One-Shot Learning on Attributed Sequences

Zhongfang Zhuang, Xiangnan Kong, Elke, Rundensteiner
Worcester Polytechnic Institute
{zzhuang, xkong, rundenst}@wpi.edu

Jihane Zouaoui, Aditya Arora
Amadeus IT Group
{jihane.zouaoui, aditya.arora}@amadeus.com

***Abstract*—One-shot learning has become an important research topic in the last decade with many real-world applications. The goal of one-shot learning is to classify *unlabeled* instances when there is only one labeled example per class. Conventional problem setting of one-shot learning mainly focuses on the data that is already in a feature space (such as images). However, the data instances in real-world applications are often more complex and feature vectors may not be available. In this paper, we study the problem of one-shot learning on attributed sequences, where each instance is composed of a set of attributes (*e.g.*, user profile) and a sequence of categorical items (*e.g.*, clickstream). This problem is important for a variety of real-world applications ranging from fraud prevention to network intrusion detection. This problem is more challenging than the conventional one-shot learning since there are dependencies between attributes and sequences. We design a deep learning framework OLAS to tackle this problem. The proposed OLAS utilizes a twin network to generalize the features from pairwise attributed sequence examples. Empirical results on real-world datasets demonstrate the proposed OLAS can outperform the state-of-the-art methods under a rich variety of parameter settings.**

***Index Terms*—One-shot learning, Attributed Sequence**

I. Introduction

Humans are capable of learning from one, or just a few examples [1], and grasp the patterns. We recognize a person even if we have seen this person's photo only once [2]. Inspired by this capability, one-shot learning, where the goal is to classify previously unseen instances based on **only one example per class**, has become an important research topic [3], [4], [5].

In the literature, conventional approaches to one-shot learning focus on using feature vectors as input in the learning process [2], [6], [7], in which each instance is represented as a fixed-size vector (*e.g.*, images). However, data instances in real-world big data applications are often more complex and heterogeneously structured. In this work, we target at one complex data composed of a variable length sequence of categorical items (*e.g.*, a user's clickstream) along with a set of attributes (*e.g.*, a user's profile). We refer to this complex data as *attributed sequences*. Here are two examples of attributed sequences:

Example 1 (Network Traffic as Attributed Sequences): Network traffic can be modeled as attributed sequences. Namely, it consists of a sequence of packages being sent or received by the routers and a set of attributes indicating the context of the network traffic (*e.g.*, user privileges, security settings, *etc*).

Example 2 (Genes as Attributed Sequences): Genes can be represented as attributed sequences, where each gene consists of a DNA *sequence* and a *set* of *attributes* (*e.g.*, PPI, gene ontology, *etc*.) indicating the properties of the gene.

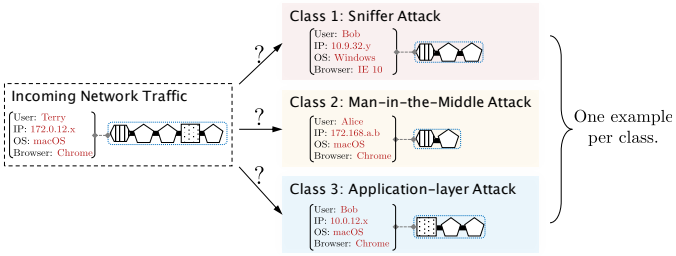


Fig. 1: Network attack detection using one-shot learning on attributed sequences. Each instance is composed of a user profile as the *attributes* and a *sequence* of user actions (depicted using different shapes). A system administrator is interested in finding out if the incoming network traffic is malicious with **only one** sample per class.

Designing one-shot learning to work with attributed sequences promises to be beneficial for a wide range of critical big data applications that require timely responses at scale, such as financial fraud detection and network intrusion detection. In the real-world scenarios, these applications often work on large-scale datasets, yet very few data instances are labeled. Continuing with our Example 1, to respond in a timely fashion to potential network intrusion threats, one first has to determine what the intrusion type of incoming potentially malicious traffic even if only one or a few examples per known intrusion type have been seen previously (as depicted in Fig. 1). Despite its importance in real-world applications, one-shot learning on attributed sequences remains unexplored to date.

In this paper, we study this new problem of one-shot learning on attributed sequences, with the goal of generating a label for each *unlabeled* attributed sequence with only one training example per known class. This problem is different from previous one-shot learning work, as we now need to extract feature vectors from not only the *attributes* but also the structural information from the *sequences* and the *dependencies* between attributes and sequences. We summarize the specific challenges as follows:

- **Attribute-sequence dependencies.** Fundamental problems arise when learning to classify attributed sequence data. Contrary to the simplifying assumption that the attributes and sequences in these real-world scenarios are independent, various dependencies between them can

arise. For example, in network traffic data, one's behavior of *sending/receiving TCP/UDP packets* (*i.e.*, sequences) may depend on the *device type* (*i.e.*, attributes). Since conventional one-shot learning approaches focus on a single data type, these *dependencies* would thus not be captured.

- **Generalization in complex data type.** The key difficulty in one-shot learning is to generalize beyond the single training example. It is more difficult to generalize from a more complex data type [8], such as attributed sequence data, than from a simpler data type due to the larger search space and slower convergence.

Our Approach. To address the above challenges, we propose an end-to-end one-shot learning model, called OLAS, to accomplish one-shot learning for attributed sequences. The OLAS model includes two main components: a CoreNet to encode the information from attributes, sequences and their dependencies and a PredictNet to learn the similarities and differences between different attributed sequence classes. The proposed OLAS model is beyond a simple concatenation of CoreNet and PredictNet. Instead, they are interconnected within one network architecture and thus can be trained synchronously. Once the OLAS is trained, we can then use it to make predictions for not only the new data but also for entire previously unseen new classes. Our paper offers the following core contributions:

- We formulate and analyze the problem of one-shot learning on attributed sequences.
- We develop a deep learning model that is capable of inferring class labels for attributed sequences based on one instance per class.
- We demonstrate that the OLAS network model trained on attributed sequences significantly improves the accuracy of label prediction compared to state-of-the-art methods.

We organize the rest as follows. We first define our problem in Section II. We detail our study of this problem and solve it using a distance metric learning-based solution in Sections III. Next, we present the experimental methodology and results in Section IV. We analyze related work in Section V. We conclude our findings in Section VI.

II. Problem Formulation

In this section, we introduce the key definitions and problem formulation of one-shot learning on attributed sequences. The important notations are summarized in Table I.

A. Preliminaries

Definition 1 (Sequence): Given a finite set $\mathcal{I}$ composed of r categorical items, a sequence $s_i = \left(x_i^{(1)}, \cdots, x_i^{(t_i)}\right)$ is an ordered list of t_i items, where $\forall x_i^{(t)} \in \mathcal{I}$.

The subscript i is used to distinguish different instances. One common method for preprocessing variable-length sequences for deep learning is to first *zero-pad* each sequence to the maximum length of the sequences in a dataset, followed by *one-hot encoding* each sequence [9]. We adopt this approach in this work. We denote the maximum length of sequences as $t_{\max}$. Learning models are capable of disregarding the padding so that the padding has no effect in the training of models. We denote the one-hot encoded form of sequence s_i as a matrix $\mathbf{s}_i \in \mathbb{R}^{t_{\max} \times r}$.

