Reconstruction and Regression Loss for Time-Series Transfer Learning

Nikolay Laptev
Stanford University
Stanford, CA
nlaptev@stanford.edu

Jiafan Yu
Stanford University
Stanford, CA
jfy@stanford.edu

Ram Rajagopal
Stanford University
Stanford, CA
ramr@stanford.edu

ABSTRACT

Reliable and accurate time-series modeling is critical in many fields including energy, finance, and manufacturing. Many time-series tasks, however, suffer from a limited amount of clean training data resulting in poor forecasting, classification or clustering performance. Recently, convolutional neural networks (CNNs) have shown outstanding image classification performance even on tasks with small-scale training sets. The performance can be attributed to transfer learning through ability of CNNs to learn rich mid-level image representations. For time-series, however, no prior work exists on general transfer learning. In this paper, motivated by recent success of transfer learning in image-related tasks, we show its applicability for time-series and define a new architecture and a new loss function for time-series transfer learning that is able to outperform the baseline methods typically used in practice for transfer learning.

We demonstrate the effectiveness of our approach in many diverse domains.

ACM Reference Format:
Nikolay Laptev, Jiafan Yu, and Ram Rajagopal. 2018. Reconstruction and Regression Loss for Time-Series Transfer Learning. In *Proceedings of ACM Conference (SIGKDD MiLeTS'18).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Accurate time-series forecasting is critical for load forecasting, financial market analysis, anomaly detection, optimal resource allocation, budget planning, and other related tasks. While time-series forecasting has been investigated for a long time, the problem is still challenging, especially in applications with limited history (e.g., holidays, sporting events) where practitioners are forced to use adhoc machine learning approaches achieving poor forecasting performance [26].

Recently, time-series modeling based on a particular recurrent neural network, the Long Short Term Memory (LSTM) model [10], has gained popularity due to its end-to-end modeling, ease of incorporating exogenous variables, and automatic feature extraction abilities [1]. Inspired by the success of deep convolutional neural network (CNN), [9] use stacked LSTM cells with different weight matrices in different layers for text prediction; [8] use deep LSTM cells for speech recognition; [6] use deep LSTM and CNN for video recognition; [16] show that an LSTM forecasting model is able to outperform classical time series methods in cases with long, interdependent time series. The superior performance of deep LSTM structure on different tasks empirically prove its capability of modeling complex nonlinear feature interactions.

However, similar to training a deep CNN model, training a deep LSTM network needs updating the weight matrices for each LSTM cell, which requires a large amount of data across numerous dimensions. The data requirement hinders the application of deep LSTM model in time series forecasting. For example, recent results on time-series forecasting using LSTM only apply a single layer of LSTM [3].

Transfer learning [20] can address this problem. In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task. When the target dataset is significantly smaller than the base dataset, transfer learning can be a powerful tool to enable training a large target network without overfitting.

[2] shows preliminary results of using transfer learning on images. While the deep CNN models also suffer from requiring massive training data, [7, 11, 13, 19, 22–24, 28, 29] explore transfer learning on images, language and video. In image-based transfer learning [2], deep neural networks exhibit a curious phenomenon: when trained on images, they all tend to learn first-layer features that resemble either Gabor filters or color blobs [2]. The appearance of these filters is so common that obtaining anything else on a natural image dataset causes suspicion of poorly chosen hyperparameters or a software bug. This phenomenon occurs not only for different datasets, but even with very different training objectives, including supervised image classification [14], and unsupervised learning of sparse representations [17]. [29] quantitatively explore transferability of different layers for the image classification task. The authors discover that because finding the Gabor filters on the first layer seems to occur regardless of the exact cost function and natural image dataset, these first-layer features are called *general.* On the other hand, the features computed by the last layer of a trained network must depend greatly on the chosen dataset and task. For example, in a network with an N-dimensional softmax output layer that has been successfully trained toward a supervised classification objective, each output unit will be specific to a particular class. [29] thus calls the last-layer features *specific.* [7] applied transfer learning on CNN models which serves as an input of following RNN

SIGKDD MiLeTS'18, August 2018, London, United Kingdom
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

model for action recognition. Transfer learning has been introduced to language model works. For example, [22] discusses using variational RNNs to capture underlying temporal latent dependencies. [23] use pre-trained weights to initialize seq2seq language models. [24] used an RNN to generate data as the input of another RNN for language models. [28] proposed to use transfer learning by freezing several layers of RNNs for language modeling.

Motivated by these findings, we investigate if transfer learning applies to time series forecasting. Until now, the success of model generalization in deep CNN models for image-related tasks and in RNN models for language modeling still does not happen in deep LSTM models for time series-related tasks. Though [27] proposed to use transfer learning for sequence tagging problems, the sequence tagging problem can be explicitly decomposed into different sub-problems and the transfer learning is also explicitly divided. Transfer learning has also been recently used in language translation where an auto-encoder architecture is typically used [18]. Previous work has shown that an auto-encoder is useful in time-series for feature extraction [16], but not in time-series forecasting.

