Performance Analysis of Deep Neural Network based on Transfer Learning for Pet Classification

Bhavesh Jaiswal[1], Nagendra Gajjar[2]
Department of Electronics and Communication Engineering
Institute of Technology, Nirma University, SG Highway
Ahmedabad, Gujarat, India

***Abstract*—Deep learning frameworks have progressed beyond human recognition capabilities and, now it's the perfect opportunity to optimize them for implementation on the embedded platforms. The present deep learning architectures support learning capabilities, but they lack flexibility for applying learned knowledge on the tasks in other unfamiliar domains. This work tries to fill this gap with the deep neural network-based solution for object detection in unrelated domains with a focus on the reduced footprint of the developed model. Knowledge distillation provides efficient and effective teacher-student learning for a variety of different visual recognition tasks. A lightweight student network can be easily trained under the guidance of the high-capacity teacher networks. The teacher-student architecture implementation on binary classes shows a 20% improvement in accuracy within the same training iterations using the transfer learning approach. The scalability of the student model is tested with binary, ternary and multiclass and their performance is compared on basis of inference speed. The results show that the inference speed does not depend on the number of classes. For similar recognition accuracy, the inference speed of about 50 frames per second or 20ms per image. Thus, this approach can be generalized as per the application requirement with minimal changes, provided the dataset format compatibility.**

Keywords*—Machine learning; knowledge distillation; transfer learning; domain adaptation

I. Introduction

Deep neural networks are thriving, due to vast data availability, newer complex models, and heterogeneous compute capacity. The data accumulation ease and its open-source availability are opening new doors for the research community. So new models are popping up almost every day on how to solve real-world problems using that data. Now crunching the data is also getting cheaper day by day, and one does not require a personal high-end custom configured system for this job. It is offloaded to cloud-based solutions provided by Amazon, Google, and Microsoft. Traditional machine learning & data mining algorithms make predictions using statistical models and trained on labelled or unlabelled training datasets. As the labelled data may be too few in practical applications; so to build a good classifier; semi-supervised classification done by using a large amount of unlabelled data and a small amount of labelled data [1], [2], [3]. In [4], the problem of how to deal with the noisy-class label is explored. Similarly, in [5], cost-sensitive learning is considered. In [6], it is shown that having a minimum depth to the network is vital for the model performance. All these approaches assumed that the distributions of the labelled and unlabelled data were the same.

For implementation on edge-based devices, the model size could be cut down by the compression techniques at various levels in the model, data, and computation. The classic Alexnet [7] was trained on the Imagenet dataset and performed 2.27 billion operations with 238MB of memory usage for storing the model data itself. In the compressed model, Squeezenet [8] performs 2.17 billion of operation but with a smaller footprint of 4.8MB, while Darknet [9], an open source for the Yolo [10], does less than 1 billion operations with 28MB of the footprint. Note that this comparison is assuming a baseline accuracy of 80 per cent in recognizing the labelled visuals. It does not include the run time memory requirements while performing the computation, which is not directly proportional to the number of operations performed as neural networks are non-linear models. Also, compression-decompression takes more computation power. Mobilenet [11] tries to address this problem to an extent. Though the model size is reduced from the storage perspective but while performing the inference on a lightweight platform, they may fail to give the real-time response due to resource constraints. To make them predict with a high confidence value [12] the sematic segmentation approach from [13], [14] is used by [15], [16], [17].

The practical implementation of the deep neural network in real-life scenarios is quite the opposite of the earlier description. IoT based heterogeneous devices have resource constraints that limit their use on them. They mostly offload this to the cloud, but that solution is not always feasible due to the latency involved. This work explores the knowledge distillation approach in deep neural networks for IoT edge devices for real-time applications. The contribution of this work is to train a smaller model for a lightweight target platform with a negligible loss of accuracy. The proposed lightweight model can be easily customized to the different domains and can be easily ported to IoT based edge devices. Section 2 details why deep learning on heterogeneous edge devices is difficult to implement. In Section 3, the state of the art approaches for model reduction is presented. Section 4 delves into the knowledge distillation approach and how it can be used on the edge-based device followed by experimental setup and performance analysis of the implementation.