Definition 2 (Attributed Sequence): An attributed sequence p_i is a pair composed of an attribute vector $\mathbf{v}_i$ and a one-hot encoded sequence $\mathbf{s}_i$, denoted as $p_i = (\mathbf{v}_i, \mathbf{s}_i)$. A u-dimensional attribute vector $\mathbf{v}_i$ is composed of u attributes in the dataset.

B. Problem Definition

Inspired by the work in [10], we formulate our problem as finding the parameters θ of a predictor Θ that minimizes the loss $\mathcal{L}_{\text{one-shot}}$. Given a training set of g attributed sequences $\mathcal{G} = \{(p_1, c_1), \cdots, (p_g, c_g)\}$, where each attributed sequence p_i has a unique class label c_i, we formulate the objective for one-shot learning for attributed sequences as:

$$\underset{\theta}{\text{minimize}} \sum_{(p_i, c_i) \in \mathcal{G}} \mathcal{L}_{\text{one-shot}} \left(\Theta\left(p_i; \theta\right), c_i\right) \tag{1}$$

That is, we want to minimize the loss calculated using the label predicted using parameter θ and the true label. One-shot learning is known as a hard problem [2] mainly as a result of unavoidable overfitting caused by insufficient data. With a complex data type, such as attributed sequences, the number of parameters that need to be trained is even larger, which further complicates the problem.

III. The OLAS Model

A. Approach

In this work, we adopt an approach from the distance metric learning perspective. Distance metric learning methods are well known for several important applications, such as face recognition, image classification, *etc*. Distance metric learning is capable of disseminating data based on their dissimilarities using pairwise training samples. Recent work [2] has empirically demonstrated the effectiveness of the distance metric learning approach. In addition to the pairwise training samples, there are two key components in distance metric learning: a *similarity* label depicting whether the training pair is similar and a distance function d. The similar and dissimilar pairs can be randomly generated using the class labels [2]. We define *attributed sequence triplets* in Definition 3.

Definition 3 (Attributed Sequence Triplets): An attributed sequence triplet (p_i, p_j, ℓ_{ij}) consists of two attributed sequences p_i, p_j, and a *similarity* label $\ell_{ij} \in \{0, 1\}$. The similarity label indicates whether p_i and p_j belong to the same class ($\ell_{ij} = 0$) or different classes ($\ell_{ij} = 1$). We denote $\mathcal{P} = \{(p_i, p_j, \ell_{ij}) | \ell_{ij} = 0\}$ as the *positive set* and $\mathcal{N} = \{(p_i, p_j, \ell_{ij}) | \ell_{ij} = 1\}$ as the *negative set*.

However, attributed sequences are not naturally represented as feature vectors. Therefore, we define a transformation function $\Omega(p_i; \omega)$ parameterized by ω as a part of the predictor Θ. Ω uses attributed sequences as the inputs and generates the corresponding feature vectors as the outputs. With two

TABLE I: Important Mathematical Notations

Notation	Description
$\mathbb{R}$	The set of real numbers
r	The number of possible items in sequences.
s_i	A sequence of categorical items.
$x_i^{(t)}$	The t-th item in sequence s_i.
t_{max}	The maximum length of sequences in a dataset.
$\mathbf{s}_i$	A one-hot encoded sequence in the form of a matrix $\mathbf{s}_i \in \mathbb{R}^{t_{\text{max}} \times r}$.
$\mathbf{x}_i^{(t)}$	A one-hot encoded item at t-th time step in a sequence.
$\mathbf{v}_i$	An attribute vector.
p_i	An attributed sequence. *i.e.*, $p_i = (\mathbf{v}_i, \mathbf{s}_i)$
$\mathbf{p}_i$	An n-dimensional feature vector of attributed sequence p_i.
Ω	A function transforming each attributed sequence to a feature vector.
d	A distance function. *e.g.*, Mahalanobis distance, Manhattan distance.
γ	An activation function within fully connected neural networks. Possible choices include `ReLU` and `tanh`.
σ	A logistic activation function within LSTM, *i.e.*, $\sigma(z) = \frac{1}{1+e^{-z}}$

attributed sequences p_i and p_j as inputs, the n-dimensional feature vectors of the respective attributed sequences are:

$$\begin{aligned} \mathbf{p}_i &= \Omega(p_i; \omega) \\ \mathbf{p}_j &= \Omega(p_j; \omega) \\ \mathbf{p}_i, \mathbf{p}_j &\in \mathbb{R}^n \end{aligned} \tag{2}$$

The other key component in distance metric learning approaches is a distance function (*e.g.*, Mahalanobis distance [11], Manhattan distance [10]). A distance function is applied to the feature vectors in distance metric learning.

Distance metric learning-based approaches often use the Mahalanobis distance [11], [12], which can be equivalent to the Euclidean distance [11]. Using the two feature vectors of attributed sequences in Equation 2, the Mahalanobis distance can be written as:

$$d_\omega(\mathbf{p}_i, \mathbf{p}_j) = \sqrt{(\mathbf{p}_i - \mathbf{p}_j)^\top \mathbf{\Lambda} (\mathbf{p}_i - \mathbf{p}_j)} \tag{3}$$

where d_ω is a specific form of distance function d denoting the inputs (*i.e.*, $\mathbf{p}_i, \mathbf{p}_j$) are the results of transformations using parameter ω. $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is a symmetric, semi-definite, and positive matrix, and $\mathbf{\Lambda}$ can be decomposed as:

$$\mathbf{\Lambda} = \mathbf{\Gamma}^\top \mathbf{\Gamma}, \tag{4}$$

where $\mathbf{\Gamma} \in \mathbb{R}^{f \times n}, f \leq n$. By [13], Equation 3 is equivalent to:

$$\begin{aligned} d_\omega(\mathbf{p}_i, \mathbf{p}_j) &= \sqrt{(\mathbf{p}_i - \mathbf{p}_j)^\top \mathbf{\Gamma}^\top \mathbf{\Gamma} (\mathbf{p}_i - \mathbf{p}_j)} \\ &= \|\mathbf{\Gamma}\mathbf{p}_i - \mathbf{\Gamma}\mathbf{p}_j\|_2. \end{aligned} \tag{5}$$

Instead of directly minimizing the loss of the predictor function Θ predicting a label of each attributed sequence as in Equation 1, we can now achieve the same training goal by minimizing the loss of predicting whether a pair of attributed sequences belongs to the same class using distance metric learning-based methods. The overall objective can be written as:

$$\underset{\omega}{\text{minimize}} \sum_{(p_i, p_j, \ell_{ij}) \in \mathcal{P} \cup \mathcal{N}} \mathcal{L}\left(d_\omega\left(\mathbf{p}_i, \mathbf{p}_j\right), \ell_{ij}\right) \tag{6}$$

In recent work on distance metric learning applications [10], [11], deep neural networks are serve as the nonlinear transformation function Ω. Deep neural networks can effectively learn the features from input data without requiring domain-specific knowledge [2], and also generalize the knowledge for future predictions and inferences. These advantages make neural networks become an ideal solution for one-shot learning.

B. OLAS *Model Design*

We next describe the design of the two key components of the OLAS model. First, we design a CoreNet for the nonlinear transformation of attributed sequences. Then, a PredictNet is designed to learn from the contrast of attributed sequences with different class labels. The specific parameters of the OLAS used in our experiments are detailed in Section IV.

