MetaCitta: Deep Meta-Learning for Spatio-Temporal Prediction Across Cities and Tasks

Ashutosh Sao[1(✉)], Simon Gottschalk[1], Nicolas Tempelmeier[2], and Elena Demidova[3,4]

[1] L3S Research Center, Leibniz Universität Hannover, Hanover, Germany
{sao,gottschalk}@L3S.de
[2] Volkswagen Group, Hannover, Germany
nicolas.tempelmeier@volkswagen.de
[3] Data Science & Intelligent Systems Group (DSIS), University of Bonn, Bonn, Germany
elena.demidova@cs.uni-bonn.de
[4] Lamarr Institute for Machine Learning and Artificial Intelligence, Bonn, Germany
https://lamarr-institute.org/

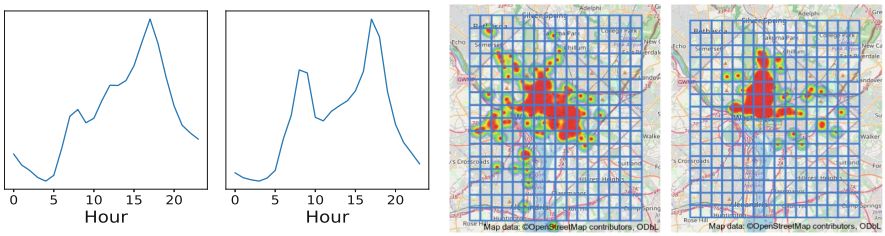
Abstract. Accurate spatio-temporal prediction is essential for capturing city dynamics and planning mobility services. State-of-the-art deep spatio-temporal predictive models depend on rich and representative training data for target regions and tasks. However, the availability of such data is typically limited. Furthermore, existing predictive models fail to utilize cross-correlations across tasks and cities. In this paper, we propose MetaCitta, a novel deep meta-learning approach that addresses the critical challenges of data scarcity and model generalization. MetaCitta adopts the data from different cities and tasks in a generalizable spatio-temporal deep neural network. We propose a novel meta-learning algorithm that minimizes the discrepancy between spatio-temporal representations across tasks and cities. Our experiments with real-world data demonstrate that the proposed MetaCitta approach outperforms state-of-the-art prediction methods for zero-shot learning and pre-training plus fine-tuning. Furthermore, MetaCitta is computationally more efficient than the existing meta-learning approaches.

Keywords: Spatio-Temporal Prediction · Meta-Learning · Pre-training

1 Introduction

Spatio-temporal predictions are critically important for planning and further developing smart cities. For instance, accurate bike and taxi demand prediction can enhance mobility services and city traffic management. Recently, deep learning-based approaches achieved high effectiveness in various spatio-temporal prediction tasks, including traffic forecasting and crowd flow prediction [1,15].

H. Kashima et al. (Eds.): PAKDD 2023, LNAI 13938, pp. 70–82, 2023.
https://doi.org/10.1007/978-3-031-33383-5_6



(a) CHI, Bike (b) NYC, Bike (c) DC, Bike (d) DC, Taxi

Fig. 1. (a) and (b) depict bike usage patterns in Chicago (CHI) and New York City (NYC), whereas (c) and (d) illustrate the bike and taxi usage in Washington, D.C. (DC), between 9 and 10 am.

The effectiveness of such approaches depends heavily on the availability of large amounts of training data. However, spatio-temporal data is typically (i) locked in organizational silos and rarely available across different cities and tasks and (ii) not sufficiently exploited for pre-training generalizable models. Consequently, we identify two crucial challenges in the spatio-temporal domain:

- **Lack of data of the target city and prediction task**: Spatio-temporal prediction typically requires data from the specific target city and the task of interest. However, while rich data is available for a few selected cities and tasks, data for the specific city and task of interest is often unavailable.
- **Lack of pre-training strategy in the spatio-temporal domain**: Pre-training requires a large amount of data from different tasks to learn an initialization for the target task, to converge better and faster. However, the adoption of pre-training in the spatio-temporal domain is currently limited.

Meta-learning is typically used to transfer knowledge from multiple cities (e.g., MetaStore [10] and MetaST [13]). However, existing approaches only learn from a specific task (e.g., bike demand prediction) and do not exploit the correlations between the tasks (e.g., taxi and bike demand prediction) and geographic regions. Examples of such correlations are illustrated in Fig. 1a and Fig. 1b, which indicate similar bike usage trends in central business areas in Chicago and New York City. Similarly, Fig. 1c and Fig. 1d indicate similar demand for bikes and taxis across regions. Therefore, we aim to build a predictive model that benefits from incorporating such correlations across tasks and cities.

