IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 28, 2019, accepted December 25, 2019, date of publication January 3, 2020, date of current version January 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963742

Source Model Selection for Deep Learning in the Time Series Domain

AMIEL MEISELES AND LIOR ROKACH
Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Be'er Sheva 8410501, Israel
Corresponding author: Amiel Meiseles (amielm@post.bgu.ac.il)

ABSTRACT Transfer Learning aims to transfer knowledge from a source task to a target task. We focus on a situation when there is a large number of available source models, and we are interested in choosing a single source model that can maximize the predictive performance in the target domain. Existing methods compute some form of ''similarity'' between the source task data and the target task data. They then select the most similar source task and use the model trained on it for transfer learning. Previous methods do not account for the fact that it is the model parameters that are transferred rather than the data. Therefore, the ''similarity'' of the source data does not directly influence transfer learning performance. In addition, we would like the possibility of confidently selecting a source model even when the data it was trained on is not available, for example, due to privacy or copyright constraints. We propose to use the truncated source models as encoders for the target data. We then select a source model based on how well it clusters the target data in the latent encoding space, which we calculate using the Mean Silhouette Coefficient. We prove that if the encodings achieve a Mean Silhouette Coefficient of 1, optimal classification can be achieved using just the final layer of the target network. We evaluate our method using the University of California, Riverside (UCR) time series archive and show that the proposed method achieves comparable results to previous work, without using the source data.

INDEX TERMS Deep learning, model selection, time series, transfer learning.

I. INTRODUCTION

Multiple factors make training state-of-the-art Deep Learning (DL) models non-trivial. Powerful hardware is necessary to train DL models in a reasonable amount of time. In addition, large amounts of relevant, domain-specific data are necessary to train a high performing model; and, depending on the domain, data may not be easy to attain. Finally, designing a high-performing Deep Neural Network (DNN) requires expertise and usually some trial-and-error.

In almost all Machine Learning (ML) settings it is required that the training data and test data be drawn from the same distribution [1]. This is very different than the way humans learn. Humans can make use of knowledge from many different domains to inform their decisions, and make inferences. For example, even a small child can copy a gesture that they see performed by someone else [2], i.e., humans can easily transfer knowledge from the visual domain to the physical domain to make a gesture. Transfer Learning (TL) improves the training of ML models in a similar way. Essentially, TL is the process of using knowledge gained from learning a previous task (or tasks) to perform a new task. We refer to the previous task as the **source task** and the new task as the **target task**. The underlying assumption is that some of the knowledge necessary to perform the two classification tasks is similar between tasks, and can, therefore, be shared. This was shown to be the case with DL models for image classification, where earlier layers usually learn Gabor filters in many different domains [3].

Transfer learning allows us to achieve superior model performance in far less time, using far less data and modest hardware than would otherwise be possible. Many research groups in both academia and industry regularly publicize state-of-the-art pre-trained models they develop. An abundance of such pre-trained models is publicly available online, on sites such as ModelDepot,[1] which alone has over 50,000 pre-trained models. This motivates us to use TL to adapt these pre-trained models to new domains. We believe that as the number

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Luo.

[1] https://modeldepot.io

of available pre-trained models grows, TL will become an increasingly useful tool in the machine learning practitioner's toolbox. However, there is no consensus on how to select the best possible source model. Therefore, DL practitioners typically select a source model based on their intuition and experience.

Pan and Yang [1] created a transfer learning taxonomy that contains three types of Transfer Learning; Inductive Learning, Transductive Learning, and Unsupervised Learning. Inductive learning is when the source and target tasks are different. Transductive learning is when the source and target tasks are the same but the data is different and labels are available only for the source data. Unsupervised learning is when no labeled data is available, for both the source and target tasks. This work focuses on the Inductive learning setting, where at least some labeled target data is available.

Inductive learning is further divided into four subtypes of transfer learning. **Instance Transfer** uses some of the data from the source task when learning to perform the target task. In this case, a model trained on the source task is not used at all for the target task. An additional type of transfer learning, named **Feature Representation Transfer**, uses the source data and the target data to learn a representation of both sets of features that reduces the error in both domains. **Relational Knowledge Transfer** transfers knowledge about the relationship between samples within the source data, to the target data. The final subtype of inductive transfer learning is **Parameter Transfer**, in which model parameters learned from the source task are reused when learning to perform the target task. Yosinski *et al.* [3] use this type of transfer learning between two image classification tasks. It is important to note, that only Parameter Transfer can be used if the source model is available but the source data is not.

In general, TL research questions can be divided into three categories [1]:

1) *What* knowledge can be transferred?
2) *How* should this knowledge be transferred?
3) *When* is transferring knowledge beneficial?

The first two questions are interrelated since the method we use for transferring knowledge dictates what knowledge can be transferred. For example, if TL is carried out by transferring model parameters, the raw data from the source task can not be used, even if it would potentially be useful for learning to perform the target task.

The majority of research in TL relates to *what* knowledge can be transferred and *how* to transfer it [1]. In our view, the question of when to transfer is of great importance since TL can sometimes be detrimental to learning the target task, i.e. it may cause *negative transfer* [4]. If we can not predict negative transfer in advance the first two questions become irrelevant, as we do not know if conducting transfer learning will be beneficial. In this work, we propose a method of answering the question of when to transfer. We propose a method of selecting which pre-trained model to fine-tune by predicting its relative predictive performance after fine-tuning.

A naive method would be to fine-tune all of the models and measure their performance on a holdout validation set. This would be incredibly inefficient, being that a very large number of potential source models exist. The search for the optimal pre-trained model could be optimized by using methods designed to solve the Optimal Stopping Problem [5]. However, even these methods would require fine-tuning a very large number of models.

Existing approaches in the literature calculate the similarity between the dataset used for pre-training (source data) and the dataset to be used for fine-tuning (target data) [6]. This similarity is then used as a predictor for the performance of the fine-tuned model after TL. The intuition behind this approach is that if the source and target datasets are similar the tasks they represent are likely similar. Therefore, knowledge of how to perform the source task, which is embedded in the source model, will likely help in performing the target task. Unfortunately, these methods do not account for the loss of information between the source data and a model trained on the source data. Such methods also require access to the source data, which is often unavailable, among other reasons, due to privacy or copyright constraints.

