Neuron Manifold Distillation for Edge Deep Learning

Zeyi Tao, Qi Xia, Qun Li

Department of Computer Science
William & Mary
Williamsburg, Virginia 23185
Email: {ztao, qxia01}@email.wm.edu, liqun@cs.wm.edu

***Abstract*—Although deep neural networks show their extraordinary power in various object detection tasks, it is very challenging for them to be deployed on resource constrained devices or embedded systems due to their high computational cost. Efforts such as model partition, pruning or quantization have been used at an expense of accuracy loss. Recently proposed knowledge distillation (KD) aims at transferring model knowledge from a well-trained model (teacher) to a smaller and faster model (student), which can significantly reduce the computational cost, memory usage, and prolong the battery lifetime. In this work, we propose a novel neuron manifold distillation (NMD), where the student models not only imitate teacher's output activations, but also learn the feature geometry structure of the teacher. Our approach produces a high-quality, compact, and lightweight student model. We conduct comprehensive experiments with different distillation configurations over multiple datasets, and the proposed method demonstrates a consistent improvement in accuracy-speed trade-offs for the distilled model.**

***Index Terms*—Cloud Computing, Edge Computing, Machine Learning, Manifold Learning, Dimension Reduction**

I. Introduction

Recently, deep neural networks (DNNs) [1] have achieved state-of-the-art accuracy on many tasks ranging from computer vision to natural language processing. In particular, the success of object detection via deep convolutional neural networks has spawned many exciting applications, such as augmented reality, autopilot, and mobile shopping, on various mobile devices. The advances in machine learning have heavily relied on very deep network architectures or their ensembles. Therefore it makes them very hard to be deployed on resource constrained devices due to the high demand of computational power and battery supply.

To address the above issue, people usually deploy the DNNs on the Cloud or on the edge [2]. When users run machine learning (ML) based applications on mobile devices, a lot of raw data, such as images and personal information, will have to be delivered to the Cloud where powerful learning models reside. As a consequence, transmitting a large amount of data may cause network congestion and data plan waste. On the other hand, time-sensitive applications such as autopilot, AR games and control sensors, demand fast responses from the server. Network delay and cloud server outages, however, may result in significant quality of service (QoS) degradation. Hence, practitioners face the dilemma of processing massive data in a local environment or sending it to a cloud server.

In order to address the issue of DNNs deployability, many approaches have been proposed. They can be roughly categorized into three types: deep neural network partition, model pruning and quantization, and knowledge distillation. By partitioning models [3]–[8] or providing infrastructure that supports model layer sharing [9], we can benefit from executing the learning mode in parallel on different devices. Slicing the large DNNs across the edge of the cloud, the application pipeline becomes extended by breaking up the model layer which provides more flexibility in placement. However, how to discover the optimal model partition strategy and deal with the trade-off between runtime and large training meta-data transmission time becomes a major issue of this approach.

Network pruning [10]–[12] iteratively prunes model by removing neurons or weights that are less important to ensure the model does not have significant loss in accuracy. Parameter quantization [13]–[15] decreases the representation precision of weights to reduce computation costs and memory usages to accelerate the DNNs. In addition, to accelerate convolutional layers in DNNs, [16] and [17] use low-rank decomposition of the convolutional kernel. However, most of the techniques mentioned above are very hard to apply to the cloud-based ML applications because of two reasons. First, cloud-based ML applications are usually running on many different devices and embedding systems. In order to fit the model into different devices, it requires lots of effort on finding appropriate sub-models via network pruning which is very inefficient. Second, pruned or quantized models usually can not be reused for model training when feeding new data. In addition, even if we could run models on local devices, the model inference speed is also a well-known bottleneck for many such applications. In this paper, we focus on knowledge distillation to reduce the model size so that the model can be run on mobile devices efficiently. Knowledge distillation [18] is a promising approach to overcoming the problem of DNNs deployment. The conventional knowledge distillation uses a student-teacher paradigm in transferring the knowledge from a deeper and wider network (teacher) to a shallow and fast network (student). Fig. 1.(a)

978-0-7381-3207-5/21/$31.00 ©2021 IEEE

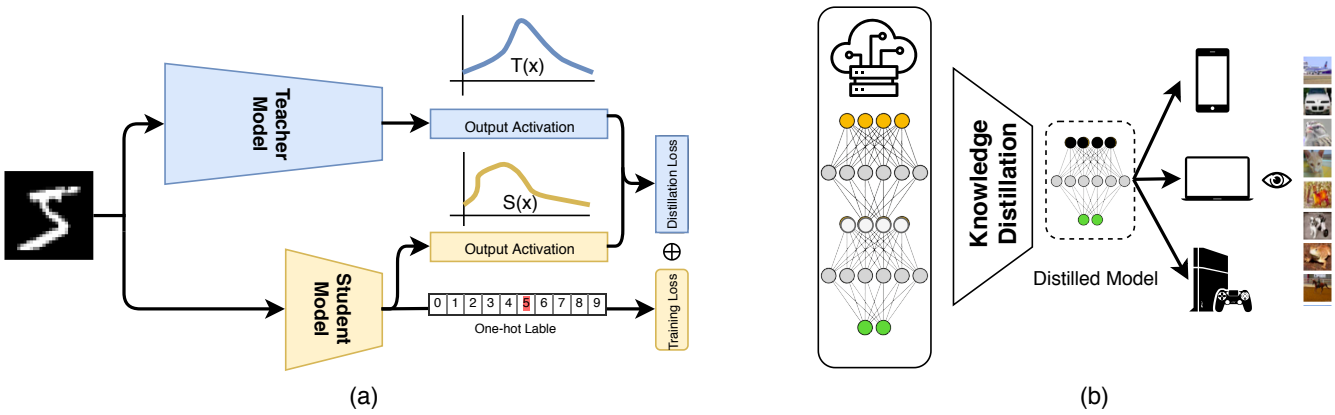

Fig. 1. (a) Conventional knowledge distillation. The well-trained state-of-the-art teacher model (deeper and wider) displays in blue color and the distilled student model displays in cream-coloured. Given an input data (handwriting 5), we train the student model via knowledge distillation by two losses. First is soft loss where we minimize the difference of student activation and teacher activation. This is also known as student mimic. The second is hard loss where we compare the prediction result with a one-hot label. (b): Cloud-based machine learning with knowledge distillation. Portable distilled models from the cloud are deployed on the various end devices.

illustrates a typical knowledge distillation process. The insight of knowledge distillation is that the student network keeps mimicking the output distribution of the teacher model, and thus students acquire information encoded in the teacher's classifiers to achieve the same prediction results.

