Matching Guided Distillation

Kaiyu Yue, Jiangfan Deng, and Feng Zhou

Algorithm Research, Aibee Inc.

Abstract. Feature distillation is an effective way to improve the performance for a smaller student model, which has fewer parameters and lower computation cost compared to the larger teacher model. Unfortunately, there is a common obstacle — the gap in semantic feature structure between the intermediate features of teacher and student. The classic scheme prefers to transform intermediate features by adding the adaptation module, such as naive convolutional, attention-based or more complicated one. However, this introduces two problems: a) The adaptation module brings more parameters into training. b) The adaptation module with random initialization or special transformation isn't friendly for distilling a pre-trained student. In this paper, we present Matching Guided Distillation (MGD) as an efficient and parameter-free manner to solve these problems. The key idea of MGD is to pose matching the teacher channels with students' as an assignment problem. We compare three solutions of the assignment problem to reduce channels from teacher features with partial distillation loss. The overall training takes a coordinate-descent approach between two optimization objects — assignments update and parameters update. Since MGD only contains normalization or pooling operations with negligible computation cost, it is flexible to plug into network with other distillation methods. The project site is http://kaiyuyue.com/mgd.

1 Introduction

Deep networks [41,18] enjoy massive neuron parameters for achieving the state-of-the-art performances on lots of technique lines, such as visual recognition [9], image captioning [17], object detection system [37] and language understanding [38,5]. However, the industry prefers to carry out model inference on cheap devices, therefore the small model with few parameters is needed. The dilemma of achieving analogous performance to the large model in lightweight backbone recently motivates extensive research directions, such as channel pruning [10], lightweight model design [25,30], quantization [34] and neural architecture search (NAS) for efficient model [36,32]. Among them, model distillation is another active track, which aims to transfer knowledge or semantic feature information from a large teacher model into a light student model. Pioneered by dark knowledge [13], the main body of recent works [39,2] focuses on using distilling intermediate features to enrich the learnable information for guiding student in different tasks, such as classification [28] and detection [2].

However, a prominent challenging problem for these methods is on how to fill the gap in semantic feature structure between teacher and student. Roughly

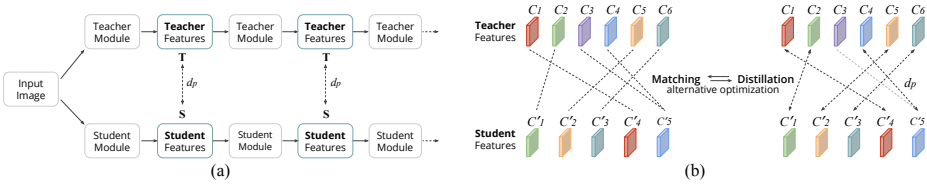

Fig. 1. Matching guided distillation. (a) MGD follows the general distillation paradigm. **T** and **S** are teacher and student feature tensors. d_p is the feature distance function. (b) MGD distills teacher channels (C_i) of intermediate features to student channels (C'_i) by solving a joint matching-distillation problem (alternative optimization).

speaking, the contrast lies in two aspects: 1) The different channel dimensions of feature outputs used for distillation. 2) The different perceptual information or activations between two channel sets. Previous works [28,39,11] overcome these two obstacles by building an adaptive module between hidden layers of teacher and student. Whereas this manner can alleviate the issue, it still has two limitations: 1) The adaptation module introduces more parameters (including weights, gradients and optimizer states) into training [27]. These additional parameters induce the training harder despite the fact that they would be brushed off for model inference. 2) The adaptation module with random initialization or special non-parameter transformation [31,42] isn't friendly for distilling a pre-trained student, because it would potentially disturb the student features. To avoid this break, [39,28] performs stage-wise training by separating optimization into multiple steps. But this way isn't perfect yet because it will plunge training into a cumbersome one.

To crack the limitations of former works, we propose a novel distillation method named Matching Guided Distillation (MGD). As shown in Fig. 1, MGD follows the general distillation paradigm. Given batches of data fed into teacher and student, MGD matches their intermediate channels from distillation position. The motivation is that whether the student has been pre-trained or not, each channel of it should be guided by its high related teacher channel directly to narrow the semantic feature gap. In order to implement element-wise losses, teacher channels would be reduced according to the matching graph. The method for channel reduction is flexible, we propose three manners: sparse matching, random drop and absolute max pooling. Experiments show that all three ways are effective on various tasks. For the whole training, we apply coordinate descent algorithm [35] to alternate between two optimizations for channels matching and weights updating. Furthermore, distilling a pre-trained student using MGD is efficient due to its parameter-free nature as same as training from-scratch.

2 Related Work

Correspondence Problem. Finding optimal correspondence between two sets of instances is a crucial step for a wide range of computer vision problems, such as shape retrieval [16], image retrieval [29], object categorization [6] and video action

recognition [1]. Linear Assignment (LA) is the most classical correspondence problem that can be efficiently solved with Hungarian algorithm [20], unlike the NP-hard quadratic assignment problem [43]. Matching based training losses [7,15] contain the ideology of matching features, but they have a heavy computation. For example, the Wasserstein loss [7] uses the iterated optimization [3] to approximate the matching matrix, its computation cost will dramatically become large along with the growth of feature dimensions. It only can be used for the feature from the last fully-connected layer, so does [15]. In this paper, we treat relationships modeling between teacher and student channels as a LA problem, particularly the min cost assignment problem. The total matching cost function would be minimized by the Hungarian method to achieve a bipartite assignment graph. This graph represents the high related channel pairs between teacher and student feature sets.

Knowledge Distillation. Pioneered by [13], the classic method for knowledge distillation contains two constituents: logits from the last teacher layer used as the soft targets, and Kullback-Leibler divergence loss used to let student match these targets. However, the performance of output distillation is limited due to the very similar supervised signal from teacher model with ground-truth. More works [28,39,2] switch to feature distillation by combining intermediate features together to strongly supervise the student. All these works rely on certain adaptation modules between hidden layers, in order to solve the contrast of semantic feature structures. Feature correlation based methods provide more fine-grained recipes to perform knowledge distillation, such as attention transfer [42,22], neuron selectivity transfer [15]. These works focus on capturing and transferring the spatial information for intermediate feature maps. Another technique fashion for distillation is to design the loss function, including activation transfer loss with boundaries formed by hidden neurons [12], the loss for penalizing structural differences in relations [26]. In this paper, we propose a novel perception that performing distillation after matching intermediate channels between teacher and student. Our proposed approach is intuitive and lightweight. It introduces marginal computation costs during training. In the end, although the work of Jacobian matching-based distillation [31] seems related to ours, it still suffers from the problems discussed in Section. 1. Because it not only uses adaption modules but also a specialized loss function.