The two main networks in CoreNet, a fully connected neural network with m layers and a long short-term memory (LSTM) network [14], correspond to the tasks of encoding the information from attributes and sequences in attributed sequences, respectively. By augmenting with another layer of fully connected neural network on top of the concatenation of the above networks, CoreNet is also capable of learning the attribute-sequence dependencies.

Given the input of an attribute vector $\mathbf{v}_k \in \mathbb{R}^u$, we define a fully connected neural network with m layers as:

$$\begin{aligned} \boldsymbol{\alpha}_1 &= \gamma\left(\mathbf{W}_1 \mathbf{v}_i + \mathbf{b}_1\right) \\ \boldsymbol{\alpha}_2 &= \gamma\left(\mathbf{W}_2 \boldsymbol{\alpha}_1 + \mathbf{b}_2\right) \\ &\vdots \\ \boldsymbol{\alpha}_m &= \gamma\left(\mathbf{W}_m \boldsymbol{\alpha}_{m-1} + \mathbf{b}_m\right) \end{aligned} \tag{7}$$

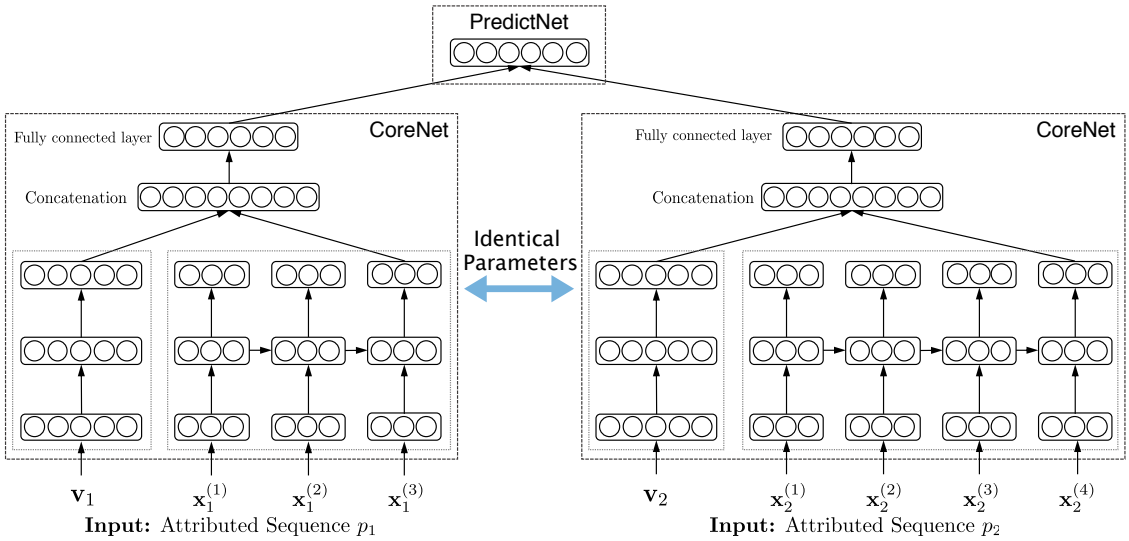


Fig. 2: The network architecture of OLAS. The concatenation only happens after the *last time step* of the sequence so the information of the complete sequence is used.

where γ is a nonlinear transformation function. Although we use hyperbolic tangent $\tanh$ in our model, other nonlinear functions such as rectified linear unit (ReLu) [15] can also be used depending on the empirical results. We denote the weights and bias parameters as:

$$\mathbf{W}_{\mathrm{F}} = [\mathbf{W}_1, \cdots, \mathbf{W}_m]^\top, \mathbf{b}_{\mathrm{F}} = [\mathbf{b}_1, \cdots, \mathbf{b}_m]^\top \tag{8}$$

Note that the choice of m is task-specific. Although neural networks with more layers are better at learning hierarchical structure in the data, it is also observed that such networks are challenging to train due to the multiple nonlinear mappings that prevent the information and gradient passing along the computation graph [16].

$\mathbf{W}_{\mathrm{F}}$ and $\mathbf{b}_{\mathrm{F}}$ are used to transform the input of each layer to a lower dimension. This transformation is imperative given the often large number of dimensions of attribute vectors in real-world applications. Different from attribute vectors, the categorical items in the sequences in attributed sequences obey temporal ordering. The information of sequences is not only in the item values, but more importantly, in the temporal ordering of these items. In this vein, the CoreNet also utilizes an LSTM network. LSTM is capable of handling not only the ordering of items, but also the dependencies between different items in the sequences. Given a sequence $\mathbf{s}_i$ as the input, we use an LSTM [14] to process each item $\mathbf{x}_k^{(t)}$ in this sequence as:

$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma\left(\mathbf{W}_{\mathrm{i}}\mathbf{x}_k^{(t)} + \mathbf{U}_{\mathrm{i}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{i}}\right) \\
\mathbf{f}^{(t)} &= \sigma\left(\mathbf{W}_{\mathrm{f}}\mathbf{x}_k^{(t)} + \mathbf{U}_{\mathrm{f}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{f}}\right) \\
\mathbf{o}^{(t)} &= \sigma\left(\mathbf{W}_{\mathrm{o}}\mathbf{x}_k^{(t)} + \mathbf{U}_{\mathrm{o}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{o}}\right) \\
\mathbf{g}^{(t)} &= \tanh\left(\mathbf{W}_{\mathrm{c}}\mathbf{x}_k^{(t)} + \mathbf{U}_{\mathrm{c}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{c}}\right) \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \mathbf{g}^{(t)} \\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \tanh\left(\mathbf{c}^{(t)}\right)
\end{aligned} \tag{9}$$

where σ is a sigmoid activation function, $\odot$ denotes the bitwise multiplication, $\mathbf{i}^{(t)}$, $\mathbf{f}^{(t)}$ and $\mathbf{o}^{(t)}$ are the internal gates of the LSTM, $\mathbf{c}^{(t)}$ and $\mathbf{h}^{(t)}$ are the cell and hidden states of the LSTM. Without loss of generality, we denote LSTM kernel parameters $\mathbf{W}_{\mathrm{L}}$, recurrent parameters $\mathbf{U}_{\mathrm{L}}$ and bias parameters $\mathbf{b}_{\mathrm{L}}$ as:

$$\begin{aligned}
\mathbf{W}_{\mathrm{L}} &= [\mathbf{W}_{\mathrm{i}}, \mathbf{W}_{\mathrm{f}}, \mathbf{W}_{\mathrm{o}}, \mathbf{W}_{\mathrm{c}}]^\top \\
\mathbf{U}_{\mathrm{L}} &= [\mathbf{U}_{\mathrm{i}}, \mathbf{U}_{\mathrm{f}}, \mathbf{U}_{\mathrm{o}}, \mathbf{U}_{\mathrm{c}}]^\top \\
\mathbf{b}_{\mathrm{b}} &= [\mathbf{b}_{\mathrm{i}}, \mathbf{b}_{\mathrm{f}}, \mathbf{b}_{\mathrm{o}}, \mathbf{b}_{\mathrm{c}}]^\top
\end{aligned} \tag{10}$$

