradient update on synthetic datasets using LSTM

we observe that in some graphs, the distance may remain similar or decrease even though a dataset is intuitively further from the source.

**4.1.3. Experiment 2: Few-Shot Transfer Learning.** In this experiment, we will implement few-short learning by transferring knowledge learned from a source dataset with sufficient data onto a target dataset that suffers data scarcity. Thus, in this section, we will treat the down-sampled dataset 0 as the target dataset and datasets 1 to 9 as potential source datasets.

**Down sample the target dataset.** To manually induce data scarcity, we perform downsampling on dataset 0 by increasing the stride of window sliding when constructing the train and test set. By increasing the number of stride, the total number of window chunks will naturally decrease. We increase the stride from 1 to 200, 100, and 20 and obtain three down-sampled datasets correspondingly. The resulting number of windows should be 'int(len(dataset0)-windowsize)) / stride'. The constructed numbers of windows are 50,100 and 500. The full dataset with 10000 windows is also presented for reference. Again, we split the the down-sampled data into a train and a test set with 9-1 ratio.

**Model selection.** We first use the same RNN architecture as in experiment 1 with a learning rate of 0.001 to train 9 converged RNN models on synthetic datasets 1 to 9. For each of the down-sampled dataset 0, we would select the best model among these 9 source models to conduct transfer learning. We used 2 metrics to select the best model, one is the fisher distance between the source dataset and the trainset of down-sampled datasets, and another is the performance (MSE loss) of the source model on the trainset of the down-sampled data. Figure 11 shows the results when using fisher distance as a metric. The corresponding best model for all down-sampling datasets is model 1 if we choose the model that is "fisher-closest" to down-sampled datasets. However, when checking the model compatibility based on the MSE loss on the down-sampled trainset, the results change. As shown in 18, it turns out that the models selected for 3 down-sampled datasets

