# Teacher-Class Network: A Neural Network Compression Mechanism

Shaiq Munir Malik$^{1[0000-1111-2222-3333]}$, Mohbat Tharani$^{2,3[0000-0001-9204-6812]}$, and Murtaza Taj$^{3[2222--3333-4444-5555]}$

Computer Vision and Graphics Lab, School of Science and Engineering, Lahore University of Management Sciences, Lahore, Pakistan `18030012, 16060073, murtaza.taj@lums.edu.pk`

**Abstract.** To solve the problem of the overwhelming size of Deep Neural Networks (DNN) several compression schemes have been proposed, one of them is *teacher-student*. *Teacher-student* tries to transfer knowledge from a complex teacher network to a simple student network. In this paper, we propose a novel method called a *teacher-class* network consisting of a single teacher and multiple student networks (i.e. class of students). Instead of transferring knowledge to one student only, the proposed method transfers a chunk of knowledge about the entire solution to each student. Our students are not trained for problem-specific logits, they are trained to mimic knowledge (dense representation) learned by the teacher network. Thus unlike the logits-based single student approach, the combined knowledge learned by *the class of students* can be used to solve other problems as well. These students can be designed to satisfy a given budget, e.g. for comparative purposes we kept the collective parameters of all the students less than or equivalent to that of a single student in *teacher-student* approach . These small student networks are trained independently, making it possible to train and deploy models on memory deficient devices as well as on parallel processing systems such as data centers. The proposed *teacher-class* architecture is evaluated on several benchmark datasets including MNIST, FashionMNIST, IMDB Movie Reviews and CAMVid on multiple tasks including classification, sentiment classification and segmentation. Our approach outperforms the state-of-the-art single student approach in terms of accuracy as well as computational cost and in many cases it achieves an accuracy equivalent to the teacher network while having $10-30$ times fewer parameters.

**Keywords:** Model Compression, Teacher-Student Network, Convolution Neural Networks

## 1 Introduction

The availability of a large amount of training data in combination with very powerful graphics processing units and a series of state-of-the-art deep neural network architectures [6,7,10,18,20] have enabled the deep learning domain to continuously improve its accuracy. However these state-of-the-art networks have huge number of parameters and are resource heavy, hence deploying such networks on resource deficient devices such as mobile phones is almost impractical. For example, over 528MB of memory and over 16 GFLOPs are required by a single forward pass of VGG-16 [20]. Subsequently, compact deep models with comparable accuracy are critically required.

There have been several efforts to compress these networks such as efficient architectural blocks (separable convolution [27] and pyramid pooling [25]), pruning layers and filters [2,4,9,17], quantization [14,24], and knowledge distillation [5,8,16,23,28,29]. Efficient architectural blocks and pruning schemes make the model smaller without any reduction in complexity of the problem thus resulting in degraded performance [8]. Similarly, quantization causes loss of data due to approximation resulting in performance drop [24].

Teacher(s)-student architecture [5] uses a large pre-trained network (teacher) to train a small model (student). The hypothesis is that the student will be able to learn the underlying concepts and knowledge learned by the teacher that the student otherwise wouldn't 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 [5]) 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 [1,5,8,13,16,19,22,23], 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 chunk each student has to learn depends on the number of students. After all of the students have been trained independently, the knowledge learned by each individual student is combined and output layers are applied. These layers can be borrowed from teacher network with pre-trained weights and can also be fine-tuned to further improve the loss occurred while transferring the knowledge to students.

## 2 Related Work

### 2.1 Single Teacher Single Student

Transferring knowledge from one teacher to one student has been widely studied [1,5,8,13,16,22,23]. In Single Teacher Single Student (STSS), one student is encouraged to mimic teacher's knowledge (logits), as introduced by Hinton et. al. [5]. Most of such schemes transfer knowledge to student by minimizing the error between the knowledge extracted from teacher and the knowledge transferred to student [5,8]. Rather than matching actual representation, Passalis et al. and Watanabe et al. [16,23] propose to model the knowledge using probability distribution and then match the distribution of teacher and student networks. Nikolaos et. al. [16] try to cater non-classification problems in addition to classification problems. Yunhe Wang et al. [22] argue that even though the teacher-student networks are able to achieve considerable compression with little or no loss of accuracy, 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. To reduce the dimension of extracted knowledge, Lee et al. [8] applied singular value decomposition on feature maps and [1] enable student to learn low-dimensional space using locality preservation loss while maintaining relationship with high dimensional space (teacher's features). Since, teacher can transfer limited amount of knowledge to student, so Mirzadeh et al. [13] propose multi-step knowledge distillation, which employs intermediate-sized networks.

