Applied Intelligence (APIN) Journal

Deep Network Compression with Teacher Latent Subspace Learning and LASSO

Oyebade K. Oyedotun[⋆1] · **Abd El Rahman Shabayek**[1] · **Djamila Aouada**[1] · **Björn Ottersten**[1]

Received: date / Accepted: date

Abstract Deep neural networks have been shown to excel in understanding multimedia by using latent representations to learn complex and useful abstractions. However, they remain unpractical for embedded devices due to memory constraints, their high latency and considerable power consumption at runtime. In this paper, we propose the compression of deep models based on learning lower dimensional subspaces from their latent representations while maintaining a minimal loss of performance. We leverage on the premise that deep convolutional neural networks extract many redundant features to learn new subspaces for feature representation. We construct a compressed model by reconstruction from representations captured by an already trained large model. As compared to state-of-the-art, the proposed approach does not rely on labeled data. Moreover, it allows the use of sparsity inducing LASSO parameter penalty to achieve better compression results than when used to train models from scratch. We perform extensive experiments using VGG-16 and wide ResNet models on CIFAR-10, CIFAR-100, MNIST and SVHN datasets. For instance, VGG-16 with 8.96M parameters trained on CIFAR-10 was pruned by 81.03% with only 0.26% generalization performance loss. Correspondingly, the size of the VGG-16 model is reduced from 35MB to 6.72MB to facilitate compact storage. Furthermore, the associated inference time for the same VGG-16 model is reduced from 1.1 secs to 0.6 secs so that inference is accelerated. Particularly, the proposed student models outperform state-of-the-art approaches and the same models trained from scratch.

This work was funded by the National Research Fund (FNR), Luxembourg, under the project reference R-AGR-0424-05-D/Björn Ottersten and CPPP17/IS/11643091/IDform/Aouada

O.K. Oyedotun[1], A. Shabayek[1], D. Aouada[1], and B. Ottersten[1]
[1]Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg
Tel.: (+352) 46 66 44 5949
Fax: (+352) 46 66 44 35949
E-mail: {oyebade.oyedotun, abdelrahman.shabayek, djamila.aouada, bjorn.ottersten}@uni.lu

1 Introduction

Many computer vision tasks work well with features that are learned using deep neural networks (DNNs) of few (i.e. 3-10) layers of latent representations [1; 2]. However, in recent times, analysing more complex multimedia with reasonable accuracies has necessitated that we rely on features learned from deep networks of several layers of latent representations [3; 4; 5; 6] (i.e. over 10 layers), as many works [7; 8; 6] posit the benefit of depth and width for approximating complex target functions. The success of very deep networks on learning hard multimedia tasks has motivated their deployment in various electronic devices. However, memory consumption is a major concern when deploying applications on devices such as smart phones and other portable devices. Furthermore, running large very deep models on portable devices may result in unsatisfactory long inference time for operation. Lastly, such large very deep models would inevitably increase the power consumption of portable electronic devices and subsequently shorten battery life. The different aforementioned problems can work together to limit the deployment of very deep models in portable electronic devices. For example, the popular AlexNet [9] has 61M parameters and is about 250MB in memory size [10]. Model compression aims to reduce the number of model parameters and memory size of large deep models, while incurring minimal loss of model performance. Essentially, given a large model, referred to as *teacher*, that has been trained to an acceptable performance; model compression attempts to obtain from the teacher a small model, referred to as *student*, with minimal performance loss [11].

Knowledge Distillation (KD) has been proposed as a solution to achieve model compression [12; 13; 14] by training a student model to mimic an ensemble [11] or via intermediate-level teacher hidden layers to guide the training process of a single large model known as *FitNets* [15]. Both KD and FitNet employ objective functions that aim to constrain the student model to learn global mapping functions[1] that are similar to that of the teacher model. However, the FitNet is an improvement over KD, as in addition to directly employing the KD training objective, the hint from the middle layer of the Teacher model is used to guide (i.e. constrain) the latent representation in the middle layer of the student model to closely mimic the Teacher model. Nevertheless, we argue that the constraints used for learning both KD and FitNet are *considerably weak* and do not provide a tight constraint on the family of local mapping functions[2] that can be learned for composing the global mapping function of the teacher model. Indeed, one can observe that the different hidden layers of both KD and FitNet still have a large number of family of functions that can be used to approximate the teacher's global mapping function. In addition, the objective functions for both KD and FitNet rely on data true labels for training. In real-life scenarios, availability of labeled data is usually limited; consequently, both KD and FitNet become inapplicable.

In this paper, we address the aforementioned problems of KD and FitNet by viewing model compression as a subspace learning problem. On one hand, the proposed

[1] Mapping from input data space to output (softmax) space

[2] Mapping from a hypothetical hidden layer to the adjoining one

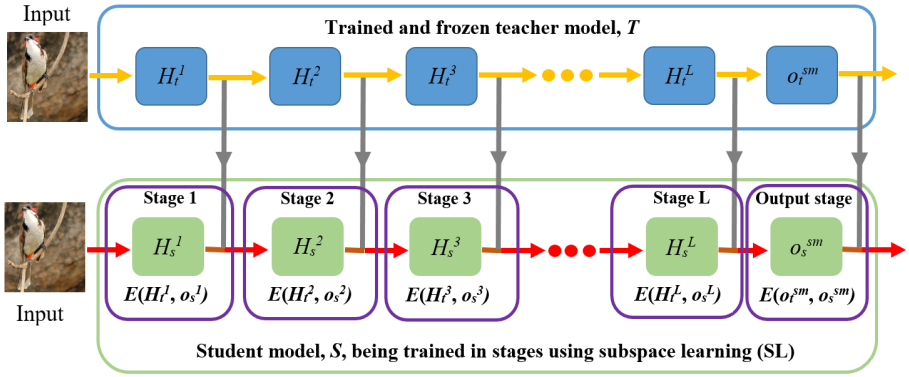


Fig. 1: Overall framework for the proposed subspace learning. It is assumed that the teacher model, T, is already trained to convergence and then its parameters are frozen. The Student model, S, is trained in a layer-wise fashion. In each stage, the hidden layer of the student model, H_s^l, is used to learn a subspace of the corresponding hidden layer of teacher model, H_t^l; a cost function $E(H_t^l, o_s^l)$ that minimizes the reconstruction error of the hidden repersentation of the teacher model H_t^l and the student model's output o_s^l is defined to achieve this. The output of each layer is the variable given in the layer's block. o_t^{sm} and o_s^{sm} represent the softmax outputs of the teacher model and student model, respectively.

subspace learning allows knowledge transfer from a large model to a small model for the purpose of model compression. On the other hand, the same proposal permits small models to mimic the hidden representations of trained large models without using the KD training objective. Our approach imposes a tighter constraint (than KD and FitNet) on the latent representations that the different hidden layers of the student model can learn, given the teacher model. Furthermore, the proposed approach suffices for training a student model with unlabeled data where both KD and FitNet cannot be used. In addition, contrary to unpromising results reported in [16] using sparsity inducing Least Absolute Shrinkage and Selection Operator (LASSO) [17] as a penalty for model pruning, even after model retraining, the proposed latent subspace learning allows LASSO to be used with improved results without the need for retraining. We posit that the end-to-end LASSO optimization of DNNs with complex loss landscapes can impact convergence and solution stability [18].

Particularly, the model compression approach proposed herein is motivated by two well-known important premises about deep networks: (i) they learn many redundant features [19; 20], (ii) they are often over-parameterized [21]. Based on (i) and (ii), one can assume that, with a high probability, there exist feature subspaces that can considerably characterize the target function without much loss of generalization capacity. As such, our contributions in this paper are as follows:

1. A new student model is constructed by learning latent subspaces for the representations captured by an already trained large model. This provides a constraint

on the family of local functions that is learnable by all layers of the student in comparison with KD [11] and FitNet [15].
2. A teacher model is compressible even when available data is unlabeled; where, KD and FitNet are not applicable. In fact, the proposed student model results can be further improved by using additional unlabeled data.
3. LASSO penalty can be incorporated in the proposed student model to improve compression results without retraining as in [16]; this approach outperforms conventional training with the LASSO penalty.

The overall framework of the proposed subspace learning is shown in Fig. 1. This concept is similar in spirit to model pruning where a large model is first trained, and then pruned using some scheme or criteria. Essentially, the idea is that large models have enough parameters such that we have reasonably large search spaces to explore different configurations of solutions for fitting complicated target functions. However, such large models are generally cumbersome so that we are interested in trimming them to obtain compact versions.

Note that this paper is different from KD [11] and FitNet [15] as follows: (i) compression is performed in a layer-wise fashion; KD and FitNet compress all layers jointly, (ii) the teacher and student models are of the same depth; the student is deeper than the teacher in FitNet, and as such can undesirably increase inference time (iii) the student model does not use the KD objective as in FitNet.

The proposed model compression approach has been applied to VGG-16 and wide ResNet (W-ResNet) models using CIFAR-10 [22], CIFAR-100 [22], MNIST [23] and SVHN [24] datasets. We achieve a considerable reduction in the number of model parameters and therefore memory size with minimal loss of performance; its application results in reduced power consumption and inference time. Our extensive experimental results show that the proposed approach outperforms similar compression techniques, namely, KD and FitNets. Furthermore, the proposed approach outperforms several state-of-the-art results [25; 26; 27; 28] for similar models trained from scratch. The rest of this paper is organized as follows.
In Section 2, related works are discussed. Section 3 gives the background on the proposed model compression approach along with the problem statement. Section 4 presents the proposed approach and Section 5 contains experimental results. The paper is concluded in Section 6.

