Few-Shot Learnable Augmentation for Financial Time Series Prediction under Distribution Shifts

Anonymous Author(s)
Affiliation
Address
email

Abstract

We address the problem of distribution shift in financial time series prediction, where the behavior of the time series changes over time. Satisfactory performance of forecasting algorithms requires constant model recalibration or fine-tuning to adapt to the new data distribution. Specifically, the ability to quickly fine-tune a model with only a few training samples available from the new distribution is crucial for many business applications. In this paper, we develop a novel method for learnable data augmentation that effectively adjusts to the new time series distribution with only a few samples. We demonstrate the effectiveness of our method compared to the state-of-the-art augmentation methods on both univariate time series (e.g., stock data) and multivariate time series (e.g., yield rate curves) in the presence of distribution shift due to the COVID market shock in 2020.

1 Introduction

Time series prediction is the task of classifying or categorizing sequential inputs to gain further insight into their behavior, with important applications in multiple domains such as weather forecasting, medical diagnosis as well as financial prediction. As our society evolves continuously, financial data is prone to distribution shifts over time, where the time series dynamics deviate from previous patterns. Time series models trained with past data are no longer effective on current data. Similarly, it is common in practice to have wider access to the time series data for higher liquidity assets – and it is sometimes necessary to adapt models trained for highly liquid assets to low liquidity ones with a small number of data samples. To address the above distribution shift challenges, we focus on a setup of few-shot fine-tuning where a model can be quickly re-calibrated using only a few data points.

Related Work. There are three common types of distribution shifts in supervised learning corresponding to whether the changes in distributions happen to the input samples, referred to as covariate shifts [1, 2, 3], or to the outputs, label/concept shifts [4, 5, 6, 7]. Recently, [8] proposes a new categorization framework to enable more fine-grain analysis on distribution shifts. To address learning with distribution shifts, domain generalization works [9, 10] construct a model that is robust to a wide range of distributions. [11] leverages adversarial learning on a few samples in target distribution for domain adaptation. However, these methods cannot synthesize additional samples in target distribution for fast model fine-tuning with limited data, especially in time series domain. Although [12, 13, 14] introduces various time series augmentation methods, they are not designed for distribution shifts, thus they are unable to transfer knowledge from source to target distribution.

Contributions. We develop an augmentation framework to synthesize multiple variants of time series samples in order to facilitate few-shot model fine-tuning. Our contributions are as follows:

Submitted to Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

- We propose a learnable augmentation technique based on an autoencoder for time series.
- We design our method for few-shot model fine-tuning where a model is pretrained on many samples from a source distribution and updated with limited samples from a target distribution.
- We demonstrate the effectiveness of our method on both univariate and multivariate time series data for stock and yield rate curve predictions, respectively.

2 Few-shot Learnable Augmentation for Distribution Shifts

2.1 Problem Setting

Let $\mathcal{D}_s = \{(\boldsymbol{X}_s, y_s) | \boldsymbol{X}_s \sim P_s\}$ be the training set from a source distribution, with $\boldsymbol{X}_s \in \mathbb{R}^{f \times n}$ the input time series having t time steps, each with dimension f, and y_s its ground-truth label. In addition, $\mathcal{D}_t = \{(\boldsymbol{X}_t, y_t) | \boldsymbol{X}_t \sim P_t\}$ is data from a target distribution, which is different from the source distribution in terms of time series $\boldsymbol{X}$ (covariate shift [1, 2, 3]) or labels y (label shift [4, 5]). In this work, we mainly focus on covariate shifts in time series, i.e., temporal shifts with different time series distributions $P_t \neq P_s$. Specifically, we are interested in the problem of few-shot learning with distribution shifts where only limited training samples are available in target distribution, $|\mathcal{D}_t| \ll |\mathcal{D}_s|$. The goal is to transfer knowledge from the source distribution, $\mathcal{D}_s$, to the target distribution, $\mathcal{D}_t$, to learn a classifier that generalizes well to the target distribution P_t.

