Mentee Networks

Vivek Singh 1209521349 vsingh22@asu.edu

Jau-Yuan Shiao 1209386552 jshiao@asu.edu

Abstract— Deeper and deeper neural networks are created using large number of layers and large datasets. These pretrained networks are used by computer scientist with varying degree of success but often these large networks are not portable. In this project we implement concept of Mentee Networks which is training mid sized networks on small datasets from scratch and also getting supervision from a large pre trained network which acts as a Mentor for the smaller network.Such Midsized networks provide more generalization as they are trained using our target dataset from scratch and can give better accuracies.

1. INTRODUCTION

There are large number of pretrained neural networks available such as VGG-19, R-CNN.Computer Scientist use these kinds of large pretrained networks by tuning some paramters for various tasks.These networks are trained using large datasets and the accuracy depends on how similar is the data used by the computer scientist to the training data .

Tuning the parameters of such networks works to certain extent but there is no guarantee that the features in the pretrained neural network are best representation of the target dataset.But training deep neural networks from scratch faces various problems such as GPU insufficiency, time constraints etc.In this project, we create a shallower mentee net- work learn the same representation as a larger, well-trained mentor network at various depths of its network hierarchy. Mentorship happens by tagging on to the loss of the mentee network, a dissimilarity loss for each layer that we want mentored.These mentee networks can learn better features for the target data.

In this project we try to create shallower mentee network and for sake of simplicity we are using MNSIT data to train both our mentor and mentee network.The rest of the report is organized as follows section 2 discusses the related work

2. RELATED WORKS

The concept of dark knowledge was used by HInton et al. which tried to make networks portable by learning the softmax output. of mentor network.Using this concept they were trying to make mentee network mimic same mapping that of mentor network.The main drawback of this was that they relied on fact that the mentor network was trained perfectly.There are other extensions of dark knowledge All these method use the same concept of dark knowledge using the softmax layer.Their mentee network are much more deeper than the mentor network making the model more complex.

3. MENTOR TO MENTEE KNOWLEDGE TRANSFER

For getting the supervision from the Mentor network we define probes.Probes are the path of knowledge flow from neuron in Mentor network to neuron in Mentee network.Consider a Mentor network with n layers M_n pretrained on large dataset D.Suppose we represent the kth neuron activations of the ith layer in the network as $M_n(i, k)$. Consider smaller mentee network with m layer S_m.

We define probe as error measure between $M_n(l)$ and $S_m(j)$ which is modeled as RMSE error as follows,

$$\Psi(l,j) = \sqrt{1/a \sum_{i=0}^{a} (Mn(l,i) - Sn(j,i))^2} \quad (1)$$

where a is the minimum number of neurons between mentor and mentee layer.By adding this the cost of our Mentee networks.By adding this cost during backpropogation learn not just a discrminative enough representation for the samples and labeles but also for layers j in a pre determined set of layers,a representation closer to the one produced by M.

Thus,Our overall network cost becomes,.

$$e = \alpha_t Ls(d_l^b) + \beta_t \sum_{all\,l,j} \Psi(l,j) + \gamma_t \Psi(m,n) \quad (2)$$

where L(.) is the network loss for the mini batch, α_t and β_t weighs two losses together to enable balanced training and γ_t is the weight of probe between two softmax layers.These extra probe terms make weight exploration for mentor network in direction of the mentor network while not letting gradient to explode or vanish..

In the (2) equation the second and third term act as penalties for the difference in activations that do not match with the mentor network.These losses are modeled using RMSE error mentioned in equation 2.These penalties act as guidance for weights in mentee network towards weights of mentee network.

