Time Series Classification with Meta Learning

Aman Gupta and Yadul Raghav

Department of Computer Science and Engineering
Indian Institute of Technology (BHU)
Varanasi, India 221–005

Abstract. Meta-Learning, the ability of learning to learn, helps to train a model to learn very quickly on a variety of learning tasks; adapting to any new environment with a minimal number of examples allows us to speed up the performance and training of the model. It solves the traditional machine learning paradigm problem, where it needed a vast dataset to learn any task to train the model from scratch. Much work has already been done on meta-learning in various learning environments, including reinforcement learning, regression task, classification task with image, and other datasets, but it is yet to be explored with the time-series domain. In this work, we aimed to understand the effectiveness of meta-learning algorithms in time series classification task with multivariate time-series datasets. We present the algorithm's performance on the time series archive, where the result shows that using meta-learning algorithms leads to faster convergence with fewer iteration over the non-meta-learning equivalent.

Keywords: Time Series, Classification, Meta Learning, Few Shot Learning, Convolutional Neural Network

1 Introduction

Computers have always been an immense source of fascination for human beings. Be it the computers beginning large computations, forecasting, accounting, it has always fascinated human beings. Furthermore, with the origin of the field of Artificial Intelligence, humans have been eluded to it never than before. The concept of an artificial agent learning to make decisions and perform tasks like humans is very much intriguing in itself. Learning (also referred to as Machine Learning due to the involvement of computers) has been studied in three significant categories Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Besides all these advances in supervised, unsupervised, and reinforcement learning, learning real-life tasks, learning algorithms, and carrying out simple instructions, learning from a few examples, is still regarded as a challenging problem. These are the class of problems that came to highlight in recent years; however, they have been known since the early 70's, known as Meta-Learning. Meta-Learning can be found as early as in the works of Donald B. Maudsley, where he referred to it as "the process by which learners become aware of and increasingly in control of habits of perception,

David C. Wyld et al. (Eds): NLP, JSE, MLTEC, DMS, NeTIOT, ITCS, SIP, CST, ARIA - 2020
pp. 189-201, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.101415

inquiry, learning, and growth that they have internalized". John Biggs redefined the same in a significantly more straightforward language as "being aware of and taking control of one's learning". In the context of computer science, meta-learning can be defined as simple as knowledge adaptability. It can be as simple as learning to identify previously seen objects with significantly fewer examples (few-shots learning), learning a task from some demonstrations (learning from demonstrations), or even following simple instructions.

Humans and animals adapt to any environment much faster and more efficiently. Whenever they try to learn any new skills or concepts, they hardly start from scratch. They start from the skills learned in the experience related to the new task, based on the knowledge they already have, that worked well before. For example, A child who has seen a dog, flower, bird only a few times can easily distinguish between them. A person who speaks one language can quickly learn to speak other languages with little practice, or a person who knows how to ride a bike can ride a motorbike with little practice. With every learned skill related to the given task domain, they can learn a new skill with a few examples and training. They simply learn how to learn across different tasks. Machine learning systems have surpassed humans at many tasks; it still requires training with a large number of samples on a specific task, and generally, models are trained from scratch using these samples to reach the same. So, it is not entirely fair for an AI algorithm to compare with humans as humans have prior knowledge and experience in their brain and DNA. Is it possible to imbue an AI system model with similar properties as a human with the ability to learn from experience and knowledge to learn new concepts and skills quickly with a few training examples rather than learning for scratch.

Much of the work has been proposed over the past few years. The earlier works address the problem of meta-learning as Few-Shot learning. The concept behind this is to design a deep neural network that can learn by simulating the datasets with very few instances, just like the babies learn to identify objects by seeing only a picture or two. In [1], the author proposed a method using the convolutional siamese neural network to do a few-shot image classification. Consecutively next year, an embedding method called Full Contextual Embeddings (FCE) [2], which uses bidirectional LSTM to encode the input vector to do the K-shot classification. With the recent success of optimization-based techniques, MAML [3], a model-agnostic algorithm for meta-learning, has been proposed, which has the ability to combined with any other trained gradient descent model, applicable to a variety of machine learning task. They looked at the learning process as maximizing the sensitivity of the loss functions of new tasks with respect to the model parameters such that small changes in the parameters will produce significant improvements on the loss function of any task. Training the model with a small number of gradient steps

with a few samples of a task gives good generalization performance on that task.