Transfer Learning. Commonly fine-tuning is one of the effective methods for knowledge transfer learning. The student has been already pre-trained on a specific domain data, and then it's fine-tuned for another task with priori knowledge. The work [8] finds that the model with random initialization could be trained no worse than using pre-trained parameters. However, the model may suffer from the low capacity trained on a small dataset like Caltech-UCSD Birds 200 [33]. A number of previous works use ImageNet pre-trained models on different tasks such as detection and segmentation [37]. In this paper, we show that MGD can be also used for knowledge transfer learning with a pre-trained student for the further performance improvements in another task, particularly for the fine-grained categorization.

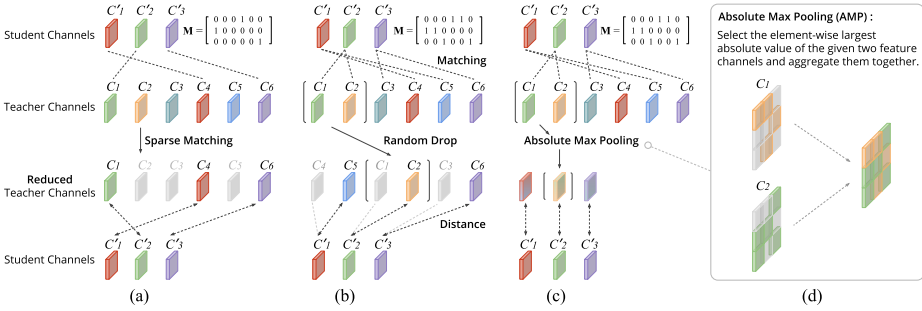

Fig. 2. Channel reduction methods. C_i indicates teacher channels of intermediate features as same as C'_i for student. $\mathbf{M}$ is the matching matrix. We propose three effective methods to play reduction for teacher channels: sparse matching, random drop and absolute max pooling. (a) Sparse Matching. Each student channel only matches one teacher channel. Unmatched teacher channels are directly ignored. (b) Random Drop. Each student channel matches more than one teacher channel. Matched teacher channels with the same student channel are *randomly* dropped to leave just one for guiding student. (c) Absolute Max Pooling (AMP). Each student channel matches more than one teacher channel. AMP picks out the element-wise largest absolute value along the channel axis over a group of matched teacher channels with the same student channel. (d) shows the detail about how AMP works on two channel tensors.

3 Methodology

In this section, we introduce a general formulation of the proposed Matching Guided Distillation (MGD). MGD consists in a parameter-free channel matching module that can guide to shave teacher channels in three effective manners: sparse matching, random drop and absolute max pooling, as shown in Fig. 2.

3.1 Feature Distillation Revisit

We begin by briefly reviewing feature distillation in general formulation. Suppose that 2D images[1] $\mathbf{X}$ are fed into the teacher f_T and student f_S networks that generate intermediate feature sets

$$\mathbf{T} = f_T(\mathbf{X}) \in \mathbb{R}^{C_T \times N}, \ \mathbf{S} = f_S(\mathbf{X}) \in \mathbb{R}^{C_S \times N}, \tag{1}$$

respectively at the target distillation positions. Without loss of generality, we assume the feature maps are of the same spatial size $N = HW$ (height H and width W) but could consist of different number of channels, e.g. $C_T = 512$ while $C_S = 128$. Given a teacher network f_T with frozen parameters, we wish to

[1] Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_i$ and $\mathbf{x}^j$ represents the i^{th} column and j^{th} row of the matrix $\mathbf{X}$ respectively. x_{ij} or $[\mathbf{X}]_{ij}$ denotes the scalar in the i^{th} row and j^{th} column of the matrix $\mathbf{X}$. All non-bold letters represent scalars.

enhance the training of the student networks f_S using the hints from f_T. In a nutshell, the problem of feature distillation seeks for the optimum student network f_S that minimizes the loss of the main task together with feature discrepancy penalty:

$$Loss = L_{task} + \gamma L_{distill}, \tag{2}$$

where γ is a trade-off coefficient for distillation loss. The task loss L_{task}, for example, could be cross-entropy loss for classification or smooth-L_1 loss for object localization. The key to feature distillation is the design of the distillation loss, $L_{distill}$, which ensures the similarity between intermediate features $\mathbf{T}$ and $\mathbf{S}$:

$$L_{distill} = d_p(\sigma_T(\rho(\mathbf{T}, \mathbf{M})), \sigma_S(\mathbf{S})), \tag{3}$$

where σ_T, σ_S, d_p are the teacher and student feature transforms that convert raw feature into an easy-to-transfer form and distance functions respectively. In the past few years, various designs have been proposed to make better use of information contained in teacher networks. In MGD, σ_T is a marginal ReLU as same as [11]. Based on a recent comprehensive review [11] on these design aspects, we build the pipeline by employing the marginal ReLU for teacher transform σ_T:

$$\sigma_T(x) = \max(x, m), \tag{4}$$

where $m < 0$ is a margin value, computed as an expectation value over all training images. Following [11], we choose the partial L_2 loss function to calculate feature distance:

$$d_p(\mathbf{T}, \mathbf{S}) = \sum_{i}^{C} \sum_{j}^{N} \begin{cases} 0 & \text{if } s_{ij} \leq t_{ij} \leq 0 \\ (t_{ij} - s_{ij})^2 & \text{otherwise,} \end{cases} \tag{5}$$

for any pair of matrices $\mathbf{T}, \mathbf{S} \in \mathbb{R}^{C \times N}$ of the same dimension.

3.2 Channel Matching

The distillation loss (Eq. 3) plays a vital role in distilling the knowledge of a complex model into a simpler one. To achieve this goal, the design of distance function (d_p) and feature transforms (σ_T, σ_S) needs to ensure teacher's knowledge can be transferred to student with minimum loss. Despite that various choices have been proposed in the past few years (see [11] for an extensive review), it is still necessary to add an 1×1 convolutional layer or other module on student (σ_S) to bridge the semantic gap between $\mathbf{T}$ and $\mathbf{S}$. The presence of student transform not only adds burden on network complexity but also complicates the training procedure. We propose MGD by completely removing the student transform from the distillation pipeline, i.e. $\sigma_S(\mathbf{S}) = \mathbf{S}$. Instead, we directly match the channel via the reduction operation $\rho(\mathbf{T}, \mathbf{M})$ from teacher to student. This operation is parameter-free and efficient to optimize in training. Below we explain how to establish the correspondence $\mathbf{M}$ across channels and define the implementation of $\rho(\cdot, \cdot)$ in next section.