In this paper, we explore if there are equivalent *general* and *specific* features for time-series forecasting using a novel deep learning architecture, based on LSTM, with a new loss. We are interested in this, to the extent that features within a deep LSTM network are general, we will be able to use them for transfer learning to do more accurate forecasting on short time-series.

To the best of our knowledge, our work makes the first attempt to present the evidence of transfer learning for time-series in neural nets and to quantify its applicability to real-world applications. Our contributions are four-fold:

(1) We firstly demonstrate transfer learning for time-series forecasting.
(2) We define a new loss function and a new model architecture that improves the standard transfer learning technique and show its effectiveness for time-series.
(3) We demonstrate use-cases and impacts of time-series transfer learning, including:
  (a) forecasting with limited history,
  (b) computational resource saving,
  (c) cross-domain learning capability.
(4) We publish an online tool that democratizes time-series forecasting through a public transfer learning model motivated by ImageNet [14].

2 TRANSFER LEARNING IN TIME-SERIES

Transfer learning involves the concepts of a task and of a domain. A domain D consists of a marginal probability distribution $P(X)$ over the feature space $\mathcal{X} = \{x_1, ..., x_n\}$. Thus, given a domain $\mathcal{D} = \{\mathcal{X}, P(\mathcal{X})\}$, a task T is composed of a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ that is usually learned from training examples consisting of pairs $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Given a source domain $\mathcal{D}_S$ and a source task $\mathcal{T}_S$ as well as a target domain $\mathcal{D}_T$ and a target task $\mathcal{T}_T$, transfer learning aims to learn the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_T$ from the information learned from $\mathcal{D}_S$ and $\mathcal{T}_T$. In this paper we apply transfer learning to a time-series domain and apply it to cases where $\mathcal{X}_S \neq \mathcal{X}_T$ and $P(Y_S|X_S) \neq P(Y_T|X_T)$ (e.g., target domains with limited training data and different time-series classes).

Time-series data can be decomposed into the summation of a (usually monotonic) trend component, a (periodic) seasonal component, a holiday component, a stationary (stochastic) component, and a (usually i.i.d.) error component. Most of the time-series forecasting models assume explicit models with hyper-parameters of the trend, seasonality, and the stochastic process:

$$y(t) = g(t; \alpha_i^t) + s(t; \alpha_i^s) + h(t; \alpha_i^h) + c(t; \alpha_i^c) + \epsilon(t; \alpha_i^\epsilon),$$

where $g(\cdot)$, $s(\cdot)$, $h(\cdot)$, $c(\cdot)$, $\epsilon(\cdot)$ represent the trend, seasonal, holiday, stationary, and error component, respectively. Then, these models usually use maximum likelihood estimation to determine the hyper-parameters α. However, the explicit model formulation of each component is still far from empirical understanding. For example, [25] propose to use only a piecewise-linear function and a logistic growth model for trending modeling, and a Fourier series with trigonometric function as basis for seasonality modeling. Moreover, all hyper-parameters need to be optimized for specific data sets.

Instead of defining the explicit model components, the LSTM model only consists of 5 different nonlinear components. Initially, the LSTM cell is designed to repeat infinitely, with a single set of hyperparameters, including four rectangular weight matrices W_f, W_h, W_u, W_o, acting on input vector x, serving for computing forget gate, candidate state, update gate, and output gate, respectively. Four square weight matrices R_f, R_h, R_u, R_o, acting on lagged output vector y_{t-1}, serving for the same computation procedures. By stacking LSTM cells to construct a deep LSTM model, we gain additional freedom by enabling different weight matrices in different LSTM layers. We hypothesize that the feature layers of LSTM model is a generalization of the trend, seasonality, and holiday representation, in analogy to the explicit models in traditional time-series modeling tasks. While the model representation itself rather a general feature for all time-series data, we anticipate that the representation learned from one dataset can be used for another dataset. Then, the lower levels of the LSTM model serves the role in analogy to the hyper-parameter optimization which varies for different datasets. The illustration of the generalization is shown in Fig. 1.

After model training, a set of feature layers is typically *frozen* in order to avoid changing the learned weights. In this paper, we explicitly decompose the model into two types of layers: feature layers and predictive layers. After training, we typically *freeze* the feature layer weights to reuse them.

2.1 Model Loss & Architecture

The standard time-series modeling approach consists of a set of LSTM layers and the MSE cost function on the output layer. Our loss, however, consists of a combination of a regression loss and a reconstruction loss. The regression loss is a simple MSE loss while the reconstruction loss is defined as $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ and represents the likelihood that the input data would be reconstructed by the model. The motivation for this loss is to explicitly compute the modeling loss and forecasting loss separately. Figure 2 shows our architecture. Note that the first set of layers are fully connected and we pass the original input as well as the output from our