The attribute vectors and sequences are processed simultaneously and the outputs of both networks are concatenated together. Instead of using the outputs of the LSTM at every time step, we only concatenate the last output from the LSTM to the output of the fully connected neural network so that the complete sequence information is used. After that, another layer of fully connected neural network is used to capture the dependencies between attributes and sequences. Given the output dimensions of $\boldsymbol{\alpha}_m$ and $\mathbf{h}^{(t)}$ as n_m and n_l, respectively, the concatenation and the last fully connected layer of CoreNet can be written as:

$$\mathbf{p}_i = \gamma\left(\mathbf{W}_{\mathrm{p}}\left(\boldsymbol{\alpha}_m \oplus \mathbf{h}^{(t_i)}\right) + \mathbf{b}_{\mathrm{p}}\right) \tag{11}$$

where $\oplus$ represents the concatenation of two vectors, $\mathbf{W}_{\mathrm{p}} \in \mathbb{R}^{n \times (n_m + n_l)}$ and $\mathbf{b}_{\mathrm{p}} \in \mathbb{R}^n$ denote the weight matrix and bias vector in this fully connected layer for an n-dimensional output. In summary, the CoreNet can be written as:

$$\Omega : \left(\mathbb{R}^u, \mathbb{R}^{t_{\max} \times r}\right) \mapsto \mathbb{R}^n \tag{12}$$

The two outputs of CoreNet ($\mathbf{p}_i$ and $\mathbf{p}_j$) are first generated. Then, $\mathbf{p}_i$, $\mathbf{p}_j$ and the similarity label ℓ_{ij}, are used by the PredictNet to learn the similarities and differences between them. The PredictNet is designed to utilize a contrastive loss function [17] so that attributed sequences in different categories are disseminated. The contrastive loss function is

Algorithm 1 Training using attributed sequence triplets

INPUT: A positive set $\mathcal{P}$ and a negative set $\mathcal{N}$ of attributed sequence triplets, the number of layers in fully connected neural networks m, learning rate λ, number of iterations ϕ and convergence error ϵ.

OUTPUT: Parameters of OLAS ($\{\mathbf{W}_{\mathrm{F}}, \mathbf{b}_{\mathrm{F}}, \mathbf{W}_{\mathrm{L}}, \mathbf{U}_{\mathrm{L}}, \mathbf{b}_{\mathrm{L}}\}$).

```
Initialize OLAS network.
for each φ' = 1, ..., φ do   ▷ φ is the maximum number of training epochs.
    for each (p_i, p_j, ℓ_ij) ∈ P ∪ N do
        p_i ← Ω(p_i; ω).
        p_j ← Ω(p_j; ω).
        Compute d_ω.                                   ▷ Equation 5.
        Compute the loss L_φ'(p_i, p_j, ℓ_ij).         ▷ Equation 13.
        if |L_φ'(p_i, p_j, ℓ_ij) − L_φ'−1(p_i, p_j, ℓ_ij)| < ε then
            break   ▷ Early stopping to avoid overfitting.
        else
            Compute ∂L/∂d_ω, ∂d_ω/∂Ω.                  ▷ Equation 16, 17.
            Compute ∇L.                                ▷ Equation 14.
            Update network parameters.                 ▷ Equation 18.
        end if
    end for
end for
```

composed of two parts: a partial loss for the dissimilar pairs and a partial loss for similar pairs. The specific form of contrastive loss of PredictNet can be written as:

$$\mathcal{L}(\mathbf{p}_i, \mathbf{p}_j, \ell_{ij}) = \underbrace{\frac{1}{2}\ell_{ij}\Big[\max\left(0, \xi - d_\omega(\mathbf{p}_i, \mathbf{p}_j)\right)\Big]^2}_{\text{Partial loss for } \textit{dissimilar} \text{ pairs.}} + \underbrace{\frac{1}{2}(1-\ell_{ij})d_\omega^2(\mathbf{p}_i, \mathbf{p}_j)}_{\text{Partial loss for } \textit{similar} \text{ pairs}} \tag{13}$$

where ξ is a margin parameter used to prevent the dataset being reduced to a single point [13]. That is, the attributed sequences with $\ell_{ij} = 1$ are only used to adjust the parameters in the transformation function Ω if the distance between them is larger than ξ. The architecture of OLAS is illustrated in Fig. 2.

C. OLAS Model Training

With the contrastive loss $\mathcal{L}$ computed using Equation 13, we can now calculate the gradient $\nabla\mathcal{L}$, which is used to adjust parameters in the network as:

$$\nabla\mathcal{L} \equiv \left[\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{\mathrm{F}}}, \frac{\partial\mathcal{L}}{\partial\mathbf{b}_{\mathrm{F}}}, \frac{\partial\mathcal{L}}{\partial\mathbf{W}_{\mathrm{L}}}, \frac{\partial\mathcal{L}}{\partial\mathbf{U}_{\mathrm{L}}}, \frac{\partial\mathcal{L}}{\partial\mathbf{b}_{\mathrm{L}}}\right] \tag{14}$$

With the transformation function Ω and distance function d, the explicit form of $\nabla\mathcal{L}$ can be written as:

$$\nabla\mathcal{L} = \frac{\partial\mathcal{L}}{\partial d_\omega}\frac{\partial d_\omega}{\partial\Omega}\left[\frac{\partial\boldsymbol{\alpha}_m}{\partial\mathbf{W}_{\mathrm{F}}}, \frac{\partial\boldsymbol{\alpha}_m}{\partial\mathbf{b}_{\mathrm{F}}}, \frac{\partial\mathbf{h}^{(t_i)}}{\partial\mathbf{W}_{\mathrm{L}}}, \frac{\partial\mathbf{h}^{(t_i)}}{\partial\mathbf{U}_{\mathrm{L}}}, \frac{\partial\mathbf{h}^{(t_i)}}{\partial\mathbf{b}_{\mathrm{L}}}\right] \tag{15}$$

where

$$\frac{\partial\mathcal{L}}{\partial d_\omega} = -\ell_{ij}\max(0, \xi - d_\omega(\mathbf{p}_i, \mathbf{p}_j)) + (1-\ell_{ij})d_\omega(\mathbf{p}_i, \mathbf{p}_j) \tag{16}$$

$$\frac{\partial d_\omega}{\partial\Omega} = (\mathbf{p}_i - \mathbf{p}_j)\cdot(\mathbb{1} - (\mathbf{p}_i - \mathbf{p}_j)) \tag{17}$$

where $\mathbb{1}$ is a vector filled with ones.

We present the derivation of OLAS update functions in our network in Appendix A. With the learning rate λ, the parameters $\mathbf{W}_{\mathrm{F}}, \mathbf{W}_{\mathrm{L}}, \mathbf{U}_{\mathrm{L}}, \mathbf{b}_{\mathrm{F}}$ and $\mathbf{b}_{\mathrm{L}}$ can be updated by the following equation until convergence is achieved:

$$\begin{aligned}
\mathbf{W}_{\mathrm{F}} &= \mathbf{W}_{\mathrm{F}} - \lambda\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{\mathrm{F}}} \\
\mathbf{b}_{\mathrm{F}} &= \mathbf{b}_{\mathrm{F}} - \lambda\frac{\partial\mathcal{L}}{\partial\mathbf{b}_{\mathrm{F}}} \\
\mathbf{W}_{\mathrm{L}} &= \mathbf{W}_{\mathrm{L}} - \lambda\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{\mathrm{L}}} \\
\mathbf{U}_{\mathrm{L}} &= \mathbf{U}_{\mathrm{L}} - \lambda\frac{\partial\mathcal{L}}{\partial\mathbf{U}_{\mathrm{L}}} \\
\mathbf{b}_{\mathrm{L}} &= \mathbf{b}_{\mathrm{L}} - \lambda\frac{\partial\mathcal{L}}{\partial\mathbf{b}_{\mathrm{L}}}
\end{aligned} \tag{18}$$

We summarize the algorithms for updating the OLAS network in Algorithm 1.