## 2.2 Multi-Teacher Single Student

Motivated by the learning principle in human education system where one student may learn from different teachers, Multi-Teacher Single Student (MTSS) strategies utilize various domain expert (teachers) to transfer knowledge to a multi-domain single student. Employing several domain experts, Ruder et al. [19] propose to transfer knowledge from many teachers, where each teacher is an expert of one language, to one multi-lingual student, for sentiment classification. Another study that utilizes MTSS to train an all rounder single student achieves promising results on automatic speech recognition [30]. Unlike most of the knowledge distillation approaches,You et al. [28] utilize hints from intermediate layers as well as knowledge from output layers of the teacher network to train a single student network. Dividing the learning process in two stages, [26] first pre-trains student network using STSS scheme and then jointly learns single student from numerous teacher models using multi-teacher paradigm. MTSS increases learning burden for a simple student, consequently, student may not be able to distill complete knowledge from many teachers, resulting loss in overall accuracy.

## 2.3 Single Teacher Multi-Student

Inspired by the way an experienced coach can lead players of a sports team to victory, You et. al. [29] proposed to use multiple binary classifiers as students to solve the multi-class classification problem. They learned multiple gated Support Vector Machines (gSVMs) 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 though 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 sub-domain, our proposed approach is the first CNN based Single Teacher Multi-Student network which once trained becomes teacher independent and has wide variety of applications including classification.

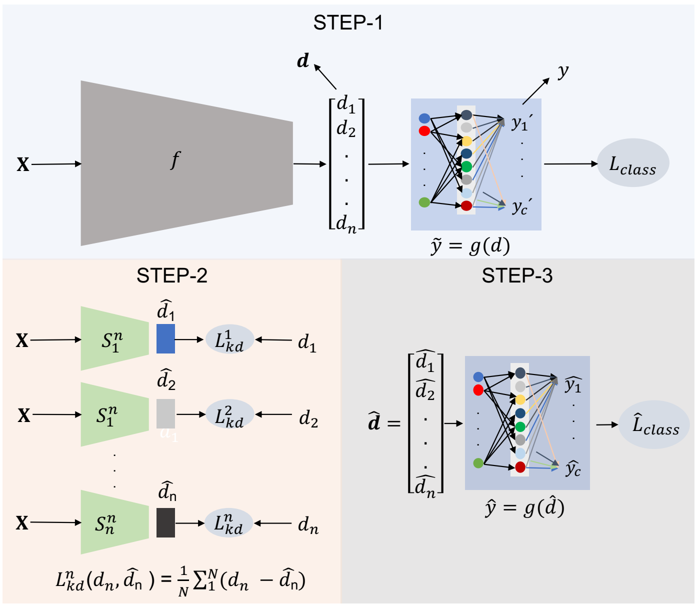


Fig. 1: Overview of the process: The dense feature representation learned by teacher network is divided into chunks. Each chunk is then learned by an individual student, finally the knowledge from all students is merged and fed to an output layer for final decision.

In our proposed approach, Single Teacher Multi-Student (STMS), first, teacher is trained to extract the knowledge, then knowledge is split into chunks and each chunk is transferred to a student. Thus, in the second step, multiple students are trained to be experts on different chunks of the complete knowledge extracted from the teacher. Finally, in third step, the knowledge from all students is merged to mimic the knowledge of teacher which is then fed to output layer to make the final decision. An overview of proposed methodology is demonstrated in Fig. 1.

## 3 Methodolgy

A large state-of-the-art network well trained for a certain problem is considered as a teacher, whereas, comparatively a smaller and simpler network is deemed as a student. The teacher(s)-student [5] approach transfers the knowledge embedded in logits (input to softmax layer) to a single student using soft targets. Unlike this approach that employs a single student to extract knowledge from the teacher's soft logits, our proposed methodology takes advantage of multiple students and transfers dense representation knowledge from teacher(s) to several 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:

