16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

Gradual Sampling Gate for Bidirectional Knowledge Distillation

Soma Minami, Takayoshi Yamashita, Hironobu Fujiyoshi
Chubu University
1200 Matsumotocho, Kasugai, Aichi, Japan
{minami@mprg, yamashita@isc, fujiyoshi@isc}.cs.chubu.ac.jp

Abstract

Knowledge distillation is an efficient approach for model compression. It is based on a unidirectional scheme that transfers knowledge from a large, pre-trained network to a smaller one. A bidirectional scheme was recently proposed that achieved a higher performance than a unidirectional distillation. However, network training is disturbed at the early training stage of bidirectional distillation by the transfer of knowledge between them. We propose a "gradual sampling gate" that controls soft target loss by referring to the training accuracy of each network. Our bidirectional distillation method can improve the accuracy without increasing the computational cost. To evaluate our method, we compare classification accuracies with several network models (ResNet32, ResNet110, Wide ResNet, and DenseNet) over various datasets (CIFAR-10, CIFAR-100, SVHN, and Tiny ImageNet). Experimental results show that our method can effectively train networks and achieve higher accuracies.

1 Introduction

Deep neural networks are very successful in performing several computer vision tasks. In many cases, increasing the number of layers and channels yields a high performance model. However, increasing the model parameters increases memory usage and computational costs. On the other hand, a network that has few parameters is desirable in the deployment phase; however, a smaller network's performance is lower than a larger network's performance. Therefore, gaining a high performance with a smaller network is necessary.

Knowledge distillation [8] is one approach to solving the problem. This approach trains a *student network*, which has few parameters, using a *teacher network*, which is a larger, pre-trained model, to help train the student network. The student network is trained on the basis of soft targets, the teacher network's predictions, hard targets, and one-hot correct labels. The student imitates the teacher's class probabilities. Knowledge distillation can transfer the network's knowledge from a teacher to a student more effectively.

Another network distillation framework is deep mutual learning (DML) [19]. DML uses a number of student networks that transfer their knowledge mutually. This bidirectional distillation approach outperforms unidirectional approaches. Furthermore, DML improves the accuracy by collaborative learning of both a large and a small model and small models of the same architecture. However, because the parameters of each network are randomly initialized, DML transfers soft targets whose probability are close to random noise at the early stage of training. Such soft targets would disturb training.

We propose a *gradual sampling gate* (GSG) that controls the amount of knowledge that is transferred between the networks. At the initial training epoch, the GSG suppresses soft targets that disturb the network training by controlling the soft target loss depending on the training accuracy of each network. Moreover, soft target loss is stochastically sampled from mini-batch samples. Our method is independent of internal structure by transferring soft targets from the output layer. Therefore, the GSG is applicable to any model architecture.

To evaluate our method, we conducted experiments with various datasets (CIFAR-10, CIFAR-100 [10], SVHN [12], and Tiny ImageNet [1]) and various models (ResNet32, ResNet110 [7], Wide ResNet [16], and DenseNet [9]). Experimental results demonstrate that the GSG outperforms DML. We also verified the effects of increasing the soft target loss throughout the training process and stochastically sampled the soft target loss.

2 Related Work

2.1 Knowledge Distillation

Distillation transfers a large, pre-trained model's knowledge to a small model. Typically, a teacher network transfers the knowledge to the output layer of a student network [4, 8]. Posterior probability distributions that output from a network contain feature representation. Therefore, a student learns from a teacher's output probability as a true label, which enables us to transfer the underlying feature representations of the teacher to the student. However, probability values output from a network have extremely low values, except for the top-1 class because of a softmax function.