D. Labeling Attributed Sequences

Once we have trained the OLAS network to recognize the similarities and dissimilarities between exemplars of attributed sequence pairs. The OLAS is then ready to be used to assign labels to unlabeled attributed sequences in one-shot learning. Given a test attributed sequence p_k from a set $\mathcal{K}$ of unlabeled instances, a set $\mathcal{G} = \{p_g\}_{g=1}^{G}$ of attributed sequences with G categories, in which there is only one instance per category, and the goal is to classify p_k into one of G categories. We can now use the OLAS network with only one forward pass to calculate the distance between p_k with each of the G attributed sequences and the label of the instance that is closest to p_k is then assigned as the label of p_k. This process can be defined using maximum similarity as:

$$\widehat{c_k} = \underset{g}{\operatorname{argmin}}\, d_\omega(\mathbf{p}_k, \mathbf{p}_g) \tag{19}$$

where $\widehat{c_k}$ is the predicted label of $\mathbf{p}_k$.

IV. Experiments

A. Datasets

Our solution has been motivated in part by use case scenarios observed at Amadeus related to attributed sequences. For this reason, we now work with the log files of an Amadeus [18] internal application. Also, we apply our methodology to real-world, public available Wikispeedia data [19]. We summarize the data descriptions as follows:

- **Amadeus data (AMS1∼AMS6)**. We sampled six datasets from the log files of an internal application at

TABLE II: Number of Classes in Datasets

Dataset	Training	One-shot Learning
AMS1, WS1	6	4
AMS2, WS2	12	8
AMS3, WS3	18	12
AMS4, WS4	24	16
AMS5, WS5	30	20
AMS6, WS6	36	24

TABLE III: Compared Methods

Name	Data Used	Note
OLAS	Attributed Sequences	This Work
OLASEmb	Attributed Sequence Embeddings	This work + [22]
ATT	Attributes Only	[2]
SEQ	Sequence Only	[23] + [2]

Amadeus IT Group. Each attributed sequence is composed of a user profile containing information (*e.g.*, system configuration, office name) and a sequence of function names invoked by web click activities (*e.g.*, login, search) ordered by time.

- **Wikispeedia data (WS1∼WS6)**. Wikispeedia is an online game requiring participants to click through from a given start page to an end page using fewest clicks [19]. We select the *finished* path and extract several properties of each path (*e.g.*, the category of the start path, time spent per click). We also sample six datasets from Wikispeedia. The Wikispeedia data is available through the Stanford Network Analysis Project[1] [20].

Following the protocols in recent work [2], we utilize the attributed sequences associated with 60% of categories to generate attributed sequence triplets and use them in training.

The class labels used in training and one-shot learning are disjoint sets. Similar to the strategy in [21], where the authors designed a 20-way classification task that attempts to match an alphabet with one of the twenty possible classes, we randomly select one instance in the one-shot learning set and attempt to give it a correct label. We selected 2000 instances for each set used in one-shot learning and compute the accuracy. We summarize the number of classes in Table II.

B. Compared Methods

We focus on one-shot learning methods on different data types. We summarize the compared methods in Table III. Specifically, we compare the performance of the following one-shot learning methods:

- OLAS: We first evaluate our proposed method using attributed sequences data.
- OLASEmb: Instead of using the attributed sequence instances as input, we use the embeddings of attributed sequences as the input. We want to find out whether a simpler heuristic combination of state-of-the-art would achieve better performance.

[1]https://snap.stanford.edu/data/wikispeedia.html

Algorithm 2 One-shot learning for attributed sequences.

INPUT: Trained networks CoreNet Ω and PredictNet, a set of unlabeled attributed sequences $\mathcal{K}$, a set of labeled attributed sequence with one example per class $\mathcal{G}$ and a distance function d.
OUTPUT: A set of labeleled attributed sequences $\mathcal{K}'$.

$\mathcal{K}' \leftarrow \emptyset$
for each $p_k \in \mathcal{K}$ **do**
  $\varepsilon \leftarrow +\infty$ ▷ Set initial minimum distance to $+\infty$
  $\mathbf{p}_k \leftarrow \Omega(p_k; \omega)$
  for each $(p_g, c_g) \in \mathcal{G}$ **do**
    $\mathbf{p}_g \leftarrow \Omega(p_g; \omega)$
    if $d(\mathbf{p}_k, \mathbf{p}_g) \leq \varepsilon$ **then**
      $\varepsilon \leftarrow d_\omega(\mathbf{p}_k, \mathbf{p}_g)$ ▷ Using PredictNet.
      $\widehat{c_k} \leftarrow c_g$ ▷ Assign the same label of p_g to p_k.
    end if
    $\mathcal{K}' \leftarrow (p_k, \widehat{c_k})$
  end for
end for
return $\mathcal{K}$

- ATT: This is the state-of-the-art method [2] using only attributes of the data.
- SEQ: We combine the state-of-the-art in one-shot learning [2] with sequence-to-sequence learning [23] to be able to utilize sequences in one-shot learning.

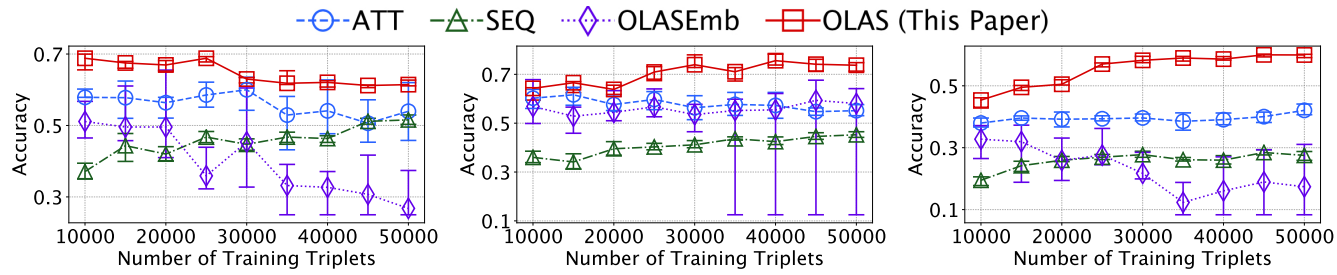
C. Experiment Settings

1) Protocols: The goal of one-shot learning is to correctly assign class labels to each instance. In order to compare with state-of-the-art work [2], [10], we also use accuracy to evaluate the performance. A higher accuracy score means a method could make more correct class label predictions. For each experiment setting, we repeat ten times and report the median, 25 percentile and 75 percentile of the results using error bars. For each training process using attributed sequence triplets, we hold out 20% of the training data as the validation set. The holdout portion is not limited to the instances with certain labels, but instead, they are randomly chosen from all possible classes.

