Smaller Networks for State of the Art Classification
CS698O Mid-Term Project Report
(Group 6)

Sandipan Mandal
13807616

Teekam Chand Mandan
13744

1 Introduction

Deep Neural Networks have been very successful for visual recognition tasks. But generally the best of architectures are very wide and deep comprising of numerous parameters. These models generally occupy lots of memory as well as bandwidth. So even during inference the systems take significant amount of time and memory. These problems hinder the models to be deployed on mobile or cloud platforms. So we would like to have smaller networks having lesser number of parameters occupying lesser memory and bandwidth whose performance is comparable to the larger networks during inference time so that it can be deployed on mobile and cloud platforms. In this project we would like to explore some of the works in this domain.

This project is primarily based on the seminal works **Distilling the Knowledge in a Neural Network**[1] by *Geoffrey Hinton, Oriol Vinyals, and Jeff Dean*, **Fitnets : Hints for Thin Deep Nets**[3] by *Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio*. If time permits we would like to explore the work **XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks**[2] by *Mohammad Rastegari, Vicente Ordonez, Joseph Redmon and Ali Farhadi.*

2 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. We would like to implement Fitnets and Knowledge Distillation on Tensorflow.

As mentioned earlier if time permits we would like to implement Binary Network and/or X-Nor Network from the paper **XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks**[2] and similarly experiment on some datsets for classification.

3 Progress Report

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

3.1 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.

3.2 Adversarial Knowledge Distillation (Novel Idea)

The overview of this 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).

We want to mention here that in our experiments **Adversarial Knowledge Distillation** outperforms **Knowledge Distillation** method given in [1].

4 Experiments

We implemented both 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 a pre-trained student network as the starting point to implement Knowledge Distillation (both vanilla and adversarial).
Network Description of student and teacher networks for all the datasets are given below.

4.1 MNIST

- **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.

4.2 notMNIST

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.

- **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.

4.3 SVHN

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.

- **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.

Experimental Result is Tabulated in the following table -

Table 1: Experimental Results

Dataset	Number of Mis- Classifications					No. of test examples
	T	S (initial)	S (w/o KD)	S (KD)	S (adver. KD)	
MNIST	476	593	597	583	**375**	10000
notMNIST	1200	1398	1344	1321	**1089**	10000
SVHN	8339	9224	9101	9014	**8659**	26032

References

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

[2] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[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.