Knowledge Factorization

Anubhav Ashok **Khushi Gupta** **Nishant Agrawal**

Abstract

We propose a new technique called Knowledge Factorization which can factor the knowledge present in the large pre-trained model generic network to improve training of a smaller model on a new, smaller task. Transfer learning by reusing pre-trained weights has been shown to greatly improve performance when training on smaller datasets. However, in order to reuse these weights, we generally need to reuse the same model architecture or a contiguous subset of layers of the architecture. This may cause performance on the smaller dataset to suffer due to poor model selection or overfitting. While methods such as [1] have proposed using pruning based approaches to address this problem, these methods often require many rounds of fine-tuning for each round of pruning. Furthermore, depending on the heuristic used to perform pruning, it could possibly be discarding regularities that would be beneficial for the new task. We leverage ideas from Knowledge Distillation [2] and Multi-Task learning to solve our task.

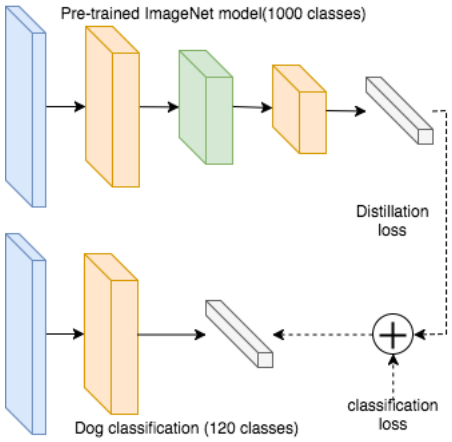

Figure 1: Example application of knowledge factorization of ImageNet model to dog classification

1 Introduction

Pretraining has proven to be an effective strategy in transferring knowledge from a larger diverse dataset to a more fine-grained task. However reusing a model of the same capacity that was trained on the larger dataset for a smaller task is prone to overfitting. Furthermore, pretraining does not allow us to directly reuse the weights of the larger model when training a much smaller architecture (the student) for a fine grained task. To address this, we introduce a concept called Knowledge Factorization. Knowledge Factorization is similar in spirit to pretraining in that we want to leverage the useful features learned on a larger dataset for a smaller task. We borrow ideas from both Knowledge Distillation, a model compression technique where knowledge from a larger teacher model is distilled to a smaller teacher model *on the same task*. Our approach attempts to "factorize" the knowledge from the teacher model to student model on *a subset of the original task* it was

10707 Class Project (2017).

trained on. This idea is useful in practice since it allows us to use pretrained models to generate smaller models for specific tasks in an **unsupervised** manner. In our experiments, we successfully demonstrate this by using a large model pretrained on ImageNet to train a smaller classifier on the Stanford Dogs dataset without using the fine-grained labels. In addition to this, we explore multiple novel variations on Knowledge Factorization and show experiments on the MNIST and CIFAR-10 datasets.

2 Related Work

The proposed task lies at the intersection of transfer learning, model compression and knowledge distillation and we cover each below.

2.1 Transfer Learning

It is the process of adapting knowledge learned from a one task to solve a different task. In case of deep neural networks, the process involves fine tuning a network trained on a generic task for a specific task. Fine-tuned approaches often outperform training from scratch. Most work in Visual Question Answering [[3], [4] object detection [5], [6], involve finetuning a network trained on Imagenet [7].

2.2 Knowledge Distillation

In [2], the authors propose a way of learning a small model that can mimic the performance of an ensemble by using the geometric / arithmetic mean of the class probabilities produced by the ensemble as "soft targets" for training of the small model.

[8] improved on this approach by matching intermediate layers to improve the performance of deeper and thinner student networks. Net2Net [9] also uses a teacher–student network system with a function-preserving transform to initialize the parameters of the student network according to the parameters of the teacher network.

More recently, [10] view the neural network as a graph through which information flows. They constrain the student network to mimic the flow of information through the teacher network. They define flow between two layers as the dot product between the feature maps present in these two layers. The loss term is then the l_2 norm between the layer-wise flow of the teacher network and the student network.

