Multi-Task Time Series Forecasting With Shared Attention

Zekai Chen*, Jiaze E*, Xiao Zhang†, Hao Sheng‡ and Xiuzheng Cheng*
*Department of Computer Science
George Washington University, Washington, DC, USA
{zech_chan, ejiaze, cheng}@gwu.edu
†School of Computer Science and Technology, Shandong University, China
xiaozhang@sdu.edu.cn
‡ School of Computing Science and Engineering, Beihang University, China
shenghao@buua.edu.cn

***Abstract*—Time series forecasting is a key component in many industrial and business decision processes and recurrent neural network (RNN) based models have achieved impressive progress on various time series forecasting tasks. However, most of the existing methods focus on single-task forecasting problems by learning separately based on limited supervised objectives, which often suffer from insufficient training instances. As the Transformer architecture and other attention-based models have demonstrated its great capability of capturing long term dependency, we propose two self-attention based sharing schemes for multi-task time series forecasting which can train jointly across multiple tasks. We augment a sequence of paralleled Transformer encoders with an external public multi-head attention function, which is updated by all data of all tasks. Experiments on a number of real-world multi-task time series forecasting tasks show that our proposed architectures can not only outperform the state-of-the-art single-task forecasting baselines but also outperform the RNN-based multi-task forecasting method.**

***Index Terms*—Time series forecasting, Multi-task learning, Transformer, Self-attention**

I. Introduction

Multi-task time series forecasting, i.e. the prediction of multiple time series data from different tasks, is a crucial problem within both time series forecasting and multi-task learning. In contrast to single-task learning, multi-task time series forecasts provide users with access to estimates across multiple related time series paths, allowing them to optimize their actions in multiple related domains simultaneously in the future. The development of multi-task time series forecasting can benefit many applications such as stock prices forecasting, weather forecasting, business planning, traffic prediction, resources allocation, optimization in IoT and many others. Especially in recent years, with the rapid development of the Internet of Things (IoT), billions of connected mobile devices have generated massive data and further bring many novel applications that can change human life [1], [2]. Analyzing these data appropriately can bring considerable socio-economic benefits such as target-advertising based on accurate prediction of cellular traffic data, real-time health status monitoring, etc. Different from general single-task forecasting problems, practical multi-task forecasting applications commonly have access to a variety of data collection resources as shown in Fig. 1. In this cellular traffic forecasting problem, all the base stations are well deployed in certain urban areas. Station A and Station B share a similar pattern possibly due to geographical proximity while different from the traffic pattern of Station C a lot. If we want to forecast the future cellular traffic of any of them, one main challenge is that how we can fully utilize both commonality and difference among these time series from different stations with the aim of mutual benefit. It is vital especially when there is little acquired data from each station due to failure or privacy reasons. Traditional time series forecasting methods include auto-regressive integrated moving average (ARIMA) [3], [4], vector auto-regression (VAR) [5], support vector regression (SVR) [6], etc. Recently, deep neural networks [7]–[10] offers an alternative. The recurrent neural networks (RNNs) have become one of the most popular models in sequence modeling research. Two variants of RNN in particular, the long short term memory (LSTM) [11] and the gated recurrent unit (GRU) [12], have significantly improved the state-of-the-art performance in time series forecasting and other sequence modeling tasks. Especially, meta multi-task learning [13]–[15] proposed a new sharing scheme of compo-

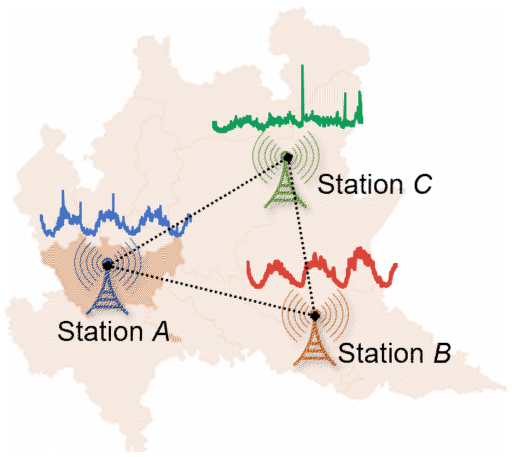


Fig. 1: A paradigm of cellular traffic data collected from multiple base stations.

sition function across multiple tasks based on LSTM models. Most recently, as the ability to capture long term dependency with good parallelism, the Transformer architecture [16], [17] has been widely used in natural language processing (NLP) and yields state-of-the-art results on a number of tasks. Despite the popularity of various sequence modeling research, most of the work focus on either single-task learning or combining multi-task learning with recurrent neural networks and there have been few works in combining MTL with Transformer, especially the self-attention mechanism.

In this paper, we propose to bridge the gap between multi-task learning and Transformer attention-based architectures by designing a shared-private attention sharing scheme MTL-Trans to jointly train on multiple related tasks. Inspired by shared external memory [14] based on LSTM models, we propose two architectures of sharing attention information among different tasks under a multi-task learning framework. All the related tasks are integrated into a single system that is trained jointly. Specifically, we use an external multi-head attention function as a shared attention layer to store long-term self-attention information and knowledge across different related tasks.

We demonstrate the effectiveness of our architectures on a real-world multi-task time series forecasting task. Experimental results show that jointly learning of multiple related tasks can improve the performance of each task relative to learning them independently. Additionally, attention-based sharing architectures can outperform the RNN-based sharing architectures. In summary:

- We are the first to propose an attention-based multi-task learning framework (MTL-Trans) to solve multi-task time series forecasting problems.
- We propose two different attention sharing architectures for sharing self-attention information among different tasks during jointly training process. The external public multi-head attention helps to capture and recording self-attention information across different tasks.
- We conducted extensive experiments on a real-world multi-task time series forecasting task, and the proposed approach obtains significant improvement over state-of-the-art baseline methods.