In this paper, we propose MetaCitta, a novel deep meta-learning approach for spatio-temporal predictions across cities and tasks. In contrast to other approaches [10,13], MetaCitta adopts the knowledge not only from several cities but also different tasks. As mentioned above, different tasks in the same city can exhibit spatial correlations, and a task across cities can exhibit task-specific correlations. Therefore, we learn spatially invariant and task invariant feature representations using the Maximum Mean Discrepancy (MMD) [4]. These representations enable MetaCitta to make accurate predictions when data is unavailable for the target city and task. We also demonstrate how to adopt MetaCitta as a pre-training strategy when some target data is available for fine-tuning.

In summary, our contributions are as follows:

- We propose MetaCitta[1], a novel deep meta-learning algorithm for prediction in spatio-temporal networks.
- To the best of our knowledge, we are the first to utilize the knowledge of multiple tasks and cities to improve spatio-temporal prediction.
- MetaCitta outperforms best-performing baselines by 9.39% (zero-shot) and 3.86% (pre-training + data-abundant fine-tuning) on average on six real-world datasets regarding RMSE and is more efficient than state-of-the-art meta-learning methods regarding training time.

2 Problem Statement

In the following, we provide a formal definition of the problem of spatio-temporal prediction (based on [12,15]) via meta-learning from multiple cities and tasks.

Definition 1 (City and Region). *We represent a city c as an $m \times n$ grid map [12] with equally sized cells. We refer to each cell as a region.*

For example, we can split the city of Chicago into 20×20 equally sized regions, where each region represents a squared area.

Definition 2 (Spatio-Temporal Image). *A spatio-temporal image (as coined in [12], an image in short) $x_t^{c,\tau}$ of a city c and a task τ is a multi-channel image having the dimension $\mathbb{R}^{q \times m \times n}$ where q is the number of observations relevant for the task τ at a given time point t.*

For example, an image of Chicago for the taxi prediction task contains $q = 2$ observations (taxi pickup and drop-off) for each region at a time point t.

Definition 3 (Spatio-Temporal Prediction). *Given a sequence of spatio-temporal images of length i, $\mathcal{X}_{c,\tau} = \langle x_{t-i-1}^{c,\tau}, x_{t-i}^{c,\tau}, \ldots, x_{t-1}^{c,\tau} \rangle$, where $t-i-1, t-i, \ldots, t-1$ are consecutive time points, spatio-temporal prediction estimates the image at the next time point t, i.e., $x_t^{c,\tau} \in \mathbb{R}^{q \times m \times n}$.*

Given Chicago's spatio-temporal images for the taxi prediction in a given period, we aim to predict the image, i.e., taxi drop-off and pickup, at the next time point.

Definition 4 (Meta-learning from Multiple Cities and Tasks). *Given are a set of cities $\mathcal{C} = \{c_1, c_2, \ldots\}$, a target city $c_{target} \in \mathcal{C}$, as well as a set of tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots\}$ and a target task $\tau_{target} \in \mathcal{T}$. Training data is available in the form of image sequences for each combination of a city $c \in \mathcal{C}$ and a task $\tau \in \mathcal{T}$ except for the target task τ_{target} of the target city c_{target}: $\mathcal{D} = \{\mathcal{X}_{c,\tau} | c \in \mathcal{C}, \tau \in \mathcal{T}\} \setminus \{\mathcal{X}_{c_{target},\tau_{target}}\}$. The goal is to predict $x_t^{c_{target},\tau_{target}}$ based on $\mathcal{D}$.*

As an example, consider three cities (Chicago (*CHI*), Washington, D.C. (*DC*) and New York City (*NYC*)) and two tasks (bike demand prediction (*bike*) and taxi demand prediction (*taxi*)). With *NYC* as the target city c_{target} and *taxi* as the target task τ_{target}, the following image sequences are available:

[1] https://github.com/ashusao/MetaCitta.