In the vast majority of previous works, the source model is selected manually, and the effect of using alternative models is not evaluated or discussed. To truly assess how well a method selects the optimal source model, it is necessary to calculate the ground truth, i.e. which model yields the best predictive performance on the target task after fine-tuning. We, therefore, must perform transfer learning many times between potential source tasks and target tasks. The largest scale transfer learning experiment of this kind was done by Fawaz *et al.* [6] in the time-series classification domain. They employed transfer learning a total of 7140 times, once between each of the datasets in the UCR archive [7], and presented a method for selecting the optimal source model. We, therefore, assess our method based on the results of their transfer learning experiments. By using the same ground truth results as Fawaz *et al.* [6] we ensure that changes in performance are the product of our source model selection method and not of the transfer learning procedure. Additionally, the UCR archive [7] is a good testing ground for TL, in general, as many of the datasets in the archive have very few samples (e.g. the DiatomSizeReduction dataset has only 16 training samples) and are, therefore, prime candidates for TL.

The main contributions of this paper are:

- We present a method of selecting the source domain with the best performance after fine-tuning using only the target data and the pre-trained model. We show that our method performs comparably to the state-of-the-art method for source model selection [6] without access to the training data.
- We prove that if our method assigns an optimal score to a source model an optimal classifier can be constructed using only a single layer multiclass perceptron.

II. NOTATION AND DEFINITIONS

We adopt notation and definitions similar to those of Fawaz *et al.* [6].

We define the domain we are transferring knowledge from as the *source domain* and the domain we are transferring knowledge to as the *target domain*. Similarly, we will refer to the model we are transferring from as the *source model* or *source network* and the model we are transferring to as the *target model* or *target network*.

Throughout this paper, we will denote entities related to the source domain with the subscript *source* and entities related to the target domain with the subscript *target*.

Definition 1: A dataset $D = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ is a collection of N labeled time-series, where x_i is a time-series of length T and y_i is its corresponding label. We denote the time-series $\{x_i | i \in \{1, 2, \cdots, N\}\}$ and labels $\{y_i | i \in \{1, 2, \cdots, N\}\}$ for dataset D_j, as X_j and Y_j, respectively.

Definition 2: Transfer Learning (TL) is the process of training a model, M_{source}, on a source dataset, D_{source}, initializing a second model with all or some of the first model's parameters, and then training the second model, M_{target}, on the target dataset, D_{target}. In our work, all models are DNN's and the parameters of the models are the network's weights.

Definition 3: We define **Source Model Selection** (SMS), as selecting a single source model, M_{source}, out of a given set of source models, that maximizes the predictive performance on the target dataset, after fine-tuning. Our method performs source model selection without using the data from the source dataset.

III. RELATED WORK

Deep Learning approaches to Time-Series Classification (TSC) are either Generative or Discriminative [8]. Generative methods create a representation of the time-series before training the classification model. These methods are composed of multiple stages, which adds potential sources of error to the learning process. On the other hand, discriminative methods train a classification model directly on the time-series data, in an end-to-end fashion. Discriminative methods are also generally more accurate than generative methods for TSC [8]. We, therefore, prefer to use a discriminative approach and focus our survey of related literature on discriminative DL models.

In this section, we survey previous works that are related to our method. In section III-A we present previous works on deep learning for time-series data. Section III-B details previous works on transfer learning using DNN's for time-series data. Finally, we present previous works on selecting the appropriate source model for transfer learning, in the time-series domain, in section III-C.

A. DEEP LEARNING FOR TIME-SERIES CLASSIFICATION

Multi-Scale Convolutional Neural Networks (MCNN) [9] was the first method to use DL directly on the raw time-series in the UCR archive [7]. The MCNN framework first applies multiple transformations on the input sequences, in parallel. All transformations are fed into a convolutional layer and then a max pool layer. The outputs are then concatenated and fed into convolutional, max-pooling, fully connected and softmax layers, sequentially. The MCNN framework is arguably not an end-to-end approach, as the transformations it applies to the input sequences are not learnable.

The first end-to-end DL architecture for TSC was t-LeNet [10]. The architecture of t-LeNet is a CNN very similar to LeNet [11]. To achieve competitive performance with MCNN, Le Guennec *et al.* [10] had to employ data augmentation and a non-trivial training procedure.

Both MCNN and t-LeNet connect the final convolutional layer directly to a fully connected layer. This direct connection constrains the input sequence length to be constant. Moreover, this makes translation invariance between the different parts of the sequence impossible, which is undesirable in most real-world time-series problems.

Wang *et al.* [12] propose a Fully Convolutional Network (FCN) and a Residual Network (ResNet). FCN and ResNet are both classic CNN architectures (i.e. blocks composed of convolution and batch normalization), except that a global average-pooling operation is performed between the final convolutional layer and the fully connected layers used for classification. The global pooling makes these architectures independent of the input sequence length and, therefore, fully transferable between datasets (except the final layer used for classification, which has one neuron per class). An added benefit is that the global pooling reflects which part of the sequence is being used for classification, enabling visual interpretations of model predictions [12]. FCN and ResNet are evaluated on a subset of 44 datasets from the UCR archive [7]. FCN achieved state-of-the-art results, compared to MCNN and previous distance-based TSC methods. ResNet also achieved competitive results. It is worth noting that Fawaz *et al.* [8] report that, in their experiments on the entire UCR archive [7], ResNet significantly outperformed FCN.

B. TRANSFER LEARNING FOR TIME-SERIES CLASSIFICATION

Transfer learning has been successfully employed in many domains including Image Classification [13], Robotics [14], Medical Imaging [15], Natural Language Processing [16], [17], Automatic Speech Recognition [18], [19] and others.

Within the time-series domain, TL has been applied to problems such as wind speed prediction [20] and electricity load prediction [21]. However, most research in the TSC domain is evaluated on the UCR archive [7].

Earlier approaches to TL for TSC learn a fixed-length encoder from sequences in a source domain (or source domains). This encoder is then applied to the target domain sequences to create encodings of the target sequences. A DL model is then trained to classify these encodings, using the target data.

TimeNet [22] uses Recurrent Neural Networks (RNN), composed of multilayered Gated Recurrent Units, to generate

fixed-length encodings in a sequence-to-sequence [23] fashion. First, an encoder RNN and a decoder RNN are trained on the source sequences. The encoder RNN is then re-used in the target domain whereas a new decoder RNN is learned from scratch on the target domain. All of the source sequences, from the 24 datasets used for training were combined into one meta dataset of source sequences. This enables TimeNet to train only a single decoder RNN during the initial training phase. TimeNet is evaluated on 30 datasets from the UCR archive [7], that were not used during the training phase.