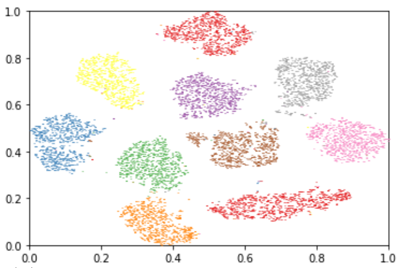
$$\hat{y} = \frac{exp(z_i/T)}{\sum_j^c exp(z_j/T)}, \tag{1}$$

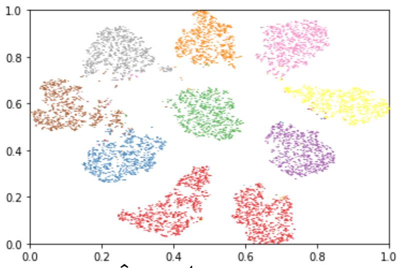
where $c$ is total number of classes in the dataset and $T$ is the temperature ($T > 1$ results in softer probability distribution over classes [5]). Usually, teacher-student network minimize the cross entropy between soft targets $\hat{y}$ of the large teacher network and soft targets of the small student 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 overall information about the dataset learned by the network is stored in the dense feature representation $\mathbf{d}$, this can be observed in a typical transfer learning scenario, where the logit $z_i$ are removed and only the knowledge in dense feature layer $\mathbf{d}$ is used to learn a new task by transfer of knowledge from a related task [15]. For example, in case of VGG-16 [20] the output layer of 1000 class specific logits is removed and the output of rest of the network through FC2-4096 is used for feature extraction. Similarly, in case of ResNet34 [3] and GoogLeNet [21], FC1-512 and Flattened-1024 is used for feature extraction respectively.

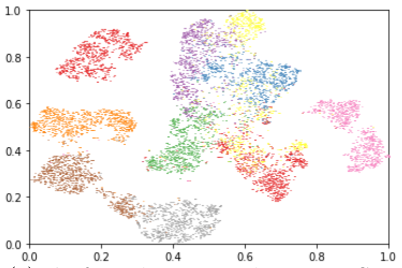
Thus, we redefine the goal of knowledge transfer to that of training a small student model to mimicking the dense feature of a large pre-trained teacher network. In other words the goal is to minimize the reconstruction error between the dense feature vector of the teacher network and the one produced 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 total number of training samples, $\mathbf{d}$ and $\hat{\mathbf{d}}$ is 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 the student has learned to reconstruct dense feature 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 simply 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 and each chunk can be learned by an independent student model as discussed in the next sections.


(a) **d** of teacher on MNIST dataset.


(b) $\hat{\mathbf{d}}$ of $S^4$ on MNIST.


(c) **d** of teacher on Fashion MNIST.

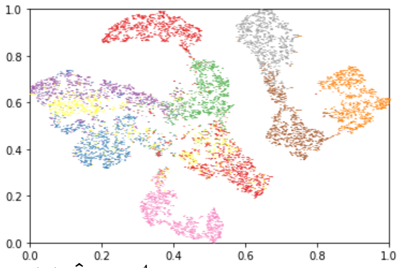

(d) $\hat{\mathbf{d}}$ of $S^4$ on Fashion MNIST.

Fig. 2: t-SNE visualization of dense feature representation learned by teacher and all 4 Students of $S^4$ configuration where first column is dense feature representation of teacher network and second column is combination of dense feature representation learned by 4 students. Row 1 is on MNIST dataset and row 2 is on Fashion MNIST dataset.

### 3.2 Learning dense representation using $n$ students

Teacher-student networks [5,13,23], attempt to distill complete 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 ensemble of binary classifiers [29], as each logit only contains information about a specific class only. In case of 1000 class classification such as ImageNet [7] this will require 1000 such students.

Instead in our case the dense feature vector **d** can be divided into any number of chunks each containing partial knowledge. The knowledge can be divided into chunks either by splitting dense vector into $n$ equal parts or by using standard vector factorization methods such as Singular Value Decomposition. The later becomes impractical when the dataset has large number of examples. So, we simply split the **d** vector into $n$ parts as $\mathbf{d} = [d_1, d_2, ..., d_n]'$ where all $d_k$ can be concatenated to regenerate dense vector **d** and each $d_k$ would be learned independently by the $k^{th}$ student.