2.3 Network Pruning

Data-free parameter pruning [11] provides a data independent method for discovering and removing entire neurons from the network instead of individual weights. Deep compression [12] integrates the complementary techniques of weight pruning, scalar quantization to encode the remaining weights with fewer bits, and Huffman coding. Dynamic network surgergy [13] iteratively prunes and splices network weights. The novel splicing operation allows previously pruned weights to be reintroduced. Weights are pruned or spliced based on thresholding their absolute value. All weights, including pruned ones, are updated during backpropagation.

In [1], the concept of jointly fine tuning and pruning using a Bayesian framework is first introduced motivated by the fact that network pruning should not be task-agnostic.

3 Methods

3.1 Factorization through Knowledge distillation

Figure 2: $P(Y|X)$=[0.01, 0.01, 0.01, 0.4, 0.02, 0.5, 0.01, 0.02, 0.01, 0.01]

In this approach, we aim to perform Knowledge Factorization by distilling the information learned by the teacher model to the student model. Our motivating idea is similar to the main idea of pretraining - that we can use the features learned on a larger, more diverse dataset to improve performance on a fine-grained task.

To illustrate how this is possible with Knowledge Distillation, let's consider the example showed in Figure 1. While the label for the image is 5, the information that this example also looks like a 3 is useful information. We can obtain such a distribution over the classes by utilizing the the teacher model as supervision. While the outputs of the teacher model are indeed noisy and may contain errors, this still provides more information than using hard one-hot labels.

In our Knowledge Factorization approach, we still match the un-normalized logits between a teacher and student model as with standard Knowledge Distillation. However, unlike standard Knowledge Distillation we only use the smaller fine-grained dataset to perform distillation as opposed to the entire dataset that the teacher was trained on.

Our learning objective for this approach is:

$$\mathcal{L}_{\text{KD}}(f(x;\theta), z) = \frac{1}{N}\sum_i ||f(x^{(i)};\theta) - z^{(i)}||_2^2$$

where θ represents the weights of the student network, $z^{(i)}$ are the logits of the teacher model and $f(x^{(i)};\theta)$ is the model prediction on the i^{th} training data sample.

We also experimented with using KL divergence instead of L-2 loss. The motivation for doing so was to match distributions of the models instead of the exact magnitudes of the outputs. This would provide more robustness to the initialization of the student model as well as the expressiveness of the student architecture. The modified objective function is defined below:

$$\mathcal{L}_{\text{KL}}(f(x;\theta), z) = \frac{1}{N}\sum_i \text{KL}(f(x^{(i)};\theta), z^{(i)})$$

3.2 Dropout

In addition to the standard knowledge distillation approach we tried to see whether adding a dropout layer to the teacher outputs before computing the loss would allow the fine-grained model to learn more generalizable features. In all our experiments with dropout we used 0.1 as the dropout ratio.

3.3 Natural gradient

In standard gradient descent, distance means Euclidean distance in the parameter space. However, defining "distance" in terms of how much our parameters is adjusted isn't always correct. For example in Fig 3 the mean is changed from -1 to 1. However, the distribution on the left changes far more than the distribution on the right. This leads to the intuition that not all parameters are equal. Hence, rather than treat all parameters equally, we should scale the gradient of a parameter by how much it affects the output distribution.[1]

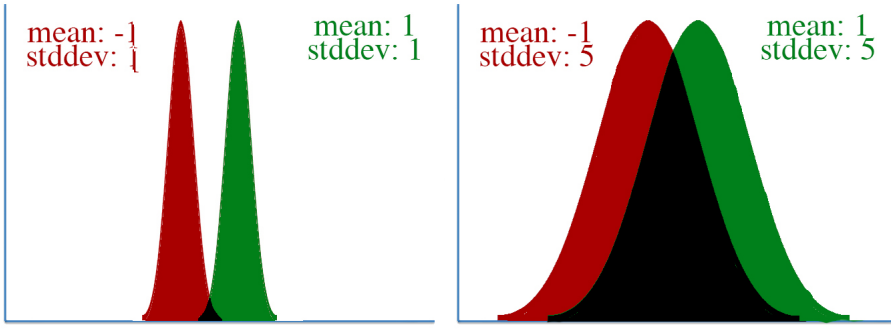

Figure 3

[1]Ref: `http://kvfrans.com/`

The essence of our idea is to shape the probability distribution of the fine-grained model to benefit from the general decision boundary learned by the larger model. In such a context, it makes sense to use Natural Gradient Descent to transfer knowledge to the fine-grained model. In Section 5.2, we show that this is indeed the case. We propose using a new objective that both ensures that the distribution of the student model moves towards that of the teacher model on the weight manifold by minimizing the KL-divergence of the teacher to the student and the KL-divergence of the old student to the new student.