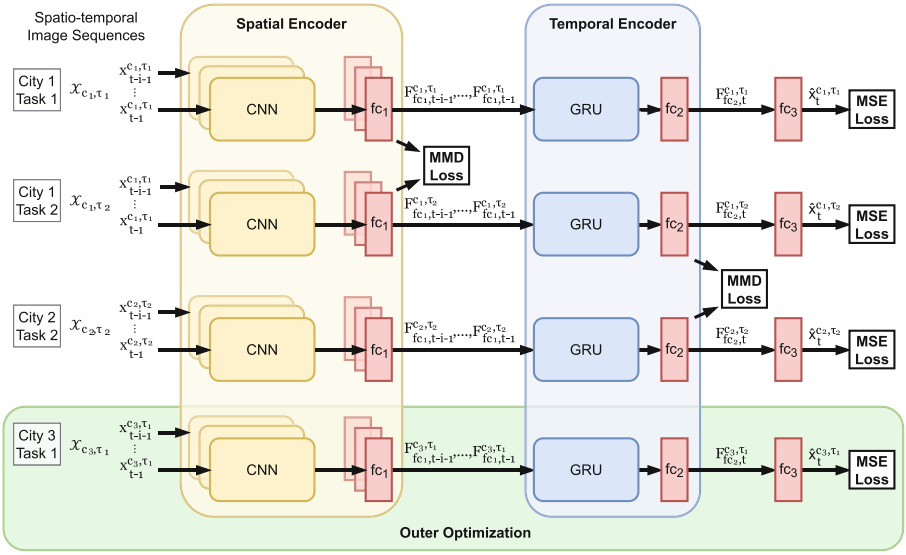


Fig. 2. Training procedure of MetaCitta with three cities (c_1, c_2 and c_3) and two tasks (τ_1 and τ_2). c_3 is the target city and τ_2 is the target task. The fully connected layers fc_1, fc_2 and fc_3 follow Eqs. (1), (2) and (3).

$\mathcal{D} = \{\mathcal{X}_{DC,bike}, \mathcal{X}_{CHI,bike}, \mathcal{X}_{DC,taxi}, \mathcal{X}_{CHI,taxi}, \mathcal{X}_{NYC,bike}\}$. Based on these image sequences, the goal of meta-learning from multiple cities and tasks is to predict the image $x_t^{NYC,taxi}$.

We also consider a variation of Definition 4 where limited data for the target task of the target city is available, specifically $\mathcal{D} = \{\mathcal{X}_{c,\tau} | c \in \mathcal{C}, \tau \in \mathcal{T}\}$ with $\mathcal{X}_{c_{target},\tau_{target}}$ restricted to only a few days.

3 The MetaCitta Approach

MetaCitta extends a typical spatio-temporal network (ST-Net) which has a spatial encoder, a temporal encoder, and a prediction component. Figure 2 illustrates MetaCitta's training with an example of three cities and two tasks. Spatio-temporal image sequences for each city-task pair are encoded by the spatial and temporal encoder. Spatial and task alignment is achieved through alignment losses (MMD loss). The outer optimization step trains on data from the target city (c_3) but not the target task (τ_2).

3.1 Spatial Encoder

The spatial encoder $\boldsymbol{f_s}$ captures the spatial dependencies between regions (see Fig. 1c and Fig. 1d). Given an image $x_t^{c,\tau}$ of a city c and a task τ at the time t, $F_{s,t}^{c,\tau} = \boldsymbol{f_s}(x_t^{c,\tau})$ is the encoded representation of the input image $x_t^{c,\tau}$. In MetaCitta, the spatial encoder $\boldsymbol{f_s}$ is a CNN consisting of three blocks where each

block contains a convolution layer, followed by batch normalization and a ReLU activation layer. We reduce the dimensionality of the spatial encoder's output $F_{s,t}^{c,\tau}$ by using a linear layer fc_1:

$$F_{\text{fc}_1,t}^{c,\tau} = \text{ReLU}(\boldsymbol{W}_{\mathbf{fc_1}} F_{s,t}^{c,\tau} + \boldsymbol{b}_{\mathbf{fc_1}}), \tag{1}$$

where $\boldsymbol{W}_{\mathbf{fc_1}}$ and $\boldsymbol{b}_{\mathbf{fc_1}}$ are trainable parameters, and $F_{\text{fc}_1,t}^{c,\tau}$ is the spatial representation of city c for the task τ at the time t.

3.2 Temporal Encoder

The temporal encoder $\boldsymbol{f_g}$ captures the temporal dependencies (see Fig. 1a and Fig. 1b). Given the spatial representations $F_{\text{fc}_1,t-i-1}^{c,\tau}, F_{\text{fc}_1,t-i}^{c,\tau}, \ldots, F_{\text{fc}_1,t-1}^{c,\tau}$ of a city c and a task τ over time, $F_{g,t}^{c,\tau} = \boldsymbol{f_g}(F_{\text{fc}_1,t-i-1}^{c,\tau}, F_{\text{fc}_1,t-i}^{c,\tau}, \ldots, F_{\text{fc}_1,t-1}^{c,\tau})$, is the encoded temporal representation at a time t. In MetaCitta, we use a GRU as the temporal encoder $\boldsymbol{f_g}$. The task-specific representation is generated by applying a linear layer fc_2 on $F_{g,t}^{c,\tau}$:

$$F_{\text{fc}_2,t}^{c,\tau} = \text{ReLU}(\boldsymbol{W}_{\mathbf{fc_2}} F_{g,t}^{c,\tau} + \boldsymbol{b}_{\mathbf{fc_2}}), \tag{2}$$

where $\boldsymbol{W}_{\mathbf{fc_2}}$ and $\boldsymbol{b}_{\mathbf{fc_2}}$ are trainable parameters.