Due to the small number of samples in the target distribution, $\mathcal{D}_t$, simply training a classifier on these few samples would be prone to overfitting, as shown in the experimental section. Thus, we propose a novel time series augmentation technique to diversify training data in the target distribution.

2.2 Proposed Method

To address the distribution shifts between source and target data, we introduce a learnable augmentation method based on Δ-encoder [15]. We review this method and then improve upon the original design by proposing latent code perturbation and augmentation re-labeling for temporal shifts.

Background: Δ-encoder. Instead of using heuristic augmentation to synthesize new samples, [15] proposes a learnable data augmentation by capturing the inner-class variances of samples. They leverage an autoencoder architecture to encode the transformation from one sample to another into a latent code and reuse these codes to augment new samples. Specifically, let $(\boldsymbol{X}_s, \boldsymbol{X}_{s'})$ be a pair of source distribution samples from the same class, $y_s = y_{s'}$. An encoder, E, aims to capture the transformation from $\boldsymbol{X}_s$ to $\boldsymbol{X}_{s'}$ into a latent code as:

$$E(\boldsymbol{X}_s, \boldsymbol{X}_{s'}) = \boldsymbol{z}_{s \to s'}, \quad (1)$$

where $\boldsymbol{z}_{s \to s'} \in \mathbb{R}^d$ is a low-dimensional latent vector encoding the transformation from sample s to s'. Given the latent code, a decoder, D, is trained to reconstruct sample s' given s:

$$D(\boldsymbol{X}_s, \boldsymbol{z}_{s \to s'}) = \hat{\boldsymbol{X}}_{s'}. \quad (2)$$

Both the encoder, E, and decoder, D, are trained end-to-end by minimizing the l_1 reconstruction loss between decoder's output and original sample as $|\boldsymbol{X}_{s'} - \hat{\boldsymbol{X}}_{s'}|_1$. Thus, the encoder learns to capture class-invariance transformation in the source distribution.

Δ-encoder extracts latent code from source distribution pairs and applies these codes on few samples from target distribution to synthesize new data as: $D(\boldsymbol{X}_t, \boldsymbol{z}_{s \to s'}) = \hat{\boldsymbol{X}}_t^{s \to s'}$. Although this improves performances on few-shot learning where both train and test data are from the same distribution but different classes, it is ineffective when dealing with covariate shifts. Specifically, when $P_s \neq P_t$, augmenting $\boldsymbol{X}_t$ with latent code $\boldsymbol{z}_{s \to s'}$ will construct a new sample $\hat{\boldsymbol{X}}_t^{s \to s'}$ that follow P_s but not the target distribution P_t. To address this, we introduce a novel latent code perturbation scheme that conditions the latent codes on the target distribution, D_t, to capture the target distribution, P_t.

Latent Code Perturbation. Instead of relying on the latent codes from the source distribution, $\boldsymbol{z}_{s \to s'}$, we extract latent codes from the target distribution samples from the same class, $y_t = y_{t'}$:

$$E(\boldsymbol{X}_t, \boldsymbol{X}_{t'}) = \boldsymbol{z}_{t \to t'}, \quad (3)$$