II. Related Work

Time Series Forecasting. Even though forecasting can be considered as a subset of supervised regression problems, some specific tools are necessary due to the temporal nature of observations. Traditional data-driven approaches such as auto-regressive integrated moving average (ARIMA) [3], [4] model, Kalman filtering [4], support vector regression (SVR) [6], and holt-winters exponential smoothing [18] remain popular. Also, with the rise of various deep learning techniques, many efficient deep models have been proposed for time series forecasting. The recurrent neural networks (RNNs) [7], [8], [11], [19] are powerful tools to model the temporal sequence data. Specifically, based on the variational auto-encoder (VAE) framework [20], [21], several variants of the RNNs have been proposed to process a highly structured natural sequence by capturing long-term dependencies. DCRNN [22] proposed a deep learning framework for traffic forecasting that incorporates both spatial and temporal dependency in the time serial traffic flow. DSSM [23] presented a probabilistic way that combined state-space models with a recurrent neural network. DeepAR [24] estimated a time series' future probability distribution given its past by training an auto-regressive recurrent neural network model.

Transformer framework. Even though the problems of gradient vanishing or explosion have been overcome by LSTMs to some extent, the RNN based models are still not able to modeling very long term dependency [11]. Self-attention, also known as intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading [25], abstractive summarization, or image description generation. With the help of the attention mechanism [16], [26], [27], the dependencies between source and target sequences are not restricted by the in-between distance anymore. Among all the attention based variants, the Transformer model [16] emerges as one of the most effective paradigms for dealing with long-term sequence modeling. It presented a lot of improvements to the soft attention [28] and make it possible to do sequence to sequence modeling without recurrent network units. The proposed "transformer" model is entirely built on the self-attention mechanisms without using sequence-aligned recurrent architecture. Recently, temporal fusion transformer [29] combines high-performance multi-horizon forecasting with interpretable insights into temporal dynamics, which further demonstrated the advantages of attention mechanism in time sequence forecasting. However, most existing research approaches focus on the single-task learning problem. When faced with multiple time series sequences collected from many other related domains, the existing models have to train each task separately without a strong multi-task generalization capability.

Multi-task Learning. Multi-task learning (MTL) is an important machine learning paradigm that aims at improving the generalization performance of a task using other related tasks [5], [7], [20], [30]. Particularly, CellScope [31] applied multi-task learning to resolve the trade-off between data collection latency and analysis accuracy in real-time mobile data analytic, in which data from geographically nearby base stations were grouped together. Luong et al. [10] examined three multi-task strategies for sequence to sequence models: the *one-to-many* setting, the *many-to-one* setting and the *many-to-many* setting. Liu et al. [13]–[15] proposed several multi-task sequence learning architectures by using enhanced and external memory to share information among paralleled RNN models. Despite the wide interest of various sequence modeling research, there is hardly any previous work done on combining multi-task time series forecasting with attention based architectures based on my knowledge.

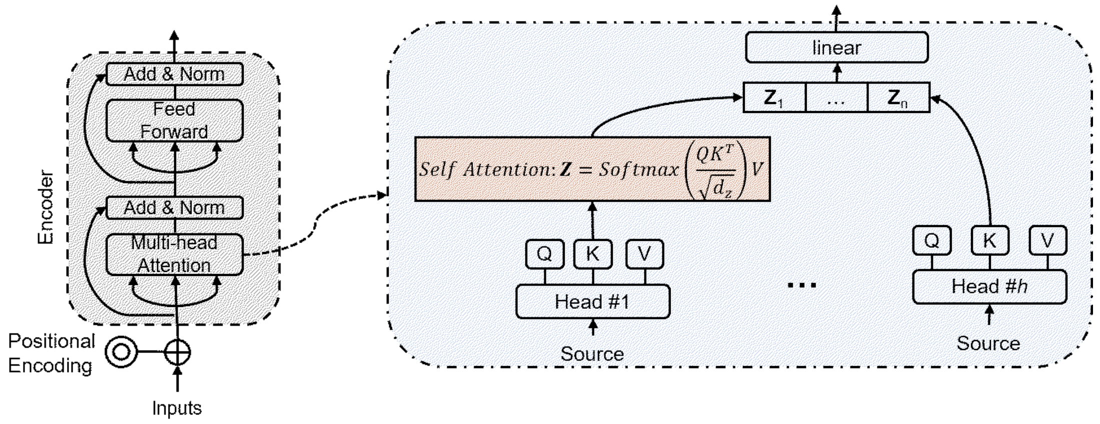


Fig. 2: Multi-head attention architecture in Transformer.

III. Shared-Private Attention Sharing Scheme

A. Task Definition

In this work, we focus on single-step forecasting. The basic definition of a multi-task time series single-step forecasting problem is: Given a dataset $\mathcal{D} = \{\{\mathbf{x}_{mn}, \mathbf{y}_{mn}\}|_{n=1}^{N_m}\}|_{m=1}^{M}$ with multiple sequence tasks, M denotes the number of tasks, N_m means the number of instances in m-th task, $\mathbf{x}_{mn}$ is the n-th sample in m-th task. For instance, $\mathbf{x}_{mn} = \{x_{mn}^{t_1}, \ldots, x_{mn}^{t_s}\}$ could be historical observation values with length s, and $\mathbf{y}_{mn} = \{y_{mn}^{t_2}, \cdots, y_{mn}^{t_{s+1}}\}$ means the future time series sequence with the same length s corresponding to $\mathbf{x}_{mn}$. Hence, the goal of the multiple single-step time series forecasting tasks is to learn a function that maps observation sequence $\{\mathbf{x}_{mn}|_{n=1}^{N_m}\}|_{m=1}^{M}$ to future sequence $\{\mathbf{y}_{mn}|_{n=1}^{N_m}\}|_{m=1}^{M}$: $f(\mathbf{x}_{mn}) \rightarrow \mathbf{y}_{mn}$ jointly by utilizing the latent similarities among the tasks based on multi-task learning.