2) Network Initialization and Settings: Gradient-based methods often require a careful initialization of the neural networks. In our experiments, we use normalized random distribution [24] to initialize weight matrices $\mathbf{W}_\mathrm{F}$ and $\mathbf{W}_\mathrm{L}$, orthogonal matrix is used to initialize recurrent matrices $\mathbf{U}_\mathrm{L}$ and biases are initialized to zero vector $\mathbf{0}$. Specifically, the m-th layer of the fully connected neural network is initialized as:

$$\mathbf{W}_m \sim \mathrm{Uniform}\left[-\frac{\sqrt{6}}{\sqrt{n_{m-1}+n_{m+1}}}, \frac{\sqrt{6}}{\sqrt{n_{m-1}+n_{m+1}}}\right]$$

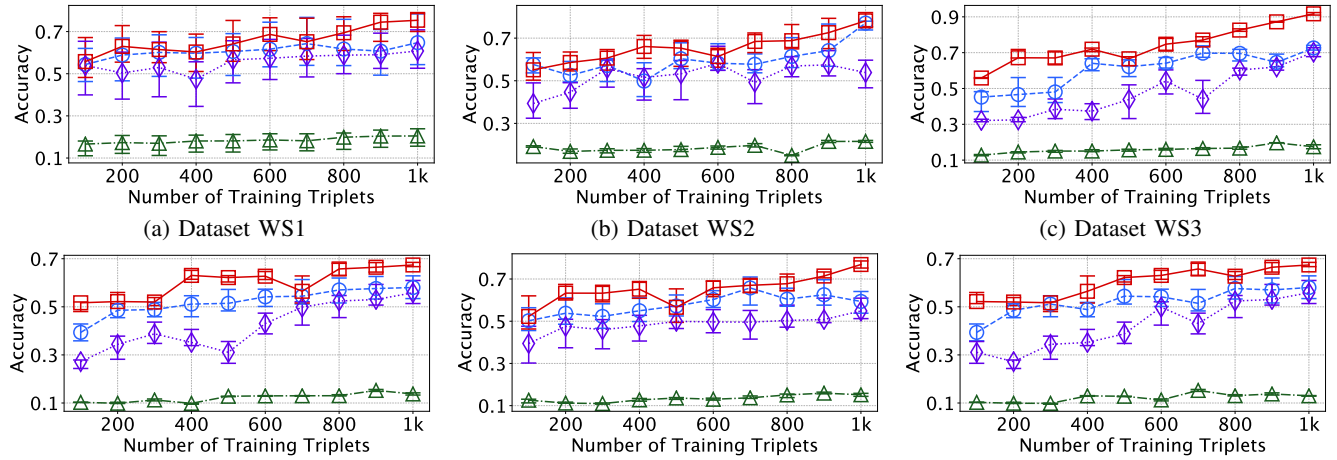
where n_m is the output dimension of the m-th layer. There are three layers used in our experiments. Meanwhile, the weight



(a) Dataset AMS1 (b) Dataset AMS2 (c) Dataset AMS3

(d) Dataset AMS4 (e) Dataset AMS5 (f) Dataset AMS6

Fig. 3: Accuracy of the label prediction on AMS datasets using **Euclidean** distance function.



(a) Dataset WS1 (b) Dataset WS2 (c) Dataset WS3

(d) Dataset WS4 (e) Dataset WS5 (f) Dataset WS6

Fig. 4: Accuracy of the label prediction on Wikispeedia datasets using **Euclidean** distance function.

matrices $\mathbf{W}_\mathrm{i}, \mathbf{W}_\mathrm{f}, \mathbf{W}_\mathrm{o}, \mathbf{W}_\mathrm{c}$ are initialized as:

$$\mathbf{W}_\mathrm{i}, \mathbf{W}_\mathrm{f}, \mathbf{W}_\mathrm{o}, \mathbf{W}_\mathrm{c} \sim \text{Uniform}\left[-\sqrt{\frac{6}{n_l}}, \sqrt{\frac{6}{n_l}}\right]$$

where n_l is the dimension of the output. In our experiments, we use 50 dimensions for both n_l and n_m. We utilize ℓ_2-regularization with early stopping to avoid overfitting. The validation set is composed of 20% of the total amount of attributed sequence triplets in the training set.

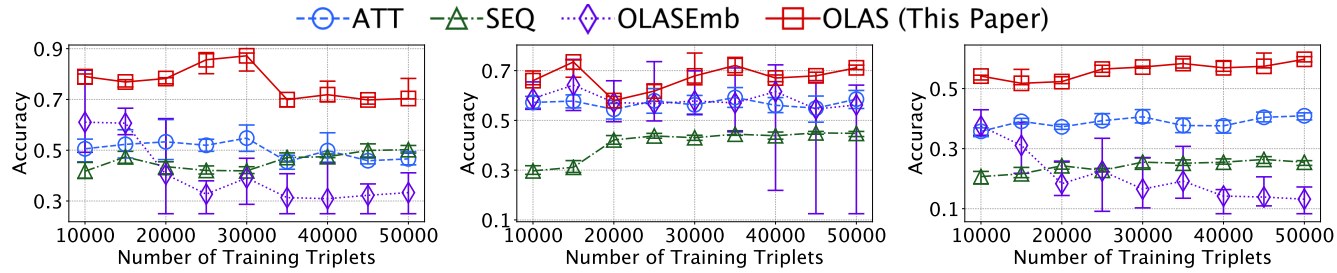
D. Performance Studies

In this section, we present the performance studies of the proposed OLAS network and compare it with techniques in the state-of-the-art.

1) Varying number of training triplets: Fig. 3 and 4 present the results where each setting has a fixed number of labels while the number of training triplets increases. Based on the experiment result figures, we have the following observations:

- As more triplets being used in the training process, the accuracy of one-shot learning keeps increasing with the trained OLAS network. Intuitively, with more examples demonstrated to the OLAS, it could better gain a better capability of generalization, even though the data instances used in one-shot learning are previously unseen.
- Overfitting challenges the performance of all one-shot learning approaches. Although we use early stopping and ℓ_2-regularization in all experiments, overfitting can still be challenging due to there is only one example per class in the one-shot learning.
- OLAS can achieve better performance than other baseline methods when there are more possible classes. While OLAS maintains a stable performance outperforming state-of-the-art under various parameter settings, OLAS can achieve a better performance when the classification task become *harder* with more possible class labels.

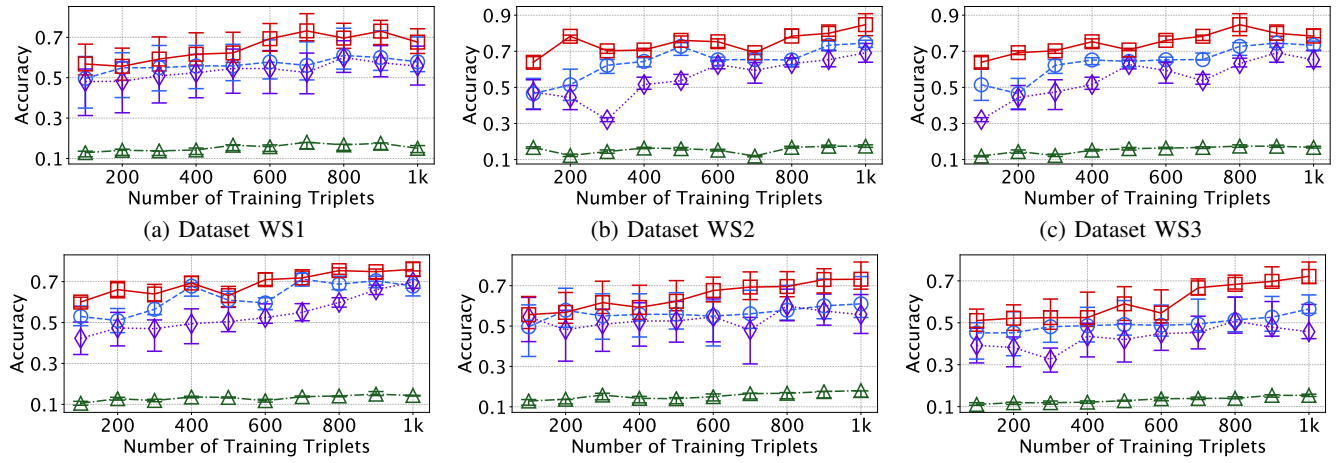
2) Observations using different datasets: Different from the synthetic datasets, the real-world applications often consist of diverse and noisy data instances. It is also interesting to examine the results using different real-world datasets. We find



(a) Dataset AMS1 (b) Dataset AMS2 (c) Dataset AMS3

(d) Dataset AMS4 (e) Dataset AMS5 (f) Dataset AMS6

Fig. 5: Accuracy of the label prediction on AMS datasets using **Manhattan** distance function.