Given a pair of teacher feature $\mathbf{T} \in \mathbb{R}^{C_T \times N}$ and student one $\mathbf{S} \in \mathbb{R}^{C_S \times N}$, we first encode their pairwise relation in a distance matrix, $\mathbf{D} \in \mathbb{R}^{C_S \times C_T}$, whose element d_{ij} computes the Euclidean distance between i^{th} student and j^{th} teacher channels:

$$d_{ij} = \sum_{k=1}^{N} (s_{ik} - t_{jk})^2. \tag{6}$$

Our goal is to find a binary matrix $\mathbf{M} \in \{0,1\}^{C_S \times C_T}$ that encodes the channel-wise correspondence, where $m_{ij} = 1$ if i-th student and j-th teacher channels are pertinent. In the special case when the teacher and student feature maps are of the same dimension (i.e., $C_T = C_S$), the matching is assumed to be one-to-one and the resulting matrix $\mathbf{M}'$ defines a permutation of C_T channels:

$$\mathit{\Pi} = \Big\{ \mathbf{M}' \in \{0,1\}^{C_T \times C_T} | \sum_{j=1}^{C_T} m'_{ij} = 1, \sum_{i=1}^{C_T} m'_{ij} = 1 \Big\}. \tag{7}$$

In general, we resort to a many-to-one matching as the teacher channel number C_T is often several times more than the student one C_S. In order to make the distillation procedure evenly distributed over feature channels, we further constrain that each student channel has to be associated with $\alpha = \lfloor C_T / C_S \rfloor$ teacher channels. More specifically, the many-to-one balanced matching $\mathbf{M}$ satisfies:

$$\mathit{\Pi}_b = \Big\{ \mathbf{M} \in \{0,1\}^{C_S \times C_T} | \sum_{i=1}^{C_S} m_{ij} = 1, \sum_{j=1}^{C_T} m_{ij} = \alpha \Big\}. \tag{8}$$

This constraint enforces that $\mathbf{M}$ is a wide-shape matrix, where the sum of each column equals to one because each teacher channel can only be connected to one student channel. On the other hand, the sum of each row needs to be α. In another word, each student channel has to be associated with α teacher ones.

Given two sets of feature channels with the associated pairwise distance, the problem of channel matching consists in finding a balanced many-to-one mapping $\mathbf{M}$ such that the sum of matching cost is minimized:

$$\min_{\mathbf{M}} \operatorname{trace}(\mathbf{D}^T \mathbf{M}) = \sum_{i=1}^{C_S} \sum_{j=1}^{C_T} d_{ij} m_{ij}, \text{ subject to } \mathbf{M} \in \mathit{\Pi}_b. \tag{9}$$

Now Eq. 9 is not a standard linear assignment problem, because the Hungarian algorithm works on the square cost matrix. To satisfy this prerequisite, let $\mathbf{D}' = [\mathbf{D}; \cdots; \mathbf{D}] \in \mathbb{R}^{C_T \times C_T}$ to be a square matrix by concatenating α matrices $\mathbf{D}$ vertically[2]. Our solution proceeds by optimizing a standard linear assignment problem to solve out $\mathbf{M}'$ first:

$$\min_{\mathbf{M}'} \operatorname{trace}(\mathbf{D}'^T \mathbf{M}') = \sum_{i=1}^{C_T} \sum_{j=1}^{C_T} d'_{ij} m'_{ij}, \text{ subject to } \mathbf{M}' \in \mathit{\Pi}, \tag{10}$$

[2] For the case when C_T is not divisible by C_S, we simply shave $C_T - \alpha C_S$ dummy teacher channels off the cost matrix. Although this solution is not optimal, we found the result is still promising on several datasets.

using Hungarian algorithm. We then evenly slice the resulting matrix $\mathbf{M}' = [\mathbf{M}'_1; \cdots; \mathbf{M}'_\alpha]$ in row blocks, where each sub-matrix $\mathbf{M}'_i \in \{0,1\}^{C_S \times C_T}$ is of the same size. It's easy to prove that the optimal solution for Eq. 9 is:

$$\mathbf{M} = \sum_{i=1}^{\alpha} \mathbf{M}'_i \in \{0,1\}^{C_S \times C_T}. \tag{11}$$

3.3 Channel Reduction

Once the channel-wise correspondence $\mathbf{M}$ is established, the distillation loss d_p in Eq. 3 would encourage the student channel to mimic the hidden feature of the related teacher channels. Because of the many-to-one nature for the mapping, we discuss below three parameter-free choices for reducing teacher feature via operation $\rho(\mathbf{T}, \mathbf{M})$ to match with student feature.

Sparse Matching. To match the C_T teacher channels with C_S student ones, the straightforward way is to pick an optimal subset of C_S teacher channels and construct a one-to-one matching with the C_S student channels. To do so, we simply formalize another linear assignment problem by introducing $C_T - C_S$ dummy student channels, each of which is put in an infinity distance from any teacher channel. This linear assignment problem is in the same form as Eq. 10 except the distance matrix $\mathbf{D}' \in \mathbb{R}^{C_T \times C_T}$ is constructed by appending $C_T - C_S$ rows of large constant (e.g. $1e10$) to the end of the original $\mathbf{D} \in \mathbb{R}^{C_S \times C_T}$. After applying the Hungarian algorithm on Eq. 10, we could find for each student channel the most relevant teacher one, which are encoded in the first C_S rows of the resulting correspondence matrix, i.e., $\mathbf{M} = \mathbf{M}'_1 \in \{0,1\}^{C_S \times C_T}$. For instance, Fig. 2(a) illustrates an example of matching $C_T = 6$ teacher channels with $C_S = 3$ student ones, where the correspondence matrix $\mathbf{M} \in \{0,1\}^{3\times 6}$ denotes three one-to-one matching pairs. In this case, the reduction operation can thus be defined as:

$$\rho_{\text{SM}}(\mathbf{T}, \mathbf{M}) = \mathbf{M}\mathbf{T} \in \mathbb{R}^{C_S \times N}. \tag{12}$$