Now the new update rule for weights with probes any layer in mentee network now becomes as,

$$w^{t+1} = w^t - \eta \Big[\alpha_t \frac{\partial}{\partial w} \mathcal{L}_s + \beta_t \sum_{\forall (l,j) \in B} \frac{\partial}{\partial w} \Psi(l,j) + \gamma_t \frac{\partial}{\partial w} \Psi(n,m) \Big] \quad (4)$$

t is the number of iterations,η is the learning rate and last two term here act as guidance for the mentee network from the mentor network.The value of β(t) and γ(*t*) decreases with every iteration.β(t) can be considered as the weight for the RMSE error of probes and γ(*t*) can be considered as weight for RMSE error of softmax layer.By varying these values according to iteration we can make our mentor netwok learn according to labeled and under guidence of mnetor network.

These values produces various configuration of our mentee network which we have discussed in section 5.

4. IMPLEMENTATION DETAILS

4.1. MNIST Dataset

MNIST dataset is a dataset that contains the handwritten digits with their correct classification. This dataset is widely used to test the classification ability of neural networks. The dataset we are using contains the 28 x 28 pixels greyscale images. So the input layer of our neural network contains 784 neurons and the output layer contains 10 neurons that correspond to each digit.

4.2. MENTOR AND MENTEE ARCHITECTURE

We tested the network with different number of neurons and finally decide to use two layers mentor network and one layer mentee network. In the mentor network, we have 100 neurons in the first hidden layer and 15 neurons in the second hidden layer. The mentor network givest 98.2% accuracy after training 50 epochs and the batch size is 10. In mentee network, we have 15 neurons in the only hidden layer that is the same number as the second hidden layer of mentee.

We have set the probe between the second hidden layer of mentor and only hidden layer of mentee. Since we choose the same number of neurons in these two layer, we can easily set up the probe from i-th neuron in second hidden layer of mentor to i-th neuron in only hidden layer of mentee as seen in fig 1..

In the figure 1 we have show connection between softmax layer between mentor and mentee network this denotes last term in update rule mentioned in equation (4).

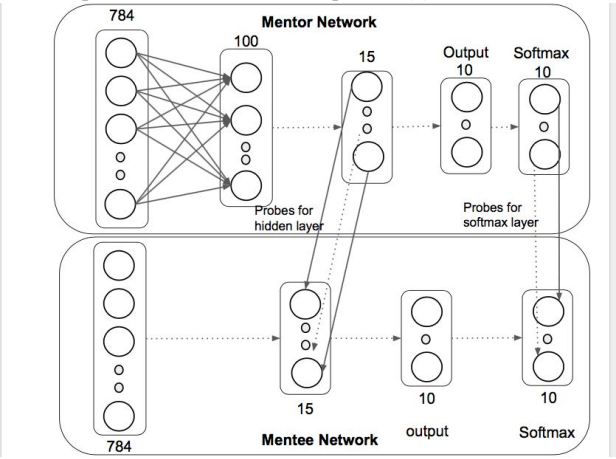

Figure 1. Mentor and Mentee Architecture

4.3 NEURAL NETWORK LIBRARY

We have used numpy based Neural network library developed by Michal Daniel Dobrzanski[4].This library has a simple implementation and does not using any computation libraries such as keras and theanos.This library does not have implementation of softmax layer and gives final output using maximum of final output layer.The backpropagation algorithm is implemented using the chain rule of differentiation without using any external libraries.We would be extending the implementation of this chain rule to implement the update rule as mentioned in equation (2).

4.4 TEMPERATURE SOFTMAX IMPLEMENTATION

In mathematics softmax is a generalized logistic function that compresses k-dimensional vector z of arbitrary values to k-dimensional vector σ(z) of values in range(0,1). The function σ(z) is defined as,

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

In Neural Network the activation function gives a value in final layer and to convert the value into probability of prediction of each class softmax function can be used accurately.

Then the class ω_i of the given input can be found by finding the class having maximum probability or softmax value.

In our implementation we added the softmax layer after the final output layer.This softmax layer takes the input from final output layer and produces probability corresponding to each class from 0 to 9 for MNSIT data.We also used this softmax in case of backpropagation to find error from final layer and propagate that error to previous layers.