3.3 Prediction

In an ST-Net, a linear operation and activation function transform the output into the desired shape and range. Then, a task-specific loss function is applied to train the network. MetaCitta's ST-Net uses a linear layer fc_3 and the *tanh* activation on the task-specific representation $F_{\text{fc}_2,t}^{c,\tau}$ of a city c and a task τ:

$$\hat{x}_t^{c,\tau} = tanh(\boldsymbol{W}_{\mathbf{fc_3}} F_{\text{fc}_2,t}^{c,\tau} + \boldsymbol{b}_{\mathbf{fc_3}}), \tag{3}$$

where $\boldsymbol{W}_{\mathbf{fc_3}}$ and $\boldsymbol{b}_{\mathbf{fc_3}}$ are trainable parameters, and $\hat{x}_t^{c,\tau} \in \mathbb{R}^{q \times m \times n}$ is the prediction at time t. We utilize the mean squared error as the task-specific loss:

$$L^{c,\tau} = \frac{1}{N} \sum_{i=1}^{N} (\hat{x}_t^{c,\tau} - x_t^{c,\tau})^2, \tag{4}$$

where N is the number of images, $\hat{x}_t^{c,\tau}$ and $x_t^{c,\tau}$ are the predicted and ground truth labels, respectively.

3.4 Training Procedure

Different tasks originate from different distributions. MetaCitta should learn the distribution-invariant properties from the tasks across the cities to perform well on an unseen task. To learn distribution-invariant properties, we use the Maximum Mean Discrepancy (MMD) [4] to minimize the distance between two

Algorithm 1. MetaCitta Training

Input: Data ($\mathcal{D}$), step size (α), cities ($\mathcal{C}$), tasks ($\mathcal{T}$), target city (c^{target}), target task (τ^{target}), randomly initialized model parameters (θ)
while *not converged* **do**
  for each $c_i \in \mathcal{C} \setminus \{c_{target}\}$ **do** ▷ Spatial Alignment
    for each $\tau_j, \tau_k \in \mathcal{T}$ **do**
      Extract pairs $(\mathcal{X}_{c_i,\tau_j}, \mathcal{X}_{c_i,\tau_k})$ from $\mathcal{D}$
      Compute $L^{spatial}_{mmd}, L^{c_i,\tau_j}, L^{c_i,\tau_k}$ using Equations (1, 3, 4, 5)
    $\theta = \theta - \alpha(\nabla_\theta L^{c_i,\tau_j} + \nabla_\theta L^{c_i,\tau_k} + \nabla_{\theta_s} L^{spatial}_{mmd})$
  for each $\tau_i \in \mathcal{T}$ **do** ▷ Task Alignment
    for each $c_j, c_k \in \mathcal{C} \setminus \{c_{target}\}$ **do**
      Extract pairs $(\mathcal{X}_{c_j,\tau_i}, \mathcal{X}_{c_k,\tau_i})$ from $\mathcal{D}$
      Compute $L^{c_j,\tau_i}, L^{c_k,\tau_i}, L^{task}_{mmd}$ using Equations (2, 3, 4, 6)
    $\theta = \theta - \alpha(\nabla_\theta L^{c_i,\tau_j} + \nabla_\theta L^{c_i,\tau_k} + \nabla_{\theta_\tau} L^{task}_{mmd})$
  for each $\tau_i \in \mathcal{T} \setminus \{\tau_{target}\}$ **do** ▷ Outer Optimization
    Extract $\mathcal{X}_{c_{target},\tau_i}$ from $\mathcal{D}$
    Compute L^{c_{target},τ_i} using Equation (4)
    $\theta = \theta - \alpha\nabla_\theta L^{c_{target},\tau_i}$

distributions. In this way, the network learns the common or invariant properties between the distributions and thus better generalizes to unseen distributions.

Different tasks of a city have the same underlying regions and thus share similar spatial characteristics. Therefore, to extract the spatially invariant representation between the tasks τ_1 and τ_2 in a city c, we apply the MMD constraint to the spatial representations generated by the spatial encoder. As a result, we compute the *spatial alignment loss* $L^{spatial}_{mmd}$:

$$L^{spatial}_{mmd} = MMD(F^{c,\tau_1}_{\mathrm{fc}_1,t}, F^{c,\tau_2}_{\mathrm{fc}_1,t}). \tag{5}$$

Similarly, to extract the task-invariant features from the task τ in two different cities, c_1 and c_2, we apply the MMD constraint on the task-specific representations generated by the temporal encoder. As a result, we compute the *task alignment loss* L^{task}_{mmd}:

$$L^{task}_{mmd} = MMD(F^{c_1,\tau}_{\mathrm{fc}_2,t}, F^{c_2,\tau}_{\mathrm{fc}_2,t}). \tag{6}$$

Finally, to enrich the model with the target city features, we perform knowledge transfer from all available tasks of the target city.

