Teacher-Class Network: A Neural Network Compression Mechanism

Shaiq Munir Malik[1]
18030012@lums.edu.pk
Mohbat Tharani[2]
mohbaf@rpi.edu
Muhammad Umair Haider[1]
18030031@lums.edu.pk
Muhammad Musab Rasheed[1]
19030008@lums.edu.pk
Murtaza Taj[1]
murtaza.taj@lums.edu.pk

[1] Computer Vision and Graphics Lab
Lahore Univ. of Management Sciences
Lahore, Pakistan
https://cvlab.lums.edu.pk/

[2] Rensselaer Polytechnic Institute
New York, USA

Abstract

To reduce the overwhelming size of Deep Neural Networks, *teacher-student* techniques aim to transfer knowledge from a complex teacher network to a simple student network. We instead propose a novel method called the *teacher-class* network consisting of a single teacher and multiple student networks (class of students). Instead of transferring knowledge to one student only, the proposed method divides learned space into sub-spaces, and each sub-space is learned by a student. Our students are not trained for problem-specific logits; they are trained to mimic knowledge (dense representation) learned by the teacher network; thus, the combined knowledge learned by the *class of students* can be used to solve other problems. The proposed *teacher-class* architecture is evaluated on several benchmark datasets such as MNIST, Fashion MNIST, IMDB Movie Reviews, CIFAR-10, and ImageNet on multiple tasks such as image and sentiment classification. Our approach outperforms the state-of-the-art single student approach in terms of accuracy and computational cost while achieving a $10-30$ times reduction in parameters. Code is available at https://github.com/musab-r/TCN.

1 Introduction

Deep neural networks have effectively tended to various real-world problems, e.g., image classification [1, 12], visual detection and segmentation [22, 43], and audio recognition and analysis [24, 28]. The availability of a large amount of training data, compute power, and improved deep neural network architectures [11, 16, 29, 30] have enabled the deep learning domain to enhance its accuracy continuously. However, these networks have massive parameters and are resource-heavy; hence deploying such networks on resource deficient devices such as mobile phones is almost impractical. Subsequently, compact models with comparable accuracy are critically required.

There have been several efforts to compress these networks, such as efficient architectural blocks (separable convolution [40] and pyramid pooling [38]), pruning layers and filters [2, 5,

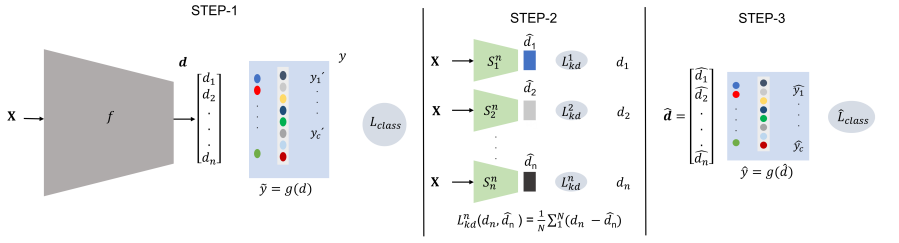


Figure 1: Process overview: The dense feature **d** learned by the teacher network is divided into sub-spaces d_i. Each sub-space is learned by an individual student. Finally, the knowledge from all students is merged and fed to an output layer for the final decision.

[10, 13, 32], quantization [25, 37], and knowledge distillation [9, 15, 18, 26, 36, 41]. Efficient architectural blocks and pruning schemes make the model smaller without any reduction in the complexity of the problem, thus resulting in degraded performance [18]. Similarly, quantization causes loss of data due to approximation resulting in performance drop [37].

Teacher(s)-student architecture [15] uses a huge pre-trained network (teacher) to train a small model (student), so it can learn the knowledge extracted by the teacher that the student otherwise would not be able to learn because of its simpler architecture and fewer number of parameters. This knowledge transfer is achieved by minimizing the loss between the soft labels (probabilities produced by the softmax at a higher temperature [15]) produced by the teacher and the student.

This paper proposes a novel neural network compression methodology called *teacher-class* networks. As compared to existing literature [6, 15, 18, 23, 26, 35, 36], our proposed architecture has two key differences, (i) instead of just one student, the proposed architecture employs multiple students to learn mutually exclusive chunks of the knowledge and (ii) instead of training student on the soft labels (probabilities produced by the softmax) of the teacher, our architecture tries to learn dense feature representations, thus making the solution problem independent. The size of chunks (sub-space) each student has to learn depends on the number of students. Unlike the ensemble-based/multi-branch method [17], all of the students in our proposed approach have been trained independently, thus allowing model parallelism while reducing compute and memory requirements. The knowledge learned by each individual student in our case is combined and output layers are applied. These layers can be borrowed from the teacher network with pre-trained weights and can also be fine-tuned to further improve the loss occurred while transferring the knowledge to students. An overview of the proposed methodology is demonstrated in Fig. 1.