In this work, we have focused on the optimization-based approach of meta-learning for time series classification tasks. Reptile [4], a meta-learning algorithm, is used as a framework for the experiment combined with a baseline architecture of a convolutional neural network, where the convolutional layer acts as a feature extractor for our classification model. We demonstrate the experiments on the UCR time-series datasets [6]. For the evaluation purpose, we have compared the loss function curve of algorithms showing that the meta-learning approach favors time-series tasks over non-meta-learning counterparts resulting in faster convergence over a sample of tasks with fewer iteration. We successfully demonstrate that reusing knowledge from past tasks and combining them with deep neural networks may provide a better result. To the simplest of our knowledge, this is the first successful attempt in adapting meta-learning for a time series classification task.

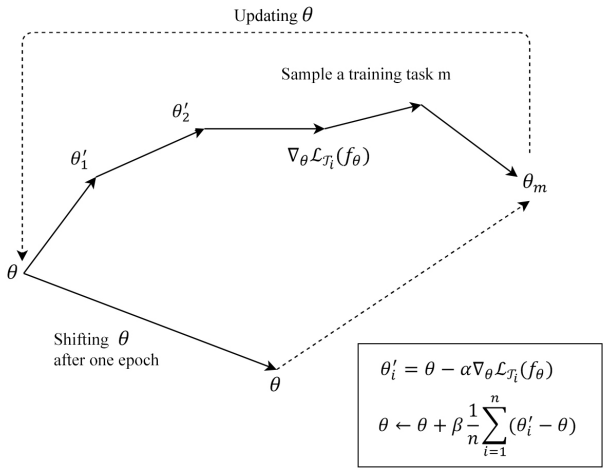

Fig. 1: A Meta-learning model optimizes the model parameters θ that can quickly adopt for a new task with parameters θ_i'. The model updates the initialization parameter θ.

Organization. The rest of the paper is organized as follows. Section 2 reviews the existing research work is done in the field of Meta-Learning and Time-Series Classification and builds the base for our main objective. The proposed work to scale the Meta-Learning algorithm to time-series classification tasks and demonstrate its performance on some dataset from the UCR time-series achieve. Section 4 gives

the details of the experiment conducted using the explained methods, discusses an experimental study consisting of an evaluation strategy—finally, section 5, where we conclude with our observations in the chapter.

2 Literature Review

2.1 Time Series Classification

TSC has gained much attention in recent years due to its vast application in various domains such as financial services, human activity recognition, healthcare, weather forecasting [11]. Time series is nothing but just a measurement of statistical data taken several discrete times in chronological order. Mathematically, it can be written as,

$$h = f(t) \tag{1}$$

where h is the phenomena (function) at any given time.

Formally, TSC problem can be defined as follows: for a given set of classes Y, a training dataset $T = \{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$ is a collection of pairs (X_i, Y_i) where $X_i = [X_i^1, X_i^2, \cdots, X_i^m]$ consists of M different univariate time series with Y_i as it is their corresponding class label $Y_i \subset Y$. The goal is to learn a classifier or model on a dataset T which, when fed with unseen time-series data points, the classifier is expected to predict its class correctly i.e., finding a function F such that $F(X_j) = Y_j$, and $Y_i \subset Y$.

As the earliest baseline, Lines and Bagnall [14] first demonstrate the effectiveness of distance-based methods such as Euclidean distance or Dynamic time warping (DTW) coupled with the nearest neighbor (NN) classifier to work directly on raw time series data. In the paper [15], the author proposed a feature-based approach extracting a set of features representing the global/local patterns of time series. These sets of features are combined to form Bag-of-words (BoW) or Bag-of-features and feed to the classifier [19] [20]. Several ensemble-based approaches have been explored Elastic Ensemble (PROP) [14], transform-based ensembles (COTE) [21] combining different classifiers over different time series representations. All these approaches need heavy preprocessing; feature extraction has higher time complexity. Earlier, the researchers neglected the fact of pure feature learning. With the success of deep learning models after 2012 [8], researchers began to exploit the idea of feature learning instead of handcrafted features.