4.5 IMPLEMENTATION OF PROBES

We have connected mentor and mentee layer with 15 neurons to act as probes as shown in figure 1.The knowledge from mentor network flows from mentor to mentee network via these probes and the weights are updated using the new update rule.

We first trained our mentor network over the MNSIT data and saved its parameters to load it later to create our mentor network.

Now the training data from MNSIT data is sent to both the networks batch by batch.The Mentor network generates the output and the activations for every neuron, this data is used by mentee network for guidance.Then use these values to calculate the following

$$\sum_{\forall(l,j)\in B} \frac{\partial}{\partial w}\Psi(l,j) \cdot \frac{\partial}{\partial w}\Psi(n,m)$$

terms from equation (4).

Once these values are calculated we update the weight using the new update rule.This process is repeated for many iteration.We also set the the annealing function for β(t) and γ(*t*) decreases with every iteration.

5. VARIOUS CONFIGURATION OF MENTEE NETWORK

We studied the with various configurations of α(t) , β(t) and γ(*t*) mentioned in the paper[1].

Equation (4) has three parameters α , β and γ .Different configuration of these network produces different characteristics of mentor network.

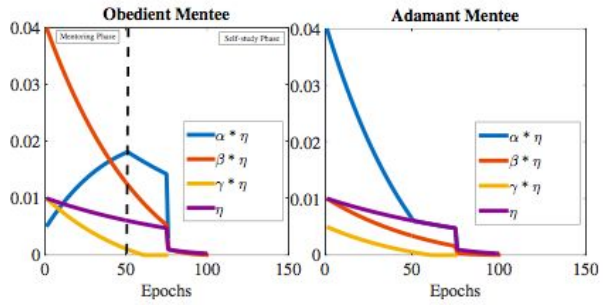

Figure 2 Configuration α , β and γ

Figure 2 shows configuration of α , β and γ as mentioned in the paper[1].We have also used similar configurations to create our mentor network

An obedient network can be defined as network that at begining tries to learn more and more from mentor network i.e from the label cost.Once good representation is leant it starts to focus on label space.We modeled such type of network in our implementation of mentee network using monotonically decreasing annealing function for β(t) and $\gamma(t)$,and also decreasing value of α(t).We started from greater values of β(t) and $\gamma(t)$ which monotonically decreases at t increases.

In this configuration we saw

We tried two configuration for β(t) and $\gamma(t)$ .Figure 3 depicts our configuration results for obedient and adamant network.

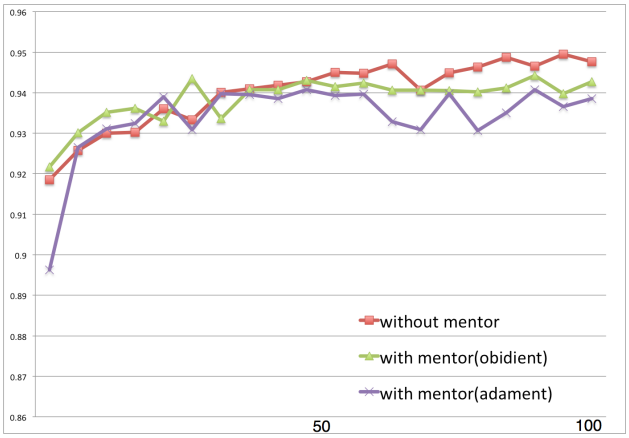

Figure 3 Training accuracy for Mentee Network

Figure 3 shows the accuracy of obedient and adamant network compared to adamant network when configuration from figure 2 is used.We see that the obedient network performs better than the adamant network.

The obedient network accuracies are close to the mentor network while in case of adamant network the accuracies vary away from mentor network.

There is also third configuration for the mentee network.In this configuration we make β(t) =0 and $\gamma(t) = 0$ .Here the network learns just from the labeled space.