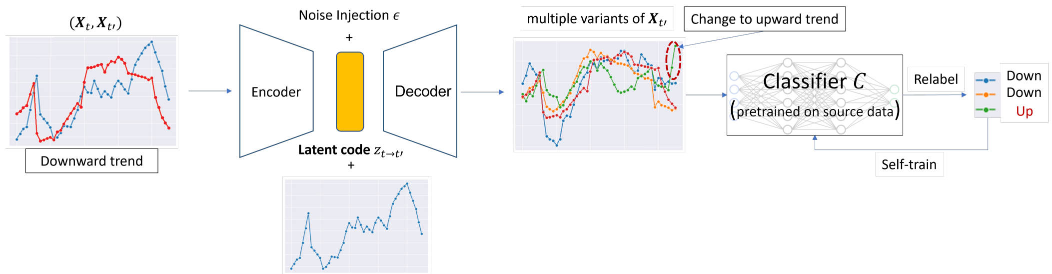

Figure 1: Given a pair of sequences $(\boldsymbol{X}_t, \boldsymbol{X}_{t'})$ in the target distribution, our framework extracts a latent vector $z_{t\to t'}$ which captures the transformation from $\boldsymbol{X}_t$ to $\boldsymbol{X}_{t'}$. By slightly perturbing this latent vector, we can synthesize multiple variants of $\boldsymbol{X}_{t'}$. Finally, we use a classifier C pretrained on source data to re-label augmented samples to capture any chances in label semantics.

where $\boldsymbol{z}_{t\to t'}$ is a latent code in target distribution. Naively using this code on $\boldsymbol{X}_t$ would simply reconstruct the original data $\boldsymbol{X}_{t'}$ without diversifying the training set. Thus, we propose to slightly perturb the latent code based on random noise, ϵ, as:

$$D(\boldsymbol{X}_t, \boldsymbol{z}_{t\to t'} + \epsilon) = \hat{\boldsymbol{X}}^{\epsilon}_{t'}, \tag{4}$$

where $\hat{\boldsymbol{X}}^{\epsilon}_{t'}$ is the augmented variant of $\boldsymbol{X}_{t'}$ based on random noise ϵ. By using perturbed latent code instead of transferring latent code from the source distribution as in [15], our method effectively captures the target distribution and avoids overfitting to latent code from the source distribution.

Augmentation Re-labeling for Temporal Shifts. As we randomly perturb the latent code, the label of the augmented sample $\boldsymbol{X}^{\epsilon}_{t'}$ might be changed compared to its original label of $\hat{\boldsymbol{X}}_{t'}$, e.g., form downward to upward trends, due to the non-interpretability of the augmentation operation, $D(\cdot)$ as shown in Figure 1. Thus, we propose re-labeling the augmented sample to account for any semantic changes during the augmentation progress. We train a classifier, C, on source distribution $\mathcal{D}_s$ and assign its most confident prediction on an augmented sample as its new label:

$$\text{argmax}_y\, C(y|\hat{\boldsymbol{X}}^{\epsilon}_{t'}) = \hat{y}^{\epsilon}_{t'}, \tag{5}$$

where $\hat{y}^{\epsilon}_{t'}$ is the new label for augmented sample $\hat{\boldsymbol{X}}^{\epsilon}_{t'}$. Here, we assume that temporal shift only effects the series distribution $\boldsymbol{X}$ while the conditional label distribution $P(y|\boldsymbol{X})$ remains unchanged similar to [1, 2, 3].

Learning with Mixture of Real and Augmented Samples. Finally, we fine-tune the classifier, C, on the mixture of both real and augmented samples to adapt it to the target distribution as follows:

$$\min_{C} \sum_{(\boldsymbol{X}_t, \boldsymbol{X}_{t'}, y_t)\in\mathcal{D}_t} \left[\mathcal{L}(C(\boldsymbol{X}_t), y_t) + \lambda\mathcal{L}(C(\hat{\boldsymbol{X}}^{\epsilon}_{t'}), \hat{y}^{\epsilon}_{t'})\right], \tag{6}$$

where λ is the mixture coefficient that controls the influence of augmented samples on the classifier. The larger λ is, the more emphasis we put on augmented samples.

Remark 1 *Unlike prior work [12, 14], which cannot share knowledge between source and target distributions, our method combines latent codes from target distribution samples and the decoder pretrained on source distribution to effectively transfer knowledge between distributions.*

3 Experiments

We evaluate our proposed framework on the forecasting task of stock trend prediction for univariate time series, and yield rate curves prediction for multivariate time series. We present both quantitative and qualitative results to demonstrate the effectiveness of our method. For further information about datasets, baselines, and implementation details, please refer to the supplementary material section 5.