ConvTimeNet (CTN) [24] is a more recent DL architecture proposed for transfer learning for TSC. CTN draws inspiration from a few modern DL architectures. The CTN architecture is similar to FCN [12] in that it has convolutional layers followed by global average-pooling and then a fully connected output layer with a softmax activation. In addition, the residual connections in CTN are similar to those of ResNet. The core advantage of the CTN architecture is the use of multiple 1-d convolutional filters, of different lengths, in parallel, similar to inception modules [25]. CTN is trained and evaluated using the same method and data as TimeNet [22]. Kashiparekh *et al.* [24] report that CTN outperforms ResNet, however, they do not compare it to other DL approaches for TSC such as TimeNet [22] or FCN [12].

Serrá *et al.* [26] use transfer learning in a multi-task learning setting. They divide the UCR time-series archive [7] into seven domains, each comprised of multiple datasets. A meta-dataset is constructed for each subset of six domains by combining all of their datasets. Next, they train an encoder on each meta-dataset. Each of these encoders is then transferred to the seventh domain not included in the source meta-dataset. The encoder architecture they propose is a CNN followed by an attention mechanism and then a fully connected layer. They show that their method achieves state-of-the-art performance on the UCR archive.

C. SOURCE MODEL SELECTION FOR TIME-SERIES CLASSIFICATION

In some cases, TL can have a negative impact on performance when compared to training from scratch on the target data alone. This phenomenon is known as *negative transfer* [4]. It is, therefore, useful to anticipate a source model's performance before carrying out TL. Almost all previous TL methods for DL propose to select the source model either manually, based on subjective similarity of the domains or randomly. Some methods circumvent this decision by training the source model on many, or all, source domains at once and, therefore, do not have to decide which domain to transfer from. However, this approach is very inefficient and quickly becomes intractable with the vast number of pre-trained models currently available for use.

Inter-Datasets Similarity (IDS) [6] was the first method proposed for SMS. The underlying assumption of IDS is that the greater the similarity the source and target datasets are, the better the target model will perform after fine-tuning on the target dataset. First, IDS represents each class in each dataset with one prototype sequence using Dynamic Time Warping Barycenter Averaging [27]. Next, for each pair of datasets, D_{source} and D_{target} the Dynamic Time Warping (DTW) distance is calculated between each source-target pair of prototype sequences. IDS defines the similarity between two datasets to be the minimum DTW distance between all of their prototype sequences. IDS calculates a pairwise similarity measure between all the of the datasets in a given archive. To validate IDS, Fawaz *et al.* [6] perform transfer learning between all datasets in the UCR archive [7] and record their test performance. All models used the FCN [12] architecture.

IDS uses the similarity between the source and target datasets as a proxy for predicting the performance of the fine-tuned model. Unfortunately, this does not take into account the effect of training the base model at all. In an extreme case, the base model could be completely random, but IDS would not take this into account. Our method closes this gap by using the source model directly to predict the performance of transfer learning.

IV. METHOD

In this section, we describe our method in detail. Section IV-A describes the architecture and training procedure for the base network we used. We then proceed to detail our method for source model selection in section IV-B.

A. BASE NETWORKS

DNNs can be viewed as a composition of successive feature extractors. Each layer accepts the previous layer's extracted features as input and outputs more abstract features. Due to this hierarchical structure, it is necessary that the target network be initialized to have the same architecture and parameters as the source network from the input up to layer ℓ.

If all sequences in both D_{source} and D_{target} are of equal length the initialization is trivial, however, this is rarely the case. We could transform all target sequences to have the same length as the source dataset sequences if it is constant. This would add complexity to the transfer process and make the analysis of the results more difficult due to an additional source of potential error. However, when the source dataset sequences are not of constant length then this approach is not applicable. It is therefore preferable, to use a base model that is sequence length invariant, in our case FCN [12], which is depicted in figure 1, from left to right.

The FCN architecture is composed of three sequential parts:

1) **Three 1-d convolutional layers** - The first, second and third convolutional layers have 128 filters of length 8, 256 filters of length 5 and 128 filters of length 3, respectively. Each layer is followed by Rectified Linear Unit (ReLU) non-linearity and a Batch Normalization layer. Each filter generates it's own channel which is input into the next layer.

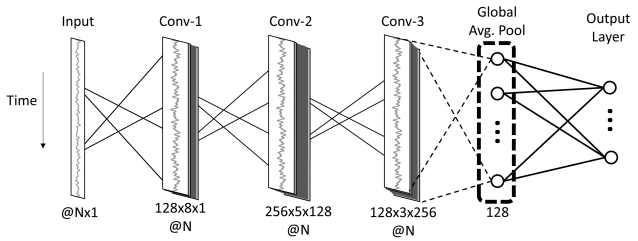

FIGURE 1. Base model architecture- We use the FCN [12] architecture for all models. The input is a single channel time-series sequence of length N. The three convolutional layers are labeled as: *number of filters* x *filter length* x *filter depth* @ *sequence length*, similar to the notation in DeepFace [28]. The thick dashed rectangle represents the encodings that are extracted after the average pooling layer.

2) **Global average pooling layer** - An average of all the values in each channel in the previous layer.
3) **Fully connected output layer** - The output layer contains one unit for each of the classes in the dataset. A softmax activation is used in order to transform the raw outputs into a probability distribution.

We adopt the training procedure proposed by Fawaz *et al.* [6] and make use of their trained models. All networks were trained independently using the Adam [29] optimizer for 2000 epochs with a batch size of 16. Parameters for Adam were set to $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, where α, β_1 and β_2 are the learning rate, first-moment exponential decay rate and second-moment exponential decay rate, respectively. This is in accordance with the best practices recommended by Kingma and Ba [29].

B. SOURCE MODEL SELECTION

Our method has three stages:

a) Encode the target training data using each of the source models truncated after the global average pooling layer. This is visually represented by the dashed rectangle in figure 1.
b) Calculate the Mean Silhouette Coefficient (MSC) for each set of encodings using the cosine distance metric.
c) Sort the source models according to their MSC score.

Figure IV-B is a visual illustration of the stages of our method, from left to right. The lettering we use in the figure corresponds to the lettering used above.

In this work, we choose to truncate the source models after the global average pooling layer because all of the layers except the output were transferred from the pre-trained source models to the target models before fine-tuning in the transfer learning experiments performed by Fawaz *et al.* [6]. We view these truncated models as *encoders* and use them to produce *encodings* of X_{target_train}. These encodings are in effect vectors in a 128-dimensional encoding space (the output size of the global average pooling layer).