Lets assume that we have set of $n$ students such that $S^n = \{S^n_k \mid k \in \mathbb{Z} \land 1 \leq k \leq n\}$, where $S^n$ is set of students with $n$ student configuration and $S^n_k$ is $k^{th}$ student in the set. Mathematically, transferring the knowledge from teacher to

$n$ students can be defined as:

$$\hat{d_k} = S_k^n(X, \theta_s^k), \text{where k} = 1, ..., n \tag{3}$$

where, $\hat{d_k}$ is knowledge distilled by $k^{th}$ student. Note that each student essentially tries to learn a fraction of the dense representation $(d_k)$ which is a real valued vector. Each student learns its chunk of knowledge by minimizing mean square error as:

$$L_{kd}^k(d_k, \hat{d_k}) = \frac{1}{m}\sum_{i=1}^{m}(d_k - \hat{d_k})^2, \tag{4}$$

where, $L_{kd}^k$ is knowledge distillation loss of the $k^{th}$ student and $m$ is total number of training instances. Fig.1 shows how the splits of dense representations are learned by $n$ students by minimizing knowledge distillation loss $(L_{kd})$.

Table 1: Knowledge distillation error $(L_{kd}^k)$ which is mean square error of each student while learning each chunk of knowledge in $S^4$ configuration.

| Dataset | $S_1^4$ | $S_2^4$ | $S_3^4$ | $S_4^4$ |
|---|---|---|---|---|
| MNIST | 0.3432 | 0.3930 | 0.3948 | 0.3682 |
| Fashion MNIST | 0.1571 | 0.1759 | 0.1728 | 0.1743 |
| IMDB Movie Reviews | 0.0043 | 0.0030 | 0.0024 | 0.0019 |

Table 1 shows the chunk reconstruction error for each student in 4 student configuration. It can be seen that on all three datasets (MNIST, Fashion MNIST and IMDB Movie Reviews) each student has converged to a similar error value. Furthermore, these values are between 0.0019 to 0.3948 which indicate successful reconstruction of dense feature vector by the student models.

## 3.3 Combining learned chunks of knowledge

After all the $n$ students have been trained independently, the learned chunks of knowledge or dense vector chunks $(\hat{d_k})$ are concatenated together to estimate the knowledge $(\hat{\mathbf{d}})$ learned by all $n$ students and is defined as:

$$\hat{\mathbf{d}} = [S_1^n(\mathbf{X}), S_2^n(\mathbf{X}), ..., S_n^n(\mathbf{X})]', \tag{5}$$

where $[...]'$ is concatenation and $\mathbf{X}$ is input data such as images or text.

Ideally, the vector $\hat{\mathbf{d}}$ should be similar to the dense representations $\mathbf{d}$ extracted from teacher network and can be used to solve the problem the teacher was solving. The t-SNE visualizations [12,11] of the dense representations from teacher networks and 4 student configuration of our proposed methodology on MNIST and Fashion MNIST datasets are shown in Fig. 2. It can be observed that the class specific clusters for the teacher as well as the student network are well separated in the learned space. This indicates that the dense representations from the combined student network closely mimic the dense representations of the teacher network.

Since, the students have now collectively learned the teacher's dense representations, therefore, 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, function with *softmax* can generate the probability vector as:

$$\hat{y} = g([S_1^n(X), S_2^n(X), ..., S_n^n(X)], \theta_g) \tag{6}$$

where, the function $g$ represents the output layers (class specific logit and softmax) applied on concatenation of output from all pre-trained students and $\theta_g$ 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:

$$L_{class}(y, \hat{y}) = \sum_{i=0}^{N} y_i log(\hat{y}_i) \tag{7}$$

where, $L_{class}(y, \hat{y})$ is classification loss, $\hat{y}$ is predicted labels by the combined student model, $y$ is ground truth label.

Table 2: Comparison of knowledge distillation teacher-student network $S^{KD}$[5] with proposed method in terms of test accuracy reported on different tasks and datasets. $T$ is teacher network, $S^n$ is $n$ student configuration of our proposed approach.

| Dataset | $T$ | $S^{KD}$ [5] | $S^1$ | $S^2$ | $S^4$ | $S^6$ | $S^8$ |
|---|---|---|---|---|---|---|---|
| MNIST | 98.12% | 92.24% | 98.65% | 87.23% | 97.81% | 97.97% | 93.97% |
| Fashion MNIST | 91.84% | 80.87% | 89.97% | 75.45% | 86.54% | 86.92% | 82.33% |
| IMDB Movie Reviews | 88.26% | 49.97% | 88.37% | 88.91% | 88.34% | 88.52% | 88.64% |
| CAMVid | 44.7% | 32.72% | - | 33.4% | - | - | |