$$\mathcal{L}_{\text{KD}}(f(x;\theta), z) = \frac{1}{N}\sum_i \text{KL}(f(x^{(i)};\theta), z^{(i)})$$

$$\text{such that: } \text{KL}(f(x^{(i)};\theta), f'(x^{(i)};\theta)) < \epsilon$$

3.4 Simultaneous training and distillation

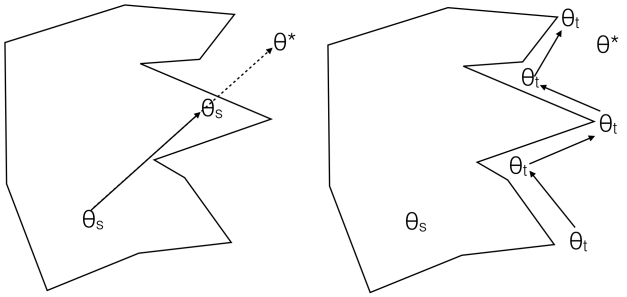

Figure 4: Left: Distillation after training, Right: Simultaneous training and distillation. In this figure, θ_s refers to the student parameters, $\theta*$ refers to the converged teacher parameters and θ_t refers to the intermediate teacher parameters during finetuning.

Instead of performing Knowledge Distillation with the pre-trained teacher model, we introduce a concept called Simultaneous training and distillation. In this approach, the pretrained teacher model is finetuned on the smaller dataset which we intend to perform Knowledge Factorization on and the student's objective is to match the moving distribution of the teacher model during each iteration of training. The high level intuition for this idea is illustrated in Figure 4, which shows the movement of the student parameters on the weight manifold. If we follow the moving distribution of the teacher during training, we hope to prevent getting stuck at a bad local minimum or saddle point. This idea also seems to be supported by the recent theory introduced by [14], where Neural Networks are viewed as an information bottleneck where training follows a "learning" and "compression" phase. More information about this and how it relates to this approach is provided in Section A of the appendix.

In this approach we match the outputs of multiple intermediate points in the teacher and student model as opposed to simply matching the final outputs. Our new objective function is:

$$\mathcal{L}_{\text{student},t}(f(x;\theta_t), z_t) = \mathcal{L}_{\text{KD}}(f(x;\theta_t), z_t)$$

where z_t is the output of the teacher model after the t^{th} training epoch.

3.5 Multidimensional Scaling

So far we looked at ways of training the student to mimic the output distribution of the teacher network. An alternative approach would be to learn the embedding space learned by the teacher network. Neural networks can be viewed as two step systems. In the first step, the input is transformed to a feature space and in the second stage a classifier is learned over this feature over the multiple classes. The idea being that the features are rich and the classifier, fairly simple.

This notion is also why when we fine tune, we typically learn the weights of the last few layers that correspond to the classifier. We can then formulate the the task of the student network as to learn the embedding space of the teacher. The simplest way to do to this is to minimize L_2 difference between the features of the student network and that of the teacher network.

$$L = \|f_{\text{student}} - f_{\text{teacher}}\|_2^2$$

This approach has two major drawbacks

- We do not care about the absolute values of the feature vectors in the latent space. Its the relative distribution of the vectors in the feature space that the classifier exploits to find a hyper plane.
- Second, this constrains the latent space dimension of the student to be the same as that of the teacher. As is well hypothesized, naturally occurring images lie on a low dimensional manifold, hence it's possible that some dimensions learned by the teacher are redundant.

We propose using multidimensional scaling (MDS) [15]. The MDS algorithm learns to place each object in an N-dimensional space such that the pairwise-object distances are preserved.

The loss can be formulated as a regression problem

$$L = \sum_{i<j}^{N} \sum_{j}^{N} (\|f(i) - f(j)\|_2 - d_{ij})^2$$

where d_{ij} is the ground truth distance matrix that we are trying to learn.