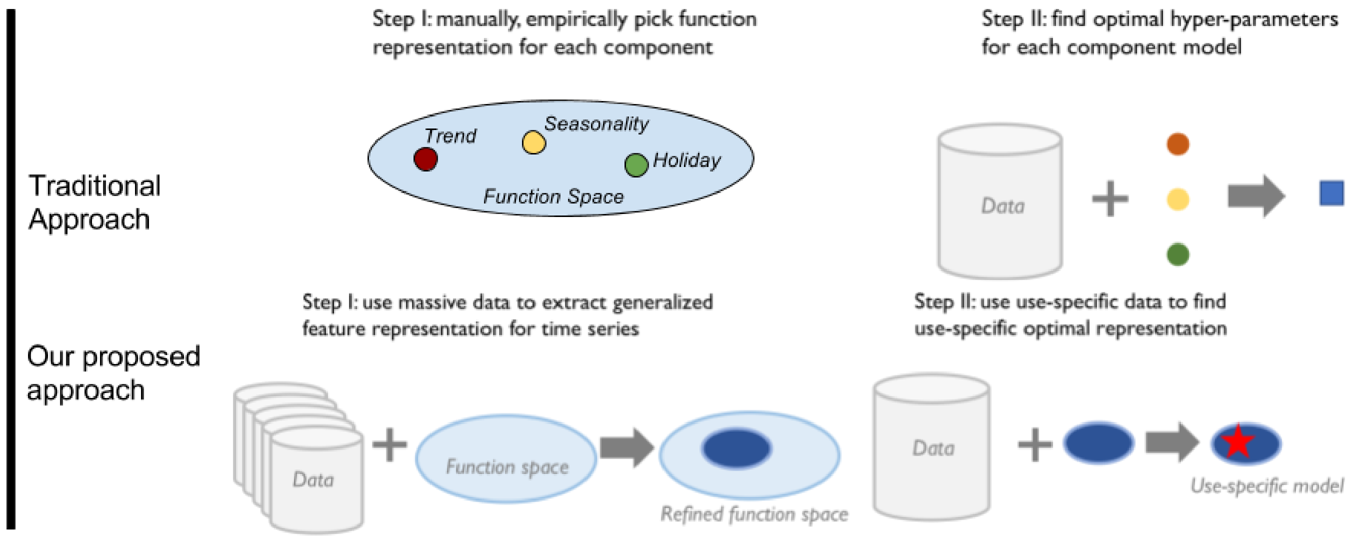

Step I: manually, empirically pick function representation for each component
Step II: find optimal hyper-parameters for each component model
Traditional Approach
Trend
Seasonality
Holiday
Function Space
Data
Step I: use massive data to extract generalized feature representation for time series
Step II: use use-specific data to find use-specific optimal representation
Our proposed approach
Data
Function space
Refined function space
Data
Use-specific model

Figure 1: Transfer learning visualization

reconstruction layers to the prediction layers which are LSTM-based. Specifically, our layer structure is as follows:

(1) L1 - Fully connected layers to extract time-series features
(2) L2 - Bottleneck layer where the reconstruction loss is computed
(3) L3 - A set of LSTM layers that as input take the output of a fully connected layer and the original input to perform prediction and compute the forecasting loss.

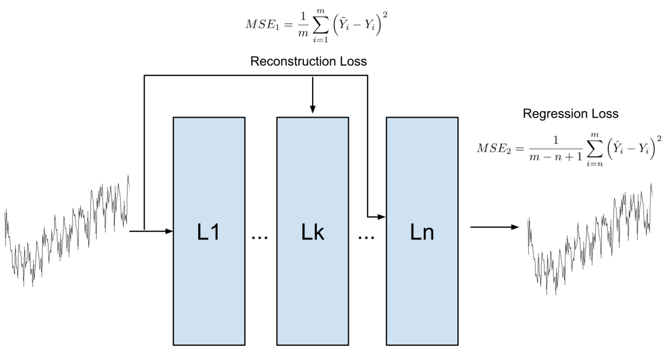

$MSE_1 = \frac{1}{m}\sum_{i=1}^{m}\left(\tilde{Y}_i - Y_i\right)^2$
Reconstruction Loss
Regression Loss
$MSE_2 = \frac{1}{m-n+1}\sum_{i=n}^{m}\left(\hat{Y}_i - Y_i\right)^2$
L1
...
Lk
...
Ln

Figure 2: Transfer learning visualization. The first loss function represents the reconstruction loss. The Y_i's are original time series. The $\tilde{Y}_i$'s are reconstructed time series, which is the input of the following forecasting module. The second loss function represents the forecasting loss. The Y_i's are original time series. The $\hat{Y}_i$'s are forecasted time series.

By combining the reconstruction with the regression loss we are able to separate the two functionalities, modeling and forecasting, into individual components and optimize them jointly. Note that while we observed an average of 20% improvement in MSE reduction compared to the standard LSTM model with a single MSE loss at the output layer, this paper focuses on transfer learning ability of our model and leaves the comparison against standard LSTM methods with various hyper-parameters for the longer version of the paper.

