Learning Ensembles of Convolutional Neural Networks

Liran Chen
The University of Chicago
Faculty Mentor: Greg Shakhnarovich
Toyota Technological Institute at Chicago

1 Introduction

Convolutional Neural Networks (CNN) have demonstrated impressive performance in image classification. When fitting complex models with non-convex objectives to train the network, the resulting model depends on stochastic learning procedure, i.e., the final network trained with gradient descent depends on factors such as the order of data in each epoch, initialization, learning rates, etc.

Ensemble learning is a method for generating multiple versions of a predictor network and using them to get an aggregated prediction. Given a learning set Ω consists of data $\{(y_n, \vec{x}_n), n = 1, ..., N\}$ where y is the class label and $\vec{x}$ is the inputing feature, we train a predictor $\varphi(\vec{x}, \Omega)$. With different initialization, we obtain a series of predictors $\{\varphi_k\}$. Our object is to use the $\{\varphi_k\}$ to get a better predictor, φ_A.

In the last few years, several papers have shown that ensemble method can deliver outstanding performance in reducing the testing error. Most notably, (Krizhevsky et al., 2012) showed that on the ImageNet 2012 classification benchmark, their ensemble model with 5 convnets achieved a top-1 error rate of 38.1%, compared to the top -1 error rate of 40.7% given by the single model. In addition, (Zeiler & Fergus, 2013) showed that by the ensemble of 6 convnets, they reduced the top -1 error from 40.5% to 36.0%. In 1994, Breiman introduced the concept of bagging, which helped us gain some understanding of why the ensemble of classification tree and regression tree work when they were trained by random samples from the whole dataset (Breiman, 1996). However there is still no clear understanding of

why the ensemble of CNNs performs so well, what is the relation between the number of models in ensemble and the amount of error reduced, or other methods of ensemble instead of averaging prediction.

2 Experiments

2.1 The Data Set

The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten zip code digits provided by the U.S. Postal Service. This dataset is commonly used for training various image processing systems. The database contains 60,000 training images and 10,000 testing images. The digits have been size-normalized and centered in a fixed-size image.

2.2 The Architecture

The architecture of our network contains three learned layers - two convolutional layers, $H1$, $H2$ and one fully-connected layer, $H3$. The output of the fully-connected layer is fed to a Softmax layer which produces a vector of length 10. Each element of the vector represents the probability that the input belongs to a certain class (from 0 to 9). We construct our CNN in a similar way as (LeCun et al., 1989) did.

The first convolutional layer, $H1$, is fed by a 28×28 normalized input image. This layer consists 12 14×14 feature maps, designated as $H1.1$, $H1.2$, ..., $H1.12$. Each pixel in each feature map in $H1$ takes input on a 5×5 receptive field on the input plane. In $H1$, pixels that are next to each other have their receptive fields two pixels apart. For pixels in a certain feature map, all receptive fields share the same set of 5×5 weights. However, pixels in another feature map share a different sets of 5×5 weights. Thus, in total, there are 12 sets of 5×5 weights to create 12 features maps for $H1$. Before being fed to $H2$, we operates a nonlinear ReLU transformation to all pixels in all maps.

From $H1$ to $H2$, a similar process occurs - convolution and nonlinear transformation. $H2$ consists 12 $7 \times 7 \times 12$ feature maps, each contains 12 units arranged in a 7×7 plane, designated as $H2.1$, $H2.2$, ..., $H2.12$. For $H2.1$, pixels in the first unit take a 5×5 receptive field from $H1$ and share a set of 5×5 weights, pixels in another unit share a different set of 5×5 weights, and so on. Thus, to obtain $H2.1$, we need a set of weights sized $5 \times 5 \times 12$.

In total, $H2$ is created from 12 sets of $5 \times 5 \times 12$ weights. The output of $H2$ are 12 $7 \times 7 \times 12$ feature maps, which are then fed to $H3$.

$H3$ is a fully-connected layer, which consists of 30 units, each is produced from the dot product of $H2$ and 30 sets of weights each sized $7 \times 7 \times 12$ and nonlinear ReLU transformation. Before being fed to the Softmax layer, a bias term is added to $H3$. Thus, the output of $H3$ is 30 numerical numbers plus a bias.