II. Deep Learning on Edge Devices

Convolution Neural Networks models have evolved to surpass the human capabilities in image classification task but when it comes to their deployment on the edge devices, there use is limited due to various resource constraints as described below:

A. Limited Data

The large dataset is not available in all circumstances for the training of the network and even it is available it may be quite expensive in terms of time and feasibility. Data privacy is another factor which forces to work with lesser data locally on the device itself.

B. Limited Model Footprint

Models which thrusted the growth of DNN with their accuracy limits surpassing humans are quite bulky in terms of memory requirements for storage. This memory could be either for storage of the model or for the storage of the millions of the weights calculated during the runtime. For practical implementation, the memory footprint of the application should be small enough to fit into any embedded device.

C. Limited Computation

Even with lesser data and smaller models the solution does not work out. Because of the millions of intermediate weights computation, it involves during the model run, it may require a desktop/server capability to finish the task in real time. The computational latency is not tolerable in the practical application involving the heterogeneous edge device. Now as the heterogeneous edge devices has limited capabilities, one need to devise the ways to eliminate or reduce these limitation causes. The next section describes this in detail.

III. Deep Neural Network Reduction

Though DNNs have the tremendous diversity of structures, still the core computation of a network is the variations of matrix-multiplications or more precisely multiply-and-accumulate (MAC) operations. The factors which effect the MAC operations are batch size, image dimensions, filter type, no. of channels, kernel size and activation size. These combined for every neuron to neuron connections make the millions of hyper-parameters of the DNN.

To reduce these transformation functions parameterized by learnable weights, researchers worldwide have developed their own various model compression techniques, but only some of the well-researched approaches are covered here for brief overview.

A. Pruning

The hyper parameter space of the DNN is reduced by trimming the network physically or pruning the network itself in various ways.

The unimportant weight connections can be pruned if they are below a predefined threshold or if they are redundant. About 50% of the weights can be pruned without fine-tuning and with fine-tuning, more than 80% of the weights can be pruned [18]. The pruning of the weights can be driven by energy distribution for the network [19]. In Energy Inference Engine [20], the sparse weights after pruning can also be compressed to reduce memory access bandwidth. Huffman coding is used to reduce storage and bandwidth requirements for weights by 20-30% [21].

Another approach to trim the individual neurons is that if they are redundant [22]. As these are basic element of the network so the associated connections of the neuron will also be obliterated. In the literature many ways are researched to do this type of pruning, even some of the neuron layers which do not contribute much in the network updation can also be removed [23].

Convolutional filters are applied to the data and according to their importance, they can be eliminated from the network. The filter's importance can be known by their influence on the weight calculation or L1/L2 norm [24]. Other methods are also researched in the literature which is not the scope of this work.

B. Quantization

The network architecture can be improved in many ways e.g. by reducing the quantity of weights and number of operations. The large convolution operation can be replaced with a number of smaller convolution operators having fewer weights in total, keeping the effective receptive field same i.e. large filters can be emulated with several of the smaller size filters in cascade e.g. convolution of size n by n can be made by combining 1 by n convolution with N by 1 convolution [21]. SqueezeNet [25] uses this approach to achieve an overall reduction in number of weights up to 50x compared to AlexNet, while keeping the accuracy in similar range.

Weights of fully connected layers can be quantized using Regularization technique [26], [27]. Clustering by 'k-means' [28] achieved more than 20x compression with negligible loss of accuracy. Hashed Nets [29] use a low-cost hash function to group weights into hash buckets to share parameters.

C. Knowledge Distillation

Knowledge distillation (KD) was introduced by [30] as:

- Train a large model that performs and generalizes very well. This is called the teacher model.
- Take all the data you have and compute the predictions of the teacher model. The total dataset with these predictions is called the knowledge, and the predictions themselves are often referred to as soft targets. This is the knowledge distillation step.
- Use the previously obtained knowledge to train the smaller network, called the student model.

Fig. 1 summarizes this pictorially.

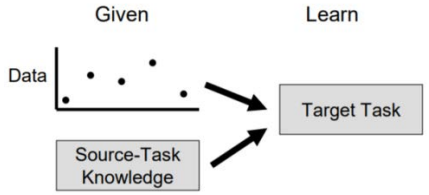

Fig. 1. Example of a Transfer Learning Model.