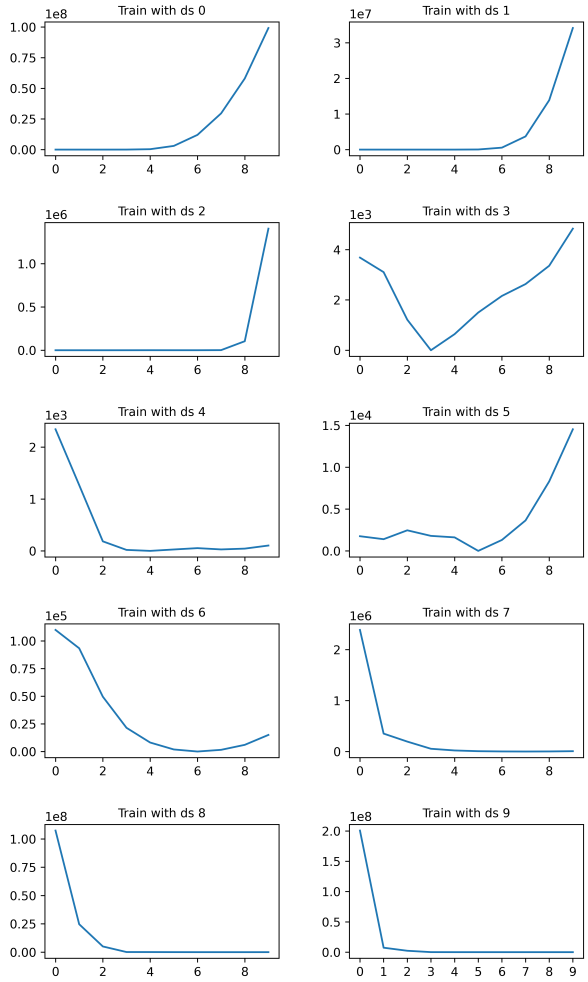


Figure 10. Fisher distance from synthetic dataset $i$ and other datasets

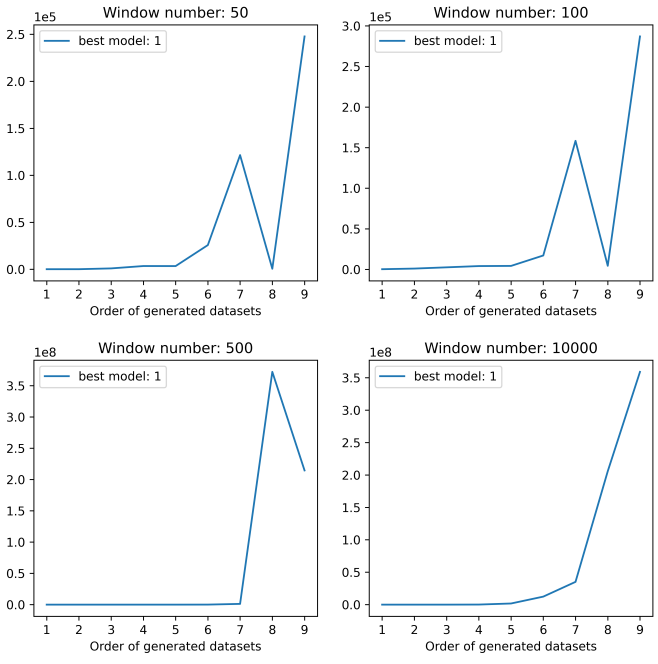


Figure 11. Fisher distance on downsampled dataset

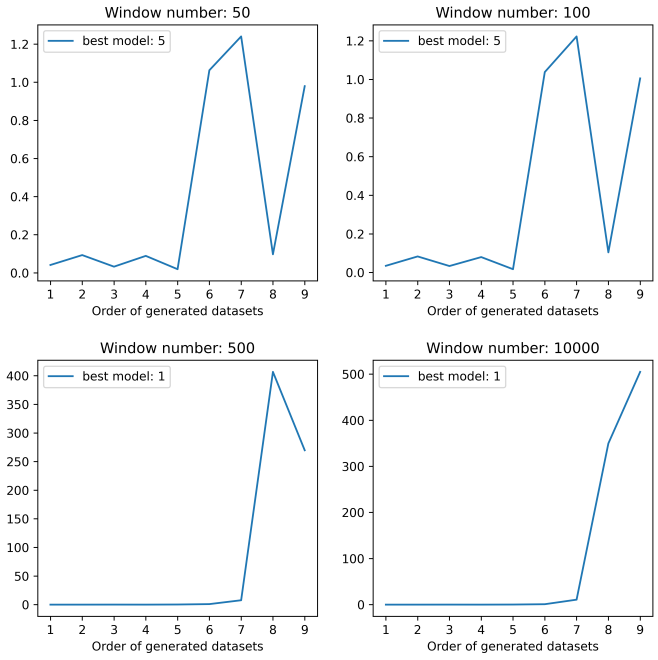


Figure 12. MSE loss on downsampled dataset

TABLE 1. TRAIN LOSS ON MODEL SELECTED BASED ON FISHER DISTANCE (SYNTHETIC)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 0.015290 | 0.012908 | 0.012585 |
| 100 | 0.014174 | 0.011287 | 0.011140 |
| 500 | 0.013858 | 0.013425 | 0.013384 |
| 10000 (full) | 0.015660 | 0.013886 | 0.013488 |

are model 5, 5, 1 respectively, which does not accord with our expectation because dataset 1 is intuitively viewed as the closest one to dataset 0. One point that should also be highlighted is that the model selected using fisher distance is consistent and aligns with one selected using full dataset 0. It indicates that fisher distance probably could capture some dataset characteristics that are invariant with the size of the dataset. This suggests that fisher distance might be a more reliable metric when choosing source models compared to the MSE loss.

**Transfer Learning.** To implement transfer learning, we perform gradient updates on the selected source model with the down-sampled target dataset. We perform 100 steps of gradient update, with each step using the entire trainset of the down-sampled data, and record the loss after $20^{th}$, $50^{th}$, and $100^{th}$ update. Table 1 and 2 report the train and test loss on models selected by fisher distance. Table 3 and 4 are the ones from MSE loss. Here, the last row serves as a benchmark that we will use to evaluate the efficiency of few shot learning.

Two patterns in these four tables are worth mentioning. First, both the train and test losses decrease as the number of gradient updates increases. Second, generally, for the same number of gradient updates, losses are lower with more data points used.

