INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Energy-friendly keyword spotting system using add-based convolution

Hang Zhou, Wenchao Hu, Yu Ting Yeung, Xiao Chen

Huawei Noah's Ark Lab, China
{zhouhang25, huwenchao, yeung.yu.ting, chen.xiao2}@huawei.com

Abstract

Wake-up keyword of a keyword spotting (KWS) system represents brand name of a smart device. Performance of KWS is also crucial for modern speech based human-device interaction. An on-device KWS with both high accuracy and low power consumption is desired. We propose a KWS with add-based convolution layers, namely Add_TC-ResNet. Add-based convolution paves a new way to reduce power consumption of KWS system, as addition is more energy efficient than multiplication at hardware level. On Google Speech Commands dataset V2, Add_TC-ResNet achieves an accuracy of 97.1%, with 99% of multiplication operations are replaced by addition operations. The result is competitive to a state-of-the-art fully multiplication-based TC-ResNet KWS. We also investigate knowledge distillation and a mixed addition-multiplication design for the proposed KWS, which leads to further performance improvement.

Index Terms: keyword spotting, energy-friendly, human-computer interaction

1. Introduction

Nowadays, smartphones have become an important part of daily life. Speech interaction, as a natural way of operating a smartphone, is increasingly popular. A Keyword Spotting (KWS) system, with wake-up keyword representing the brand name of the device, must be always-on. Therefore, reducing power consumption of a KWS system becomes extraordinarily valuable for KWS system design.

In recent years, various neural network based end-to-end keyword spotting (E2E-KWS) systems have been investigated. These systems have achieved remarkable performance. E2E-KWS was firstly proposed by Chen et al. [1]. The authors trained a deep neural network to predict keywords or sub-word units of keywords directly. They applied a post-processing method to obtain corresponding confidence scores. Their work demonstrated superior performance of E2E-KWS over traditional hidden Markov model based methods. Since then, convolutional neural networks (CNN) [2], recurrent neural networks (RNN) [3], convolutional recurrent neural networks (CRNN) [4] and attention based models [5] are adopted to improve accuracy of KWS system. Meanwhile, there are a lot of researches on reducing number of parameters and flops of KWS systems, but maintaining high accuracy. In particular, Zhang et al. [6] compared performance of seven different models with limited parameters and computation. They proposed a depth-wise separable convolutional neural network (DS-CNN) and achieved high accuracy on Google Speech Commands V1 dataset [7]. Later, the performance of DS-CNN was surpassed by temporal convolution ResNet (TC-ResNet) [8]. Rybakov et al. [9] re-implemented previously published KWS models and compared their latency and accuracy on real mobile devices. Their results showed that multi-head attention RNN (MHAtt-RNN) reached the highest accuracy (98.0%), while gated recurrent units (GRU) and CRNN achieved the lowest latency. Singular value decomposition filter (SVDF) [10] and TC-ResNet achieved a balance of low latency and high accuracy.

Recently, more research efforts have been focusing on reducing computation and latency, without sacrificing high accuracy of a KWS system. Smaller models such as SVDF, which achieve balance of size, accuracy and latency, attract more interests than more complex models [10, 11]. Various techniques have been applied to reduce the amount of computation, for example, model structure optimization, model compression, knowledge distillation, and quantization [12, 13, 14]. All these methods achieve remarkable progress. However, conventional methods become more difficult to further reduce the amount of computation. Replacing energy-intensive operations with energy-friendly operations becomes a reasonable option [15]. In this paper, we adopt a method called AdderNet [16], which replaces multiplication operations with addition operations in CNN layers. An adder circuit requires fewer logic gates than a multiplier circuit, which implies fewer transistor counts and lower power consumption. AdderNet leads to significant power consumption reduction, especially when the design is implemented at hardware level such as field-programmable gate array (FPGA) [17]. Fundamental idea of AdderNet is to use ℓ_1-norm as distance measure when calculating similarity between CNN filters and features. In [16], Chen et al. have evaluated AdderNet in image classification. AdderNet achieves comparable accuracy with conventional CNN of similar architecture.

