Contrastive Adversarial Knowledge Distillation for Deep Model Compression in Time-Series Regression Tasks

Qing Xu[a], Zhenghua Chen[a], Mohamed Ragab[b], Chao Wang[c], Min Wu[a], Xiaoli Li[a]

[a]*Machine Intellection Department, Institute for Infocomm Research, A*STAR, Singapore*
[b]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
[c]*School of Computer Science, University of Science and Technology of China, Hefei, China*

Abstract

Knowledge distillation (KD) attempts to compress a deep *teacher* model into a shallow *student* model by letting the student mimic the teacher's outputs. However, conventional KD approaches can have the following shortcomings. First, existing KD approaches align the global distribution between teacher and student models and overlook the fine-grained features. Second, most of existing approaches focus on classification tasks and require the architecture of teacher and student models to be similar. To address these limitations, we propose a contrastive adversarial knowledge distillation called CAKD for time series regression tasks where the student and teacher are using different architectures. Specifically, we first propose adversarial adaptation to automatically align the feature distribution between student and teacher networks respectively. Yet, adversarial adaptation can only align the global feature distribution without considering the fine-grained features. To mitigate this issue, we employ a novel contrastive loss for instance-wise alignment between the student and teacher. Particularly, we maximize similarity between teacher and student features that originate from the same sample. Lastly, a KD loss is used to for the knowledge distillation where the teacher and student have two different architectures. We used a turbofan engine dataset that consists of four sub-datasets to evaluate the model performance. The

Email addresses: `xu_qing@i2r.a-star.edu.sg` (Qing Xu), `chen0832@e.ntu.edu.sg` (Zhenghua Chen), `mohamedr002@e.ntu.edu.sg` (Mohamed Ragab), `cswang@ustc.edu.cn` (Chao Wang), `wumin@i2r.a-star.edu.sg` (Min Wu), `xlli@i2r.a-star.edu.sg` (Xiaoli Li)

results show that the proposed CAKD method consistently outperforms state-of-the-art methods in terms of two different metrics.

Keywords: Knowledge Distillation, Contrastive Learning, Adversarial Learning, Time Series Regression

1. Introduction

Recent years have witnessed a significant achievement of deep neural networks (DNNs) in various real-life applications, such as face recognition, machine translation, autonomous vehicles, etc. With the increasing concerns on data privacy, energy efficiency and communication latency, lots of these DNNs are required to be deployable on edge devices, e.g., mobiles and Internet of Things (IoT) devices. However, a deep learning model with better performance often comes with more complex architecture, resulting millions of model weights and tons of floating-point operations [1]. It hinders the deployment of complex deep models on resource-limited environments like edge devices.

A real industry case is prognostic and health management (PHM) for intelligent industrial manufacturing. As an essential part of PHM, machine remaining useful life (RUL) prediction is crucial for reducing maintenance cost and improving system reliability [2, 3]. The RUL prediction algorithms are often required to run on edge devices for real-time processing and fast decision-making in smart factories [4]. However, most of previous works pay much more attention on prediction accuracy than model complexity. Long short-terms memory (LSTM) and convolutional neural network (CNN) are two commonly-used deep learning networks for RUL prediction. Moreover, LSTM has exhibited outstanding capability on extracting informative features and outperformed CNN based models [5, 6, 7]. However, the LSTM generally has much higher computational complexity than CNN due to its unique structure of cascade connections. Hence, a question comes up: is there a deep learning algorithm that can achieve similar performance as LSTM but also as compact as CNN?

To address the conflict between model performance and model efficiency, many advanced techniques have been proposed to compress deep learning models, such as parameter pruning [8], parameter quantization [9, 10], low-rank factorization [11], and knowledge distillation (KD) [12, 13]. Among them, KD is particularly effective due to the paradigm of transferring knowledge (soft logits) [13] and/or intermediate representations [14] from a large network (termed *Teacher*) to a small network (termed *Student*). Previous works have shown that the compact student trained with KD can converge faster and achieve a better performance than an independent network trained without a teacher's supervision [15]. However, there are still some challenges by using KD for model compression. Firstly, previous KD research mainly focuses on classification tasks and few of them focus on regression tasks like object localization [16] and camera pose regression[17]. There is lack of research on KD methods for time-series regression tasks. Secondly, as aforementioned, for time-series regression tasks, it is more practical to transfer knowledge between disparate network architecture (e.g., from a LSTM based network to a CNN based network). But students in previous works usually share a similar network architecture as teachers, either shallower [18, 19] or thinner but deeper [14]. Thirdly, most existing methods either only align the global feature distributions between teacher and student models [20, 21], or only consider the sample-wise feature alignment [22, 23]. We show that the integration of distribution-wise and sample-wise feature alignment outperforms other state-of-the-art KD methods in cross-architecture knowledge distilling scenario.

To address the above issues, in this paper, we propose a novel contrastive adversarial knowledge distillation (CAKD) approach for model compression in the regression task of machine RUL prediction. In particular, the proposed approach aims to distill knowledge from a complex LSTM-based teacher to a simple CNN-based student for RUL prediction. The adversarial loss is used for the automatic alignment of global features, while the contrastive loss is used to align fine-grained features by instance-wise discrimination.

The main contributions of the proposed method are summarized as follows.

- We propose an adversarial learning based approach to automatically minimize the discrepancy of feature distributions between the student and teacher. Meanwhile, we propose a fine-grained sample-wise feature adaptation between the student and teacher models by using contrastive learning.
- The integration of distribution-wise and sample-wise feature alignment can effectively transfer the knowledge between disparate network architectures, which is more practical in time series regression tasks.
- Extensive experiments demonstrate that the propose CAKD method achieves better performance than state-of-the-art KD methods.

The rest of the paper is organized as follows. Section 2 reviews some related works on knowledge distillation and RUL prediction. Section 3 presents the details of our proposed CAKD method. Section 4 introduces our experimental setup, followed by experimental results, ablation study and sensitivity analysis. Finally, Section 5 concludes this work.