In our machine learning design, as shown in the Fig. 1.(b) (Fig. 3 for more details), the state-of-the-art models (teacher model) are trained in the cloud or on the edge servers, e.g., through distributed training [19]–[21]. Through the knowledge distillation, we produce various student models that fit for different kinds of environments. We also call student models as pre-trained student models because they are trained by knowledge distillation. These pre-trained student models are delivered and deployed on mobile devices for different tasks. The pre-trained student model is adapted to the running environment and we can re-train it with new data locally. Users, therefore, can safely execute their private tasks on devices and update the local model. A perfect knowledge distillation would enable us to seamlessly transform the knowledge from one neural network to another, while preserving the high accuracy and generalization ability.

In this paper, we propose a novel knowledge distillation method to improve the accuracy of distilled models such that we can execute machine learning tasks on mobile devices with low computational cost. To this end, we combine the knowledge distillation with edge-cloud framework and use the method of local feature geometric extraction to build our end-to-end trainable framework for cloud-based machine learning applications. In knowledge distillation, through extracting feature geometric information, we can easily deploy lightweight learning models on resource constrained mobile devices. A key concept in our proposed knowledge distillation method is feature manifold. The feature manifold is a feature space representation but is represented in a low-dimensional space. We use a feature manifold to describe all key characteristics in the feature space. Compared with the traditional dimension reduction methods, feature manifold preserves structural relation in between features. In other words, we can use the feature manifold to recover the feature space when giving a relatively small number of feature points. In our proposed knowledge distillation method, closely matching teacher's feature manifolds with those of students means that they have an agreement in feature extraction from both numerical and relational perspectives.

Our approach has two advantages. First, our distillation method uses merely a small fraction of features to minimize the difference between teacher and student models. The reduction in the computational costs allows us to generate distilled models efficiently. The overhead that is introduced by feature manifold approximation is negligible. Because we use a simple but efficient method called feature tangent space to construct the low-dimensional manifold from a series of neighboring feature points so that we can preserve the local geometric information. Second, since our proposed distillation method using a feature manifold which can preserve both numerical and geometric information, our distilled model has been proved by our experiments that it achieves high accuracy and better generalization ability on distilled student models. In summary, the contributions of this paper are in threefold:

- We propose an end-to-end trainable framework for cloud-based machine learning applications that is running on resource constrained mobile devices. To the best of our knowledge, this is the first demonstration of knowledge distillation for an edge-cloud environment.
- We propose a new knowledge distillation method that effectively addresses the problem of deep neural network deployment. In particular, our proposed method shows advantages in the computation and communication efficiency, flexibility, portability and security.
- Our proposed method allows the distilled model to be executed on resource constrained devices where it retains high accuracy as the state-of-the-art models. The over-

head in the distillation process is low.

II. Related Work

In recent years, extensive works have been proposed to explore the knowledge distillation methods. The key to understand knowledge distillation is understanding how to properly define the knowledge in the student-teacher paradigm shown in Fig. 1.(a). The existing efforts can be broadly categorized into three types, each of which is described below.

Soft knowledge was first proposed by [18]. Because of their extraordinary pioneering work, knowledge distillation becomes more and more popular. The key insight in this work is that the network knowledge is defined as soft outputs (soften activation) of the teacher network. The one-hot hard label projects the samples of each class into a single point in the label space, while the soft activations project the samples into a continuous distribution, Student networks mimic teacher soft activations distribution as their learning outcomes. In other words, given an input, students and teachers have an agreement on input-output mapping which indicates that they have the same ability to classify. A similar work by [22] uses a perturbation logist to create a multiple teachers distillation environment. By using noise-based regularizers in the experiment, they show the reduction of intra-class variation. They claim that a proper noise level can help the student to achieve better performance. However, the drawback of using soft knowledge is also obvious, that is soft knowledge only fits for classification tasks and is limited by the number of classes. For example, in a binary classification problem, soft knowledge is barely helpful in improving the performance.

Model feature knowledge is proposed to help the knowledge distillation to use more useful information other than soft activations. DNNs extract the sample features via operational layers such as convolutional layer, ReLU, or residual block [23] in ResNet. Researchers want to utilize these intermediate features to improve the performance of distilled models. FitNet [24] forces students to mimic the full features from the teacher model. However, the depth of the student network is the same as the teacher network because they think the deep networks are more accurate than the shallow one when it is given with the same parameter budget. Later, there is evidence that FitNet may adversely affect the performance and convergence [25]. Then [26] proposed Attention Transfer (AT) by defining the network knowledge as a spatial attention map of input images. In a deep network, the size of attention map is determined by the number of neurons. Therefore, during the distillation, the student model mimics the teacher attention map point-wisely. To reduce the computational cost, they introduce an activation-based mapping function which compresses a 3D tensor to a 2D spatial attention map. A similar work applied in the vision field is proposed by [27], they call their knowledge as Flow of Solution Procedure (FSP) which distills the Gram matrix of features. Their approach is also inefficient in computation.

Jacobian knowledge This approach has been explored by [28]. In this work, authors describe the neural network as a nonlinear function. Based on this, the distillation process can be regarded as matching the Jacobians of two networks by using first-order approximation of the neural network. Our proposed knowledge distillation method is inspired by another Jacobian knowledge variation [29]. We use local affine space to approximately obtain Jacobian knowledge residing in nonlinear feature manifolds.

III. Knowledge Distillation

In this section, we will give an introduction to the main idea of knowledge distillation followed by a discussion to the issues with knowledge distillation and a possible solution.

