A Study of Distillation in Neural Networks

Darpan Tushar Sanghavi Sri Sudhamsu Krishna Manne
University of Massachusetts Amherst
{dsanghavi,smanne}@cs.umass.edu

Abstract

Distillation in neural networks is the process of transferring knowledge from one neural network to another - usually from a larger, cumbersome network to a smaller, simpler network. It is useful in compressing models to save computation and memory at test time. The core idea is that the second network is trained to match a softer, temperature annealed distribution of logits obtained from the first network instead of matching the provided hard labels. We study the effect of different parameters on this process of knowledge transfer and report our findings. Following that, we demonstrate the importance of relative class probabilities in the transferred knowledge.

1. Introduction and Problem Statement

State-of-the-art neural networks are doing incredibly well in a vast spectrum of problems now. Trends are going towards using deeper networks, which increases the number of parameters, hence increasing the memory requirement, and also the number of computations, increasing training and testing time. For large scale applications, current training times can reach upto 6 months. For practical applications, we need the least possible test time. Computationally and statistically efficient compression makes way for training and deployment of powerful neural networks in a small size, allowing more complex and intricate systems to develop.

Moreover, using an ensemble of networks almost always improves the results by a bit. This works because different networks (depending on their different initializations and training) will make different errors on the test set. Taking a combination of their predictions will reduce the errors overall as the errors made by one network would probably not be made by several other networks. But this also introduces huge redundancies. At test time, the test data is fed to all the networks in the ensemble, and several computations - many of them leading to the same information - need to be performed. Can we not represent this ensemble, or even a large complex network, with a single, simpler model? That is the problem we deal with, using the concept of distillation to accelerate the training of a simpler network.

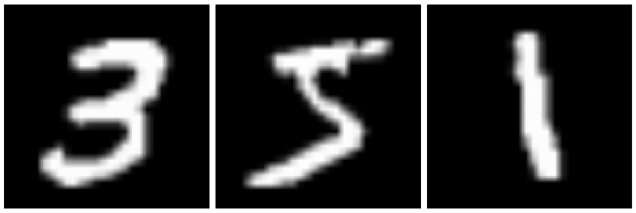


Figure 1: Image examples for 3, 5, and 1 from the MNIST dataset. It is clear that the 3 looks much more similar to 5 than to 1. This knowledge of similarity can be transferred through distillation.

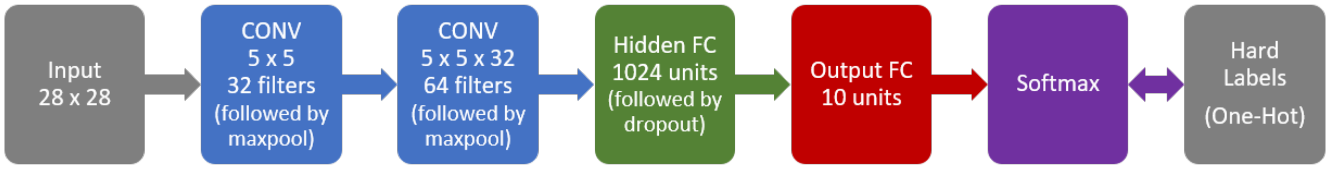
2. Technical Approach

Our investigation and analysis of distillation of knowledge in neural networks starts off by replicating and verifying the work of Hinton et al [3] on the MNIST dataset [5], and then experimenting along the same lines with the CIFAR-10 dataset [4]. We first train a deep high capacity neural network on the dataset. Once we have the trained model (or an ensemble of them), we generate a transfer set to train a simpler "distilled" network using the "soft" targets obtained from the cumbersome network. The soft targets are obtained using a temperature-defined softmax function; it is exactly the same as the usual softmax function but with the logits divided by a temperature value T. With logits z and a temperature T, we can calculate its probability q using the following equation:

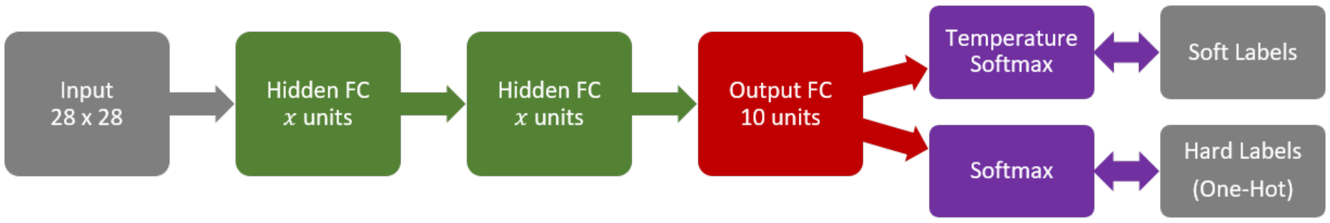
$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{1}$$

To see the effect of temperature softmax, consider $(1, 5, 4)$ as sample output logits for some network. A normal softmax would lead to $(0.01, 0.72, 0.27)$, whereas a temperature softmax with T=5 would lead to $(0.20, 0.44, 0.36)$. The normal softmax assigned sharp higher probability to 5, but the temperature softmax dissipated some of the high probability into the other two classes while maintaining their relative order.

In a separate lecture [2] based on the paper, Geoffrey



(a) Cumbersome network. Logits are extracted from the Output FC layer to be used as soft labels for the distilled model after passing through a temperature softmax.



(b) Distilled network. We test this network for different values of x. The network is made to match only the hard labels when $\alpha = 1$, and only the soft labels when $\alpha = 0$, and a linear interpolation of their respective cross-entropies otherwise, as per equation 2.

Figure 2: Networks for MNIST dataset

Hinton refers to this as the "dark knowledge" from the cumbersome network, comparing how we dont really look at most of what the network has learned (in obtaining probabilities for the wrong classes) with the dark matter in the universe which isnt directly noticed but seems to affect a lot of the observable universe. The temperature defined softmax function reveals this "dark" knowledge by scaling the raw values of the logits appropriately. That is, higher temperatures tend to distribute the probabilities among all classes, thus increasing the probability scores of the next highest scoring classes. In Figure 1, the 3 looks much more like the 5 than the 1. These relative probabilities are the hidden information that enable distillation.

The smaller network minimizes two objectives. One is the cross entropy between the soft targets and the soft logits obtained during the forward pass using the same temperature that was used to obtain the soft targets. The other is the cross entropy between the hard labels and hard logits - which are obtained the same way as soft logits but with temperature of 1, i.e. the usual softmax. We can define a hyperparameter α that defines a ratio for interpolation between these two cross entropies, and end up with an expression for the final objective function as shown in equation 2. Increasing α would increase the contribution of the soft cross entropy to the net objective.

$$CE_{net} = \alpha\, CE_{soft} + (1-\alpha)\, CE_{hard} \tag{2}$$

When using an interpolation of the two cross entropies, it is important to scale up the gradient contribution of the soft cross entropy by a factor of T^2 because the gradients produced by them are scaled by $1/T^2$ as a result of the involvement of temperature in the softmax.

A key factor in distillation is the temperature at which the soft targets are calculated. Another key factor is the ratio of interpolation between the hard and the soft targets provided to the simpler network. Along with demonstrating the results, we focus on exploring the impact of these parameters on the overall training and test-time performance of the distilled network.

3. Experiments and Observations

We see significantly faster convergence during training of the distilled net with consistent performance improvements over un-distilled networks on both, the MNIST and CIFAR-10 datasets. Our distilled "simple" models are 2-layer fully connected networks, with hidden sizes [H,H] for H = 30, 100, 300 and 800. The final layer outputs are trained to match the hard labels as well as the soft logits obtained from the cumbersome network, with an interpolation parameter α, as shown in Figure 2b. Note that the temperatures used for the simple network must match with that used for the raw logits of the cumbersome network.

We investigate the effect of changing temperature T for distilled networks with different architectures, and plot the results. We study how different interpolations between hard and soft losses affect the final accuracy of the distilled net. We also replicate the experiments as explained in the paper on the MNIST dataset, like removing all occurrences of 3 from the transfer set, and training the distilled network only on 7 and 8, expecting to still get good test accuracies for all classes as in the cumbersome network. On similar lines, we train the model on the CIFAR-10 dataset without 'Cats' and check the performance.