2 Related work

DNNs have found applications in various learning problems such as document representation [29] and traffic flow prediction [30]. However, large DNNs are typically cumbersome so that we can have problems that include large memory for storage, enormous computing resource for operation and high latency during inference. Consequently, several works have addressed model compression. They include parameter pruning [16], quantization [31; 32], low-rank factorization [33; 34; 35], sparse convolution filters [36], and KD [37; 11; 15]. Pruning performs well in removing model parameters that are redundant or not critical to the performance of the model [16]. Generally, all model parameters that fall below a certain threshold are removed from

the model, leading to a notable reduction in their number. We note that pruninng methods that are based on $L1$-norm or $L2$-norm sparsity inducing penalties require significantly higher number of training iterations for successful convergence [38].
The advantage of parameter quantization is the reduction of the required number of bits for representing the individual model parameters [31; 32]. However, the drawbacks of the quantization approach are significant performance loss [38] and specialized algorithms and hardware [39; 40] for running the compressed model.
Since most of the computational bottleneck in deep models usually lies in the convolution filters, low-rank factorization has been proposed in [33; 34] for tensor decomposition that allows compressed models to perform faster inference. For example, [35] used low rank factorization to achieve a $4.5\times$ speedup while incurring only 1% generalization loss on the ILSVRC2012 dataset [41]. Unfortunately, low-rank factorization methods are computationally expensive, and require extensive model retraining after compression [38].
Sparsity constraints have also been employed for model compression. By learning considerably sparse solutions, many model parameters can be removed from the model without a significant loss of generalization capacity. The approach proposed in [36] yielded over 90% of model parameters that were zeros, while incurring a generalization loss of under 1% on the ILSVRC2012 dataset [41]. Furthermore, [42] used regularization penalty to enforce model parameters to have small values, have been explored for building compact and explainable models. In [43], LASSO penalty was used for model selection for random forest models given high-dimensional inputs; good results are reported for the resulting models. However, this approach suffers from inconsistent results; for example, [16] reported unpromising results, and thus employed $l2$-norm penalty instead for enforcing model parameters to have small values. Other works that employed LASSO penalty for enforcing parameter sparsity include feature selection [44], classification of Byopse images [45], temporal effects in a data mining approach [46], regularization of hybrid models that work for falls detection [47], time series forecasting [48] and regularizing fuzzy neural network [49].
One of the closely related works to this paper is KD. The idea is essentially either of the following: (i) compress the knowledge from several models (i.e. ensemble) into a single model [37; 11], or (ii) compress the knowledge from an already trained large model (i.e. teacher) to a small one (i.e. student) [15]. In the ensemble case, the aim is to obtain a single model that mimics the average prediction characteristics of the ensemble. For instance, using automatic speech recognition, the KD proposed approach [11] successfully compressed the knowledge in an ensemble of 10 models into a single model with 0.49% loss in test frame accuracy. In [14], online native ensemble is proposed for learning a single multi-branch network. The work reported improved model generalization over the original models. Furthermore, knowledge distillation has been combined with generative adversarial networks (GAN) for compression [50], where it is argued that direct knowledge distillation results in suboptimal learning of student models. Although interesting results were reported in the work [50], a fair comparison with other compression methods is difficult due to the data augmentation influence of the GAN employed. The proposed FitNets in [15] took inspiration from [11] for model compression. FitNet is essentially KD with intermediate layer guidance for the student model such that the similarity of the repre-

sentations learned by the large and small models is improved.
Herein, we propose a new paradigm for deep learning model compression where the goal is to learn a student model from an already learned teacher model as opposed to training a student model from scratch.

3 Background and problem statement

3.1 Background: KD and FitNet

For distilling knowledge [11; 15] from a single teacher model (T) into a student one (S), the softened softmax outputs of T are incorporated into the target outputs for training S. The goal is to constrain S to mimic the global mapping function of T. Let T be the teacher model with an output softmax probability P_T, S be the student model with parameters W_S and output softmax probability P_S, τ the hyperparameter for softening the logits of the teacher and student models, P_S^τ the softened version of P_S and P_T^τ the softened softmax output of the teacher model. The student model in [11] is trained by optimizing a KD loss function L_{KD} such that:

$$L_{KD}(W_S) = \lambda H(y, P_S) + H(P_T^\tau, P_S^\tau), \tag{1}$$

where H is the cross-entropy loss function, λ is a hyperparameter for controlling the impact of the hard target cross-entropy and y is the ground truth label for the training data.

In FitNet [15], S is *much deeper* than T, and training is achieved using hint, which is the latent representation of T at the middle of the model for guiding a single hidden layer choosen in the student model, S. Note that by construction, S is deeper and thinner than T for the FitNet. The parameters of S from the input (layer 1) up to layer l in the network $W_S^{1:l}$ are first pre-trained by reconstructing the outputs of T at layer l, after which all the parameters of S denoted W_S, are trained using KD objective as in (1). A problem with the FitNet is that choosing the appropriate hidden layer in S to be guided by the hints from T can be tricky in relation to compression results. This can significantly increase the range of experiments required for obtaining decent compression results.

3.2 Problem statement

Let the global mapping functions for T and S be denoted by F_t and F_s, respectively. Also, let the local mapping functions for the different layers be f. Deep networks learn compositional functions, thus using input data x. We can write the global mapping functions for T and S with L hidden layers and softmax layer denoted by sm as follows

$$F_t = f_t^{sm}(f_t^L(...f_t^1(x; W_t^1)...); W_t^L); W_t^{sm}), \tag{2}$$

and

$$F_s = f_s^{sm}(f_s^L(...f_s^1(x; W_s^1)...); W_s^L); W_s^{sm}). \tag{3}$$

First, from (2) and (3), it can be seen that KD in (1) constrains only f_s^{sm} to mimic f_t^{sm}. However, layers in S have a large family of functions to learn local mapping functions from f_s^1 to f_s^L. FitNet attempts to address this problem by constraining f_s^{sm} to mimic f_t^{sm} as in KD and also f_s^j to mimic f_t^j; where j denotes a layer in the middle of S and T. Nevertheless, the other layers in S still have a large family of functions to learn local mapping functions from. As such, it seems that the task of joint knowledge transfer from all the (or several) hidden layers of the teacher model to the student model is somewhat complicated. Subsequently, this may impact generalization capacity of student models. Furthermore, the success of LASSO for DNN compression based on sparsity requires the models to be well optimized so that many parameters can attain *very small values*, and can therefore be removed without hurting model performance. Unfortunately, training DNNs with many layers is challenging due to units' outputs saturation and explosion [51; 52], internal covariate shift [53], and gradients instability [51; 54]. Subsequently, incorporating an additional constraint such as the LASSO penalty can make optimization even more difficult. This problematic scenario using LASSO is reported in [16], where the work settled for the $L2$-norm penalty instead. As such, decomposing the difficult joint optimization of all the hidden layers into a layer-wise optimization scheme would help improve model optimization even with LASSO penalty, since the non-convexity of the optimization problem is greatly reduced. Interestingly, we note that layer-wise training typically improves model optimization and generalization [55; 56].
Second, both KD and FitNet rely on the true data label y for training the student model and therefore limit their applicability.

In this work, the goal is to address the aforementioned compression problems: (i) learn a student model in which local mappings are constrained to follow the local mappings of the teacher model and (ii) a more general scenario in which an already trained teacher model is available, but labeled data is unavailable for training the student model.

4 Proposed latent subspace learning for compression

This section describes the compression of DNNs using the proposed latent subspace learning. The overall compression framework is given in Fig. 2. The main aspects of the framework include training the teacher model to convergence, extracting its latent representations, training the Student model in a layer-wise fashion to reconstruct the latent representations of the Teacher model, and finally fine-tuning the Student model in an end-to-end fashion to align estimated parameters.

4.1 Latent subspace reparameterization

Deep networks typically learn many redundant features [19; 20; 33; 36] and are over-parameterized [21; 36]; therefore there exists, with high likelihood, a latent subspace that approximately captures the same latent representation. As such, we propose to learn subspace features which construct a lower dimensional manifold that has

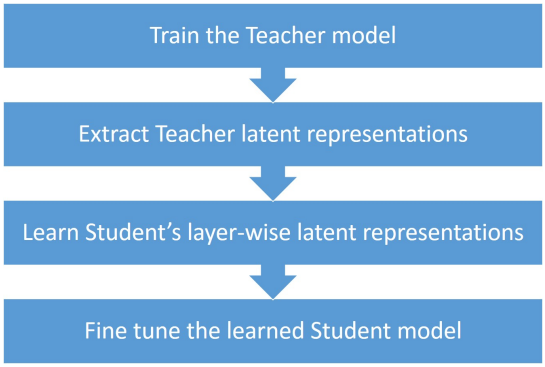


Fig. 2: Overall framework for compressing a Teacher deep model to obtain a Student deep model via subspace learning

learned latent representations of a Teacher model. This has the benefit of a streamlined solution space that the individual layers have to explore. The proposed approach consists in learning the Student model in a layer-wise fashion which decomposes and eases the task of approximating the function learned by the teacher model. Hence, the student model can focus on the solution that approximates the function that is already captured by the teacher model via latent subspace learning. Consequently, this approach provides a tight constraint on the latent representations that S can learn, given T, as supported by our experimental results in Section 5. Furthermore, the proposed layer-wise latent subspace learning (i.e. layer pre-training) eases DNN training so that the incoporation of the LASSO penalty does not plague optimization, since layers are trained one at a time, as opposed to the conventional approach of joint training of all layers [57]. That is, the original task of learning the Student model in an end-to-end fashion is decomposed into a simpler optimization problem by learning the student model in a layer-wise fashion using the latent representations of the Teacher model as intermediate targets. On top of this, layer-wise training is useful for improving model regularization [57; 58], as seen even in recent works [59; 60; 61].