For our case, d_{ij} is the the pairwise distance between latent feature representations between two inputs i and j and N is the number of inputs.

$$L = \sum_{i<j}^{N} \sum_{j=1}^{N} (\|f_{\text{student}}(i) - f_{\text{student}}(j)\|_2 - \|f_{\text{teacher}}(i) - f_{\text{teacher}}(j)\|_2)^2$$

This approach tackles both the problems stated above. It's also worth noticing that the above loss term is completely unsupervised and can be added as an auxiliary task while training any student-teacher system.

4 Datasets

For the MNIST and CIFAR-10 datasets, we construct new data subsets containing samples from the first 5 classes (MNIST-5 and CIFAR-5 respectively). Lastly we ran experiments on ImageNet/Stanford Dogs to demonstrate our approach on a larger scale experiment.

4.1 MNIST/ MNIST-5

The MNIST [16] dataset consists of 28×28 pixel grey-scale images depicting handwritten digits. We use the standard 60,000 training images and 10,000 test images for experiments. We performed a few experiments on the MNIST dataset as a proof of concept since training time was fast and GPUs were not required to train on MNIST.

4.2 CIFAR-10/ CIFAR-5

The CIFAR-10 [17] dataset consists of 10 classes of objects and is divided into 50,000 train and 10,000 test images (32x32x3 pixels). We performed most of our experiments on CIFAR-10 since it used natural RGB images and was a slightly more challenging dataset than MNIST.

4.3 ImageNet (ILSVRC)/ Stanford Dogs

ILSVRC uses a subset of ImageNet with roughly 1000 images (224x224x3) in each of the 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images and 150,000 testing images. The Stanford Dogs dataset [18] is a subset of the ImageNet dataset that includes over 22,000 annotated images of dogs belonging to 120 species. We used these datasets to show that our approach works on a larger scale experiment. The number of classes as well as the size of the images were larger than the previous two datasets. 4 Titan-X gpus were required to train models on this dataset.

5 Results

In this section, we compare the approaches we described in Section 3 to several baseline experiments. These experiments were using the MNIST, CIFAR-10, ImageNet and Stanford Dogs datasets and subsets of them. We observe that these experiments seem to confirm our hypothesis that the Knowledge Factorization approaches we have described outperform standard techniques such as training from scratch, finetuning or pruning. We demonstrate the viability of our approach in large scale scenarios by performing an experiment that uses the distribution learned on the ImageNet dataset to improve learning on the Stanford dogs dataset.

5.1 Baselines

Since we are working with relatively small and simple datasets, the generic task (**G**) was defined as full 10 way classification for both MNIST and CIFAR-10 and the specific task (**S**) was defined as doing a 5 way classification where the 5 classes were the first five classes for both MNIST and CIFAR-10.

Furthermore, in the case on MNIST the teacher/reference model was a small 4 layered convolutional network (performance improvement was hard to establish if a more complex network was chosen) trained on **G**. For CIFAR-10 the network was VGG13 trained on **G**.

We compare our technique with three baselines described below

1. **Scratch.** Train from scratch on **S**
2. **Pretrained.** Use reference network trained on **G** to make predictions on **S**: The assumption here is that **S** is a subset of **G**, hence a network trained on **G** should be able to classify **S** off the shelf
3. **Fine tuned.** Fine tune reference model trained on **G** on **S** by replacing the classification layer - This captures how traditionally finetuning is done. In our case we just finetune the last layer, keeping the weights of the other layers fixed though generally one also updates the lower layers albeit with a smaller learning rate.
4. **Pruning.** First fine tune the reference model trained on **G** on **S** by replacing the classification layer then prune the network to get rid of redundant weights.

We hypothesize that our method performs better than all the above baselines and uses fewer parameters than the teacher model.