With back propagation through the whole architecture, a test error rate of 3% is achieved with single model. Since we want to investigate into the effect of ensemble, we dumb the CNN on purpose by fixing H1 and H2 and let learning occurs only in the fully-connected layer and the Softmax layer.

2.3 Training on Different Epochs

In our experiment, we train 30 CNNs independently and each CNN is trained from 1 to 20 epochs, designated as $\varphi_{1.1}$, $\varphi_{1.2}$,..., $\varphi_{1.20}$, $\varphi_{2.1}$, $\varphi_{2.2}$,..., $\varphi_{2.20}$, ..., $\varphi_{30.1}$, $\varphi_{30.2}$,..., $\varphi_{30.20}$. The testing error, the vertical axis of Figure 1, is plotted against the training epochs, the horizontal axis. Lower testing error indicates better performance. The red line in the middle is the averaged testing error of all CNN trained under a certain number of epochs. Figure 1 shows there is no obvious observation that increasing the training number of epochs would lead to better performance. Note that with different structure and dataset, this observation may vary.

2.4 Averaging Predictions

From the previous setting, we have trained 30 groups of CNNs designated as $\varphi_{1.1}$, $\varphi_{1.2}$,..., $\varphi_{1.20}$, $\varphi_{2.1}$, $\varphi_{2.2}$,..., $\varphi_{2.20}$, ..., $\varphi_{30.1}$, $\varphi_{30.2}$,..., $\varphi_{30.20}$. Since the output is numerical, an obvious procedure of ensemble learning is averaging the prediction over the predictors trained under the same number of epochs. Designating $\varphi_{\bar{N}.e}$ as the ensemble of Q predictors each trained with e epochs. In Figure 2, the testing error is plotted against Q, the number of CNNs averaged. Figure 2 shows that as Q increases, the testing error reduces, indicating that the ensemble via averaging the prediction contributes to achieving better performance. However, we notice that as more and more predictors being averaged, the rate of reducing testing rate goes down and eventually the lines go flat, thus it is unlikely that by averaging infinite number of CNNs, testing error can be reduced to zero.