The assumption of transfer learning is that weights from a pre-trained model are a better starting point for fine-tuning than randomly selected weights. We suggest that if this is the case the encodings representing sequences belonging to the same target class will be close to each other and far from encodings representing sequences from the other classes, before fine-tuning the source model. In this sense we expect the encodings from the same class to be clustered together. If we can assess the *quality* of this clustering, then we can give an apriori estimate of how the fine-tuned model will perform. We, therefore, use the Mean Silhouette Coefficient (MSC) [30], a clustering quality metric, to assess the *quality* of the encodings produced by a given truncated source model. We assess the effect of using alternate clustering functions in section VI-B. It is important to note that our method can compare models built using different architectures being that the MSC is calculated independently for each source domain.

The Silhouette Coefficient for a single encoding vector i, with the label A is defined as:

$$s(i) = \frac{b - a}{max(a, b)} \tag{1}$$

where:

$$a = \frac{1}{|A| - 1} \sum_{\substack{x \in A, \\ x \neq i}} d(i, x) \quad \text{and} \quad b = \min_{C \neq A} \frac{1}{|C|} \sum_{x \in C} d(i, x)$$

d is the distance between two encodings and $|\cdot|$ is the number of encodings labeled with a given label. We compare the performance of the Manhattan, Euclidean and Cosine distances in section VI-C. The cosine distance metric provides the best performance.

The Silhouette Coefficient is the relationship between the within label distances (a) and the nearest label distances (b). Being that $|b - a| \leq max(a, b)$, the Silhouette Coefficient is bounded between -1 and 1.

The MSC of all encodings of X_{target_train} by a given truncated M_{source}, which we denote as E is then:

$$MSC(E, labels) = \frac{1}{|E|} \sum_{i \in E} s(i) \tag{2}$$

Similar to the Silhouette Coefficient, the MSC value is bounded between -1 and 1. We illustrate the entire procedure of calculating the MSC in figure 3. In order to help develop a visual intuition for MSC values, in figure 4, we show three sample 2-d encodings along with their MSC scores. Each of the three encodings is shown in a separate scatter plot. We represent the different ground truth classes using different shapes and colors. In figure 4, $encoding_3$ achieves a relatively high MSC of 0.78 and indeed we see that the classes are well separated in the encoding space. The classes are not as well separated by $encoding_1$ in the encoding space and therefore it scores a lower MSC of 0.51. The scatter plot for $encoding_2$ shows that the classes are very mixed in encoding space and therefore it achieves an MSC of approximately 0.

A simple mathematical analysis of MSC shows that it approaches -1 when both $a > b$ and b approaches 0. However, if b is equal to 0, indicating all encodings are almost co-located,[2] the classes are not separable in encoding space.

[2]If all encodings are actually co-located, the MSC is undefined

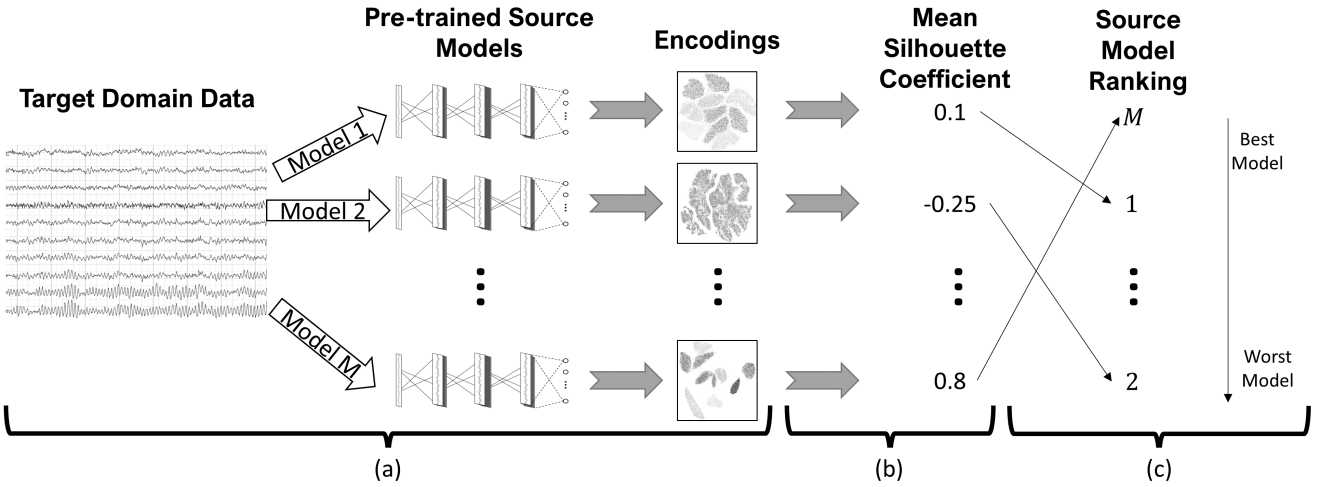

FIGURE 2. A visual depiction of source model selection for a single target dataset, from left to right. **(a)** First, the training target sequences are encoded using each of the truncated pre-trained source models. **(b)** Next, the MSC is calculated for each of the source model encodings in encoding space. **(c)** Finally, the source models are sorted by their MSC, in non-increasing order, producing a ranking of estimated performance of transfer learning from each source model.

