Few-shot Time-Series Forecasting with Application for Vehicular Traffic Flow

Abstract—**Few-shot machine learning attempts to predict outputs given only a very small number of training examples. The key idea behind most few-shot learning approaches is to pre-train the model with a large number of instances from a different but related class of data, classes for which a large number of instances are available for training. Few-shot learning has been most successfully demonstrated for classification problems using Siamese deep learning neural networks. Few-shot learning is less extensively applied to time-series forecasting. Few-shot forecasting is the task of predicting future values of a time-series even when only a small set of historic time-series is available. Few-shot forecasting has applications in domains where a long history of data is not available. This work describes deep neural network architectures for few-shot forecasting. All the architectures use a Siamese twin network approach to learn a difference function between pairs of time-series, rather than directly forecasting based on historical data as seen in traditional forecasting models. The networks are built using Long short-term memory units (LSTM). During forecasting, a model is able to forecast time-series types that were never seen in the training data by using the few available instances of the new time-series type as reference inputs. The proposed architectures are evaluated on vehicular traffic data collected in California from the Caltrans Performance Measurement System (PeMS). The models were trained with traffic flow data collected at specific locations and then are evaluated by predicting traffic at different locations at different time horizons (0 to 6 hours). The Mean Absolute Error (MAE) was used as the evaluation metric and also as the loss function for training. The proposed architectures show lower prediction error than a baseline nearest neighbor forecast model. The prediction error increases at longer time horizons.**

Index Terms—**vehicular traffic, one-shot classification, time-series**

I. Introduction

Time-series forecasting is the task of predicting future values of a measurement given a sequence of past measurements. Time-series forecasting methods have been extensively studied and has been applied in several domains, notably in finance, business, and environmental studies [1]. Time-series forecasting has grown in importance in recent years with the Internet-of-Things (IoT) and the large amount of time-series data continuously produced by the embedded sensors.

A common feature of most time-series forecasting methods is that the forecast model is learned from a long record of historical data. Thus forecasting can only be applied in areas where sufficient historical data is available. "Few-shot" learning approaches attempt to predict outputs given only a very small number of training/historical examples. The key idea behind most few-shot learning approaches is to pre-train the model with a large number of instances from *a different but related* class of data.

Few-shot learning has been most successfully demonstrated in classification using deep learning neural networks []. For instance, a one-shot face classification system can recognize a face of a new person given only one image of that person's face. This is enabled by pre-training the system with several images of other persons' faces. The lower layers of the deep neural network then learn the features common to all faces and only a few images from a new class (i.e., new face) are sufficient to distinguish an instance of the new class from other classes.

However, few-shot learning has not been extensively applied to time-series forecasting, with a few notable exceptions [2]. Few-shot forecasting is the task of predicting future values of a time-series even when only a small set of historic time-series is available. Few shot forecasting has applications in domains where a long history of data is not available. For instance, time-series forecasting methods are used in vehicular traffic prediction. Few-shot forecasting methods can enable traffic prediction along new roads, along rural roads where traffic records are sparse, or along pedestrian pathways where sensors designed for vehicles are not applicable. In these scenarios, few-shot forecasting can ensure that neural network models do not need to be re-trained.

In this work, we developed and evaluated three deep neural network architectures for few-shot forecasting. All the architectures use a Siamese twin network approach to learn a difference function between pairs of time-series, rather than directly forecasting based on historical data as seen in traditional forecasting models. The networks are built using Long short-term memory units (LSTM). During forecasting, a model is able to forecast time-series types that were never seen in the training data by using the few available instances of the new time-series type as reference inputs.

We evaluated the proposed architectures by forecasting vehicular traffic using data collected in California. The models were trained with traffic flow data collected at specific locations and then are evaluated by predicting traffic at different locations at varying time horizons (0 to 6 hours). Our evaluation shows that our straightforward Siamese twin architecture does *not* produce accurate forecasts in a few-shot setting. However, our two more complex architectures show lower prediction error than a baseline nearest neighbor forecast model. As expected, the prediction error increases at longer time horizons.

The main contributions of this work are: 1) novel deep learning neural network architectures that implement few-shot time-series forecasting, and 2) evaluation of the proposed architectures on large real-world datasets for vehicular traffic forecasting.

The rest of the paper is organized as follows. Section II lists related work in time-series forecasting. Section III describes the problem of few-shot forecasting and the architectures associated with our approach. Section IV gives detailed information on the vehicular traffic dataset used to evaluate this work. We present our results in Section V and give our conclusions in Section VI.