We start by assuming that all the different layers of a deep model are over-parameterized such that we obtain an overall over-parameterized teacher model T. We consider a deep teacher model indexed, t, with latent representation H_t^l at layer l and the number of free parameters $\theta_t^l = |W_t^l|$, where W is the weight matrix. Generally, T has $\theta_t^l \gg 10^p$, where p can range from 2 to 6 such that a large solution space (or solution configurations) can be explored during optimization [20]. This is obvious for deep models which are typically over-parameterized relative to the number of training examples and therefore result in under-determined systems [62]. The objective herein is to learn a deep student model S indexed s from T such that the number of student's free parameters θ_s^l, where $\theta_s^l = |W_s^l|$, is less than their corresponding teacher's free parameters θ_t^l without loss of model generalization performance. The problem of finding a compressed model S is formulated as learning a feature subspace (i.e. lower dimensional manifold) for the latent representations already captured by the teacher model T. The task of compressing via feature subspace learning allows us to learn a

deep student model S such that $\theta_s^l < \theta_t^l$ via reconstruction as summarized in Fig. 2 and detailed below.

The teacher model T is first trained to good convergence using a training dataset $\{x^n, y^n\}_{n=1}^N$. Thereafter, $\{x^n\}_{n=1}^N$ or another unlabeled dataset $\{u^n\}_{n=1}^K$ is used for obtaining the teacher latent representations H_t^l at every layer, where $1 \leq l \leq L$ and L is the depth of the model excluding the output (i.e. softmax) layer.

The student model S is then constructed of the same number of hidden layers as T, but with reduced number of convolution filters or hidden units for every layer; that is $\theta_s^l < \theta_t^l$. Use H_s^{l-1} as the input and task the hidden layer H_s^l to encode the reconstruction of the teacher's latent representation H_t^l in the student model output layer as o_s^l . Note that H_s^1 is obtained by using $\{x^n\}_{n=1}^N$ as the input and reconstructing H_t^1 as the target output. Henceforth, the weights between the input and hidden layer are referred to as the 'encoder weights'. Similarly, the weights between the hidden and output layer of any model are referred to as the 'decoder weights'. The teacher model latent representation reconstruction at layer l using the student model layer l with encoder weights $W_s^{l^{enc}}$, decoder weights $W_s^{l^{dec}}$ encapsulated as W_s^l and activation function φ, can be performed by optimizing the cost function:

$$J(W_s^l) = \underset{W_s^l}{\operatorname{argmin}} \sum_{n=1}^{N} \left(H_t^{l^n} - o_s^{l^n}\right)^2, \tag{4}$$

where,

$$o_s^l = W_s^{l^{dec}} H_s^l = W_s^{l^{dec}} \varphi(W_s^{l^{enc}} H_s^{l-1}). \tag{5}$$

Model parameters $W_s^{l^{dec}}$ and $W_s^{l^{enc}}$ can be updated with block gradient descent algorithm using the following equations.

$$\Delta W_s^{l^{dec}} = -\eta \frac{\partial J(W_s^l)}{\partial W_s^{l^{dec}}}, \tag{6}$$

$$\Delta W_s^{l^{enc}} = -\eta \frac{\partial J(W_s^l)}{\partial W_s^{l^{enc}}}. \tag{7}$$

For the student model softmax layer output o_s^{sm}, we use H_s^L as input and task o_s^{sm} to reconstruct the softmax layer output of the teacher model o_t^{sm} as its output; the incoming and outgoing weights of output layer altogether are denoted W_s^o. Note that the output logits are not softened for both the teacher and student models. The true labels $\{y^n\}_{n=1}^N$ are not used for the input data $\{x^n\}_{n=1}^N$ to learn o_s^{sm}. Rather, the softmax layer outputs o_t^{sm} of the teacher model are used. The encoder weights of W_s^l and W_s^o, $W_s^{l^{enc}}$ and $W_s^{o^{enc}}$, respectively, are saved for the alignment stage.

Furthermore, model compression via sparsity inducing LASSO[3] constraint is employed to improve the level of compression while incurring minimal (or no loss) of model test performance. This is achieved by adding $L1$-norm of model parameters [17] as a penalty term to (4); thus we can write a new training cost function as

$$J(W_s^l) = \underset{W_s^l}{\operatorname{argmin}} \left\{ \sum_{n=1}^{N} \left(H_t^{l^n} - o_s^{l^n}\right)^2 + \lambda \parallel W_s^{l^{enc}} \parallel_1 \right\}, \tag{8}$$

[3] LASSO and $L1$-norm are used interchangably

Algorithm 1: Batch-wise subspace learning and fine-tuning

1. Given an already trained teacher model, T, do (2).
2. Sample H_t^l, the activations of T at layer l, using training data.
3. Initialize the layer weights of the student model S.
4. Using H_s^{l-1} as input and student model layer H_s^l, reconstruct H_t^l in the student's model output layer. At convergence, do (5)
5. Save $W_s^{l^{enc}}$, the encoder parameters of H_s^l.
6. Update $l \longrightarrow l+1$, and do (2) to (5) while $l \leq L$
7. Using H_s^L as input and student model output layer, o_s^{sm}, o_t^{sm} as output, and save encoder weights, $W_s^{o^{enc}}$.
8. Restore $W_s^{l^{enc}}$ and $W_s^{o^{enc}}$, all the individual layer weights of S obtained via subspace learning.
9. Align all student layer weights via fine-tuning.

where λ controls the impact of $L1$-norm penalty on $W_s^{l^{enc}}$. Note that $L1$-norm penalty is not applied to the weights connections from the hidden layers to the output layer.

4.2 Parameters assembly, fine-tuning and pruning

Note that during subspace learning, the different layers are trained without the explicit knowledge of the other layers. Hence, there is a need to perform parameters alignment by assembling all the encoder weights of all the different layers of the student model, and training the whole student model weights altogether again using gradient descent algorithm. The procedure of parameters alignment is referred to as 'fine-tuning' and is described as follows. The student model is assembled for fine-tuning to improve the alignment of the parameters of the different layers towards the global cost function. All the parameters of the student model, W_s, are trained in an end-to-end fashion; that is, using input x^n, target teacher output ${o_t}^n$ and cost function $J(W_s)$, we can minimize the negative conditional loglikehood of the training data defined as follows:

$$J(W_s) = -\underset{W_s}{\operatorname{argmin}} \sum_{n=1}^{N} \log P({o_t}^n | x^n). \tag{9}$$

Models that employ LASSO penalty for subspace learning also employ the same penalty term for parameters fine-tuning. Similar to the subspace learning stage, an $L1$-norm penalty term is added to the conventional cost function in (10); thus the fine-tuning cost function for models that use LASSO penalty can be written as

$$J(W_s) = \underset{W_s}{\operatorname{argmin}} \left\{ -\sum_{n=1}^{N} \log P({o_t}^n | x^n) + \lambda \parallel W_s \parallel_1 \right\}, \tag{10}$$

where notations remain as earlier described. For the sake of conciseness, the proposed compression approach is summarized in Algorithm 1 using a suitable training batch size; LASSO penalty can also be employed.

For student models that are trained using sparsity inducing LASSO penalty, parameters pruning is performed after model fine-tuning to further increase compression level at minimal performance loss. This follows from the fact that depending on the value of λ for $L1$-norm penalty, there are many parameters that are zeros. Also, many other parameters have very small values. These parameters can be pruned (i.e. removed) from the model without significantly impacting model performance. In order to achieve this, a pruning threshold, T_h, is set such that any model parameter, w_j, below the threshold is pruned: $w_j \longrightarrow 0, \because w_j < T_h$. However, in contrast to a similar approach [16], the pruning method in this paper is not iterative; that is, the student model is not retrained after pruning. Later on, we show that the initial subspace learning indeed conditions the parameters of the student model such that good compression results without the need for model retraining after pruning.

For compressing W-ResNet models with skip connections, we consider each ResNet block as a layer. Specifically, each ResNet block in the student model is tasked with reconstructing the output of the corresponding ResNet block in the teacher model.

4.3 Subspace learning, mutual information and convergence

In this section, we show the relationship between the proposed subspace learning and mutual information I maximization similar to auto-encoding. Considering (4) and (5), where H_s^l is used to learn the reconstruction of H_t^l; we can write I between H_s^l and H_t^l as

$$I(H_s^l; H_t^l) = \mathbb{E}\left[\log \frac{P(H_s^l, H_t^l)}{P(H_s^l)P(H_t^l)}\right], \tag{11}$$

where (11) can be written in terms of entropy $\mathbb{H}$ as

$$I(H_s^l; H_t^l) = \mathbb{H}(H_t^l) - \mathbb{H}(H_t^l|H_s^l), \tag{12}$$

and maximizing I in (11) amounts to maximizing

$$\underset{W_s^l}{\operatorname{argmax}}\, I(H_s^l; H_t^l) = -\underset{W_s^l}{\operatorname{argmax}}\, \mathbb{H}(H_t^l|H_s^l). \tag{13}$$

Particularly, observe from (11) that I is zero when H_s^l and H_t^l are independent; this is the case before training in which the weights of the student model are randomly initialized. However, subspace learning aims to maximize I as in (13); after training, H_s^l and H_t^l are highly dependent [63]. In Section 5, experiments that validate this are reported.

Furthermore, DNNs can be hard to train due to numerous local minima that arise owing to their compositional structure. Namely, propagating input data over several layers in the forward pass and backpropagating error gradients via several layers for weights update can make optimization challenging. Interestingly, the proposed subspace learning decomposes learning the student model into stages, where each layer is trained singly and thus eliminates the aforementioned problems associated with joint training of all layers. Again, in Section 5, experiments that support faster convergence as compared to training from scratch are given.