Random Drop. The major limitation of the first sparse matching choice is that it only retains a small fraction (C_S/C_T) of information conveyed in the original teacher features. To reduce the information loss, our second choice for teacher reduction is to sample a random teacher channel from the ones associated with each student channel. More specifically, there are α non-zero elements in each row of $\mathbf{M}$ according to the constraint (Eq. 8). The random drop operation modifies the correspondence as $\mathbf{M}^{RD} \in \{0,1\}^{C_S \times C_T}$ by randomly keeping one non-zero element in each row, i.e., $\sum_{j=1}^{C_T} m_{ij}^{RD} = 1$ for any $i = 1, \cdots, C_S$. To have a better understanding, we visualize one case in Fig. 2(b), where the second student channel C'_2 is associated with C_1 and C_2 teacher channels after the channel matching step. In random drop reduction, we randomly pick one of them (e.g. C_2) to match with the student. In order to maximize the randomness, we generate

correspondence matrices $\mathbf{M}_i^{RD}$ independently for different spatial positions of the feature map. The overall reduction operation can be defined as:

$$\rho_{\mathrm{RD}}(\mathbf{T}, \mathbf{M}) = \left[\mathbf{M}_1^{RD}\mathbf{t}_1, \cdots, \mathbf{M}_N^{RD}\mathbf{t}_N\right] \in \mathbb{R}^{C_S \times N}, \tag{13}$$

where $\mathbf{t}_j \in \mathbb{R}^{C_T}$ denotes the j^{th} column of the feature $\mathbf{T}$.

Absolute Max Pooling. Following [11], we place the distillation module before ReLU. Therefore both positive and negative values are transferred from teacher via the partial distance loss d_p to student. We hope the reduced teacher features still take the maximum activations in the same spatial position, including positive (usable) and negative (adverse) information. To reach this purpose, we propose a novel pooling mechanism, named Absolute Max Pooling (AMP), as shown in Fig. 2(c). Given a set of feature activations $\mathbf{x} = [x_1, \ldots, x_C]^T \in \mathbb{R}^C$, the AMP is designed to choose the element that yields the largest magnitude:

$$f_{\mathrm{AMP}}(\mathbf{x}) = \arg\max_{x_i} |x_i|. \tag{14}$$

Similar to the random drop idea, the AMP operation is performed independently for each spatial position $\mathbf{t}_j \in \mathbb{R}^{C_T}$ of the teacher feature map $\mathbf{T} \in \mathbb{R}^{C_T \times N}$. For each student channel $i = 1, \cdots, C_S$, AMP is used to select the most active teacher channel among the associated α ones. Because this teacher-student association has been encoded as the non-zero elements of i^{th} row $\mathbf{m}^i \in \{0, 1\}^{\mathbf{C}_T}$ of matrix $\mathbf{M}$, we can write the overall reduction operation in matrix form as:

$$\rho_{\mathrm{AMP}}(\mathbf{T}, \mathbf{M}) = \left[f_{\mathrm{AMP}}(\mathbf{m}^i \circ \mathbf{t}_j)\right]_{ij} \in \mathbb{R}^{C_S \times N}, \tag{15}$$

where $\circ$ indicates the element-wise product between two vectors. As shown in Fig. 2(d), AMP pools these α feature nodes into single one along the channel axis.

Theoretical Summary. In the theoretical perspective, we describe the understanding on these three channel reduction methods as following. All these methods belong to variants of the LA problem. With the same matrix notation, their goal can be connected as seeking for the optimal matching matrix $\mathbf{M}$ by optimizing certain linear objective. The first sparse matching (Eq. 12) is a simple modification of the original Hungarian algorithm. The second random drop (Eq. 13) and the last absolute max pooling (Eq. 14) ideas progressively improve sparse matching by different approaches to reduce information loss from $\sigma_{\mathbf{T}}$. To better understand and verify this insight, we have done many experiments and ablation studies in Section. 4.

3.4 Implementation Details

Distillation Position. In our experiments, we use contemporary models in recent years, including ResNet [9], MobileNet-V1 [14], -V2 [30] and ShuffleNet-V2 [25]. Commonly, these models contains four stages, each of which is composed of repeated unit blocks, as shown in Fig. 3. We apply distillation in the last unit

block of each stage. In the cases of distilling MobileNet-V2 and ShuffleNet-V2, the first stage is skipped, because their first stage is only a convolutional layer and also its feature map size ($H \times W$) is distinct from teacher's.

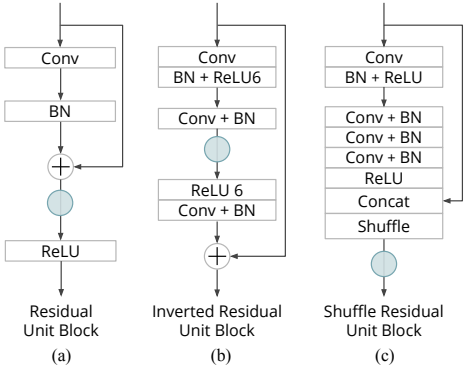

Fig. 3. The distillation position in a unit block is pinpointed by a color point. (a) For the standard residual unit in ResNet and MobileNet-V1, we use the output before ReLU. (b) For the inverted unit in MobileNet-V2, we use the intermediate feature before ReLU6. Although the output from the whole block isn't activated by ReLU-like operations too, the channel number is narrowed down into an unusable one for distillation. (c) The feature after channel shuffle is used for the shuffle unit.

Coordinate Descent. To optimize the whole system (Eq. 3), we use Coordinate Descent algorithm [35] by alternating between solving the combinatorial matching problem and updating network weights. Postulating the matching is solved, we plug $\mathbf{M}$ in Eq. 3 and employ Stochastic Gradient Descent (SGD) to update student network weights f_S. After several epochs or iterations of training with SGD, the student is switched into evaluation mode without learning. Then we feed a dataset that is randomly sampled from *training* data into student and teacher, in order to update matching $\mathbf{M}$ that optimizes Eq. 9 for the next training rounds. Solving Eq. 9 takes the computational complexity of $O(C_T^3)$. This step introduces negligible cost in our implementation.

4 Experiments

Datasets. We run a number of major experiments to compare our MGD with other methods, using sparse matching (SM), random drop (RD) and absolute max pooling (AMP). We evaluate them in multiple tasks, including large-scale classification, fine-grained recognition with transfer learning, detection and segmentation. We have done on four popular open datasets. **CIFAR-100** [19] is composed of 50,000 images within 100 classes, and has a fixed input size of 32×32. ImageNet (**IN-1K**) [4] has 1.2 million training images and 50,000 validaton images in 1000 object categories. Birds-200-2011 (**CUB-200**) [33] is a dataset for categorizing the fine-grained objects, which contains 11,788 images of 200 bird classes. CUB-200 is used for testing MGD in transfer learning. **COCO** [24] is a standard object detection and instance segmentation benchmark.