## 4 Evaluation and Results

To prove the efficacy of our proposed multi-student *(teacher-class)* approach, we compare it with well known teacher-student architecture [5] that employs only one student to distill knowledge from the bigger teacher network. For a fair comparison we keep the total number of parameters in all students combined equivalent to or less than a single student in the teacher-student approach, therefore, as $n$ increases, students become smaller and simpler. Subsequently, in our case each student network would require much less memory and compute, hence can be trained on *CPU* or even edge device such as Raspberry Pi. The student that has a very similar architecture to the teacher tends to give the best results.

We performed four experiments to analyse the proposed dense feature representation based *teacher-class* methodology. More specifically in experiment 1 we analyzed the effect of increasing the number of students in a class and its implications on the total number of parameters. In experiment 2 we compared the performance of class of students having identical student network with a class having non-identical students. Experiment 3 demonstrates the effect of fine-tuning class/task specific output layer. Finally, we compare the number of parameters and the required number of FLOPs between teacher, single student and each teacher-class configuration.

These experiments were conducted on *MNIST*, *Fashion MNIST*, and *IMDB Movie Reviews* datasets. *MNIST* and *Fashion MNIST* are image classification datasets containing 60000 images each, where each image belongs to one of the 10 classes. *IMDB Movie Reviews* dataset is a sentiment classification dataset containing 25000 movie reviews, each review can be classified as Positive or Negative.

## 4.1 Analysis of student population

The network architectures used in our study are inspired by [5], particularly the student of a single student configuration. We analysed the effect of increasing the number of students by designing five different *teacher-class* architectures for each dataset having 1, 2, 4, 6 and 8 student models. These student configurations have been designed by gradually reducing the number of filters in layers as the number of students in *teacher-class* setup increases i.e. value of $n$ increases. Student models in each of these configurations are designed in such a way that the collective number of parameters for all the students are equivalent to or less than the number parameters of a single student model (i.e. parameters of single student model in the teacher-student approach[5]).

In case of MNIST and Fashion MNIST each student is a 8 layer CNN and in case of IMDB Movie Reviews each student is a 5 layer CNN. The filters in each of the layers decreases as the number of students increases from 1 to 8 to keep within the allocated budget (i.e. parameters of single student model in the teacher-student approach[5]). Table 2 shows the comparison of these multiple student models as well as a single student model used in our approach with the teacher and a single student model used by Hinton et al. [5]. For MNIST dataset, choosing 1, 4 or 6 students show performance similar to the teacher network. However, employing more than 6 students has adverse effects. The single student(Hinton et al.'s approach[5]) achieves accuracy of 92.24% which is almost 6% less than teacher's accuracy, while the single student(our proposed approach) achieves an accuracy of 98.65%. The two student configuration for our approach achieves an accuracy of 87.23%. When 4 students are employed accuracy rises to 97.81% which is 5.57% better than the single student model(Hinton et al.'s approach[5]) and just 0.31% less than the teacher's accuracy. Fashion MNIST dataset follows a very similar trend, using 1, 4 or 6 students gives the best results which is approximately 6% better than the single student model(Hinton et al.'s approach[5]).

For IMDB Movie Reviews, the results obtained for $1, 2, 4, 6$ and 8 students were resembling and hence changing the number of students had no impact on accuracy. Our proposed methodology outperforms the teacher-student [5] approach on all four datasets.

Table 3: Reduction in model parameters from the teacher network when knowledge is learned using one student (teacher-student [5]) and two, four, six and eight students (our proposed approach). The table also shows the required FLOPs (floating point operations) for each network. Here, k = 1000, and M = Million