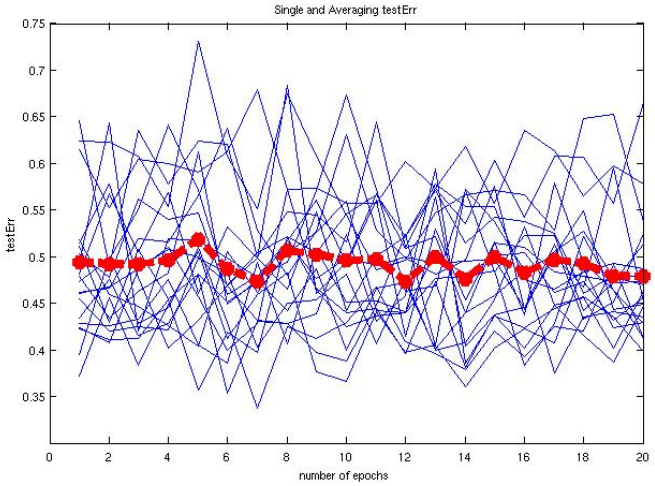

Figure 1: Testing Error of CNNs Trained with Different Epochs

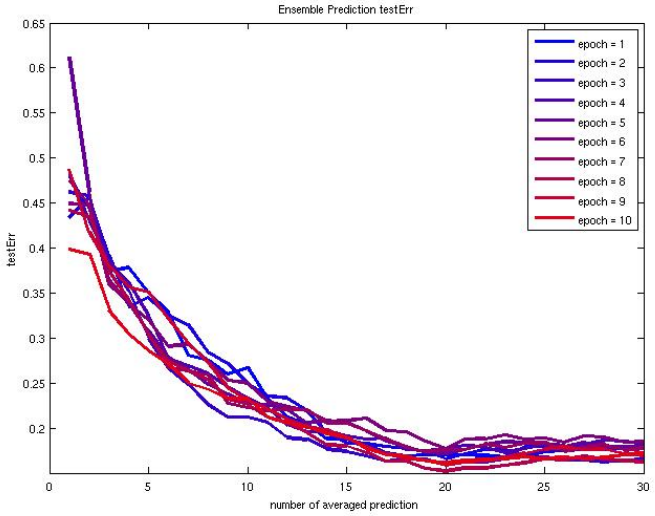

Figure 2: Testing Error of CNNs with Averaged Prediction

2.5 Fixing the Total Number of Training Epochs

Now that we are aware that for our architecture and dataset, increasing Q, the number of models averaged, helps obtain better performance, while

increasing e, the learning duration, does not. Since both increasing Q and e lead to higher costs, it is natural to think about the tradeoff between Q and e. If we fix $Q \times e$, then we fix the total cost of training. In Figure 3, testing error is plotted against different combinations of Q and e. The blue line shows that when $Q \times e = 30$, the testing error is reduced as Q increases. The red line is the test error of CNNs with $Q = 1$ and e equals that of the blue line. Use the red line as a control group, Figure 3 shows that the predictors gain more and more accuracy as the number of models involve in the ensemble increases. Since the models are trained independently, if we spread the training onto different machines, we can effectively reduce the training time.

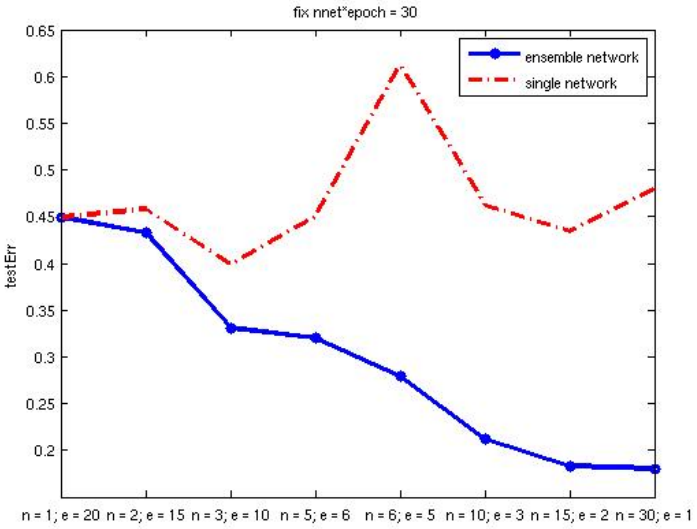

Figure 3: Testing Error of the ensemble of CNNs with Fixed Amount of Training Cost

2.6 Creating New Softmax Layer

Here we experiment a new way of ensemble instead of simply averaging the prediction numerically. As discussed in the previous section, the output of a CNN is a vector of length 10. With a fixed training duration, i.e., e is a constant, we collect the output of 30 independently-trained CNNs and stack them as a new feature map. This map is then fed to a new Softmax layer which outputs a vector of length 10. The elements of the new output still

represent the probability of the original input belongs to a certain class. In Figure 4, testing error is plotted against e ranged from 1 to 20. The blue line is the plot of testing error against epochs under the new way of ensemble, while the red line is testing error against epochs without ensemble. Use the red line as a control group, Figure 4 shows that with ensemble by stacking the outputs from independently-trained models and creating a new Softmax layer, the predictor performs better.