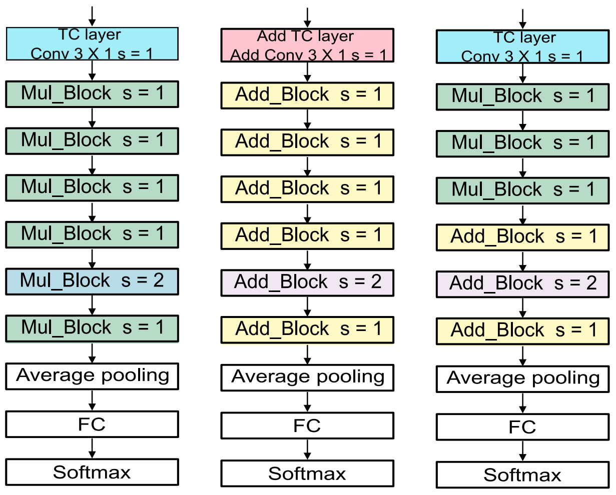
We propose an add-based TC-ResNet with AdderNet to reduce power consumption of KWS, namely Add_TC-ResNet. We train conventional TC-ResNet as baseline. We gradually substitute multiplication-based convolution layers with add-based convolution layers in the TC-ResNet, to evaluate the effectiveness of AdderNet in KWS. We adopt data augmentation to improve system performance. We further evaluate knowledge distillation methods [18], to transfer useful information from TC-ResNet to Add_TC-Resnet. To the best of our knowledge, this is the first add-based KWS system in the literature.

This paper is organized as follows. We describe our add-based TC-ResNet in Section 2. The experimental setup and results are presented in Section 3. In Section 4, we present and discuss our ablation studies, including our experiments on knowledge distillation. Finally, we conclude our work in Section 5.

2. System architecture

2.1. Baseline TC-ResNet

TC-ResNet has advantages of high accuracy and low latency. The architecture of TC-ResNet is shown in Figure 1(a), which consists of three parts. The first part is a temporal convolution layer (TC layer), in which input features are considered as time series instead of feature images. This is the key



(a) *TC-ResNet* (b) *Add_TC-ResNet* (c) *Mix_TC-ResNet*

Figure 1: *The architectures of different TC-ResNet based KWS models. Their architectures are similar, except that conventional multiplication based convolution layers are replaced by add-based convolution layers.*

to small footprint and low computational complexity. Readers may refer to [8] for further details. The second part consists of six ResNet blocks with multiplication based convolution (Mul_Block), which is the core module of the model. There are two types of Mul_Block, with different sizes of strides and convolution filter kernels. The first type is shown in Figure 2(a), in which the size of convolution filter kernel is set to (9 $\times$ 1) and the stride size is 1, followed by batch-norm and activation. The second type is shown in Figure 2(b), in which the stride size is 2, and an extra convolution layer with (1 $\times$ 1) kernel is applied to residual connection for dimension matching. Note that all the convolution layers do not have biases in our implementation, which is the same setting as in [8], but differs from the setting in [9]. The third part is a classification module, which consists of average pooling , a fully-connected (FC) layer, and a softmax layer.

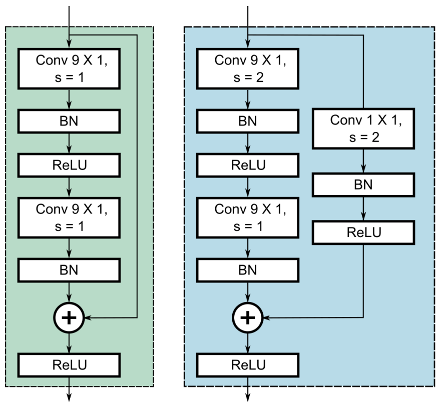
2.2. Add-based convolution layer

We denote a filter as $F \in \mathbb{R}^{d\times g\times c_{in}\times c_{out}}$ with kernel size $d\times g$, number of input channels c_{in} and number of output channels c_{out}. We also denote input feature as $X \in \mathbb{R}^{H\times W\times c_{in}}$ with H and W as height and width of the feature respectively. The key idea is that computation of convolution can be regarded as measuring similarity between input feature and filter, with Y is the output feature [16],

$$Y(m,n,p) = \sum_{i=0}^{d}\sum_{j=0}^{g}\sum_{k=0}^{c_{in}} S\big(X(m{+}i,n{+}j,k),F(i,j,k,p)\big), \tag{1}$$

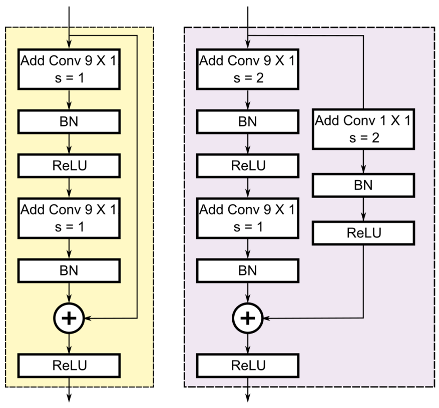
where $S(\cdot,\cdot)$ is a pre-defined distance measure. When $S(\cdot,\cdot)$ is cross-correlation, Equation 1 becomes conventional convolution in deep learning. We may define $S(\cdot,\cdot)$ with ℓ_1-norm as the distance measure. The forward pass thus becomes,

$$Y(m,n,p) = -\sum_{i=0}^{d}\sum_{j=0}^{g}\sum_{k=0}^{c_{in}} \big|X(m{+}i,n{+}j,k){-}F(i,j,k,p)\big|. \tag{2}$$