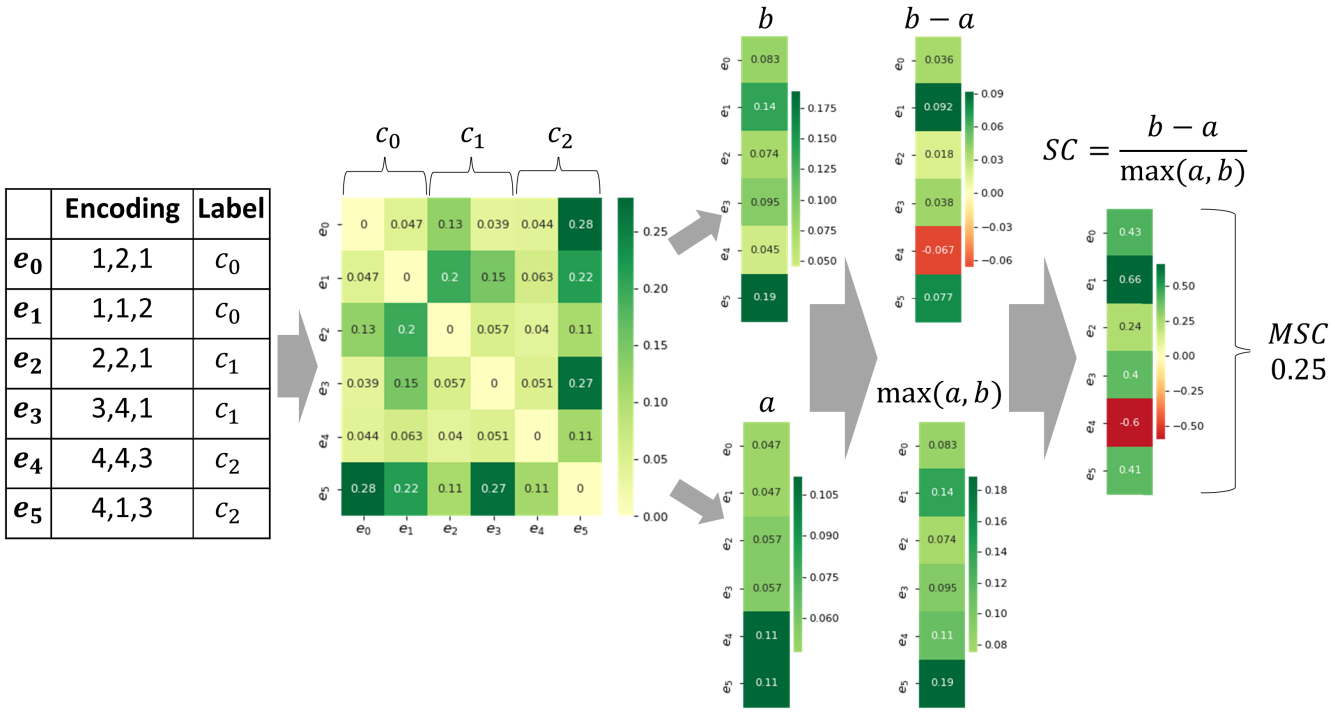

FIGURE 3. MSC Calculation- We first calculate the cosine distance between all encodings in a pairwise fashion. We can then calculate the values of a and b, the within label distances and nearest label distances, respectively, for each encoding. Given a and b, we calculate $b - a$ and $max(a, b)$ for each encoding. Finally, we calculate the silhouette coefficient for each encoding and calculate the mean.

On the other hand, MSC approaches 1 when both $b > a$ and a approaches 0. If a is equal to 0, then all encodings with the same label are co-located in the encoding space and if $b > a$ each label is encoded to a different point in the encoding space, indicating that a perfect separation is possible. Furthermore, we prove that when MSC is 1, perfect classification of the encodings can be achieved using a single layer multiclass perceptron. This is of critical importance, since, the base model's last layer, which is equivalent to a single layer multiclass perceptron, learns a mapping from the encodings of the previous layers directly to the classes.

We present figure 5 as a visual aid for the idea behind the proof in 2 dimensions.

Proof 1: When the MSC of the encodings is 1, perfect classification can be achieved using a single layer multiclass perceptron.

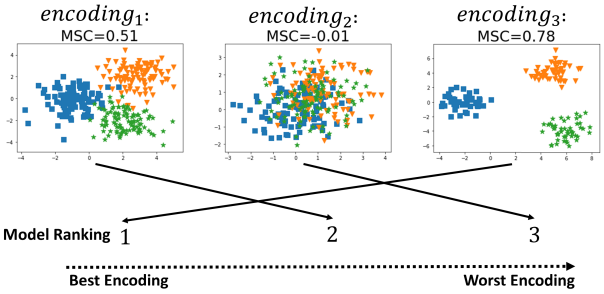

FIGURE 4. Visual intuition of MSC sorting - The figure shows three different 2-d encodings and their MSC scores. The figure also shows how these three encodings would be ranked by our method.

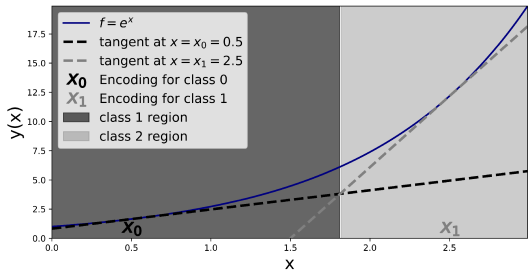

FIGURE 5. Visual intuition of proof - The figure illustrates how a perfect classifier can be created for the 1-d class encodings x_0 and x_1. Each dashed line is represented by a single neuron in the last layer of the network. When a softmax is applied to the outputs of all the neurons a perfect classification is achieved. We use e^x as an example of a convex function.

Since when $MSC = 1$, $a = 0$, all the vectors belonging to the same class are encoded as the same vector. We denote the vector representing the i-th class as $\vec{x_i}$. Let $g(x)$ be some convex function. Let $f_i(\vec{x})$ be the hyperplane tangent to $g(\vec{x})$ at $\vec{x_i}$.

Lemma 1: If $MSC = 1$ then $\underset{i}{\text{argmax}} f_i(\vec{x_j}) = j$ for all i.

Boyd and Vandenberghe [31] prove that:

$$f_i(\vec{x_i}) = f_j(\vec{x_i}) \text{ iff } \vec{x_i} = \vec{x_j}, \quad \text{otherwise}, f_i(\vec{x_i}) > f_j(\vec{x_i}) \tag{3}$$

Since when MSC $= 1$, $a = 0$ and $b > a$, $b > 0$. Therefore, all of the classes have different encodings, i.e. $\vec{x_i} = \vec{x_j}$ iff $i = j$. Therefore:

$$\underset{i}{\text{argmax}} f_i(\vec{x_j}) = j$$

Lemma 2: The perceptron where each function f_i is a hyperplane tangent to $g(\vec{x})$ at x_i, classifies all encodings x_i as class i, i.e. a perfect classification.

From Lemma 1 we have that for any vector, $\vec{x_a}$ representing the encodings of class a,

$$\underset{i}{\text{argmax}} f_i(\vec{x_a}) = a$$

Being that a multiclass perceptron is defined as a model for which:

$$\hat{y} = \underset{i}{\text{argmax}} f_i(x) \quad \text{where} \quad f_i(x) \text{ is linear in } x \tag{4}$$

the perceptron composed of $\{f_i | i \epsilon \{classes\}\}$ provides a perfect classification of the encodings x_i. ■

We hypothesize, based on the proof above, that the higher the MSC of the encodings of X_{target_train} from a specific truncated source model, the better transfer learning from that source model will perform. We, therefore, calculate the MSC of all possible source models to encode the target data and sort the source models by their MSC in decreasing order. We then use this list to select which source domain to transfer from.