| Configuration | MNIST | | Fashion MNIST | | IMDB Movie Reviews | |
|---|---|---|---|---|---|---|
| | #Para | FLOPs | #Para | FLOPs | #Para | FLOPs |
| Teacher | 2.38M | 26.46M | 2.38M | 26.46M | 2.21M | 7.20M |
| $S^1$ | 94.65k | 11.67M | 94.65k | 11.67M | 673.80k | 2.57M |
| $S^2$ | 95.32k | 12.53M | 95.32k | 12.53M | 662.03k | 2.23M |
| $S^2_1$ | 46.36k | 6.26M | 46.36k | 6.26M | 330.88k | 1.11M |
| $S^4$ | 91.24k | 12.54M | 91.24k | 12.54M | 662.05k | 2.25M |
| $S^4_1$ | 22.17k | 3.14M | 22.17k | 3.14M | 165.45k | 562.11k |
| $S^6$ | 67.84k | 9.46M | 67.84k | 9.46M | 642.48k | 1.49M |
| $S^6_1$ | 10.9k | 1.57M | 10.9k | 1.57M | 106.8k | 249.2k |
| $S^8$ | 41.37k | 1.19M | 41.37k | 1.19M | 705.3k | 913.02k |
| $S^8_1$ | 5k | 149.12k | 5k | 149.12k | 88.13k | 114.06k |

### 4.2 Identical vs non-identical students

In order to study the effect of students' architecture on overall performance, we employed students with non-identical structure by improving the students with higher mean squared error (see Table 1). Experiments were conducted with 4 students configuration on MNIST, Fashion MNIST, and IMDB Movie Reviews datasets. Note that students were trained to learn chunk of knowledge using mean square error (MSE). Once we have identified the weaker students (ones that have a higher error value) we make these students stronger by increasing their number of parameters, this could be done either by increasing number of filters in existing layers or number of layers in the network. In our case, the students were improved by introducing additional filters in the network. Students $S^4_3$, $S^4_4$, $S^4_2$ had higher error values for Fashion MNIST dataset as can be observed in Table. 4. We increased the number of parameters in each of the three students by

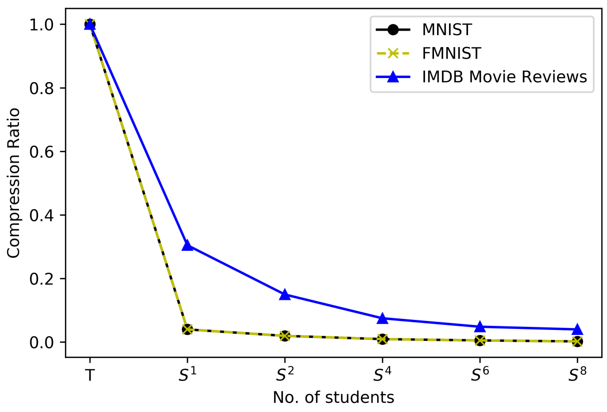


Fig. 3: Compression of student network with respect to teacher as we increase number of students. The values of model size are normalized with respect to teacher's size.

approximately 60k and train them again. As a result, weaker students improved making $S_3^4$ the best of all students. Similarly students $S_2^4$ and $S_3^4$ had higher error values for MNIST, students $S_1^4$ and $S_2^4$ had higher error values for IMDB Movie Reviews as can be observed in Table 4. After increasing the number of parameters in each of these students, we re-trained each one of them. As a result, weaker students improved making $S_3^4$ the best of all students for MNIST and $S_1^4$ the best of all students for IMDB Movie Reviews. Consequently, the MSE for all the students combined improved for all the datsets. Overall, enhancing the weaker students ameliorates the performance at the cost of some additional computation due to increased parameters.

### 4.3 Fine tuning student networks

As discussed in section 3, when knowledge from the dense representation of the teacher has been transferred to the students, the knowledge learned by students is combined and fed to a output specific layer. If you are trying to solve the same problem the teacher was solving, then these layers can be adapted from the teacher network and initialized with pre-trained weights. These layers may or may not need fine-tuning, the fine-tuning would improve the performance in some cases. To study the effect of using pre-trained layers ($g$) with or without fine-tuning, experiments were performed on 4 student configuration. From Table 5, it can observed that for MNIST and Fashion 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 were already very similar to the teacher's dense representation ($\mathbf{d}$), and the students had converged during independent training. You could

Table 4: The knowledge transfer error(mean squared error) of the stronger students in comparison to weaker students when all the students have identical architecture vs when weaker students have been improved. Fashion MNIST$^\dagger$ represents the results after the weaker students have been improved. These results are computed for 4 student configuration ($S^4$).