(a) Dataset WS1 (b) Dataset WS2 (c) Dataset WS3

(d) Dataset WS4 (e) Dataset WS5 (f) Dataset WS6

Fig. 6: Accuracy of the label prediction on Wikispeedia datasets using **Manhattan** distance function.

that the performance of OLAS remains superior when we use the twelve datasets sampled from two real-world applications.

3) Advantage of the end-to-end model: Although it is possible to use attributed sequence embeddings [22] with one-shot learning, the experiment results have proven that the performance of the end-to-end solution in this work is far superior to and more stable than OLASEmb. Specifically, the performance of our closest baseline method OLASEmb has varied more compared to all other methods. Building an end-to-end model allows the back-propagation of gradient throughout all layers in the OLAS model. On the other hand, the two gradients in OLASEmb, *i.e.*, the gradient in the model for generating attributed sequence embedding and the gradient in one-shot learning model, are independent and thus the parameters within this method cannot be better adjusted than our solution OLAS.

4) Effect of different distance functions. : Recent work [10] has observed significant differences in performance when using different distance functions. Here, we substitute the Euclidean distance function with the Manhattan distance function to see the performance of all compared methods. We observe that the proposed OLAS model is capable of achieving the best results despite which one of the two distance functions is used.

V. Related Work

A. One-shot learning

One-shot learning, where the goal is to classify instances with only one example per class, has been the center of many applications [2], [4], [10], [25], [26], [27], [28], [29], [30], [31]. It has been a useful approach to classification when the number of labeled instances is scarce. While conventional approaches to one-shot learning often involve Bayesian and shared probability densities [3], [5], recent works [2], [10] take advantage of the feature learning capability of neural networks to further one-shot learning. The common objective of these tasks is to train a model so that the distance between instances from different classes is enlarged as much as possible [2],

[10]. Siamese network structure [2], [10] is often used in such models, where two instances are taken as input, and the difference between them is learned. This example-based learning schema is flexible and has been applied to image classification tasks. However, these works focus on one-shot learning with only one type of data. In this paper, we further the state-of-the-art one-shot learning methods to learn from a more complicated data type (*i.e.*, attributed sequence).

B. Distance Metric Learning

Distance metric learning, where the goal is to learn a distance metric from pairs of similar and dissimilar examples, has been extensively studied [13], [32], [33], [34], [35], [36], [11], [37], [38]. These tasks share a common objective of learning a distance metric, which could be used to reduce the distance between similar pairs of instances and increase the distance between dissimilar pairs of instances. Distance metric learning has shown its powerfulness in various tasks [13], [32], [33]. Many applications in various domains require distance metric learning to achieve a good performance, such as identifying patient similarity in health informatics [34], image recognition [2], face verification [35], [36], [11], and sentence semantic similarity analysis [37], [38]. With the recent advancement in deep learning, distance metric learning has expanded to use various deep learning architectures to achieve its goal [2], [37], [11].

C. Deep learning

Deep learning has attracted a significant amount of research interest in recent years due to its capability of extracting features. Deep learning models, with a number of layers, are capable of learning features at various granularities. The capability of effective feature learning has advanced various research topics, including image recognition [12], [39] and sequence learning [40], [23], [41], [38]. It has also been applied in diverse problem domains, such as medical [42] and traffic flow prediction [43]. Many of these applications involve only one type of data [23], [41] while some applications make use of two types of data [12], [39]. However, none of these works has focused on this new data type of attributed sequence nor performing one-shot learning tasks on attributed sequences.

VI. Conclusion

In this paper, we study this new problem of one-shot learning for attributed sequences. We present the OLAS network design to tackle the challenges of utilizing this new data type in one-shot learning. OLAS incorporates two sub-networks, CoreNet and PredictNet, that integrated into one structure together effectively learn the patterns hidden in this data type using only one example per class. OLAS uses this trained knowledge to generate labels for incoming unlabeled instances. Our experiments on real-world datasets demonstrate that OLAS on attributed sequences outperforms state-of-the-art one-shot learning methods.

Appendix
Gradients and Back-propagation in OLAS

For the m-th layer in a fully connected neural network, we employ the following update functions:

$$\begin{aligned}\frac{\partial \boldsymbol{\alpha}_m}{\partial \mathbf{W}_m} &= \boldsymbol{\alpha}_m \left(\mathbb{1} - \boldsymbol{\alpha}_m\right) \boldsymbol{\alpha}_{m-1} \\ \frac{\partial \boldsymbol{\alpha}_m}{\partial \mathbf{b}_m} &= \boldsymbol{\alpha}_m \left(\mathbb{1} - \boldsymbol{\alpha}_{m-1}\right)\end{aligned} \tag{20}$$

Here we use three steps to explain how OLAS back-propagates the gradients. We use a $\delta_{\mu,\nu}$ function to simplify the equations with $\mu = \{\text{i, f, o}\}$ and $\nu = \{\text{i, f, o, c}\}$:

$$\delta_{\mu,\nu} = \begin{cases} 1, & \text{if } \mu = \nu \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

First, we have the following equations for $\mathbf{h}^{(t)}$ and $\mathbf{c}^{(t)}$:

$$\begin{aligned}\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{W}_\nu} &= \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{W}_\nu} \odot \tanh\left(\mathbf{c}^{(t)}\right) + \mathbf{o}^{(t)} \\ &\qquad \odot \left(1 - \tanh^2(\mathbf{c}^{(t)})\right) \frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{W}_\nu} \\ \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{U}_\nu} &= \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{U}_\nu} \odot \tanh(\mathbf{c}^{(t)}) + \mathbf{o}^{(t)} \\ &\qquad \odot \left(1 - \tanh^2(\mathbf{c}^{(t)})\right) \frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{U}_\nu} \\ \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{b}_\nu} &= \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{b}_\nu} \odot \tanh(\mathbf{c}^{(t)}) + \mathbf{o}^{(t)} \\ &\qquad \odot \left(1 - \tanh^2(\mathbf{c}^{(t)})\right) \frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{b}_\nu}\end{aligned} \tag{22}$$