A. The Main Idea of Knowledge Distillation

The conventional use of knowledge distillation is training a shallow student network from a large teacher ensembles. The teacher model activations are used to guide the training process. Suppose we have dataset $\{x_i, y_i\} \in \mathcal{X} \times \mathcal{Y}, i = 1, 2, \cdots, n$ where $x_i \in \mathcal{X}$ is the i-th input image and $y_i \in \mathcal{Y}$ is its class label. Let $\mathcal{T}(\cdot)$ to be the teacher model, and let $p_i^t = \text{softmax}(z_t^i)$ to be its class probability where z_i^t is the output logit for i-th image in the teacher model. Similarly, we also define $\mathcal{S}(\cdot)$ as a student model with class probability $p_i^s = \text{softmax}(z_i^s)$ where z_i^s is the output logit for i-th image in the student model. The student network $\mathcal{S}$ is trained to optimize the following loss function:

$$L_{\text{KD}} = \sum_{i \in [n]} \alpha L_{\text{hard}}(p_i^s, y_i) + (1 - \alpha) L_{\text{soft}}(p_i^s, p_i^t) \qquad (1)$$

where L_{hard} is the hard loss by using ground truth labels, and L_{soft} is the soft loss by using teacher activations and α is the hyper-parameter to balance the hard and soft losses. For example, in the original work of Hinton [18], when set $\alpha = 0$ and uses Kullback-Leibler divergence for L_{soft}, they use a soft activations ($\text{softmax}(\frac{\mathcal{T}(x_i)}{t})$) to distill the knowledge from teacher model $\mathcal{T}$ to the student model $\mathcal{S}$ as

$$L_{\text{KD}} = \sum_{x_i \in \mathcal{X}} \text{KL}(\text{softmax}(\frac{\mathcal{T}(x_i)}{t}), \text{softmax}(\frac{\mathcal{S}(x_i)}{t})) \qquad (2)$$

the temperature t aims to produce a soft probability distribution over classes. In this setting, the knowledge is transferred to the distilled student model by using a soft target distribution. Notice that the temperature is only used during the training phase. After that, the temperature will set to be 1 for inference.

Through knowledge distillation, we receive a shallow student network who has the same ability as teacher network but it can run learning tasks in portable devices. Although it seems to meet our requirements that the distilled model could fit for running on an embedded system, in fact, it has been shown that the distilled model suffers from accuracy loss and poor generalization ability. The major reason for that is due to an inappropriate knowledge selection. Properly choosing the knowledge to distill can significantly improve the model performance. In Section IV.B, we theoretically reveal the underlying principle of knowledge distillation with model features.

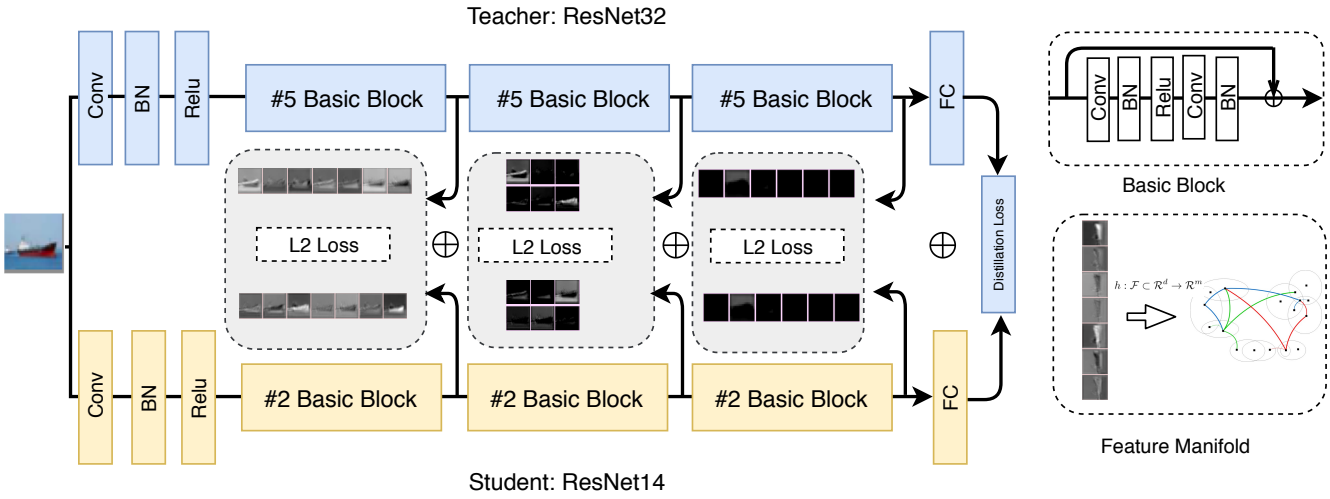

Fig. 2. Overview of our proposed method. Teacher network in light blue is a standard deep Residual Network [23](ResNet32). The student network in cream-colour is a shallow network (ResNet14). Deep neural networks do not benefit from stacking more hidden layers because of vanishing gradient problems. However, ResNet can overcome this problem by introducing identity shortcut. Given a input image, we distill the knowledge block-wisely. We optimize the feature manifold loss for each basic block.

B. Issues with Knowledge Distillation

Modern over-parameterized deep learning models have the ability to extract millions of features from the input data so that it dramatically increases their performance. These features are encoded in the different layers such as convolution layers or activation layers. They are always presented in a high-dimensional feature space.

In conventional knowledge distillation, the student models do not benefit from the features in the teacher model because they only use the information encoded in the teacher's output activations. Instead, people started to investigate feature based knowledge distillation to better use the features in the teacher model. For example, attention transfer extracts the model knowledge from multiple teacher features and then compresses these features into a small compact cubes called attention map. Although feature based approaches outperform conventional knowledge distillation, they have two major drawbacks. First, feature based methods force student model to generate the same features as in the teacher model. However, such features usually lie in a very high dimensional space. Minimizing the difference between the features of student and teacher is inefficient and time consuming. Second, learning teacher features can be regarded as conducting a very strong and strict regularization for student models [30]. Learning a student model heavily based on the features in a teacher's model will lead to model overfitting and thus poor generalization. Therefore, it is very important to decide what appropriate features to include for training the student model, and determine the parameters for regularization. Moreover, conventional knowledge distillation and feature based knowledge distillation, such as attention transfer, do not preserve the information that lies in the local geometric structure in the feature space. Here, the local geometric structure refers to the different intra-features that have relation to or in contrast with other features in a system of representations.

Suppose the high-dimensional features lie close to a low-dimensional nonlinear manifold (feature manifold). Inspired by Whye et al. [31] and Zhang et al. [32], we could use local tangent space to approximate a low-dimensional feature manifold. The tangent space we used is constructed from a series of neighborhood feature points which can preserve the local geometric information of the nonlinear manifold. The feature manifold plays a very crucial role in our proposed method because the feature manifold not only has all key characteristics to recover the entire feature space, but also reveals the structural relation among all features. In the process of knowledge distillation, closely matching a teacher feature manifolds with that of a student indicates they have an agreement in feature extraction. In other words, both models have the same output based on the same input data. In this way, we can obtain a smaller compact student model. Compared to feature-based distilled models, our method learns knowledge more efficiently with the same amount of training time and resources. It is not trivial to discover the structure of the manifold from a set of features sampled from neural networks possibly with noise. We will elaborate our approach in the next section.