| Dataset | $S_1^4$ | $S_2^4$ | $S_3^4$ | $S_4^4$ | $\sum_{i=0}^4 S_i^4$ | $\frac{1}{4}\sum_{i=0}^4 S_i^4$ |
|---|---|---|---|---|---|---|
| MNIST | 0.3432 | 0.3930 | 0.3948 | 0.3682 | 1.4994 | 0.3748 |
| MNIST$^\dagger$ | 0.3432 | 0.2984 | 0.2976 | 0.3682 | 1.3074 | 0.3268 |
| Fashion MNIST | 0.1571 | 0.1759 | 0.1728 | 0.1743 | 0.6801 | 0.1700 |
| Fashion MNIST$^\dagger$ | 0.1571 | 0.1293 | 0.1112 | 0.1131 | 0.5107 | 0.1276 |
| IMDB Movie Reviews | 0.0043 | 0.0030 | 0.0024 | 0.0019 | 0.0116 | 0.0029 |
| IMDB Movie Reviews$^\dagger$ | 0.0017 | 0.0021 | 0.0024 | 0.0019 | 0.0081 | 0.0020 |

Table 5: Determining the impact of fine-tuning output layers ($g$) on accuracy. The reported score is for 4 student configuration i.e $S^4$.

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

choose not to fine tune your combined student network and use it directly for inference using the teacher assigned weights, or you could choose to fine tune it using teacher assigned weights. Results for both cases are summarized in Table 5.

## 4.4 Computational Cost

Table 3 and Fig. 3 shows the comparison between teacher-student network [5] and our proposed approach in terms of model sizes, computation cost and respective compression. It can be observed in Table 3 that the single student in the teacher-student approach has 11.67M FLOPs and 94.65k parameters for MNIST and Fashion MNIST while a single student in the 2, 4, 6, 8 student configuration of our approach has 6.26M, 3.14M, 1.57M, 149.12k FLOPs and 46.36k, 22.17k, 10.9k, 5k parameters, respectively. For IMDB Movie Reviews the single student has 2.57M FLOPs and 673.80k parameters, while a single student in the 2, 4, 6, 8 student configuration for our approach has 1.11M, 562.11k, 249.2k, 114.06k FLOPs and 330.88k, 165.45k, 106.8K, 88.13K parameters, respectively. The number of parameters and FLOPs for a single student in the teacher-student approach and our $n$ student configuration are almost similar yet our approach outperforms the teacher-student approach. Figure 3 demonstrates the normalized model size of a single student with respect to teacher network. Here, the teacher and the student model for MNIST and Fashion MNIST dataset were the same. It can be observed that as we increase the number of students, the individual student becomes smaller. The student model in the teacher-student network [5] is almost 4% of the teacher's size for MNIST and Fashion MNIST

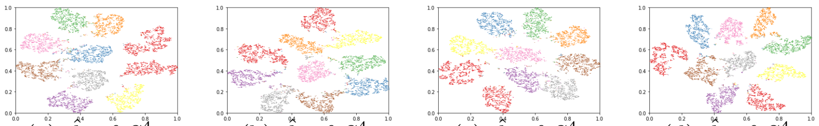

(a) $\hat{d}_1$ of $S_1^4$. (b) $\hat{d}_2$ of $S_2^4$ (c) $\hat{d}_3$ of $S_3^4$ (d) $\hat{d}_4$ of $S_4^4$

Fig. 4: t-SNE visualization of dense feature representations $\hat{d}_k$(where $\hat{d}_k$ is the knowledge distilled by the $k^{th}$ student) from the student networks in $S^4$ configuration on MNIST dataset.

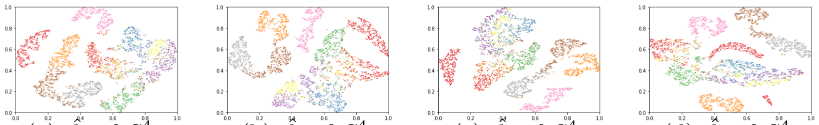

(a) $\hat{d}_1$ of $S_1^4$. (b) $\hat{d}_2$ of $S_2^4$ (c) $\hat{d}_3$ of $S_3^4$ (d) $\hat{d}_4$ of $S_4^4$

Fig. 5: t-SNE visualization of dense feature representations $\hat{d}_k$(where $\hat{d}_k$ is the knowledge distilled by the $k^{th}$ student) from the student networks in $S^4$ configuration on Fashion MNIST dataset.

datasets, and almost 30% of the teacher's size for IMDB Movie Review dataset. For a single student in 2, 4, 6 and 8 student configurations of our approach, the student size reduces to approximately 2%, 1%, 0.5% and 0.2% of the teacher's size for MNIST and Fashion MNIST respectively and approximately 15%, 7.5%, 4.8% and 4% of the teacher's size for IMDB Movie Reviews, respectively. For MNIST and Fashion MNIST datasets, a single student $S^1$ had 94.56 thousand parameters which reduce to almost half the number of parameters in two student configuration. Thus, when 8 student configuration $S_1^8$ is used, the individual student becomes as small as just 5000 parameters, that makes training a model much easier. A similar trend can also be observed with floating point operations (FLOPs).