$$\begin{aligned}\frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{W}_\nu} &= \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{W}_\nu} \odot \mathbf{c}^{(t-1)} + \mathbf{f}^{(t)} \odot \frac{\partial \mathbf{c}^{(t-1)}}{\partial \mathbf{W}_\nu} + \\ &\qquad \frac{\partial \mathbf{i}^{(t)}}{\partial \mathbf{W}_\nu} \odot \mathbf{g}^{(t)} + \mathbf{i}^{(t)} \odot \frac{\partial \mathbf{g}^{(t)}}{\partial \mathbf{W}_\nu} \\ \frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{U}_\nu} &= \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{U}_\nu} \odot \mathbf{c}^{(t-1)} + \mathbf{f}^{(t)} \odot \frac{\partial \mathbf{c}^{(t-1)}}{\partial \mathbf{U}_\nu} + \\ &\qquad \frac{\partial \mathbf{i}^{(t)}}{\partial \mathbf{U}_\nu} \odot \mathbf{g}^{(t)} + \mathbf{i}^{(t)} \odot \frac{\partial \mathbf{g}^{(t)}}{\partial \mathbf{U}_\nu} \\ \frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{b}_\nu} &= \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{b}_\nu} \odot \mathbf{c}^{(t-1)} + \mathbf{f}^{(t)} \odot \frac{\partial \mathbf{c}^{(t-1)}}{\partial \mathbf{b}_\nu} + \\ &\qquad \frac{\partial \mathbf{i}^{(t)}}{\partial \mathbf{b}_\nu} \odot \mathbf{g}^{(t)} + \mathbf{i}^{(t)} \odot \frac{\partial \mathbf{g}^{(t)}}{\partial \mathbf{b}_\nu}\end{aligned} \tag{23}$$

Then, we have the following equations for $\mathbf{i}^{(t)}, \mathbf{f}^{(t)}$ and $\mathbf{o}^{(t)}$:

$$\begin{aligned}\frac{\partial \Delta_\mu}{\partial \mathbf{W}_\nu} &= \Delta_\mu(1 - \Delta_\mu)\boldsymbol{\alpha}^{(t)}\delta_{\mu,\nu} \\ \frac{\partial \Delta_\mu}{\partial \mathbf{U}_\nu} &= \Delta_\mu(1 - \Delta_\mu)\mathbf{h}^{(t-1)}\delta_{\mu,\nu} \\ \frac{\partial \Delta_\mu}{\partial \mathbf{b}_\nu} &= \Delta_\mu(1 - \Delta_\mu)\delta_{\mu,\nu}\end{aligned} \tag{24}$$

where $\Delta_{\text{i}} = \mathbf{i}^{(t)}$, $\Delta_{\text{f}} = \mathbf{f}^{(t)}$ and $\Delta_{\text{o}} = \mathbf{o}^{(t)}$.

Finally, we have the gradients for $\mathbf{g}^{(t)}$ as:

$$\begin{aligned}\frac{\partial \mathbf{g}^{(t)}}{\partial \mathbf{W}_\nu} &= (1-(\mathbf{g}^{(t)})^2)\boldsymbol{\alpha}^{(t)}\delta_{c,\nu} \\ \frac{\partial \mathbf{g}^{(t)}}{\partial \mathbf{U}_\nu} &= (1-(\mathbf{g}^{(t)})^2)\mathbf{h}^{(t-1)}\delta_{c,\nu} \\ \frac{\partial \mathbf{g}^{(t)}}{\partial \mathbf{b}_\nu} &= (1-(\mathbf{g}^{(t)})^2)\delta_{c,\nu}\end{aligned} \tag{25}$$

REFERENCES

[1] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the Cognitive Science Society*, vol. 33, 2011.

[2] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*.

[3] L. Fei-Fei *et al.*, "A bayesian approach to unsupervised one-shot learning of object categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2003, pp. 1134–1141.

[4] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[5] E. G. Miller, N. E. Matsakis, and P. A. Viola, "Learning from one example through shared densities on transforms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2000, pp. 464–471.

[6] J. Wang, A. Kalousis, and A. Woznica, "Parametric local metric learning for nearest neighbor classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 1601–1609.

[7] Z. Xing, J. Pei, and E. Keogh, "A brief survey on sequence classification," *ACM Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 40–48, 2010.

[8] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access*, vol. 2, pp. 514–525, 2014.

[9] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[10] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Advances in Neural Information Processing Systems*, 2016, pp. 523–531.

[11] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1875–1882.

[12] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.

[13] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Information Processing Systems*, 2003, pp. 521–528.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010, pp. 807–814.

[16] T. Pham, T. Tran, D. Q. Phung, and S. Venkatesh, "Column networks for collective classification." in *AAAI*, 2017, pp. 2485–2491.

[17] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1735–1742.

[18] Amadeus, "Amadeus IT Group," http://www.amadeus.com, accessed: 2017-09-23.

[19] R. West, J. Pineau, and D. Precup, "Wikispeedia: An online game for inferring semantic distances between concepts," in *International Joint Conference on Artificial Intelligence*, 2009.

[20] J. Leskovec, "Wikispeedia navigation paths," https://snap.stanford.edu/data/wikispeedia.html, accessed: 2018-04-09.

[21] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum, "One-shot learning by inverting a compositional causal process," in *Advances in neural information processing systems*, 2013, pp. 2526–2534.

[22] Z. Zhuang, X. Kong, E. Rundensteiner, J. Zouaoui, and A. Arora, "Unsupervised attributed sequence embedding: a deep learning approach," *In submission*, http://goo.gl/hfnSQB.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[25] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, 2016, pp. 1842–1850.

[26] W.-S. Zheng, S. Gong, and T. Xiang, "Towards open-world person re-identification by one-shot group-based verification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 591–606, 2016.

[27] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha, "An empirical study and analysis of generalized zero-shot learning for object recognition in the wild," in *European Conference on Computer Vision*. Springer, 2016, pp. 52–68.

[28] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.

[29] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in neural information processing systems*, 2017, pp. 1087–1098.

[30] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS central science*, vol. 3, no. 4, pp. 283–293, 2017.

[31] Y. Wu, Y. Lin, X. Dong, Y. Yan, W. Ouyang, and Y. Yang, "Exploit the unknown gradually: One-shot video-based person re-identification by stepwise learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5177–5186.

[32] D.-Y. Yeung and H. Chang, "A kernel approach for semisupervised metric learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 141–149, 2007.

[33] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *International Conference on Machine Learning*, 2007, pp. 209–216.

[34] F. Wang, J. Sun, and S. Ebadollahi, "Integrating distance metrics learned from multiple experts and its application in patient similarity assessment," in *International Conference on Data Mining*. SIAM, 2011, pp. 59–70.

[35] A. Mignon and F. Jurie, "Pcca: A new approach for distance learning from sparse pairwise constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2666–2672.

[36] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2288–2295.

[37] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity." in *Association for the Advancement of Artificial Intelligence*, 2016, pp. 2786–2792.

[38] P. Neculoiu, M. Versteegh, M. Rotaru, and T. B. Amsterdam, "Learning text similarity with siamese recurrent networks," *Proceedings of the 1st Workshop on Representation Learning for NLP*, p. 148, 2016.

[39] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[40] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[41] Y. Xu, J. H. Lau, T. Baldwin, and T. Cohn, "Decoupling encoder and decoder networks for abstractive document summarization," *MultiLing 2017*, p. 7, 2017.

[42] J. Sun, D. Sow, J. Hu, and S. Ebadollahi, "Localized supervised metric learning on temporal physiological data," in *International Conference on Pattern Recognition*. IEEE, 2010, pp. 4149–4152.

[43] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.