2 Related Work

Knowledge distillation via single student: In knowledge distillation, as introduced by Hinton *et al.* [15], a single student either tries to mimic a single teacher's [6, 15, 17, 18, 19, 21, 23, 26, 35, 36, 44] or multiple teachers [9, 39, 42]. Most of such schemes transfer knowledge to student by minimizing the error between the knowledge of the teacher and the student [15, 18]. Rather than matching actual representation, Passalis *et al.* [26] and Watanabe *et al.* [36] propose to model the knowledge using probability distribution and then match the distribution of teacher and student networks. Nikolaos *et al.* [26] try to cater non-

classification problems in addition to classification problems. Wang *et al.* [35] argue that it is hard to figure out which student architecture is more suitable to quantify the information inherited from teacher networks, so they use generative adversarial network (GAN) to learn student network. Belagiannis *et al.* [3] even studied the distillation of dense features using GANs. Since, teacher can transfer limited amount of knowledge to student, so Mirzadeh *et al.* [23] propose multi-step knowledge distillation, which employs intermediate-sized networks. Peng *et al.* [27] propose a framework named correlation congruence for knowledge distillation (CCKD), which transfers the sample level knowledge, yet in addition, it also transfers the correlation between samples.

Few studies that utilize the dense features along with soft-logits claim that the dense features help to generalize the student model [19, 21, 44]. Heo *et al.* [14] set out a novel feature distillation technique in which the distillation loss is intended to make an alliance among different aspects: teacher transform, student transform, distillation feature position and distance function. An online strategy has also been proposed that eliminates the need for a two-phase strategy and performs joint training of a teacher network as well as a single multi-branch student network [17]. All these methods, including multi-branch strategy [17] train only a single student on the final logits; we instead train multiple students, each on a chunk of dense representation.

Transferring knowledge to multiple student: The only know method that uses multiple students is proposed by You *et al.* [41]. They learned multiple binary classifiers (gated Support Vector Machines) as students from a single teacher which is a multi-class classifier. There are three problems with this approach. Firstly, as the number of classes in the dataset increases, the number of students required would also increase i.e., 1000 students would be required for 1000 class classification problem; secondly, it is applicable only for the classification tasks; thirdly, even after the students have been trained, the output from the teacher network is needed at inference time. To the best of our knowledge no further work has been done in the Single Teacher Multi-Student domain; our proposed approach is the first CNN-based Single Teacher Multi-Student network, which, once trained, becomes teacher independent and has a wide variety of applications including, classification.

3 Methodology

A large state-of-the-art network well trained for a certain problem is considered as a teacher, comparatively, a smaller network is deemed as a student. Unlike [7, 8, 15] that employ a single student to extract knowledge from the teacher's soft logits; our proposed methodology transfers knowledge from dense representation and takes advantage of multiple students.

3.1 Extracting dense representation from the teacher

Neural networks typically produce a dense feature representation $\mathbf{d}$ which, in case of classification, is fed into class-specific neurons called logits, z_i (the inputs to the final softmax). The "softmax" output layer then converts the logit, z_i computed for each class into a probability, $\hat{y}$, defined as:

$$\hat{y} = \frac{exp(z_i/T)}{\sum_j^c exp(z_j/T)}, \tag{1}$$

where c is the total number of classes in the dataset and T is the temperature ($T > 1$ results in a softer probability distribution over classes [15]). Usually, the teacher-student network

Table 1: Knowledge distillation error (L_k) of each student while learning knowledge sub-spaces using only MSE loss and MSE + GAN loss in S^4 configuration.