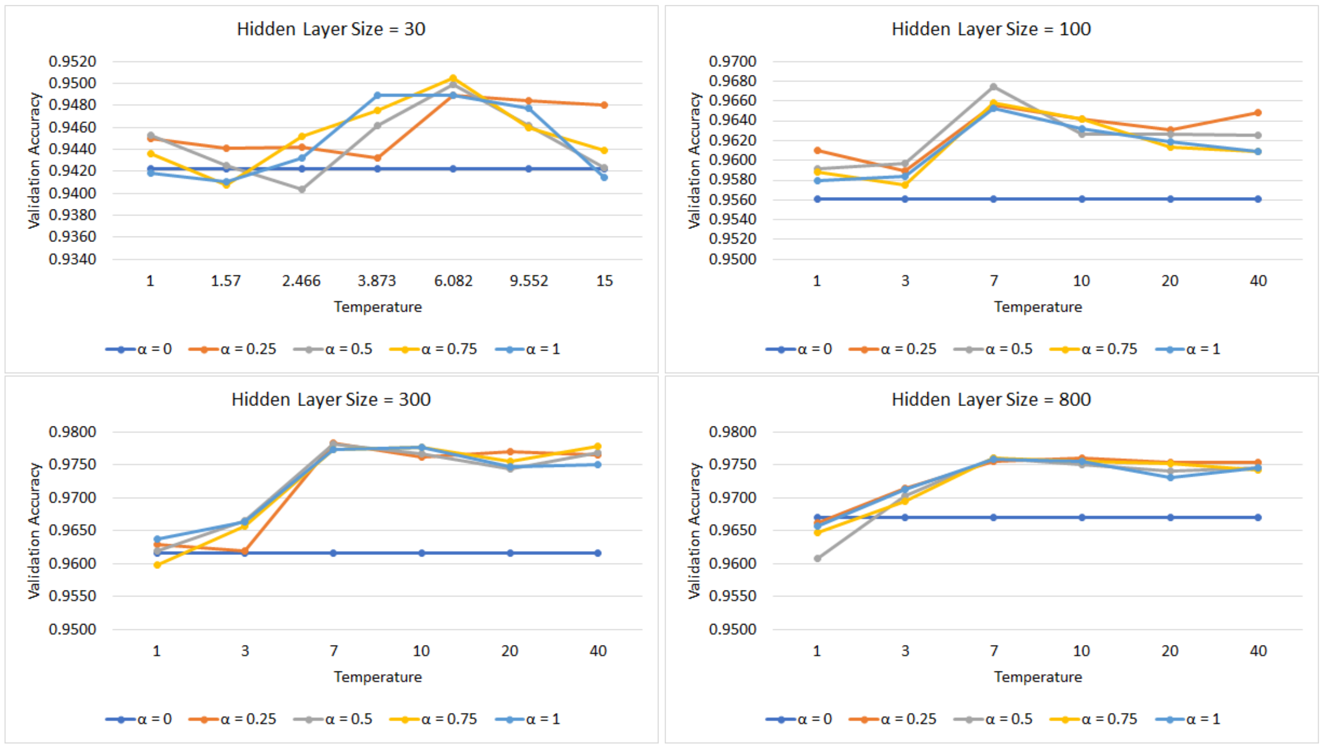


Figure 3: Validation accuracies vs Temperature for four hidden layer sizes and five values for α on the MNIST dataset

3.1. On MNIST

Figure 2a shows the specifications of our cumbersome network for the MNIST dataset. With 2 conv layers (with normalization and max-pooling), one hidden and one output layer, the network had almost 3.3 million parameters to learn. Using TensorFlow [1] as the neural network framework for our experiments, we fine-tuned the network and were able to achieve 99.28% prediction accuracy on the test set.

Figure 2b shows the distilled network, as has been explained previously. We change the number of units (x) in its hidden layers and test their performance various temperatures and values for α. The fine-tuned test accuracies for each simple model is tabulated in Table 1. For all the simple models, we see a consistent improvement in performance by the distilled model. This implies that the model does learn new information from the relative probabilities. We also notice that the training of the model is much faster (lesser epochs required to reach the same training/validation accuracy).