3.1 Experimental Results

Univariate time series Prediction: Stock Price. Table 1 shows the performances of different augmentation methods on stock data. Given very few training samples of 10, 20 and 30 shots, our

k-shot	Baseline Augmentation				No Augmentation		Our Proposed Augmentation			
	Gaussian Jitter [12]	Time Warp [12]	RGW [14]	DGW [14]	Pretrain $\mathcal{D}_s$	Fine-tune $\mathcal{D}_t$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 2.0$
10	77.6 + 5.8	74.8 + 9.7	77.1 + 9.8	72.9 + 9.7	75.5 + 14.6	83.8 + 6.2	81.2 + 6.4	**84.2 + 5.9**	83.7 + 7.1	83.6 + 6.0
20	72.6 + 12.0	69.5 + 10.6	73.4 + 13.7	73.3 + 14.5	78.2 + 12.5	77.8 + 10.5	74.0 + 12.8	81.2 + 7.9	80.6 + 7.7	**82.3 + 7.0**
30	83.6 + 3.6	83.3 + 5.8	79.6 + 9.3	74.4 + 13.1	81.9 + 9.9	78.2 + 11.7	83.7 + 6.5	**85.0 + 5.5**	79.8 + 8.3	79.0 + 12.1
40	85.0 + 2.9	**87.4 + 1.3**	84.8 + 4.3	82.5 + 4.9	84.7 + 1.0	83.5 + 6.7	86.1 + 5.4	85.5 + 5.6	86.2 + 5.1	86.6 + 4.9
50	85.6 + 1.9	85.2 + 2.8	**86.5 + 1.0**	85.3 + 2.0	83.8 + 6.8	83.8 + 5.9	84.0 + 6.7	84.8 + 6.7	85.1 + 6.5	85.4 + 6.3
60	86.2 + 2.3	**87.5 + 1.0**	85.7 + 1.5	84.7 + 1.9	85.1 + 7.6	86.6 + 6.4	86.7 + 6.9	87.3 + 6.4	86.5 + 6.3	86.3 + 6.1

Table 1: Stock trend prediction performances (mean + standard deviation) for every 6-month period after 2020. **Bold** and underline indicate best and second best performances, respectively.

k-shot	Baseline Augmentation				No Augmentation		Our Proposed Augmentation			
	Gaussian Jitter [12]	Time Warp [12]	RGW [14]	DGW [14]	Pretrain $\mathcal{D}_s$	Fine-tune $\mathcal{D}_t$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 2.0$
10	47.0 + 18.8	47.8 + 19.8	47.8 + 18.6	46.3 + 17.8	**58.0 + 14.1**	47.0 + 22.3	53.0 + 15.1	52.0 + 16.2	56.3 + 13.4	52.2 + 18.9
20	44.5 + 18.1	43.8 + 20.7	41.9 + 17.3	43.8 + 18.3	54.5 + 14.2	44.8 + 23.0	42.4 + 21.8	53.3 + 15.1	49.3 + 19.4	**59.0 + 15.1**
30	52.9 + 16.9	53.4 + 15.7	53.4 + 17.7	51.6 + 14.8	56.1 + 10.0	47.6 + 18.9	46.3 + 18.2	60.8 + 11.5	53.4 + 15.2	**73.4 + 12.5**
40	65.0 + 16.2	62.6 + 17.5	62.1 + 18.8	63.8 + 18.5	50.9 + 13.1	60.9 + 10.2	70.0 + 6.4	**71.8 + 12.9**	61.5 + 11.6	70.0 + 11.7
50	61.3 + 17.7	67.7 + 11.9	70.0 + 10.9	66.0 + 12.4	49.0 + 14.5	60.3 + 7.1	71.0 + 2.6	64.7 + 5.0	72.3 + 9.1	**73.7 + 4.0**
60	50.8 + 15.3	64.2 + 18.1	60.0 + 21.7	59.2 + 8.5	50.0 + 16.0	60.4 + 2.3	55.8 + 7.3	53.1 + 17.1	**68.5 + 8.5**	57.3 + 13.1

Table 2: Yield rate trend prediction performances (mean + standard deviation) for every 6-month period after 2020. **Bold** and underline indicate best and second best performances, respectively.

method surpasses prior works by at least 0.4%, 3%, and 1.7% accuracies, respectively, with $\lambda = 0.5$ which demonstrates the effectiveness of our method when dealing with very few numbers of target distribution samples. Without leveraging our method, we observe that simply fine-tuning a classifier on few-shot data offers no significant improvement compared to no fine-tuning. Notice our work can improve performances on a wide range of training shots (including those with a small number of shots), while prior augmentation methods only work for a larger number of shots (when there are at least 40 training shots in our example).