IV. TEACHER-STUDENT LEARNING

Knowledge distillation starts with training a larger model, the teacher 'T'. As it is trained on a heavier platform (GPU), it achieves high performance. Then a lightweight model known as student 'S' is deployed to learn from 'T'. Now, 'S' is supposed to give comparable performance as 'T' but with less memory and more speed.

To improve knowledge transfer from teacher to student various types of methods are researched. Assuming a trained 'T' has already eliminated some label errors contained in the ground truth data, the authors in [29] treated the hard label predicted by 'T' as the underlying knowledge. While in [30], the soft label produced by 'T', i.e., the classification probabilities, are focused to provide more information to transfer. In general, knowledge is transferred from the 'T' to 'S' by minimizing a loss function in which the target is the distribution of class probabilities predicted by 'T'. This probability distribution has the correct class at a very high probability (close to '1') with all other class probabilities very close to '0'. As such, it does not provide much information beyond the ground truth labels already provided in the dataset. For this, Hinton [30], introduced the concept of "softmax temperature". As it grows, the probability distribution generated by the softmax function becomes softer, providing more information as to which classes 'T' found more like the predicted class. This is the "dark knowledge" embedded in the 'T' and transferred to 'S' in the distillation process. The distillation related work can be categorized as below:

- Feature Map: The feature map across channel dimension can be averaged to obtain spatial attention map [31]. The inner product of two feature maps can be used for the inter-layer flow [32]. The author in [33] improved this idea with singular value decomposition (SVD). A recent work [34] demonstrated the effectiveness of mimicking feature map directly in distillation.
- Transfer strategy: FitNets [35] selected a hidden layer from 'T' and 'S' to be hint layer and guided layer respectively. 'S' can get a better initialization through pre-training the guided layer with the hint layer as supervision. Net2net [36] proposed a function-preserving transformation, which makes it possible to directly reuse it from 'T' to initialize the hyperparameters of 'S'.
- Hybrid strategy: Adversarial learning is used with distillation by using a comparator to check the outputs of 'S' and 'T' are close enough or not [37]. The author in [38] exploited reinforcement learning to search the best network structure of 'S' under the influence of 'T'. In [39] and [40] progressive or lifelong learning is referred to make knowledge transfer step by step.

Looking to this a novel approach to developing deep learning models for various domains is proposed. As every student in a class distribution may not have a similar capability or generally it a Gaussian curve. To flatten the curve on the higher side of learning capability, the model tries to imbibe the relative knowledge which can be used on the lightweight students to perform the object detection task with comparable performance. The aim is, to provide a generic solution to the problem with the assumption that the model can only be transferred successfully using the smaller dataset avoiding the limitations of the domain transfer.

V. EXPERIMENTAL EVALUATION AND DISCUSSION

A popular framework Caffe [41] is chosen for the binary image classification task. The Redux Dogs vs. Cats competition dataset [42] is used for training and testing purpose on NVIDIA GTX 770 with 1536 GPU cores [43]. The training data consists of 12500 images of each for Cats and Dogs. For testing phase 12500 images random images from the dataset are chosen.

The model calculates the probability of a pet and assigns a numeric value between 0 and 1 for the predicted class. Currently the implementation involves binary classification; cat and dog, but it can be extended easily to include other types of pets. For that the model will give the probability values for each class and the highest value is the closest. The accuracy of the implemented training models depends on the model training parameters and varies with change in hyper parameters that itself is a separate research area. The log-loss formula can be used to represent the accuracy of any model:

$$Logloss = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right] \quad (1)$$

Where, n represents the number of images in the test set. y the prediction probability for the dog and y^ equals to '1' if the current image is identified as a dog or equals to '0' if a current image is predicted as cat. The log-loss probability is calculated for each run and note that a smaller value of log loss is desired.

First, the complex teacher model is trained using labeled data and then same is tested with the unlabeled data to classify the pet either cat or dog. Fig. 2(a) shows the learning curve of the teacher model achieving 75% validation accuracy in 1500 iterations which occurred in about 2 and ½ hours. Further, the weights from the teacher model are used to pass to the student model as per knowledge distillation criteria, i.e., the student model is initialized with the pre-trained data/weights from the teacher model. Fig. 2(b) shows the training/learning curve of the student model achieving a 95% validation accuracy in about 1000 iterations which occurred in less than 2 hours.

In Table I, while doing transfer learning, the accuracy has jumped from 75% to 95% that too in lesser run time. The log loss value also comes down close to unity, the ideal log loss value for this problem.

