Influence of Training Set Size on Generalization of Artificial Neural Network

Hoang-Trieu TRINH
Advanced Program in Computer Science
Faculty of Information Technology
Ho Chi Minh University of Science
thtrieu@apcs.vn

Huu-Phat HO
Advanced Program in Computer Science
Faculty of Information Technology
Ho Chi Minh University of Science
hhphat@apcs.vn

Nhat-Thanh KIM
Advanced Program in Computer Science
Faculty of Information Technology
Ho Chi Minh University of Science
knthanh@apcs.vn

***Abstract*—Artificial Neural Networks have long been considered to be capable of solving a wide range of difficult problems. In addition, the available training data is also growing exponentially everyday thanks to a more open and globally connected information network. According to the conventional belief that more data should always refine the model's estimators, it is reasonable for one to be optimistic about future capability of neural networks. To verify this, the authors investigate the performance of different neural network architectures on MNIST database's test set (10k) after training them on different portions of the full training set (60k). The purpose of this experiment is to find out whether using the full training set is best in terms of the performance on test set. Result shows that for some architectures of neural network, the answer is negative. This casts doubt on situations where other classifiers are applied without any consideration of a partial usage from the corresponding training set. The result also implies that a new approach should be proposed while assessing the relative performance of different classifiers after being trained on the same data.**

Keywords—artificial neural network, MNIST, accuracy, generalization.

I. Introduction

Inspired by biological neural networks, artificial neural networks are a group of probabilistic and statistical models designed to learn from a large amount of input in order to approximate unknown functions. Thanks to their flexible nature as universal approximators, neural networks have been able to solve a wide range of problems that are previously considered to be difficult in machine learning, computer vision, and speech recognition [1], [2]. After a time being outran by new methods such as Support Vector Machines and Linear models, Neural Networks regain their popularity in early 2000s under the new name Deep Learning, thanks to new in-depth development on their architecture and advances in modern computer's computational capabilities [3], [4].

Mutually assistant to this growth is the astronomical amount of data that is still growing ever since the establishment of Internet and large data centers [5]. Specifically, more data gives a richer source of training material, and more refined models are more accurate in making sense of new data. An epitome of the successful application of neural networks is the MNIST (Mixed National Institute of Standards and Technology) database of handwritten digits, which consists of 60,000 training images and 10,000 testing images, each is a 28x28 pixel gray scale image [6]. The MNIST database is presented in a way that minimizes user's effort on preprocessing and formatting, therefore it is commonly used for practicing various machine learning and image processing techniques [7], [8].

A number of scientific papers on different algorithms have been introduced in attempt to lower the state-of-the-art error rate on MNIST test set [9], [10]. Some of these include additional preprocessing steps applied to the training set such as deskewing, noise removal, blurring, subsampling, or augmentation with distorted versions of the original training examples [11]. In 2012, a group of researchers built a committee of 35 Convolutional Neural Networks, trained on the width-normalized MNIST training set, achieved an accuracy of 0.23 percent [12].

All of the existing classifiers implemented on MNIST dataset are trained on the complete training set, which consists of 60,000 handwritten digit patterns, or its augmented version [11]. However, it is reported that more training data does not guarantee a better estimation of the model's parameters in terms of their mean squared error [13], [14]. Practically speaking, this can be due to the fact that the training data is biased, or the given data distribution disagrees in relevant ways with some of the model assumptions [14]. This experiment examines the validity of training a classifier on the complete MNIST training set and evaluate its performance on the complete MNIST test set.

To address this, the authors tested the performance of a number of neural network's architectures on the complete MNIST test set after training them on different subsets of the MNIST training set. The training subsets are arranged in an order such that the smaller one is a subset of the bigger one. This is to expose the changes in a particular algorithm's accuracy when adding more data to the resource it has already seen. Different neural network architectures with various capacities when tested on MNIST test set also shows the effect of adding more data into different models.

Result shows that the dominant trend when adding more data is an improvement of accuracy on the MNIST test set. For some sizes of hidden layer, however, 60,000 training examples does not give the best performance on MNIST test set. Interestingly, the worsening effect of adding more data on neural network's accuracy can be seen more clearly when these networks are trained on two subsets of size 30,000

and 40,000. At these points, there is a considerable number of neural networks that appear to have failed to improve their performance on MNIST test set. In fact, their accuracy significantly dropped off the increasing tendency previously made by smaller training subsets with sizes 10,000 and 20,000.