II. Related Work

Deep learning methods based on Long Short Term Memory (LSTM) have been found to produce accurate forecasts from time-series data (e.g., [3]). However, these methods are not designed for few-shot learning. The concept of one-shot and few-shot learning has been proposed mostly for classification models. Koch et al. [4] propose a one-shot learning model for image classification of written characters. Specifically, their approach trains a Siamese Convolution Neural Network (CNN) to learn a difference function, rather than traditionally training on images and labels. The Siamese CNNs would share the same weights during training, and their purpose was to encode two input images into a large, flattened layer. The intuition is that the flattened layer would capture differentiating features of the two compared character images. Then the two flattened layers is fed into an element-wise difference layer. Finally, the difference layer is fully-connected into a single neuron, activated by the Sigmoid function. A value of 0 would indicate that the two images differ. A value of 1 would indicate that the two images are the same. Using an n-way one-shot comparison, their Siamese model was able to accurately classify new sets of characters that were not included in the training set [4]. In our approach, instead of having a single output node that represents a difference score, our proposed architectures output a difference vector instead.

As in this work, LSTMs are used in the few-shot time-series model proposed by Iwata and Kumagai [2]. However, they do not use a Siamese network approach but add an attenuation mechanism to a recurrent neural network. They reason that time series data used as the training set may carry similar features for forecasting a completely different test set. They introduce an attention mechanism that captures patterns based on support windows fed into the model called a support set along with the test window. One of our proposed models also includes an attenuation mechanism that aims to generalize the patterns of all traffic data in the training set, with the notion that the test set will follow these patterns too.

Transfer learning is another approach to forecasting given insufficient historical data [5]. Transfer learning involves taking a model trained with an abundance of certain data and further retraining it with similar data. The main difference from our approach is that our architectures do not require re-training with a large collection of instances from the new time-series type. The time savings from not requiring retraining is useful in vehicular traffic forecasting application as traffic patterns often change as new roads are built, destroyed, or are under reconstruction.

Finally, the concept of meta-learning through neural networks has been proposed for time-series forecasting [6]. The goal of meta learning is to train on a diverse dataset, understand overarching knowledge shared within the dataset, and apply the knowledge to a different task without any auxiliary references (zero-shot) [7]. Our model is not to be zero-shot as it requires a reference instances to achieve a higher forecasting accuracy.

III. Problem Statement and Approach

The problem of Few-shot time-series forecasting can be stated as follows:

Let $l(\mathbf{w})$ denote the length of a time-series $\mathbf{w}$. Given a set of time-series, W, and and a much shorter time-series, $\mathbf{x}$, where $l(\mathbf{x}) << l(\mathbf{w}), \mathbf{w} \in W$, forecast the future values of $\mathbf{x}$ at time $l(x) + h$, $x_{l(x)+h}$, where h is called the forecast horizon.

A. Few-Shot Forecasting Models

We describe three Few-Shot Forecasting Models (FSFMs). All three models use Long short-term memory units (LSTM) [8]. The main advantage of LSTM as compared to a standard cell in a Recurrent Neural Network is the addition of an update gate and a forget gate to control the flow of temporal information whilst keeping a hidden state within the network [9]. Each model is structured as a Siamese neural network – given pairs of historical time-series data, it attempts to predict the *difference between their forecasts*. Thus, the FSFM models learns through a difference function, rather than directly forecasting based on historical data seen in traditional forecasting models.

1) **Siamese FSFM:** The Siamese FSFM is created using two identical LSTM neural networks, meaning that the architecture and weights of both networks are identical throughout training (Fig. 1). Each twin is comprised of 128 LSTM units with return sequences set to true. This allows for 128 features to be observed for every single time-step of the historical input data. This is fed to a Dropout layer with a weight of 0.2, which helps prevent over-fitting during training. A Flatten layer is applied to unravel the data into one dimension of nodes. This layer is fully-connected to a dense layer with 24 nodes—the number of time steps in the forecast horizon. That is the output of the Siamese LSTM module. The output is a pseudo-forecast of the historical data input. Notice that this pseudo-forecast does not have to match the true forecast values. The pseudo-forecast can be any sequence of values so long

as the *difference* between the two Siamese pseudo-forecasts is accurate as the model is learning a difference function, rather than directly forecasting. Finally, the two pseudo-forecasts are run through an element-wise difference layer to output the difference vector.