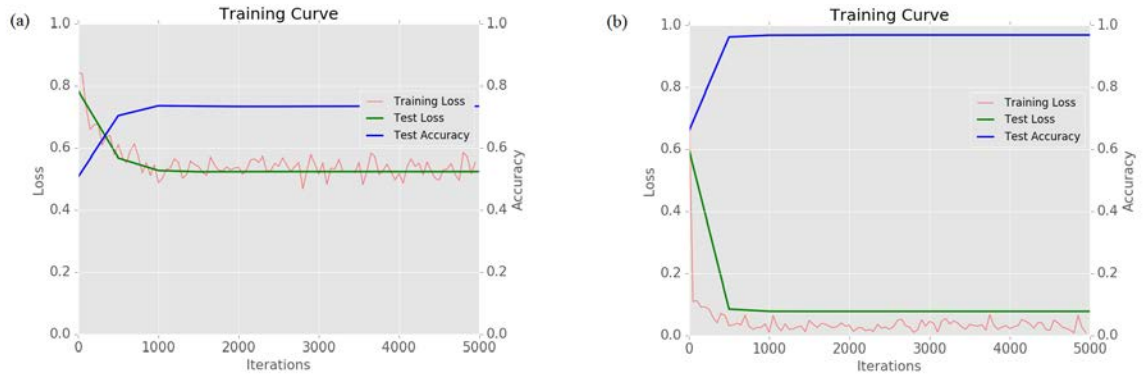

Fig. 2. Learning Curve for (a) Teacher Model; (b) Student Model.

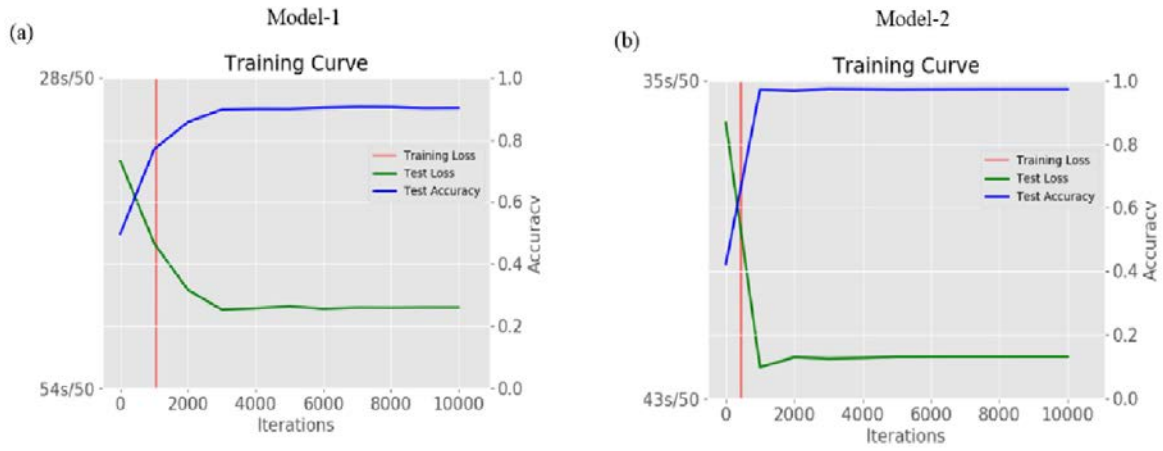

Fig. 3. Learning Curve for the GPU with Previous (a) Teacher Model; (b) Student Model.

TABLE I. PERFORMANCE COMPARISON FOR TWO MODELS

Parameters	**Teacher (Training run)**	**Student (Transfer learning)**
Accuracy (%)	75	95
Iteration	1500	1000
Time Taken (minutes)	155	110
Log Loss	8.9	1.1

To validate the transfer learning results further multiple runs are conducted on the higher capacity single GPU (NVIDIA GTX 1080Ti with 3584 cores) [44] configuration and with dual GPU configuration. The results are shown in Fig. 3 and summarized in Table II. It shows the comparison matrix from all the test runs on various GPU platforms. In model 1, using the GPU with significantly greater number of cores reduces the number of iterations to achieve the similar results and it converges faster. On the other hand, with model 2, the results are similar both in terms of accuracy and number of iterations irrespective of the platform availability. It indicates that transfer learning-based approach comes out as a clear winner for limited resource environment.