2.2 Meta Learning

Meta-Learning is one of the most exciting areas of research in Artificial Intelligence and has been tackled for a long time. It has been tackled through different researchers as meta learner [16], few shots learning [17], meta reinforcement learning [18]. However, the latest mega boom in meta-learning began with the inception of Deep Learning with meta-learning. Much of the work has been proposed over the past few years. The earlier works address the problem of meta-learning as Few-Shot learning. The concept behind this is to design a deep neural network that can learn by simulating the datasets with very few instances, just like the babies learn to identify objects by seeing only a picture or two.

Over the recent year, many of the work has been published related to meta-learning, which classified into three approaches:- metric-based, model-based, and optimization based [10]. In a metric-based approach, the core idea is to learn embedding vectors of input data explicitly and use them to learn the best kernel functions. In the model-based approach, the idea is to design a model that updates its parameters rapidly with a few training steps, learning from the knowledge stored in the memory from the past training to learn a new task. Santoro *et al.* [12] built upon Differentiable Neural Computer and propose a new model, Memory-Augmented Neural Network (MANN). They described a few-shot learning specific data feeding pipeline wherein the answer (output label) of a previous input image is sent concatenated along with the current input. The idea is to encode new information in external memory and using this memory to updates its model parameters rapidly with a few training steps. Thus, the model, with external memory, through its controller had to learn to store the input Santoro *et al.* [12] representation in the external memory, associate the label provided with the current input with the previous input and retrieve the content of the relevant memory locations to produce the answer for the current query. Another approach is optimization-based, the core idea of learning a way to adjust the optimization parameters of the algorithm so that model converges within a small number of optimization steps learning with a few examples. Andrychowicz *et al.* [13] put forward a revolutionary idea of learning the optimizing function instead of hand designing it. The logic being, since we usually are learning a classifier "function" from our training data. Therefore we can also learn an optimizer function that performs the optimization process better. The idea seems simple but is very much revolutionary. This model puts forward meta-learning as well as transfer learning, wherein we can transfer the learned optimizer to other tasks. This meta learner can be scaled to thousands of parameters. The learned LSTM optimizer performed significantly well as compared to the state of the art models such as Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSProp), Adam Optimizer (ADAM), etc.

Finn *et al.* [3] introduced the concept of model agnostic meta-learning (MAML). Here, there is a meta learner and a learner. The meta learner trains the learner on a training set that contains a different number of tasks. Through the meta learner, the model will acquire prior experience from training and will learn the common feature representations of the task. Thus, whenever there is a new task, the model will use its prior experience and will be fine-tuned to the new task on a small number of training data. MAML provides a good initialization of model parameters to achieve optimal fast learning.

3 Proposed Work

Frequently, tasks in meta-learning can be expressed as the problem of optimizing an objective function of a model $f(\theta)$ defined across the distribution of task over some domain $\theta \in \Theta$. So, the goal, in this case, is the meta-learning model should be able to find an optimal initialization for parameters of a randomly sampled task where each task is associated with dataset T_i (each dataset is considered as one data sample). Mathematically, it can be expressed as

$$\theta^* = argmin_\theta \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathcal{L}_{\mathcal{T}_i} f(\theta)] \tag{2}$$

Our work is built on the recently proposed model Reptile, an optimization-based approach somewhat similar to MAML as they are both trained with gradient descent and model-agnostic, but much simpler in implementation and training. Reptile works repeatedly sampling a task, training on it through multiple gradient descent steps, therefore, shifting the model weights towards the new parameters of the unseen task. The algorithm updates the model into two stages: First, it considers the model as function $f(\theta)$ with parameter θ, then the classifier is trained on a given task T_i, changing the model parameter θ to $\theta_i^{'}$.