B. Preliminary Exploration

Scaled Dot-Product Attention. The original Transformer used a particular scaled dot-product attention [16]. The input consists of queries and keys of dimension d_k, and values of dimension d_v. The dot product of the query with all keys would be computed and divided each by $\sqrt{d_k}$. A softmax function would be applied to obtain the weights on the values. In practice, the attention function on a set of queries is computed simultaneously by being packed together into a matrix Q. The keys and values are also packed together into matrices K and V, as a result, the matrix of outputs is as following:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

More specifically, this attention mechanism operates on an input sequence, $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ with n elements where $x_i \in \mathcal{R}^{d_x}$, and computes a new sequence $\mathbf{z} = (z_1, z_2, \cdots, z_n)$ of the same length where $z_i \in \mathcal{R}^{d_z}$.

Each output element, z_i, is computed as weighted sum of a linearly transformed input elements:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V\right) \tag{2}$$

Each weight coefficient, α_{ij}, is computed using a softmax funtion:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \tag{3}$$

And e_{ij} is computed by the attention function that essentially finds the similarity between queries and keys using this dot-product so as to perform a soft-addressing process:

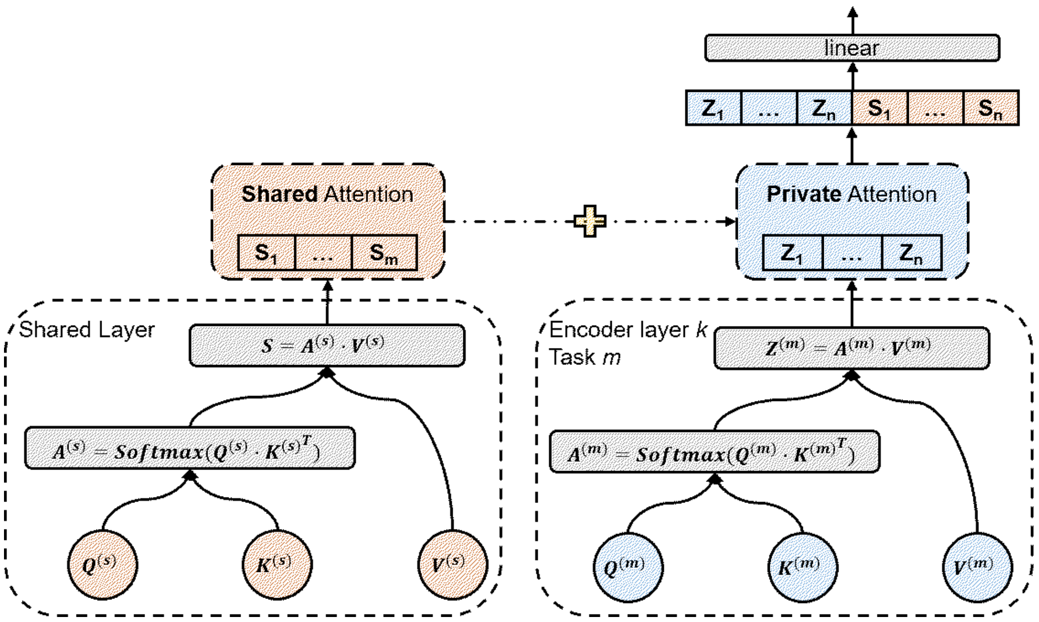
$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}} \tag{4}$$

where $W^Q \in \mathcal{R}^{d_x \times d_k}$, $W^K \in \mathcal{R}^{d_x \times d_k}$, $W^V \in \mathcal{R}^{d_x \times d_v}$ are parameter matrices. In practice, we usually set $d_k = d_v = d_z$.

Multi-head Attention. Instead of performing a single attention function with d_{model}-dimension keys, values, and queries, it is beneficial to linearly project the queries, keys, and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. Parallel attention function can be performed on each of these projected versions of queries, keys, and values, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values. This multi-head attention mechanism (MHA) allows the model to jointly attend to information from different representation subspaces at different positions.

Generally, once we capture the new sequences output from the multi-head functions as $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(h)}$ where $z^{(i)}$ means the attention score computed by the ith head. We concatenate these scores as $[\mathbf{z}^{(1)}\mathbf{z}^{(2)} \cdots \mathbf{z}^{(h)}]$ and multiple them with an additional weight matrix to align the dimension with targets. See Fig. 2 for an illustration of the multi-head attention model used in Transformer.

Masking Self-Attention Heads. In order to prevent from attending to subsequent positions, we apply attention masks, combined with the fact that the output embeddings are offset



linear
Z_1 … Z_n S_1 … S_n
Shared Attention
S_1 … S_m
Private Attention
Z_1 … Z_n
Shared Layer
$S = A^{(s)} \cdot V^{(s)}$
$A^{(s)} = Softmax(Q^{(s)} \cdot K^{(s)^T})$
$Q^{(s)}$
$K^{(s)}$
$V^{(s)}$
Encoder layer k
Task m
$Z^{(m)} = A^{(m)} \cdot V^{(m)}$
$A^{(m)} = Softmax(Q^{(m)} \cdot K^{(m)^T})$
$Q^{(m)}$
$K^{(m)}$
$V^{(m)}$

Fig. 3: A global Shared-Private multi-head attention scheme for multi-task learning.

by one position, ensuring that the predictions for position i can depend only on the known outputs at positions before i.