2. Related Works

Knowledge Distillation. The authors in [12] first introduced the idea of using a compact model to approximate the function learned by a larger and better-performing model. The authors in [13] further extended this idea by making the student mimic teacher's soften logits and termed it as knowledge distillation. In addition to just mimicking the logits, Romero et al. proposed to adopt hint-based training scheme for aligning the feature maps [14] (i.e., feature distillation), where the L_2 distance was chosen as the metric for measuring the distance between two feature maps.

Recently, feature distillation has attained more attention. Zagoruyko and Komodakis proposed to transfer the spatial attention maps from a powerful teacher network to a smaller student network [21]. A novel pairwise similarity matrix was proposed in [20] to preserve interrelationships of similar samples in student's representation space as those in teacher's. Yim et al. defined the inner product between features from two layers as flow and transferred the flow to student instead of knowledge [24]. Nikolaos et al. directly matched the probability distribution of the data between teacher's and student's feature spaces [25]. Instead of hand-crafting the knowledge, adversarial methods are introduced in [26, 27, 28, 29] for feature alignment between teacher and student networks. Another work closely related to our research is the contrastive distillation approach introduced in [22]. It aims to maximize similarity between teacher and student's representations that are originated from the same instance while minimizing similarity between teacher and student's representation that come from different samples. Above adversarial methods and contrastive learning method work with the configuration that teacher and student share a similar network architecture for classification tasks. In our work, we extend the adversarial learning to cross-architecture scenario for feature alignment in time-series regression tasks. Besides, the success of KD methods is due to the informative knowledge lying in the logits from teacher. Hence, most of previous works focus on classification task. Meanwhile, KD methods have also been demonstrated to be suitable for regression tasks like object localization [16] and camera pose regression [17]. We empirically show that for time-series regression problems, the 'Soft Labels' from teacher can also provide an approximate solution space to effectively train a compact student.

RUL Prediction. In the regression task of machine RUL prediction, deep learning methods have been attracting remarkable attentions due to their superior capability of automatically mapping the input sensory data to the corresponding RUL values. [30] was the first attempt to employ a CNN for RUL prediction. The convolution operations were applied along the temporal dimension over all the input sensor data to learn high-level abstract features. Thereafter, different CNN variants were proposed for RUL prediction, such as NB-CNN [31], Deep-CNN [32], Double-CNN [33], etc. Another popular deep learning architecture of LSTM has also been widely explored for RUL prediction as LSTM can capture the temporal dependency among sequential sensory data. Many works have shown that LSTM-based models can outperform CNN-based models in the task of RUL prediction [5, 6, 7]. However, LSTM-based models often have higher computational complexity and require much more memory compared with CNN-based models. In this paper, we aim to propose a method to bridge the gap between model performance and model complexity.

3. Methodology

In this section, we first give an overview of our proposed CAKD method and then present the details of each component in CAKD.

3.1. Overview of CAKD

The overall structure is illustrated in Figure 1. Specifically, our method transfers the feature representations and the final knowledge from a cumbersome teacher to a compact student with disparate network architectures. It consists of two main steps, namely, 1) feature distillation through adversarial and contrastive learning, and 2) knowledge distillation learning. Both the Teacher (T) and Student (S) models include two modules, i.e., feature extractor and regressor. The feature extractor is to learn a valuable feature map for the final regression task. Let $x \in \mathcal{R}^{k \times l}$ represent the input sample with k input sensors and l time steps, $\psi_T(x)$ and $\psi_S(x)$ represent the feature maps from the feature extractors ψ_T and ψ_S, respectively. As shown in Figure 1(a), both adversarial learning and contrastive learning are designed to learn the student's feature extractor. The extracted feature maps are then fed into the regressors of T and S, and generate the outputs $\hat{\mathbf{y}}_T$, $\hat{\mathbf{y}}_S$ respectively. Here, we consider the teacher output $\hat{\mathbf{y}}_T$ as 'Soft Label', while the actual label (i.e., ground truth) is denoted as $\mathbf{y}_{true}$. Given the soft labels and actual labels, the KD learning part in Figure 1(b) can thus minimize both the soft loss L_{Soft} and the hard loss L_{Hard} to update the student's feature extractor and regressor.

Note that the architectures of teacher and student could be similar or dissimilar. For example, a similar CNN-based structure could be employed for both teacher and student models in an image classification task, except that model depth or weights in the student network are less than those in the teacher network. However, transferring knowledge between dissimilar network architectures are generally considered more challenging than similar network architectures as the latent feature spaces learnt by teacher and student with dissimilar architectures could be totally different. In this work, we employ dissimilar structures for teacher's and student's feature extractors, i.e., a complex LSTM-based structure for teacher and a CNN-based structure for student. For the regressors, we use stacked fully-connected layers for both teacher and student, except that student's regressor has less hidden units.

3.2. Feature Distillation

Intuitively, distilling features between dissimilar network architectures is more challenging since the feature spaces could be totally disparate. As aforementioned, most of previous works focus on pre-defining a decent metric to measure the disparity between teacher's and student's feature spaces [14, 20]. Those metrics have been shown to be effective in related areas, such as image classification and natural language processing. However, it is not clearly stated whether these pre-defined metrics are also feasible for other tasks. Therefore, we are motivated to design an automatic process that can learn such a metric to align the feature maps from the teacher and student.