Section II introduces background and related works, Section III shows the details of the experiment, Section IV reports our experiment's result and finally, Section V sums up the authors' investigation and gives our final conclusion.

II. Background and Related Work

A. Main Concepts

a) Perceptron: First Artificial Neural Network invented in 1957 by F. Rosenblatt is Perceptron. Perceptron is a type of linear classifier that establishes its predictions based on a linear predictor function, which is a set of coefficients and explanatory variables, combining a set of weights with the feature vector to make a prediction [15]. The model of Perceptron classifier:

$$f(x) = sign(w^T x + b)$$

The update rule:

$$w_j = w_j + \alpha \left(f_i - y_i\right) x_{ij}$$

Unfortunately, Perceptron could not be trained to recognize many classes of patterns. Furthermore, Perceptron learning algorithm does not terminate if the learning set is not linearly separable.

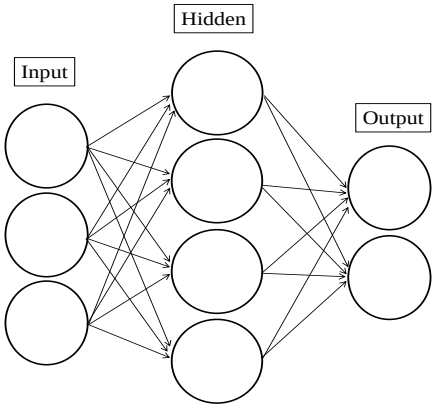


Fig. 1: Multilayer Perceptron

b) Multilayer Perceptron: Because of the above limitations of Perceptron, researchers in this field came up with Multilayer Perceptron (MLP). As illustrated in Fig. 1, MLP is a feed forward neural network that consists of three or more layers of nodes and each layer is fully connected to the next one [16]. If MLP has linear activate function, it can be proved that any number of layers can be reduced to the standard two-layer input-output model (Perceptron). Except for the activate function of output units, MLP uses sigmoidal/rectifier function instead of heavy-side as in Perceptron and is trained using back propagation [17]. MLP is capable of both classification and regression. It also can distinguish non-linear separable data.

c) Artificial Neural Networks: Examinations of the human's brain system inspired the concept of Artificial Neural Networks (ANN). In ANN, simple artificial nodes, known as neurons, are connected together to construct a network. In detail, the structure of basic neural network includes many layers: input layer, hidden layer, output layer. Layers includes sets of interconnected nodes. Patterns are shifted through the network starting from the input layer. Specifically, the input layer communicates this information to hidden layers where the data is processed by a system of weighted connections. The hidden layers then transmit this processed information to the output layer [18]. Due to the fact that ANNs have a powerful pattern classification and pattern recognition capability, the main function of this network is to forecast and estimate values from inputs by approximating functions [19]. Application-wise, neural networks help improve solutions in hand writing recognition, which is known as the ability of a computer or a machine to receive and interpret apprehensible handwritten input from outside sources [20].

d) Back-Propagation Neural Networks: Although there are many different categories of learning rules for training Artificial Neural Networks on supervised-learning tasks, the authors concern only with traditional gradient-based methods [21]. These methods are made possible in the case of Neural Network thanks to the introduction of back-propagation (BP), an abbreviation for "backward propagation of errors". BP is an algorithm that helps compute the loss function's gradient with respect to each weight in the network. These gradients is evaluated recursively, starting from the gradient of output units and then propagate this information backward by chain rule. Suppose the forward flow of information is:

$$z_j = g\left(a_j\right)$$
$$a_k = \sum_j w_{kj} z_j$$

Where $g()$ is the activation function and is required to be differentiable, w_{kj} is the weight of the connection that sends information from unit j to unit k. After that, the propagation process happens as follows:

$$\Delta_j = g'(a_j) \sum_k w_{kj} \Delta_k$$

Where Δ_i is the gradient with respect to unit i. Unfortunately, BP have some drawbacks. After defining the general construction of a network and initially seeding the network with some random numbers, there is no other complicated work to do except feeding the network input, watching it trains, and waiting for the output. Another disadvantage of BPs is that it is likely to be slower to train than the others. Therefore, many techniques have been presented to deal with these problems [22]–[24].

e) Nonlinear conjugate gradient method using Polack-Ribiere formula: Nonlinear conjugate gradient is a method to generalize the conjugate gradient method to nonlinear optimization. They use relatively little memory for large-scale problems and require no numerical linear algebra, but they typically converge much more slowly than Newton or quasi Newton methods [25]. The authors use Polak-Ribiere function to compute search directions and perform a line search using quadratic and cubic polynomial approximations [26].