	MSE				MSE + GAN			
Dataset	S^4_1	S^4_2	S^4_3	S^4_4	S^4_1	S^4_2	S^4_3	S^4_4
MNIST	0.343	0.393	0.395	0.368	0.279	0.312	0.314	0.347
F-MNIST	0.157	0.176	0.173	0.174	0.546	0.704	0.58	0.546
IMDB Movie Reviews	0.004	0.003	0.002	0.002	-	-	-	-

minimizes the cross-entropy between soft targets $\hat{y}$ of the large teacher network and soft targets of the small student network. Since these soft targets $\hat{y}$ being the probabilities produced by the softmax on the logits z_i contain the knowledge only about categorizing inputs into respective classes, learning these soft targets limits the student network to solving a specific problem, making it problem-centric. The general information about the dataset learned by a network is stored in the dense feature representation $\mathbf{d}$ that helps the student to better mimic the teacher and stabilizes the training [3]. It can be observed in a typical transfer learning scenario, where the logit z_i is removed, and only the knowledge in the dense feature layer $\mathbf{d}$ is used to learn a new task by transfer of knowledge from a related task. For example, in the case of VGG-16, the output layer of 1000 class-specific logits is removed and the output of FC2-4096 is used for feature extraction. Similarly, in case of ResNet34 [11] and GoogLeNet [30], FC1-512 and Flattened-1024 are used for feature extraction respectively.

Thus, we redefine the goal of knowledge transfer to that of training a small student model to learn the space spanned by a large pre-trained teacher network. This is achieved by minimizing the reconstruction error between the dense feature vector of the teacher network and the one produced by the student network as shown below:

$$L(\mathbf{d},\hat{\mathbf{d}}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{d}-\hat{\mathbf{d}})^2, \tag{2}$$

where m is the total number of training samples, $\mathbf{d}$ and $\hat{\mathbf{d}}$ are the dense feature representation of teacher and student networks, respectively. The dense representation $\mathbf{d}$ is obtained from a teacher network by extracting the output of the layer before the logits layer. Once student has learned to reconstruct dense representation, the output layer (e.g., class-specific logit and softmax in case of a classification problem) can be introduced to obtain the desired output as in transfer learning. This output layer could be the teacher's output layer with pre-trained weights. The same strategy can be extended to multiple student networks (*teacher-class*) where the dense feature representation $\mathbf{d}$ can then be divided into multiple chunks (sub-spaces); and each sub-space can be learned by an independent student model.

3.2 Learning dense representation using n students

Teacher-student methods [15, 23, 36] attempt to distill knowledge using one student, which becomes cumbersome for a simple network. Multiple students can also be utilized to mimic the teacher's knowledge. A previous such attempt resulted in an ensemble of binary classifiers [41]. In case of 1000 class classification such as ImageNet [16] this will require 1000 student networks making the solution impractical for larger datasets.

Instead, in our case, the dense features (vector space) $\mathbf{d}$ is divided into certain number of sub-spaces, each containing partial knowledge. The vector space could be split into n mutually exclusive sub-spaces or by using standard vector factorization methods such as

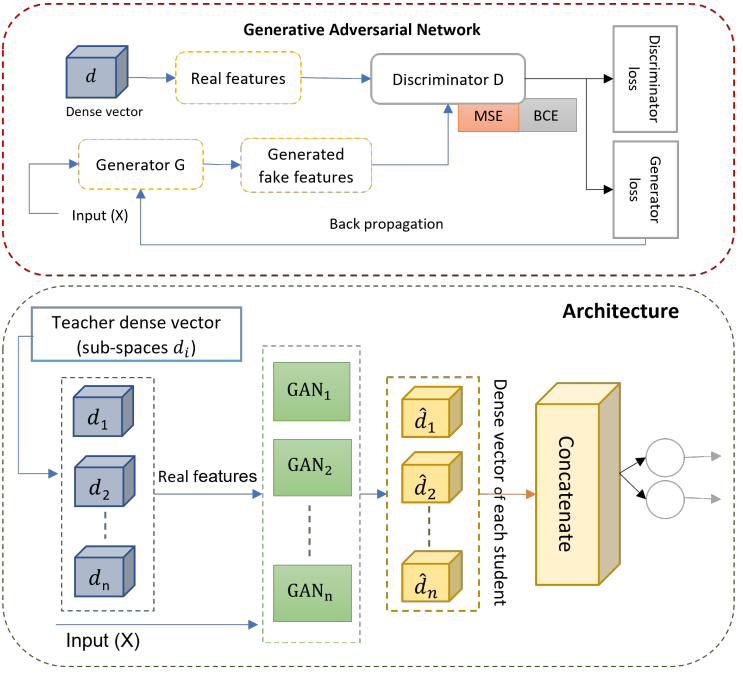


Figure 2: GANs' architecture diagram showing dense vector **d** produced by pre-trained Teacher (T) from input image **X** and fake samples generated by Generator G. The Discriminator D then uses binary cross entropy (BCE) to discriminate between real and fake feature vectors as well as mean squared error (MSE) to ensure reconstruction of dense representations. In multi-student configurations the the dense vector **d** is first divided into sub-spaces and then each student generates its respective sub-space d_i.