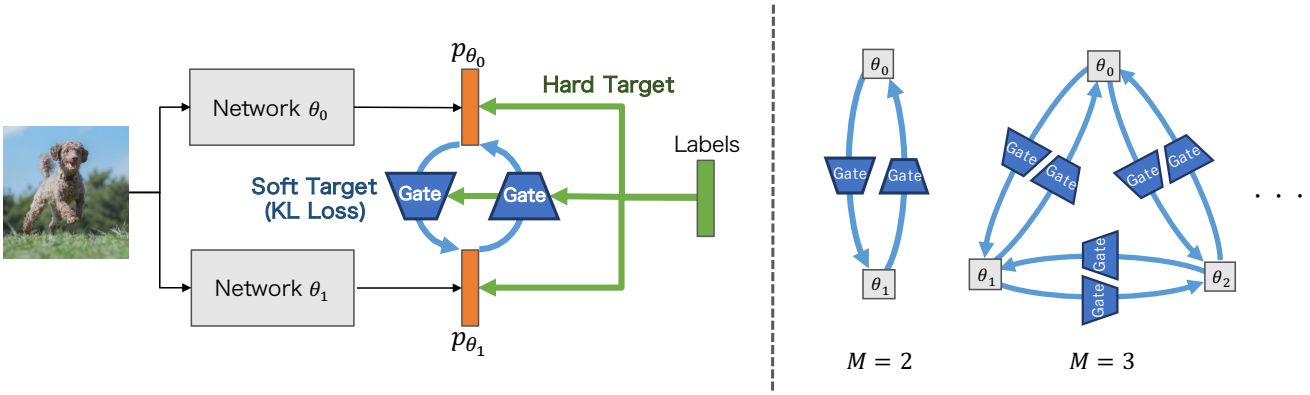

Figure 1: Overview of our method. Left: a training scheme with a gradual sampling gate (GSG) on two networks. Right: paths for transferring knowledge on the N-models.

Bucila et al. [4] use logits before applying softmax as a soft target. This approach facilitates the transfer of knowledge of all classes. Hinton et al. [8] proposed introducing a temperature parameter into a softmax function to effectively transfer the knowledge. However, there is a way to transfer the knowledge from a hidden layer as well as an output layer[14, 17]. By transferring from the hidden layer, the teacher's knowledge can be transferred effectively even in a deeper layer model [14]. Zagoruyko et al. [17] proposed an approach that transfers an underlying feature representation of a teacher network to a student network as an attention. Distillation has recently been connected to curriculum learning [3] in information learning theory [11]. Furthermore, it has been applied to object detection [5], domain adaptation [6], and text-to-speech [13].

2.2 Bi-Directional Distillation

Deep mutual learning [19] is a bidirectional distillation method. DML uses a number of student networks that teach each other collaboratively and do not require a teacher network. As a criterion for knowledge transfer between these networks, Kullback-Leibler (KL) divergence is used to equalize probability distributions output from each network. DML is related to entropy-regularization-based approaches to finding wide minima on loss landscape [19]. DML has been applied to acceleration of large scale distributed neural network training [2] and re-identification [18].

3 Method

In this section, we introduce the details of our gradual sampling gate (GSG). Figure 1 overviews the proposed network architecture. The GSG stochastically controls the soft target loss by introducing the gate architecture in a connection between the θ_0 and θ_1 networks. In early stages of training, each network learns individually with only hard targets because the GSG cuts off the soft target loss. During a training process, networks learn collaboratively by increasing the amount of soft targets.

3.1 Gradual Sampling Gate

To calculate soft target loss, the GSG does not use every sample. Instead, the number of samples used to calculate soft target loss is controlled based on the accuracy of a network θ_n. When the accuracy rate of the network is low, soft target loss is computed from only a few batch samples. Meanwhile, many batch samples are used when it is high.