IV. Neuron Feature Manifold Distillation

In this section, we will introduce our proposed knowledge distillation method. Neuron manifold distillation (NMD) focuses on transferring both numerical and structural knowledge by using low dimensional feature manifolds in the teacher network. As mentioned before, conventional knowledge distillation is only focusing on mimicking teacher output distributions. As a consequence, those methods show high training accuracy but may fail on the test data where the poor

generalization ability occurs. On the other hand, for feature based methods, distilling compact representations from teacher features is not enough. The valuable knowledge resides in infinitely many data structures and should not be neglected. We combine the method of local feature geometric extraction and knowledge distillation and propose NMD to distill a flexible, portable and computation efficient model.

Fig. 2 illustrates the complete architecture of our proposed method. We extract the features generated by basic block in ResNet to guide student training. However, unlike method of attention transfer or flow of solution procedure, we do not use full features for distillation. We use feature manifold, a low-dimensional feature representation, to distill the knowledge from teacher to student network. How can we find a proper representation which can not only characterize the feature space but also carry on structural feature relations becomes a key challenge of our work. To this end, constructing a feature manifold becomes to our solution. By gathering both teacher features $\mathcal{F}_t$ and student features $\mathcal{F}_s$, we first compute their feature manifolds by using manifold extraction function $\psi(\cdot)$. Then we minimize the l_2 distance of the feature manifold as $||\psi(f_i^t) - \psi(f_i^s)||_2^2$ for each feature, where $f_i^t \in \mathcal{F}_t$ and $f_i^s \in \mathcal{F}_s$. The optimization problem can be formulated as

$$\begin{aligned} L_{\text{KD}} =& \alpha L_{\text{hard}}(p_s, y) + \gamma L_{\text{soft}}(p_s, p_t) \\ &+ \sum_{f_i^t \in \mathcal{F}_t, f_i^s \in \mathcal{F}_s,} \beta_i ||\psi(f_i^t) - \psi(f_i^s)||_2^2 \end{aligned} \tag{3}$$

where α, β_i, γ are hyper-parameters that control the weight of each component. As discussed before, comparing all the features f_i^t and f_i^s in the loss function requires much computation. To address the problem, we intend to use a feature manifold generation function $\psi(\cdot)$ to map the feature space to lower dimensional space so that the comparison of the features in the teacher model and student model can be much more efficient. The choices of hyper-parameter β_i are very important. For lower level features, we choose bigger β value, in contrast, for deeper level features, we choose smaller β value. In the next section, we carefully explain how to extract the feature geometric information from a given feature space as a feature manifold.

A. Feature Manifold Extraction

We assume there is a d-dimension manifold $\mathcal{M} \subset \mathcal{R}^d$ is embedded in a m-dimension space $\mathcal{R}^m$ such that $m \gg d$. we define a construction function $h(\cdot)$ as

$$h : \mathcal{M} \subset \mathcal{R}^d \to \mathcal{R}^m \tag{4}$$

Here, the construction function $h(\cdot)$ will produce a high dimension vector from a low dimension space. Suppose we have deep learning feature space $\mathcal{F}$ is a subset of $\mathcal{R}^m$ such that $\mathcal{F} \subset \mathcal{R}^m$. Given N learning features, the feature set F is $F = [f_1, f_2, \cdots, f_N]$. For each $f_i \in \mathcal{R}^m$ and $i \in [N]$, we have feature construction function $h_f(\cdot)$ as

$$f_i = h_f(v_i) + \epsilon_i, \quad i = 1, \cdots, N \tag{5}$$

where v_i is the feature manifold for i-th feature f_i and ϵ_i is error. We would like to represent the learning feature f_i by using a low-dimension vector v_i. In the following we show how to find such a low dimension manifold representation v_i for feature f_i.

Next, we start by showing the linear feature manifold approximation to give some intuition first and then explain the method for a realistic non-linear approximation.

In the construction of *linear* feature manifold (which is not always the case), we assume the feature are constructed by a d-dimensional affine subspace, i.e.,

$$f_i = c + \mathcal{A}v_i + \epsilon_i, \quad i = 1, \cdots, N \tag{6}$$

where $c \in \mathcal{R}^m$, $v_i \in \mathcal{R}^d$ and $\epsilon_i \in R^m$ is error. $\mathcal{A} \in \mathcal{R}^{m \times d}$ is a matrix forms an orthonormal basis of the affine subspace. In matrix form, we have:

$$F = ce^\top + \mathcal{A}V + E \tag{7}$$

where $F = [f_1, f_2, \cdots, f_N]$, $V = [v_1, v_2, \cdots, v_N]$, $E = [\epsilon_1, \epsilon_2, \cdots, \epsilon_N]$ and e is an N-dimensional column vector of all ones. Therefore, Equation 7 forms our linear feature manifold construction function. Our goal is going to find c, $\mathcal{A}$ and V to minimize the error E, i.e.,

$$\arg\min_{c,\mathcal{A},V} ||E||_2^2 = \arg\min_{c,\mathcal{A},V} ||F - (ce^\top + \mathcal{A}V)||_2^2 \tag{8}$$

where $||\cdot||_2$ is the Frobenius norm. Optimization problem in Equation 8 can be solved by singular value decomposition (SVD) [31]. Letting $c = \bar{f} = Fe/N$ to be the mean of F, the optimal solution of low-rank matrix $\mathcal{A}V$ is computed by the SVD of $F - \bar{f}e^\top$, i.e.,

$$F - \bar{f}e^\top = U\Sigma Q^\top \tag{9}$$

We can set $\Sigma_d = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d)$ of the d largest singular values of $F - \bar{f}e^T$ such that $\sigma_d \gg \sigma_{d+1}$, and U_d, Q_d are the matrices of the corresponding left and right singular matrix respectively such that $\mathcal{A}V = U_d\Sigma_d V_d^\top$. Remember our goal is to find the low-dimension representation $v_i \in V$ by given its corresponding feature f_i. Since the optimal $\mathcal{A}$ is given by U_d, we have the close form solution of construction function h_f as

$$h_f(V) = \bar{f}e^\top + U_d V^\top \tag{10}$$

and low-dimension representation V as

$$V = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_d)Q_d^\top \tag{11}$$

However, in deep learning feature space, things are more complicated and discovering the nonlinear manifolds is challenging. There are evidences from [33] showing that the traditional dimension reduction techniques such as PCA or multidimensional scaling (MDS) fail to identify the underlying structure of data points. In other words, the result from Equation 11 does not hold for non-linear feature space where the V is unable to preserve the structural information in low dimension space. We will use an approach inspired by [32] and [33] to find the learning feature manifold.