3 EXPERIMENTAL SETUP

Dataset: The dataset used for demonstrating transfer learning contains 116,000 anonymous residential scale electricity loads at hourly granularity from Pacific Gas and Electric Company (PG&E dataset). The data is from 08/01/2011 to 7/31/2012. The 116,000 time series are from 13 different climate zones and are with large diversity [15]. Besides the anonymous PG&E dataset, we also use a publicly available M3 dataset for validation. All time series are logarithm transformed and min-max scaled to [0, 1] interval.

Deep LSTM Model: The deep LSTM model consists of 3 fully connected layers and 6 LSTM cells. Each fully connected layer and an LSTM layer has a 128-dimensional state vector. For a given timestamp t (hour), the input vector $\boldsymbol{x}$ consists of 60-hour historical electricity load: $\boldsymbol{x} = [l_t, \cdots, l_{t-59}]$, and the output vector $\boldsymbol{y}$ consists of 60-hour electricity load from time t: $\boldsymbol{y} = [l_{t+1}, \cdots, l_{t+60}]$. We train the network using 60 hours for the forecast horizon and 60 hours for the lookback.

Data Separation: The PG&E dataset is separated into four parts for experiments. In particular, we randomly split the dataset into two subgroups, A and B, with same size. Then, we divide each subgroup into first six-month data and second six-month data. The four sub-groups of data is labeled with A1, A2, B1, and B2, shown in Table 1.

In the following experiments, the whole subset A1 is used to train the base deep LSTM model, called Base. Then, for any time series b^i in subset B, we use $b_1^i \in$ B1 for training, and $b_2^i \in$ B2 for testing. The transfer learning is implemented as: given n as the

	A (58k)	B (58k)
08/01/2011 - 01/31/2012	A1	B1
02/01/2012 - 07/31/2012	A2	B2

Table 1: Data separation

transferred layer from `Base`, we *freeze* the first n layers of the `Base` model, and use b_1^i to train the other $6 - n$ layers. If $n = 6$, then we only use the *specific* data b_1^i to train the last fully connected layer. We call the transfer learning model as `AnB` for a given n. We can also initialize the deep LSTM model with randomly chosen hyperparameters, and only use b_1^i to train the deep LSTM model. We call the model `Single`. All the three models are tested using $b_2^i \in$ B2. The training data and test data setup is shown in Table 2. The symmetric mean absolute percentage error (SMAPE) is used for performance evaluation.

	`Base`	`AnB`	`Single`
Training data	A1	A1 + b_1^i	b_1^i
# *frozen* layers	-	n	-
Test data	b_2^i	b_2^i	b_2^i

Table 2: Training and test data for i-th customer for different deep LSTM models

4 RESULTS AND DISCUSSION

In this section, we systematically analyze the feature transferability and their stability among diverse time series. We also show that by using transfer learning, we can use deep LSTM model for accurate time-series forecasting with limited history at a very small computational cost.

4.1 Time-Series Feature Transferability

First, we compare the performance of the `AnB` model and the `Single` model on all customers from subgroup B, to show the improvements using transfer learning. We also use the classical forecasting method HoltWinters as a baseline. In this experiment, we fix the number of freeze layer at 3. The result is shown in 3a. When the training size is very small, transfer learning provides substantial performance improvement of the `A3B` model over the `Single` model. In particular, the SMAPE drops from around 200% to less than 75%. While increasing the training size helps both two model get smaller SMAPE, the performance difference between the two models also shrinks. When the training size is large enough, the performance of the two models converges, with a small learning gap. We refer to the final gap between the two networks when full data is available as the "transfer learning gap" or *TLG*. In our future work we will look at *TLG* in terms of data complexity.

Furthermore, by fixing the training size, we can also control the number of frozen layers, n, in the transfer learning model `AnB`, and investigate the performance improvement of different n's. The result is shown in Figure 3b. As we increase the depth of transfered layers, the performance gain diminishes. We also reveal that there is a diminishing rate of return on forecasting performance as a function of the number of trainable layers.

In our experiments we used simple NULL count and *VAR* computation approaches to do threshold classification of time-series into sparse and noisy classes. In Figure 4 different time-series class performance is shown. Three classes of time-series are used as input: (i) seasonal, (ii) noisy and (iii) sparse. A model that uses transfer learning always outperforms an equivalent model with limited history, however, its performance is highest for noisy data. This shows that the nonlinearity presented in the noisy data are modeled better by the transfer learned model.

4.2 Computation resources

A major motivation for this work was to cut the training and inference cost of time-series models in production. The current state of the art is to train a single model per time-series. This is computationally unsustainable as the number of time-series increases. The transfer learning approach presented in this paper is able to alleviate the computational burden while providing competitive results.

With transfer learning, it is possible to train a single model that does inference on N time-series where N can be in the thousands (or hundreds of thousands). This results in many orders of magnitude reduction if resources needed for training, inference and storage with a small performance hit (see Figure 5a).