5 Experiments

5.1 General settings

For validating our proposal, we perform extensive experiments using images from benchmarking datasets. In order to demonstrate the effectiveness of the proposed compression approach, we consider two scenarios: (1) Labeled data is available (CIFAR-10, CIFAR-100 and MNIST datasets) for training the teacher and student models as in [11; 15] and (2) An already trained teacher model is available (SVHN dataset), but labeled data is unavailable for training the student model.

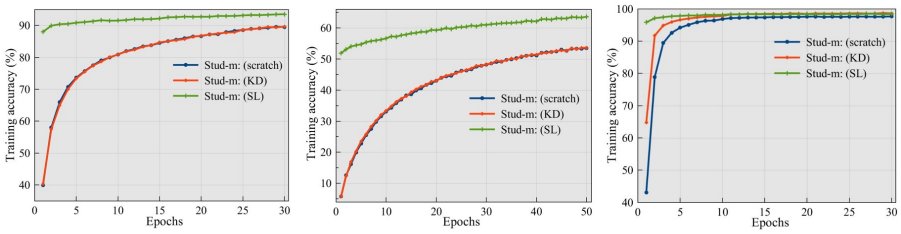
For the teacher model, plain network VGG-16, and W-ResNet-16-8 with skip connections models are used. The architectures of the student models are akin to the teacher model except for the reduction in the number of convolution filters or hidden units. The teacher and student models are trained with mini-batch stochastic gradient descent using the ADAptive Moment estimation (ADAM) optimization technique [64], dropout [65] and batch normalization [66]. For the teacher and student models, their respective initial learning rates are annealed from 10^{-3} to a final value of 10^{-6}; the stagnation of the training loss within a value of 10^{-3} after 5 epochs is used to achieve the annealing. For all the proposed student models reported in this paper, a maximum of 30 epochs are used for the reconstruction of teacher model latent representations. i.e. subspace learning. All the reported teacher and student models have the same depth. The design of the architectures of student models for KD approaches is usually achieved via experimentation [12; 15; 11]. However, we simplify the architectural design of the student models by only reducing the layer width of the teacher models by a constant factor. We experiment with two different compressed student models referred to as stud-s ("s" denotes small) and stud-m ("m" denotes medium) that are obtained by reducing the number of filters (i.e. for convolution layers) and units (i.e. for fully connected layers) in every layer of the teacher model by one-half and one-quarter, respectively[4].

In the first scenario, the teacher model is trained, after which the proposed approach is employed for compressing the knowledge already acquired into the student models. To further improve compression results based on number of model parameters and memory size (in Megabytes: MB), experiments on student models trained with $L1$-norm are performed as discussed in Sections 4.1 and 4.2. Different pruning threshold values of T_h in the range 10^{-5} to 5×10^{-3} are used for training. Particularly, detailed result metrics after pruning such as test accuracy, % (percentage) loss of test accuracy, new number of model parameters, memory size and % (percentage) reduction in memory size are reported. All the results for comparison are obtained after fine-tuning. For control experiments and results comparison, the students are trained from scratch on the same dataset with and without $L1$-norm penalty. In addition, knowledge distillation is employed for training the student models.

[4] Codes will be made publicly available upon paper acceptance

Table 1: Compression results on CIFAR-10 dataset

Models	Param.	P%	Sz(MB)	Acc.(%)
VGG-16				
Stud-s: (scratch)	0.56M	93.75	2.20	88.72
Stud-m: (scratch)	1.7M	81.03	6.72	91.90
Stud-s: (KD)	0.56M	93.75	2.20	88.93
Stud-m: (KD)	1.7M	81.03	6.72	92.06
Stud-s: (SL)	**0.56M**	**93.75**	**2.20**	**91.72**
Stud-m: (SL)	**1.7M**	**81.03**	**6.72**	**93.55**
Pruning [67] in [68]	2.6M	82.7	9.90	91.25
Pruning [67] in [68]	1.08M	92.8	4.30	86.85
L2PF [68]	2.6M	82.7	9.90	92.15
L2PF [68]	1.08M	92.8	4.30	89.85
Slimming [69]	2.26M	88.7	9.22	92.27
LSTM pruning [69]	2.26M	88.7	9.22	93.30
Fitnet [15]	2.50M	72.20	9.90	91.61
W-ResNet-16-8				
Stud-s: (scratch)	0.7M	93.64	2.83	93.37
Stud-m: (scratch)	2.7M	75.45	10.87	94.89
Stud-s: (KD)	0.7M	93.64	2.83	93.76
Stud-m: (KD)	2.7M	75.45	10.87	94.74
Stud-s: (SL)	**0.7M**	**93.64**	**2.83**	**94.35**
Stud-m: (SL)	**2.7M**	**75.45**	**10.87**	**95.21**
Pruning [67] in [68]	3.53M	67.9	13.86	88.73
Pruning [67] in [68]	2.38M	78.4	9.37	74.23
L2PF [68]	3.53M	67.9	13.86	93.53
L2PF [68]	2.38M	78.4	9.37	92.33



(a) CIFAR-10 dataset (b) CIFAR-100 dataset (c) MNIST dataset

Fig. 3: Compressed VGG models convergence rate for the first few epochs

5.2 Results and discussion

5.2.1 CIFAR-10

The CIFAR-10 dataset contains natural color images that belong to 10 different classes. There are 50,000 and 10,000 training and testing images, respectively. Also, we perform standard data augmentation as in horizontal flipping, translation by 4 pixels and border reflection. The teacher and student models are trained for a maximum of 250 epochs. The student models are fine-tuned for 250 epochs after initial subspace learn-

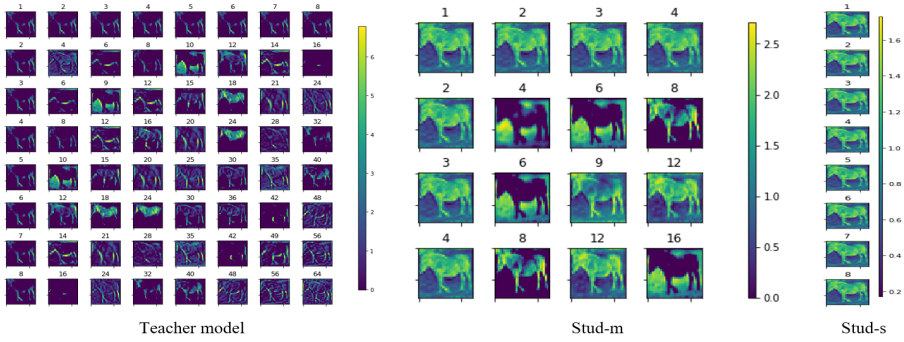


Fig. 4: Convolution maps activations for the first layer of VGG-16 models trained on CIFAR-10 dataset; models have been simulated using an image from the 'horse' class. Compression enforces increased units activities in stud-m and stud-s relative to the teacher model

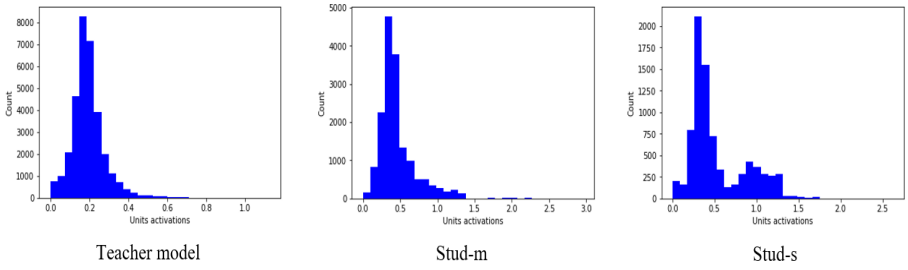


Fig. 5: Average units' activations over CIFAR-10 dataset; the third convolution layer of trained VGG-16 models are shown. Most units in the teacher models operate with small activations; most units' activations in stud-m and stud-s have been redistributed to have larger values to compensate for compression

ing. The VGG-16 teacher model has 8.96M parameters (param.) of size 35MB, and achieves a test accuracy of 93.79%. W-ResNet-16-8 has 11M param. of size 43MB, and achieves a test accuracy of 95.23%. The results of comparing the compressed and trained student models are given in Table 1, where Param., P%, Sz(MB) and Acc.(%) denote the number of model parameters after compression, percentage of pruned parameters, model size and test accuracy. Also, scratch, SL and N.A. denote models trained from scratch, using the proposed Subspace Learning and Not Applicable, respectively. Fig. 3a shows that the proposed VGG stud-m model trained with subspace learning (i.e. stud-m (SL)) is close to the final solution than when the same models are trained from scratch or with KD, since layerwise subspace learning can explore the reduction of model non-convexity to estimate initial model parameters that are close to one of the optima solutions for the end-to-end fine-tuning phase. During fine-tuning, stud-m with SL starts with a training accuracy of 88%. Using the VGG model, it is observed that before fine-tuning, stud-s and stud-m already yield

Table 2: Compression results for proposed stud-m model trained with LASSO and then pruned using different T_h (pruning threshold value) values on CIFAR-10 dataset; P% is in reference to the VGG teacher model used for Table 1

T_h ($\times 10^{-3}$)	0.01	0.05	0.1	0.5	1	5
Proposed: Acc.(%)	92.82	92.82	92.83	**92.75**	92.79	92.32
Proposed: Param.	0.78M	0.74M	0.7M	**0.56M**	0.48M	0.28M
Proposed: P%	91.29	91.74	92.19	**93.75**	94.64	96.88
Proposed: Sz(MB)	3.04	–	–	**2.20**	–	1.17
Stud-m (scratch): Acc.(%)	90.95	–	–	**89.85**	–	89.68

*Baseline (proposed stud.-m without LASSO): Param. = 1.7M, P%=81.03, 6.72MB and Acc.(%)=92.98

test accuracies of 87.94% and 91.81%, respectively; this validates that local mapping functions are indeed learned during subspace learning. Fig. 4 shows the convolution maps for the first layer of the teacher, stud-m and stud-s models. We observe that compression enforces the hidden units of stud-m and stud-s to be more active (i.e. attain higher values) as feature detectors than the units in the teacher model. To further validate this, we report in Fig. 5 the distribution of average activations of the hidden units over the entire dataset, where increased units activitions allow the model to compensate for the pruned units (i.e. compression), and thus ameliorate performance loss; the work [16] made a similar observation on the weights of pruned models. From Table 1, it is seen for both VGG-16 and W-ResNet that the student models obtained using the proposed approach outperform the same models trained from scratch and those obtained using knowledge distillation and FitNet. In addition, the test errors for deep models from earlier works with comparable depth and parameters as the student models trained in this paper are given.