6. MENTEE NETWORK WITH VARIATION OF MNSIT DATA

In this section we study the mentee network which mentored by mentor network trained on original MNSIT data.Here Mentee Netwrok is ran over variations of MNSIT data as follows,f from original MNSIT dataset new variation of it are created.The generative process has following steps:

1. Pick a sample(x,y) from original dataset
2. Create new version of it using some variations such as rotation, translation, background insertion
3. Add this new data set to form new MNSIT data
4. GO to 1 until enough samples are generated

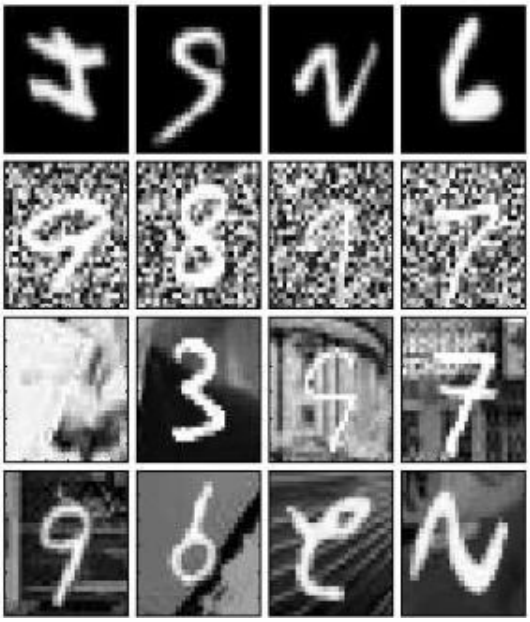

Figure 4.Variations of MNSIT dataset

we used three types of variation of MNSIT dataset

1. MNSIT-rot:The digits were rotated by an angle generated uniformly between 0 and 2π radians. Thus the factors of variation are the rotation angle and those already contained in MNIST
2. MNSIT-back-rand: A random background image was inserted in the digit image.Each pixel value of the back- ground was generated uniformly between 0 and 255
3. MNSIT-back-image: a random patch from a black and white image was used as the background for the digit image. The patches were extracted randomly from a set of 20 images downloaded from the internet.

To conduct our experiment we first trained the Mentor network from scratch.Now we used this Mentor network to guide our Mentee network. For sake of comparison we are doing three things.Firstly, Training Mentor network for variation data to get the accuracy but this is not used for mentoring.

Secondly , we are running mentee network for these variation twice i.e with and without guidance from the mentor network.The following table 1 gives the output of our experiment with varied MNSIT dataset.

Data/Configuration	**Mentor Network**	**Mentee**	**Independent**
MNSIT-ROT	84.58%	82.8%	81.09%
MNSIT-BACK-RAND	94.64%	87.77%	86.6%
MNSIT-BACK-IMAGE	82.76%	76.54%	75.23%

Table 1 Accuracies for varied MNSIT data

7. CONCLUSION AND FUTURE WORK

It took us lot of time to train the mentor network for 100 epochs to get around 97% accuracy. Also, it will take lots of time to train the mentee network if we have much more layers. After doing these training works, we finally find out the importance of shorten the training epochs and time. Using mentor network to help training a smaller network is an efficient way to do that.

There are also some parameter that will affect the mentee networks. We took lot of time to figure out the value of β. If we choose a large value for β, than the mentee might become very unstable in the beginning.
In our case, we find out that the obedient network learn faster in the beginning and the adamant gives less accuracy compared to obedient network.

In this project we saw that by training our mentee network under guidance of mentor network we are able to make our neural network learn more features for our target data thus improving the classification accuracy.

REFERENCES

[1] R. Venkatesan and B. Li. Diving deeper into mentee networks. arXiv preprintarXiv:1604.08220, 2016.

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

[3] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio.Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550, 2014.

[4] Neuaral Network and Deep learning Book https://github.com/mnielsen/neural-networks-and-deep-learning

[5] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Proceedings of the 24th international conference on Machine learning, pages 473–480. ACM, 2007.