(a) *Mul_Block (s = 1)* (b) *Mul_Block (s = 2)*

Figure 2: *The architecture of ResNet blocks with conventional multiplication based convolution layers (Mul_Blocks).*



(a) *Add_Block, (s = 1)* (b) *Add_Block, (s = 2)*

Figure 3: *The architecture of ResNet blocks with add-based convolution layers (Add_Blocks).*

Multiplication is avoided as computation of ℓ_1-norm only requires addition and absolute operations. We refer this type of filter operation with ℓ_1-norm instead of cross-correlation to as add-based convolution. For back-propagation and gradient computation of add-based convolution layer, we keep consistent with the procedures described in [16].

2.3. Add-based ResNet Block (Add_Block)

As shown in Figure 2(a) and (b), conventional convolution layer in a standard ResNet block (Mul_Block) involves multiplication between input feature and convolution kernels. By replacing conventional convolution layer by add-based convolution layer, we build an Add_Block as shown in Figure 3(a) and (b). Except the convolution layer, other parts of an Add_Block are the same as those of an Mul_Block.

2.4. Proposed add-based TC-ResNet

The model architecture of Add_TC-ResNet in Figure 1(b) is similar to that of TC-ResNet in Figure 1(a). The main difference is that we apply Add_Blocks in Figure 3 to Add_TC-

Table 1: *Overall performance of multiplication based TC-ResNet and the proposed add-based Add_TC-ResNet*

System	# Parameters	Accuracy (%)
TC-ResNet [9]	365K	97.4
TC-ResNet (Our)	364K	97.8
Add_TC-ResNet (Our)	364K	97.1

ResNet instead of Mul_Blocks in Figure 2. To balance system performance and computation, we may vary the amount of Add_Blocks by replacing only some Mul_Blocks. We name the mixed-block design as Mix_TC-ResNet as shown in Figure 1(c).

3. Model training and system performance

3.1. Dataset

We perform our experiments with Google Speech Commands dataset V2 [7]. Ten words including “Yes”, “No”, “Up”, “Down”, “Left”, “Right”, “On”, “Off”, “Stop”, “Go” are selected as target keywords, and “silence” and “unknown” are selected as negative classes. The dataset is split into train set with 36923 samples, validation set with 4445 samples and test set with 4890 samples. The setting is the same as [9]. In order to build a more robust system, we apply following data augmentation methods:

- Mixing background noise [6];
- Random time shifting [6];
- SpecAugment without time warping [9, 19].

3.2. Model training

We apply 40-dimension Mel-frequency cepstral coefficients (MFCC) as audio feature. A MFCC feature vector is computed every 20 ms over a window of 40 ms. The shape of the input feature is thus 49 $\times$ 40 for one second of audio. The upper cut-off frequency of MFCC is set to 7800 Hz. We train all models in PyTorch framework with cross-entropy objective and Stochastic Gradient Descent optimizer. We generate 10 million training samples, and train the networks for 10 epochs using a batch size of 100. The initial learning rate is set to 0.1 for conventional multiplication based TC-ResNet and 0.01 for all add-based settings. We deploy learning rate decay during training.

3.3. System performance

We evaluate the models every 100 steps using the validation set. The checkpoint with the highest accuracy is evaluated on the test set. We carry out experiments three times for each setting with different random seeds, and report averaged results as in Table 1. Our TC-ResNet reaches an accuracy of 97.8% which is slightly better than the reported accuracy of 97.4% in [9]. The performance gain should be attributed to better data augmentation. For the proposed Add_TC-ResNet, when all the multiplication operations in TC layers and ResNet blocks are replaced by addition operations, the system achieves an accuracy of 97.1%. With 0.7 percent accuracy lower than our TC-ResNet implementation, Add_TC-ResNet is still one of the best systems reported in the literature.