Shared-Private Attention Scheme. The main challenge of multi-task learning is how to design the sharing scheme. Despite the big success of recurrent neural networks in temporal pattern recognition, long-term information has to sequentially travel through all cells before getting to the present processing cell which means it can be easily corrupted by being multiplied much time by small negative numbers. This is the major cause of shared information forgetting. Fortunately, the Transformer helps drawing global dependencies between inputs and outputs by creatively relies entirely on the attention mechanism result in setting the distance between any two elements in a sequence to 1. Additionally, its good parallelism is well suited for multi-task learning. In this paper, we plan to provide a shared attention model MTL-Trans among multiple tasks based on the Transformer with two different sharing architectures.

C. General Global Shared Attention

Though the classic Transformer model employs an encoder-decoder structure, consisting of stacked encoder and decoder layers, in this work, we only consider the self-attention without giving concern to the encoder-decoder attention since our work focuses on a sequence self-modeling process. To exploit the shared information between different tasks, the general global shared attention architecture consists of private (task-specific) encoder layers and a shared (task-invariant) attention layer. The shared multi-head attention layer captures the shared information for all the tasks. In this architecture, the source time series is modeled by task-specific stacked self-attention based encoders. More formally, given an input time series sequence $\mathbf{x}^{(m)} = (x_1, x_2, \cdots, x_n)$ from a random selected task m, the shared attention information output $\mathbf{s}^{(m)} = (s_1, s_2, \cdots, s_n)$ from the public multi-head attention layer is defined as

$$\mathbf{s}^{(m)} = \text{MultiheadAttention}_{shared}(\mathbf{x}^{(m)}) \tag{5}$$

where $s_i \in \mathcal{R}^{d_s}$. Simultaneously, the task-specific attention output $\mathbf{z}_k^{(m)} = (z_1, z_2, \cdots, z_n)$ of multi-head attention from the kth encoder layer is computed as

$$\mathbf{z}_k^{(m)} = \text{MultiheadAttention}_k(\mathbf{z}_{k-1}^{(m)}) \tag{6}$$

where $\mathbf{z}_{k-1}^{(m)}$ is the output of the $(k-1)$th encoder from task m. The shared attention values and private values are then arranged in concatenated manner. The task-specific encoders take the output of the shared layer as input. The attention output from kth encoder layer is updated as

$$\mathbf{z}_k^{(m)} = \begin{bmatrix} \mathbf{z}_k^{(m)} \\ \mathbf{s}^{(m)} \end{bmatrix}^T W^O \tag{7}$$

where $W^O \in \mathcal{R}^{(d_s+d_z)\times d_z}$ is a parameter matrix that computes the weighted average information on a combination of both shared attention and private attention. This also helps align the outputs as the same dimension with our target sequences. The output is then fed into a fully connected feed-forward network (FFN) just as the original Transformer does. See Fig. 3 for the illustration of a general global attention sharing scheme.

D. Hybrid Local-global Shared Attention

Different from the general global attention sharing scheme, a hybrid local-global shared attention mechanism can make all tasks share a global attention memory, but can also record task-specific information besides shared information.

More generally, given an output sequence $\mathbf{z}_k^{(m)} = (z_1, z_2, \cdots, z_n)$ from the kth encoder layer for a random task m. The output will be fed back into the shared multi-head attention layer defined as

$$\mathbf{s}_{updated}^{(m)} = \text{MultiheadAttention}_{shared}(\mathbf{z}_k^{(m)}) \tag{8}$$

Again, the shared attention values and private outputs are arranged in concatenated manner and fed into the next encoder layer. The multi-head attention output from $(k+1)$th encoder layer is finally as

$$\mathbf{z}_{k+1}^{(m)} = \text{MultiheadAttention}_{k+1}(\begin{bmatrix} \mathbf{z}_k^{(m)} \\ \mathbf{s}_{updated}^{(m)} \end{bmatrix}) \tag{9}$$

By recurrently feeding outputs from task-specific encoders to the shared multi-head attention layer, this attention sharing architecture can enhance the capacity of memorizing while general global shared attention enables the information flowing from different tasks to interact sufficiently. Fig. 4 and Fig. 5 clearly describe the two attention sharing architectures and illustrate the difference.

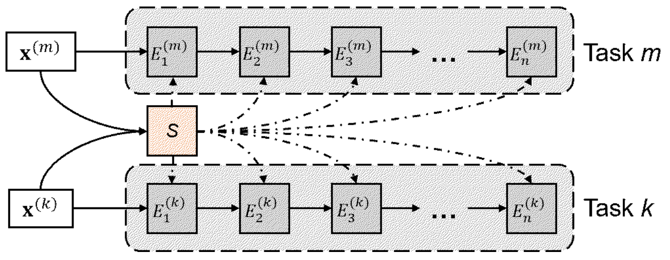


Fig. 4: General global attention sharing architecture.

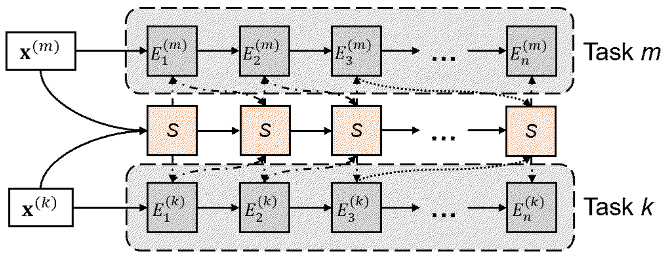


Fig. 5: Hybrid attention sharing architecture.