Table 2 reports results on compression results for the proposed stud-m, using VGG-16 as the teacher model, trained based on latent subspace learning with a $L1$-norm penalty (λ) of 5×10^{-5} and then pruned using different T_h values. It is seen that further compression after pruning is achieved with minimal loss of model generalization performance, especially compared to student models with the same number of model parameters that are trained from scratch. For example, with 0.56M parameters, the proposed student model achieves a test accuracy of 92.75%, while a similar student model with $L1$-norm trained from scratch achieves only 89.85% test accuracy.

5.2.2 CIFAR-100

The CIFAR-100 dataset contains natural colour images belong to 100 different classes. There are 50,000 and 10,000 training and testing images, respectively. Also, we perform data augmentation similar to the CIFAR-10 dataset. The VGG-16 teacher model has 8.96M parameters (param.) of size 35MB, and achieves a test accuracy of 72.32%. W-ResNet-16-8 has 11M param. of size 43MB, and achieves a test accuracy of 78.53%. Fig. 3b shows that stud-m trained using subspace learning (i.e. stud-m (SL)) is again closer to the final solution than when training from scratch or when using KD; during fine-tuning, stud-m with SL starts with a training accuracy of 52%.

Table 3: Compression results on CIFAR-100 dataset

Models	Param.	P%	Sz(MB)	Acc.(%)
VGG-16				
Stud-s: (scratch)	0.56M	93.75	2.20	63.55
Stud-m: (scratch)	1.70M	81.03	6.72	67.19
Stud-s: (KD)	0.56M	93.75	2.20	64.50
Stud-m: (KD)	1.70M	81.03	6.72	67.64
Stud-s: (SL)	**0.56M**	**93.75**	**2.20**	**68.83**
Stud-m: (SL)	**1.70M**	**81.03**	**6.72**	**70.07**
Slimming [69]	3.4M	83.00	13.47	69.41
LSTM pruning [69]	5.98M	70.10	23.41	73.26
Fitnet [15]	2.50M	72.20	9.90	64.96
W-ResNet-16-8				
Stud-s: (scratch)	0.70M	93.64	2.83	71.73
Stud-m: (scratch)	2.70M	75.45	10.87	75.88
Stud-s: (KD)	0.7M	93.64	2.83	71.76
Stud-m: (KD)	2.7M	75.45	10.87	75.97
Stud-s: (SL)	**0.70M**	**93.64**	**2.83**	**73.04**
Stud-m: (SL)	**2.70M**	**75.45**	**10.87**	**77.12**

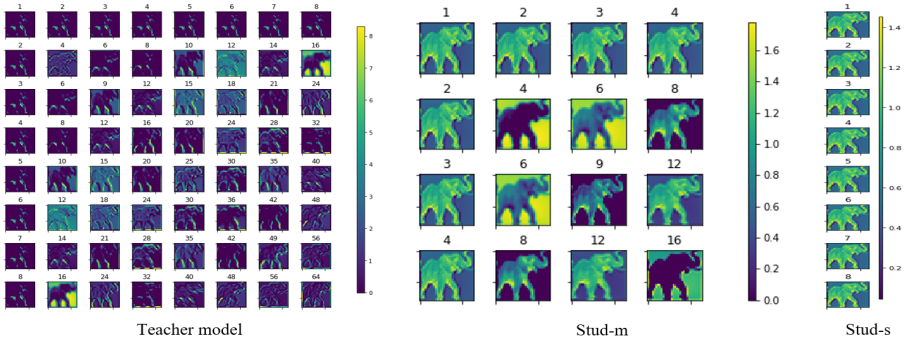
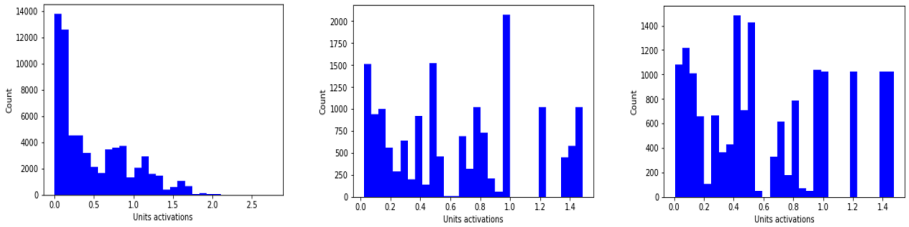


Fig. 6: Convolution maps activations for the first layer of VGG-16 models trained on CIFAR-100 dataset; models have been simulated using an image from the 'elephant' class. It is observed that compression imposes increased units activities in stud-m and stud-s relative to the teacher model

The results of comparing the compressed and trained student models are given in Table 3 as the test classification accuracy.

Fig. 6 shows the convolution maps for the first layer of the teacher, stud-m and stud-s models. We observe that compression enforces the hidden units of stud-m and stud-s to be more active (i.e. attain higher values) as feature detectors than the units in the teacher model. To further validate this, we report in Fig. 7 the distribution of average activations of the hidden units overall the entire dataset. From Table 3, it is seen that the student models obtained using the proposed approach outperforms the same models trained from scratch and those obtained using knowledge distillation. Again, it is seen that for VGG-16 and W-ResNet architectures, the proposed student models



Teacher model Stud-m Stud-s

Fig. 7: Average units' activations over CIFAR-100 dataset; the first convolution layer of trained VGG-16 models are shown. Most units in the teacher models operate with small activations; most units' activations in stud-m and stud-s have been redistributed to have larger values to compensate for compression

Table 4: Compression results for proposed stud-m model trained with LASSO and then pruned using different T_h (pruning threshold value) values on CIFAR-100 dataset; P% is in reference to the VGG teacher model used for Table 3

T_h ($\times 10^{-3}$)	0.01	0.05	0.1	0.5	1	5
Proposed: Acc.(%)	71.52	71.54	71.52	71.59	71.58	70.03
Proposed: Param.	0.95M	0.94M	0.93M	**0.86M**	0.8M	0.54M
Proposed: P%	89.40	89.51	89.62	**90.40**	91.07	93.97
Proposed: Sz(MB)	3.80	–	–	**3.50**	–	2.20
Stud-m (scratch): Acc.(%)	68.92	–	–	**68.22**	–	64.34

*Baseline (proposed stud.-m without LASSO): Param. = 1.7M, P%=81.03, 6.77MB and Acc.(%)=71.31

yield better results as compared with student models trained from scratch, KD and FitNet. In addition, the test errors for deep models from earlier works with comparable depth as the student models trained in this paper are given.

Again, using VGG-16 as the teacher model, compression results for the proposed stud-m trained based on latent subspace learning with a $L1$-norm penalty (λ) of 5×10^{-5} and then pruned using different T_h values are given in Table 4. It is seen that further compression after pruning is achieved with *improved* model generalization performance, especially compared to student models with the same number of model parameters that are trained from scratch. For example, with 0.86M parameters, the proposed student model achieves a test accuracy of 71.59%, while a similar student model with $L1$-norm trained from scratch achieves only 68.22% test accuracy. We observe that compression also allows the model to perform faster inference; this is an extremely important attribute for processing multimedia such as photos in real (or near real) time. Fig. 8 shows the inference time for compressed VGG and W-ResNet models on CIFAR-10 and CIFAR-100 test sets. The direct implication is that with minimal loss of generalization performance (i.e. Tables 1 & 3), model latency for operation can be reduced. The results in Fig. 8 are obtained using a workstation with 32GB of RAM, i7 processor and Nvidia GTX 1080Ti GPU.