TABLE 2. TEST LOSS ON MODEL SELECTED BASED ON FISHER DISTANCE (SYNTHETIC)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 0.032710 | 0.026634 | 0.025394 |
| 100 | 0.026598 | 0.018409 | 0.018188 |
| 500 | 0.010673 | 0.010648 | 0.010821 |
| 10000 (full) | 0.016741 | 0.016682 | 0.013588 |

TABLE 3. TRAIN LOSS ON MODEL SELECTED BASED ON MSE LOSS (SYNTHETIC)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 0.134949 | 0.020281 | 0.016565 |
| 100 | 0.138783 | 0.016078 | 0.013744 |
| 500 | 0.013858 | 0.013425 | 0.013384 |
| 10000 (full) | 0.015660 | 0.013886 | 0.013488 |

TABLE 4. TEST LOSS ON MODEL SELECTED BASED ON MSE LOSS (SYNTHETIC)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 0.125056 | 0.035502 | 0.023869 |
| 100 | 0.122836 | 0.022382 | 0.014879 |
| 500 | 0.010673 | 0.010648 | 0.010821 |
| 10000 (full) | 0.016741 | 0.016682 | 0.013588 |

Importantly, there are two observations suggesting few shot learning is efficient with fisher task affinity. First, by comparing Table 2 and Table 4, we found that by selecting models using fisher distance, we can achieve substantially smaller test losses compared to using MSE as the selection criterion when the number of data points and epochs are relatively smaller. This is consistent with our previous finding that fisher distance could select more reliable source models given limited data points from the target task. Second, observing the last column in Table 2, we found that with a moderate number of gradient updates, even datasets with limited available data could achieve a loss that is comparably low as if we had access to the full data. These results conclude that the fisher task affinity score is very effective in implementing few-shot transfer learning on time series datasets even in data scarcity scenarios.

## 4.2. Results on Weather Data

In this section, we applied the simulation study to a set of weather time series.

**4.2.1. Weather Data Construction.** Series in this section are the hourly dew point temperature recorded in 10 weather stations in 10 different states, including CA, KY, MD, MI, NC, NY, SC, VA, WA, and WI. The hourly dew point temperature data is extracted from the Local Climatological Data (LCD) downloaded from National Centers for Environmental Information. The selected period of our data is from 2013-11-20 to 2022-11-20. In data preprocessing, we keep data in date where all the 24 hourly temperatures are available in all states. We then further select the first 10,000 data points in each of the 10 series for this application. In this application, we treat the dew point temperature in NC as the source data, and we ordered the remaining 9 series by their intuitive climate similarity to NC's. The intuitive order is SC, VA, KY, MD, NY, MI, CA, WA, WI, meaning that NC's climate is most similar to SC's and most different from WI's. Figure 4 plots the first 1,500 observations of these 10 time series datasets.

**4.2.2. Experiment 1: Fisher Task Affinity.** As in the simulation study, we first train the base model using NC's data, with 9-1 train-test split. The model architecture and parameters are the same as the RNN used in the simulation study. The predicted value from this trained source model is shown in Figure 14. The fisher distance from source dataset NC to datasets in the other 9 states is then evaluated based on the source model, which is shown in Figure 15. As expected, the fisher distances evaluated with data from states that have a similar climate to NC are smaller.

We then update the model trained with NC data using data from other states. For each of the 9 remaining series, we perform

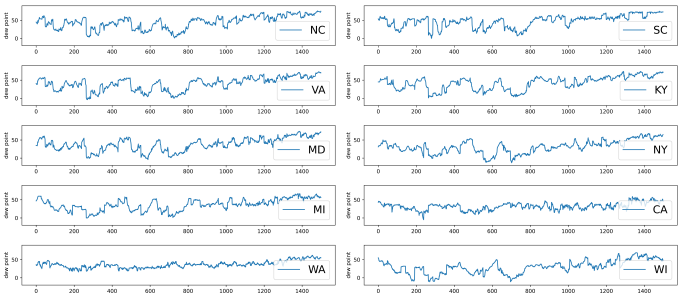


Figure 13. Weather data (first 1500 observations only)

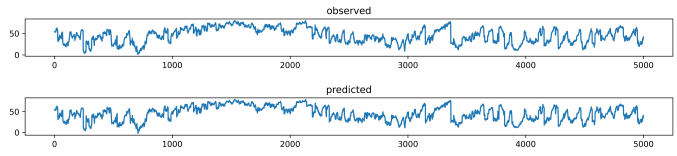


Figure 14. Dataset 0 and its predicted value using RNN (first 5000 period only)

4 steps of gradient update and record the loss in each update. Results are shown in Figure 16. Ideally, the loss evaluated on each state data should increase in the intuitive order of climate dissimilarity and decreases as we perform more steps of gradient update. As in the simulation study, this should give us a set of parallel monotonically increasing lines. However, this is not the case for this weather data application, which is possibly due to the fact that noise takes a great part in the weather data.