Experimental Setting. Classification tasks use a standard training scheme. For the pre-trained student, we set init learning rate (LR) in 0.01, otherwise 0.1 for training from scratch. On CIFAR-100, the number of total epochs is 200,

LR is dropped twice by 0.1 at 100^{th} and 150^{th} epoch. On IN-1K and CUB-200, total epochs number is 120, and LR is dropped by 0.1 after every 30 epochs. Momentum is 0.9 and weight decay is 5e-4. We randomly crop 224 from 256 and then perform horizontal flip in IN-1K and CUB-200.

For object detection and instance segmentation, we follow the configurations in `Detectron2` [37]. The input size of training image is restricted in maximum size of 1333 and minimal size of 800. Horizontal flip is the only data augmentation. We use 8 GPUs and set mini-batch size of 2 images for each GPU. The total iterations number is 90k. We use standard 1x and 2x training schedules settings.

The number of images for solving $\mathbf{M}$ depends on the dataset scale. We randomly sample 20k images on IN-1K and use all the training images on CIFAR-100 and CUB-200. In the detection and segmentation task, about 5k images are used for updating $\mathbf{M}$.

4.1 Main Results

Classification. Following the competitor [11], student models are randomly initialized in tasks on CIFAR-100. We use WideResNet (WRN) [41] as the teacher, which is in the model setting of depth 28 and wide factor 4, indicated by s:28-4. The student has multiple settings in different compression aspects: depth, wide factor, and architecture. Results comparison is shown in Table 1. WRN with s:16-4 has the same wide factor but smaller depth, so its channels are in same number of teacher's at the same distillation positions. Since there is no need to reduce teacher channels, we only use MGD-SM to achieve the lowest error rate 20.53%. WRN with s:28-2 has the same depth but a larger wide factor, thus teacher channels need to be reduced. The results show that MGD-SM is better than [11] with +0.76%, MGD-RD is a little worse with +0.23%, and MGD-AMP achieves the lowest error of 21.12%. In experiments with s:16-2, whose both depth and wide factor are smaller than those of teacher, MGD-SM & RD are worse than the competitor and MGD-AMP is competitive. Another setting for student is using a different architecture, ResNet with s:56-2, which is deeper but with the same wide factor. MGD has the competitive results under this setting as well. The Table 1 shows that MGD can handle various compression types for student.

model setting	method	error.	model setting	error.	model setting	error.	model setting	error.
WRN w/ s:28-4	Teacher	21.09	-	21.09	-	21.09	-	21.09
WRN w/ s:16-4	Baseline	22.72	WRN w/ s:28-2	24.88	WRN w/ s:16-2	27.32	ResNet w/ s:56-2	27.68
	KD [13]	21.69		23.43		26.47		26.76
	FitNets [28]	21.85		23.94		26.30		26.35
	AT [42]	22.07		23.80		26.56		26.66
	Jacobian [31]	22.18		23.70		26.71		26.60
	AB [12]	21.36		23.19		26.02		26.04
	Overhaul [11]	20.89		21.98		24.08		**24.44**
	MGD - SM	**20.53**		21.22		24.72		25.20
	MGD - RD	-		22.21		24.64		26.01
	MGD - AMP	-		**21.12**		**24.06**		24.91

Table 1. Comparison of error rates (%) with various model settings on CIFAR-100. We average 5 runs to report final results.

In large-scale classification on IN-1K, as shown in Table 2, we use ResNet-152 to distill ResNet-50. Since the channel number is identical in each stage of them, we only investigate MGD-SM. The overall results from other works, except the Overhaul method, MGD beats other methods with maximum 1.45% improvement in top-1 accuracy. In the case of distilling MobileNet-V1 by ResNet-50, using MGD-SM has similar result with [11]. MGD w/ AMP is the best overall methods.

model	method	top-1 err.	top-5 err.
ResNet-152	Teacher	21.69	5.95
ResNet-50	Baseline	23.85	7.13
	KD [13]	22.85	6.55
	AT [42]	22.75	6.35
	AB [12]	23.47	6.94
	Overhaul [11]	**21.65**	5.83
	MGD - SM	22.02	**5.68**
ResNet-50	Teacher	23.85	7.13
MobileNet-V1	Baseline	31.13	11.24
	KD [13]	31.42	11.02
	AT [42]	30.44	10.67
	AB [12]	31.11	11.29
	Overhaul [11]	28.75	9.66
	MGD - SM	28.79	9.65
	MGD - RD	29.55	10.02
	MGD - AMP	**28.53**	**9.65**

Table 2. Comparison of error rates (%) with MGD and previous works in large-scale classification on IN-1K.

model	method	top-1 err.	top-5 err.
ResNet-50	Teacher	20.02	6.06
MobileNet-V2	Baseline - FT	24.61	7.56
	Baseline - FS	54.97	27.0
	KD [13]	23.52	6.44
	AT [42]	23.14	6.97
	AB [12]	23.08	6.54
	Overhaul [11]	21.69	5.64
	MGD - SM	21.82	5.68
	MGD - RD	21.58	5.92
	MGD - AMP	**20.64**	**5.38**
ShuffleNet-V2	Baseline - FT	31.39	10.9
	Baseline - FS	66.28	35.7
	KD [13]	28.31	9.67
	AT [42]	28.58	9.29
	AB [12]	28.22	9.48
	Overhaul [11]	27.42	8.04
	MGD - SM	28.22	8.85
	MGD - RD	27.71	8.72
	MGD - AMP	**25.95**	**7.46**

Table 3. Comparison of error rates (%) on CUB-200. The students are pre-trained on IN-1K. FT: fine-tune. FS: from-scratch.

Transfer Learning. We use fine-grained categorization on CUB-200 to investigate distillation for transfer learning. We implement MGD and our competitor Overhaul for using ResNet-50 to distill light students. The teacher ResNet-50 has been pre-trained on IN-1K and then trained on CUB-200. We use two prevailed lightweight models, MobileNet-V2 and ShuffleNet-V2. Conspicuously, they have fewer parameters than ResNet-50. The main results have been shown in Table 3. We have two baselines for each student: one is trained from scratch (Baseline-FS), and the other is fine-tuned (Baseline-FT) from IN-1K. We summarize the experimental results in two folds.

First, the results of two baselines show that transfer learning from a general data domain is helpful to the specific task. Students pre-trained on IN-1K could bring ∼30% accuracy improvements at least.