IV. Experiments

In this section, we investigate the empirical performances of our proposed architecture MTL-Trans on the following multi-task real-world dataset.

A. Dataset Description

TRA-MI This traffic dataset was published by Telecom Italia[1] and it contained network measurements in terms of total cellular traffic volume obtained from Milan city in Italy, where the city was partitioned into 100×100 grids of equal size $235m \times 235m$. The measurements were logged over 10-minute intervals between 1 Nov 2013 and 1 Jan 2014. Interestingly, each divided area is regarded as an independent task while there are also some hidden connections between each area. As an example, region A and region B are geographically adjacent which means these two areas are somehow related, such as sharing similar geographic information or municipal resources. If our proposed model can learn the similarity between different tasks, there is no doubt it will enhance the generalization ability to forecasting other related tasks even without pre-training it. For computing efficiency, we geographically merge all the small grids into 10 regions as 10 different tasks. Each region contains 1000 samples and naturally be marked as Task#1, Task#2, etc.

[1]https://dandelion.eu/datamine/open-big-data/

B. Benchmarks

We extensively compare MTL-Trans to a wide range of models for time series forecasting. Hyperparameter optimization is conducted using random search over a pre-defined search space, using the same number of iterations across all benchmarks for the same given dataset. Specifically, for single-task learning, the methods in our comparative evaluation are as follows.

- **LSTM** [11] Recurrent neural network with two-layer hidden long-short memory units and dropout applied.
- **Seq2Seq-Attn** [8], [26] Sequence to sequence network is a model consisting of two RNNs called the encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder reads that vector to produce an output sequence. Additionally, attention mechanism is applied.
- **DeepAR** [24] Auto-regressive RNN time series model which consists of an LSTM that takes the previous time points and co-variates as input for next time step.
- **DSSM** [23] Deep state-space model is a probabilistic time series forecasting approach that combines state-space models with deep learning by parameterizing a per-time-series linear state-space model with a jointly-learned recurrent neural network.

For multi-task learning, we compare our proposed approaches with the RNN-based generic sharing schemes.

- **SSP-MTL** [13], [14] An LSTM-based multi-task sequence learning model with a shared-private sharing scheme by stacking hidden states from different tasks.

For the single-task learning methods above, we trained each model on each task independently. All the models forecast one-time step forward with a consistent historical horizon.

C. Evaluation Metrics

These methods are evaluated based on three commonly used metrics in time series forecasting, including:

- Empirical Correlation Coefficient (CORR)

$$\text{CORR} = \frac{\sum_{t=1}^{n} \left(\hat{y}_t - \bar{\hat{y}}\right)\left(y_t - \bar{y}\right)}{\sqrt{\sum_{t=1}^{n} \left(\hat{y}_t - \bar{\hat{y}}\right)^2} \sqrt{\sum_{t=1}^{n} \left(y_t - \bar{y}\right)^2}} \tag{10}$$

- Root Mean Squared Error (RMSE)

$$\text{RMSE} = \mathbb{E}\left[\frac{\sum_{t=1}^{n}\left(\hat{y}_t - y_t\right)^2}{n}\right]^{1/2} \quad (11)$$

- Symmetric mean absolute percentage error (sMAPE)

$$\text{sMAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{\left(|\hat{y}_t| + |y_t|\right)/2} \quad (12)$$

where y_t is the ground truth value and $\hat{y}_t$ is the forecast value.

D. Training Procedure

We partition all time series of all tasks into 3 parts in chronological order – a training set (60%) for learning, a validation set (20%) for hyperparameter tuning, and a hold-out test set (20%) for performance evaluation. All time series have been preprocessed by applying Min-Max normalization such that all the values range from -1 to 1. Hyperparameter optimization is conducted via random search, using 50 iterations. Additionally, we use AdamW optimizer [32] with learning rate decay strategy applied: the learning rate of each parameter group decayed by gamma γ every pre-defined steps [2]. Full search ranges for all hyperparameters are below, with optimal model parameters listed in Table. I.

- **Shared and each task-specific embedding dimension** – 16, 32, 64, 128
- **Number of heads** – 2, 4, 8
- **Number of encoder layers** – 1, 2, 3, 4, 5, 6
- **Dimension of feed-forward layer** – 128, 256, 512, 1024
- **Dropout rate** – 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9
- **Mini-batch size** – 32, 64, 128, 256
- **Learning rate** – 0.0003, 0.003, 0.03
- **Max. gradient norm** – 0.01, 0.7, 1.0, 100.0
- **Learning rate decay rate** – 0.80, 0.95, 0.99
- **Decay step size** – 1.0, 5.0, 10.0

Following [14], [33], the training is achieved in a stochastic manner by looping over tasks:

1) Randomly select a task m.
2) Train a consecutive mini-batch $\mathbf{b}$ of samples from this task m.
3) Update the parameters for this task by gradient backward with respect to this mini-batch $\mathbf{b}$.
4) Go to Step 1.

Across all training process, all task-specific models were trained on the same single NVIDIA Tesla P100 GPU, and can be deployed without the need for extensive computing resources.

E. Loss Function

Both global-shared attention architecture and hybrid architecture are trained by minimizing the squared L^2 norm loss [34], summed across all outputs:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = (x_n - y_n)^2 \quad (13)$$

where N is the batch size. x and y are sequences of arbitrary shapes with a total of n elements each.