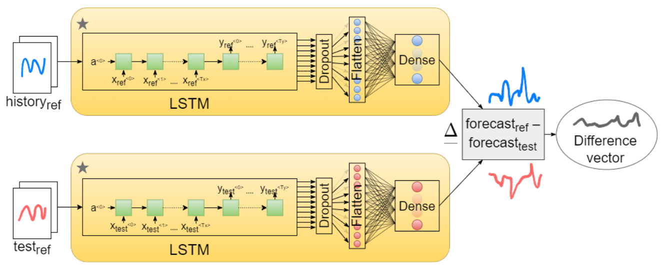

Fig. 1. The pairs $(\vec{h}_{ref}, \vec{h}_{test})$ are passed into their own Siamese sibling. Each Siamese sibling produces a pseudo-forecast. The difference is taken between those two vectors and outputs the difference vector $\vec{d}$. During testing, this difference can be used to calculate the forecast of a test window: $\vec{f}_{test,test} = \vec{f}_{test,ref} - \vec{d}_{test}$.

2) **Difference FSFM:** The Difference FSFM is created using just a LSTM model identical to one of the twins in the Siamese FSFM (Fig. 2). First, the difference of the reference historical data and the test historical data is immediately taken and then fed into the model. The output is the difference vector. An advantage of this model is the ability to generalize to any time-series data. However, by taking the difference of the historical data initially, the model loses knowledge of the general shape of the input window. Therefore, this model cannot capture specific features of the historical data to make more accurate forecasts.

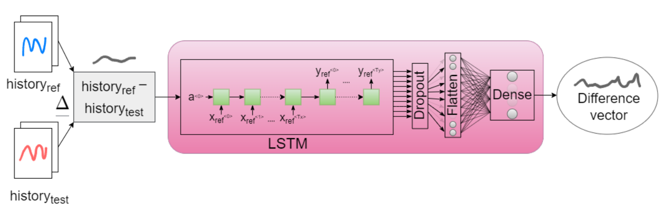

Fig. 2. The difference between $\vec{h}_{ref}$ and $\vec{h}_{test}$ is initially taken and then fed to the LSTM network to output the difference vector $\vec{d}$.

3) **Combined FSFM:** The Combined FSFM is created using the Siamese-twins model as an attenuation mechanism for the Difference FSFM (Fig. 3). In addition, the Siamese network in this architecture uses a CNN instead of an LSTM. The CNN captures particular features from shorter segments of the input data. Specifically, the model uses two Conv1D layers. The first layer has 32 filters and a kernel size of 3. The second layer has 64 filters and a kernel size of 3. Both layers use the Rectified Linear Activation Unit (ReLU) as its activation function. ReLU prevents vanishing gradients when training on deep networks with uninitialized weights such as in our architecture [10]. Then the difference between the two CNNs are taken before being concatenated with the difference of the historical data. This is fed into the LSTM module along with the difference input.

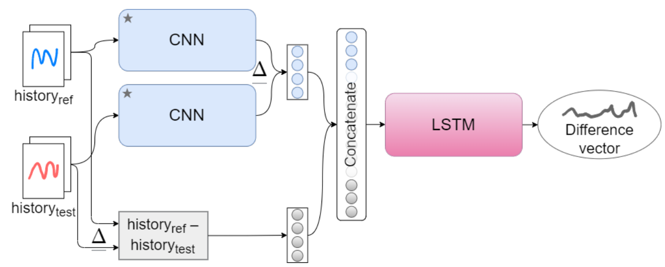

Fig. 3. The Siamese CNN in blue is used as an attenuation mechanism to the Difference Model. The model concatenates the cells with ($\vec{h}_{ref}$ - $\vec{h}_{test}$) before being fed to the LSTM network.

B. Pair-wise Training

The inputs to the models are pairs of historical time-series, $(\vec{h}_{ref}, \vec{h}_{test})$ called the *reference window* and the *test window* respectively. The reference window acts as a baseline to compare its traffic flow with the test window. The fluctuating differences between the historical data will reveal the fluctuating differences between their forecasts. By knowing the difference between the forecasts, $\vec{d}$, one would only need to additionally know the forecast of the reference window, $\vec{f}_{ref}$, to estimate the forecast of the test window, $\vec{f}_{test}$. Therefore, given just the historical data of a test window that is similar to the reference window, the model can predict a difference vector of their forecasts rather than an actual forecast. Then, the difference can be subtracted from the known forecast of the reference window to achieve the predicted forecast of the test window: $\vec{f}_{test} = \vec{f}_{ref} - \vec{d}$. Learning how to predict differences allows the model to (1) generalize to other windows that were never seen in the training data before and (2) require only a few instances of a new class of time-series, used as reference windows, to make predictions for a given test window.