$$\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i} f(\theta) \tag{3}$$

Generally, a task contains a training set having few or limited examples for each of the classes with one or more test examples (few-shot learning) [7]. The reptile objective is to learn the optimal initialization for the parameters of the neural network model such that the classifier learns faster while optimizing at test time, leads to generalizing the model with fewer examples from the test task. In the final optimization step, it simply updates θ to $(\theta_i^{'} - \theta)$ as a reptile gradient applied with stochastic gradient descent (SGD).

$$\theta = \theta + \beta \frac{1}{n} \sum_{i=1}^{\infty} (\theta_i^{'} - \theta) \tag{4}$$

In this work, we have used the same architecture proposed in the paper [6] as the baseline architecture for our classifier. The author proposed a straightforward

deep neural network-based architecture for TSC, which gives remarkable results with 44 UCR time-series datasets. They have tested on three deep neural network architectures, Multilayer Perceptrons (MLP), Fully Convolutional Networks (FCN), Residual Network, to provide a fully comprehensive baseline. For our experiment, we have used only FCN architecture, which can be applied to the dataset without any feature crafting and heavy data preprocessing. An intuition behind using an FCN network is that applying several convolutional layers on time series would be more helpful in learning a discriminative feature for the classification task.

3.1 Network Architecture

Deep convolutional neural network (CNN) has gained much attention in many different domains like regression, classification task, natural language processing (NLP), information retrieval, etc. after the AlexNet [8], won the 2012 ImageNet competition. With the increasing success of this architecture in different research fields, researchers started exploring the success of convolutional network architecture for time-series analysis. As we have seen applying the convolution on the image by sliding the filters in two dimensions (height and width), similarly, we could apply the convolution over time-series by sliding the filter to exhibit in one dimension (time).

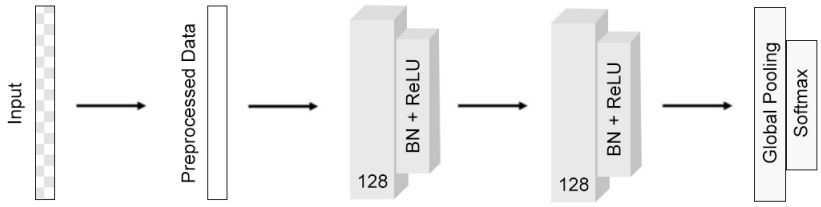

Fig. 2: The baseline architecture of fully convolutional neural network used for training the meta-learning algorithm.

In our problem setting, applying FCN over time-series helps extract multiple discriminative features for the classification task. The softmax layer finally outputs a probability distribution over the class variables in a task. The basic block of FCN mainly consists of three components viz., convolutional layer, followed by batch normalization (BN), and a final non-linearity function Rectified Linear Unit (ReLU). $\otimes$ is the convolution operator, applied with a filter W on a time series data x, and a bias parameter b.

$$\mathsf{C} = \mathsf{ReLU}[\mathsf{BN}(\mathbf{W} \otimes \mathbf{x} + \mathbf{b})] \tag{5}$$

The architecture is built up by stacking three convolutional blocks. The final extracted features from the convolutional block are fed to the global average polling layer reducing the number of weights in a model, thus preventing the risk of overfitting. Batch normalization has been performed after each convolutional layer helping the network to converge quickly and improve generalization. In the end, the softmax layer is applied for classification.

3.2 Preprocessing

Normalization of the dataset has been performed across each channel (attribute); missing values are filled with zeros. Many of the preprocessing steps have been applied for generating the task to fed FCN. To generate the distribution of tasks of the same domain, we hide some of the channels with zeros padding in the dataset and also shuffle the order of the dataset across the channels. We have divided the dataset into the training set and test set such that samples in the train set have classes that are not present in the test set, making a more significant number of task distribution for training the model. The initialization of hyper-parameters, kernel size, and others have been followed the same as given in the paper. *The implementation was done purely in Python and Tensorflow.

3.3 Algorithm

Reptile is the first-order gradient-based meta-learning algorithm. Given the FCN architecture, the adaptation of meta-learning using the Reptile framework can be summarized in algorithm 1. It looks very similar to the ordinary learning tasks, but for each training sample, we create a pseudo-task T_i because the algorithm treats one dataset as one data sample.