[2]All the experiments were done by using `Pytorch` library.

	TRA-MI
Dataset Details	
Target Type	$\mathbb{R}$
Number of Tasks	10
Network Parameters	
Embedding Dimension	32
Number of Heads	4
Number of Encoder Layers	2
Dimension of FFN	128
Dropout Rate	0.1
Training Parameters	
Mini-batch Size	64
Learning Rate	0.0003
Max Gradient Norm	0.7
Learning rate decay rate	0.95
Decay step size	1.0

TABLE I: Information on dataset and optimal training configuration.

F. Main Results

We compare our proposed two architectures with a wide range of baselines for both single-task forecasting and multi-task forecasting. Table. II summarizes the forecasting performance of our proposed method through three evaluation metrics. Our proposed MTL-Trans architectures significantly outperform all benchmarks over the variety of metrics and tasks. Moreover, the multi-task frameworks (SSP-MTL & ours) that jointly train the data outperform the single-task training framework as the model performance consistently tells. It demonstrates the shared information scheme across tasks can enhance modeling ability and capture both similarities and difference between tasks that finally benefits the model. With the help of the shared attention layer, the performances of all tasks by our proposed methods are significantly improved roughly around 2% across all metrics compared to the LSTM-based architecture SSP-MTL. This consistent improvements demonstrate the long term dependency modeling capability of the self-attention mechanism. Moreover, the two different attention sharing schemes share the winning tickets since the global sharing scheme performs better on Tasks 2, 3, 7, 8, 10 while the other one performs better on the rest tasks. As we described in section III, for tasks with highly similar patterns, a general global attention memory might be more suitable since consistent global attention helps capture the similarity and backward this information to each specific task more efficiently. For tasks with more inconsistent patterns, a local-global attention sharing scheme might be more appropriate because it can also record task-specific information besides globally shared information which can diversify each task-specific pipeline.

Fig. 6 shows some predicted time series by local-global sharing architecture. The predicted curve almost coincides with the groundtruth which further demonstrate the model's predictive capability. Fig. 8 further shows the good predictive

Task	Metrics	Single-Task				Multi-Task	MTL-Trans (ours)		Δ
		LSTM	Seq2Seq-Attn	DeepAR	DSSM	SSP-MTL	Global	Local-Global	
#1	CORR	0.7108	0.8005	0.8536	0.8640	0.8885	0.9045	**0.9049**	+1.85%
	RMSE	0.1138	0.1050	0.0986	0.0979	0.0952	0.0937	**0.0934**	+1.82%
	sMAPE	16.20%	14.91%	13.92%	13.96%	13.50%	13.22%	**13.17%**	+2.46%
#2	CORR	0.6781	0.7673	0.8149	0.8248	0.8492	**0.8628**	0.8623	+1.61%
	RMSE	0.1279	0.1176	0.1102	0.1105	0.1068	**0.1044**	0.1046	+2.25%
	sMAPE	14.66%	13.45%	12.53%	12.51%	12.17%	**11.96%**	**11.96%**	+1.74%
#3	CORR	0.6967	0.7784	0.8392	0.8418	0.8679	**0.8836**	0.8835	+1.81%
	RMSE	0.1129	0.1041	0.0977	0.0966	0.0941	**0.0925**	0.0931	+1.77%
	sMAPE	24.88%	22.76%	21.31%	21.44%	20.72%	**20.31%**	20.36%	+2.00%
#4	CORR	0.7339	0.8301	0.8882	0.8901	0.9211	0.9390	**0.9392**	+1.96%
	RMSE	0.1488	0.1354	0.1276	0.1278	0.1236	0.1211	**0.1205**	+2.50%
	sMAPE	83.24%	76.10%	71.75%	71.68%	69.42%	68.16%	**68.15%**	+1.83%
#5	CORR	0.7585	0.8575	0.9183	0.9242	0.9489	0.9638	**0.9646**	+1.65%
	RMSE	0.1134	0.1045	0.0980	0.0974	0.0948	0.0929	**0.0921**	+2.81%
	sMAPE	51.84%	47.89%	44.90%	44.84%	43.34%	42.48%	**42.03%**	+3.02%
#6	CORR	0.6775	0.7618	0.8149	0.8199	0.8457	0.8587	**0.8593**	+1.61%
	RMSE	0.1635	0.1493	0.1405	0.1394	0.1356	0.1331	**0.1330**	+1.98%
	sMAPE	70.18%	64.76%	60.65%	60.15%	58.63%	57.31%	**56.89%**	+2.98%
#7	CORR	0.7572	0.8523	0.9131	0.9149	0.9444	**0.9593**	0.9580	+1.58%
	RMSE	0.0728	0.0667	0.0628	0.0626	0.0608	**0.0594**	0.0602	+2.37%
	sMAPE	7.98%	7.32%	6.89%	6.88%	6.67%	**6.52%**	6.69%	+2.30%
#8	CORR	0.5364	0.6001	0.6454	0.6494	0.6679	**0.6814**	0.6785	+2.01%
	RMSE	0.1051	0.0972	0.0910	0.0902	0.0880	**0.0865**	0.0868	+1.77%
	sMAPE	9.75%	8.99%	8.40%	8.41%	8.15%	**7.97%**	7.99%	+2.28%
#9	CORR	0.6369	0.7144	0.7659	0.7726	0.7969	0.8107	**0.8111**	+1.78%
	RMSE	0.1865	0.1712	0.1606	0.1602	0.1558	0.1533	**0.1532**	+1.70%
	sMAPE	53.70%	48.84%	46.29%	45.76%	44.61%	**43.70%**	43.85%	+2.03%
#10	CORR	0.6457	0.7258	0.7706	0.7785	0.8035	**0.8231**	0.8227	+2.44%
	RMSE	0.1711	0.1571	0.1472	0.1468	0.1432	**0.1399**	0.1404	+2.28%
	sMAPE	33.79%	30.92%	29.00%	28.90%	28.04%	27.42%	**27.37%**	+2.40%