The MetaCitta training procedure is depicted in Fig. 2 and described in Algorithm 1. The algorithm takes the training data $\mathcal{D}$, a set of cities C, a set of tasks $\mathcal{T}$, the target city c^{target} and target task τ^{target} as input. The goal is to learn the model parameters θ, where θ_s are the parameters responsible for generating the spatial representation and θ_τ for the task-specific representation. The algorithm performs the following three steps:

1. The **Spatial Alignment (lines 3–7)** is performed between different tasks of a city on pairs of examples (line 5) and by updating the model parameters using the spatial alignment loss and the task-specific loss (lines 6–7).
2. The **Task Alignment (lines 8–12)** happens by extracting example pairs from the same task but of different cities (line 10), and with an update step using the task-alignment loss and the task-specific loss (lines 10–12).
3. The **Outer Optimization (lines 13–16)** step is done by directly performing an update step on all the available tasks of the target city.

Using the example from Sect. 2 with *NYC* as c_{target} and *taxi* as τ_{target}, the spatial alignment considers $(\mathcal{X}_{DC,bike}, \mathcal{X}_{DC,taxi})$ and $(\mathcal{X}_{CHI,bike}, \mathcal{X}_{CHI,taxi})$, while the task alignment involves $(\mathcal{X}_{CHI,bike}, \mathcal{X}_{DC,bike})$, and $(\mathcal{X}_{CHI,taxi}, \mathcal{X}_{DC,taxi})$. The outer alignment updates using $\mathcal{X}_{NYC,bike}$.

Pre-training. Following Definition 4, MetaCitta is tailored towards cases where no data from the target task τ_{target} in the target city c_{target} is available. However, if such data $\mathcal{X}_{c_{target},\tau_{target}}$ is (partially) available, MetaCitta can also be used for pre-training and fine-tuning, i.e., it first learns initialization weights from available data of other cities and tasks ($\{\mathcal{X}_{c,\tau} | c \in \mathcal{C}, \tau \in \mathcal{T}\} \setminus \{\mathcal{X}_{c_{target},\tau_{target}}\}$) and is then fine-tuned on data of the target task $\mathcal{X}_{c_{target},\tau_{target}}$. To use MetaCitta for pre-training, we skip the task alignment and instead perform spatial alignment between all available pairs of source cities and tasks. This method extracts the more general properties of the input data typically captured in the initial layers of a network [8,14], which is required for pre-training.

4 Evaluation Setup

This section describes the datasets, baselines, and experimental settings used to evaluate MetaCitta.

4.1 Datasets

We utilize two tasks (taxi and bike demand) from three cities: NYC, CHI, and DC. Each dataset consists of six months of data, with five months used for training and one month used as a test set. The data for TaxiNYC[2] (36M), TaxiCHI[3] (7M), TaxiDC[4] (3M), and BikeNYC[5] (9M) is from 01/2019 to 06/2019. The data for BikeCHI[6] (2.5M) and BikeDC[7] (1.2M) is from 07/2020 to 12/2020.

[2] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
[3] https://data.cityofchicago.org/Transportation/Taxi-Trips-2019/h4cq-z3dy.
[4] https://opendata.dc.gov/documents/taxi-trips-in-2019/explore.
[5] https://ride.citibikenyc.com/system-data.
[6] https://www.divvybikes.com/system-data.
[7] https://www.capitalbikeshare.com/system-data.

4.2 Baselines

We compare MetaCitta to the following seven baselines:

- The Historical Average (**HA**) is calculated by taking the mean value regarding the specific hour for the specific region.
- No pre-training (**NoPretrain**): The network is initialized with random weights and fine-tuned on the target task.
- **Joint**: The model is trained jointly by mixing all the samples.
- **MAML** [3] is a meta-learning method that learns an initialization from multiple tasks. We use the same underlying ST-Net as in MetaCitta.
- **MetaStore** [10] is a MAML-based approach that learns to generate the city-specific parameters based on the city's encoding during training.
- **MetaST** [13] is also based on MAML and learns a pattern-based spatio-temporal memory from source cities.
- **MLDG** [9] extends MAML for domain generalization. It randomly leaves one task out and updates its parameters on the left-out task during training.

4.3 Experimental Settings

In our experiments, each city is divided into 20×20 grid cells of size $1km^2$ each. We use the same underlying ST-Net and parameters for MetaCitta and the baselines (except for HA) to allow for a fair comparison. As spatial encoders, we use CNNs with 32 filters of size 3×3. As temporal encoders, we use GRUs where the input and hidden sizes are set to 256. Their input sequence length is 12 at an interval of 1 hour. The size of the fully connected layers (fc_1, fc_2) is 256. The batch size is 64, and each model is trained for 500 epochs on an NVIDIA GeForce GTX 1080 Ti (11 GB) at a learning rate of $1e^{-5}$ using an Adam optimizer. For the MAML-based approaches, the inner and outer learning rates are set to $1e^{-5}$, and there are 5 update steps.