Algorithm 1 Reptile with FCN

Require: Training Dataset $\mathcal{D} = \{x^{(j)}, y^{(j)}\}$
Require: α, β step size hyperparameters
Require: Number of task, examples and batch-size
Denote $p(\mathcal{T})$ as distribution over tasks
Randomly initialize θ
for $iteration = 1, 2, \ldots$ **do**
    Sample task $\tau_1, \tau_2, \ldots, \tau_n$, batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
    for $i = 1, 2, \ldots, N$ **do**
        Evaluate $\mathsf{W}_i = \mathsf{SGD}(\mathcal{L}_{\tau_i}, \theta, k)$
        Compute updated parameter with gradient descent using (1)
    end for
    Update $\theta \leftarrow \theta + \beta \frac{1}{n} \sum_{i=1}^{n} (\mathsf{W}_i - \theta)$
end for

4 Experimental Study

In this section, we present the information about the datasets we have used, evaluation strategy to compare between meta and non-meta-learning algorithms, and the experimental result analysis and training setups. We have built the architecture by stacking two convolution blocks with the filter sizes $128, 128$ in each block. After each convolutional layer we have applied batch-normalization and ReLu activation layer to improve the generalization capability. Reptile uses 50 examples per task, so we followed the same sample size for the task. We have used Adam optimizer with learning rate 0.01 to optimize the loss.

4.1 Dataset

The performance of our model has been evaluated on the UCR time-series repository, containing more than 40 distinct time-series datasets. The dataset in the repository has been broken into the different domains (Image Outline, Sensor Readings, Motion Capture, Spectrographs, ECG, Electric Devices, and Simulated Data), having different characteristics and varying length. We have used four datasets from the repository and one other dataset (human activity recognition), which we have collected in our lab. All the chosen dataset is multivariate time series, the preprocessing on the dataset as described in Section 3. The details about the dataset used in our experiment are given in Table 1.

Table 1: Time Series Dataset

Name	Dataset				
	Train Size	Test Size	No. of attributes	Length	No. of classes
PenDigits	7494	3498	2	8	10
UWaveGestureLibrary	896	3582	3	315	8
CharacterTrajectories	1422	1432	3	182	20
Motion	390	390	3	300	12
PeekDB	1000	1000	20	60	5

4.2 Results

To evaluated our model, we first trained the meta-learning model on tasks generated from the training dataset. After we got the optimal parameters from training, we trained two models on the task generated from the test dataset. The first model with parameters initialization which we get from meta learning one and other model with randomly initialized parameters. Then we compare the loss function curve of these models while training. The tasks we generate is used for binary classification

for each dataset, so that we could generate variety of the tasks from same domain. Here, we calculated the total cross entropy loss to measures the performance of a classification model on test task. Cross-entropy loss decreases as the predicted probability converges to the actual label.

Figure 3. shows the experimental results of our model on the dataset. The graph represents the training loss function curve on the test tasks with the number of epoch while training the model. We observe that the meta-learning model converges more quickly with the fewer number of epoch than the model, which are trained from scratch (non-meta- learning) on every dataset. We believe that the success of metal learning models is because of the optimal initialization of parameters that we obtained from the Reptile algorithm. This experiment confirms that there is a significant difference between the models which are trained with optimal parameter initialization then random initialization. From the experiment, we observe that the meta-learning algorithm performs better with the dataset having more number of attributes (PeekDB) and a larger length of time-series (UWaveGestureLibrary, CharacterTrajectories, Motion). So, using the meta-learning algorithm with time-series tasks can be very useful.

5 Conclusion

In this work, we incorporate a meta-learning algorithm with a fully convolutional network for time-series classification tasks. With this, we establish what different type of learnings are, the emergence of meta-learning, coupling of meta-learning with deep learning, and the state of the art models in meta-learning (MAML, and Reptile). Employing this architecture, we find the optimal parameter initialization for the task. The proposed method is evaluated using a loss function curve, which shows its superiority over the non-meta-learning counterpart. The test suggests the using meta-learning algorithms with time series could be quite effective. Meta-learning helps to learn any new task quickly based on prior experience gained from other similar tasks. It is better to use meta-learning in real-world tasks, as there are plenty of opportunities to learn from prior experience.