Next, the task is performed for binary, ternary, and multiclass identification and their performance is compared on basis of inference speed. The results show in Table III that the inference speed does not depend on the number of classes. The multiple domains are also considered to prove that for similar recognition accuracy the inference speed achieved is about 50 frames per second or 20ms per image.

TABLE II. RESULTS FROM VARIOUS GPU CORES

Platform NVIDIA GPU core count	**Model-1 (Training)**		**Model-2 (Transfer Learning)**	
	Accuracy (%)	***Number of Iterations***	***Accuracy (%)***	***Number of Iterations***
1536	75	1500	95	1000
3584	75/90	1000/4000	96	1000
3584 x 2	75/90	1000/4000	96	<1000

TABLE III. PERFORMANCE COMPARISON FOR MULTICLASS DETECTION

Classification	**Object Labelled**	**Inference Speed**
Binary	Cat and Dog	1.5fps 0.663s/image
Ternary1	Date, fig and hazelnut	2.95 fps 0.339s/image
Ternary2	Platelets, RBC, and WBC	142 fps 0.007s/image
Multi-class (5-classes)	Docks, Boats, Lifts, Jetskis, and Cars	48 fps 0.020s/image

VI. Conclusion

It can be safely concluded that using transfer learning approach a student model converges faster than the original complex teacher model. This directly translates to the saving in resource for each run of the learning which is exactly what is required for the implementation of CNN on heterogeneous embedded platform with lesser resources as now the lesser powerful embedded GPU (compared to discrete ones) can achieve similar accuracy. The results will make deployment and inferencing of DNN in heterogeneous devices easier and devices friendly. General-purpose experimentation platforms like raspberry-pi can be also used for the same. For the future work Nvidia latest platform like Jetson-TK [45] can be considered for real time implementation of this approach.

GPU acceleration and model compression are orthogonal to each other. How much a model can be compressed and accelerated subject to given resource constraints (storage, computational power, and energy) and user-specified performance goals (accuracy, latency) is open research question. The development of the generalized model compression and acceleration framework would add another value to it. More research in this area can lead to trade-off between model compression and acceleration dynamically.

Acknowledgments

The first author would like to thank the faculty/staff of Institute of Technology, Nirma University for the laboratory work support.

References

[1] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features off-the-shelf: An Astounding Baseline for Recognition," *arXiv.org*, 2014. https://arxiv.org/abs/1403.6382v3.

[2] J. Donahue, et.al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," Proceedings of the 31st International Conference on Machine Learning, vol. 32, pp. 647-655, 2014.

[3] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," Computer Vision, ECCV 2014, pp. 818-833, 2014.

[4] X. Zhu, X. Wu, "Class Noise vs. Attribute Noise: A Quantitative Study," Artificial Intelligence Review 22, pp. 177-210, 2004.

[5] Qiang Yang, C. Ling, X. Chai, and Rong Pan, "Test-cost sensitive classification on data with missing values," IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 5, pp. 626-638, 2006.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" Advances in Neural Information Processing Systems, vol. 27, 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, vol. 25, 2012.

[8] F. N. Iandola, et.al., "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," arXiv.org, 2016. https://arxiv.org/abs/1602.07360.

[9] J. Redmon, "Darknet: Open Source Neural Networks in C," [online] http://pjreddie.com/darknet/, 2013-2016.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, doi: 10.1109/cvpr.2016.91.

[11] A. G. Howard, et.al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv.org, 2017. https://arxiv.org/abs/1704.04861.

[12] J. Long, E. Shelhamer, and T. Darrell, "*Fully Convolutional Networks for Semantic Segmentation,*" CVPR 2015. arXiv.org https://arxiv.org/pdf/1411.4038v2.pdf.

[13] A. Nguyen, J. Yosinski, and J. Clune, "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images," arXiv.org, 2015. https://arxiv.org/abs/1412.1897v4.

[14] L. C. Chen, et.al., "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," arXiv.org, 2014. https://arxiv.org/abs/1412.7062v4. ICLR2015.

[15] R. Girshick, "*Fast R-CNN,*" IEEE International Conference on Computer Vision (ICCV), 2015.