singular value decomposition. The latter becomes impractical when the dataset has large numbers of examples. So, we simply split the **d** vector space into n non-overlapping sub-spaces as $\mathbf{d} = [d_1, \mathbf{0}, ...] + [\mathbf{0}, d_2, \mathbf{0}, ...] + ... + [\mathbf{0}, ...d_n]$ where each d_k would be learned independently by the k^{th} student. Let's assume that we have a set of n students such that $S^n = \{S^n_k \mid k \in \mathbb{Z} \wedge 1 \leq k \leq n\}$, where S^n_k is k^{th} student in the set. Mathematically, transferring the knowledge from teacher to n students can be defined as:

$$\hat{d}_k = S^n_k(\mathbf{X}, \theta^k_s), \text{where k} = 1, ..., n \tag{3}$$

where, $\hat{d}_k$ is knowledge distilled by k^{th} student by simply distillation loss or adversarial loss (in case the problem is addressed through adversarial learning). The loss for both the training methods is given in Table 1 for three datasets where different loss for identical students indicates that learning each sub-space converges at different point.

3.3 Combining learned sub-spaces

After all the n students are trained independently, the learned sub-spaces $(\hat{d}_k)$ are merged together to estimate the knowledge $(\hat{\mathbf{d}})$ learned by all n students and is defined as:

$$\hat{\mathbf{d}} = S^n_1(\mathbf{X}) + S^n_2(\mathbf{X}) + ... + S^n_n(\mathbf{X}) \tag{4}$$

Table 2: The impact of fine-tuning output layers (g) on FF in configuration i.e S^4.

Dataset	Without Fine-tuning	With Fine-tuning
MNIST	97.66%	97.81%
Fashion MNIST	85.70%	86.54%
IMDB Movie Reviews	88.29%	88.34%

where $\mathbf{X}$ is input data (e.g. images). Since the students have now collectively learned the space spanned by teacher, so they should ideally give the same results as the teacher network (if solving the same problem the teacher was solving) when fed to the teacher's output layer. Thus, in case of classification, the softmax layer can generate the probability vector as:

$$\hat{y} = g([S_1^n(\mathbf{X}) + S_2^n(\mathbf{X}) + ... + S_n^n(\mathbf{X})], \theta_g) \tag{5}$$

where the function g represents the output layers (softmax in case of classification) applied on concatenation of output from all pre-trained students, $\hat{y}$ is predicted label, and θ_g are its weights which can also be pre-trained weights acquired from the teacher network. The knowledge learned by teacher i.e., $\mathbf{d}$ and n students $\hat{\mathbf{d}}$ might have minor errors (see Table 1). To compensate this error and enhance the overall accuracy of the students, this output layer could be fine-tuned while keeping the students non-trainable. Thus, in case of classification, only last output layer can be optimized using cross-entropy loss function as:

$$L_{class}(y, \hat{y}) = \sum_{i=0}^{N} y_i log(\hat{y}_i). \tag{6}$$

3.4 Mapping as reconstruction problem

Inspired by the success of generative adversarial networks (GANs) for solving sub-optimal problems of learning distributions [4, 7, 8, 35], we pose the dense representation learning as generative adversarial problem. Features from pre-trained Teacher (T) are considered as the real data distribution and they are mimic by student generative networks (G) as fake distribution where discriminator (D) distinguishes between the real and fake features. Unlike GANs based distillation [3, 35], several students are schooled in the adversarial fashion (see Fig. 2). The choice of cross-entropy loss in discriminator provides information only about the real or fake sample based upon where the point lies relative to the decision boundary. However, the distance from the decision boundary would further penalize the generation of dense representation. Therefore, along with adversarial loss, we try to minimize the distance between real and fake samples through MSE loss.

4 Evaluation and Results

We compare the proposed multi-student *(teacher-class)* approach with well-known *teacher-student* architectures [2, 15, 33, 34, 35]. For a fair comparison, we keep the total number of parameters in all students combined equivalent to or less than a student in the *teacher-student* approach for one set of experiments. Therefore, as n increases, students become smaller and simpler.

Table 3: Comparison of knowledge distillation (S^{KD}) [15], and KDGAN [34] with the proposed feed-forward (FF) and GAN based methods in terms of test accuracy reported on two different tasks and four datasets. T is the teacher network, S^n is n student configuration of our proposed approach.