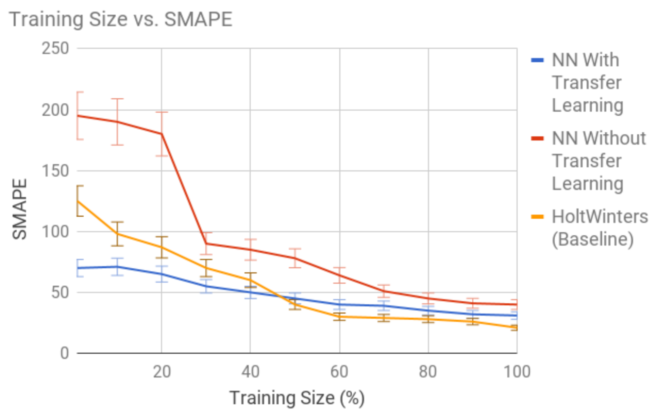
4.3 Democratizing Forecasting

Time-series forecasting has been primarily accessible to experts [5]. With a growing importance of the time-series forecasting field, however, ability for non-experts to generate reasonable forecasts becomes increasingly important. Previous work [12] aims to provide automated time-series model selection, however, these techniques do not scale for millions of time-series due to per-timeseries model retraining.

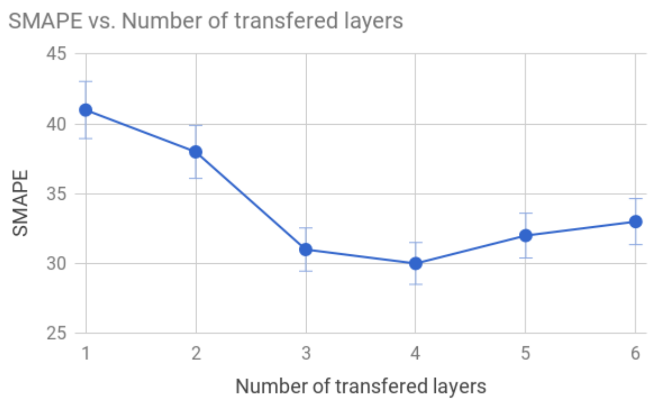
Our goal is to use transfer learning to democratize the time-series forecasting field making non-experts get decent results competitive with the experts in the field. Figure 5b provides an overview of our online forecasting tool that accepts a time-series data and using the novel time-series transfer learning strategy presented, provides a forecast. This tool will allow ImageNet-like data collection [14] for an important and niche market that time-series forecasting is becoming.

4.4 Abstract Feature Extraction from LSTM for Time Series Classification

The proposed method can successfully extract generalized features from time series (via the bottleneck layer discussion in Section 2.1). Such features are used for unlabelled time series classification and we show substantial improvements on classification accuracy over only using traditional features/statistics of time series. We use the standard UCR time-series classification dataset [4]. The accuracy of the classifier is shown in Figure 6. The y-axis is the accuracy of the classifier. Each bar shows the accuracy for a certain category. The classifier is trained using either only standard time-series features (e.g., variance, seasonality, trend), or the abstract features extracted by the proposed method. Besides the difference of features, all other settings (machine learning model, training/testing size, etc.) are identical for two classifiers. The top of the blue bar is the accuracy



(a)



(b)

Figure 3: (a) Performance comparison of deep LSTM models between training with single time series and training with transfer learning. The x-axis is the training size; the y-axis is SMAPE. The red curve represents single time series-trained model; the blue curve represents model using transfer learning. The performance gap is huge for short training sizes. When training size increases, the performance difference shrinks. Each round dot represents the mean SMAPE of all customers from B2, and the error bar illustrates the standard deviation of SMAPE over 58,000 customers. (b) Transfer learning performance for models with different transferred layers. The x-axis is the number of transferred layers from the base model, the y-axis is SMAPE. The meaning of round dots and error bars is the same as the meaning in (a).

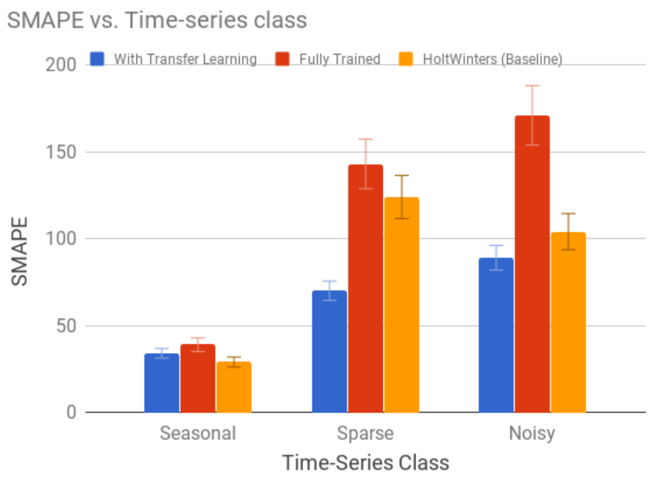


Figure 4: Transfer learning performance for different types of time series. The y-axis is SMAPE. Transfer learning brings substantial improvements for sparse and noisy time series. The meaning of the bars and error bars is the same as the meaning of round dots and error bars in (a).