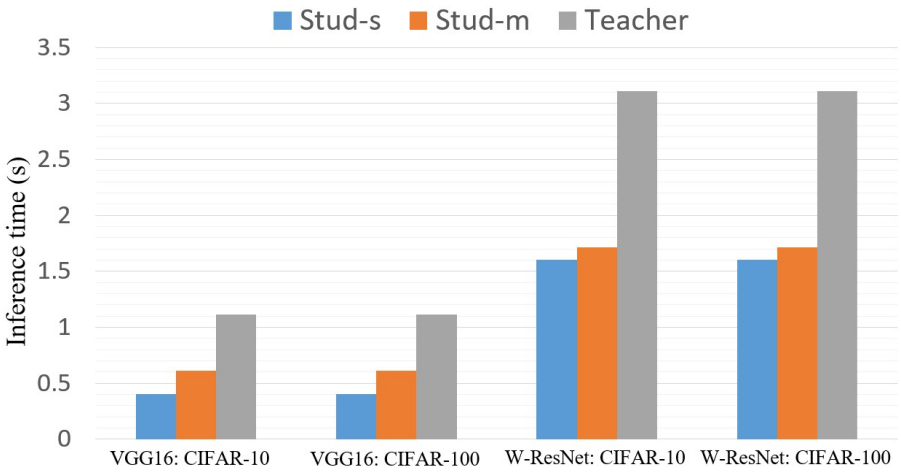


Fig. 8: Model inference time for test set of dataset

Table 5: Compression results on MNIST dataset

Models	Param.	P%	Sz(MB)	Acc.(%)
Stud-s: (scratch)	8.4K	98.50	0.12	99.11
Stud-m: (scratch)	29K	94.82	0.20	99.34
Stud-s: (KD)	8.4K	98.50	0.12	99.20
Stud-m: (KD)	29K	94.82	0.20	99.49
Stud-s: (SL)	**8.4K**	**98.50**	**0.12**	**99.30**
Stud-m: (SL)	**29K**	**94.82**	**0.20**	**99.54**
Fitnet [15]	30K	91.69	0.26	99.49
Highway nets. [5]	39K	N.A.	0.30	99.43

5.2.3 MNIST

The MNIST dataset contains binary images of handwritten digits 0-9. There are 50,000 and 10,000 test samples in the dataset. For this dataset, we train a VGG-like teacher model with 560K parameters on the MNIST dataset with no data augmentation. W-ResNet is not trained on MNIST dataset which is considered simpler to learn than CIFAR and SHVN datasets. The VGG-16 teacher model has 560K param. of size 2.32MB, and achieves a test accuracy of 99.61%. Subsequently, two student VGG-like models with 8.4K (stud-s) and 29K (stud-m) parameters are constructed and trained using the proposed compression approach. Again, Fig. 3c shows that stud-m trained using subspace learning (i.e. stud-m (SL)) converges to a solution that is in the proximity of the final solution than when training from scratch or using KD; during fine-tuning, stud-m with SL starts with a training accuracy of 96%. It is observed that before fine-tuning, stud-s and stud-m have test accuracies of 98.36% and 99.30%, respectively; this once again shows that local mapping functions are indeed learned by the student models.

Table 5 shows the experimental results, along with other results from models of

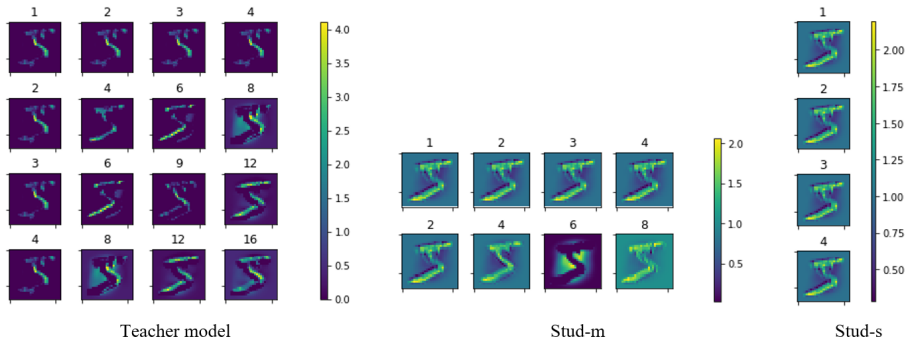


Fig. 9: Convolution maps activations for the first layer of models trained on MNIST dataset; models have been simulated using an image from the 'five' class. Compression enforces increased units activities in stud-m and stud-s relative to the teacher model

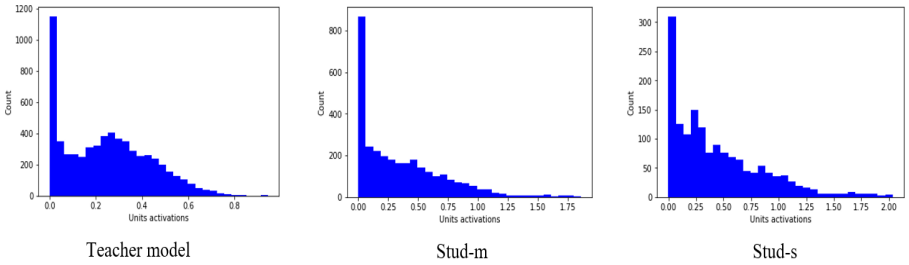


Fig. 10: Average units' activations over MNIST dataset; the first convolution layer of trained models are shown. It is observed that most units in the teacher models operate with small activations, while most units' activations in stud-m and stud-s have been redistributed to have larger values to compensate for compression

similar depth reported earlier works. Again, it will be seen that the student models obtained using the proposed approach outperforms the same models trained from scratch and those obtained using knowledge distillation. For example, at 94.82% and 91.38% reduction in parameters and memory size respectively, the proposed stud-m incurs a generalization loss of 0.07%, while stud-m trained from scratch incurs a generalization loss of 0.27% and the same model obtained using knowledge distillation has generalization loss of 0.12%. Also, stud-m with 29K parameters that is obtained using the proposed approach outperforms the Fitnet with 30K parameters. In addition, the test errors for deep models from earlier works with comparable depth as the student models trained in this paper are given. Fig. 9 shows the convolution maps for the first layer of the teacher, stud-m and stud-s models. We observe that compression enforces the hidden units of stud-m and stud-s to be more active (i.e. attain higher values) as feature detectors than the units in the teacher model. To further validate this, we report in Fig. 10 the distribution of average activations of the hidden units

Table 6: Compression results for proposed stud-m model trained with LASSO and then pruned using different T_h (pruning threshold value) values on MNIST dataset; P% is in reference to the teacher model used for Table 5

T_h ($\times 10^{-3}$)	0.01	0.05	0.1	0.5	1	2
Proposed: Acc.(%)	99.44	99.44	99.44	99.40	99.43	**99.38**
Proposed: Param.	20.07K	17.46K	15.21K	9.29K	7.84K	**6.53K**
Proposed: P%	96.42	96.88	97.28	98.34	98.60	**98.83**
Proposed: Sz(MB)	0.19	–	–	0.15	–	**0.14**
Stud-m (scratch): Acc.(%)	99.44	–	–	99.32	–	**99.27**

*Baseline (proposed stud.-m without LASSO): Param. = 29K, P%=94.82, 0.2MB and Acc.(%)=99.54

Table 7: Compression results on SVHN dataset

Models	Param.	P%	Acc.(%)
Stud-m: SL	**1.7M**	**81.03**	**96.74**
Stud-m: (KD) [11] with $\lambda = 0$	1.7M	81.03	96.04

overall the entire dataset.

Compression results for the proposed stud-m trained based on latent subspace learning with a $L1$-norm penalty (λ) of 10^{-3} and then pruned using different T_h values are given in Table 6. It will be seen that further compression after pruning is achieved with minimal loss of model generalization performance, especially compared to student models with the same number of model parameters that are trained from scratch. For example, with 6.53K parameters, the proposed student model achieves a test accuracy of 99.38%, while a similar student model with $L1$-norm trained from scratch achieves only 99.27% test accuracy.

5.2.4 SVHN

For this dataset, which has over 600,000 samples, the aim is to demonstrate that one can easily construct a model compression scenario in which neither knowledge distillation nor Fitnet is applicable. Particularly, one can assume that only a trained teacher model and unlabeled data that comes from the same distribution as the original data used for training teacher model are available. This scenario is not uncommon in practice where the collection of labeled data is quite expensive. For this situation where knowledge distillation and Fitnet cannot be employed since they rely on labeled data, we show that the proposed compression approach yields very promising results using the SVHN dataset. Furthermore, it demonstrates that the proposed approach can be employed for large datasets such as SVHN.

The SVHN (Street View House Number) dataset contains real-life Google street view images with digits in the range 0-9. There are 73,257 and 26,032 digits for training and testing, respectively. In addition, there are 531,131 *somewhat easier* digits that are used as extra training data. For this dataset, the standard training is not followed as in [26]. Instead, an interesting scenario that allows the validation of applicability of the proposed approach is crafted. A VGG teacher model of 8.96M

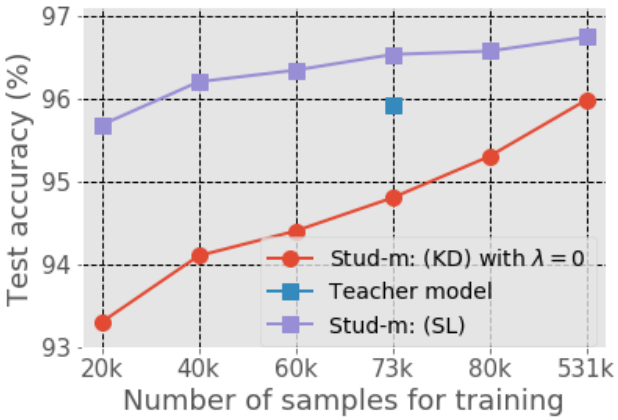


Fig. 11: Student model test classification accuracy with number of training samples for the SVHN dataset

parameters is trained on the training data (i.e. 73,257 samples) for 50 epochs. We assume that the extra training data provided is unlabeled. For model compression, *only* the extra training data are used for obtaining a trained student model of 1.7M (stud-m) parameters based on the proposed compression approach; the student model is trained for 50 epochs. The teacher model achieves a test accuracy of 95.92%.

Table 7 shows the classification accuracy of the trained models on the SVHN test data (i.e. 26,032 samples) that were not used for training the teacher and student models; a test accuracy of 96.74% is achieved. It is observed that the student model outperforms the teacher model. This observation is interesting, since we would expect the student model to slightly lag the teacher model in test performance. The student model was never trained on any of the data samples found in the training set for the teacher model. Our explanation is that in contrast to the teacher model that acquires knowledge using only the original training set, the student model in this case acquires knowledge in two ways (i) by learning the latent representations of the teacher model, it indirectly learns from the original training set (ii) learns from the newly provided unlabeled data. KD requires true data labels y for training (i.e. as in (1)); [11] particularly mentioned that this is the formulation for which KD works well, and no results without true data labels were reported. However, one can set $\lambda = 0$ in (1) so that y is not required; the first term is ignored and we arrive at the scenario similar to [37]. This approach is thus employed for testing the performance of KD using the same training settings as in subspace learning; test accuracy of 96.04% is obtained for KD as given in Table 7.