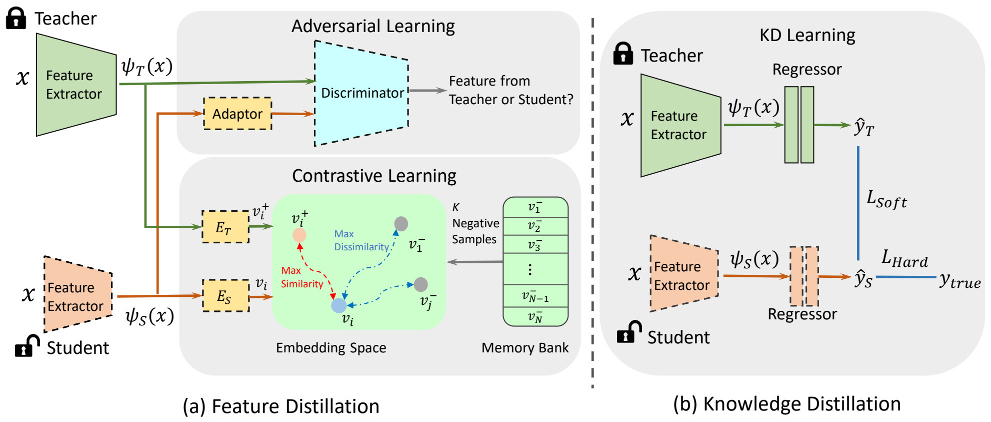

Figure 1: The proposed CAKD method with a two-stage training scheme: (a) feature distillation by adversarial and contrastive learning, and (b) Knowledge distillation Learning.

3.2.1. Adversarial Learning

Inspired by [34], the adversarial learning scheme is exploited in this paper to automatically learn this metric for feature distillation. As depicted in Figure 1(a), a binary classification network *Discriminator*, denoted as F_d, is employed to discriminate whether the input feature map is from the teacher or student network. Considering the fact that the feature maps dimensions of teacher and student generally are different, we add a single-layer linear network (named *Adaptor*) to match them. The adversarial learning can be formulated as follows:

$$\min_{F_d} L_D = -\mathbb{E}_x[\log(F_d(\psi_T(x))) + \log(1 - F_d(f(\psi_S(x))))], \tag{1}$$

$$\min_{\psi_S, f} L_G = \mathbb{E}_x[\log(1 - F_d(f(\psi_S(x))))]. \tag{2}$$

Here, ψ_S and ψ_T represent the feature extractors of the student and teacher networks, respectively. $\psi_S(x)$ and $\psi_T(x)$ represent the feature maps from student's and teacher's feature extractors, respectively. f represents the adaptor and is trained together with ψ_S. Note that the parameters of ψ_T are always fixed during the training stage.

We train ψ_S and F_d in an adversarial manner. First, we fix the parameters of ψ_S and f, and train the discriminator F_d. The goal of training discriminator F_d is to maximize the probability of correctly classifying an input feature maps as 'real' (from teacher) or 'fake' (from student). It consists of two calculation steps for training the discriminator as follows. First, a batch of 'real' samples are constructed from teacher's feature extractor and forwarded pass through F_d. After the loss $\log(F_d(\psi_T(x)))$ is calculated, the gradients are then calculated with a backward pass. Secondly, a batch of 'fake' samples are constructed from the student's feature extractor, which are also forwarded pass through F_d. Then we calculate the loss $\log(1-F_d(f(\psi_S(x)))$ and accumulate the gradients with a backward pass. By minimizing loss Equation (1) with the gradients accumulated from both 'real' and 'fake' batches, we can maximize the probability that F_d correctly classifies $\psi_S(x)$ from student and $\psi_T(x)$ from teacher. Second, we fix the parameters of F_d and then train the ψ_S by minimizing $\log(1 - F_d(f(\psi_S(x)))$ in order to generate better 'fake' feature maps as shown in Equation (2). It allows the student to generate feature maps more like the teacher's such that F_d cannot tell whether they are from ψ_S or ψ_T. By alternately applying above two steps, the discriminator F_d eventually cannot distinguish whether the feature maps are from the teacher or student network. In other words, the student's feature extractor ψ_S can generate features pretty close to the teacher's.

3.2.2. Contrastive Learning

Although the adversarial learning is capable of learning a promising latent metric for feature distillation, there are still some challenges. First, according to Equation (2), the optimization of ψ_S totally depends on the accuracy of the discriminator F_d. The training loss of ψ_S is sometimes difficult to converge, especially in the early training stage. Second, adversarial alignment can only align the overall distribution and overlook the fine-grained features. To mitigate these issues, we employ contrasting learning [22, 35, 36] for instance-wise feature alignment between the teacher and student. Particularly, the contrastive loss aims to maximize the mutual information between teacher and student features which originate from same sample.

As aforementioned, the dimension of feature maps from teacher's and student's feature extractors may be different. Given a sample x_i, we employ a linear Embedded network E to transform $\psi_S(x_i)$ and $\psi_T(x_i)$ as follows so that v_i and v_i^+ have the same dimension.

$$v_i = E_S(\psi_S(x_i)), \tag{3}$$

$$v_i^+ = E_T(\psi_T(x_i)). \tag{4}$$

As shown in Equations (3) and (4), the feature vectors v_i and v_i^+ are from the student and teacher, respectively. E_S and E_T are trainable. For v_i, we consider v_i^+ as its positive vector, while we also derive K negative vectors $\{v_1^-, v_2^-, .., v_K^-\}$ as shown in Figure 1(a). v_i and v_i^+ should have a similar probability distribution if the student can perfectly mimic the teacher, and v_i and v_j^- should have different distributions since they are from different samples. The objective of contrastive learning is to push the vector v_i close to its positive sample v_i^+ and pull it away from those K negative samples.

Following prior work [22], we formulate the posterior probability of two vectors u and v from same data distribution in Equation (5).

$$H(u, v) = \frac{\exp(u^T v)}{\exp(u^T v) + \frac{K}{N}} \tag{5}$$

Here, N is total number of training samples and K negative samples are formulated as a random uniform distribution over total N training samples. Our optimization objective of contrastive learning is thus to maximize the above posterior distribution of the positive and negative samples, which is equivalent to minimize the contrastive loss function L_c defined in Equation (6) [22]. In Equation (6), v

and its positive sample v^+ are from the same distribution P_d, while v and its negative sample v^- are from different distribution P_n. We can then update student's feature extractor ψ_S and Embedded network E by minimizing this contrastive loss.

$$\min_{\psi_S, E_T, E_S} L_c = -\mathbb{E}_{(v^+,v)\sim P_d}[\log(H(v^+, v)] - K * \mathbb{E}_{(v^-,v)\sim P_n}[\log(1 - H(v^-, v))] \tag{6}$$

A memory bank is used to store the embedded feature vectors of all training samples from the previous learning iteration. Let $V = \{v_j\}$ be the memory bank, where $j \in [0, N)$. Suppose that the dimension of embedded feature vector v_j is m, hence the memory bank is a memory buffer with size $N \times m$. During each learning iteration, we use the alias method, i.e., an efficient sampling method with many discrete outcomes introduced in [37], to select K vectors from memory bank V as the negative samples $v_j^- \in \{v_1^-, v_2^-, .., v_K^-\}$. Here, for a specific feature vector v_i, we make sure the indexes of selected K negative vectors are not equal to i.