Datasets	T	S^{KD}	KDGAN	S^1		S^2		S^4		S^8	
		[15]	[34]	FF	GAN	FF	GAN	FF	GAN	FF	GAN
MNIST	98.11%	93.34%	99.25%	98.65%	99.16%	96.79%	**99.30%**	97.81%	99.21%	93.97%	98.10%
F-MNIST	91.98%	82.87%	-	89.97%	89.35%	89.43%	**90.74%**	86.54%	89.49%	82.33%	90.67%
CIFAR-10	89.85%	79.99%	86.50	82.17%	81.74%	82.32%	84.70%	81.57%	86.04%	81.91%	**86.96%**
IMDB	86.01%	67.78%	-	**84.58%**	84.48%	83.01%	83.28%	83.29%	83.28%	83.58%	83.67%

4.1 Analysis of student population

We analyzed the effect of increasing the number of students by designing different student architecture using both feed-forward (FF) and GAN-based strategy. In order to clearly study the effect of knowledge distillation via dense vectors, we kept our comparison with vanilla knowledge distillation [15] which uses logits-based cost function. As shown in Table 3, for all four configurations on all datasets, the proposed method achieves accuracy comparable to their teacher network and higher than knowledge distillation (KD) [15]. Overall, GAN-based distillation attains better accuracy than FF due to adversarial and distillation MSE losses. The single student created using vanilla KD approach [15] has $6-19\%$ less accuracy than the teacher, whereas through our approach, using FF, S^1 performs better and using GANs, all S^k configurations have equal or better accuracy on MNIST dataset. For Fashion MNIST (F-MNIST) poor performing student configuration is within 1% of teacher's accuracy. For CIFAR-10, S^{KD} has almost 10% less accuracy than the teachers, whereas our student configuration attain within $2-7\%$ of the teacher's accuracy.

4.2 Identical vs. non-identical students

The convergence of each student at different loss values indicates that all sub-spaces are distinct; therefore, spaces hard to learn may require a better student network. Once all identical students converged, we improved low-performing students. This could be done either by increasing model depth (layers) or width (filters in layers); we followed the prior strategy. As shown in Table 1, for MNIST S^4_2 and S^4_3, for F-MNIST S^4_2, S^4_3, S^4_4, and for IMDB S^4_1 and S^4_2 have relatively higher error. Therefore, by improving these, the student S^4_3 showed better error on MNIST and F-MNIST datasets whereas S^4_1 showed better error on IMDB dataset. Consequently, the error for learning the space by n students combined also improved for all datasets. Overall, enhancing the weaker students ameliorate the performance at the cost of some additional computation due to increased parameters.

4.3 Fine-tuning student networks

As discussed in section 3, once students have been trained, the knowledge learned by all students is combined and fed to an output-specific layer. If the task for students is the same as that of the teacher, then output-specific layers can be borrowed from the teacher network and initialized with pre-trained weights. These layers may or may not need fine-tuning, although it would improve the performance in some cases. To study the effect of using

Table 4: The impact of improving poor performing student in (S^4) configuration in terms of knowledge transfer error (MSE). Dataset with † symbolizes the improved students.

Dataset	S_1^4		S_2^4		S_3^4		S_4^4		$\sum_{i=0}^{4} S_i^4$	
	MSE	#Para	MSE	#Para	MSE	#Para	MSE	#Para	MSE	#Para
MNIST	0.3432	22.17k	0.3930	22.17k	0.3948	22.17k	0.3682	22.17k	1.4994	88.68k
MNIST†	0.3432	22.17k	0.2984	85.74k	0.2976	85.74k	0.3682	22.17k	1.3074	215.82k
F-MNIST	0.1571	22.17k	0.1759	22.17k	0.1728	22.17k	0.1743	22.17k	0.6801	88.68k
F-MNIST†	0.1571	22.17k	0.1293	85.74k	0.1112	85.74k	0.1131	85.74k	0.5107	278.39k
IMDB Movie Reviews	0.0043	165.45k	0.0030	165.45k	0.0024	165.45k	0.0019	165.45k	0.0116	661.6k
IMDB Movie Reviews†	0.0017	331.6k	0.0021	331.6k	0.0024	165.45k	0.0019	165.45k	0.0081	994.1k

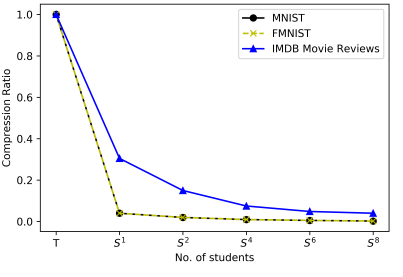
pre-trained layers (g) with or without fine-tuning, experiments were performed on 4 student configurations. From Table 2, it can be observed that for MNIST and F-MNIST, there is an improvement of approximately 1% in test scores. For the IMDB Movie Reviews dataset, it is even less than 1%. This indicates that the dense representation ($\hat{\mathbf{d}}$) produced by all students together was already similar to the teacher's dense representation ($\mathbf{d}$).