Multivariate time series Prediction: Yield Rate Curve. Table 2 presents yield rate trend performances. With just 20, 30, and 40 training samples, our method significantly improves accuracy by 4.5%, 17.3%, and 5%, respectively, compared to other augmentation with $\lambda = 2.0$. With less than 30 training shots, fine-tuning the classifier even degrades prediction accuracies with respect to only pretrained the classifier, which shows the challenges of few-shot model fine-tuning without overfitting on limited target distribution samples. Without data augmentation, we observe a performance gap of around 10% between only pre-training on $\mathcal{D}_s$ and fine-tuning on $\mathcal{D}_t$ for $k \geq 40$. This demonstrates that the distribution shifts can cause the model pretrained on $\mathcal{D}_s$ to be ineffective on $\mathcal{D}_t$.

Qualitative Results. We visualize the distribution of augmented and real samples using t-SNE [16]. When overlaying the distribution of few-shot samples from $\mathcal{D}_t$ and synthetic samples from our augmentation model in Figure 2 (a), we observe that the augmented samples generalize beyond few-shot samples used to synthesize them thanks to our decoder which transfers knowledge from source to target distributions. To validate the effectiveness of augmented samples, Figure 2 (b) shows the distributions of all target samples and synthetic samples where the augmented samples manage to capture the target distribution.

Without augmentation re-labeling, we find that our method could not improve performances for both datasets compared to fine-tuning on $\mathcal{D}_t$, due to the label changes when augmenting time series.

4 Conclusion

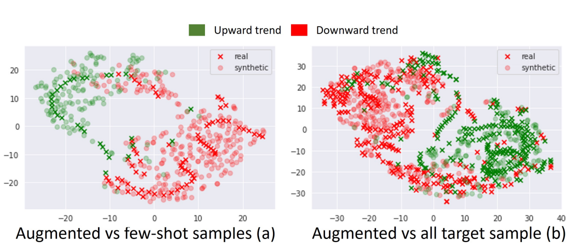

Figure 2: t-SNE visualization of augmented (synthetic) and real Delta airline stocks in the target distribution.

We propose a novel method that learns to augment samples for the problem of few-shot model fine-tuning under distribution shifts. Our work aims to encode a class-agnostic transformation between inner-class samples into a compact latent code. During inference, we perturb the latent code to simulate different augmentations on a few samples in the target distribution. When combined with augmentation re-labeling, our method significantly improves model fine-tuning performances for both univariate and multivariate time series in financial applications.

References

[1] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, 2007.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International Conference on Machine learning*, 2015.

[3] N. Tripuraneni, B. Adlam, and J. Pennington, "Overparameterization improves robustness to covariate shift in high dimensions," *Neural Information Processing Systems*, 2021.

[4] K. Azizzadenesheli, A. Liu, F. Yang, and A. Anandkumar, "Regularized learning for domain adaptation under label shifts," *International Conference on Learning Representations*, 2019.

[5] X. Liu, B. Hu, L. Jin, X. Han, F. Xing, J. Ouyang, J. Lu, G. E. Fakhri, and J. Woo, "Domain generalization under conditional and label shifts via variational bayesian inference," *International Joint Conference on Artificial Intelligence*, 2021.

[6] P. Vorburger and A. Bernstein, "Entropy-based concept shift detection," *Sixth International Conference on Data Mining*, 2006.