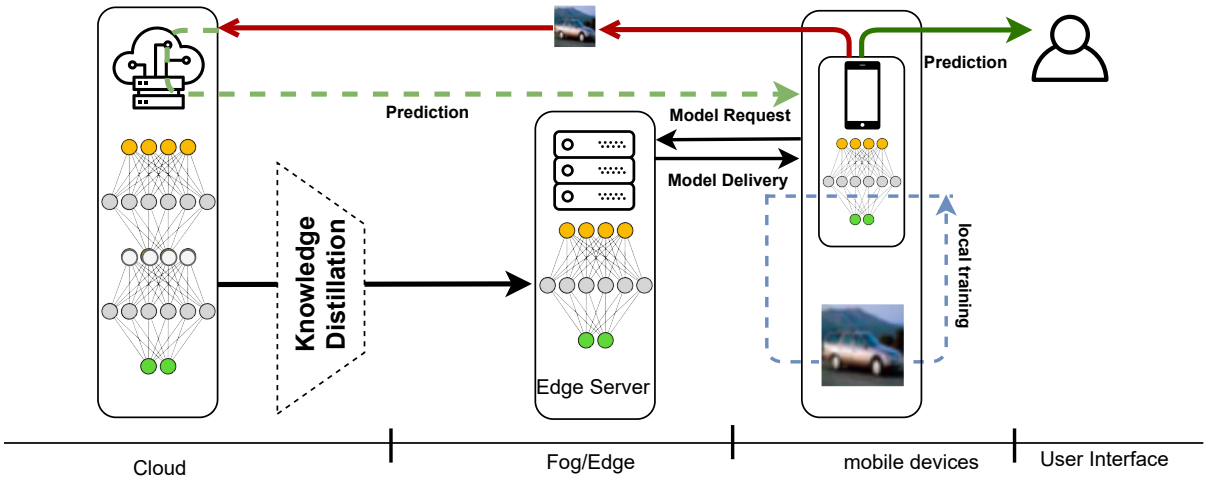

Fig. 3. Knowledge distillation based machine learning on cloud. Pre-trained the-state-of-the-art deep networks deploy on the cloud server. The knowledge distillation is running on the cloud server. The distilled models are sent to the edge server and they are ready to deliver to deploy on the edge devices. Users can request for learning service by collecting the learning models from edge server. The local model can be privately trained with new data. If user would like to share their data to improve the model, they could send the data to the cloud as well.

Let us assume the feature f_i are presented as weighted linear combinations of selected neighbor features (features that are close to f_i). Consider Equation 5 ignoring the noise term. Assume that the function h_f is smooth enough, using first-order Taylor expansion at a fixed v, and define a small neighbourhood with $\{||\hat{v} - v||^2 < \epsilon\}$, we have:

$$h_f(\hat{v}) = h_f(v) + \nabla_v h_f(v)^T(\hat{v} - v) + \mathcal{O}(||\hat{v} - v||^2) \quad (12)$$

The Equation 12 is known as the local affine approximation of the function $h_f(\cdot)$ around the local neighbourhood. Let $\nabla_v h_f(v)^T = \mathcal{J}_h(v)$ to be the Jacobian matrix of h_f at v in $\mathcal{R}^{m\times d}$, we have

$$\mathcal{J}_h(v) = \begin{bmatrix} \partial h^{(1)}/\partial v^{(1)} & \cdots & \partial h^{(1)}/\partial v^{(1)} \\ \vdots & \vdots & \vdots \\ \partial h^{(m)}/\partial v^{(1)} & \cdots & \partial h^{(m)}/\partial v^{(d)} \end{bmatrix}$$

where $h^{(i)}$ and $v^{(i)}$ indicate the i-th element of vector. We omit h_f and use h for convenience. We know that the tangent space $\mathcal{K}_v$ of h at v is spanned by this d column Jacobian matrix $\mathcal{J}_h(v)$ i.e., $\mathcal{K}_v = \text{span}(\mathcal{J}_h(v))$ which ensure the dimension of v at most d. Now, we would like to use the following lemma to illustrate how the local affine transformation can help us to find the v.

Lemma 1. *(Local Affine Transformation [32]) Assume that V is a d-dimensional manifold in a m-dimensional space with unknown generating function $h(v)$ and $v \in R^d$, and we are given a data set consists of N m-dimensional vectors $F = [f_1, f_2, \cdots, f_N]$ and $f_i \in R^m$. The first-order Taylor expansion of nosie-free model $f_i = h(v_i)$ at a fixed v_i has Jacobi matrix $\mathcal{J}_h(v_i)$. If the orthonormal basis Q_v of tangent space $\mathcal{K}_v = span(\mathcal{J}_h(v))$ exist such that $\mathcal{J}_h(v)(v-\hat{v}) = Q_v\theta_v$ where $\theta_v = Q_v^\top(h(\hat{v})-h(v))$. When the data point f_i has form of $f_i = \bar{F} + Q_i\theta_i$, the corresponding low dimension v_i can be expressed as $v_i = \bar{v} + P_i\theta_i$ where P_i is unique affine matrix.*

This lemma applies the local affine approximation from high-dimensional space $h(v)$ to a low-dimensional space v by using Jacobian matrix. Similarly, we can utilize this result to address our feature manifold approximation in a non-linear feature space. We use same settings mentioned in the section IV.B.

For each feature f_i, let $\mathcal{N}_{f_i} = [f_{i1}, \cdots, f_{ik}]$ be a matrix such that $f_{i1} = f_i, f_{i2}, \cdots, f_{ik}$ are the top-k features in $\{f_1, f_2, \cdots, f_N\}$ that are closest to f_i (including f_i). Given Q_i, which are the left singular vectors of $\mathcal{N}_{f_i}(I - \frac{1}{k}ee^\top)$ and $\Theta_i = [\theta_1^i, \cdots, \theta_k^i]$ where $\theta_j^i = Q_i(f_{ij} - \bar{\mathcal{N}}_{f_i})$, we can express f_{ij} in the following form:

$$f_{ij} = \bar{\mathcal{N}}_{f_i} + Q_i\theta_j^i + \epsilon_j^i \quad (13)$$

According to the Lemma 1, we have

$$v_{ij} = \bar{V}_i + P_i\theta_j^i + \epsilon_j^i \quad (14)$$

where $V_i = [v_{i1}, v_{i2}, \cdots, v_{ik}]$ and P_i is a local affine transformation with respect to the local geometry determined by the θ_j^i. Now we proceed to solve for P_i and v_{ij}.
Let $E_i = [\epsilon_1^i, \cdots, \epsilon_k^i]$. Equation 14 can be expressed in the following matrix form, i.e.,

$$V_i = \frac{1}{k}V_iee^T + P_i\Theta_i + E_i \quad (15)$$

To optimize the local construction error $||E_i||$:

$$\arg\min_{V_i,P_i} ||E_i|| = \arg\min_{V_i,P_i} ||V_i(I - \frac{1}{k}ee^\top) - P_i\Theta_i||_2 \quad (16)$$