In algorithm IV-B we present the algorithm for ranking possible source models. The input to the algorithm is all

Algorithm 1 Source Model Ranking

Input: *models*: List of all pre trained base models
ℓ: layer from which to generate encodings
X_{target_train}: training sequences from D_{target}
Y_{target_train}: training class labels from D_{target}
Output: *rankings*: A |*models*|-length list with the source models ranked by their predicted transfer learning performance

1 *distances* $\leftarrow$ |*models*|-length list initialized to Inf
2 **for** $s \leftarrow 1$ **to** |*models*| **do**
3 $M_{source} \leftarrow models[s]$
4 *encoder* $\leftarrow M_{source}$ truncated at layer ℓ
5 *encodings* $\leftarrow encoder(X_{target\_train})$
6 $distances[s] \leftarrow -MSC(encodings, Y_{target\_train})$ ▷ calculate Mean Silhouette Coefficient using Cosine distance
7 *rankings* $\leftarrow argsort(distances)$ ▷ rank source models according to MSC distance in increasing order
8 **return** *rankings*

of the trained base models and a target dataset. We split the target dataset into the train and test subsets found in the UCR archive [7] and use only the train set for source model selection. We denote the training set sequences and class labels as X_{target_train} and Y_{target_train}, respectively. In line 1 we initialize *distances*, a list of the dissimilarity of each of the encodings of X_{target_train} to Y_{target_train}. In lines 2 through 6, we iterate over the indices of the pre-trained model list. In line 3 we fetch the pre-trained source model we are currently assessing. Next, in line 4, each source model is truncated after the global average pooling layer as illustrated by the dashed rectangle in figure 1. In line 5, we use the encoder to calculate the encodings for X_{target_train}. In line 6, we calculate the Mean Silhouette Coefficient of the encodings and the class labels, Y_{target_train}, using cosine distance. We then negate the MSC so that it can be interpreted as a distance metric between the source model and the target training data. After completing all iterations in lines 2-6, in line 7 the source models are ranked according to their distance calculated in line 6. Finally, in line 8 the ranked list of source models is returned.

V. EVALUATION

A. DATASET

We evaluate our method using the UCR time series archive [7]. The UCR archive [7] contains 85 univariate time-series

TABLE 1. A summary of the UCR archive - We summarize the datasets based. The table lists the minimum, maximum, median, mean and standard deviation of the number of classes per dataset, the length of the time-series, the size of the training set, the size of the test set and the total size of the dataset.

	Number of classes	Time series Length	Size of training set	Size of testing set	Total Size
min	2	24	16	20	40
max	60	2709	8926	8236	16637
median	3	300	200	400	780
mean	7.67	422.21	432.88	1164.8	1597.68
std. dev.	11.25	429.06	1016.19	1665.58	2417.73

datasets. Some of the datasets were originally multivariate but were divided up into multiple univariate datasets within the archive. An example of such a dataset is the Cricket dataset which is split into CricketX, CricketY, and CricketZ which correspond to the X, Y and Z measurements from an accelerometer. All sequences within each dataset are of equal length. In addition, all sequences are z-score normalized [32]. The archive splits each dataset into train and test sets which are kept fixed to eliminate the possible effects of different splits chosen by different researchers. In table 1 we summarize the basic characteristics of the UCR archive. Datasets in the archive vary in the number of classes, length of sequences and number of sequences allocated to training and testing. Interestingly, in the UCR archive, most datasets allocate the majority of the data to the test set.

In our work, the train sets are used for source domain selection, training the base models and fine-tuning the transferred models. The test sets are used only to evaluate the performance of the fine-tuned models.

1) ADVANTAGES OF THE UCR ARCHIVE

Using the UCR archive is appealing for multiple reasons. First, the UCR archive is the standard used by many time series researchers. Over a thousand papers have cited the UCR archive [33]. The archive ensures that researchers are not cherry-picking datasets that their method performs well on, as they are using the same datasets as the previous works they are comparing to. Furthermore, the UCR archive recommends a specific train-test split in order to ensure that even the training data can not be cherry-picked for a given method. Another significant advantage of the UCR archive is that it contains many datasets. This strengthens the hypothesis that a method that does well on the archive will generalize well. A third advantage of the UCR archive is that all the data is in the same format and has been preprocessed in the same way, namely z-score normalization [32]. This both makes the use of the datasets more straightforward and ensures that preprocessing is not the source of performance increases. Our final consideration is that we are interested in comparing our method to the previous baseline for source model selection which was evaluated on the UCR archive [6].

B. EXPERIMENTAL SETUP

In order to make full use of the UCR archive, transfer learning experiments should be performed between every pair of datasets, twice. First, one dataset should be used as the source and the other as target and then the reverse. There are $\binom{85}{2} = 3570$ pairs of datasets and each pair is used twice, therefore, 7140 transfer learning experiments are necessary.

Running all 7140 transfer learning experiments, sequentially, would take an estimated 168 days [6]. Using a cluster with 60 GPUs performing all of the experiments took Fawaz *et al.* [6] approximately a week. All trained models and raw results are freely available.[3] Since the base model training and fine-tuning are held constant between our method and IDS [6], we can confidently attribute the changes in performance to our method, as opposed to other sources of noise.

C. PERFORMANCE METRICS

Our method provides an apriori estimate of transfer learning performance for transferring from all possible source domains to all possible target domains. The ultimate goal of source model selection is to select the source model that performs best after fine-tuning on the target data. We will call this model the *optimal source model* for a given target domain. In order to assess the performance of the various source model selection methods we use multiple metrics to compare them with the results of the actual transfer learning experiments performed by Fawaz *et al.* [6].

The first metric we use is Top-1 Accuracy, as it best correlates with selecting the best source model. Top-1 Accuracy is the percentage of target domains for which the optimal source model is ranked first. Being that the UCR archive has 85 datasets, there are 84 potential source models for each target task. Therefore, for random selection, the expected accuracy is $1/84 = 1.2\%$ and the expected number of times the optimal source model is selected is $85 * (1/84) = 1.01$.

A disadvantage of using Top-1 accuracy is that it only takes into account the source model which was ranked first by our method. However, in the cases in which our method does not rank the optimal source model first, we would like to know how far off it was placed, as there is a big difference in ranking the optimal model second as opposed to 84th. Mean Reciprocal Rank (MRR) [34] enables us to measure how far the optimal source is from the top of the rankings. We, therefore, assess our performance using the mean MRR

[3] https://germain-forestier.info/src/bigdata2018/

in addition to the Top-1 Accuracy. MRR is defined as:

$$MRR = \frac{1}{|Q|}\sum_{i=0}^{|Q|}\frac{1}{rank_i} \tag{5}$$

where Q is a list of all the rankings and $rank_i$ is the rank of the source domain that yields the highest accuracy when transferred to the i-th target domain. The expected MRR for a random ranking of n possible source models is: $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{n}$. Specifically, the expected MRR for a random ranking of the 84 possible source models for each target is: $\frac{1}{84}\sum_{i=1}^{84}\frac{1}{i} = 0.0597$. We, therefore, expect the mean MRR for a random ranking to also be 0.0597. For brevity, we will refer to the mean MRR as MRR in the Results section.