[16] S. Ren, et.al., "*Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,*" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, Jun. 2017.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "*Mask R-CNN,*" arXiv.org, ICCV, 2017. https://arxiv.org/abs/1703.06870v3.

[18] Han, S., Pool, J., Tran, J., Dally, W. Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28, pp. 1135-1143, 2015.

[19] Yang, T.-J., Chen, Y.-H, Sze, V. "Designing energy-efficient convolutional neural networks using energy-aware pruning", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6071-6079, 2017.

[20] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M.A., Dally, W. "EIE: efficient inference engine on compressed deep neural network", ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 243-254, 2016.

[21] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. "Rethinking the inception architecture for computer vision", IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818-2826, 2016.

[22] Srinivas S., Babu R.V. "Data-free parameter pruning for deep neural networks", British Machine Vision Conference (BMVC), pp. 31.1-31.12, 2015.

[23] Chen S., Zhao, Q. "Shallowing deep networks: layer-wise pruning based on feature representations", IEEE Transactions on Pattern Analysis and Machine Intelligence, 41, 12, 3048-3056, 2018.

[24] Li H., Kadav A., Durdanovic I., Samet H., Graf H.P. "Pruning filters for efficient convnets", 5th International Conference on Learning Representations (ICLR), 2017.

[25] Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1 MB model size", 5th International Conference on Learning Representations (ICLR), 2017.

[26] Bucila, C., Caruana, R., Niculescu-Mizil, A. "Model compression", Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 535-541, 2006.

[27] Souli´e, G., Gripon, V., Robert, M. "Compression of deep neural networks on the fly", International Conference on Artificial Neural Networks, Lecture Notes in Computer Science, 9887, pp. 153-160, 2016.

[28] Gong, Y., Liu, L., Yang, M., Bourdev, L. "Compressing deep convolutional networks using vector quantization," 2014. https://arxiv.org/abs/1412.6115.

[29] Ba, J., Caruana, R. "Do deep nets really need to be deep?" Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'2014), 2, pp. 2654-2662, 2014.

[30] Hinton, G., Vinyals, O., Dean, J. "Distilling the knowledge in a neural network," 2015. https://arxiv.org/abs/1503.02531.

[31] Zagoruyko, S., Komodakis, N. "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer", 5th International Conference on Learning Representations (ICLR), 2017.

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

[33] Lee, S. H., Kim, D. H., Song, B. C. "Self-supervised knowledge distillation using singular value decomposition", European Conference on Computer Vision (ECCV), pp. 339-354, 2018.

[34] Gao, M., Shen, Y., Li, Q., Yan, J., Wan, L., Lin, D., Loy, C. C., Tang, X. "An embarrassingly simple approach for knowledge distillation," 2018, arXiv:1812.01819

[35] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., Bengio, Y. "Fitnets: Hints for thin deep nets", 3rd International Conference on Learning Representations (ICLR), 2015.

[36] Chen, T., Goodfellow, I., Shlens J. "Net2net: Accelerating learning via knowledge transfer", 4th International Conference on Learning Representations (ICLR) 2016.

[37] Belagiannis, V., Farshad, A., Galasso, F. "Adversarial network compression", European Conference on Computer Vision (ECCV), pp. 431-449, 2018.

[38] Ashok, A., Rhinehart, N., Beainy, F., Kitani, K. M. "N2n learning: Network to network compression via policy gradient reinforcement learning", 6th International Conference on Learning Representations (ICLR), 2018.

[39] Wang, H., Zhao, H., Li, X., Tan, X. "Progressive blockwise knowledge distillation for neural network acceleration", Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), pp.2769-2775, 2018.

[40] Gao, Mengya, et al. "Residual knowledge distillation", arXiv preprint arXiv:2002.09168. 2020.

[41] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. "Caffe: convolutional architecture for fast feature embedding", Proceedings of the 22nd ACM International Conference on Multimedia, pp. 675-678, 2014.

[42] Kaggle's Dogs Versus Cats Competition. or https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/.

[43] NVIDIA GTX 770, https://www.nvidia.com/en-us/geforce/graphics-cards/geforce-gtx-770/specifications/.

[44] NVIDIA GTX 1080Ti, https://www.nvidia.com/en-sg/geforce/products/10series/geforce-gtx-1080-ti/..

[45] NVIDIA Jetson-TK1 developer kit, https://developer.nvidia.com/embedded/jetson-tk1-developer-kit.