Table 1: Results on MNIST

Experiment	Accuracy
Baselines	
Scratch	0.995
Pretrained	0.987
Fine Tuning	**0.997**
Pruning	0.993
Knowledge Factorization	0.980
KL	**0.997**
Dropout	0.985

Table 2: Results on CIFAR10

Experiment	Accuracy
Baselines	
Scratch	0.883
Pretrained	0.837
Fine Tuning	**0.910**
Pruning	0.852
Knowledge Factorization	0.863
KL	**0.893**
Dropout	0.846
Natural Gradient	0.914
Simultaneous	0.935
MDS + CE	0.88

Table 3: Large Scale Experiments (ImageNet to Stanford dogs)

Experiment	Accuracy (Top-5)
Scratch	0
Knowledge Factorization	**0.284**

5.2 Factorization through Knowledge distillation

1. **Knowledge Distillation**: When using L_2 loss between the outputs of the student network and the teacher network we found that our model converged but did not perform the best. Using KL divergence greatly improved the results on both CIFAR-10 and MNIST.
2. **Dropout after outputs:** The dropout layer did not improve performance but instead made training less stable in the MNIST experiment. This approach performed the worst in both experiments.

5.3 Natural gradients:

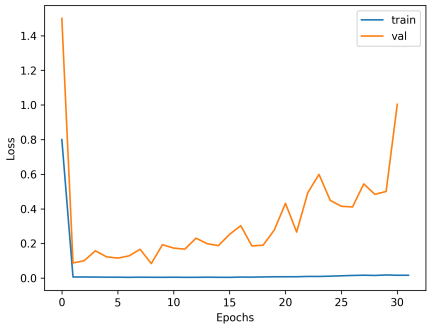

Figure 5: Training and validation losses for the Natural Gradient experiment

The validation loss for the natural gradient experiment initially decreased, achieving a peak accuracy of 91.4%. However, as training progressed, the loss increased after a certain point. We think this is due to the kl divergence being restricted to a fixed value. Annealing the value as epochs increases might solve this issue. These results are summarized in Figure 5.

5.4 Simultaneous training and distillation

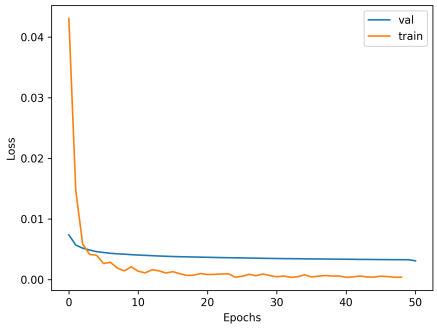

Figure 6: Loss plot for simultaneous training and distillation

For this experiment finetuned the teacher on the CIFAR-5 dataset for 50 epochs while making the students match outputs at each residual block match the teacher. We observed that this approach outperformed all the other approaches we tried, achieving a final accuracy of 93.46%. Interestingly, the student model even outperformed the higher capacity pre-trained teacher model. Figure 6 shows the loss plots for simultaneous training and distillation.

5.5 Imagenet to Stanford Dogs

In this experiment we used a ResNet-34 model trained on the ImageNet as the teacher model and a ResNet-18 model which was initialized with random weights as the student. The baseline version did not converge in 10 epochs with SGD and achieved a top-5 accuracy of 0. Our implementation on the other hand, achieved a peak accuracy of 28.4%. The final results are summarized in Table 4

5.6 Multi-Dimensional Scaling

In this experiment we used a ResNet-18 model trained on the CIFAR-10 dataset as the teacher model and a small 4 layered CNN model which was initialized with random weights as the student. We finetuned the teacher on the CIFAR-5 dataset for 50 epochs.

In order to check the value of this loss term we perform the following experiments

- Unsupervised MDS Loss
- Unsupervised MDS Loss + Supervised cross entropy loss

Table 4: MDS Experiments

Experiment	Accuracy
MDS	0.356
MDS + CE	0.888

We notice that unsupervised loss gives an accuracy of 35.6% which is much better than random chance. This seems to be a promising indication that learning the structure of the training data (pairwise distances) without any additional supervision offers some benefit to the task at hand.

While adding MDS to the CE loss does not improve performance, we do see a reasonably high performance. Also the student model is much smaller than the models used for our highest performing model.