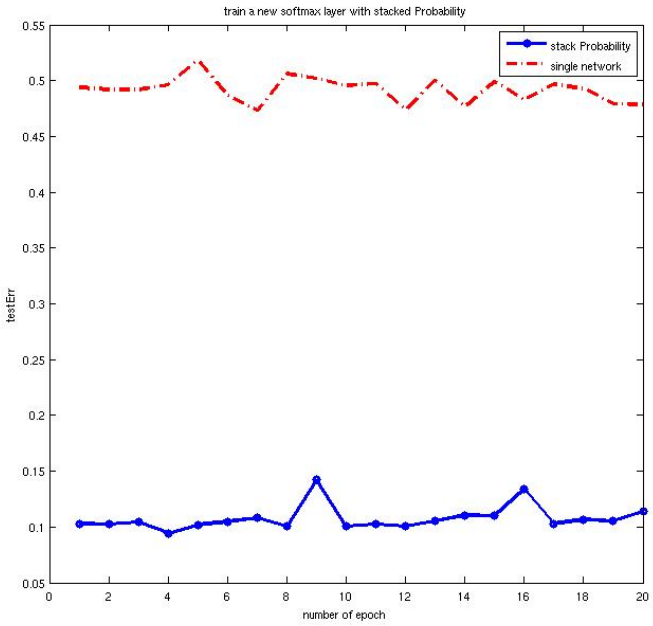

Figure 4: Testing Error of Ensemble by Creating New Softmax Layer

3 Why Ensemble Works

For a single network from the distribution $P(Net)$, the expected test error is

$$e = E_{\varphi} E_{x,y}(y - \varphi(x))^2 \tag{1}$$

The aggregate predictor is

$$\varphi_A(x) = E_{\varphi \sim P(Net)}(\varphi(x)) \tag{2}$$

The expected aggregate error is

$$e_A = E_{x,y}(y - \varphi_A(x))^2 \tag{3}$$

The empirical mean of the predictors is

$$\varphi_{\bar{Q}} = \frac{1}{Q}\sum_i \varphi_i(x) \tag{4}$$

The expected error of the mean of the predictor is

$$e_{\bar{Q}} = E_\varphi E_{x,y}(y - \frac{1}{Q}\sum_i \varphi_i(x))^2 \tag{5}$$

To prove $e \geq e_A$, we have

$$\begin{aligned} e &= E_\varphi E_{x,y}(y - \varphi(x))^2 = E_\varphi E_{x,y}(y^2 - 2y\varphi(x) + \varphi^2(x))^2 \\ &= E_{x,y}E_\varphi(y^2 - 2y\varphi(x) + \varphi^2(x))^2 = E_{x,y}(y^2 - E_\varphi(2y\varphi(x)) + E_\varphi(\varphi^2(x))) \\ &\geq E_{x,y}(y^2 - 2y\varphi_A(x) + E^2_\varphi(\varphi(x))) = E_{x,y}(y^2 - 2y\varphi_A(x) + \varphi^2_A(x)) \\ &= E_{x,y}(y - \varphi_A(x))^2 = e_A \end{aligned} \tag{6}$$

Thus we prove that the expected error from ensemble is always smaller than the expected error from a single predictor. So theoretically, we always gain from ensemble in expectation.

We can decompose (1) by bias and variance as following:

$$\begin{aligned} e &= E_\varphi E_{x,y}(y - \varphi(x))^2 \\ &= E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x) + E_{x,y}\varphi_A(x) - \varphi(x))^2 \\ &= E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x))^2 + 2E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x))(E_{x,y}\varphi_A(x) - \varphi(x)) \\ &+ E_\varphi E_{x,y}(\varphi(x) - E_{x,y}\varphi_A(x))^2 \\ &= E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x))^2 + E_\varphi E_{x,y}(\varphi(x) - E_{x,y}\varphi_A(x))^2 \\ &= E_{x,y}(y - E_{x,y}\varphi_A(x))^2 + E_\varphi E_{x,y}(\varphi(x) - E_{x,y}\varphi_A(x))^2 \end{aligned} \tag{7}$$

Similarly, (3) can be decomposed as following:

$$\begin{aligned} e_A &= E_{x,y}(y - \varphi_A(x))^2 \\ &= E_{x,y}(y - E_{x,y}\varphi_A(x) + E_{x,y}\varphi_A(x) - \varphi_A(x))^2 \\ &= E_{x,y}(y - E_{x,y}\varphi_A(x))^2 + E_{x,y}(\varphi_A(x) - E_{x,y}\varphi_A(x))^2 \end{aligned} \tag{8}$$

Thus,

$$\begin{aligned} e - e_A &= E_\varphi E_{x,y}(\varphi(x) - E_{x,y}\varphi_A(x))^2 - E_{x,y}(\varphi_A(x) - E_{x,y}\varphi_A(x))^2 \\ &= E_{x,y}(E_\varphi(\varphi^2(x)) - E_\varphi^2(\varphi(x))) = E_{x,y}(Var_\varphi(\varphi(x))) \geq 0 \end{aligned} \tag{9}$$