5 Evaluation

In this section, we evaluate MetaCitta by comparing it to the baselines, by conducting an ablation study, and by training time comparison.

5.1 Comparison with Baselines

MetaCitta was evaluated in two settings: *zero-shot* and *fine-tuning*. Fine-tuning was done in two conditions: *data-limited* (15 days of target data) and *data-abundant* (5 months of target data). Results in Table 1 demonstrate that MetaCitta performs best in both settings and all datasets regarding RMSE.

In the zero-shot setting, HA performs worst, indicating that a simple heuristic approach cannot correctly capture the city's complex spatio-temporal dynamics. In this setting, MLDG is the best-performing baseline in terms of the

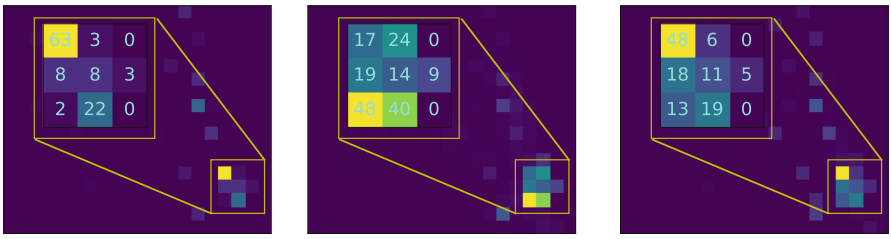
Table 1. Comparison of MetaCitta to the baselines, where RMSE is the Root Mean Squared Error and MAE is the Mean Absolute Error. Improv. (%) is the relative improvement of MetaCitta over the best baseline (underlined).

Approach	TaxiNYC		TaxiDC		TaxiCHI		BikeNYC		BikeDC		BikeCHI	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Zero-shot (no data from the target task of the target city used)												
HA	23.76	104.95	8.20	27.20	10.29	40.30	7.95	24.32	7.11	24.95	7.00	24.78
Joint	32.11	98.75	**1.72**	10.89	**4.54**	29.34	6.78	24.29	0.34	1.55	0.41	1.56
MAML	31.25	93.99	1.80	10.95	4.86	27.76	6.54	22.95	0.37	1.60	0.42	1.47
MetaStore	30.51	99.91	1.82	9.76	4.81	27.49	8.14	27.90	0.47	1.85	0.41	1.43
MetaST	25.64	85.72	1.78	10.62	5.20	25.16	6.84	23.91	0.40	1.63	0.43	1.51
MLDG	21.87	85.08	1.81	10.51	6.04	26.46	6.99	24.27	0.37	1.51	0.48	1.52
MetaCitta	**21.26**	**79.77**	1.85	**9.20**	5.34	**22.91**	**6.29**	**21.75**	**0.32**	**1.40**	**0.39**	**1.42**
Improv. (%)	2.79	6.24	-2.21	12.46	11.59	13.41	10.01	10.38	13.51	7.28	18.75	6.58
Fine-tuning (data-limited: 15 days of data from target task of target city)												
NoPretrain	170.60	220.13	52.61	72.70	130.39	169.89	55.58	73.18	11.65	4.75	15.53	19.63
Joint	5.39	22.29	0.89	4.34	**1.79**	9.10	4.28	15.54	0.35	1.05	0.48	1.50
MAML	5.66	23.32	0.89	4.18	1.91	10.26	4.23	15.36	0.57	1.34	0.78	1.55
MetaStore	8.54	38.01	0.99	5.12	2.16	13.74	5.87	20.45	0.45	1.36	0.53	1.68
MetaST	6.41	22.58	0.94	4.23	2.02	9.53	4.24	15.17	0.37	1.06	0.66	1.51
MLDG	6.21	24.61	0.94	4.46	2.21	12.87	4.70	17.15	0.57	1.41	0.51	1.56
MetaCitta	**5.14**	**20.95**	**0.85**	**4.09**	1.81	**8.96**	**4.14**	**14.76**	**0.31**	**1.03**	**0.45**	**1.35**
Improv. (%)	4.64	6.01	4.49	5.76	-1.12	1.54	3.28	5.02	11.42	1.90	6.25	10.0
Fine-tuning (data-abundant: 5 months of data from target task of target city)												
NoPretrain	21.89	40.57	4.60	9.48	7.94	22.05	7.48	19.77	1.10	1.99	2.09	4.01
Joint	3.43	14.01	0.61	2.78	1.09	6.21	2.09	7.11	0.27	0.87	0.34	0.94
MAML	3.38	14.11	0.70	2.80	1.21	6.19	2.18	7.01	0.31	0.92	0.35	0.96
MetaStore	5.42	24.81	0.77	3.81	1.52	10.19	3.11	11.30	0.31	0.96	0.39	1.17
MetaST	3.49	14.40	0.63	2.76	1.19	6.31	2.13	7.02	0.27	0.86	0.34	0.95
MLDG	3.52	14.73	0.66	2.93	1.16	6.42	2.12	7.23	0.29	0.92	0.37	1.05
MetaCitta	**3.24**	**13.24**	**0.55**	**2.65**	**1.05**	**5.91**	**1.95**	**6.92**	**0.25**	**0.85**	**0.31**	**0.91**
Improv. (%)	5.54	5.49	9.83	4.68	3.70	4.83	6.69	2.67	7.41	2.29	8.82	3.19

