Knowledge Distillation and its Variations

CS698O Final Project Report
(Group 6)

Sandipan Mandal
13807616

Teekam Chand Mandan
13744

1 Problem Statement

To implement the ideas given in the papers **Distilling the Knowledge in a Neural Network**[1] and **Fitnets : Hints for Thin Deep Nets**[3] and experiment the performance on some publicly available dataset for classification.

2 Knowledge Distillation

We have implemented ideas of Knowledge Distillation from [1] and experimented on three publicly available datasets - MNIST, notMNIST and SVHN. Additionally we also implemented three extensions to Knowledge Distillation. Details of the same are given in subsequent subsections.

2.1 Distilling the Knowledge in a Neural Network[1]

The overview of implementation of Knowledge Distillation is briefly outlined below for completeness.

- Train a wide and deep network for classification. Let it be the teacher network **T** with output $P_T = softmax(a_T)$ where a_T is the the vector of teacher pre-softmax activations.
- Let **S** be the student network with parameters W_S and output $P_S = softmax(a_S)$ where a_S is the vector of student pre-softmax activation.
- The student network will be trained such that its output P_S is similar to the teachers output P_T , as well as to the true labels y_{true}.
- Since P_T might be very close to the one hot code representation of the samples true label, a relaxation $\tau > 1$ is introduced to soften the signal arising from the output of the teacher network, and thus, provide more information during training.
- Define $P_T^\tau = softmax(\frac{a_T}{\tau})$ and $P_S^\tau = softmax(\frac{a_S}{\tau})$.
- The loss function for the student network is
  $$L(W_S) = H(y_{true}, P_S) + \lambda H(P_T^\tau, P_S^\tau)$$
  where H is the cross-entropy loss function and λ is a tunable parameter to balance both cross-entropies and W_S is the set of parameters for student network.

2.2 Adversarial Knowledge Distillation

We have not mentioned this idea in our proposal. We came up with this idea while doing the first part. It is our very own idea and if there is similar work in the literature, that is completely unintentional. As per our knowledge there is only one work which is similar to ours[4] which uses GANs to do adversarial training and their training approach is much more complicated. We would like to restate the fact that we have come up with this idea on our own and came across the work[4] later on.
Our training approach is much more simple. The overview of our idea is as follows -

- Train a wide and deep network for classification. Let it be the teacher network **T** with output $P_T = softmax(a_T)$ where a_T is the the vector of teacher pre-softmax activations.
- Train a smaller network for classification. Let it be the student network **S** with output $P_S = softmax(a_S)$ where a_S is the the vector of teacher pre-softmax activations.
- Now use the networks **T** and **S** to train a discriminator as follows -
  - Pass every example in the training set (set aside some examples for testing the discriminator) through both the networks **T** and **S**. Use the outputs a_T and a_S as features for the discriminator and 0 or 1 as label of these features (0 for **T** and 1 for **S**).
  - Train the discriminator so that it can output whether the given pre-softmax activation is coming from teacher or student network with reasonable accuracy.
- The loss function for the student network when training with adversarial Knowledge Distillation is -

$$L(W_S) = \alpha H(y_{true}, P_S) - \beta L_{class}(a_S|x = 1)$$

  where H is the cross-entropy loss function and α, β are tunable parameters and W_S is the set of parameters for student network.
  $L_{class}(a_S|x = 1)$ is the classification loss for the output a_S given that the feature is coming from the student network.
- To calculate $L_{class}(.)$, we pass a_S through the trained discriminator to get the probability of it belonging to either student or teacher network. We know the true label to be 1 (i.e, student network). Using this information the given loss function can be calculated.
- Minimizing $L(W_S)$ leads to minimizing $H(y_{true}, P_S)$ and/or maximizing $L_{class}(a_S|x = 1)$. Intention behind minimizing the first term is obvious. Maximizing the second term is equivalent to fooling the discriminator into believing that the feature is coming from the teacher network, i.e, the training procedure tries to make the pre-softmax activations coming from student and teacher network indistinguishable (which is the objective of any adversarial training routine).