## 5 Conclusion

To transfer knowledge to student from teacher, this paper proposes a new method called *teacher-class* network that decomposes the knowledge into pieces and unlike single teacher single student (STSS) architecture, it employs multiple students to learn the chunks of knowledge. Rather than distilling logits, our method transfers dense feature representation that makes it problem independent, hence can be applied to different tasks. Since, the method allows to train all students independently, therefore, these student networks can be trained on *CPU* or even edge devices over network. Through extensive evaluation, it has been demonstrated that the proposed method not only reduces the computational complexity, also improves the overall performance and outperforms the STSS approach.

## References

1. Chen, H., Wang, Y., Xu, C., Xu, C., Tao, D.: Learning student networks via feature embedding. IEEE Transactions on Neural Networks and Learning Systems pp. 1–11 (2020)
2. Chen, S., Zhao, Q.: Shallowing deep networks: Layer-wise pruning based on feature representations. IEEE Transactions on Pattern Analysis and Machine Intelligence (12), 3048–3056 (2018)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
4. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4340–4349 (2019)
5. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
6. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708 (2017)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
8. Lee, S.H., Kim, D.H., Song, B.C.: Self-supervised knowledge distillation using singular value decomposition. In: European Conference on Computer Vision (ECCV). pp. 339–354. Springer (2018)
9. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
10. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3431–3440 (2015)
11. Van der Maaten, L., Hinton, G.: Visualizing non-metric similarities in multiple maps. Machine learning (1), 33–55 (2012)
12. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research (Nov), 2579–2605 (2008)
13. Mirzadeh, S.I., Farajtabar, M., Li, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. arXiv preprint arXiv:1902.03393 (2019)
14. Nogami, W., Ikegami, T., ichi Ouchi, S., Takano, R., Kudoh, T.: Optimizing weight value quantization for CNN inference. In: IEEE International Joint Conference on Neural Networks (IJCNN) (Jul 2019)
15. Olivas, E.S., Guerrero, J.D.M., Sober, M.M., Benedito, J.R.M., Lopez, A.J.S.: Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques - 2 Volumes. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA (2009)
16. Passalis, N., Tefas, A.: Learning deep representations with probabilistic knowledge transfer. In: European Conference on Computer Vision ECCV, pp. 283–299. Springer (2018)
17. Qin, Z., Yu, F., Liu, C., Chen, X.: Demystifying neural network filter pruning. arXiv preprint arXiv:1811.02639 (2018)

18. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99 (2015)
19. Ruder, S., Ghaffari, P., Breslin, J.G.: Knowledge adaptation: Teaching to adapt. arXiv preprint arXiv:1702.02052 (2017)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
21. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
22. Wang, Y., Xu, C., Xu, C., Tao, D.: Adversarial learning of portable student networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
23. Watanabe, S., Hori, T., Le Roux, J., Hershey, J.R.: Student-teacher network learning with enhanced features. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5275–5279. IEEE (2017)
24. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2016)
25. Xiang, W., Mao, H., Athitsos, V.: ThunderNet: A turbo unified network for real-time semantic segmentation. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (Jan 2019)
26. Yang, Z., Shou, L., Gong, M., Lin, W., Jiang, D.: Model compression with two-stage multi-teacher knowledge distillation for web question answering system. In: ACM 13th International Conference on Web Search and Data Mining (Jan 2020)
27. Yoo, B., Choi, Y., Choi, H.: Fast depthwise separable convolution for embedded systems. In: Neural Information Processing, pp. 656–665. Springer (2018)
28. You, S., Xu, C., Xu, C., Tao, D.: Learning from multiple teacher networks. In: 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1285–1294. ACM (2017)
29. You, S., Xu, C., Xu, C., Tao, D.: Learning with single-teacher multi-student. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
30. You, Z., Su, D., Yu, D.: Teach an all-rounder with experts in different domains. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (May 2019)