The optimal P_i can be founded

$$P_i = V_i(I - \frac{1}{k}ee^\top)\Theta_i^+ \quad (17)$$

where Θ_i^+ is the Moore-Penrose generalized inverse of Θ_i. Now we have:

$$\min ||E|| = \sum_i ||V_i(I - \frac{1}{k}ee^\top)(I - \Theta_i^+\Theta_i)||_2 \quad (18)$$

Finally, our goal is to determine the optimal V in the low-dimension space. Let S_i be a 0-1 selection matrix such that $V_i = VS_i$ and $W = \text{diag}(W_1, \cdots, W_N)$ with $W_i = (I - \frac{1}{k}ee^\top)(I - \Theta_i^+\Theta_i)$. Recall that $V = [v_1, v_2, \cdots, v_N]$. The standard solution of V in Equation is given by the eigenvectors of the matrix $B = SWW^\top S^\top$ where $S = [S_1, \cdots, S_N]$. At this point, we can obtain the low dimensional nonlinear feature manifold V as desired. In next section, we will explain why this feature manifold is closely related to knowledge distillation in the theory.

B. Local Affine Approximators of Neural Networks

In this section, we will answer the question that why it is necessary to use teacher feature in knowledge distillation and the question that how it relates to our feature manifold, we recall the equation.1 in Section III,

$$L_{\text{kd}} = \alpha L_{\text{hard}}(p_s, y) + (1-\alpha)L_{\text{soft}}(p_s, p_t)$$

We notice that the soft losses contain information about the relationship between different classes as discovered by teacher. By learning from soft losses, the student network inherits such dark knowledge. To see how does the knowledge distillation relate to our local affine approximation, we define a multi-layer teacher neural network $\mathcal{T} : R^{\text{input}} \to R^{\text{output}}$:

$$\mathcal{T} = t_1(t_2(\cdots(t_l))) \quad \text{and} \quad t_l(x) = \sum_{i=1}^{p_l} a_{l,i}\phi_l(\langle w_l, x\rangle + b_l) \quad (19)$$

where ϕ_l is l^{th} operation layer such as ReLU activation, w_l is the hidden weight, b_l is the bias vector, and $a_{l,i}$ is the output weight vector, and p_l indicates the number of hidden neurons. To simplify the analysis, we assume the l^{th} layer of teacher network extracts the corresponding feature according to $t_l(a_{l-1}) = f$ where a_{l-1} is the activations from layer $l-1$. We know the l-th layer feature has a construction function $h_f(v) + \epsilon$ with the respect to low dimensional feature space of V. Then we have following result.

Theorem 2. *(Local Affine Equivalence Theorem) Consider the squared error cost function for matching soft targets of two neural networks at l^{th} layer with p features, given by*

$$L_{soft}(\mathcal{T}_{t_l}(x), \mathcal{S}_{t_l}(x)) = \sum_{i=1}^{p}(\mathcal{T}_{t_l}^i(x) - \mathcal{S}_{t_l}^i(x)))^2$$

where x is the activations from layer $l-1$. and ξ is noise. Let $f_t = \mathcal{T}_{t_l}(x)$ and $f_s = \mathcal{S}_{t_l}(x)$, the $f_t, f_s \in \mathcal{F}$ are the features in feature space $\mathcal{F}$ which are generated by $\mathcal{T}_{t_l}$ and $\mathcal{S}_{t_l}$, respectively. Then matching soft targets of two neural networks is equivalent to match their feature manifold where

$$\begin{aligned} E[\sum_{i=1}^{p}(\mathcal{T}_{t_l}^i(x+\xi) - \mathcal{S}_{t_l}^i(x+\xi))^2] &= \sum_{i=1}^{p}(\mathcal{T}_{t_l}^i(x) - \mathcal{S}_{t_l}^i(x))^2 \\ &+ \sigma^2 E[\sum_{i=1}^{p}(\nabla_x h_t^i(v) - \nabla_x h_s^i(v))^2] + \mathcal{O}(\rho^4) \end{aligned} \quad (20)$$

where $\nabla_x h_t^i(v)$ is the Jacobian matrix of h_t at v.

Proof. To prove this, we recall Section IV.B, the feature manifolds are given by two construction function where $\mathcal{T}_{t_l} = h_t(\hat{v})$ and $\mathcal{S}_{t_l}(x+\xi) = h_s(\hat{v})$ where ξ is small noise. Using first-order Taylor expansion at a fixed v, and a small neighbourhood with $\{||\hat{v} - v||^2 < \rho\}$ and ρ is small, we have

$$\begin{aligned} &E[\sum_{i=1}^{p}(\mathcal{T}_{t_l}^i(x+\xi) - \mathcal{S}_{t_l}^i(x+\xi))^2] = E[\sum_{i=1}^{p}(h_t^i(\hat{v}) - h_s^i(\hat{v}))^2] \\ =&E_\rho[\sum_{i=1}^{p}(h_t^i(v) + \nabla_v h_t^i(v)\rho - h_s^i(v) - \nabla_v h_s^i(v)\rho))^2] + \mathcal{O}(\rho^4) \\ =&\sum_{i=1}^{p}(h_t^i(v) - h_s^i(v))^2 + E_\rho[\sum_{i=1}^{p}\rho^2(\nabla_x h_t^i(v) - \nabla_x h_s^i(v))^2] \\ =&\sum_{i=1}^{p}(\mathcal{T}_{t_l}^i(x) - \mathcal{S}_{t_l}^i(x))^2 + \rho^2 E[\sum_{i=1}^{p}(\nabla_x h_t^i(v) - \nabla_x h_s^i(v))^2] \end{aligned}$$

We omit $\mathcal{O}(\rho^4)$ at third and last row. □

We notice that the loss function has two components. The first term indicates the conventional loss of knowledge distillation on samples from both teacher and student. And the second regularizer term represents the difference of local affine approximation of the given networks. The final error terms are small for small ρ and can be ignored. From the observation in Theorem.2, we can conclude that it is ill-considered practice for conventional knowledge distillation that it only transfers the raw CNN activation outputs point-wisely to their offspring model. For a complete knowledge distillation method, transfer of the underlying knowledge residing in the infinitely many data points nearby is also obligatory. Now, back to our topic in Section IV.B, our feature manifold can be regarded as a highly compressed abstraction extracted from feature local affine space, which can preserve the greatest level of feature geometric information by using local tangent space.