2.3 Knowledge Distillation with GANs

Here we generalize the approach we have taken earlier. It is based on GANs. Though it uses GANs, the loss function we used is different from the one used by [4]. In this approach instead of pre-training the discriminator, we train discriminator and student network on the go much like GANs. We assume the logits generated from teacher network as **real** sample and the one generated from student network as **fake** sample. The loss function we used is slight modification of the generic GAN loss. For the sake of completeness, we summarize the approach below -

- Train a wide and deep network for classification. Let it be the teacher network **T** with output $P_T = softmax(a_T)$ where a_T is the the vector of teacher pre-softmax activations.
- Decide on structure of student **S** and discriminator **D** network.
- We simultaneously optimize two loss functions - one for discriminator and one for the student network.
- Let an arbitrary input to the network be **x**. Then we define $P_{real}(x) = \sigma(D(T(x)))$ and $P_{fake}(x) = \sigma(D(S(x)))$. That is $P_{real}(x)$ is the probability that the logit is computed by the teacher network **T** and correspondingly $P_{fake}(x)$ is the probability that the logit is computed by the student network **S**.
- The loss functions are-

$$L_{Discriminator} = -\gamma[log(P_{real}(x)) + log(1 - P_{fake}(x))]$$

$$L_{Student} = \beta H(y_{true}, P_S) - \alpha log(P_{fake}(x))$$

  where H is the cross-entropy loss function and $P_S = softmax(a_S)$ where a_S is the the vector of student pre-softmax activations.

2.4 Knowledge Distillation with Wasserstein GANs

This approach is similar to the last except that we used Wasserstein GANs instead of traditional GANs. This resulted in further improvement of accuracy of Knowledge Distillation. Only difference with previous approach is the loss function for Discriminator and Student Network.
Let an arbitrary input to the network be **x**. Define $f_{real}(x) = D(T(x))$ and $f_{fake}(x) = D(S(x))$. The loss functions are -

$$L_{Discriminator} = -\gamma[f_{real}(x) - f_{fake}(x)]$$

$$L_{Student} = \beta H(y_{true}, P_S) - \alpha f_{fake}(x)$$

where H is the cross-entropy loss function and $P_S = softmax(a_S)$ where a_S is the the vector of student pre-softmax activations.

3 Fitnets : Hints for Thin Deep Nets[3]

The approach of this paper relies on the Knowledge Distillation approach of the previous paper. This method is used to train a student network which is deeper but also thinner than the teacher network. The method is briefly outlined below.

- Train a wide and deep network for classification. Let it be the teacher network **T**. Also let W_{hint} denote the weights upto the middle layer of this network.
- Now first design as student network **S** which is deeper but thinner than the teacher. Let W_{guided} denote the weights upto the middle layer of this network.
- The training of the student proceeds in two stages -
  - Since the student network is thinner, the output of the 'guided' layer will not have the same dimensions as the output of the 'hint' layer. So we add a regressor layer on the top of 'guided' layer temporarily to have their dimensions aligned.
  - First stage of training is to train the layers of the student network till the 'guided' layer using the following loss function -

    $$L(W_{guided}, W_r) = \frac{1}{2}\|u_T(x, W_{hint}) - r(u_S(x, W_{hint}), W_r)\|^2$$

    where r is the regressor function and u_T and u_S are the the teacher/student deep nested functions up to their respective hint/guided layers.
  - The second stage of training is to use the parameters W_{guided} to train the remaining layers of student network by applying knowledge distillation.

We implemented this approach in our project. But other than for MNIST dataset, we did not have time to make it work. The performance on MNIST is tabulated in Table 2.

4 Experiments

We implemented all the the ideas mentioned above in tensorflow. We have used the datsets - MNIST (handwritten digits from 0-9), notMNIST (letters A-J), SVHN (digits from 0-9).
To avoid giving unfair advantage to some network due to random initialization, we used same starting weight to implement Knowledge Distillation (vanilla KD and its variations).
Network Description of student and teacher networks for all the datasets are given below.

4.1 Dataset and Architectures

4.1.1 MNIST

The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.
The architectures used -

1.
- **Teacher:** A 2-hidden layer neural network with each layer having 1200 neurons. Number of parameters = 2392800.
- **Student:** A 2-hidden layer neural network with each layer having 800 neurons. Number of parameters = 1275200.

2.
- **Teacher:** A neural network with two convolution layers(conv,relu,max-pool) followed by two fully connected(fc) layers. The conv layers project input to $64-$D feature maps and the fc layers have 1000 neurons in each layer. Number of parameters = 3250000.
- **Student:** A neural network with three convolution layers(conv,relu,max-pool) followed by three fully connected(fc) layers. The conv layers project input to $16-$D feature maps and the fc layers have 200 neurons in each layer. Number of parameters = 106400.