The results of varying temperature T and interpolation factor α are in figure 3. It is interesting that for every α, the validation accuracies peak at the same temperature, which shows that for hyperparameter tuning, temperature is more important than α. We also find that the optimum temperature increases as the hidden size increases. This is as expected since increasing the hidden size increases the model capacity, hence allowing the model to learn more from softened relative probabilities.

Table 1: Test accuracies with smaller network on the MNIST dataset. Cumbersome network has test accuracy of 99.28%.

Units in hidden layers, x	**Test Accuracy (%)**	
	With distillation	**Without distillation**
30	94.66	93.80
100	96.33	95.07
300	97.66	95.87
800	97.86	96.57

Table 2: Number of trainable parameters in the networks

Model	**Number of parameters**
Cumbersome (MNIST)	$3,273,504$
Cumbersome (CIFAR-10)	$1,067,584$
Distilled, $x = 30$	$24,720$
Distilled, $x = 100$	$89,400$
Distilled, $x = 300$	$328,200$
Distilled, $x = 800$	$1,275,200$
x is the number of units in the hidden layers	

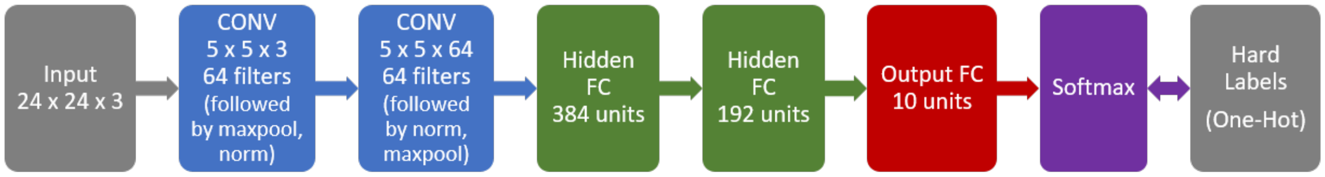


Figure 4: Cumbersome network for CIFAR-10 dataset. Logits are extracted from the Output FC layer to be used as soft labels for the distilled model after passing through a temperature softmax. Distilled network for CIFAR-10 is the same as the one used for MNIST.

3.1.1 Training without threes

As we saw in figure 1, 3 looks much closer to 5 than 1. To demonstrate that this kind of knowledge is passed on during distillation, we trained the distilled network with all the examples of 3 removed from the training set. The trained distilled network can still correctly predict 84.85% of the 3s in the test set just by increasing the bias for 3 in the output layer to compensate for the absence of 3 in the training set. This is possible because the soft targets on which the network was trained still had *some* information about the probabilities of 3 relative to other digits, even if it was never the highest probable digit itself.

Table 3 shows the confusion matrix for this network on the test set. Evidently, the next best guess for 3 is 5, and no 3 is ever labeled as 1.

Table 3: Confusion matrix for distilled network on MNIST test set after training without any examples of 3.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Errors**
0	974	0	0	2	0	0	3	0	0	1	6 / 980
1	0	1128	0	4	0	1	2	0	0	0	7 / 1135
2	2	3	1002	18	2	0	1	1	3	0	30 / 1032
3	2	0	9	857	0	109	0	3	24	6	153 / 1010
4	2	0	2	4	957	0	5	1	2	9	25 / 982
5	3	0	0	22	0	859	3	0	5	0	33 / 892
6	6	4	0	1	3	5	939	0	0	0	19 / 958
7	1	2	2	42	1	0	0	977	0	3	51 / 1028
8	5	0	3	20	3	1	2	0	938	2	36 / 974
9	1	3	0	15	6	1	1	2	1	979	30 / 1009

3.1.2 Training with only sevens and eights

Taking the class-omission experiment further, we trained the distilled network *only* on 7s and 8s. After decreasing the bias for 7 and 8 in the output layer, the network can still correctly predict 71.33% of all the previously unseen classes in the test set. Table 4 shows the confusion matrix for this network on the test set.