The Wolfe-Powells stopping-criteria is used together with the slope ratio method for guessing initial step sizes [27].

f) MNIST Database: MNIST is one of most popular examples of the successful application of ANN. The MNIST database is a remixed subset of the NIST database, specifically a half of MNIST training set and a half of MNIST test set are taken from NIST training set, the other halves are taken from NIST test set [9]. This mixing is motivated by the fact that NIST training set and test set are taken from two different sources: The American Census Bureau employees and American high school students [10]. Among different classifiers that are proposed by the MNIST database's original creators, a Support Vector Machine with polynomial kernel of degree 9 achieves an error rate of 0.8 percent, the only better one is a Convolutional Neural Network, which achieves an error rate of 0.7 percent [11].

g) The Bias-Variance Trade-Off: In supervised learning, when the model is being too simple and it could not fit the data and predict new data points accurately. The simpler model is called to have a high bias. On the other hand, when the model has too many parameters and fits almost all the data points but has poor prediction of new data points. The model is said to have high variance. The bias-variance trade-off is an acknowledged problem in machine learning and it usually occurs when we have relatively few data points or model becomes excessively complex [28], [29]. A learning algorithm that reduces the chance of fitting noise and has good generalization is called robust.

B. Related Work

Herman Chernoff investigates a two sample test originated from a satellite based experiment, which is designed to measure the Doppler effect in the relative motion of two astronomical objects. The experiment counts the number of photon emitted from both objects within a narrow energy band, during specified time interval [14]. The test consists of data from Poisson counts and raises some puzzling questions due to the fact that its designer allocated relatively short time interval for each counts, hence there are some energy band with very small or zero counts. Herman Chernoff addresses the question of why ignoring data from these short time, low count intervals is desirable [14].

Xie and Meng observe samples generated from the heteroscedastic regression model $Y_t = \beta X_t + \epsilon_t$ (where ϵ_t $N(0, {X_t}^2)$) independently and $X_t = (101 - t)^{-1}$ to conclude that the ordinary least squares estimator of β, despite being consistent, experiences an inflating variance when adding more data to the estimation procedure [13], this worsen β in terms of its Mean Squared Error. The same effect happens to the simplest ARCH regression model with a single predictor ($Y_t|F_{t-1}$ $N(X_t\beta, {\tau_t}^2)$, where $t = 1, \ldots, T$ and F_{t-1} is the σ-field generated by $\{Y_1, \ldots, Y_{t-1}\}$) [13]. Meng proposes the concept of a statistical estimation procedure being self-efficient [30]. Intuitively, a self-efficient estimator cannot be improved by linearly combining itself with its bootstrapped version [13]. Xie and Meng's work also conclude that additional data are not always helpful unless probabilistically principled methods such as Maximum Likelihood Estimation and Bayesian approaches are used [13].

III. Experiment

A. The Classifier

Since a one-hidden layer neural network is proven to have the universal approximation property [31], the authors restrict themselves only to single hidden layer and fully-connected sigmoidal neural networks. Each with one input layer of size 28 x 28 = 784, one output layer of size 10 corresponds to 10 number digits.

Specifically, there are 24 neural networks, which have respectively 20-, 40-, 60-, ..., 480-neuron in hidden layer. This range of hidden layer size is a natural choice in terms of completeness. It ensures the anticipated transition of accuracy on MNIST test set when each neural network goes from underfitting to overfitting its corresponding training set can be clearly observed. By this, the authors also examine how the influence of different training set sizes on neural network's performance differs across the 24 architectures.

B. The database

Neural networks are reportedly the best model to perform on MNIST database. This database's cleanliness in terms of handling and the abundant existing works are also among the authors' motivations of choosing this dataset.

C. Training Sets

The authors generate six training subsets from the complete MNIST training set using the following procedure:

1) Randomly sample 10,000 images from the complete MNIST training set to form the smallest training subset.
2) Exclude the previously sampled 10,000 training examples from the current MNIST training set.
3) Randomly sample another 10,000 images from the current MNIST training set and add them to the most recently created subset to form the next bigger one.
4) Repeat the procedure from its second step until the sixth subset is created, this last training set is also the complete MNIST training set.