D. ALTERNATIVE CLUSTERING QUALITY FUNCTIONS

Most clustering quality functions compare the true labels to those generated by the clustering. However, we are interested in assessing how well the encoding's distribution in encoding space matches the true class labels. We, therefore, are limited to clustering quality functions that compare vectors with a given labeling. The Mean Silhouette Coefficient, which we described earlier, is one such function. In this section, we present two other clustering quality functions.

1) VARIANCE RATIO CRITERION

The Variance Ratio Criterion (VRC) also known as the Calinski Harabasz score [35] is defined as:

$$VRC_k = \frac{SS_b/(k-1)}{SS_w/(N-k)} \tag{6}$$

where:

$$SS_b = \sum_{j=1}^{k} n_j \cdot (\bar{x}_j - \bar{x})^2 \quad \text{and} \quad SS_w = \sum_{j=1}^{k}\sum_{i=1}^{n_j}(x_{ij} - \bar{x}_j)^2$$

and k is the number of clusters, n_j is the number of sequences in class j, x_{ij} is the ith sample in class j, $\bar{x}_j$ is the element-wise mean sequence for class j and $\bar{x}$ is the element-wise mean sequence for all sequences.

SS_b is the between cluster variation and SS_w is the within-cluster variation. In this way, VRC is the ratio between the between cluster variation and the within-cluster variation. For a clustering with well-separated clusters, we expect $SS_b \gg SS_w$ which yields a high VRC score.

2) DAVIES-BOULDIN INDEX

The Davies-Bouldin index [36], which we denote as DB, is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k}\max_{i\neq j}\frac{s_i + s_j}{d_{ij}} \tag{7}$$

where k is the number of clusters, d_{ij} is the distance between the centroid of cluster i and the centroid of cluster j and s_x is the mean distance between the centroid of cluster x and all of the sequences in cluster x.

$s_i + s_j$ is, therefore, the sum of the mean distance from the centroid of the two different classes. Intuitively, if the encodings from class i and class j are poorly separated in encoding space, we expect $s_i + s_j \geq d_{ij}$. However, when clusters are well-separated we expect $s_i + s_j \ll d_{ij}$ for all clusters which yields a DB index close to 0.

VI. RESULTS

Interestingly, we did not find any significant connections between the size of the training sets and the performance increase after TL. This section is organized as follows. In section VI-A we assess the performance of our method. We explore the effects of using alternate clustering quality functions and distance metrics in sections VI-B and VI-C, respectively.

A. PERFORMANCE

Table 2 presents our evaluation of the performance of random source model selection, IDS [6] and our method (SMS) in the first, second and third rows, respectively. We discuss our evaluation in the next few paragraphs.

1) TOP-1 ACCURACY

As we mentioned earlier the expected top-1 accuracy for random selection is 1.01%. Our method ranks the optimal source model first 6 times, yielding a top-1 accuracy of 7.1%. IDS ranks the optimal source model first 4 times, yielding a top-1 accuracy of 4.7%. We conduct statistical significance testing using a two-tailed exact binomial test, with the null hypothesis that the methods select randomly, i.e. all possible source models are equally likely. Both methods are statistically significant, compared to random selection, with $p < 1e{-}3$ for our method and $p = 0.019$ for IDS. The difference between the methods is not statistically significant, despite SMS not using source data.

Being that neither method achieves a high top-1 accuracy, we turn to other metrics to assess the quality of the rest of the rankings beyond the first place.

2) MEAN RECIPROCAL RANK

As mentioned earlier the expected MRR for a random ranking is 0.0597. Our method scores an MRR of 0.115 and IDS scores an MRR of 0.1163, indicating that both models perform better than random.

B. CHOICE OF CLUSTERING QUALITY FUNCTION

In this subsection we evaluate the performance of using the Variance Ratio Criterion [35] and the Davies-Bouldin index [36] instead of the MSC and show that they achieve inferior results. We presented these clustering quality functions in detail in section V-D.

1) VARIANCE RATIO CRITERION

Our method, using VRC, ranks the optimal source model first 2 times, yielding a top-1 accuracy of 2.4%. This is not statistically different than random selection. The MRR for

TABLE 2. Comparison of source model selection methods - We compare our method (SMS) with IDS [6] and random source model selection. SMS's Top-1 accuracy is statistically significant, compared to random selection, which we signify with an *. The MRR of both SMS and IDS are similar and both are much better than random.

Method	MRR	Top-1			Uses Source Data?
		Accuracy	Hits	p-value	
Random Source Model Selection	0.0597	1.2%	1.01	1	No
IDS [6]	**0.1162**	4.7%	4	0.019 *	Yes
SMS - our method	0.115	**7.1%**	**6**	**0.0006** *	No

TABLE 3. Comparison of clustering quality functions and distance metrics - The first row shows the performance of random source model selection as a reference. Variance Ratio Criterion and Davies Bouldin Index both under-perform Silhouette Coefficient and are not statistically significant compared to random selection. Cosine distance shows a small improvement over the other distance functions used for MSC calculation all of which are statistically significant compared to random selection, signified by a *.

Clustering Quality Function	MRR	Top-1			Uses Source Data?
		Accuracy	Hits	p-value	
Random Source Model Selection	0.0597	1.2%	1.01	1	No
Variance Ratio Criterion	0.0617	2.4%	2	0.2686	No
Davies-Bouldin Index	0.0956	3.5%	3	0.0814	No
Silhouette Coefficient using Euclidean Distance	0.1066	5.9%	5	0.0036 *	No
Silhouette Coefficient using Manhattan Distance	0.1102	5.9%	5	0.0036 *	No
Silhouette Coefficient using Cosine Distance (SMS)	**0.115**	**7.1%**	**6**	**0.0006** *	No

our method using VRC is 0.0617. These results are listed in the second row of table 3.

2) DAVIES-BOULDIN INDEX

Our method, using DB, ranks the optimal source model first 3 times, yielding a top-1 accuracy of 3.5%. This is not statistically different than random selection. The MRR for our method using DB is 0.0956. These results are listed in the third row of table 3.