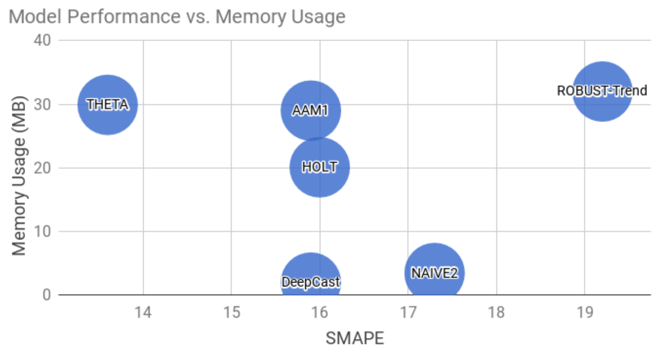
for classifier with standard features, and the top of the red bar is the accuracy for classifier with auto-encoder extracted features. Hence, the length of the red bar demonstrates a great boost in performance relative to the baseline method of using standard time-series features.

4.5 Transfer Learning for Time Series Disaggregation

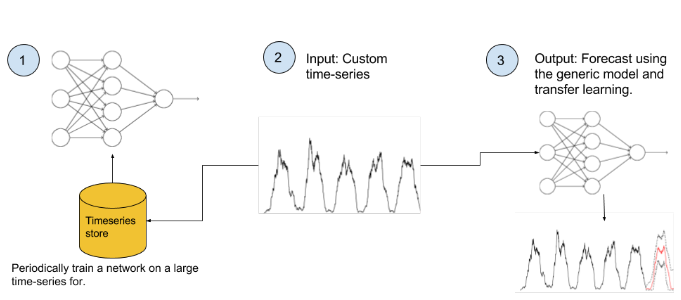
We show transferability of the learned time-series features for disaggregation. In particular, we aim to estimate individual appliance usage given an aggregate power consumption. We compare the disaggregation performance relative to the standard LSTM model, by using Pecan Street dataset [21], which contains hourly measurement of power consumption of homes and individual appliances from 345 homes with each having complete record for at least one appliance, mainly located in Austin, Texas, in 2016. We use the pre-trained model on a large, generalized dataset, then fine-tune the last prediction layer for each individual time series on the disaggregation task. The individual time series is different from any time series in the generalized dataset. The disaggregation performance is shown in Figure 7. The layout of Figure 7 is similar with the layout of Figure 6. It is shown that using transfer learning, the accuracy for disaggregation of different appliance types is substantially increased.

4.6 Transfer Learning for Time Series Forecasting

We also demonstrate the transferability of time series forecasting models. To train the forecasting model with transfer learning, we also first use the pre-trained LSTM based forecasting model on a large dataset. Then, for any individual time series (not from the large dataset), we fine tune the fully connected layers of the LSTM model based on the individual data. We use a large-scale electricity dataset described in Section 3. We also test the transferability of time-series forecasting on standard M3 dataset. The result is shown in Figure 8a and Figure 8b. The y-axis is the symmetric mean absolute



(a)


1
2
Input: Custom time-series
3
Output: Forecast using the generic model and transfer learning.
Timeseries store
Periodically train a network on a large time-series for.

(b)

Figure 5: (a) The presented Transfer Learned Model excels in compute power savings relative to the performance hit. (b) Our online forecasting tool will use the transfer learning technique for time-series to democrotize the time-series forecasting process provide high quality forecasts for everyone with dramatically low computational cost.