By including the full set of 60,000 training images into this experiment, the authors test the conventional anticipation that each of the classifiers would perform best when they are trained on this full dataset.

D. Training Procedure

24 neural networks are trained on the above 6 training sets with squared loss criterion and back-propagation in full-batch setting. Since this dataset is relatively small, the update rule is the nonlinear conjugate gradient rule with Polak-Ribiere version [26]:

1) Calculate the steepest direction:
$$\Delta x_n = -\nabla_x f(x_n)$$
2) Compute β_n according to Polak-Ribiere formula [32].
3) Update the conjugate direction:
$$s_n = \Delta x_n + \beta_n s_{n-1}$$

4) Perform a line search:

$$\alpha_n = \arg\min_{\alpha} f(x_n + \alpha s_n)$$

5) Update the position:

$$x_{n+1} = x_n + \alpha_n s_n$$

The script used is $fmincg$, developed and freely provided by Carl Edward Rasmussen.

The quantities that this experiment aims at is the generalization rate $G(\theta_{ij})$ of these 24 neural networks after being trained on the six training subset:

$$G_\chi(\theta_{ij}) = \int_\chi L(f(\theta_{ij}, x), C(x)) P_\chi(x) dx$$

where θ_{ij} is the weight vector of neural network i after being trained on training subset j, $f(\theta_{ij}, \cdot)$ is the function represented by neural network with weight θ_{ij}, $C(x)$ is the true class identity of x, and $P_\chi(\cdot)$ is the probability density function of input population χ. Since the population is unavailable, G_χ is approximated unbiasedly by replacing the population χ by the test set $\chi test$, P_χ is replaced by the empirical density function F, which puts equal mass on every point in its domain.

$$\widehat{G}_{\chi test}(\theta_{ij}) = \int_{\chi test} L(f(\theta_{ij}, x), C(x)) F(x) dx$$

Thus, the quantities that we are to evaluate is $\widehat{G}_{\chi test}(\widehat{\theta_{ij}})$ where $\widehat{\theta_{ij}}$ is obtained by train on $\chi train$:

$$\widehat{\theta_{ij}} = \arg\max_{\theta_{ij}} \widehat{G}_{\chi train}(\theta_{ij})$$

Due to the non-linear nature of the cost function, it is impossible for heuristic algorithms such as gradient based methods to guarantee convergence at the global minimum $\widehat{\theta_{ij}}$. For this reason, $\widehat{\theta_{ij}}$ is in turn approximated by $\widehat{\theta^*_{ij}}$:

$$\widehat{\theta^*_{ij}} = \underset{\widehat{\theta^k_{ij}}, k=1\ldots K}{\arg\min} \hat{G}_{\chi train}(\hat{\theta}^k_{ij})$$

where

$$\widehat{\theta}^k_{ij} = fmincg(\theta^k_{ij})$$

Specifically, each network is trained K times (the authors opt for K=3), starting with K different configurations for its initial set of weights to avoid early convergence due to shallow local minima. The initial weight vectors θ^k_{ij} are generated by uniformly sampling from $[0,1]^\Theta$ (Θ is the dimension of the weight vector). Each training stops when no more progress can be made by the algorithm due to convergence at a minimum, or numerically so close to a minimum to the extent that the current weight vector is technically indistinguishable to the optimal one. Among the three trained weight settings for each of 24 architectures, the one with best accuracy on MNIST training set is chosen to represent.