After each iteration, we update the memory bank with the corresponding entry of each sample with a momentum m. Empirically, we observe that momentum m barely affects the final performance. Hence we set $m = 0.9$ in all the experiments. Another hyper-parameter in contrastive learning is the negative sample size K and we will show its impact on the final results in Section 4.

Finally, we combine both L_G from adversarial learning and L_C from contrastive learning together to train the student's feature extractor ψ_S. The overall loss $L_{G-overall}$ for training student's feature extractor is defined as Equation (7). We will show how model is sensitive to hyper-parameter β in experiments. We set $\beta = 1.0$ in all of our experiments.

$$L_{G-overall} = L_G + \beta * L_C \tag{7}$$

3.3. Knowledge Distillation

Logits from the teacher model contain useful information among classes for a classification task [13]. The 'soften' logits controlled by a 'Temperature' parameter are often used to guide the student's training in a classification task. However, the prediction output of regression tasks is a single value, but not a probability distribution over classes in classification tasks. There are no such logits being soften in regression tasks, hence, we discard the 'Temperature' parameter in Equation (8). But the predictions from teacher are still helpful as it provides an approximate solution space where student can easily get to. Therefore, we consider the predictions/outputs from teacher, $\hat{\mathbf{y}}_T$, as the soft labels to guide the training for student. The KD loss L_{KD} can thus be defined as follows:

$$L_{KD} = \alpha * \|\hat{\mathbf{y}}_S - \hat{\mathbf{y}}_T\|_2 + (1-\alpha) * \|\hat{\mathbf{y}}_S - \mathbf{y}_{true}\|_2 \tag{8}$$

The first term in Equation (8) is the 'soft loss' as stated in Figure 1(b), which measures the L_2 distance between student's predictions $\hat{\mathbf{y}}_S$ and the soft labels $\hat{\mathbf{y}}_T$. The second term is the hard loss, which measures the distance between $\hat{\mathbf{y}}_S$ and the actual labels $\mathbf{y}_{true}$. A hyperparameter α adjusts each term's contribution to the finial loss. Empirically, a higher α often yields a better student as shown in our experiments later, which demonstrates the effectiveness of the soft labels (i.e., teacher's predictions) in assisting student's training.

4. Experiments

In this section, we conduct several experiments to evaluate the effectiveness of our proposed method in the regression task of RUL prediction.

4.1. Experimental Setup

4.1.1. Dataset and Pre-Processing

In this paper, the proposed method is evaluated with the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset [38]. The C-MAPSS dataset contains four sub-datasets with different engine operating conditions and faulty modes. Each sub-dataset can be further divided into training and test data. For training data, each trajectory represents an engine unit with varying initial state and consists of 21 run-to-failure sensor measurements. While the trajectory in test data represents measurements at certain degradation period. The objective of this dataset is to precisely predict the remaining useful life of turbofan engines. The details of each sub-dataset are shown in Table 1.

Table 1: C-MAPSS - NASA's Turbofan Engine Dataset

Dataset	FD001	FD002	FD003	FD004
Train Trajectories	100	260	100	249
Test Trajectories	100	259	100	248
Operating Conditions	1	6	1	6
Fault Modes	1	1	2	2

In our experiments, we further separate the training dataset into training set and validation set with a ratio of 9:1 in terms of the number of engines, and we use the validation set to select model parameters. For example, we randomly select 90 engine trajectories for training and 10 trajectories for validation on FD001 and FD003. Similar to previous works [6, 7], we discard 7 out of 21 sensor measurements (i.e., sensors T2, P2, P15, epr, farB, Nf-dmd and PCNfR-dmd) whose readings remain constant in the data collection process and thus the number of input sensors k is 14. Then, a sliding window method with window size l and step size s is applied to segment the training data as illustrated in Figure 2. For example, the RUL for the first sample is $C - l$, and the $(i+1)^{th}$ sample has a RUL of $C - l - s * i$. Here, C is the total cycle life of an engine. For the test data, we only extract the last segmentation with the same window size to estimate its RUL. The degradation of engine is usually negligible at the beginning stage and linearly increases when engine gets to the end-of-life. Therefore, the piece-wise linear method [39, 6, 7] is employed to label the RUL, where the true RUL is set to the maximal RUL value RUL$_{max}$ if it is larger than RUL$_{max}$. In our experiments, we set window size $l = 30$ (the dimension of input samples $k \times l$ is thus 14×30), step size $s = 1$, RUL$_{max} = 130$, following the previous studies [39, 6, 7].