**4.2.3. Experiment 2: Few-shot Transfer Learning.** Same as one we conduct in the simulation study, we first down-sample the source NC data using the same method, and then select the best model for each of the down-sampled datasets using MSE loss and fisher distance. The selection results using fisher distance are shown in 17, and the ones using MSE loss are shown in 18. Surprisingly, though all three down-sampled datasets are all NC data, the selected models are all different, and even states (MI, WI) that have very different climates are selected. One possible explanation could be that when we down-sample the NC data, the selected window chunks happen to match the climate pattern of these states. Table 5-8 report the train and test loss from models selected by fisher distance and MSE loss. Since the model selected in this weather application is problematic, the resulting loss are therefore very unreasonable. For instance, the loss in the most sparse down-sampled dataset gives the lowest error.

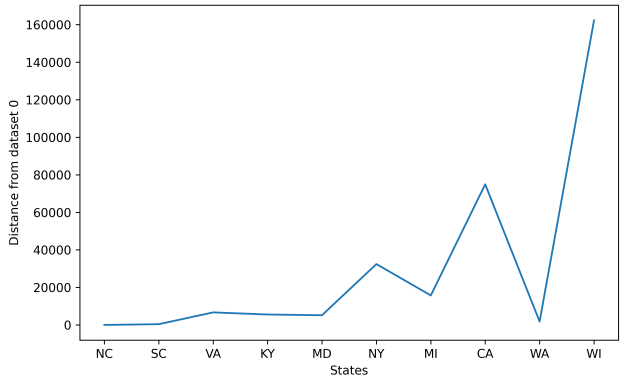


Figure 15. Fisher distance from NC dataset using RNN

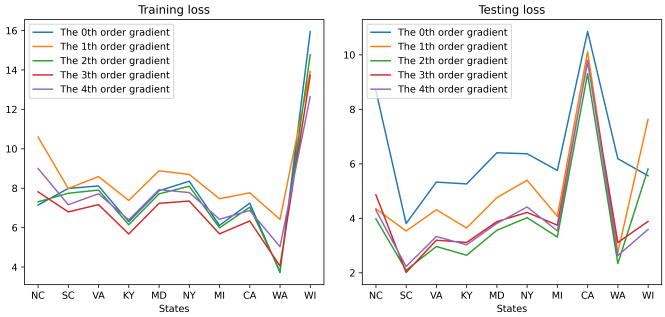


Figure 16. Gradient update on weather datasets using RNN

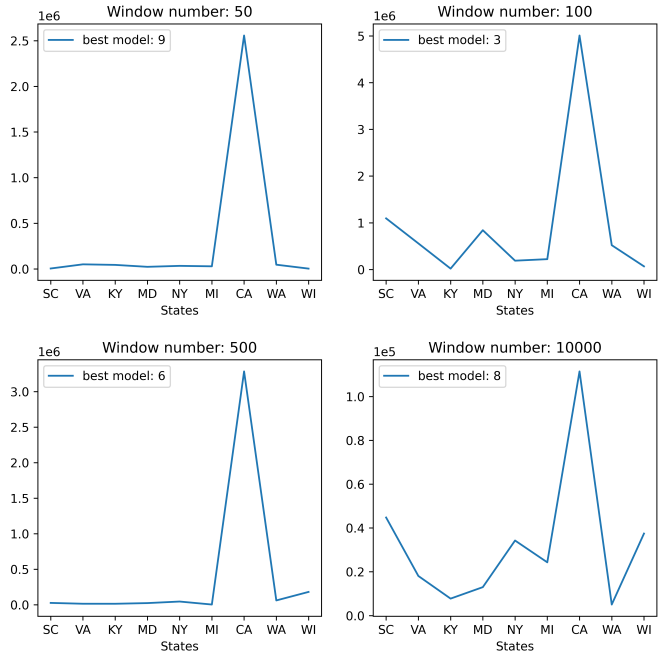


Figure 17. Fisher distance on downsampled NC weather dataset

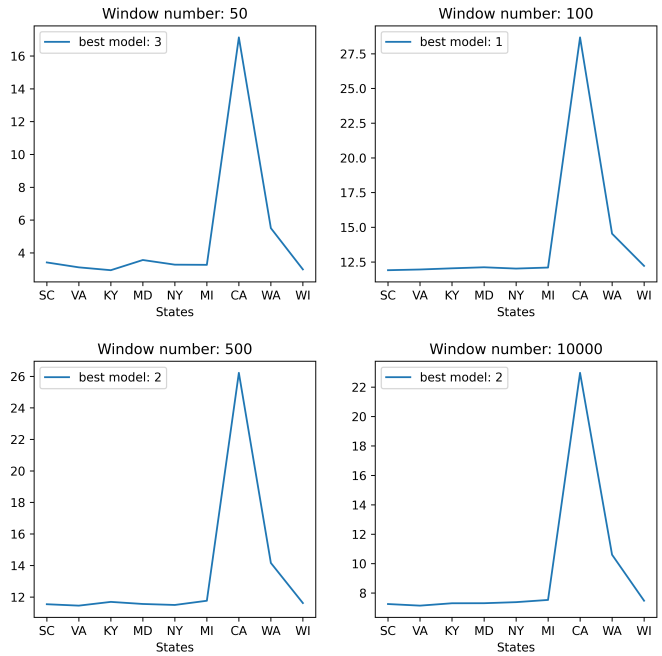


Figure 18. MSE loss on downsampled NC weather dataset

TABLE 5. TRAIN LOSS ON MODEL SELECTED BASED ON MSE LOSS (WEATHER)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 2.535975 | 2.327906 | 2.304632 |
| 100 | 12.055164 | 11.801847 | 11.793556 |
| 500 | 11.187809 | 11.018916 | 10.996233 |
| 10000 (full) | 7.907366 | 7.192274 | 7.071080 |

TABLE 6. TEST LOSS ON MODEL SELECTED BASED ON MSE LOSS (WEATHER)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 0.514099 | 0.562218 | 0.600461 |
| 100 | 4.474498 | 3.703633 | 3.688567 |
| 500 | 8.951347 | 8.755836 | 8.804464 |
| 10000 (full) | 4.846540 | 3.802727 | 3.768546 |

TABLE 7. TRAIN LOSS ON MODEL SELECTED BASED ON FISHER DISTANCE (WEATHER)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 2.492070 | 2.295393 | 2.265540 |