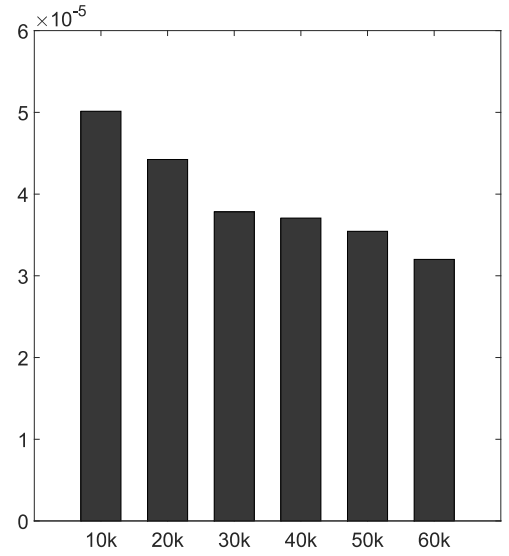


Fig. 2: Class Proportion Variance vs. Training Set

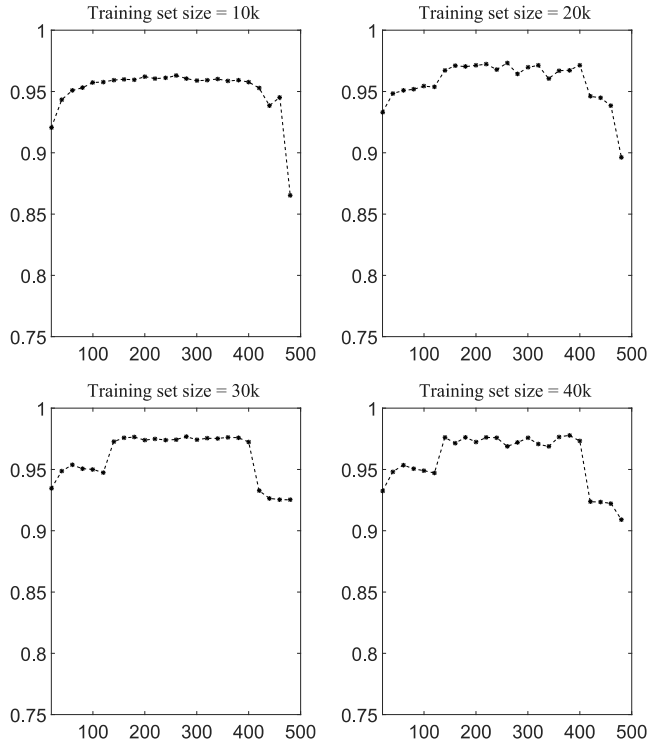


Fig. 3a: Accuracy vs. Hidden layer size

IV. Result

Table I shows the proportion of the ten classes present in six training sets, which are generated by the procedure introduced in section III. It is clear that the proportions of these ten classes are relatively even by the random sampling procedure. It can be seen from the table that each of them deviates not too much from 0.1. This suggests an unbiasness

Class labels

Training Subsets

	0	1	2	3	4	5	6	7	8	9
1	0.10010	0.11270	0.09910	0.10320	0.09800	0.08630	0.10140	0.10700	0.09440	0.09780
2	0.09970	0.11405	0.09645	0.10380	0.09725	0.08875	0.09855	0.10465	0.09610	0.10070
3	0.09870	0.11410	0.09827	0.10243	0.09753	0.09030	0.09917	0.10357	0.09583	0.10010
4	0.09810	0.11407	0.09858	0.10203	0.09773	0.09010	0.09938	0.10313	0.09650	0.10040
5	0.09864	0.11356	0.09936	0.10202	0.09718	0.09012	0.09902	0.10350	0.09684	0.09976
6	0.09872	0.11237	0.09930	0.10218	0.09737	0.09035	0.09863	0.10442	0.09752	0.09915

TABLE I: Class Proportion vs. Training Set

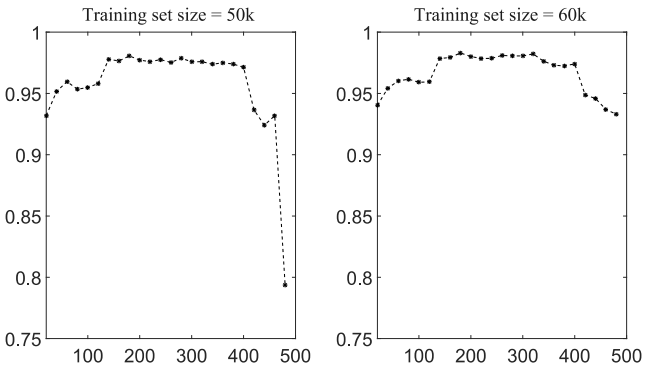
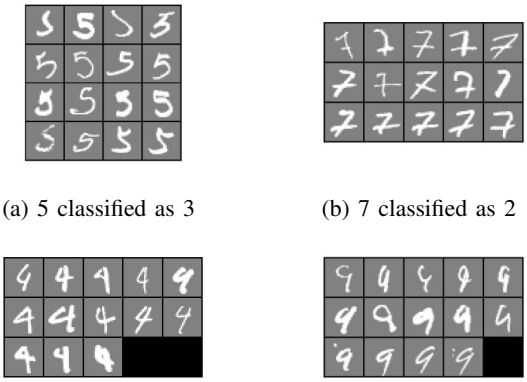