Table 5: Comparison of knowledge distillation (S^{KD}) [15] with the proposed method in terms of inference time (in seconds) on slave machines within a network cluster for several configurations of the proposed approach.

Dataset	T	S^{KD}	S_1^2	S_1^4	S_1^6	S_1^8
F-MNIST	0.171s	0.135s	0.096s	0.083s	0.082s	0.075s
IMDB Movie Reviews	0.162s	0.125s	0.056s	0.056s	0.053s	0.051s

Table 6: The comparison of network size (parameters) and FLOPS of Teacher, one student([15]), and several configurations of the proposed method. The graph on the right shows compression of the student with reference to the teacher as we increase the number of students. The values of model size are normalized by the teacher's size.

Config.	MNIST & F-MNIST		IMDB Movie Reviews	
	#Para	FLOPs	#Para	FLOPs
Teacher	2.38M	26.46M	2.21M	7.20M
S^{KD}	94.65k	11.67M	673.80k	2.57M
S^2	95.32k	12.53M	662.03k	2.23M
S_k^2	46.36k	6.26M	330.88k	1.11M
S^4	91.24k	12.54M	662.05k	2.25M
S_k^4	22.17k	3.14M	165.45k	562.11k
S^6	67.84k	9.46M	642.48k	1.49M
S_k^6	10.9k	1.57M	106.8k	249.2k
S^8	41.37k	1.19M	705.3k	913.02k
S_k^8	5.00k	149.12k	88.13k	114.06k



4.4 Computational cost

While designing students, the total number of parameters in the multi-student framework was kept equivalent to one student setup [15], as shown in Table 6. Such as, for MNIST and F-MNIST, the teacher has 11.67M parameters, and one student [15] has 94.65k. While, each student in the 2, 4, 6 student configuration of our approach has 46.36k, 22.17k, 10.9k parameters, respectively. Thus, when 8 student configuration S_1^8 is used, the individual student becomes as small as just 5000 parameters, which makes training a model much easier. Similarly, for the IMDB Movie Review dataset, one student of 673.80k parameters was halved

to 330.88k parameters to design two students and quartered to 165.45k parameters to create four students. Effectively, the decline in model size also reduced FLOPS per student. The graph adjacent to Table 6 demonstrates the normalized model size of a single student concerning the teacher network. Here, the teacher and the student model for MNIST and F-MNIST datasets were the same. It can be observed that as we increase the number of students, the individual student becomes smaller.

4.5 Inference time on distributed cluster

We tested n students on a cluster of virtual machines (VM's) for two datasets where student-n would run on slave-n. The master computer would ask the slave computers to infer a sample of data and send back the results to it. The weights and sample data each student requires for inference were all placed in a shared folder such that they were accessible to all systems within the cluster. The detail of the cluster is discussed in the supplementary material. The average round trip time for our virtual cluster was 0.0365 milliseconds. A teacher and a student of the knowledge distillation approach were executed on a single slave system to compute their inference times. As evidenced in Table 5, our proposed methodology outperforms the teacher and the single student of the KD [15] approach in terms of inference time with the added benefit that students can execute in parallel in a cluster.

4.6 Results on ImageNet

To prove the efficacy of the proposed method, we compare it with seven up-to-date approaches [3, 6, 9, 19, 20, 33, 44] on the ImageNet dataset. We employed ResNet-50 pre-trained as a Teacher, ResNet-18 as one student (FF-S^1), and ResNet-9 (removing the recurring residual block in ResNet-18) in two student setup (FF-S^2). Table 7 depicts that using FF training with only mse loss based distillation. Our method is better/comparable than four of the methods (AEKD [9]), AVER [9], ANC [3], HKSANC [19]) while four methods outperform our approach (Online KD via CL [9], LC KD [20], CRD [33], CRCD [44]). Improved performance of Online KD via CL [9] is due to the use of information fusion in an online manner. Similarly, to supplement distillation loss, LC KD [20] uses local correlation where they apply knowledge distillation on the hidden layers in addition to soft labels. Likewise, CRD [33] and CRCD [44] uses additional data correlation information for knowledge transfer. It should be noted that our approach offers a new strategy for distillation and many advancements proposed over vanilla KD could be employed to further improve each of our students.