6 Acknowledgement

We want to thank Rafael Radrumond, Lucas Brinkmeyer, for helpful discussions and suggestions during the early phases of the research. We would also like to thank the anonymous reviewers for sharing their helpful suggestions and Chaitanya Bhatia and Karan Siwach, for their valuable feedback.

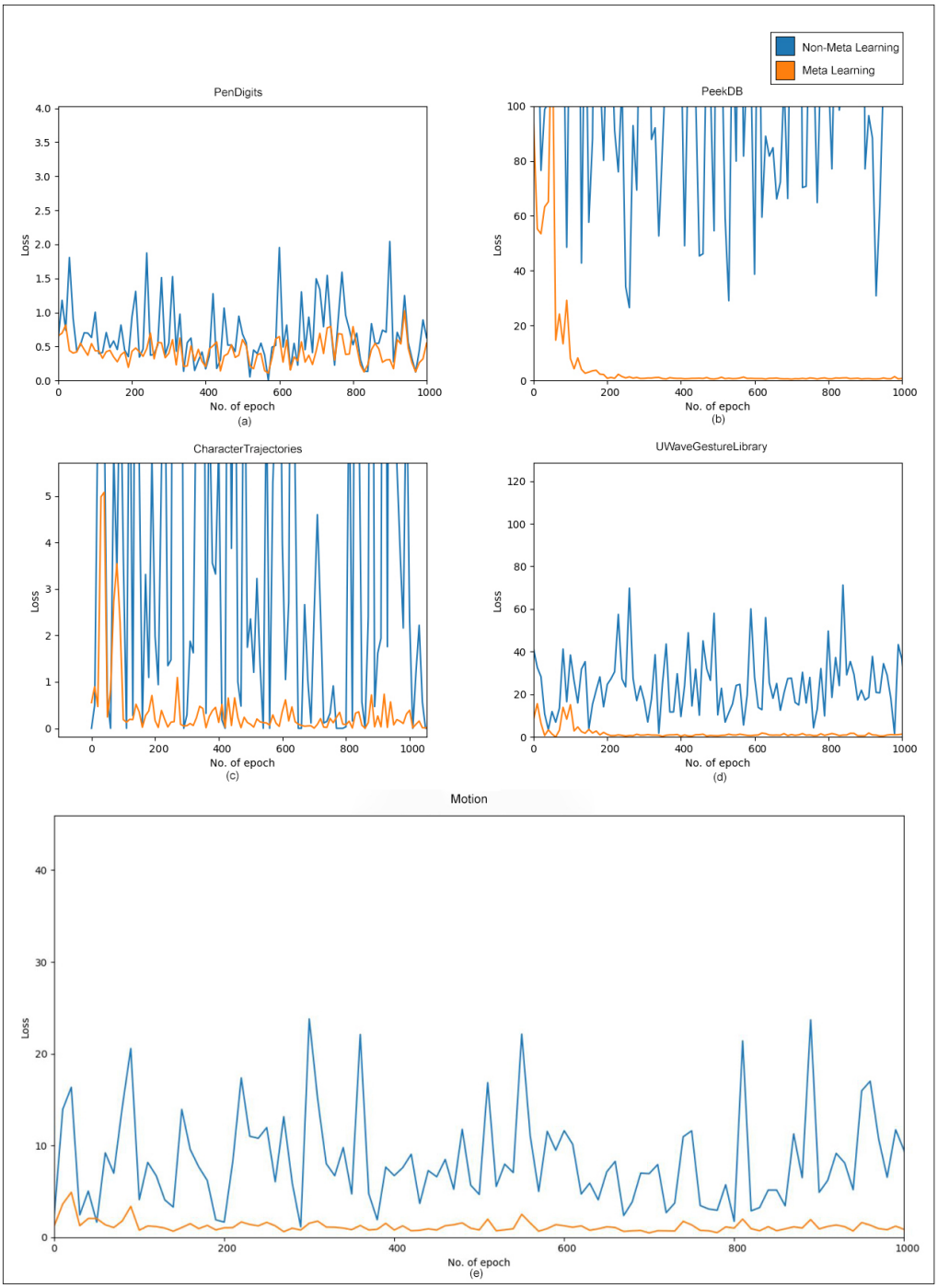

Fig. 3: Demonstrate the effectiveness of Meta-Learning over the Non-Meta-learning model with the help of a loss function curve on a time series dataset.