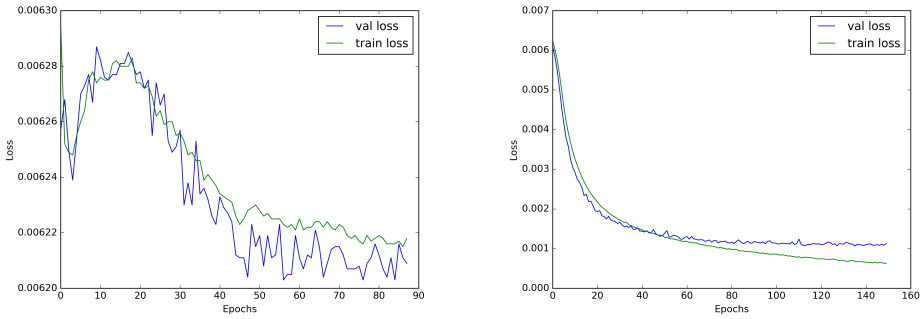

Figure 7: Left: Loss plot for MDS, Right: Loss plot for MDS with Cross Entropy Loss

6 Discussion and Analysis

In conclusion, we introduced the concept of Knowledge Factorization, where we "factorize" a general distribution to a more specific one. We show that models trained on large datasets can indeed transfer knowledge to a smaller model on a specific subset of that dataset. We tried various different approaches and compared them to several baselines. Although we have shown initially successful results for this idea, we believe that there is a lot of future directions that could be explored. Specifically, we think that the idea of unsupervised finetuning could be very powerful in the real world. If we could identify salient images in a stream of images and then transfer knowledge from a more general model using our approach, we could have a model that performs lifelong unsupervised learning. This could be extremely valuable in the web setting as works such as (Never Ending Image Learner) NEIL has shown.

References

[1] Frederick Tung, Srikanth Muralidharan, and Greg Mori. Fine-pruning: Joint fine-tuning and compression of a convolutional network with bayesian optimization. *CoRR*, abs/1707.09102, 2017.

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. cite arxiv:1503.02531Comment: NIPS 2014 Deep Learning Workshop.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[4] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. *CoRR*, abs/1611.08669, 2016.

[5] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[6] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014.

[9] Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *CoRR*, abs/1511.05641, 2015.

[10] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[11] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for deep neural networks. *CoRR*, abs/1507.06149, 2015.

[12] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[13] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *CoRR*, abs/1608.04493, 2016.

[14] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *CoRR*, abs/1503.02406, 2015.

[15] Quan Wang and Kim L. Boyer. Feature learning by multidimensional scaling and its applications in object recognition. *CoRR*, abs/1306.3294, 2013.

[16] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[18] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

A Information bottleneck theory and Simultaneous training and distillation

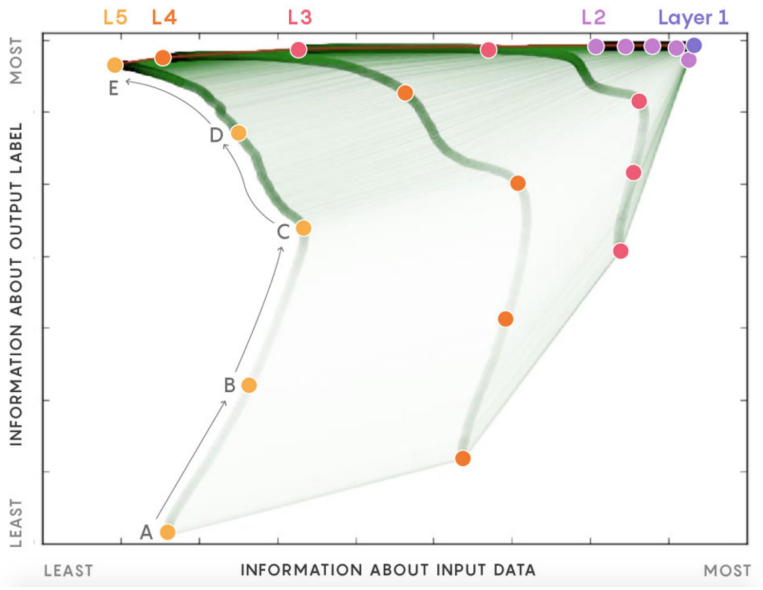

Figure 8: Visualization illustrating the relationship between the mutual information between the label and in the input as number of epochs increases

[14] introduces a new view of neural networks where the network is viewed as an information bottleneck. Figure 8 shows the relationship between the mutual information of the input and the label as training progresses. The novel observation that Tishby et. al made is that the network undergoes an initial learning process where mutual information is maximized, followed by a compression phase where the mutual information is compressed in a way that retains previous learned information (thus avoiding catastrophic forgetting).