4. Ablation study

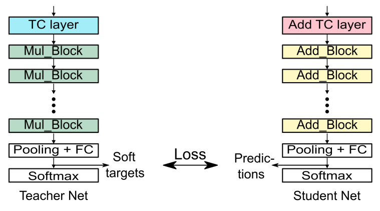
4.1. Performance with mixed-block design

There are 6 ResNet blocks in the original TC-ResNet KWS. In order to understand effects of add-based convolution layers, we gradually replace Mul_Blocks in TC-ResNet with Add_Blocks. As a result, we train different Mix_TC-ResNet models with variable amounts of multiplication and addition operations. To maintain model stability, we replace the Mul_Blocks with Add_Blocks layer-by-layer starting from the one closest to softmax output, finally replacing the TC layer by add-based convolution to become Add_TC-ResNet. We name the setting according to ResNet block combinations. For example, the Mix_TC-ResNet as shown in Figure 1(c) with 3 Mul_Blocks and 3 Add_Blocks is labelled as mul3_add3_TC-ResNet. The performance of different Mix_TC-ResNet models are shown in Table 2. With an increased number of Mul_Blocks being replaced by Add_Blocks, accuracy decreases from 97.8% in TC-ResNet to 97.1% in Add_TC-ResNet. This result provides us guidance for balancing power consumption and accuracy by choosing suitable proportion of addition and multiplication operations. For example, we can apply mul1_add5_TC-ResNet model in which 80% of multiplication operations (13.6 M $\rightarrow$ 2.6 M) are replaced by addition operations at an expense of 0.5% accuracy reduction. We do not replace multiplication operations in the final FC layer due to insignificant amount of computation.

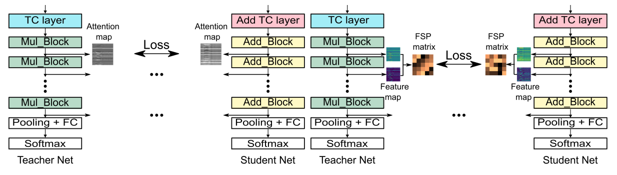
4.2. Performance with knowledge distillation

We also evaluate with different knowledge distillation methods. We consider TC-ResNet as teacher model. Then we apply knowledge distillation to Add_TC-ResNet which is the student model. The following 3 knowledge distillation methods are investigated as shown in Figure 4:

- Soft target (ST): To mimic output of final layer [18] as shown in Figure 4(a);
- Attention transfer (AT): To imitate from feature map of every layer [20] as shown in Figure 4(b);
- Flow of solution procedure (FSP): To gain knowledge from relationship between adjoining layers [21] as shown in Figure 4(c);



(a) *ST*



(b) *AT* (c) *FSP*

Figure 4: *The 3 different knowledge distillation methods evaluated in this work: (a) Soft target (ST), (b) Attention transfer (AT), and (c) Flow of solution procedure (FSP).*

Table 2: *System performance of TC-ResNet, Mix_TC-ResNet and Add_TC-ResNet models with different number of Add_Blocks and Mul_Blocks. TC layers with multiplication based and add-based convolution are represented by "mul" and "add" respectively.*

Model	TC layer	# Mul_Block	# Add_Block	# Multiplication	# Addition	Accuracy (%)
TC-ResNet	mul	6	0	13.6M	13.6M	97.8
mul5_add1_TC-ResNet	mul	5	1	11.3M	15.9M	97.6
mul4_add2_TC-ResNet	mul	4	2	9.2M	18.0M	97.5
mul3_add3_TC-ResNet	mul	3	3	7.0M	20.2M	97.6
mul2_add4_TC-ResNet	mul	2	4	4.8M	22.4M	97.5
mul1_add5_TC-ResNet	mul	1	5	2.6M	24.6M	97.3
mul0_add6_TC-ResNet	mul	0	6	0.4M	26.8M	97.2
Add_TC-ResNet	add	0	6	0.1M	27.1M	97.1

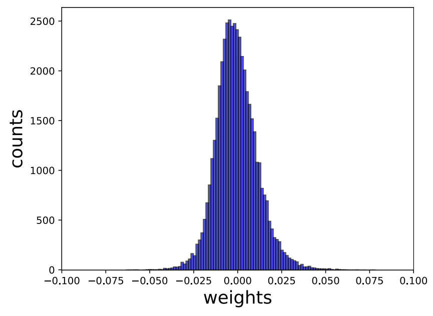
Table 3: *System performance with different knowledge distillation methods*

System	KD method	Accuracy (%)
TC-ResNet	-	97.8
Add_TC-ResNet_ST	ST	97.2
Add_TC-ResNet_AT	AT	97.1
Add_TC-ResNet_FSP	FSP	97.1
Add_TC-ResNet	-	97.1