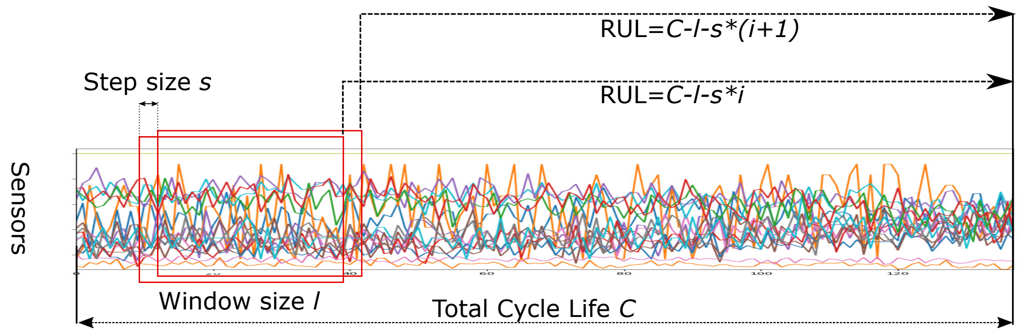

Figure 2: Data Pre-processing

4.1.2. Evaluation Metrics

To quantify the performance of various models, we adopt two commonly used evaluation metrics, i.e., Root Mean Square Error (RMSE) and Score function. Their definitions are shown in Equation (9) and Equation (10), where $d_i = \hat{y}_i - y_i$, $\hat{y}_i$ is the model prediction and y_i is the ground truth for the i^{th} sample. Note that the score function is used to penalize late predictions which may cause worse catastrophe than early predictions in real world.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i)^2} \tag{9}$$

$$\text{Score} = \begin{cases} \sum_{i=1}^{N}(e^{-\frac{d_i}{13}} - 1) & d_i < 0 \\ \sum_{i=1}^{N}(e^{\frac{d_i}{10}} - 1) & d_i \geq 0 \end{cases} \tag{10}$$

4.1.3. Network Architecture

To validate our proposed CAKD approach, we firstly pre-train a powerful but luxurious teacher network with 5 LSTM layers (32 hidden units in each layer) as feature extractor and 2 fully-connected (FC) layers as regressor. With proper parameter tuning, we obtain a teacher with decent performance on each sub-dataset and its model weights are then fixed when guiding the student's training.

For the student's feature extractor, we adopt a dilated CNN structure as [40]. This simple CNN architecture has shown promising capability of dealing with long-range temporal dependencies for time series sensory data. Figure 3 illustrates the details of student's feature extractor and regressor. Note that, *1D CNN(3,2,1)* represents a 1-Dimension convolutional operation with the kernel size of 3, the stride size of 2 and the dilation of 1. We apply different kernel size with different dilation size to ensure that the student's feature extractor has a large receptive field.

The two fully-connected (FC) layers of both teacher's and student's regressors can be denoted as $f_{reg} : \mathbf{R}^{\mathbf{D}} \xrightarrow{FC_1} \mathbf{R}^{\frac{D}{2}} \xrightarrow{FC_2} \mathbf{R}$, where D is the dimension of the flattened feature vector. For example, D is 42 for student's output feature vector as shown in Figure 3. A non-linear activation function (i.e., ReLU) and a dropout layer with dropout rate of 0.5 are added between the two FC layers.

4.2. Comparison with Benchmark Approaches

In this section, we compare our proposed method with various benchmark approaches, including **Standard KD** [13], hint based transfer (**FitNet**) [14], **FitNet-**L_1 [16], activation-based Attention Transfer (**AT**) [21], Probability Knowledge Transfer between intermediate layers of teacher and student (**PKT**) [25], Distance-wise and Angle-wise Relational Knowledge Distillation (**RKD-DA**) [41], Variational Information Transfer between Intermediate layers (**VID-I**) [42], and Deep Mutual Learning (**DML**) [43]. In particular, **FitNet-**L_1 [16] is a variant of FitNet

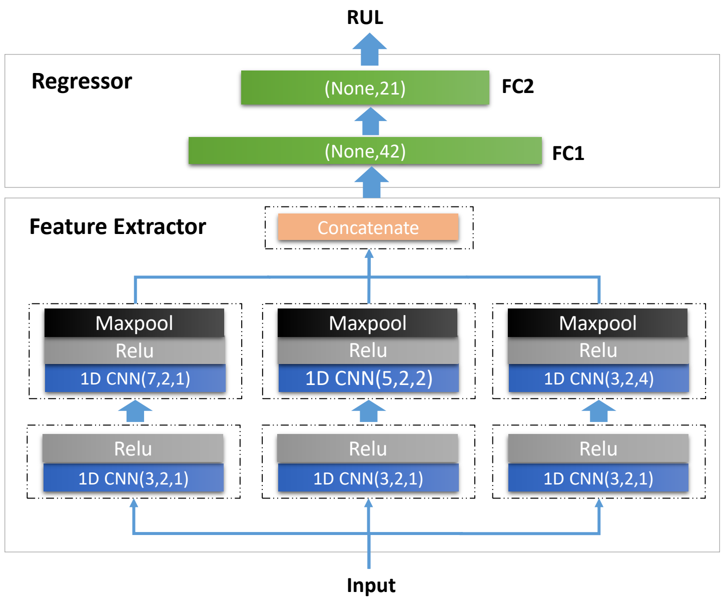

Figure 3: Student Network Architecture

using L_1 to measure feature maps disparity on feature distillation stage. For **RKD-DA** [41], we follow the original paper and set $\lambda_{RKD-D} = 1$ and $\lambda_{RKD-A} = 2$. For **DML** [43], we use 2 networks by following the original paper.

We conducted the experiments with batch size of 64, optimizer of Adam, learning rate of 1e-3 for the proposed method. We adopted a grid search for negative sample size $K \in [2^0, 2^1, \cdots, 2^{10}]$, memory bank updating momentum $m \in [0.1, 0.99]$ and $\alpha \in [0.0, 1.0]$ in Equation (8). Considering the randomness caused by factors like model initialization and dropout, the reported results are averaged over 5 repeats. All experiments and algorithms are implemented with Pytorch framework and the models are trained on a NVIDIA 2080Ti GPU.

Table 2 presents the evaluation results of different methods on the four sub-datasets. The CNN-based student training from scratch (named *Student Only*) performs the worst in terms of RMSE and Score. The teacher model performs much better due to its superior model complexity. By using different KD methods, the performances of the student are improved over all the four datasets, which explicitly indicates the effectiveness of the KD methods on regression tasks. Among all the KD methods, our proposed CAKD approach performs the best. Moreover, it even outperforms the teacher on FD002, FD003 and FD004 in terms of RMSE and

Score. The phenomenons of compact student outperforming cumbersome teacher are also observed in other works [14, 41, 44]. For our proposed CAKD, the possible reason is that the introduction of contrastive learning makes the student able to learn some distinct features from other negative samples, which is not available during teacher's training process.