Fig. 3b: Accuracy vs. Hidden layer size



(a) 5 classified as 3

(b) 7 classified as 2

(c) 4 classified as 9

(d) 9 classified as 4

Fig. 5: Most common confusions

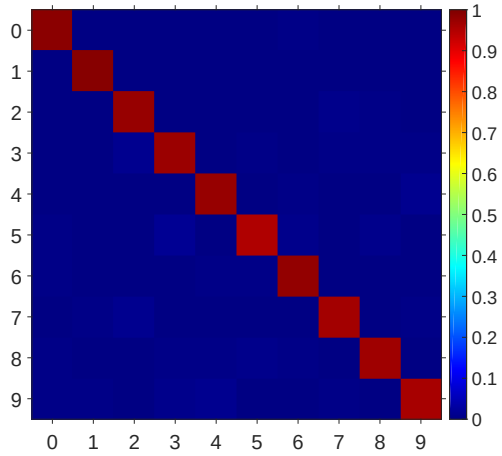


Fig. 4: Confusion matrix

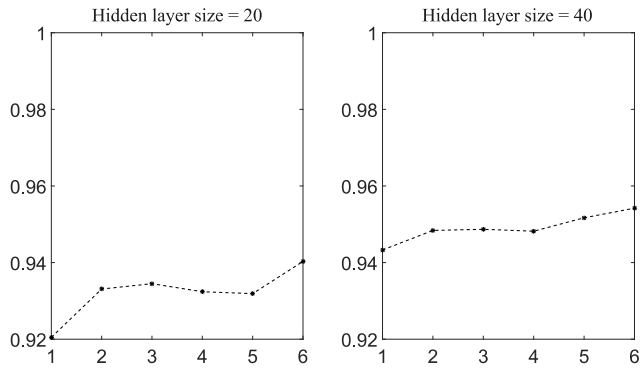


Fig. 6a: Accuracy vs. Training set size

presents in these six training subsets. Regarding this situation, one normally expects that all of the trained classifiers should perform on the MNIST test set approximately as well as it did on the training subset.

On a closer look, Table I also suggests that as the training set size increase, the unevenness between classes sizes significantly decreases, which means the dataset is more and more balanced. Thus, a classifier trained on a bigger subset should be more impartial over the ten classes, not to mention a richer source of training material it would benefit from this bigger subset. All of this is illustrated in Fig. 2, where Table I is summarized by its decreasing row-wise variances. In short, the general expectation that the biggest training set produces