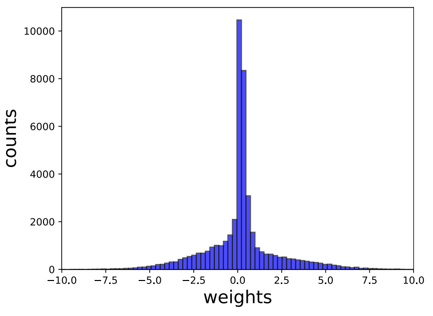
The results of different knowledge distillation methods are shown in Table 3. The ST method achieves a small improvement to an accuracy of 97.2%. There is still a performance gap between the teacher model TC-ResNet and the student model Add_TC-ResNet. Other knowledge distillation methods, AT and FSP, do not bring further improvement to Add_TC-ResNet, although they do not harm the accuracy.

4.3. Weight distribution of convolution filters

We further compare weight distributions of convolution filters in Mul_Blocks and Add_Blocks. The weight distributions of the second convolution filter of the sixth Mul_Blocks and Add_Blocks of TC-ResNet and Add_TC-ResNet respectively, i.e. the blocks closest to softmax outputs, are shown in Figure 5. The weights of the Mul_Block filter in TC-ResNet tend to follow Gaussian distribution, while the weights of the Add_Block filter in Add_TC-ResNet tend to follow Laplace distribution. We observe that other filters follow the same distribution trends, according to the corresponding ResNet block types. The same observation is also reported in [16]. This observation agrees to our expectation, as objective function of an Add_Block filter involves ℓ_1-norm, which tends to obtain sparse parameter estimation. The sparsity of filter parameters facilitates more compact storage for Add_TC-ResNet models, especially after model quantization.



(a) *Mul_Block*



(b) *Add_Block*

Figure 5: *Weight distributions of the second convolution filter in the sixth ResNet blocks, the ones closest to softmax outputs. (a) Mul_Block of TC-ResNet, (b) Add_Block of Add_TC-ResNet.*

5. Conclusion

We demonstrate the possibility of building a KWS system using add-based convolution layer. The add-based KWS achieves 97.1% accuracy on Google Speech Commands dataset V2 when all the multiplication based convolution layers are replaced by add-based convolution layers. By varying the composition of Add_Blocks and Mul_Blocks, we can trade-off performance and power consumption. When 80% of multiplication operations are replaced by addition operations, the model can achieve an accuracy of 97.3%. To improve the performance of add-based KWS, we investigate 3 knowledge distillation methods. We find that soft target method tends to slightly improve the accuracy to 97.2%. In the future, we are going to investigate low-bit rate quantization on add-based KWS system to further reduce power consumption.

6. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4087–4091.

[2] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. Interspeech 2015*, 2015, pp. 1478–1482.

[3] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, 2016, pp. 474–480.

[4] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, "Convolutional recurrent neural networks for small-footprint keyword spotting," in *Proc. Interspeech 2017*, 2017, pp. 1606–1610.

[5] C. Shan, J. Zhang, Y. Wang, and L. Xie, "Attention-based end-to-end models for small-footprint keyword spotting," in *Proc. Interspeech 2018*, 2018, pp. 2037–2041.

[6] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[7] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[8] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," in *Proc. Interspeech 2019*, 2019, pp. 3372–3376.

[9] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," in *Proc. Interspeech 2020*, 2020, pp. 2277–2281.

[10] R. Alvarez and H.-J. Park, "End-to-end streaming keyword spotting," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6336–6340.

[11] H.-J. Park, P. Violette, and N. Subrahmanya, "Learning to detect keyword parts and whole by smoothed max pooling," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7899–7903.

[12] T. Higuchi, M. Ghasemzadeh, K. You, and C. Dhir, "Stacked 1D convolutional networks for end-to-end small footprint voice trigger detection," in *Proc. Interspeech 2020*, 2020, pp. 2592–2596.

[13] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting." in *Proc. Interspeech 2016*, 2016, pp. 1878–1882.

[14] T. Bluche, M. Primet, and T. Gisselbrecht, "Small-footprint open-vocabulary keyword spotting with quantized LSTM networks," *arXiv preprint arXiv:2002.10851*, 2020.

[15] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.

[16] H. Chen, Y. Wang, C. Xu, B. Shi, C. Xu, Q. Tian, and C. Xu, "Addernet: Do we really need multiplications in deep learning?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1468–1477.

[17] Y. Wang, M. Huang, K. Han, H. Chen, W. Zhang, C. Xu, and D. Tao, "Addernet and its minimalist hardware design for energy-efficient artificial intelligence," *arXiv preprint arXiv:2101.10015*, 2021.

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[19] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.

[20] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[21] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.