Table 7: Comparison with state-of-the-art on ImageNet dataset using ResNet-50 as teacher and ResNet-18 as student.

Method	Baseline/Teacher's Acc.		Student's Acc.	
	Top-1↑	Top-5↑	Top-1↑	Top-5↑
Online KD via CL [9]	77.8%	-	73.1%	-
LC KD [20]	73.27%	91.27%	71.54%	90.30%
AE KD [6]	75.67%	92.50%	67.81%	88.21%
AVER [6]	75.67%	92.50%	67.81%	88.21%
CRD [33]	73.31%	91.42%	71.38%	90.49%
CRCD [44]	73.31%	91.42%	71.96%	90.94%
ANC [3]	72.37%	94.1%	67.11%	88.28%
HKSANC [19]	-	-	68.66%	89.15%
FF-S^1	75.24%	92.19%	68.27%	88.81%
FF-S^2	75.24%	92.19%	66.32%	87.01%

4.7 Ease of training

We divide a large and complex student model into multiple smaller and simpler students and learn them through distillation. Larger models require more data and compute for training, whereas simpler models could be trained in relatively fewer epochs [31]. From Table 8, it is clear that as we increase the number of students, each student model becomes smaller, the convergence is achieved faster and in fewer epochs. Such as using feed-forward (FF) training, the total training time reduces from 80 seconds to 30 seconds in S^2 configuration and 15 seconds in S^4 configuration on the MNIST dataset. Similarly, for Fashion MNIST, there is a decline in total training time while increasing the number of students. In the case of GANs, although there is a reduction in total training time, yet the change is relatively small compared to FF. Because the discriminator model in GANs reduces the effect of model simplicity during training time. Nevertheless, the GANs-based trained students perform better and can compete with FF during inference time. In a nutshell, the distillation of knowledge to multiple students makes each student simple, small, benefits from parallelism, and minimizes the overall training time.

Table 8: Table showing ease of training and benefit of parallelism via multi-student distillation. The training converges in same or fewer epochs, and per epoch computation time as well as total training time reduces.

	Dataset	TCN	Epochs	Time/epoch (sec)	FLOPS	Total time (sec)
FF	MNIST	S^1	20	4	2.9M	80
		S^2	10	3	2.9M	30
		S^4	10	1.5	1.5M	15
	F-MNIST	S^1	10	4	2.9M	40
		S^2	10	3.5	2.9M	35
		S^4	10	3	1.5M	30
GANS	MNIST	S^1	40	14	2.9M	560
		S^2	35	12.5	2.9M	437.5
		S^4	35	12.5	1.5M	437.5
	F-MNIST	S^1	55	13	2.9M	715
		S^2	55	13	2.9M	715
		S^4	50	13	1.5M	650

5 Conclusions

To transfer knowledge from a teacher to a student, we proposed a new method called *teacher-class* network that decomposes the teacher's learned knowledge into sub-spaces, and unlike single teacher single student (STSS) architecture, it employs multiple students to learn the sub-spaces of knowledge. Rather than distilling logits, our method transfers dense feature representation that makes it problem independent, hence it can be applied to different tasks. Since the approach allows to train all students independently, therefore, these student networks can be trained on *CPU* or even on edge devices over the network. Upcoming GPUs that support model parallelism at inference time, such as Groq, Habana Goya, Cerebras Systems, etc are best suited for our architecture. Through extensive evaluation, it has been demonstrated that the proposed method reduces the computational complexity, improves the overall performance, and outperforms the STSS approach.

References

[1] Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 2021.

[2] Ali Alqahtani, Xianghua Xie, Mark W Jones, and Ehab Essa. Pruning CNN filters via quantifying the importance of deep visual representations. *Computer Vision and Image Understanding*, 2021.

[3] Vasileios Belagiannis, Azade Farshad, and Fabio Galasso. Adversarial network compression. In *European Conf. on Computer Vision Workshops*, Sep 2018.

[4] Ting-Yun Chang and Chi-Jen Lu. Tinygan: Distilling biggan for conditional image generation. In *Asian Conf. on Computer Vision*, Nov 2020.

[5] Shi Chen and Qi Zhao. Shallowing deep networks: Layer-wise pruning based on feature representations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2018.