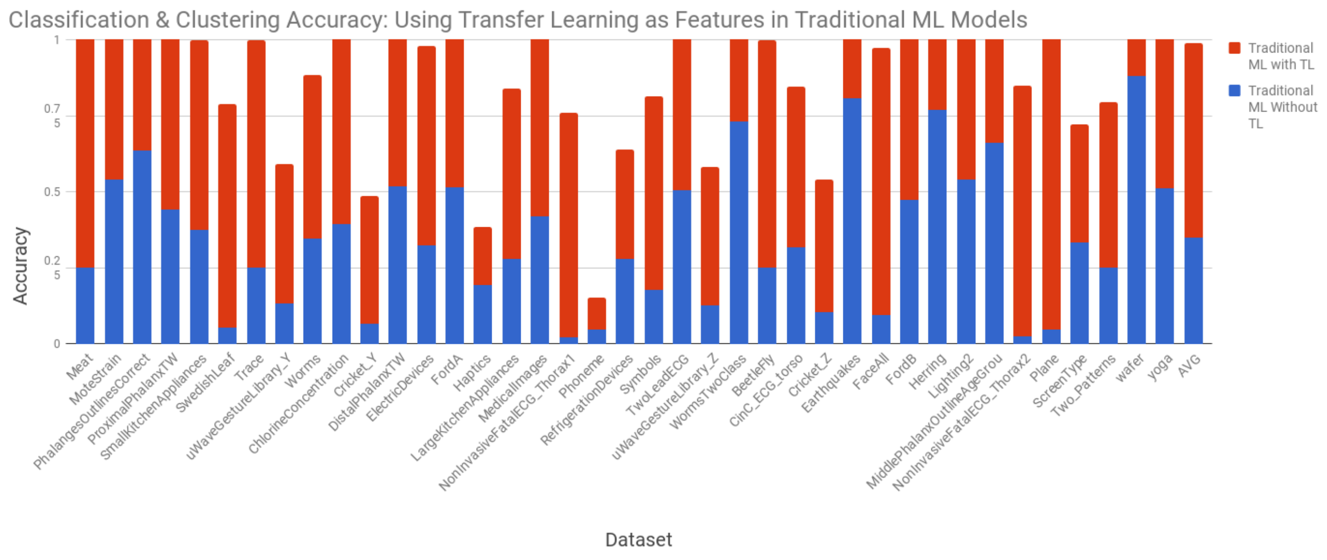


Figure 6: Performance comparison of classification/clustering task on the UCR dataset by directly using the learned features in traditional machine learning models.

percentage error(SMAPE). We compare the performances for both in-sample training data, and out-of-sample test data. The constant improvements of using transfer learning (red bars) over not using transfer learning (blue bars) show competitive performance even on short time-series such as those in the M3 dataset. Note that we trained a single model using the PG&E dataset instead of M3 leveraging transfer learning while other approaches trained a single specialized model per time-series in M3 (3K total). Transfer learning shows competitive results while using only a fraction of the training and inference cost (see Section 4.2).

5 CONCLUSION

It is well known that transfer learning exists for images. In this paper, we demonstrated transfer learning for time series. We have introduced a new loss function that aims to provide both regression loss, which is important for our forecasting objective, and a reconstruction loss, which is important for generalization and transferability of the network.

We have shown that our approach outperforms the baseline deep learning methods used for forecasting. More specifically, we have shown a dramatic forecasting accuracy improvement with transfer learning under small to medium training data size conditions. Furthermore, we have identified compute cost improvements when

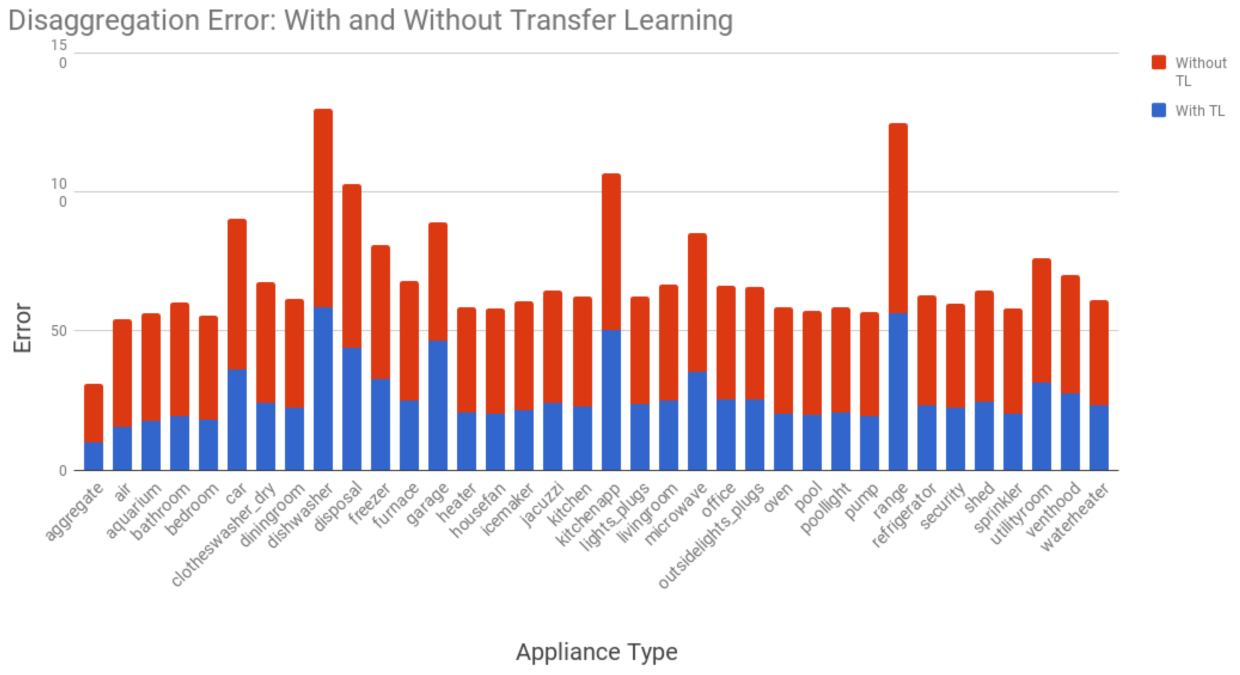
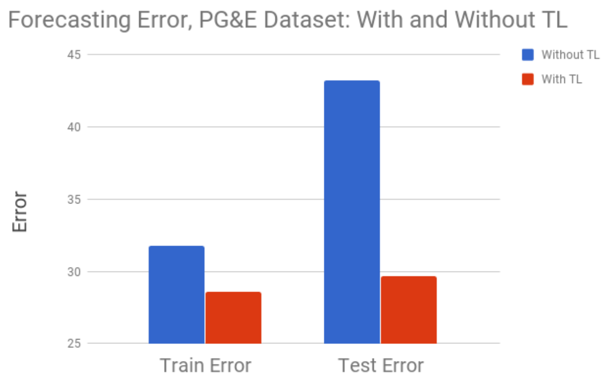
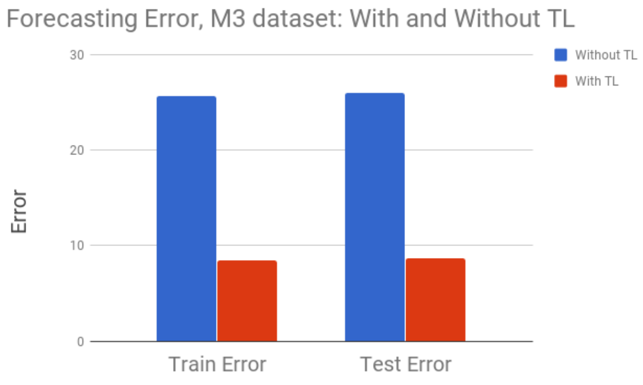