The reference and test window pairs are restricted to the same station and the same days of the week. For example, given an arbitrary station called Station A, all its windows spanning 24 hours of a Sunday and 6 hours of a Monday are paired. After that, a 6 hours stride is taken for the next set of window pairs. Hence, all windows of Station A spanning 18 hours of Sunday and 12 hours of Monday are paired. The process is continued until all possible 30-hour time spans for the windows are paired. The size of the dataset is $O\left(n^2 \cdot m\right)$, for n is the number of windows created from the station through the sliding windows method, and m is the number of stations in the dataset.

The magnitude of the data values between instances of different classes can vary widely, yielding a high standard deviation. For example, vehicular traffic at some locations is much heavier than others. Normalizing features with high standard deviations significantly improves model accuracy [11]. Every window pair is therefore normalized by Min-Max Scaling based on the test window using the formula:

$$\frac{h_i - h_{min}}{h_{max} - h_{min}}$$

where h_i is the value at timestep i of $\vec{h}_{ref}$ and $\vec{h}_{test}$, h_{max} and h_{min} are the maximum and minimum value of the test window's historical data, $\vec{h}_{test}$, respectively. Inverting the Min-Max scale after prediction is necessary as the window pairs are the model's input data. The model's output data is the difference between the forecast data of the respective window pairs given by: $\vec{d} = \vec{f}_{ref} - \vec{f}_{test}$. The models are trained with pair-wise inputs: $(\vec{h}_{ref}, \vec{h}_{test})$ and target output: $\vec{d}$.

All the FSFM models are trained under the same parameters. The batch size is 512, which is chosen for the main purpose of reducing training time. Although, large batch sizes with respect to the small size of a training instance leads to poor generalization. A smaller batch size closer to 32 introduces noise to the gradients, which improves finding flat minimizers in the loss function [12]. The Adam Optimizer is chosen with an initial learning rate of $0.0001 \cdot \sqrt{batchsize}$. The initial learning rate of 0.0001 is empirically chosen as it produced the quickest descent in loss within 100 epochs. The learning rate is further scaled by $\sqrt{batchsize}$ to further reduce training time [13]. The loss function for training is the Mean Absolute Error (MAE). The MAE is chosen instead of the commonly used Mean Squared Error (MSE)to reduce the impact of outliers in the real-world data.

IV. Dataset

California Department of Transportation's (Caltrans) Performance Measurement Systems (PeMS) collects real-time traffic data using approximately 40,000 loop detectors hidden throughout the freeway system pavements [14]. The data is then sent to a central database over the Caltrans Wide Area Network to be archived. We use the archived data in this work to evaluate the time-series forecasting methods. The data represents the mobility (the average point-to-point travel time) and reliability (the day-to-day variability between the expected average travel time and the actual travel time) of traffic at different time-resolutions and therefore can be considered a large repository of time-series. PeMS data can be analyzed to monitor traffic congestion at individual freeway segments at varying time intervals such as a certain time of day, day of the week, season, and year.

A. Data pre-processing

We next describe the steps to prepare the 5-minute resolution traffic data retrieved from PeMS's Data Clearinghouse for training and evaluation. The dataset is composed of daily features for a station, sampled every 5-minute period. PeMS provides data that measures traffic speed, the number of cars in a segment, vehicle-miles traveled, vehicle-hours traveled, and hours of delay. In this work, the training dataset consists of only one feature — *Total flow*. Total flow is defined as the sum of vehicles over a 5-minute period across all lanes of a station. Data from the entirety of 2019 from the over 2000 stations in District 12 (Orange County in southern California) is used for training. Stations with data quality issues are removed from the training dataset. For instance, stations that record zero traffic flow for many weeks and months at a time. This happens when a station is either located in a rural area, is under construction, or is even removed entirely.