Figure 2 illustrates the procedures of the GSG. A soft target loss L_i^{soft} of a sample i in batch samples $\mathcal{B} = \{\hat{y}_i, \boldsymbol{x}_i\}_{i=1}^N$ is masked by random variable $r_i \in \{0, 1\}$ taken from the Bernoulli distribution. r_i is formulated as follows:

$$\delta_{a,b} = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases} \tag{1}$$

$$acc = \frac{1}{N} \sum_{i}^{N} \delta_{\hat{y}_i, y_{(\boldsymbol{x}_i)}} \tag{2}$$

$$r_i \sim Bernoulli\,(acc)\,, \tag{3}$$

where acc is the accuracy rate of the network to which the other network transfers the soft target, $\hat{y}$ is the teacher label, y is the class label predicted by the network, and δ is the delta function. r tends to become 1 when the network accuracy is high.

3.2 Loss Function

We describe how the GSG trains the networks θ_0 and θ_1 collaboratively. The output probability of class C

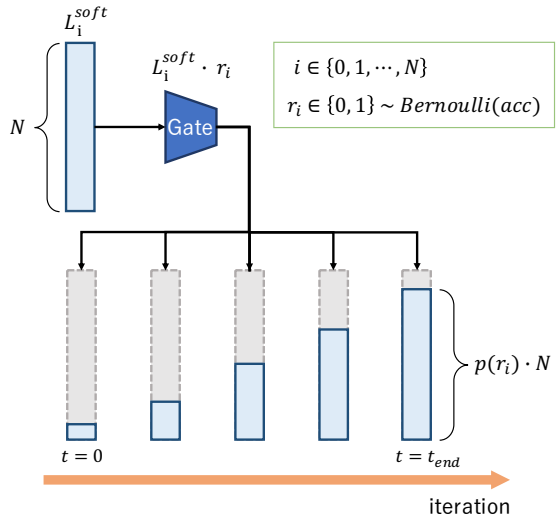

Figure 2: Overview of stochastic and gradual sampling. More details can be found in Section 3.1. t_{end} indicates the end of iteration.

from networks θ_0 and θ_1 with respect to the ith sample in batch samples $\mathcal{B} = \{\hat{y}_i, \boldsymbol{x}_i\}_{i=1}^N$ is given by:

$$p_{\theta_n}^c(\boldsymbol{x}_i) = \frac{\exp\left(z^c(\boldsymbol{x}_i)\right)}{\sum_{j=1}^C \exp\left(z^j(\boldsymbol{x}_i)\right)}, \tag{4}$$

where z refers to logits, i.e. a unnormalized output of the network. $p_{\theta_n}^c(\boldsymbol{x}_i)$ is given by calculating a softmax from z.

In order to estimate the difference between the output probability and the true label, we use cross entropy loss as hard target loss:

$$L_{CE}^{\theta_n} = -\sum_{i=1}^N \sum_{c=1}^C \delta_{c,\hat{y}_i} \log\left(p_{\theta_n}^c(\boldsymbol{x}_i)\right). \tag{5}$$

At the initial epoch, networks are trained using only hard target loss $L_{CE}^{\theta_n}$.

We used KL divergence as soft target loss in order to estimate the difference between the output probability of θ_0 and θ_1:

$$D_{KL}(\boldsymbol{p}_{\theta_n} || \boldsymbol{p}_{\theta_{1-n}}) = \sum_{c=1}^C p_{\theta_{1-n}}^c(\boldsymbol{x}) \log \frac{p_{\theta_{1-n}}^c(\boldsymbol{x})}{p_{\theta_n}^c(\boldsymbol{x})}. \tag{6}$$

In training of DML, the soft target loss is computed from D_{KL} in all batch samples. On the other hand, in our proposed method, GSG chooses samples stochastically depending on the training accuracy of network θ_n as follows:

$$L_{KL}^{\theta_n} = \frac{1}{N} \sum_{i=1}^N r_i \cdot D_{KL}(\boldsymbol{p}_{\theta_n}(\boldsymbol{x}_i) \| \boldsymbol{p}_{\theta_{1-n}}(\boldsymbol{x}_i)) \tag{7}$$