[6] Shangchen Du, Shan You, Xiaojie Li, Jianlong Wu, Fei Wang, Chen Qian, and Changshui Zhang. Agree to disagree: Adaptive ensemble knowledge distillation in gradient space. *Advances in Neural Information Processing Systems*, Dec 2020.

[7] Yinghua Gao, Li Shen, and Shu-Tao Xia. DAG-GAN: Causal structure learning with generative adversarial nets. In *IEEE Conf. on Acoustics, Speech and Signal Processing*, Jun 2021.

[8] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.

[9] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. Online knowledge distillation via collaborative learning. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Jun 2020.

[10] Muhammad Umair Haider and Murtaza Taj. Comprehensive online network pruning via learnable scaling factors. In *IEEE Conf. on Image Processing*, Sep 2021.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Jun 2016.

[12] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Jun 2019.

[13] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Jun 2019.

[14] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *IEEE Conf. on Computer Vision*, Sep 2019.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, Dec 2012.

[17] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In *Advances in Neural Information Processing Systems*, Dec 2018.

[18] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In *European Conf. on Computer Vision*. Springer, Sep 2018.

[19] Peng Li, Chang Shu, Yuan Xie, Yan Qu, and Hui Kong. Hierarchical knowledge squeezed adversarial network compression. In *AAAI Conf. on Artificial Intelligence*, Feb 2020.

[20] Xiaojie Li, Jianlong Wu, et al. Local correlation consistency for knowledge distillation. In *European Conf. on Computer Vision*. Springer, Aug 2020.

[21] Yuang Liu, Wei Zhang, and Jun Wang. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 2020.

[22] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2021.

[23] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI Conf. on Artificial Intelligence*, Feb 2020.

[24] Giovanni Morrone, Daniel Michelsanti, Zheng-Hua Tan, and Jesper Jensen. Audio-visual speech inpainting with deep learning. In *IEEE Conf. on Acoustics, Speech and Signal Processing*, Jun 2021.

[25] Wakana Nogami, Tsutomu Ikegami, Shin ichi Ouchi, Ryousei Takano, and Tomohiro Kudoh. Optimizing weight value quantization for CNN inference. In *IEEE International Joint Conf. on Neural Networks*, Jul 2019.

[26] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *European Conf. on Computer Vision*, Sep 2018.

[27] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *IEEE Conf. on Computer Vision*, Jun 2019.

[28] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 2019.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. on Learning Representations*, May 2015.

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[31] Mingxing Tan and Quoc V Le. EfficientNetV2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*, 2021.

[32] Mohbat Tharani, Tooba Mukhtar, Numan Khurshid, and Murtaza Taj. Dimensionality reduction using discriminative autoencoders for remote sensing image retrieval. In *International Conf. on Image Analysis and Processing*. Springer, Sep 2019.

[33] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conf. on Learning Representations*, Apr 2020.

[34] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. KDGAN: Knowledge distillation with generative adversarial networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., Dec 2018.

[35] Yunhe Wang, Chang Xu, Chao Xu, and Dacheng Tao. Adversarial learning of portable student networks. In *AAAI Conf. on Artificial Intelligence*, Feb 2018.

[36] Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R Hershey. Student-teacher network learning with enhanced features. In *IEEE Conf. on Acoustics, Speech and Signal Processing*, Mar 2017.

[37] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Jun 2016.

[38] Wei Xiang, Hongda Mao, and Vassilis Athitsos. ThunderNet: A turbo unified network for real-time semantic segmentation. In *IEEE Winter Conf. on Applications of Computer Vision*, Jan 2019.

[39] Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, and Daxin Jiang. Model compression with two-stage multi-teacher knowledge distillation for web question answering system. In *ACM International Conf. on Web Search and Data Mining*, Jan 2020.

[40] Byeongheon Yoo, Yongjun Choi, and Heeyoul Choi. Fast depthwise separable convolution for embedded systems. In *Neural Information Processing*. Springer, 2018.

[41] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning with single-teacher multi-student. In *AAAI Conf. on Artificial Intelligence*, Feb 2018.

[42] Zhao You, Dan Su, and Dong Yu. Teach an all-rounder with experts in different domains. In *IEEE Conf. on Acoustics, Speech and Signal Processing*, May 2019.

[43] Dingwen Zhang, Junwei Han, Gong Cheng, and Ming-Hsuan Yang. Weakly supervised object localization and detection: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2021.

[44] Jinguo Zhu, Shixiang Tang, Dapeng Chen, Shijie Yu, Yakun Liu, Mingzhe Rong, Aijun Yang, and Xiaohua Wang. Complementary relation contrastive distillation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Jun 2021.