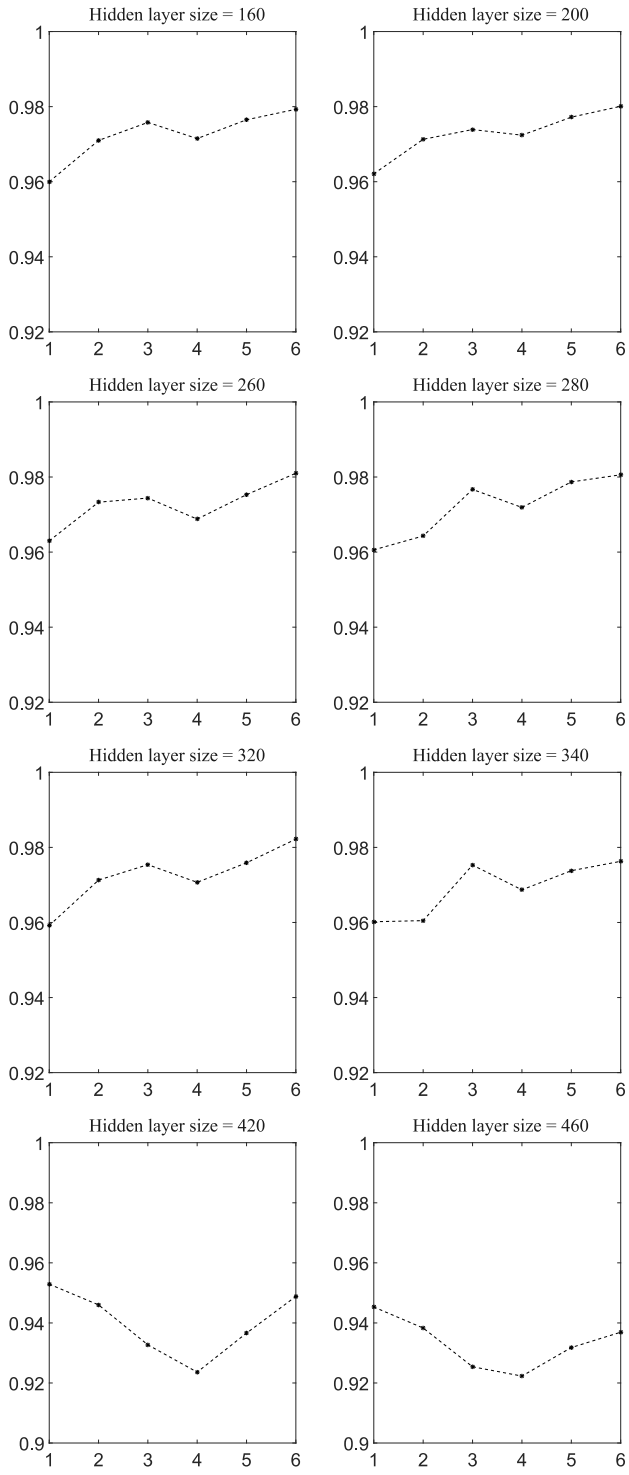


Fig. 6b: Accuracy vs. Training set size

the best generalization error rate on MNIST test set is backed up by reasonably bold evidences.

Other expected results are confirmed in Figs. 3a and 3b. Small sized hidden layers are too simple to fully capture the underlying patterns in both training set and test set. Neural networks with too much parameters, on the other hand, fit

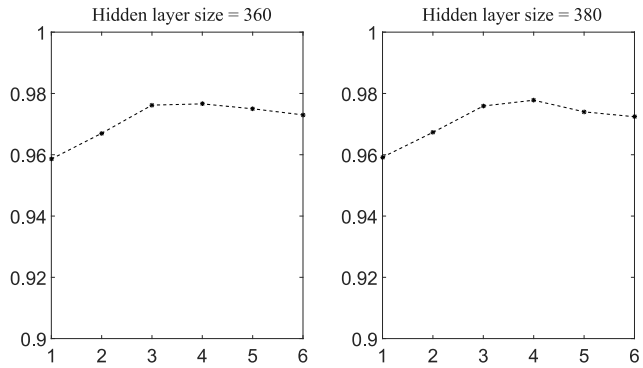


Fig. 7: Neural networks that do not perform best on test set after full-set training

even noise in the data, which is harmful to their generalization. Neural networks with intermediate hidden layer size strike the balance of these two extremes. The best neural network have 180 hidden neurons trained on 60.000 images. Fig. 4 is the confusion matrix of this case and Fig. 5 shows its most common confusions.

Although the fact that 60.000 images yield this best performance may imply more training data gives better generalization, counter-examples of this assumption present more frequently in other cases. In fact, Figs. 6a and 6b shows the prediction that adding more data is not always better has successfully applied to data from the MNIST dataset. Specifically in Fig. 7, for neural networks with hidden layer sizes of 360 and 380, the full training set of 60,000 training example does not yield better result comparing to smaller training subsets. This strange effect while adding more data is clearly illustrated by the universal concave shape presented throughout all of the 10 line graphs in Figs. 6a and 6b around the third, forth, and fifth training subsets.

While this may seem counter-intuitive, related analyzed explanations on this phenomenon have been proposed [9], [10]. Accordingly, the third, forth and fifth training subsets might disagree with some assumptions that most of the fully-connected neural networks have made, and since the 24 neural networks are trained based on the minimization of an empirical cost function instead of using probabilistic principled methods, i.e. Maximum Likelihood Estimation or Bayesian Inference, the training procedure may be not self-sufficient [9].

Unlike ARCH regression models, a mathematically rigorous analysis for this experiment remains unelaborated due to the complex nature of the mapping function represented by multilayer neural networks. The authors cease to further this point since this is beyond the purpose of our investigation.

V. Conclusion

As shown in section IV, the assumption "more data yields better accuracy" is not always true, at least on MNIST dataset. In practice, it is almost always the case that the underlying distribution is unknown and thus can not be analyzed to decide the cut-off point on the training dataset. This raises problems when one tries to reach a better generalization, since it might

not always be clear that in order to do so, which part of the whole training set should be included and which part to exclude.