TABLE II: Model performance of two proposed attention sharing schemes against state-of-the-art neural models on **TRA-MI** dataset. Best performance in boldface. Δ represents the improvements compared to SSP-MTL. Experiments on all tasks are with the same historical horizon as 15 hours and forecast window as 10 minutes. Our proposed MTL-Trans consistently outperform all benchmarks over the variety of tasks and metrics.

ability of our proposed model as the simulated forecasting based on partial real data maintains the original pattern well.

G. Ablation Analysis

One intuitive question is that what if we only train each task-specific transformer encoder separately instead of sharing public multi-head attention? If we tune the hyperparameter of each task-specific model (As an example, increase the number of heads, deepen the encoder layers, etc.) such that they own similar amount of model parameters to the shared-attention scheme, removing the performance gain induced by model complexity, will they perform better than our shared attention model? Fig. 7 tells us the answer by showing the loss decrement against training steps among three architectures – global sharing scheme, hybrid local-global sharing scheme, and pure paralleled transformer encoders without sharing information. The fastest for loss descent is by local-global attention sharing scheme followed by the global attention sharing scheme and they eventually converged together. Compared to the shared attention architecture, the loss of pure encoders without sharing information drops more slowly, and the final result is not as good as the others which again demonstrates the effectiveness of sharing paradigm in multi-task learning.

To further illustrate the effectiveness of MTL-Trans in modeling the multi-task time series data, we summarize the following reasons:

- First of all, there are similarities between all related tasks and one fundamental mission in multi-task learning is to find these similarities out and take further advantages of them to benefit in solving other unseen tasks. The shared attention captures the similarity between different tasks and feedback on all related tasks. This is the main reason why this shared attention architecture can outperform naive models.
- Self-attention mechanism is the second hero that helps to make this happen. As we have discussed in section IV, the essence of the self-attention mechanism is a soft-addressing process. Our shared multi-head attention plays an important role that helps to record this query-key pairwise addressing information that can benefit other

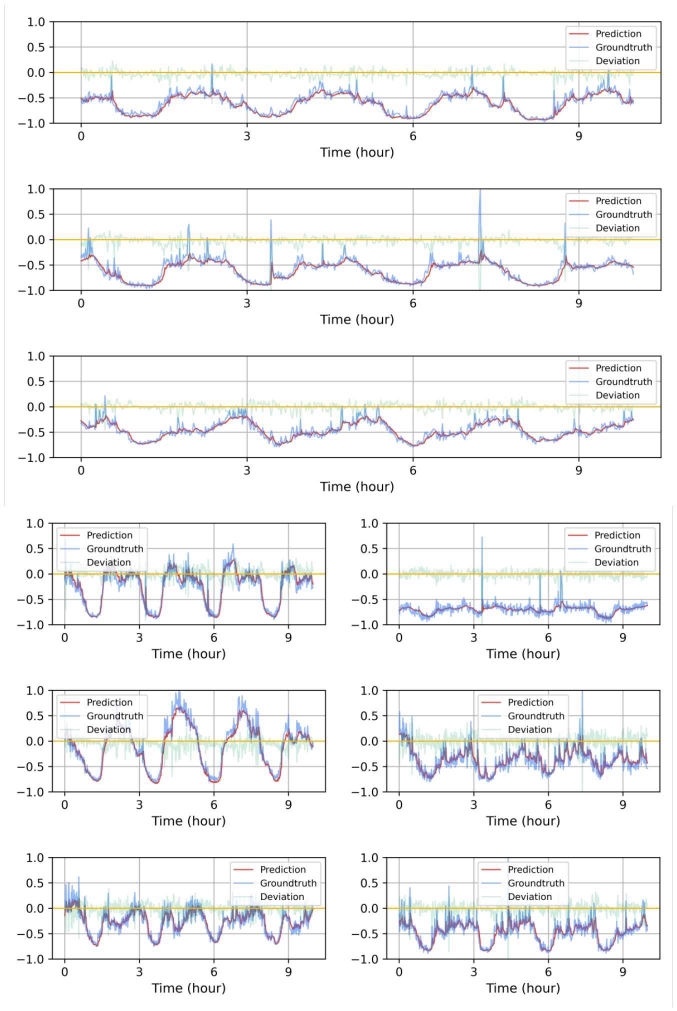


Fig. 6: Predicted time series of nine randomly selected samples from different tasks with the same historical horizon as 15 hours and forecast window as 10 mins on test set.

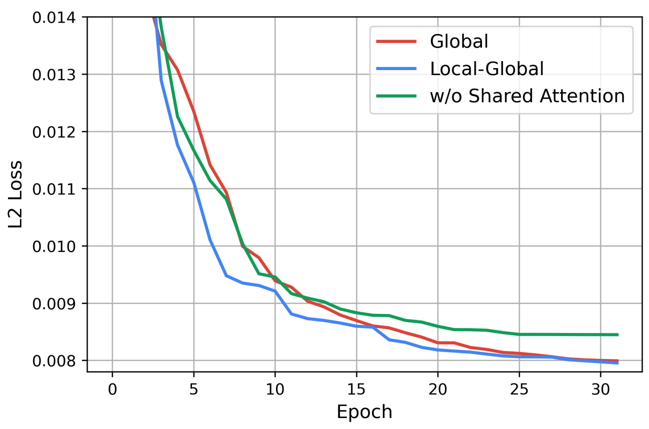


Fig. 7: Loss decrement against training steps among three architectures.