We also adjust the training dataset to account for daylight savings time. The PeMS dataset includes one hour of repeated data during the spring forward hour and excludes one hour of data during the fall back hour of daylight savings time. To handle the case of repeated data, the extra hour of data is simply removed. To handle the case of missing data, a simple imputation technique estimates the missing data by averaging the sampled total flow from the hour before and the hour after. Other anomalies such as unusually high traffic flow (magnitudes higher than the previous and future timesteps) due to invalid sensor data have already been vetted by PeMS. At the end of these pre-processing steps, the total number of stations in the dataset is 1793.

B. Creating the Training Instances

Each station represents a "class" in this application. An instances of the class is a window representing its traffic flow for 30 hours: 24 hours of historical data and 6 hours of the known forecast. A week's worth of traffic flow data can be split into multiple windows via the sliding window method. The size of the sliding window is 30 hours and the stride is 6 hours. The traffic flow data is re-sampled from every 5 minutes to every 15 minutes. A period of 15 minutes is chosen because a higher resolution would introduce too much variance that does not carry meaningful patterns conducive to generalizing to multiple stations [15]. Also, prior work has shown that a period of 15 minutes leads to higher accuracy in ARIMA models when forecasting 6 hours, as the number of horizon time-steps is reduced to 24 [16]. The historical data of 24 hours (96 time steps) is chosen due to the limited amount of memory available when training the models.

Common historical data lengths with a 6 hour forecast (24 time steps) is usually several days to a week. One problem with such a short historical data length in this approach is the lack of context within a week. A window may not differentiate a weekday that forecasts into another weekday, a weekday that forecasts into a weekend, or a weekend that forecasts into a weekday. Thus, in addition to the total flow, the day of the week is added to the dataset as a feature. The day of the week is represented as an integer between 0 and 6 (denoting Sunday through Saturday respectively).

V. Results and Discussion

The Forecasting Models (FSFM) are implemented using the Keras library. The training environment involves the libraries: Python v3.9.12, TensorFlow-GPU v2.6, Numpy v1.21.5, Pandas v.1.4.1, and Matplotlib v.3.5.1. Training is performed with an Intel® Core™ i7-4790K CPU and a NVIDIA GeForce GTX 1070 GPU.

The models are trained on the PeMS dataset, specifically data from the stations of District 13, which covers Orange County. Data from the stations of District 3, which covers all counties in North Central California, is used to test the accuracy of the FSFMs.

Tracking the accuracy of the model during training is based on the accuracy of forecasting the validation set. The validation set is commonly a subset of the training dataset in traditional neural networks. However, the validation set of FSFMs should be exclusive from the training dataset. In this case, the training dataset is built from the stations of District 13, and the the validation dataset is built from the stations of District 3.

We evaluate model performance using the Mean Absolute Error (MAE) metric:

$$MAE(\mathbf{p}, \mathbf{a}) = \frac{\sum_{i=1}^{n} |a_i - p_i|}{n}$$

where n is the number of stations, a_i is the actual forecast of the total flow of vehicles for $station_i$ at a given timestep, and p_i is the predicted forecast of the total flow of vehicles for $station_i$ at the same given timestep. This equation is be calculated 24 times for each of the 24 timesteps of the forecast horizon. Such error measurements are taken after inverting the normalization from the Min-Max Scaling.

The Nearest Neighbor Model (NNM) was implemented to provide a baseline when comparing prediction accuracy to the Difference and Combined FSFMs. For its prediction, the NMM takes the exact forecast from the reference window whose historical data matches closest to the test window's historical data measured via Euclidean distance.

In our experiments, the Siamese FSFM provided the most accurate predictions on the training set (District 12) based on having the lowest MAE loss. However, this model failed to generalize to the validation set (District 3). This is because the last two layers are Flatten and Dense layers. As these layers are fully connected, the magnitude of their weights overwhelms the weights of the hidden states of the prior LSTM network. The weights created from the Flatten and Dense layer overcompensates towards matching the target difference vectors that it overfits. Therefore, the simple Siamese FSFM architecture fails to generalize and is subsequently not viable for few-shot forecasting. As the Siamese FSFM's forecasts are less accurate than the NNM, we omit them from the results.