This theory formulates a neural network as a Markov chain where the first few layers encode information about the input and the subsequent layers decode the information relevant to identifying the label. Intuitively this is seen as squeezing information that X provides about Y through a 'bottleneck' formed by a limited set of codewords $\tilde{X}$. In this context, $I_X = I(X, \tilde{X})$ defines the mutual information between the codewords and the input while $I_Y = I(\tilde{X}, Y)$ defines the mutual information between the codewords and the output label.

B Baseline plots

Here we present the plots for the baseline implementations.

B.1 MNIST

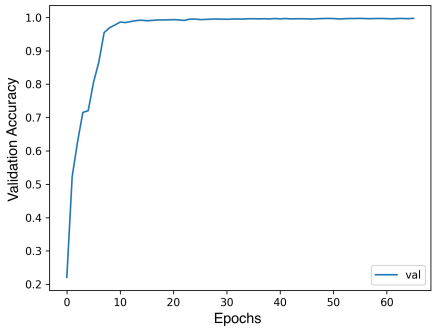

Figure 9: Training curves from scratch for MNIST

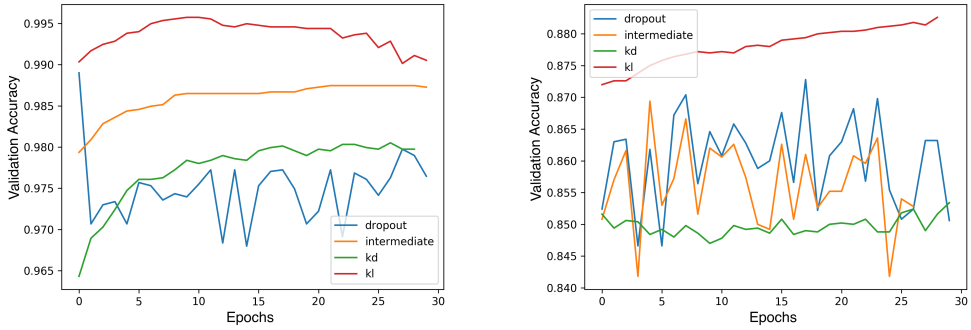

Figure 10: Left: Results for KD baselines (dropout, pruning etc) on MNIST, Right: Results for KD baselines (dropout, pruning etc) on CIFAR-10

B.2 CIFAR-10

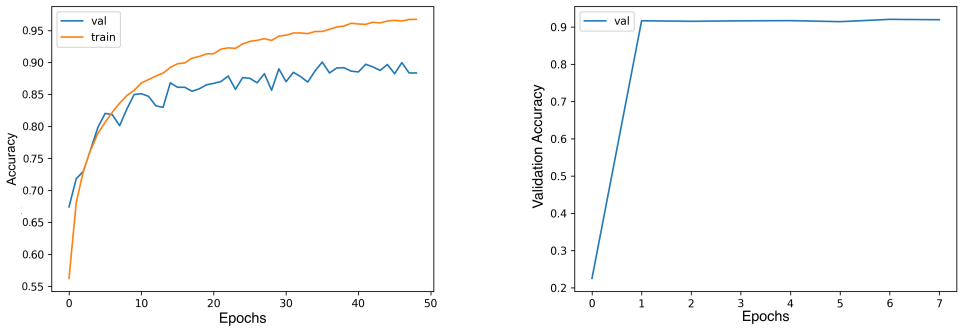

Figure 11: Left: VGG-16 trained from scratch on 5 class subset of CIFAR-10, Right: VGG-16 trained on 10 classes finetuned on 5 classes subset of CIFAR-10

B.3 Stanford dogs

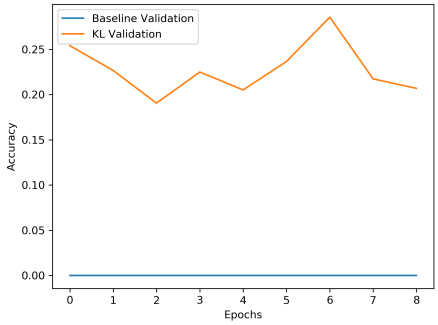

Figure 12: ImageNet to Stanford Dogs accuracy plots