Another point to note is that combining feature distillation based on Euclidean Distance with KD does not always guarantee better performance than standard KD. For instance, on FD003 and FD004, Fitnet-L_1 and FitNet achieve worse performance than the Standard KD. Besides, transferring specific knowledge, like the attention maps (AT), feature probabilistic distribution (PKT), mutual relations of data samples (RKD-DA), can also help to improve the performance of compact student. It reveals the difficulty on selecting a proper metric for features disparity measurement and also motivates us on adopting adversarial learning to automatically learn a latent metric.

Table 2: Summary of All Experimental Results

Methods	RMSE				Score			
	FD001	FD002	FD003	FD004	FD001	FD002	FD003	FD004
Student Only	15.65	15.88	15.97	17.39	477.73	1404.68	603.55	1809.17
Teacher	13.17	14.47	13.57	16.11	276.39	982.53	349.30	1288.88
Standard KD	15.44	15.57	14.90	16.85	408.71	1130.57	565.58	1361.24
FitNet-L_1	15.06	15.24	15.53	17.12	379.33	1160.58	619.64	1423.88
FitNet	15.00	15.15	15.10	16.99	384.20	1097.92	576.57	1369.45
AT	13.48	14.43	13.23	16.03	304.88	1012.43	366.61	1315.04
PKT	13.57	14.41	13.17	15.94	332.28	996.04	350.86	1291.87
RKD-DA	13.63	14.31	13.19	16.07	341.78	1007.93	354.68	1292.23
VID-I	13.68	14.45	14.46	16.09	333.07	1013.10	477.62	1316.60
DML	14.92	15.26	14.54	16.44	402.12	1191.95	480.21	1331.64
Proposed	**13.41**	**14.23**	**12.95**	**15.85**	**293.82**	**975.96**	**325.29**	**1256.82**

Table 3 compares the teacher and student networks from four perspectives: total number of model parameters, number of Floating-point Operations (FLOPs), memory usage (including model size and extra memory requirement during inference), single sample inference time on edge device. Here, we employ Raspberry Pi 3B+ as the edge device, which has a 64-bit ARMv8 SoC and 1GB RAM. We deploy both the teacher network and the student network learned by our CAKD method on Raspberry Pi 3B+ to compare their performance. The student can achieve a comparable performance with the teacher as shown in Table 2, but reduces 12.8 times model parameters, 46.2 times FLOPs and 5.7 times memory usage as shown in Table 3. Besides, the single sample inference time of the student is 7.5 times faster than that of teacher on the edge device. These results indicate the effectiveness of our proposed method on compressing over-parameterized deep learning models.

Table 3: Model Comparison Between Student and Teacher

	No. of Model Parameters	No. of FLOPs	Memory Usage	Inference Time on Edge
LSTM-based Teacher	115 K	2.4 M	24.92 MB	1.372 s
CNN-based Student	9 K	0.052 M	4.38 MB	0.182 s
Rate	12.8×	46.2×	5.7×	7.5×

Moreover, to verify the effectiveness of dilated-CNN architecture for student model, we further implemented a conventional-CNN [30]. Two scenarios are compared between dilated-CNN and conventional-CNN as shown in Table 4. In Case I, we train both networks from scratch. In Case II, we train both networks using the proposed method with the help of same LSTM-based teacher. Note that we implemented the conventional-CNN according to original paper. Due to different data pre-processing, the performance of self-implemented conventional-CNN which is trained from scratch is better than those reported in [30], and we report the results derived from our implementation for a fair comparison. From Table 4, we can observe that the dilated-CNN performs better than conventional-CNN in both two scenarios. Moreover, the proposed CAKD can also help to improve the conventional-CNN student which implies the effectiveness of the proposed method.

Table 4: Performance Comparison between Dilated-CNN and Conventional-CNN

Scenarios		RMSE				Score			
		FD001	FD002	FD003	FD004	FD001	FD002	FD003	FD004
Case I: Student Only	Conventional-CNN	16.45	16.84	16.61	18.4	595.73	1705.02	756.06	1937.46
	Dilated-CNN	15.65	15.88	15.97	17.39	477.73	1404.68	603.55	1809.17
Case II: CAKD	Conventional-CNN	15.04	15.44	15.04	16.5	411.16	1233.06	528.24	1409.45
	Dilated-CNN	13.41	14.23	12.95	15.85	293.82	975.96	325.29	1256.82

4.3. Model Ablation Study

There are three key components in our CAKD method, i.e., adversarial and contrastive learning in feature distillation (FD), and KD. To investigate the contribution of each component, we derived the following model variants for the ablation study.

- KD only: student only trained with KD.
- Con-FD: student only trained with FD using contrastive learning.
- Adv-FD: student only trained with FD using adversarial learning.
- Con-FD+KD: student trained with contrastive FD and KD.
- Adv-FD+KD: student trained with adversarial FD and KD.
- CAKD: student trained with our proposed method, which combines KD with contrastive and adversarial FD.

Table 5 presents the experimental results of different variant on all the four sub-datasets. It is obvious that comparing with Student Only, all derived variants have consistent performance improvement except for Con-FD on FD001 and FD003. We can find that both contrastive feature learning and adversarial feature learning can further assist KD on improving model performance. Comparing with contrastive learning, adversarial learning is more capable of automatically learning a latent suitable metric to align the feature maps especially when teacher and student have dissimilar network architectures. This is also supported by our results in Table 5 that adversarial learning contributes more than contrastive learning for performance improvement.