| 100 | 11.901271 | 11.724550 | 11.710960 |
| 500 | 11.288380 | 11.048060 | 11.027225 |
| 10000 (full) | 7.907366 | 7.192274 | 7.071080 |

TABLE 8. TEST LOSS ON MODEL SELECTED BASED ON FISHER DISTANCE (WEATHER)

| Number of Windows | Number of Gradient Update | | |
|---|---|---|---|
| | 20 | 50 | 100 |
| 50 | 0.431817 | 0.520370 | 0.507439 |
| 100 | 4.168914 | 4.280393 | 4.326311 |
| 500 | 8.885003 | 8.749310 | 8.691119 |
| 10000 (full) | 4.846540 | 3.802727 | 3.768546 |

## 5. Concluding Remarks

In this paper, we apply fisher distance measurement to time series data and successfully recover intuitive ordering on two distinct datasets, synthetic and real weather data using an RNN architecture.

We examine the ability and stability of fisher distance on time series data and conclude that it is well applicable to time series datasets. Neither the change of dataset composition nor the transition of architecture will affect the veracity of fisher distance as a metric in transfer learning.

We further test the ability of fisher distance as a selection criterion in finding the optimal source model in transfer learning by comparing it with using simple MSE test loss. We discover that fisher distance is more consistent and give more valuable result on few-shot transfer learning.

## References


[1] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[2] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[4] Cat Phuoc Le, Juncheng Dong, Mohammadreza Soltani, and Vahid Tarokh. Task affinity with maximum bipartite matching in few-shot learning. In *International Conference on Learning Representations*, 2022.

[5] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[6] P. Persson. Starved neural learning : Morpheme segmentation using low amounts of data: Semantic scholar, 2018.

[7] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[8] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

[9] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, 9(1):3–21, 2021.

[10] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.