Second, we adopt our three reduction methods of MGD to compare with Overhaul [11]. The experimental phenomenon of two students are same. MGD-AMP beats the Overhaul with 1.05% improvement and also makes MobileNet-V2 almost have the similar performance with teacher. ShuffleNet-V2 seems difficult to be distilled, there is a unignored gap with teacher. But MGD-AMP stably performs best to help ShuffleNet-V2 achieve the maximum top-1 error decrease from 31.39% to 25.95%.

backbone	method	AP^{bbox}	AP^{bbox}_{50}
ResNet-50	Teacher	36.37	55.37
ResNet-18	Baseline	30.30	47.53
	Overhaul [11]	30.02	46.95
	MGD - AMP	**31.15**	**48.60**
MobileNet-V2	Baseline	26.54	42.14
	Overhaul [11]	26.62	42.01
	MGD - AMP	**27.45**	**43.10**

(a) RetinaNet, **1**x schedule + **single**-scale

backbone	method	AP^{mask}	AP^{mask}_{50}
ResNet-50	Teacher	33.5	54.1
ResNet-18	Baseline	26.1	43.7
	Overhaul [11]	26.3	43.9
	MGD - AMP	**26.9**	**44.2**
MobileNet-V2	Baseline	27.1	44.8
	Overhaul [11]	27.0	44.8
	MGD - AMP	**27.6**	**45.1**

(b) EmbedMask, **1**x schedule + **single**-scale

backbone	method	AP^{bbox}	AP^{bbox}_{50}
ResNet-50	Teacher	37.01	56.03
ResNet-18	Baseline	30.78	47.88
	Overhaul [11]	30.26	47.22
	AT [42]	30.54	47.65
	AB [12]	31.32	48.70
	MGD - AMP	**31.38**	**48.79**

(c) RetinaNet, **1**x schedule + **multi**-scale

backbone	method	AP^{bbox}	AP^{bbox}_{50}
ResNet-50	Teacher	38.73	56.72
ResNet-18	Baseline	34.63	53.08
	Overhaul [11]	34.42	52.90
	AT [42]	34.43	52.97
	AB [12]	34.92	53.50
	MGD - AMP	**35.10**	**53.76**

(d) RetinaNet, **2**x schedule + **multi**-scale

Table 4. Comparison of object detection and instance segmentation results. We distill lightweight backbones of RetinaNet for object detection (a, c, d) and EmbedMask for instance segmentation (b) on COCO. Here we experiment only with AMP because it's the best operation among three reduction manners.

No matter which reduction method we use to accomplish distillation for transfer learning, all the experimental results show that MGD is more friendly for distilling a pre-trained student.

Object Detection & Instance Segmentation. To verify the generalization of MGD, we extend it to object detection and instance segmentation. RetinaNet [23] is a modern one-stage detector, which has excellent performance in both precision and speed. EmbedMask [40] is a novel framework for instance segmentation, which utilizes embedding strategy to generate instance masks on a unified one-stage structure. In this section, we experiment with RetinaNet and EmbedMask respectively, using three different backbones: ResNet-50 as teacher, ResNet-18 and MobileNet-V2 as students. All these backbones are pre-trained on IN-1K. We train the models on COCO `train2017` set and test them on `val2017`. Baselines are trained without distillation. As comparison, we also train with [11,42,12] under same configurations. The main results are presented in Table 4. For object detection, in both cases of distilling ResNet-18 and MobileNet-V2, MGD has stable improvements by $0.47 - 0.91$ point. In segmentation, in the case of ResNet-18, we freeze the first two backbone stages to avoid `OOM`. MGD can bring about 0.8 mAP point for ResNet-18 and 0.6 point for MobileNet-v2, it outperforms the competitor. These results prove that MGD works more stable than previous works in object detection and instance segmentation.

4.2 Ablation Study

Frequency of Updating M. To find the best practices, we experiment on how intense the frequency of updating $\mathbf{M}$ could affect the MGD performance. Here all experiments adopt MGD-AMP. The baseline is updating $\mathbf{M}$ in the end of every training epoch (frequency=1) as same as validation does. In Fig. 4, updating in every 2 training epochs achieves the best results both in MobileNet-V2 and

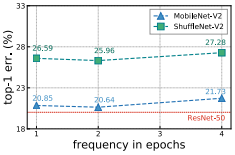

Fig. 4. Sensitivity to frequency of updating **M** on CUB-200.

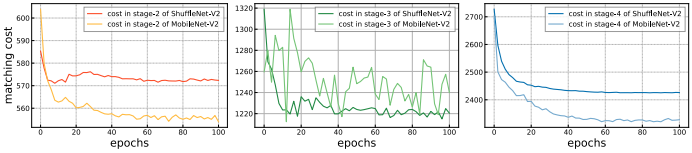

Fig. 5. Curves of matching cost in three distillation stages of MobileNet-V2 and ShuffleNet-V2, which are distilled by ResNet-50 on CUB-200.

ShuffleNet-V2 on CUB-200. If the frequency becomes larger than 2 in epochs (frequency=4), it will produce higher error rates. This study suggests that MGD should not be updated either fast or lazily. On IN-1K, we update **M** after every epoch due to its large dataset scale.

Absolute Max Pooling. Absolute max pooling (AMP) leads to largest improvements compared to proposed SM and RD. Now we compare it with basic pooling operations, max pooling (MP) and average pooling (AvgP), to perform reduction along the channel axis. Table 5 shows that AMP behaves stably better than MP and AvgP. AvgP performs worst, because average pooling operation is easily to counteract the sharpness feature for a group of channel tensors. Albeit MP works closely with AMP, it's still not perfect because it would shave negative feature values by positive ones. This result shows the effect of AMP for preventing feature information loss from reduction. For a better illustration, we have a fundamental and intuitive experiment on these three pooling operations in Section. A1.1.

With *vs.* Without Matching. This ablation checks the importance of channel mathing mechanism. We remove matching process and simply use AMP as a feature reducer along channel axis. The right table shows the results of MGD with and without channels matching. It proves the effectiveness of channels matching in MGD. Both of distilling MobileNet-V2 and ShuffleNet-V2 without matching are worse than that with matching about 1.5% in top-1 error.

model	top-1 err.	
with matching?		✓
ResNet-50	20.02	
MobileNet-V2	22.10	**20.64**
ShuffleNet-V2	27.62	**25.95**

Capacity Analysis. Next, we illustrate MGD is more efficient for training than other methods. We investigate the capacity of joint training with MGD and [11]. In experiments, we use four GeForce 1080Ti GPU cards to run training. Under the same experimental settings, MobileNet-V2 has less parameters and memory consumption than teacher without distillation. Table 6 shows [11] brings too many parameters to cause OOM. As well known, additional distillation module not only bring the learnable weights for training, but also additional gradients and optimizer states into GPU memory [27]. In contrast, training with MGD has little bit of additional parameters due to its basic nature of parameter-free. Moreover it has better results.