C. CHOICE OF DISTANCE METRIC

The Silhouette Coefficient can use any distance metric. We, therefore, evaluate the effects of using the Euclidean and Manhattan distances, instead of Cosine distance. Both Euclidean and Manhattan distances rank the optimal source model first, 5 times which yields a top-1 accuracy of 5.9%. This is statistically significant compared to random selection with $p = 0.0036$. There is a small difference in MRR between Euclidean and Manhattan distance, which score MRRs of 0.1056 and 0.1102, respectively. We list these results in the fourth and fifth rows of table 3, respectively. As can be seen in table 3, using both Euclidean distance and Manhattan distance yield lower top-1 accuracies and MRRs than using cosine distance.

VII. CONCLUSION

In this work, we presented a method for source domain selection without knowledge of the source training data. This is accomplished by encoding the target data using the truncated source models and then measuring the quality of the encoding using MSC. We proved that when the MSC is 1, perfect classification can be achieved using the final layer of the target network alone. Our method achieves an MRR of 0.115 and top-1 accuracy of 7.1%, which is statistically significant compared to random model selection. The top-1 accuracy and MRR of our method are comparable to the state-of-the-art [6], despite not having access to the source training data. This enables our method to be used in settings where the source data is not available due to privacy or copyright constraints.

We propose our method for transfer learning of TSC problems but it could be extended to other transfer learning problem types. We leave this as future work. We also leave the extension of our method to the case of multiple source models as future work.

We believe that to make transfer learning widely useful in practice, it is necessary to develop methods that select high performing source models even in the absence of the source training data.

Finally, all of our experiments and results can be reproduced using our code, which we provide online at: `https://github.com/amielm/Source-Model-Selection`.

ACKNOWLEDGMENT

The authors would like to thank I. Haran and R. Meiseles for their helpful insights during the writing of this article.

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[2] M. Carpenter, J. Call, and M. Tomasello, "Twelve- and 18-month-olds copy actions in terms of goals," *Develop. Sci.*, vol. 8, no. 1, pp. F13–F20, Jan. 2005.

[3] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 3320–3328. [Online]. Available: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

[4] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 11293–11302.
[5] T. P. Hill, "Knowing when to stop: How to gamble if you must—The mathematics of optimal stopping," *Amer. Sci.*, vol. 97, no. 2, pp. 126–133, 2009.
[6] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Müller, "Transfer learning for time series classification," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2018, pp. 1367–1376.
[7] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. (2015). The UCR time series classification archive. University of California, Riverside, Riverside, CA, USA. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data/
[8] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Müller, "Deep learning for time series classification: A review," *Data Mining Knowl. Discovery*, vol. 33, no. 4, pp. 917–963, Jul. 2019.
[9] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," 2016, *arXiv:1603.06995*. [Online]. Available: https://arxiv.org/abs/1603.06995
[10] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *Proc. ECML/PKDD Workshop Adv. Anal. Learn. Temporal Data*, 2016.
[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[12] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 1578–1585.
[13] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 558–567.
[14] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Proc. 1st Annu. Conf. Robot Learn.*, in Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78, Nov. 2017, pp. 262–270. [Online]. Available: http://proceedings.mlr.press/v78/rusu17a.html
[15] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1299–1312, May 2016.
[16] Z. Yang, R. Salakhutdinov, and W. W. Cohen, "Transfer learning for sequence tagging with hierarchical recurrent networks," 2017, *arXiv:1703.06345*. [Online]. Available: https://arxiv.org/abs/1703.06345
[17] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," 2015, *arXiv:1506.02075*. [Online]. Available: https://arxiv.org/abs/1506.02075
[18] Z. Huang, S. M. Siniscalchi, and C.-H. Lee, "A unified approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition," *Neurocomputing*, vol. 218, pp. 448–459, Dec. 2016.
[19] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4480–4490.
[20] Q. Hu, R. Zhang, and Y. Zhou, "Transfer learning for short-term wind speed prediction with deep neural networks," *Renew. Energy*, vol. 85, pp. 83–95, Jan. 2016.
[21] N. Laptev, J. Yu, and R. Rajagopal, "Reconstruction and regression loss for time-series transfer learning," in *Proc. SIGKDD MiLeTS*, 2018.
[22] P. Malhotra, V. Tv, L. Vig, P. Agarwal, and G. Shroff, "TimeNet: Pretrained deep recurrent neural network for time series classification," 2017, *arXiv:1706.08838*. [Online]. Available: https://arxiv.org/abs/1706.08838
[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[24] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "ConvTimeNet: A pre-trained deep convolutional neural network for time series classification," 2019, *arXiv:1904.12546*. [Online]. Available: https://arxiv.org/abs/1904.12546
[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[26] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series," in *Proc. CCIA*, 2018, pp. 120–129.
[27] F. Petitjean and P. Gançarski, "Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment," *Theor. Comput. Sci.*, vol. 414, no. 1, pp. 76–91, Jan. 2012.
[28] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human–level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1701–1708.
[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[30] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.
[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[32] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining Knowl. Discovery*, vol. 31, no. 3, pp. 606–660, May 2017, doi: 10.1007/s10618-016-0483-9.
[33] H. A. Dau, A. Bagnall, K. Kamgar, C.-C.-M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The UCR time series archive," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019.
[34] E. M. Voorhees, "The trec-8 question answering track report," *Trec*, vol. 99, pp. 77–82, Nov. 1999.
[35] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Statist.-Theory Methods*, vol. 3, no. 1, pp. 1–27, 1974.
[36] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.

AMIEL MEISELES received the B.Sc. degree in computer science from the Jerusalem College of Technology, Jerusalem, Israel, in 2012, and the M.Sc. degree in bioinformatics from NJIT, Newark, NJ, USA, in 2013. He is currently pursuing the Ph.D. degree with the Department of Software and Information Systems Engineering, BGU. From 2014 to 2019, he worked as a Data Scientist and applied Machine Learning Researcher on projects in various domains. He is also working as a Data Scientist with the Machine Learning Laboratory, BGU. His research interests include machine learning, deep learning, and their applications in the real world.

LIOR ROKACH is currently a Data Scientist and a Professor with the Department of Software and Information Systems Engineering, where he is also the Chair. He has established the Machine Learning Laboratory and the Big Data Laboratory, which promote innovative adaptations of machine learning and data science methods to create the next generation of intelligent systems. His research interests include the areas of data science, machine learning, big data, deep learning and data mining, and their applications to: recommender systems, cyber security, information retrieval, information extraction, and medical informatics. He also serves as an Editorial Board Member of ACM TIST and an Area Editor of *Information Fusion* (Elsevier).