References

1. Gregory Koch and Richard Zemel and Ruslan Salakhutdinov, "Siamese neural networks for one-shot image recognition," ICML Deep Learning Workshop, 2015.
2. Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, Daan Wierstra,"Matching networks for one shot learning," NIPS. 2016.
3. Chelsea Finn, Pieter Abbeel, and Sergey Levine,"Model-agnostic meta-learning for fast adaptation of deep networks," ICML 2017.
4. Alex Nichol, Joshua Achiam, John Schulman, "On First-Order Meta-Learning Algorithms," arXiv preprint arXiv:1803.02999 (2018).
5. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015b) The UCR time series classification archive. www.cs.ucr.edu/ẽamonn/time_series_data/.
6. Zhiguang Wang, Weizhong Yan, Tim Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," arXiv:1611.06455v4 [cs.LG], 2016.
7. Flood Sung, et al, "Learning to compare: Relation network for few-shot learning," CVPR. 2018.
8. Krizhevsky A, Sutskever I, Hinton GE, "ImageNet classification with deep convolutional neural networks," In: Advances in neural information processing systems vol 25, pp 1097–1105, 2012.
9. Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, Xiaodong He,"Natural Language to Structured Query Generation via Meta-Learning," arXiv:1803.02400v4 [cs.CL], 2018.
10. Weng, Lilian, "Meta-Learning: Learning to Learn Fast," lilianweng.github.io/lil-log, 2018.
11. Ismail Fawaz, H., Forestier, G., Weber, J. et al, "Deep learning for time series classification: a review," Data Min Knowl Disc 33, 917–963(2019). https://doi.org/10.1007/s10618-019-00619-1.
12. Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra and Timothy P. Lillicrap, "One-shot Learning with Memory-Augmented Neural Networks," CoRR,abs/1605.06065,2016.http://arxiv.org/abs/1605.06065.
13. Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoff-man, David Pfau, Tom Schaul, and Nando de Freitas," Learning to learn by gradient descent by gradient descent," CoRR, abs/1606.04474,2016.http://arxiv.org/abs/1606.04474.
14. Lines J, Bagnall A, "Time series classification with ensembles of elastic distance measures,"2015. Data Min Knowl Discov 29:565–592.
15. J. Lin, E. Keogh, L. Wei, and S. Lonardi, " "Experiencing sax: a novel symbolic representation of time series," Data Mining and knowledge discovery, vol. 15, no. 2, pp. 107–144, 2007.
16. Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel, "Meta-Learning with Temporal Convolutions," CoRR,abs/1707.03141, 2017.http://arxiv.org/abs/1707.03141.
17. Sung, Flood and Yang, Yongxin and Zhang, Li and Xiang, Tao and Torr, Philip H.S. and Hospedales, Timothy M.," Learning to Compare: Relation Network for Few-Shot Learning," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June, 2018.
18. Sutton, Richard S and Barto, Andrew G, "Reinforcement learning: An introduction," MIT press; 2018 Oct 19.
19. M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," IEEE transactions on pattern analysis and machine intelligence, vol.35, no.11, pp.2796–2802, 2013.
20. P. Schafer, "The boss is concerned with time series classification in the presence of noise," Data Mining and Knowledge Discovery, vol.29, no.6, pp.1505–1530, 2015.
21. A. Bagnall, J. Lines, J. Hills, and A. Bostrom,"Time-series classification with cote: the collective of transformation-based ensembles," IEEE Transactions on Knowledge and Data Engineering, vol.27, no.9, pp.2522–2535, 2015.

Authors

Aman Gupta and **Yadul Raghav**, we received a Bachelor's and Master's degree from the Indian Institute of Technology (BHU), Varanasi, majoring in Computer Science and Engineering. Our interest lies in the field of machine learning, deep learning, and data mining.

Aman Gupta

Yadul Raghav