3.2. On CIFAR-10

Figure 4 shows the specifications of our cumbersome network for the CIFAR-10 dataset. The model has almost 1.1 million parameters. After fine-tuning we were able to achieve 86.80% prediction accuracy on the test set.

Table 4: Confusion matrix for distilled network on MNIST test set after training only on 7s and 8s.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Errors**
0	856	0	24	0	3	11	50	9	9	18	124 / 980
1	0	1068	29	0	0	10	20	0	8	0	67 / 1135
2	16	14	855	0	40	6	43	10	48	0	177 / 1032
3	40	15	392	154	3	86	95	4	195	26	856 / 1010
4	18	7	9	0	867	16	18	20	1	26	115 / 982
5	20	8	39	33	11	464	140	0	160	17	428 / 892
6	88	5	121	0	121	15	594	13	0	1	364 / 958
7	0	63	64	0	7	0	0	856	0	38	172 / 1028
8	22	5	52	4	4	33	91	0	751	12	223 / 974
9	13	28	18	0	83	3	0	13	4	847	162 / 1009

For our simple distilled model, we reuse the simple network for MNIST, as shown in figure 2b. We change the number of units (x) in its hidden layers and report the test accuracies of the best distilled and undistilled models in Table 5. The marginal improvement over the undistilled model is more than that in the MNIST dataset, with the last model gaining almost 5% accuracy improvement. This is probably because CIFAR-10 images of size [32*32*3] have much more information that is transferred, as compared to the MNIST [28*28*1] images. As a result, the advantage of using the softened logits is much more prominent with the CIFAR-10 dataset.

Figure 5 shows the variation of validation accuracy with various temperature and α values. We see results similar to that on the MNIST dataset, higher capacity networks tend to prefer higher temperatures. However, we find that higher values of α perform consistently better, implying that the model learns more while using the transfer set only. An important thing to note in this case is that the incorrectly classified images (from the cumbersome network) end up generating incorrect logits in the transfer set, and the errors made by the cumbersome network are also propagated to the simple network. Nonetheless, even with partially incorrect training data, the distilled model consistently performs better than a similar sized undistilled model. This reinforces our inference that the smaller model learns much more extra information while trying to match the soft targets.