Optimization Analysis. In this part, we analyze MGD in branches of visualization for understanding MGD with comprehensive vistas. We set our experiments to check it in three aspects: matching cost, status of updating **M**, and features matching & reduction.

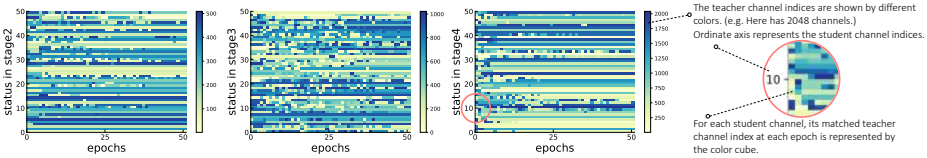

Fig. 6. Status of updating M in three distillation positions. Here we randomly select fifty student channels to check out their matched target channels, which are represented by the small cubes within different colors.

model	top-1 err.		
	MP	AvgP	AMP
ResNet-50	20.02		
MobileNet-V2	21.63	22.19	**20.64**
ShuffleNet-V2	26.20	27.27	**25.95**

Table 5. Comparison of three pooling operations for channel reduction.

model	*bs*	method	memory	parameters	top-1 err.
ResNet-50	256	Teacher	8.10	25.1	20.02
MobileNet-V2	256	Student	5.01	3.5	24.61
	128	Overhaul [11]	OOM	31.5	21.69
	128	MGD - AMP	11.8	**29.1**	**20.64**

Table 6. Capacity analysis. Memory consumption is measured by gigabyte. Parameters is in millions. Here *bs* indicates batch size.

First, Fig. 5 shows the descent curves of matching cost in three distillation positions. We track the sum of matching costs in every 2 epochs when distilling MobileNet-V2 and ShuffleNet-V2 with MGD-AMP. The curves show that all the total costs are in the trend of descent during training. This phenomenon is expected because the more related matched features are, the smaller their matching cost becomes.

Second, Fig. 6 shows the updating status of **M** in distilling MobileNet-V2 on CUB-200. Due to the massive channel number of intermediate features, we randomly select fifty student channels to visualize the updating status of **M**. All the three sub-figures have a common view that at the beginning, most of matching targets of each student channel change dramatically. Then they will become stable after several training epochs. This result concludes that coordinate descent is effective and friendly for the joint optimization with SGD.

Third, Fig. A1 in Appendix checks out the intermediate results of MGD in multiple tasks. In order to check the rightness of matching status, we use the intermediate features for visualization at the *earlier* training iterations. We can conclude the matching results can be trusted for guiding student to induce the better results.

5 Discussion and Future Work

We have presented MGD as an effective distillation method within the parameter-free nature, and evaluated its three channel reduction ways in various tasks. We also experiment in multiple perspectives of ablation study to verify its effect. In the future, it's possible to supervise student in a dense manner, for example, using more than four positions to perform distillation with MGD.

References

1. Brendel, W., Todorovic, S.: Learning spatiotemporal graphs of human activities. In: ICCV (2011)
2. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: NIPS (2017)
3. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: NIPS (2013)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
6. Duchenne, O., Joulin, A., Ponce, J.: A graph-matching kernel for object categorization. In: ICCV (2011)
7. Frogner, C., Zhang, C., Mobahi, H., Araya, M., Poggio, T.A.: Learning with a wasserstein loss. In: NIPS (2015)
8. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: ICCV (2019)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
10. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: ICCV (2017)
11. Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., Choi, J.Y.: A comprehensive overhaul of feature distillation. In: ICCV (2019)
12. Heo, B., Lee, M., Yun, S., Choi, J.Y.: Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In: AAAI (2019)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv:1503.02531 (2015)
14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
15. Huang, Z., Wang, N.: Like what you like: Knowledge distill via neuron selectivity transfer. arXiv:1707.01219 (2017)
16. Huet, B., Cross, A.D., Hancock, E.R.: Graph matching for shape retrieval. In: NIPS (1999)
17. Johnson, J., Karpathy, A., Fei-Fei, L.: DenseCap: Fully convolutional localization networks for dense captioning. In: CVPR (2016)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)
19. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
20. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval research logistics quarterly (1955)
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998)
22. Lee, S., Song, B.C.: Graph-based knowledge distillation by multi-head self-attention network. arXiv:1907.02226 (2019)
23. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
24. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)

25. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: ECCV (2018)
26. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: CVPR (2019)
27. Pudipeddi, B., Mesmakhosroshahi, M., Xi, J., Bharadwaj, S.: Training large neural networks with constant memory using a new execution algorithm. arXiv preprint arXiv:2002.05645 (2020)
28. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv:1412.6550 (2014)
29. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. IJCV (2000)
30. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenet V2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
31. Srinivas, S., Fleuret, F.: Knowledge transfer with jacobian matching. arXiv:1803.00443 (2018)
32. Tan, M., Le, Q.V.: EfficientNet: Rethinking model scaling for convolutional neural networks. arXiv:1905.11946 (2019)
33. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
34. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: Haq: Hardware-aware automated quantization with mixed precision. In: CVPR (2019)
35. Wright, S.J.: Coordinate descent algorithms. Mathematical Programming (2015)
36. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: CVPR (2019)
37. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
38. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: XLNet: Generalized autoregressive pretraining for language understanding. arXiv:1906.08237 (2019)
39. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: CVPR (2017)
40. Ying, H., Huang, Z., Liu, S., Shao, T., Zhou, K.: Embedmask: Embedding coupling for one-stage instance segmentation
41. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
42. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: ICLR (2017)
43. Zhou, F., De la Torre, F.: Factorized graph matching. In: CVPR (2012)