Table 5: Effect of Each Component on Model Performance

Methods	RMSE				Score			
	FD001	FD002	FD003	FD004	FD001	FD002	FD003	FD004
Student Only	15.65	15.88	15.97	17.39	477.73	1404.68	603.55	1809.17
KD Only	15.44	15.57	14.90	16.85	408.71	1130.57	565.58	1361.24
Con-FD	16.02	15.4	16.71	16.46	548.9	1351.51	877.57	1394.84
Adv-FD	15.41	15.37	15.01	16.58	418.17	1343.41	575.57	1332.78
Con-FD+KD	14.68	14.85	14.87	16.25	384.08	1068.85	531.9	1330.35
Adv-FD+KD	14.12	14.38	14.63	16.10	375.18	976.16	506.05	1290.06
Proposed CAKD	**13.41**	**14.23**	**12.95**	**15.85**	**293.82**	**975.96**	**325.29**	**1256.82**

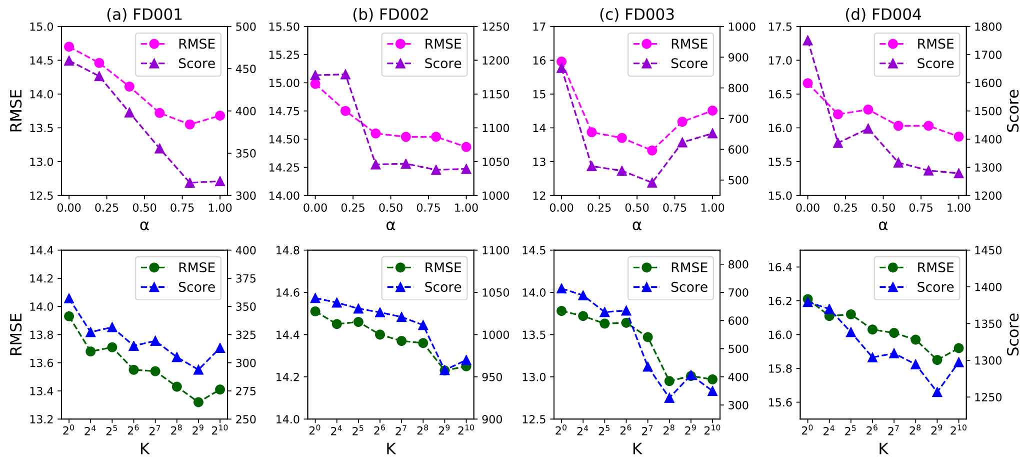

Figure 4: Sensitivity analysis of parameters α and K.

4.4. *Sensitivity Analysis*

There are four important hyper-parameters in our proposed method, which are listed as follows:

- α: smoothing parameter in Equation (8), which controls the contribution of soft-loss and hard-loss.
- K: number of negative samples in contrastive learning.
- β: weight coefficient of contrastive loss in Equation (7).
- m: the momentum of updating the memory bank.

Figure 4 illustrates how parameter α (upper row) and K (lower row) affect model performance in terms of RMSE and Score. It is clear that a higher α often yields better results, indicating teacher's soft labels are more informative. In our experiments, we set $\alpha = 0.8$ for FD001, FD002 and FD004, and $\alpha = 0.6$ for FD003. For the size of negative samples K, intuitively, the higher K values lead to better model performance. However, we observed that the performance starts to decrease on FD001, FD002 and FD004 when K is larger than 512. The reason may be that the student network is too shallow to disparate all the negative samples with the positive sample. Therefore, we set $K = 512$ in our experiments.

Figure 5 shows the impacts of β (upper row) and m (lower row) on model performance in terms of RMSE and Score. It can be found that the RMSE and Score

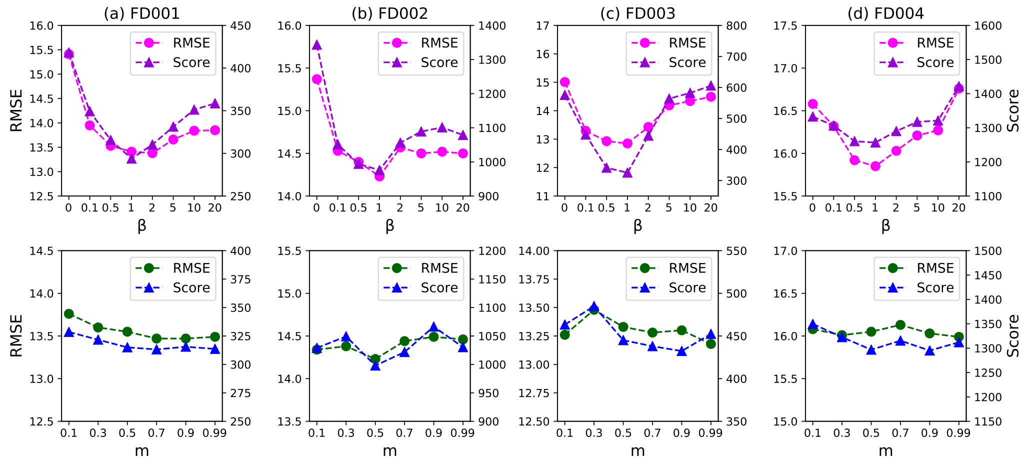

Figure 5: Sensitivity analysis of parameters β and m.

of CAKD start to decrease with the increment of β and achieve the lowest values when β is around 1.0. The performance becomes worse if we further increase β. It is clear that for β, a reasonable value range is $[0.5, 2]$. Therefore, we set $\beta = 1.0$ for all the experiments. Note that $\beta = 0$ is a special case that only using adversarial learning in feature distillation. For the memory bank updating momentum m, it is clear that model performance is not sensitive to this hyper parameter. In all the experiments, we set $m = 0.9$.

5. Conclusion

In this paper, we proposed a contrastive adversarial knowledge distillation (CAKD) method for model compression in a regression task, i.e., machine remaining useful life (RUL) prediction. Specifically, we distilled knowledge from a complex long short-term memory (LSTM) network to an efficient convolutional neural network (CNN) for RUL prediction task. Experiments have been conducted with the popular C-MAPSS dataset which contains four sub-datasets. The results show that the proposed CAKD significantly outperforms conventional KD methods for model compression in the regression task of RUL prediction. By using the proposed CAKD, the student even performs better than the teacher in three of four sub-datasets. This clearly indicates the effectiveness of the proposed method in the regression task of RUL prediction.