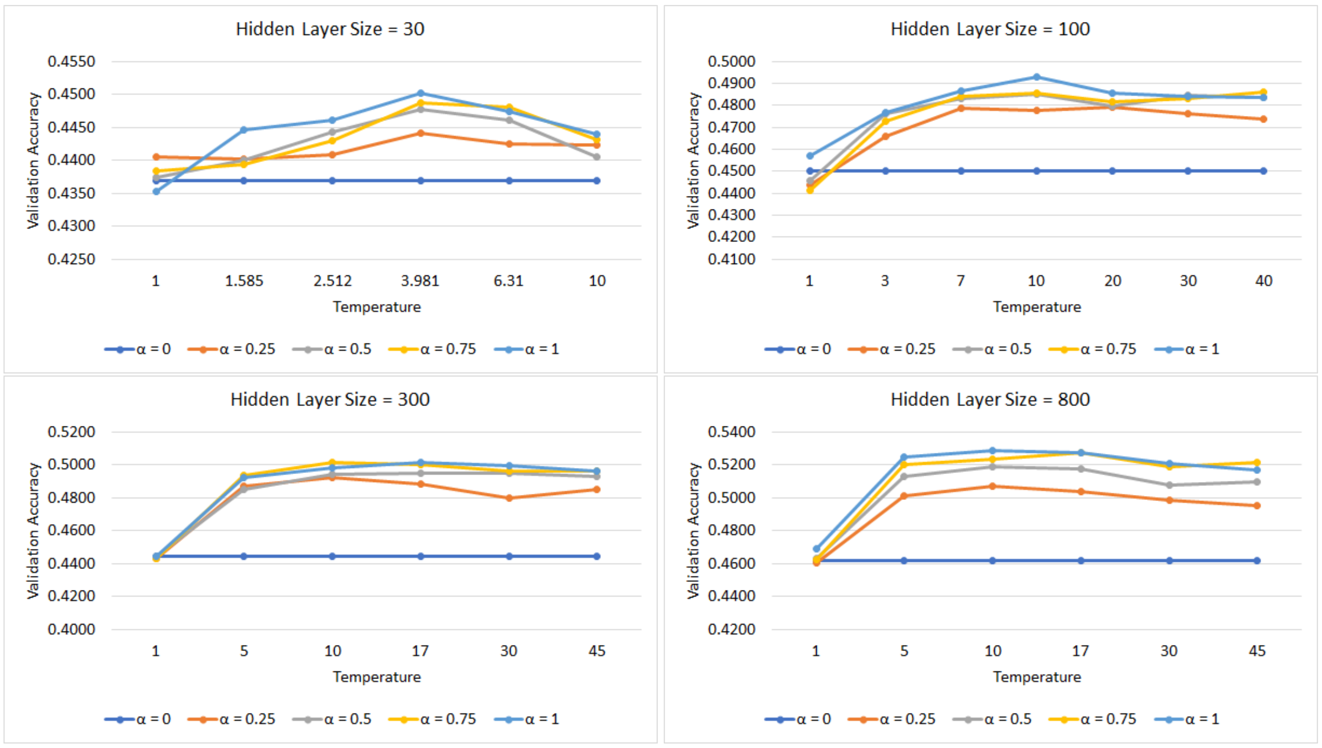


Figure 5: Validation accuracies vs Temperature for four hidden layer sizes and five values for α on the CIFAR-10 dataset

Table 5: Test accuracies with smaller network on the CIFAR-10 dataset. Cumbersome network has test accuracy of 86.80%.

Units in hidden layers, x	Test Accuracy (%)	
	With distillation	Without distillation
30	44.61	43.52
100	48.60	44.93
300	49.59	44.20
800	53.58	48.67

3.2.1 Training without 'cat' examples

We repeated the example-omission experiment with CIFAR-10 as well. After training a distilled network without any examples of 'cat', we see only 1% fall in overall accuracy on the test set. Up to 28% examples of the 'cat' class are predicted correctly in spite of never having seen any 'cat' images during training.

4. Conclusions

We demonstrated that the knowledge transferred through distillation is a treasure chest of crucial information. The "dark knowledge" found in the relative probabilities of the incorrect classes accelerates learning, and enables a low-capacity fully connected neural networks to learn concepts learnt by a deeper convolutional neural network using the temperature-softmax of its logits. It can be crucial in enabling the deployment of large complex networks repackaged into simpler networks with smaller resource footprint.

We also observe that for large distilled networks, higher temperatures are needed for optimal knowledge transfer. When using an interpolation of soft and hard targets during training, it is usually beneficial to assign a larger weight to the soft targets.

5. Future Work

The scope of our study so far was very limited on account of time as well as computational resources. Distillation of knowledge using bigger datasets like ImageNet would give more insights into the process, but require a lot of time to train. Different models of the cumbersome and simple network can give insights into what model pairs work well for distillation. One key area of work can be distillation using hidden layer logits, rather than only logits from the final layer. The hidden layer logits would store more information as the cumbersome network ultimately derives its predictions from the hidden logits. The simple model would still need to be trained on the hard labels, as that is the ultimate target.

Extending the above idea, very large and deep networks can be distilled by selecting "checkpoint" hidden layers which generate logits for the transfer set, to match with consecutive hidden layers in the simple network. This method would have multiple advantages. Checkpoints placed at regular intervals would ensure good accuracy in the simple model, simply because it provides more "conceptual" information to the simple network at every layer. The cross entropies of the checkpoint layers can be interpolated to form one final loss function to train the entire model at once. One very nice feature offered by multiple checkpoints is that a section between two or more checkpoints can be completely independently trained if all the layers before it are trained, thus allowing for step-by-step training. Essentially, this reduces the system into modules that can be trained one after the other and finally run together to mimic the cumbersome model using very few parameters and computation time.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] G. Hinton, O. Vinyals, and J. Dean. Dark knowledge. *Distinguished Lecture Series at Toyota Technological Institute at Chicago, https://www.youtube.com/watch?v=EK61htlw8hY*, 2014.

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

[4] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

[5] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database. *http://yann.lecun.com/exdb/mnist/*, 1998.