[7] J. Tian, Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Exploring covariate and concept shift for detection and calibration of out-of-distribution data," 2021.

[8] O. Wiles, S. Gowal, F. Stimberg, S.-A. Rebuffi, I. Ktena, K. Dvijotham, and A. T. Cemgil, "A fine-grained analysis on distribution shift," *International Conference on Learning Representations*, 2022.

[9] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *International Conference on Learning Representations*, 2021.

[10] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker, "Domain generalization via model-agnostic learning of semantic features," *Neural Information Processing Systems*, 2019.

[11] S. Motiian, Q. Jones, S. M. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," *Neural Information Processing Systems*, 2017.

[12] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. M. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, 2017.

[13] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLOS ONE*, 2021.

[14] ——, "Time series data augmentation for neural networks by time warping with a discriminative teacher," *International Conference on Pattern Recognition*, 2021.

[15] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. S. Feris, R. Giryes, and A. M. Bronstein, "Delta-encoder: an effective sample synthesis method for few-shot object recognition," *Neural Information Processing Systems*, 2018.

[16] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, 2008.

[17] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, 2020.

5 Supplementary Materials

5.1 Experimental Setup

Datasets. For univariate time series prediction, $f = 1$, we experiment with daily stock price trend forecasting. Specifically, to capture the distribution shifts in financial data, we select companies from the travel industry: Southwest Airlines, Delta Airlines, Carnival Cruise Line, and Royal Caribbean Group, which have dramatic changes in their stock price during the COVID lockdown in 2020. We use Yahoo! Finance's API[1] to crawl the data.

We illustrate multivariate time series prediction with the daily treasury yield dataset[2] - where the yield values are read from the yield curve at fixed maturities, 1, 3, and 6 months and 1, 2, 3, 5, 7, 10, 20, and 30 years. Time series for different maturities are known to be highly correlated.

For both stock and yield curve datasets, we construct the 30-day consecutive time-step sequences, $n = 30$, as input $\boldsymbol{X}$. We also perform mean subtraction and standard deviation division to normalize the range of these sequences. The ground-truth label is a binary indicator, $y \in \{0, 1\}$, for whether the sequence value will go down or up compared to the mean for the 31^{st} day. We select a training dataset from 2019 data, and test the models on the 2020 data to account for the distribution shift due to the COVID market shock.

Evaluation Metrics. To evaluate our performance, we split the target data into multiple non-overlapping windows of six months. Within each window, we use the first k sequences in each time window as $\mathcal{D}_t$ to construct augmented data as well as fine-tune the classifier. We measure the classification accuracy on the remaining sequences from the $k + 1$ day. We report the mean and standard deviation of prediction accuracies across all testing windows.

Baselines. For baselines, we use two popular data augmentation methods for time series data, namely jittering and time warp [12, 13]. Jittering consists of adding Gaussian noise element-wise to the time series while time warping randomly selects anchor points in the time series and smoothly distorts the time intervals between the points using a cubic spline curve. Additionally, we compare our model with two pattern mixing methods proposed for time series, Random Guided Warp (RGW) and Discriminative Guided Warp [14]. Both methods use Dynamic Time Warping (DTW), which determines an optimized distance measure for time series that is robust to temporal distortions. In RGW, time series are mixed by warping the features of an original sample pattern to match the time steps of a reference pattern, using DTW to create the warping path, with both elements within the same class. DGW introduces a discriminative teacher as a reference for guided warping.

Implementation Details. The augmentation model, $\{E, D\}$, and the classifier, C, are based on the InceptionTime architecture [17]. We use the Adam optimizer with a batch size of 32 and a learning rate of 1e-3 on 80 epochs for the augmentation model, with 10 epochs for classifier pre-training and another 5 epochs for fine-tuning. Empirically, we find that using a standard normal distribution to perturb the latent code $\epsilon \sim N(0, \boldsymbol{I})$ works best. Our code is released upon request.

[1] `https://pypi.org/project/yfinance/`

[2] `https://home.treasury.gov/interest-rates-data-csv-archive`