However, in our experiment, we can only observe the empirical mean,(4), instead of the expectation,(2). To obtain the ensemble learning effect, we decompose

$$\begin{aligned} e_{\bar{Q}} &= E_\varphi E_{x,y}(y - \frac{1}{Q}\sum_i f_i(x))^2 \\ &= E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x) + E_{x,y}\varphi_A(x) - \frac{1}{Q}\sum_i f_i(x))^2 \\ &= E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x))^2 + 2E_\varphi E_{x,y}(y - E_{x,y}\varphi_A(x))(E_{x,y}\varphi_A(x) - \frac{1}{Q}\sum_i f_i(x)) \\ &+ E_\varphi E_{x,y}(\frac{1}{Q}\sum_i f_i(x) - E_{x,y}\varphi_A(x))^2 \\ &= E_{x,y}(y - E_{x,y}\varphi_A(x))^2 + E_\varphi E_{x,y}(\frac{1}{Q}\sum_i f_i(x) - E_{x,y}\varphi_A(x))^2 \end{aligned} \tag{10}$$

In (10), the second term can be decomposed as

$$\begin{aligned} &E_\varphi E_{x,y}(\frac{1}{Q}\sum_i f_i(x) - E_{x,y}\varphi_A(x))^2 \\ &= E_{x,y}E_\varphi(E_{x,y}^2\varphi_A(x) - 2E_{x,y}\varphi_A(x)\frac{1}{Q}\sum_i f_i(x) + (\frac{1}{Q}\sum_i f_i(x))^2) \\ &= E_{x,y}E_\varphi(E_{x,y}^2\varphi_A(x) - 2E_{x,y}\varphi_A(x)\varphi_A(x) + (\frac{1}{Q}\sum_i f_i(x))^2) \end{aligned} \tag{11}$$

$$\begin{aligned} &E_{x,y}E_\varphi(\frac{1}{Q}\sum_i f_i(x))^2 = E_{x,y}E_\varphi(\frac{1}{Q^2}(Q\sum_i f_i^2(x) + \sum_{i,j}\varphi_i(x)\varphi_j(j))) \\ &= \frac{1}{Q^2}(QE_\varphi(\varphi^2(x)) + \sum_{i,j}E_\varphi(\varphi_i(x)\varphi_j(j))) \end{aligned} \tag{12}$$

Thus,

$$
\begin{aligned}
e - e_{\bar{Q}} &= E_{x,y}(E_{\varphi}(\varphi^2(x)) - \frac{QE_{\varphi}(\varphi^2(x)) + \sum_{i,j} E_{\varphi}(\varphi_i(x)\varphi_j(j))}{Q}) \\
&= E_{x,y}(\frac{Q-1}{Q}(E_{\varphi}(\varphi^2(x)) - E^2_{\varphi}(\varphi(x)))) \\
&= \frac{Q-1}{Q} E_{x,y}(Var_{\varphi}(\varphi(x))) \geq 0 \qquad (13)
\end{aligned}
$$

From (13) we know that the more predictors involve in ensemble, the less of the error. However, as Q goes to infinity, the marginal amount of error we reduce decrease to 0. This agrees with the behavior of the error we observed from Figure 2.

4 Conclusion

Ensemble is a powerful procedure which improves single network performance. It reduces the variance portion in the bias-variance decomposition of the prediction error. Our project has experimented with different ensemble methods that all tend to contribute to dramatic error reduction. In addition, the tradeoff between number of models and their complexity has been investigated and we show that ensemble learning may lead to accuracy gains along with reduction in training time.

5 References

Breiman, L., Bagging Predictors, Machine Learning, 24(2):123−140, 1996

Krizhevsky, A., Sutskever, I., and Hinton, G.E. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. Neural Comput., 1(4):541−551, 1989.

Zeiler, M., Fergus, R., Visualizing and Understanding Convolutional Networks, ECCV 2014 , Arxiv 1311.2901, 2013.