V. KD BASED MACHINE LEARNING ON CLOUD

Our cloud-based machine learning framework with the help of model knowledge distillation is shown in Fig. 3. Through knowledge distillation, cloud sends and stores the distilled model on the edge server in advance. When we initiate a machine learning request, if the mobile device already downloads the model, it can start the task immediately and quickly response to users. If the local device does not have a required model, it will generate a request according to its environmental information to the proximal edge server [34] and prepare to receive the corresponding distilled model. However, in case of outdated models which have low confidence in prediction, users can still upload the task data to the cloud to have correct results (shown in red line and green dash line). In our design, clouds can generate different intensity distilled models. In other words, different quality & size models stored in edge servers can meet the different requirements of various level tasks. It is worthy to mention that the distilled model can also be trained in a private environment (shown in blue dot line).

Ideally, once the mobile devices acquire the distilled model, all the computation and data motion will occur in an independent environment that turns out to be communication and computation efficient. In addition, running flexible and

TABLE I
THE PERFORMANCE OF DIFFERENT DISTILLATION METHODS

TCH/STD Model	KD Method	top-1 Acc.	top-5 Acc.
ResNet 50/ResNet 14	KD (baseline)	63.4%	90.3%
	AT [26]	72.2%	93.3%
	Gram matrix [27]	66.1%	85.7%
	NMD (this paper)	**73.9%**	**92.1**%
VGG 19/VGG 19	KD (baseline)	62.1%	83.4&
	AT [26]	71.3%	90.8%
	Gram matrix [27]	66.5%	89.4%
	NMD (this paper)	**72.1**%	**91.2%**

TABLE II
DISTILLED MODELS WITH STATS

	Model	AlexNet	Res18	Res34	Res50	Res101
Stats	# para(M)	5	11	21	23	42
	Memory	204	588	1280	1560	2640
	GFLOPs	0.6	1.4	3.44	3.56	6.68
	Time(min)	14	29	37	58	144
Acc.(%)	KD	82.4	89.2	90.2	89.1	91.2
	AT	-	90.3	90.0	91.1	88.9
	AT+KD	-	91.9	92.1	93.1	92.3
	NMD	88.6	91.3	92.9	92.4	93.7

TABLE III
THE PERFORMANCE OF DISTILLED MODEL

TCH/STD Model	KD Method	Training Time	Acc.
VGG 16/VGG 16	KD (baseline)	183	71.4&
	AT [26]	235	72.1%
	Gram matrix [27]	260	73.8%
	NMD (this paper)	193	74.2%

light-weight distilled models on mobile devices provides convenience for privacy concerns. In fact, users can reserve the right to determine and upload the privacy data to the cloud so that they can get high accuracy prediction results. Meanwhile, distilled models can conduct self-reinforcement (supervised learning) when meeting the new data. Last but not the least, our framework can adapt rapid model updating or maintenance. When cloud side models are updated, the newly distilled model will be stored on an edge server for other nodes downloading.

VI. EVALUATION

In this section, we evaluate the performance of our proposed method on several metric learning tasks. We compare our results with several state-of-the-art distillation methods. We also deploy our distilled model on resource constrained devices such as Raspberry Pi 4 Model B to evaluate the distilled model performance. Our experiments show that the proposed method achieves very promising performance improvement.

A. Experiment Setup

Environment To simulate the cloud-edge and mobile framework, we use two experimental settings. Training teacher networks and performing a series of knowledge distillations are conducted on a lab server with 32Mb Memory and 4 GeForce GTX 1080ti GPUs. We use Raspberry Pi 4 Model B with Quad core Cortex-A72 (ARM v8) Cup without a graphic card installed as our mobile device. We implement our model by Google TensorFlow 1.9 which can fully support the Raspberry Pi.

Datasets and Deep neural networks The datasets we use for image classification include CIFAR10, CIFAR100 and ImageNet (ILSVRC2013). The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million 32x32 colour image dataset. ImageNet consists of two parts, training data and validation data. The training data contains 1000 categories and 1.2 million images and the validation and test data consists of 150,000 photographs, collected from flickr and other search engines, hand labeled with the presence or absence of 1000 object categories. The training models we are using include ResNe50, ResNet34, VGG16 and VGG19.

Training Settings In our experiments, the SGD with momentum (SGDM) is adopted for knowledge distillation where momentum value is set to be 0.9 and weight decay is 0.0001. For conventional knowledge distillation [18], the hyper-parameter t temperature is set to be 6. We train all distilled model with 60 epochs.

B. The performance of NMD

We first evaluate the performance of NMD. We use a pre-train ResNet50 model on ImageNet as a teacher network which provides 75.2% top-1 accuracy and 92.2% top-5 accuracy as reported in [35]. And we use ResNet14 as a student network. The conventional knowledge distillation achieves 63.4% and 90.3% top-1 and top-5 accuracy respectively. Our proposed method NMD achieves 73.9% and 92.1% top-1 and top-5 accuracy respectively. Compared to the attention transfer and its variant which uses Gram Matrix, our proposed method achieves the-state-of-the-art accuracy in the distilled model.

We also conduct experiments with VGG19 [36] on ImageNet. We use two VGG19 networks where one network is used as a teacher and the other used as a student network. This type of knowledge distillation is called self-distillation. In this experiment, we want to know whether the distillation can help model with generalization or not? As shown in Table. I, our proposed method achieves 72.1% on The experiments on self-distillation also states the fact that properly distilled model can have better generalization ability than teacher. We also notice that, conventional knowledge and Garm matrix transfer have relatively poor performance even worse than the original model.

Further, we want to know how small the distilled model can be. This is very important as smaller models are appealing to mobile devices and can be fitted on their running environment. We train CIFAR-100 on a very deep neural network ResNet 152 as our teacher network. The student networks include ResNet101, ResNet50, ResNet34, ResNet18 and AlexNet. We gradually reduce the number of layers in deep networks and we collect the statistics data for each model showing in the

Fig. 4. Image query on distilled model

Table. II. Reducing the number of network layers means reducing the number of parameters that the network uses. Therefore the memory and GFLOPs decrease. We discover that an over-parameterized deep neural network has sufficient capacity to memorize training results, that is, AlexNet who has only 5 Million parameters can still achieve high accuracy. Another fact is when we reduce the DNNs layers, there is no significant accuracy drop in the distilled model. NMD can be applied on distilling to smaller models, for example, we only use the last layer of AlexNet to match feature manifolds with ResNet152, however, methods such as attention transfer can not be applied.