Fig. 11 shows how the size of the unlabeled dataset impacts the performance of the student model; the teacher model is used as a reference. For this experiment, the number of unlabeled data samples used for learning the student model is varied from 20,000 to 531,131. It is shown that the student model outperforms the teacher model starting from when 40,000 samples and beyond are used for learning. Also, it is shown that the performance of the proposed approach does not degrade drastically

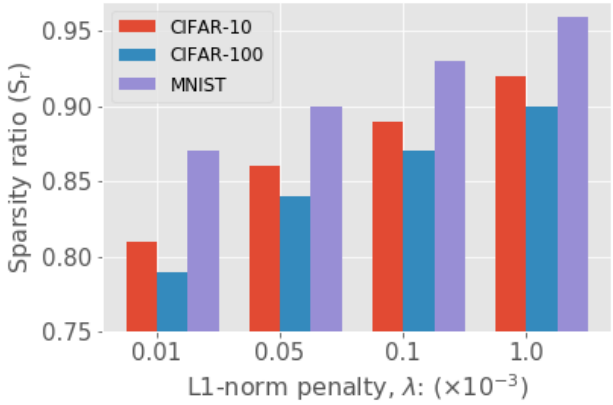


Fig. 12: Sparsity ratios for different values of $L1$-norm penalty term, λ

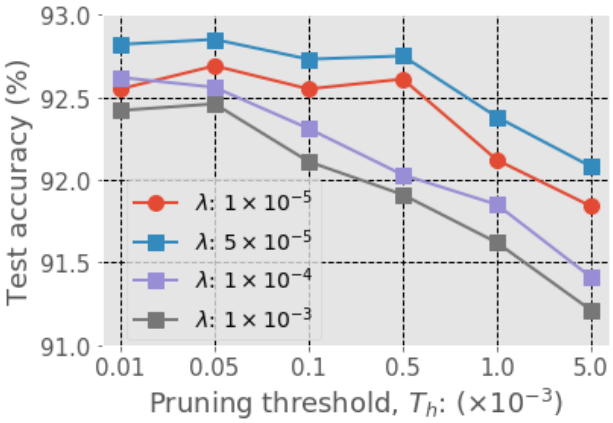


Fig. 13: CIFAR-10 test accuracy for different pruning threshold, T_h, and L1-norm penalty, λ

as the size of unlabeled dataset available for model compression decreases. For example, with 20,000 samples, the student model already achieves a test accuracy of 95.68%; this corresponds to a generalization loss of 0.25%. In addition, Fig. 11 suggests that increasing the size of unlabeled data used for model compression should result in further improvement of the student model performance.

5.3 Ablation studies

In this section, ablation studies for experiments that employed LASSO regularization for further model compression are presented. First, the sparsity ratio (i.e. amount of sparsity) that can be induced for different values of the $L1$-norm penalty, λ, is stud-

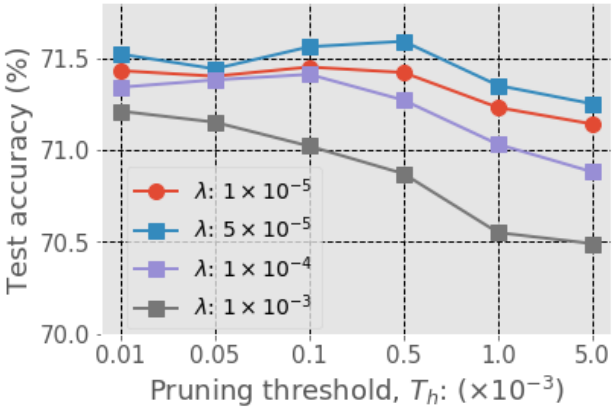


Fig. 14: CIFAR-100 test accuracy for different pruning threshold, T_h, and $L1$-norm penalty, λ

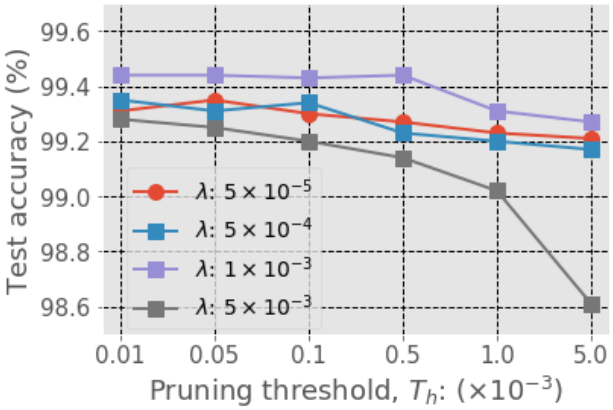


Fig. 15: MNIST test accuracy for different pruning threshold, T_h, and $L1$-norm penalty, λ

ied; the sparisty ratio, S_r, is the ratio of the number of zero-valued parameters to the total number of model parameters. The sparsity ratio results for the models trained with $L1$-norm penalty on CIFAR-10, CIFAR-100 and MNIST datasets discussed in Sections 5.2.1, 5.2.2 and 5.2.3 are given in Fig. 12. It is seen that sparsity ratios increase with an increase in the value of $L1$-norm penalty. For example, using $L1$-norm penalty of 5×10^{-4} for CIFAR-10, CIFAR-100 and MNIST datasets, sparsity ratios of 0.87, 0.84 and 0.9 can be induced, respectively.

In addition, we observe how the $L1$-norm penalty, λ, and pruning threshold values, T_h, hyperparameters impact compression results based on achieved test accuracies. For experiments, the same models and settings on CIFAR-10, CIFAR-100 and

MNIST datasets as in Sections 5.2.1, 5.2.2 and 5.2.3, respectively, are used. Fig. 13, Fig.14 and Fig.15 show results on CIFAR-10, CIFAR-100 and MNIST datasets, respectively. It is seen that $\lambda = 5 \times 10^{-5}$ yields best accuracy given compression results for the models trained on CIFAR-10 and CIFAR-100 datasets, while $\lambda = 1 \times 10^{-3}$ yields best accuracy given compression results for the model trained on the MNIST dataset. Finally, note that the sparsity ratios for different values of λ in Fig. 12 corresponds with the test accuracies reported in Fig. 13, Fig.14 and Fig.15 for similar datasets. For instance, on MNIST dataset, $\lambda = 1 \times 10^{-3}$ corresponds to a sparsity ratio of 0.96 from Fig. 12, and a test accuracy of 99.5% using $T_h = 5 \times 10^{-4}$ from Fig. 15.

6 Conclusion

Deep neural networks have proven powerful for understanding multimedia, which include photos, videos and texts. However, the large memory size of these networks and long inference time can limit their applications on electronic devices with small computational resources. As such, we propose in this paper, a generic method to compress large (teacher) models into smaller ones (student models). As deep networks typically learn lots of redundant features and are over–parameterized, therefore there exists with high likelihood a feature subspace that approximately captures the same latent representation. The essence of the approach is to learn this feature subspace in a layer-wise fashion to construct a deep student model. Moreover, we show that LASSO penalty results in improved compression results when employed for the proposed student model compared to a student model trained from scratch. Using the proposed compression method, extensive experiments on four standard datasets show that a drastic reduction of model parameters can be achieved without incurring significant loss of generalization performance. It is interesting that the proposed approach suffices in the cases where knowledge distillation and FitNets cannot be employed; that is, unavailability of labeled data. The experimental results of the proposed student models outperform those trained from scratch, using knowledge distillation, FitNet and other models with comparable number of parameters or depth. In addition, results obtained using the proposed approach suggest that abundant unlabeled data (which is usually available in real-life) can be leveraged to improve compression results such that the student model even outperforms the teacher model.

For future work, it will be interesting to explore the automatic estimation of the suitable number of convolution filters and hidden units in the Student model that results in the optimal subspace learning; this is as opposed to hand engineering of the Student model as seen in this paper and similar works [11; 12; 15]. The estimation scheme should allow the best reconstruction of the latent representations of the Teacher model using an optimal number of hidden units. A promising direction for this is the gradual increase of the Student model's convolution filters and hidden units based on the maximization of the mutual information between the Student and Teacher models. The goal is to stop adding more convolution filters and hidden units to the Student model when the mutual information objective ceases to improve.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conict of interest.

References

1. J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *IJCAI*, 2015, pp. 3995–4001.
2. O. K. Oyedotun and A. Khashman, "Deep learning in vision-based static hand gesture recognition," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3941–3951, 2017.
3. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
4. J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.
5. R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377–2385.
6. S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, 2016.
7. H. Mhaskar, Q. Liao, and T. Poggio, "Learning functions: when is deep better than shallow," *arXiv preprint arXiv:1603.00988*, 2016.
8. M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 8, pp. 1553–1565, 2014.
9. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
10. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.
11. G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Advances in neural information processing systems Workshop*, 2015, pp. 1–9.
12. L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4820–4824.
13. 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.
14. X. Zhu, S. Gong *et al.*, "Knowledge distillation by on-the-fly native ensemble," in *Advances in neural information processing systems*, 2018, pp. 7517–7527.
15. 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, pp. 1–13.
16. S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.
17. R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
18. J. Kim, Y. Kim, and Y. Kim, "A gradient-based optimization algorithm for lasso," *Journal of Computational and Graphical Statistics*, vol. 17, no. 4, pp. 994–1009, 2008.
19. S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *arXiv preprint arXiv:1507.06149*, 2015.
20. Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2857–2865.
21. D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," *arXiv preprint arXiv:1706.05394*, 2017.
22. A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10, cifar-100 (canadian institute for advanced research)," http://www.cs.toronto.edu/ kriz/cifar.html, Last accessed, Jan. 2019.