References

[1] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2017.

[2] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, “A recurrent neural network based health indicator for remaining useful life prediction of bearings,” *Neurocomputing*, vol. 240, pp. 98–109, 2017.

[3] M. Ragab, Z. Chen, M. Wu, C.-K. Kwoh, R. Yan, and X. Li, “Attention sequence to sequence model for machine remaining useful life prediction,” *arXiv preprint arXiv:2007.09868*, 2020.

[4] L. Ren, Y. Liu, X. Wang, J. Lü, and M. J. Deen, “Cloud-edge based lightweight temporal convolutional networks for remaining useful life prediction in iiot,” *IEEE Internet of Things Journal*, 2020.

[5] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, “Long short-term memory network for remaining useful life estimation,” in *2017 IEEE international conference on prognostics and health management (ICPHM)*. IEEE, 2017, pp. 88–95.

[6] C.-G. Huang, H.-Z. Huang, and Y.-F. Li, “A bidirectional lstm prognostics method under multiple operational conditions,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8792–8802, 2019.

[7] Z. Chen, M. Wu, R. Zhao, F. Guretno, R. Yan, and X. Li, “Machine remaining useful life prediction via an attention based deep learning approach,” *IEEE Transactions on Industrial Electronics*, pp. 1–1, 2020.

[8] S. Srinivas and R. V. Babu, “Data-free parameter pruning for deep neural networks.” *CoRR*, vol. abs/1507.06149, 2015.

[9] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” *arXiv preprint arXiv:1412.6115*, 2014.

[10] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized convolutional neural networks for mobile devices,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4820–4828.

[11] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 1269–1277.

[12] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[14] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *in International Conference on Learning Representations (ICLR)*, 2015.

[15] R. Sharma, S. Biookaghazadeh, B. Li, and M. Zhao, "Are existing knowledge transfer techniques effective for deep learning with edge devices?" in *2018 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2018, pp. 42–49.

[16] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Advances in Neural Information Processing Systems*, 2017, pp. 742–751.

[17] M. R. U. Saputra, P. P. de Gusmao, Y. Almalioglu, A. Markham, and N. Trigoni, "Distilling knowledge from a deep pose regressor network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 263–272.

[18] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, pp. 2654–2662, 2014.

[19] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," *Proceedings of the 35th International Conference on Machine Learning,(ICML)*, vol. 80, pp. 1602–1611, 2018.

[20] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1365–1374.

[21] S. Zagoruyko and N. Komodakis, “Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer,” *5th International Conference on Learning Representations(ICLR)*, 2017.

[22] Y. Tian, D. Krishnan, and P. Isola, “Contrastive representation distillation,” *International Conference on Learning Representations*, 2020.

[23] T. Wang, L. Yuan, X. Zhang, and J. Feng, “Distilling object detectors with fine-grained feature imitation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4933–4942.

[24] J. Yim, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[25] N. Passalis and A. Tefas, “Learning deep representations with probabilistic knowledge transfer,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 268–284.

[26] I. Chung, S. Park, J. Kim, and N. Kwak, “Feature-map-level online adversarial knowledge distillation,” *arXiv preprint arXiv:2002.01775*, 2020.

[27] X. Yao, T. Huang, C. Wu, R.-X. Zhang, and L. Sun, “Adversarial feature alignment: Avoid catastrophic forgetting in incremental task lifelong learning,” *Neural computation*, vol. 31, no. 11, pp. 2266–2291, 2019.

[28] W. C Chen, C.-C. Chang, and C.-R. Lee, “Knowledge distillation with feature maps for image classification,” in *Asian Conference on Computer Vision*. Springer, 2018, pp. 200–215.

[29] L. Gao, H. Mi, B. Zhu, D. Feng, Y. Li, and Y. Peng, “An adversarial feature distillation method for audio classification,” *IEEE Access*, vol. 7, pp. 105 319–105 330, 2019.

[30] G. S. Babu, P. Zhao, and X.-L. Li, “Deep convolutional neural network based regression approach for estimation of remaining useful life,” in *International conference on database systems for advanced applications*. Springer, 2016, pp. 214–228.

[31] F.-C. Chen and M. R. Jahanshahi, “Nb-cnn: Deep learning-based crack detection using convolutional neural network and naïve bayes data fusion,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4392–4400, 2017.

[32] X. Li, Q. Ding, and J.-Q. Sun, “Remaining useful life estimation in prognostics using deep convolution neural networks,” *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[33] B. Yang, R. Liu, and E. Zio, “Remaining useful life prediction based on a double-convolutional neural network architecture,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9521–9530, 2019.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[35] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[36] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” *arXiv preprint arXiv:1906.05849*, 2019.

[37] L. Devroye, “Sample-based non-uniform random variate generation,” in *Proceedings of the 18th conference on Winter simulation*, 1986, pp. 260–265.

[38] A. Saxena, K. Goebel, D. Simon, and N. Eklund, “Damage propagation modeling for aircraft engine run-to-failure simulation,” in *2008 international conference on prognostics and health management*. IEEE, 2008, pp. 1–9.

[39] F. O. Heimes, “Recurrent neural networks for remaining useful life estimation,” in *2008 international conference on prognostics and health management*. IEEE, 2008, pp. 1–6.

[40] Y. Kim, “Convolutional neural networks for sentence classification,” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

[41] W. Park, D. Kim, Y. Lu, and M. Cho, “Relational knowledge distillation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019.

[42] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, “Variational information distillation for knowledge transfer,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9163–9171.

[43] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328.

[44] C. Wang, X. Lan, and Y. Zhang, “Model distillation with knowledge transfer from face classification to alignment and verification,” *arXiv preprint arXiv:1709.02929*, 2017.