Algorithm 1 Gradual Sampling Gate

Input: Training data $\mathcal{D}$, Training epochs τ;
Initialize: $t = 1$, Randomly initialize θ_0, θ_1;
  while $t \leq \tau$ **do**
    Compute predictions $p_{\theta_1}^c$ and $p_{\theta_2}^c$ by (4);
    Compute loss function L_{θ_1} and L_{θ_2} by (9);
    Compute gradients and update parameters θ_0 and θ_1;
    $t \Leftarrow t + 1$
  end while

$$r_i \sim Bernoulli\left(\frac{1}{N}\sum_{i=1}^N \delta_{\hat{y}_i, y_n(\boldsymbol{x}_i)}\right). \tag{8}$$

The main difference between DML and GSG is stochastic sampling by KL loss with r_i.

The overall loss function L_{θ_n} is defined by

$$L_{\theta_n} = L_{CE}^{\theta_n} + L_{KL}^{\theta_n}. \tag{9}$$

At initial epoch, each network learns with only hard targets. As the network's accuracy improves, soft targets are also used.

3.3 Generalization for N-Models

The soft target loss function that is defined by Eq.(7) applies to only two networks, but our method can be extended to more networks (see Fig. 1). The loss function of the network $\theta_n \in \{\theta_0, \theta_1, \cdots, \theta_M\}$ can be extended as follows:

$$L_{KL}^{\theta_n} = \frac{1}{M-1} \sum_{l=1, l\neq n}^M \frac{1}{N} \sum_i^N r_i \cdot D_{KL}(\boldsymbol{p}_{\theta_n}(\boldsymbol{x}_i) \| \boldsymbol{p}_{\theta_l}(\boldsymbol{x}_i)). \tag{10}$$

Training several models with the GSG improved ensemble prediction of the models (see Section 4.4).

3.4 Optimization

The optimization algorithm is described in Algorithm 1. We take output probabilities $p_{\theta_1}^c$ and $p_{\theta_2}^c$ from the same input images. Then, we compute losses and update parameters of network θ_0 and θ_1 using SGD. Two networks are updated simultaneously by one forward calculation from the same batch samples. Therefore, the computational cost is low.

4 Experiment

We evaluated our method's performance on various network architectures and datasets.

Models	Method	CIFAR-10	CIFAR-100	SVHN	Tiny ImageNet	Param
ResNet32	Independent	93.03	70.10	97.87	52.59	0.5 M
	DML	93.00	72.00	97.94	52.86	
	GSG	**93.47**	**72.67**	**98.16**	**52.97**	
ResNet110	Independent	93.53	70.99	97.95	56.70	1.7 M
	DML	**94.20**	73.54	98.09	56.14	
	GSG	93.81	**74.30**	**98.12**	**56.85**	
DenseNet-40	Independent	93.25	72.15	97.99	54.83	1.0 M
	DML	93.77	**73.74**	97.99	55.82	
	GSG	**94.16**	73.62	**98.08**	**55.87**	
WideResNet 28-2	Independent	94.16	74.58	98.04	57.76	1.5 M
	DML	94.68	76.55	98.17	**60.36**	
	GSG	**94.94**	**76.65**	**98.18**	59.79	

Table 1: Classification performances on each dataset with different network architectures. All experiments are trained with the two same models. "Independent" is independently trained. "DML" is deep mutual learning. "GSG" is our method. "Param" is the number of model parameters.