23. Y. LeCun and C. Cortes, "Mnist handwritten digit database," http://yann.lecun.com/exdb/mnist/, Last accessed, Jan. 2019.
24. A. C. A. B. B. W. A. Y. N. Yuval Netzer, Tao Wang, "The street view house numbers (svhn) dataset," http://ufldl.stanford.edu/housenumbers/, Last accessed, Jan. 2019.
25. M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
26. I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.
27. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.
28. C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.
29. W. Zhang, Y. Li, and S. Wang, "Learning document representation via topic-enhanced lstm model," *Knowledge-Based Systems*, vol. 174, pp. 194–204, 2019.
30. L. Zhao, Y. Zhou, H. Lu, and H. Fujita, "Parallel computing method of deep belief networks and its application to traffic flow prediction," *Knowledge-Based Systems*, vol. 163, pp. 972–987, 2019.
31. M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.
32. F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.
33. M. Denil, B. Shakibi, L. Dinh, N. de Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.
34. C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, "Convolutional neural networks with low-rank regularization," *arXiv preprint arXiv:1511.06067*, 2015.
35. M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.
36. B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 806–814.
37. C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.
38. Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, 2018.
39. K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.
40. R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *International Conference on Machine Learning*, 2019, pp. 7543–7552.
41. H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. Olga Russakovsky, Jia Deng* and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, pp. 211–252, December 2015.
42. S. Srinivas, A. Subramanya, and R. Venkatesh Babu, "Training sparse neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 138–145.
43. A. Joly, F. Schnitzler, P. Geurts, and L. Wehenkel, "L1-based compression of random forest models," in *20th European Symposium on Artificial Neural Networks*, 2012.
44. Y. Zhou, R. Jin, and S. C.-H. Hoi, "Exclusive lasso for multi-task feature selection," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 988–995.
45. P. Huang, S. Zhang, M. Li, J. Wang, C. Ma, B. Wang, and X. Lv, "Classification of cervical biopsy images based on lasso and el-svm," *IEEE Access*, vol. 8, pp. 24 219–24 228, 2020.
46. S. Simsek, U. Kursuncu, E. Kibis, M. AnisAbdellatif, and A. Dag, "A hybrid data mining approach for identifying the temporal effects of variables associated with breast cancer survival," *Expert Systems with Applications*, vol. 139, p. 112863, 2020.
47. P. V. C. Souza, A. J. Guimaraes, V. S. Araujo, L. O. Batista, and T. S. Rezende, "An interpretable machine learning model for human fall detection systems using hybrid intelligent models," in *Challenges and Trends in Multimodal Fall Detection for Healthcare*. Springer, 2020, pp. 181–205.

48. T. Niu, J. Wang, H. Lu, W. Yang, and P. Du, "Developing a deep learning framework with two-stage feature selection for multivariate financial time series forecasting," *Expert Systems with Applications*, vol. 148, p. 113237, 2020.
49. P. V. de Campos Souza, L. C. B. Torres, A. J. Guimaraes, V. S. Araujo, V. J. S. Araujo, and T. S. Rezende, "Data density-based clustering for regularized fuzzy neural networks based on nullneurons and robust activation function," *Soft Computing*, vol. 23, no. 23, pp. 12 475–12 489, 2019.
50. X. Wang, R. Zhang, Y. Sun, and J. Qi, "Kdgan: Knowledge distillation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 775–786.
51. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
52. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
53. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.
54. Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.
55. E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to imagenet," in *International Conference on Machine Learning*, 2019, pp. 583–593.
56. M. Jangid and S. Srivastava, "Handwritten devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods," *Journal of Imaging*, vol. 4, no. 2, p. 41, 2018.
57. D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Artificial Intelligence and Statistics*, 2009, pp. 153–160.
58. D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.
59. D. Ghadiyaram, D. Tran, and D. Mahajan, "Large-scale weakly-supervised pre-training for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 046–12 055.
60. S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *Proc. Interspeech 2019*, pp. 3465–3469, 2019.
61. L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, "Speech model pre-training for end-to-end spoken language understanding," *Proc. Interspeech 2019*, pp. 814–818, 2019.
62. J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all–solving linear inverse problems using deep projection models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5888–5897.
63. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
64. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
65. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
66. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.
67. H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *international Conference on Learning Representation*, 2017.
68. Q. Huang, K. Zhou, S. You, and U. Neumann, "Learning to prune filters in convolutional neural networks," in *Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on*. IEEE, 2018, pp. 709–718.
69. J. Zhong, G. Ding, Y. Guo, J. Han, and B. Wang, "Where to prune: Using lstm to guide end-to-end pruning." in *IJCAI*, 2018, pp. 3205–3211.

Authors' Biographies

Oyebade Oyedotun received the M.Sc. degree in electrical and electronic engineering from Near East University, Lefkosa, Turkey, in 2015. He is currently pursuing the Ph.D. degree with the Computer Vision, Imaging and Machine Intelligence (CVI^2) Research Group at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg with a focus on deep learning, machine learning and vision applications. He has authored and co-authored many scientific articles in leading IEEE conferences and journals, including Transactions on Neural Networks and Learning Systems. He reviews for several journals, including IEEE TNNLS, IEEE TKDE, IEEE Access and Neural Computing and Applications. His current research interests include machine learning and vision applications, neural networks, cognition modeling and neuroscience.

Dr. Abd El Rahman Shabayek is a Research Associate at the SnT since June 2016 as a member of the CV Lab in SIGCOM and currently CVI^2 research group, and a visiting scholar in the VIBOT Erasmus Mundus program on computer vision and Robotics since 2014. He obtained his PhD (2012, France) and Erasmus Mundus MSc (2009, UK, Spain, France) in computer vision and robotics. He was an assistant professor in the Faculty of Computers and Informatics, Suez Canal University and an adjunct assistant professor in both Information Technology Institute and Sinai University as well as a part-time project manager in EULA-Soft in Egypt from 2014 to 2016 and a research fellow in LITIS, University of Rouen, and Le2i-CNRS, University of Bourgogne, France in 2013. His research interests include 3D and non-conventional computer vision, 3D and 2D conventional and geometric deep learning, medical imaging and remote sensing. He actively participated in attracting both academic and industrial funds for different research projects in CVI2. He is the author and co-author of over 40 publications including two best paper awards. Dr. Shabayek is a Member of the IEEE, the IEEE Signal Processing Society, and INSTICC.

Djamila Aouada is a Senior Research Scientist and an Assistant Professor at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg. She is Head of the Computer Vision, Imaging and Machine Intelligence (CVI^2) Research Group at SnT, and Head of the SnT Computer Vision Laboratory. Dr. Aouada received the State Engineering degree in electronics in 2005, from the cole Nationale Polytechnique (ENP), Algiers, Algeria, and the Ph.D. degree in electrical engineering in 2009 from North Carolina State University (NCSU), Raleigh, NC, USA. Dr. Aouada has worked as a consultant for multiple renowned laboratories (Los Alamos National Laboratory, Alcatel Lucent Bell Labs., and Mitsubishi Electric Research Labs.). She has been leading the computer vision activities at SnT since 2009. Her research interests span the areas of image processing, computer vision, pattern recognition and data modelling. Dr. Aouada is Senior Member of the IEEE, member of the IEEE Signal Processing Society, IEEE WIE, INSTICC and the Eta Kappa Nu honor society (HKN). She has served as the Chair of the IEEE Benelux Women in Engineering Affinity Group from 2014 to 2016. She is the co-author of four IEEE best paper awards.

Björn Ottersten was born in Stockholm, Sweden, 1961. He received the M.S. degree in electrical engineering and applied physics from Linkping University, Linkping, Sweden, in 1986. In 1989 he received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA. Dr. Ottersten has held research positions at the Department of Electrical Engineering, Linkping University, the Information Systems Laboratory, Stanford University, the Katholieke Universiteit Leuven, Leuven, and the University of Luxembourg. During 96/97 Dr. Ottersten was Director of Research at ArrayComm Inc, a start-up in San Jose, California based on Otterstens patented technology. He has co-authored journal papers that received the IEEE Signal Processing Society Best Paper Award in 1993, 2001, 2006, and 2013 and 7 IEEE conference papers receiving Best Paper Awards. In 1991 he was appointed Professor of Signal Processing at the Royal Institute of Technology (KTH), Stockholm. From 1992 to 2004 he was head of the department for Signals, Sensors, and Systems at KTH and from 2004 to 2008 he was dean of the School of Electrical Engineering at KTH. Currently, Dr. Ottersten is Director for the Interdisciplinary Centre for Security, Reliability and Trust at the University of Luxembourg and is a board member of the Swedish Research Council. From 2012 to 2018 Dr. Ottersten was Digital Champion of Luxembourg, acting as an adviser to the European Commission. Dr. Ottersten has served as Editor in Chief of EURASIP Signal Processing, Associate Editor for the IEEE Transactions on Signal Processing and on the editorial board of IEEE Signal Processing Magazine. He is currently a member of the editorial boards of EURASIP Signal Processing, EURASIP Journal of Applied Signal Processing and Foundations and Trends in Signal Processing. Dr. Ottersten is a Fellow of the IEEE and EURASIP. In 2011 he received the IEEE Signal Processing Society Technical Achievement Award. He has received the European Research Council advanced research grant twice, 2009-2013 and 2017-2022. His research interests include security and trust as well as signal processing in wireless communications, radar and computer vision.