4.1.2 notMNIST

This dataset was created by Yaroslav Bulatov by taking some publicly available fonts and extracting glyphs from them to make a dataset similar to MNIST. There are 10 classes, with letters A-J. We have used the code from `https://github.com/davidflanagan/notMNIST-to-MNIST` to convert notMNIST dataset to the same format as in MNIST so that we can reuse the networks from MNIST. The architectures used -

1.
- **Teacher:** A 3-hidden layer neural network with each layer having 1200 neurons. Number of parameters = 3832800.
- **Student:** A 3-hidden layer neural network with each layer having 800 neurons. Number of parameters = 1915200.

2.
- **Teacher:** A neural network with two convolution layers(conv,relu,max-pool) followed by two fully connected(fc) layers. The conv layers project input to $64-$D feature maps and the fc layers have 1000 neurons in each layer. Number of parameters = 3250000.
- **Student:** A neural network with three convolution layers(conv,relu,max-pool) followed by three fully connected(fc) layers. The conv layers project input to $16-$D feature maps and the fc layers have 200 neurons in each layer. Number of parameters = 106400.

4.1.3 SVHN[2]

SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images. We have used the code from `https://github.com/oliviersoares/mnist` to convert SVHN dataset to the same format as in MNIST so that we can reuse the networks from MNIST. The architectures used -

1.
- **Teacher:** A 2-hidden layer neural network with each layer having 4096 neurons. Number of parameters = 20029440.
- **Student:** A 2-hidden layer neural network with each layer having 3072 neurons. Number of parameters = 11876352.

2.
- **Teacher:** A neural network with two convolution layers(conv,relu,max-pool) followed by two fully connected(fc) layers. The conv layers project input to $64-$D feature maps and the fc layers have 1000 neurons in each layer. Number of parameters = 3250000.
- **Student:** A neural network with three convolution layers(conv,relu,max-pool) followed by three fully connected(fc) layers. The conv layers project input to $16-$D feature maps and the fc layers have 200 neurons in each layer. Number of parameters = 106400.

4.2 Results

Experimental Result for architecture - 1 is Tabulated in the table1. This part of the experiment was done before mid-term and doesn't contain data using GAN or WGANs.

Table 1: Experimental Results for architecture - 1

Dataset	Number of Mis- Classifications				No. of test examples
	T	S (initial)	S (KD)	S (adver. KD)	
MNIST	476	597	583	**375**	10000
notMNIST	1200	1344	1321	**1089**	10000
SVHN	8339	9101	9014	**8659**	26032

Experimental Result for architecture - 2 is Tabulated in the table2. This part was done after mid-term and contains the the KD method using GANs and WGANs.

Table 2: Experimental Results for architecture - 2

Dataset	# Mis- Classifications						
	T	S	KD	adv. KD	GAN KD	WGAN KD	Guided KD
MNIST	611	1433	809	856	753	**565**	737
notMNIST	1393	2264	1760	1615	1702	**1397**	-
SVHN	5199	19419	16652	12964	12797	**10816**	-

We want to bring it to notice that in our experiments **Adversarial Knowledge Distillation**, **GAN Knowledge Distillation** and **WGAN Knowledge Distillation** all outperforms **Knowledge Distillation** method given in [1] by a decent margin as can be seen from Table2.

5 Conclusion

This project is motivated by our interest to learn smaller networks to solve visual tasks giving performance comparable to large and complex networks.
Knowledge Distillation[1] was one of the early papers on the same. In this project we implemented the ideas (section - 2.1) presented in this paper and experimented on three different datasets for classification. On all three datasets Knowledge Distillaion did improve the performance of smaller student network.
From the way the KD task is implemented we thought of another idea to do knowledge distillation - through adversarial training (section - 2.2). It improved the performance over vanilla Knowledge Distillation on most of the tasks (except for MNIST dataset using architecture-2).
After implementing the idea, we realised, doing Knowledge Distillation using adversarial training has already been proposed in [4] where they use GANs. So in our third approach we tried using GANs for Knowledge Distillation (section - 2.3). The apporach is similar to [4], but the loss functions are not the same. This approach further refined the result over simple adversarial approach.
Finally, we thought of using Wasserstein GANs (section - 2.4) to do the same. To our surprise this approach outperformed other approaches by a large margin.

The effectiveness of Knowledge Distillation can be specifically seen for SVHN dataset using architecture-2. The student network produced 25% accuracy and even vanilla KD could only increase the accuracy to 36%. But the other modifications vastly outperformed them with adversarial KD, GAN KD, WGAN KD reaching 50.1%, 50.8% and 58.4% respectively. So, through this project we have given empirical evidence of the superiority of other approaches over vanilla Knowledge Distillation.

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

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

[4] Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. Learning loss for knowledge distillation with conditional adversarial networks. *arXiv preprint arXiv:1709.00513*, 2017.