Table. III illustrates the single batch training time of knowledge distillation of different approaches. We use self-distillation on two VGG16 networks. The training batch size is fixed as 256. The conventional knowledge distillation does not introduce any extra computation and therefore it uses the shortest time as desired. Compared with other approaches, NMD achieves 17 % to 25% speed up in training a single batch. top-1 accuracy.

C. The performance of distilled model

To evaluate the performance of the distilled model, we conduct an experiment in which we use a 7-layer LeNet as a teacher network and we use a super shallow network named Hinton-1200 [18] as a student network. We simulate the machine learning task as handwritten digits recognition. We train LeNet by the MNIST database, it has a training set of 60,000 examples, and a test set of 10,000 examples, and all digits are 28x28 images in greyscale. As shown in Fig. 4, the input image on the left is a new query data, this image does not present in either the train or test dataset. We retrieve the top six prediction results from both teacher and student. Both teacher and student give the correct answers. We can see that the teacher can always return the most accurate predictions in handwritten digits. However, when comparing teacher's model to the distilled model that solely uses teacher's lower level activations, we see the output "5" (right most in the second row) is distorted and looks similar to "6". It is also worth mentioning that the outputs of the distilled network in the red box have similar shape to the input image since NMD can preserve the spatial relation of data.

VII. Conclusion

In this paper, we propose an end-to-end trainable framework for cloud-based machine learning applications running on resource constrained mobile devices. We also propose a new knowledge distillation method that effectively addresses the problem of deep neural network deployment. By using this method, our trainable framework becomes more flexible, portable and communication efficient. Finally, we shed light on internal relations between knowledge distillation and feature geometry structure information.

ACKNOWLEDGEMENTS

We thank all reviewers for their helpful comments. This project was supported in part by US National Science Foundation grant CNS-1816399. This work was also supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[2] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," *High-Confidence Computing*, 2021.

[3] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "Learning from differentially private neural activations with edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 90–102.

[4] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A configurable cloud-scale dnn processor for real-time ai," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, June 2018, pp. 1–14.

[5] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Optimizing deep cnn-based queries over video streams at scale," *CoRR*, vol. abs/1703.02529, 2017. [Online]. Available: http://arxiv.org/abs/1703.02529

[6] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," *CoRR*, vol. abs/1802.03835, 2018. [Online]. Available: http://arxiv.org/abs/1802.03835

[7] K.-J. Hsu, K. Bhardwaj, and A. Gavrilovska, "Couper: Dnn model slicing for visual analytics containers at the edge," *SEC*, 2019. [Online]. Available: http://acm-ieee-sec.org/2019/program.php

[8] Y. Mao, W. Hong, H. Wang, Q. Li, and S. Zhong, "Privacy-preserving computation offloading for parallel deep neural networks training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1777–1788, 2021.

[9] A. H. Jiang, D. L.-K. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, and G. R. Ganger, "Mainstream: Dynamic stem-sharing for multi-tenant video processing," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, Jul. 2018, pp. 29–42. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/jiang

[10] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. N. Choudhary, and S. Chang, "Fast neural networks with circulant projections," *CoRR*, vol. abs/1502.03436, 2015. [Online]. Available: http://arxiv.org/abs/1502.03436

[11] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *CoRR*, vol. abs/1506.02626, 2015. [Online]. Available: http://arxiv.org/abs/1506.02626

[12] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: efficient inference engine on compressed deep neural network," *CoRR*, vol. abs/1602.01528, 2016. [Online]. Available: http://arxiv.org/abs/1602.01528

[13] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[14] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," *CoRR*, vol. abs/1504.04788, 2015. [Online]. Available: http://arxiv.org/abs/1504.04788

[15] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, 2014. [Online]. Available: http://arxiv.org/abs/1412.6115

[16] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *CoRR*, vol. abs/1404.0736, 2014. [Online]. Available: http://arxiv.org/abs/1404.0736

[17] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *CVPR*, 12 2014. [Online]. Available: https://arxiv.org/abs/1412.6553

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv e-prints*, p. arXiv:1503.02531, Mar 2015.

[19] Z. Tao and Q. Li, "eSGD: Communication efficient distributed deep learning on the edge," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, July 2018.

[20] Q. Xia, Z. Tao, Z. Hao, and Q. Li, "FABA: An algorithm for fast aggregation against byzantine attacks in distributed neural networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4824–4830.

[21] Q. Xia, Z. Tao, and Q. Li, "Defenses against byzantine attacks in distributed deep neural networks," *IEEE Transactions on Network Science and Engineering*, 2021.

[22] B. B. Sau and V. N. Balasubramanian, "Deep model compression: Distilling knowledge from noisy teachers," *CoRR*, vol. abs/1610.09650, 2016. [Online]. Available: http://arxiv.org/abs/1610.09650

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[24] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *CoRR*, vol. abs/1412.6550, 2014. [Online]. Available: http://arxiv.org/abs/1412.6550

[25] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," *arXiv preprint arXiv:1707.01219*, 2017.

[26] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *CoRR*, vol. abs/1612.03928, 2016. [Online]. Available: http://arxiv.org/abs/1612.03928

[27] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 7130–7138.

[28] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, "Sobolev training for neural networks," *CoRR*, vol. abs/1706.04859, 2017. [Online]. Available: http://arxiv.org/abs/1706.04859

[29] S. Srinivas and F. Fleuret, "Local affine approximations for improving knowledge transfer," *Idiap*, 2018. [Online]. Available: http://infoscience.epfl.ch/record/256319

[30] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *CoRR*, vol. abs/2006.05525, 2020. [Online]. Available: https://arxiv.org/abs/2006.05525

[31] Y. W. Teh and S. Roweis, "Automatic alignment of local representations," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 865–872. [Online]. Available: http://dl.acm.org/citation.cfm?id=2968618.2968726

[32] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," *CoRR*, vol. cs.LG/0212008, 2002. [Online]. Available: http://arxiv.org/abs/cs.LG/0212008

[33] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: https://science.sciencemag.org/content/290/5500/2323

[34] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, and Q. Li, "A survey of virtual machine management in edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1482–1499, 2019.

[35] Tensorflow, "tensorflow pre-trained models," Apr 2019. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/slim/Pretrained

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.