Fig. 4 compared the decrease in loss while training the different models. The loss is the MAE of the difference vector, which is normalized during training with the Min-Max Scaling method described in Section III-B. This validation loss is automatically calculated by the Keras library after every epoch during training. The validation data is composed of over 900 stations from District 3 as to not bias towards a few number of stations. The Combined FSFM has a slightly lower validation MAE before increasing due to over-

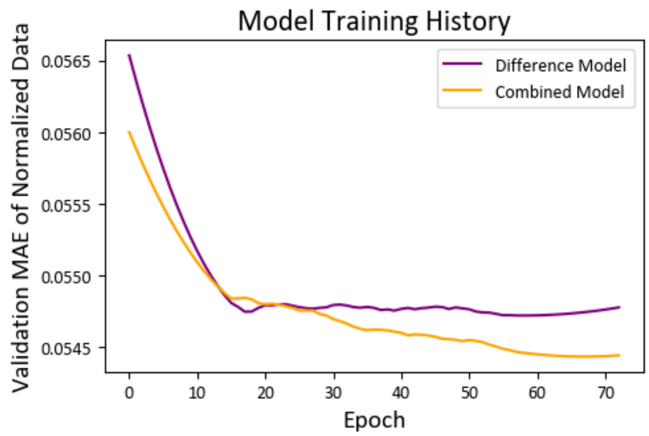

Fig. 4. Comparing the validation loss of two models: the Difference FSFM and the Combined FSFM. The Difference FSFM has a minimum of 0.05478 at epoch 55. The Combined FSFM has a minimum of 0.05445 at an epoch of 63. A Savitzky-Golay filter is used with window length of 31 and polynomial order of 2 to smooth out the two plots.

fitting. Such a slight improvement infers that the Combined FSFM is not expected to outperform the Difference FSFM for every arbitrary station data. Rather, the Combined FSFM will provide more averaged accurate results amongst a large number of windows from 900 stations. In fact, in many windows, the Difference FSFM would have often similar or even more accurate forecasts than the Combined FSFM.

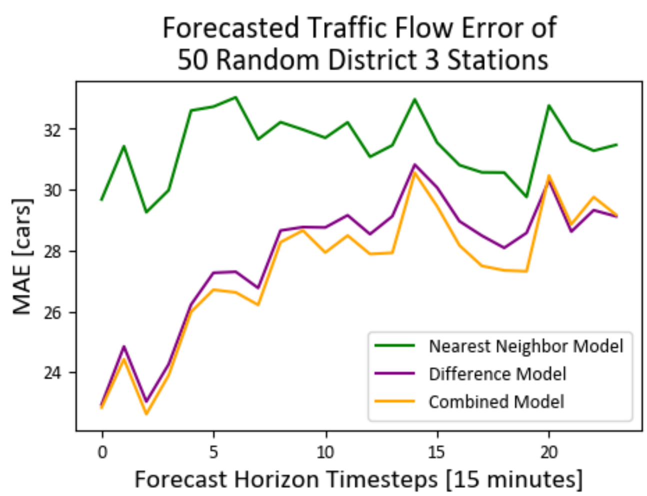

Fig. 5. Both the Difference FSFM and Combined FSFM have lower MAE than the NNM. The Combined FSFM has a slightly lower MAE than the Difference FSFM.

We next compare the prediction error of the different approaches at different forecast horizons. Fig. 5 shows these errors. A one week window is used as a reference to forecast 20 different future windows for each of the 50 random stations chosen in District 3. However, stations can vary in traffic flow. Stations with too few traffic flow would yield a lower MAE, biasing the average MAE lower. Stations with too much traffic flow would yield a height MAE, biasing the average MAE higher. Therefore, the 50 stations are limited to having a similar magnitude of maximum traffic flow —- between 100 and 1000 vehicles. As expected, the MAE of the Difference and Combined FSFM converges with the NNM for bigger forecast horizons. However, both the Difference and Combined FSFMs have noticeably lower MAEs for the first 10 timesteps.

We next show the actual forecasts made by the baseline NNM model and the Difference FSFM and Combined FSFM for a 30-hour window using the data from a randomly selected station (Figure 6). The Difference FSFM generalizes reasonably well; however, having the model miss the overall pattern of the historical data limits its forecasting accuracy. The Combined FSFM offers the extra attenuation with the Siamese CNN, which provides slightly greater accuracy. These results indicate that calibrating the amount of contribution between the attenuation mechanism and the historical data difference before concatenation could offer the higher forecast accuracy.