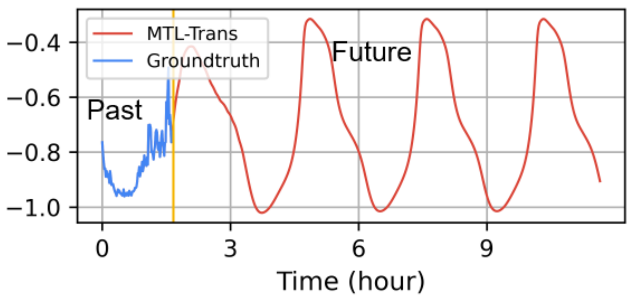


Fig. 8: Simulation test on future prediction based on real data.

unseen tasks under the hypothesis that similar tasks share similar self-addressing information.

V. Conclusion

In this paper, we presented a shared attention-based architecture with two different sharing schemes for multi-task time series forecasting. By setting an external public multi-head attention function for capturing and storing self-attention information across different tasks, the proposed architectures significantly improved the state-of-the-art results in multi-task time series forecasting on this multi-resource cellular traffic dataset **TRA-MI**. With ablation analysis and empirical evidence, we show the efficiency of the proposed architecture and the essence of why it succeeds. For future work, we will investigate the following two aspects: (1) applying the proposed model to other sequence modeling tasks such as machine translation; (2) developing other attention sharing schemes to further enhance the predictive ability; (3) finding another way or architecture that computes the shared multi-head attention more efficiently, e.g. the time and memory complexity of computing a multi-head self-attention function would cost $\mathcal{O}(L^2)$ where L is the length of input sequences. It could be hard to compute when the sequence length is very long or the computational power is limited.

Acknowledgment

The authors would like to thank all the anonymous reviewers for their insightful comments. We thank Dr. Jian Pei for the discussion and constructive suggestion on the paper organization and experiments design. This work was partially supported by the US National Science Foundation under grant IIS-1741279.

References

[1] G. D. F. Morales, A. Bifet, L. Khan, J. Gama, and W. Fan, "IoT big data stream mining," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-Augu, pp. 2119–2120, 2016.

[2] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of Things and Big Data Analytics for Smart and Connected Communities," *IEEE Access*, vol. 4, pp. 766–773, 2016.

[3] D. Asteriou, S. G. Hall, D. Asteriou, and S. G. Hall, "ARIMA Models and the Box-Jenkins Methodology," *Applied Econometrics*, pp. 275–296, 2016.

[4] D. wei Xu, Y. dong Wang, L. min Jia, Y. Qin, and H. hui Dong, "Real-time road traffic state prediction based on ARIMA and Kalman filter," *Frontiers of Information Technology and Electronic Engineering*, vol. 18, no. 2, pp. 287–302, 2017.

[5] T. Evgeniou and M. Pontil, "Regularized multi-task learning," *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 109–117, 2004.

[6] H. Drucker, C. J. Surges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in Neural Information Processing Systems*, pp. 155–161, 1997.

[7] A. Graves, "Generating Sequences With Recurrent Neural Networks," pp. 1–43, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3104–3112, 2014.

[9] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," pp. 1–15, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[10] M. T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, no. c, pp. 1–10, 2016.

[11] S. Hochreiter and J. Urgen Schmidhuber, "Long Shortterm Memory," *Neural Computation*, vol. 9, no. 8, p. 17351780, 1997.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[13] P. Liu, X. Qiu, and H. Xuanjing, "Recurrent neural network for text classification with multi-task learning," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2016-Janua, pp. 2873–2879, 2016.

[14] P. Liu, X. Qiu, and X. Huang, "Deep Multi-Task Learning with Shared Memory for Text Classification," pp. 118–127, 2016.

[15] J. Chen, X. Qiu, P. Liu, and X. Huang, "Meta multi-task learning for sequence modeling," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 5070–5077, 2018.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[17] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting," no. NeurIPS, 2019. [Online]. Available: http://arxiv.org/abs/1907.00235

[18] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using Holt-Winter's exponential smoothing," *2007 15th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2007*, pp. 310–314, 2007.

[19] G. Lai, W. C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018*, no. July, pp. 95–104, 2018.

[20] X. Ya, L. Xuejun, L. Carin, and B. Krishnapuram, "Multi-task learning for classification with Dirichlet process priors," *Journal of Machine Learning Research*, vol. 8, pp. 35–63, 2007.

[21] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, no. Ml, pp. 1–14, 2014.

[22] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–16, 2018.

[23] S. S. Rangapuram, M. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. NeurIPS, pp. 7785–7794, 2018.

[24] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, 2019.

[25] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 551–561, 2016.

[26] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[27] M. T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.

[28] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *32nd International Conference on Machine Learning, ICML 2015*, 2015.

[29] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting," no. Bryan Lim, pp. 1–27, 2019. [Online]. Available: http://arxiv.org/abs/1912.09363

[30] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pp. 543–550, 2010.

[31] A. P. Iyer, L. E. Li, M. Chowdhury, and I. Stoica, "Mitigating the latency-accuracy trade-off in mobile data analytics systems," *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pp. 513–528, 2018.

[32] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[33] R. Collobert and J. Weston, "A unified architecture for natural language processing," pp. 160–167, 2008.

[34] E. L. Lehmann and G. Casella, *Theory of Point Estimation , Second Edition Springer Texts in Statistics*, 1998.