Appendix

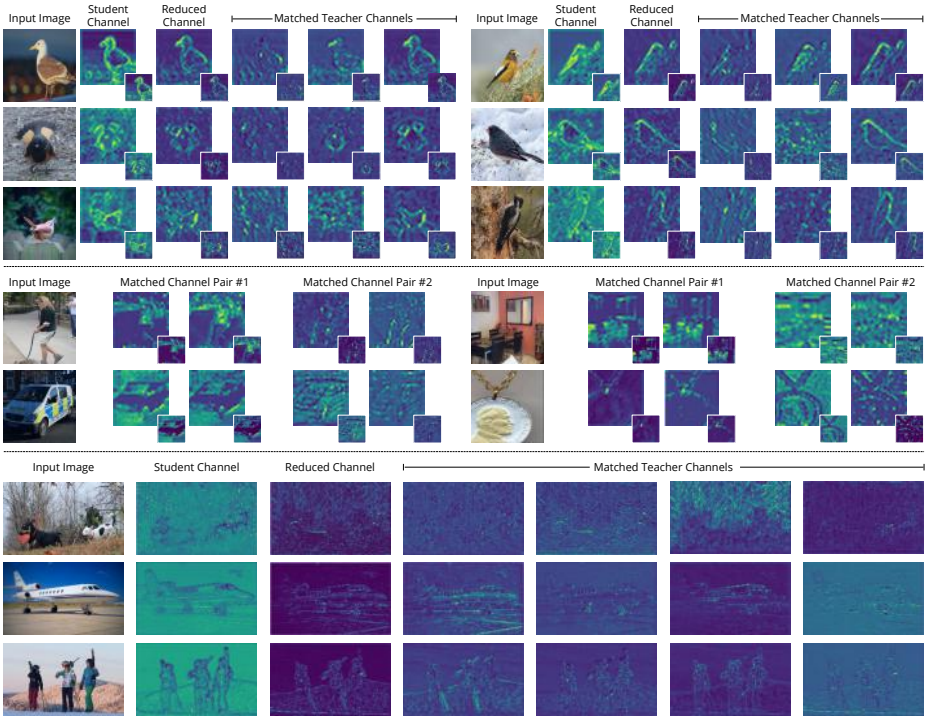

Fig. A1. Channels matching with reduction. The visualization has three parts separated by two dash lines. The first part (top) shows the matching results of *stage-2* in MobileNet-V2 on CUB-200. The channel tensors are visualized in two square patches: small one is in original size of 28×28, the large one is generated by resizing small patch into the input size of 224×224. Each student channel matches three teacher channels. The second part (middle) shows the intermediate matching results in distilling ResNet-50 on IN-1K. Here we find the one-to-one match pair because student has the same channel number with teacher. We randomly select two pairs to visualize. The last part (bottom) shows the results in distilling ResNet-18 on COCO `train2017` set. Each student channel matches four teacher channels. According to this whole visualization, we can easily conclude that the semantic features activations are same between student channels and reduced channels generated by AMP operation.

A1.1 Analysis of Pooling Operations

In order to figure out why the absolute max pooling (AMP) stably works better than average pooling (AvgP) and max pooling (MP) when performing features reduction, we do a fundamental experiment in this part. In Fig. A2, there are two input images (first column from left). First, we build a very simple convolutional

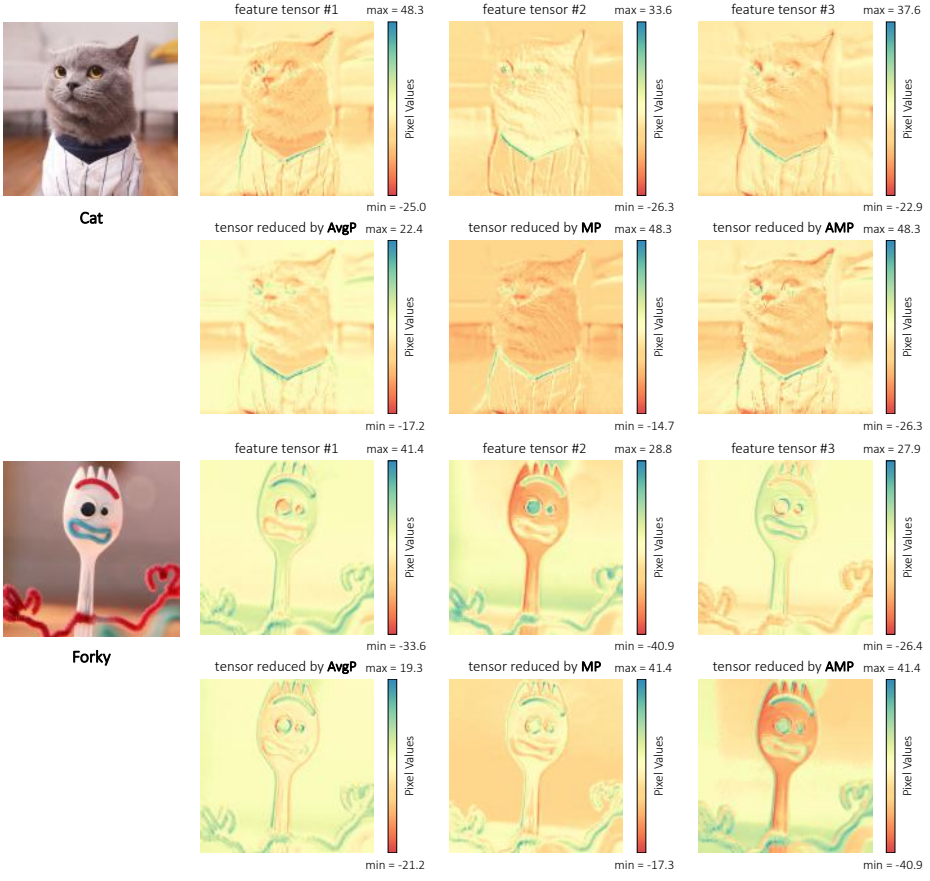

Fig. A2. Comparison of pooling operations. All the feature tensors are normalized into $[0, 1]$ for visualization in order to clearly compare their textures in pixel level and degree. But their min and max values in the color bars use original pixel values without normalization.

network (e.g. LeNet-7 [21]) with random initialization to extract features. Then we select[3] three high-related tensors (in the same row with input images), which have similar semantic feature structures[4] with each other. After using AvgP, MP and AMP operations to perform reduction, we achieve three reduced tensors of each example.

In the case of `Cat`, although AvgP keeps the responses of collar and eyes, it loses the edge activations of right shoulder. MP works well, but its responses of eyes are too weak and also its responses of head texture (including background) are stronger than those of three original features.

[3] This behavior imitates that three teacher channels have been matched with one student channel.

[4] The definition of similar feature structures is made according to their high responses in feature maps.

In the case of `Forky`, AvgP erases the face-body responses from 2^{th} feature. MP not only shades the negative face-body pixels, but also loses the activations of mouth.

Overall, AMP works stably on keeping all the negative and positive texture responses. Moreover, it has ability to hold a good balance between objective and background. This result concludes that AMP works better than both of AvgP and MP for aggregating features. It's possible to use AMP as an alternative general operation for other tasks. For example, in the video classification, AMP can be used to aggregate/pool features along the temporal dimension.