Figure 7: Transfer learning applied to the disaggregation tasks.



(a)



(b)

Figure 8: (a),(b) Forecasting task evaluation by leveraging transfer learning on new datasets.

using our time-series transfer learning approach. Finally, we have also shown the transferability of the learned time-series features to classification and disaggregation tasks.

For our future work, we will compare transfer learning across different architectures in terms of stability and applicability for unseen target classes focusing more on theoretical guarantees of transfer learning. We are also currently focusing on providing this work as a service for practitioners to use as an online or open-source tool for time-series feature generation or as an offline pre-trained model to be used as a prior for time-series machine learning tasks.

REFERENCES

[1] Mohammad Assaad, Romuald Boné, and Hubert Cardot. 2008. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion* 9, 1 (2008), 41–55.
[2] Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. *ICML Workshop on Unsupervised and Transfer Learning* (2012), 17–36.

[3] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. 2017. An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting. *arXiv preprint arXiv:1705.04378* (2017).

[4] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/.

[5] Lars Dannecker, Robert Lorenz, Philipp Rösch, Wolfgang Lehner, and Gregor Hackenbroich. 2013. Efficient forecasting for hierarchical time series. *ACM International Conference on Information & Knowledge Management* (2013), 2399–2404.

[6] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Conference on Computer Vision and Pattern Recognition* (2015), 2625–2634.

[7] Andrew Giel and Ryan Diaz. 2015. Recurrent Neural Networks and Transfer Learning for Action Recognition. (2015).

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), 6645–6649.

[9] Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. *Advances in Neural Information Processing Systems* (2013), 190–198.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computing* 9, 1 (1997), 41–55.

[11] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. *IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), 7304–7308.

[12] Rob J Hyndman, Yeasmin Khandakar, et al. 2007. Automatic time series for forecasting: the forecast package for R. 6/07 (2007).

[13] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. *IEEE conference on Computer Vision and Pattern Recognition* (2014), 1725–1732.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* (2012), 1097–1105.

[15] Jungsuk Kwac, June Flora, and Ram Rajagopal. 2014. Household energy consumption segmentation using hourly data. *IEEE Transactions on Smart Grid* 5, 1 (2014), 420–430.

[16] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. 2017. Time-series extreme event forecasting with neural networks at Uber. *International Conference on Machine Learning* (2017).

[17] Quoc V Le, Alexandre Karpenko, Jiquan Ngiam, and Andrew Y Ng. 2011. ICA with reconstruction cost for efficient overcomplete feature learning. *Advances in Neural Information Processing Systems* (2011), 1017–1025.

[18] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications. *arXiv preprint arXiv:1603.06111* (2016).

[19] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition* (2014), 1717–1724.

[20] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[21] Pecan Street Inc. 2017. Dataport from Pecan Street. https://dataport.cloud/

[22] Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. 2016. Variational recurrent adversarial deep domain adaptation. (2016).

[23] Prajit Ramachandran, Peter J Liu, and Quoc V Le. 2016. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683* (2016).

[24] Sungho Shin, Kyuyeon Hwang, and Wonyong Sung. 2016. Generative Knowledge Transfer for Neural Language Models. *ArXiv e-prints* (2016).

[25] Taylor SJ and Letham B. 2017. Forecasting at scale. *PeerJ Preprints 5:e3190v2* (2017).

[26] Desheng Dash Wu and David L Olson. 2015. Financial Risk Forecast Using Machine Learning and Sentiment Analysis. , 32–48 pages.

[27] Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345* (2017).

[28] Seunghyun Yoon, Hyeongu Yun, Yuna Kim, Gyu-tae Park, and Kyomin Jung. 2017. Efficient transfer learning schemes for personalized language modeling using recurrent neural network. *arXiv preprint arXiv:1701.03578* (2017).

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems* (2014), 3320–3328.