RMSE because it is trained to perform particularly well on an unseen task by updating its parameters based on the left-out task.

MetaCitta and the pre-trained baselines outperform NoPretrain in fine-tuning due to their ability to extract generic spatio-temporal properties that aid in convergence to the target task. These properties include traffic behavior, such as high demand during peak hours (morning and evening) and low off-peak demand (e.g., after midnight). Joint, MAML, and MetaST perform better than MLDG as they are trained to perform well on all source tasks and adapt quickly to new tasks. MetaStore performs relatively worse in fine-tuning than in the zero-shot setting, as it only fine-tunes the final prediction layers (fc_2 and fc_3), reducing its adaptability to the target task.

On average across datasets, compared to the respective best baselines, MetaCitta is 9.39% better than MLDG in the zero-shot setting, 5.04% better than Joint in the data-limited fine-tuning setting, and 3.86% better than Joint in



(a) Ground Truth (b) Joint (c) MetaCitta

Fig. 3. Comparison of Joint and MetaCitta on taxi pickup prediction of Chicago, June 6th 2019, between 4 and 5 am. Nine selected regions and their number of taxi pickups in the mentioned hour are shown in detail.

the data-abundant fine-tuning setting regarding the RMSE. According to the MAE, MetaCitta exceeds the baselines in all cases except three (e.g., zero-shot on TaxiCHI), where Joint performs better. Since MetaCitta consistently has a lower RMSE, we conclude that MetaCitta performs better in predicting extreme values, such as high or low demand values during or after peak periods, while Joint mainly predicts average values for each region.

To further investigate this behavior, we visualize the predictions of Joint and MetaCitta in the data-limited fine-tuning setting on the TaxiCHI dataset. Figure 3 illustrates the ground truth and the predicted pickup demands in different city regions between 4 and 5 am. The regions to the south and southwest of the zoomed area represent the Chicago Loop area, which is Chicago's central business district. The Northwest region is Chicago's Near North Side, a residential area. While the taxi demand in the Chicago Loop area is high during the day, it is low during off-peak hours. On the other hand, the demand for taxis in the residential area is higher compared to the business area. This is because people use taxis, as public transport does not run at 4 am. While MetaCitta correctly captures this shift in demand, as illustrated in Fig. 3c, Joint continues to forecast average high values in the business district, as observed in Fig. 3b.

5.2 Ablation Study

We analyze the contribution of MetaCitta's training components: spatial alignment, task alignment, and outer optimization. By removing one component at a time, we evaluate their effectiveness and present the results in Table 2.

Removing MetaCitta's outer optimization step leads to the most significant drop in performance, indicating the importance of having some knowledge of the target city, even if from another task. This knowledge can be obtained directly from other tasks, as observed in Fig. 2a and 2b, where similar regions indicate similar spatio-temporal behavior. Also, the performance decreases when spatial and task alignment is removed, indicating the impact of adding the MMD losses.

Table 2. Change in performance after removal of a component from MetaCitta.

Approach	TaxiNYC		TaxiDC		TaxiCHI		BikeNYC		BikeDC		BikeCHI	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
MetaCitta	**21.26**	**79.77**	1.85	**9.20**	5.34	**22.91**	**6.29**	**21.75**	**0.32**	**1.40**	0.39	**1.42**
no spatial align	23.61	86.75	2.22	9.35	5.02	23.96	6.45	22.55	0.42	1.43	**0.38**	1.43
no task align	46.58	126.72	**1.68**	10.80	**4.65**	27.04	6.30	22.39	**0.32**	1.42	0.39	1.49
no outer optim	31.64	108.38	6.56	18.26	14.52	43.00	9.99	28.06	1.35	3.41	0.77	2.08

5.3 Training Time Comparison

The training times for MetaST, MAML, MLDG, MetaStore, MetaCitta and Joint on the Chicago bike prediction task are 107.21, 106.68, 87.71, 7.99, 5.65 and 3.18 hours, respectively. As MetaCitta applies an extra alignment loss to extract invariant features, it needs slightly more training time than Joint, which has a simplified design. MetaCitta requires less training time than MetaST (94.73%), MAML (94.70%), MLDG (93.56%) and MetaStore (29.28%), as these baselines perform a two-stage optimization requiring time-consuming calculation of higher-order derivatives. Overall, MetaCitta performs most precisely and trains much faster than state-of-the-art meta-learning.