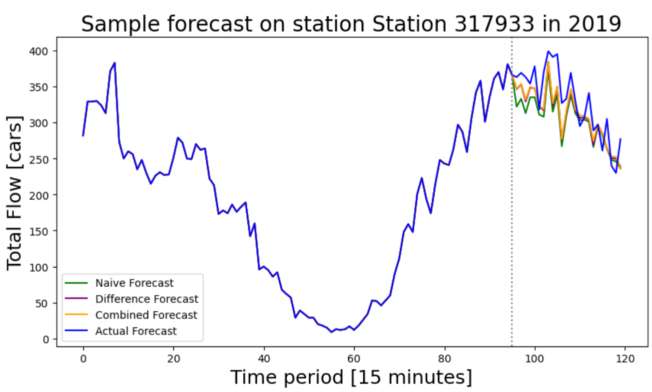

Fig. 6. A single 30-hour window taken at a random time in 2019 for Station 317933, which happens to measure a High-Occupancy vehicle lane. The left of the dotted vertical line shows the historical data of the test window. The right of the dotted vertical line shows the comparison between the Naive NNM, Difference FSFM, and Combined FSFM.

VI. Conclusions and Future Work

We presented a method of extending few-shot classification to time-series forecasting. We introduced three models that learn to predict a difference vector, rather than a direct forecast. The Siamese FSFM failed to generalize to traffic data outside of the training set. The model that had the highest forecasting accuracy based on having the lowest MAE is the Combined FSFM. The attenuation mechanism in the Combined FSFM provided only slightly greater accuracy. These results indicate that calibrating the amount of contribution between the attenuation mechanism and the historical data difference before concatenation could offer greater forecast accuracy.

The proposed architectures have several hyperparameters. Hyperparameters that could be fine tuned are batch size, learning rate, the size of the training data, the length of the historical time-series, and the length of the forecast horizon. The layers of the architecture can also be modified such as the amount of LSTM cells, CNN cells, Dropout rate, and the Subtraction layer. Other types of architectures that use LSTMs such as the Encoder-Decoder model could also be explored, especially for sequence-to-sequence prediction [17].

The current approach was evaluated only on univariate time-series data. Adding features, or a spatial component for temporal forecasting, are avenues for future work. We intend to augment this method to create an FSFM that can be used for other types of time-series data such as stream-flow data from rivers.

Acknowledgment

Removed for double-blind review

References

[1] J. G. De Gooijer and R. J. Hyndman, “25 years of time series forecasting,” *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.
[2] T. Iwata and A. Kumagai, “Few-shot learning for time-series forecasting,” *arXiv preprint arXiv:2009.14379*, 2020.
[3] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, “Time-series extreme event forecasting with neural networks at Uber,” in *International conference on machine learning*, vol. 34. sn, 2017, pp. 1–5.
[4] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, vol. 2. Lille, 2015, p. 0.
[5] A. Hooshmand and R. Sharma, “Energy predictive models with limited data using transfer learning,” in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, 2019, pp. 12–16.
[6] C. Lemke and B. Gabrys, “Meta-learning for time series forecasting and forecast combination,” *Neurocomputing*, vol. 73, no. 10-12, pp. 2006–2016, 2010.
[7] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, “Meta-learning framework with applications to zero-shot time-series forecasting,” *CoRR*, vol. abs/2002.02887, 2020. [Online]. Available: https://arxiv.org/abs/2002.02887
[8] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[9] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 338–342, 01 2014.
[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.
[11] J.-M. Jo, “Effectiveness of normalization pre-processing of big data to machine learning performance,” *The Journal of the Korea Institute of Electronic Communication Sciences*, vol. 14, pp. 547–552, 06 2019.
[12] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *CoRR*, vol. abs/1609.04836, 2016. [Online]. Available: http://arxiv.org/abs/1609.04836
[13] D. Granziol, S. Zohren, and S. Roberts, “Learning rates as a function of batch size: A random matrix theory approach to neural network training,” *Journal of Machine Learning*, p. 30, 2020.
[14] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, “Freeway performance measurement system: mining loop detector data,” *Transportation Research Record*, vol. 1748, no. 1, pp. 96–102, 2001.
[15] D. Levinson, “Spatiotemporal short-term traffic forecasting using the network weight matrix and systematic detrending,” *Transportation Research Part C Emerging Technologies*, pp. 38–52, 2019.
[16] R. Martoglia and G. Savoia, “Towards multi-model big data road traffic forecast at different time aggregations and forecast horizons,” in *8th EAI International Conference on Mobility, IoT and Smart Cities*. EAI, 2022.
[17] Z. Wang, X. Su, and Z. Ding, “Long-term traffic prediction based on lstm encoder-decoder architecture,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6561–6571, 2020.