4.1 Datasets and Models

CIFAR-10: The CIFAR-10 dataset [10] contained 50,000 training images with 5,000 images per class and 10,000 test images with 1,000 images per class. The CIFAR-10 dataset was comprised of 32×32 pixel RGB images with 10 classes. However, we padded 4 pixels on each side to make the image size 40×40 pixels. We used randomly cropped 32×32 pixel images for training, and we used the original 32×32 pixel images for testing.
CIFAR-100: The CIFAR-100 dataset [10] used 50,000 training images with 500 images per class and 10,000 test images with 100 images per class. The CIFAR-100 dataset contained 32×32 pixel RGB images with 100 classes. We applied the same augmentations as CIFAR-10.
SVHN: The Street View House Numbers (SVHN) dataset [12] consisted of 73,257 standard training images, 26,032 test images, and 531,131 extra training images. We used all the training data without any data augmentation.
Tiny ImageNet: Tiny ImageNet [1] was a subset of the ImageNet dataset [15] with 64×64 resolution. It contained 100,000 training images and 10,000 test images in 200 classes. We applied the same augmentations as CIFAR-10.
Models: We conducted experiments using small networks (ResNet32, ResNet110 [7], Wide ResNet28-2 [16], and DenseNet40 [9]). In the experiments using Tiny ImageNet, we changed the stride of the first conv layer to 2 for each model.

4.2 Implementation Details

We used SGD with Nesterov momentum as the optimization algorithm for all the experiments, and we set the initial learning rate to 0.1, momentum to 0.9, and minibatch size to 64. On CIFAR-10 and CIFAR-100, the learning rate dropped by 0.1 every 60 epochs, and we trained for 200 epochs according to [19]. On SVHN, the learning rate dropped by 0.1 at half of the maximum epoch and we trained for 40 epochs. On Tiny ImageNet, the learning rate dropped by 0.1 at half of the maximum epoch, and we trained for 80 epochs.

4.3 Results for Various Datasets and Models

Table 1 shows the classification performance of each dataset with different network architectures. All the experiments were trained with the same two models. In the results of CIFAR-10 with ResNet32, DenseNet-40, and Wide ResNet28-2, our method outperformed DML. The proposed method of ResNet32 significantly improved by 0.47% compared with DML. From the results of CIFAR-100 with ResNet32, ResNet110, and Wide ResNet28-2, the proposed method outperformed DML. Notably, the proposed method of ResNet110 improved by 0.76% compared with DML. From the results of SVHN with all models, our method outperformed DML. Particularly, the proposed method of ResNet32 improved by 0.22% compared with DML. From the results of Tiny ImageNet with ResNet32, ResNet110, and DenseNet-40, our method outperformed DML. The proposed method of ResNet110 particularly improved by 0.71% compared with DML.

4.4 GSG Ensemble vs DML Ensemble

Zhang et al. reported that an ensemble of networks trained with DML outperforms an independent model ensemble. We verified the effect of an ensemble on the GSG by increasing the number of models to be collaboratively learned. Figure 3 shows the ensemble model accuracies with ResNet32 on CIFAR-100. The GSG ensemble prediction is higher than DML. The GSG can

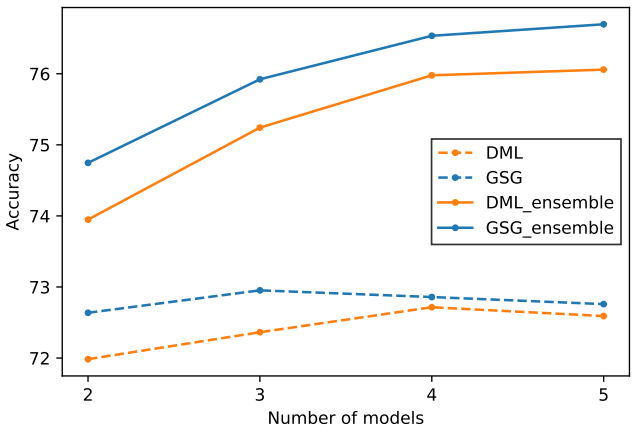

Figure 3: Performance of ensemble models with ResNet32 on CIFAR-100. The dash line is a mean accuracy of all model predictions. The solid line is a mean accuracy of ensemble predictions. All the experiments were performed 5 runs using the same seed for the random number generator.