6 Related Work

Deep learning has recently shown great success and is widely adopted for spatio-temporal predictions [1,15]. However, the success of these approaches depends on the availability of large amounts of data, which is typically a bottleneck.

Typically, transfer learning is used to deal with data scarcity, where a network trained for a specific task is fine-tuned on the target task. Recently, meta-learning has gained popularity as a way to learn from multiple tasks. Unlike transfer learning, meta-learning does not specialize in a specific task. Instead, it focuses on finding the parameters from the source tasks, so it can be quickly adapted to the target task (i.e., learn to learn) [6]. Model Agnostic Meta Learning [3] (MAML) and Meta Learning for Domain Generalization [9] (MLDG) are state-of-the-art meta-learning approaches widely used in different domains. MAML performs two optimization steps: task-specific updates on the inner step and global meta-updates on the outer step. MLDG extends MAML by leaving out a task at the inner level and performing meta-updates based on the left-out task.

Motivated by these approaches, several attempts have been made to transfer knowledge in the spatio-temporal domain. [2,5,7,11,12], use transfer learning to transfer knowledge from a data-rich source city to a data-poor target city. MetaStore [10] and MetaST [13] are MAML-based approaches to transfer knowledge from multiple cities to a target city. However, existing approaches transfer the knowledge for a specific task, introduce additional parameters and require small amounts of data from the target city and target task.

In contrast to these methods, MetaCitta learns from different tasks in multiple cities and can be effectively used in zero-shot and fine-tuning settings.

7 Conclusion

In this paper, we presented MetaCitta, a novel deep meta-learning approach for spatio-temporal predictions in cases where no or only limited training data of a target city and the task is available. MetaCitta leverages knowledge across cities and tasks by learning spatial and task-invariant feature representations. MetaCitta outperforms state-of-the-art meta-learning approaches regarding the RMSE in zero-shot and fine-tuning settings and requires 94.70% less training time compared to the state-of-the-art meta-learning approach MAML.

Acknowledgements. This work was partially funded by the DFG, German Research Foundation ("WorldKG", 424985896), the Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany ("d-E-mand", 01ME19009B), and DAAD, Germany ("KOALA", 57600865).

References

1. Bai, L., Yao, L., Li, C., Wang, X., Wang, C.: Adaptive graph convolutional recurrent network for traffic forecasting. In: NIPS (2020)
2. Fang, Z., et al.: When transfer learning meets cross-city urban flow prediction: spatio-temporal adaptation matters. In: IJCAI (2022)
3. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning (ICML) (2017)
4. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. J. Mach. Learn. Res. **13**(1), 723–773 (2012)
5. He, T., Bao, J., Li, R., Ruan, S., Li, Y., Song, L., et al.: What is the human mobility in a new city: transfer mobility knowledge across cities. In: TheWebConf (2020)
6. Huisman, M., van Rijn, J.N., Plaat, A.: A survey of deep meta-learning. Artif. Intell. Rev. **54**(6), 4483–4541 (2021). https://doi.org/10.1007/s10462-021-10004-4
7. Jin, Y., Chen, K., Yang, Q.: Selective cross-city transfer learning for traffic prediction via source city region re-weighting. In: SIGKDD (2022)
8. Lee, H., Grosse, R., Ranganath, R., Ng, A.: Unsupervised learning of hierarchical representations with convolutional deep belief networks. Commun. ACM **54**(10), 95–103 (2011)
9. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Learning to generalize: meta-learning for domain generalization. In: AAAI (2018)
10. Liu, Y., et al.: Metastore: a task-adaptative meta-learning model for optimal store placement with multi-city knowledge transfer. ACM TIST **12**(3), 1–23 (2021)
11. Wang, L., Geng, X., Ma, X., Liu, F., Yang, Q.: Cross-city transfer learning for deep spatio-temporal prediction. In: IJCAI (2019)
12. Wang, S., Miao, H., Li, J., Cao, J.: Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks. IEEE TITS **23**(5), 4695–4705 (2021)
13. Yao, H., Liu, Y., Wei, Y., Tang, X., Li, Z.: Learning from multiple cities: a meta-learning approach for spatial-temporal prediction. In: TheWebConf (2019)

14. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
15. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: AAAI (2017)