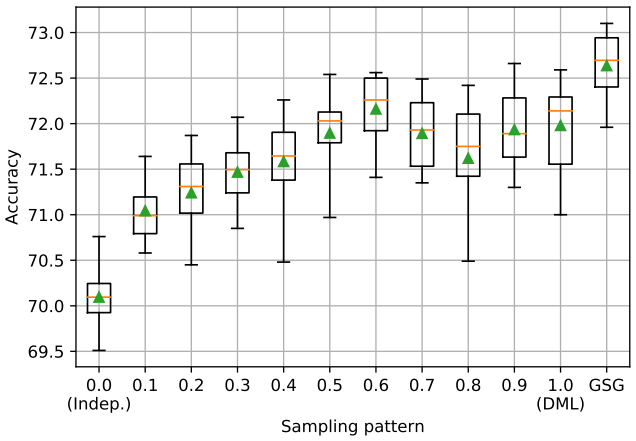

Figure 4: Accuracies over different sampling patterns. A horizontal line inside the box denotes a median and a triangle marker denotes the mean. "0.0" $\cdots$ "1.0" indicates a constant parameter C. "0.0" is the same as independent learning. "1.0" is the same as deep mutual learning.

increase the diversity of internal representations in the ensemble group because it cuts the soft targets off at an early stage in training.

4.5 Comparison with Constant Sampling

The GSG uses the value of training accuracy as the occurrence probability of r, as in Eq. (3). This makes it possible to suppress the transfer of unnecessary soft targets. To verify this effect, we evaluated the performance when we set r as the constant as follows:

$$r_i \sim Bernoulli\,(C)\,, \tag{11}$$

where C is a constant parameter $C = \{0, 0.1, \cdots, 1\}$.

Figure 4 shows the performance by using constant sampling or GSG. This experiment was performed 5 runs with two ResNet32 on CIFAR-100. We used the same seed of a random number generator used when initializing model parameters and data augmentation on each trial. GSG achieved higher performance than any constant sampling. This result shows that transferring soft targets from the beginning of training does not contribute improving performance.

4.6 Stochastic Gate vs Deterministic Gate

Our method probabilistically selects soft targets by using a Bernoulli distribution shown in Eq. (3). We select only correct samples deterministically, which is defined by

$$r_i = \delta_{\hat{y}_i, y_{(\boldsymbol{x}_i)}}. \tag{12}$$

To verify the effect that probabilistically selects samples, we evaluated the performance by using different soft target losses: Eqs. (3) and (12).

Method	Accuracy [%]	Improvement
Deterministic Gate	72.27	-
Stochastic Gate	72.64	+0.37

Table 2: Stochastic Gate vs Deterministic Gate

Table 2 shows the classification accuracies when using different soft target losses. This experimental setup is the same as Section 4.4 and 4.5. The stochastic approach achieved a higher performance than a deterministic approach.

5 Conclusion

In this paper, we proposed a gradual sampling gate (GSG) that probabilistically samples soft target loss in bidirectional knowledge distillation. The GSG can suppress unnecessary soft targets at an early stage of training by controlling the amount of soft target loss based on accuracy. Experimental results show that our method outperformed conventional DML.

References

[1] Tiny ImageNet Visual Recognition Challenge. `https://tiny-imagenet.herokuapp.com/`.

[2] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *In Proceedings of the 26th annual international conference on machine learning (ICML)*, pages 41–48. ACM, 2009.

[4] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[5] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pages 742–751, 2017.

[6] Yuhua Chen, Wen Li, and Luc Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7892–7901, 2018.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *In IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *In NIPS Deep Learning and Representation Learning Workshop*, 2015.

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[10] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[11] D. Lopez-Paz, B. Schölkopf, L. Bottou, and V. Vapnik. Unifying distillation and privileged information. In *In International Conference on Learning Representations (ICLR)*, November 2016.

[12] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[13] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

[14] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *In International Conference on Learning Representations (ICLR)*, 2015.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *In British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[17] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *In International Conference on Learning Representations (ICLR)*, 2017.

[18] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184*, 2017.

[19] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.