Problems also emerge in cases when one attempt to compare the accuracy between different models after training them on the same dataset. For example, consider the case when a mathematician are choosing between two models A and B to interpret the data that he/she observed. Suppose the accuracy on test set is represented in the Table II:

TABLE II: Performance on test set of two different models

	Training Set 1	Training Set 2
Model A	A1	A2
Model B	B1	B2

In case Training Set 1 has a bigger size than Training Set 2, it is expected that $A1 > A2$ and $B1 > B2$, hence the decision depends on the comparison between $A1$ and $B1$. However, it has been shown that there might be unexpected case such as $A1 > A2$ but $B1 < B2$. If this happened, it would require further analysis on whether he/she should decide by comparing $A1$ with $B1$ or $A1$ with $B2$.

REFERENCES

[1] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[2] "Kurzweil AI interview with JÃijrgen Schmidhuber on the eight competitions won by his Deep Learning team," http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions, 2009–2012, accessed: 08-05-2015.

[3] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 545–552.

[4] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[5] "Data, data everywhere," http://www.economist.com/node/15557443, 2010, accessed 08-05-2015.

[6] E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition," *Image and Vision Computing 22*, 2004.

[7] C. Cortes and V. Vapnik, "Support vector machines speed pattern recognition," *Machine Learning*, 1995.

[8] J. C. Platt, "Using analytic qp and sparseness to speed training of support vector machines," in *IN NEURAL INFORMATION PROCESSING SYSTEMS 11*. MIT Press, 1999, pp. 557–563.

[9] B. Zhang and S. N. Srihari, "Fast k -nearest neighbor classification using cluster-based trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[10] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," http://yann.lecun.com/exdb/mnist/, accessed: 08-05-2015.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[12] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ser. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 3642–3649.

[13] M. Xiao-Li and X. Xie, "I got more data, my model is more refined, but my estimator is getting worse! am i just dumb?." 2014.

[14] H. Chernoff, *When it Seems Desirable to Ignore Data.*, 1982.

[15] F. Rosenblatt, "The perceptron–a perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Tech. Rep. 85-460-1, 1957.

[16] I. Aizenberg and C. Moraga, "Multilayer feedforward neural network based on multi-valued neurons (mlmvn) and a backpropagation learning algorithm," *SOFT COMPUTING*, vol. 11, no. 2, p. 2007, 2007.

[17] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[18] M. H. Hassoun, *Fundamentals of artificial neural networks*. MIT Press, 1995.

[19] G. Zhang, B. Eddy Patuwo, and H. Y. Micheal, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, Mar. 1998.

[20] J. A. Pittman and M. Manu, "Handwriting recognition using neural networks," Sep 23 2008.

[21] M. Buscema, "Back propagation neural networks," *Substance use & misuse*, vol. 33, no. 2, pp. 233–270, 1998.

[22] J. Leonard and M. Kramer", "Improvement of the backpropagation algorithm for training neural networks," *Computers I& Chemical Engineering*, vol. 14, no. 3, pp. 337 – 341, 1990.

[23] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, "Improving the convergence of the backpropagation algorithm using learning rate adaptation methods," *Neural Comput.*, vol. 11, no. 7, pp. 1769–1796, Oct. 1999.

[24] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, 1993, pp. 586–591.

[25] J. E. Dennis, Jr and J. J. Moré, "Quasi-newton methods, motivation and theory," *SIAM review*, vol. 19, no. 1, pp. 46–89, 1977.

[26] L. Grippo and S. Lucidi, "A globally convergent version of the polak-ribiere conjugate gradient method," *Mathematical Programming*, vol. 78, no. 3, pp. 375–391, 1997.

[27] Z. Wei, L. Qi, and S. Ito, "New step-size rules for optimization problems," *Department of Mathematics and Information Science, Guangxi University, Nanning, Guangxi, People's Republic of China*, 2000.

[28] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, Jan. 1992.

[29] "Bias variance decomposition." in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer, 2010, pp. 100–101.

[30] X. L. Meng, "Multiple-imputation inferences with uncongenial sources of input," *Statistical Science*, vol. 9, pp. 538–558, 1994.

[31] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, Oct. 1990.

[32] E. Polak and R. G, "Note sur la convergence de methodes de directions conjuguees," *ESAIM: Mathematical Modelling and Numerical Analysis - Modelisation Mathematique et Analyse Numerique*, vol. 3, no. R1, pp. 35–43, 1969.

APPENDIX

Figs. 8a to 8c shows the performance on MNIST test set of 24 neural networks after training them on 6 training subsets.

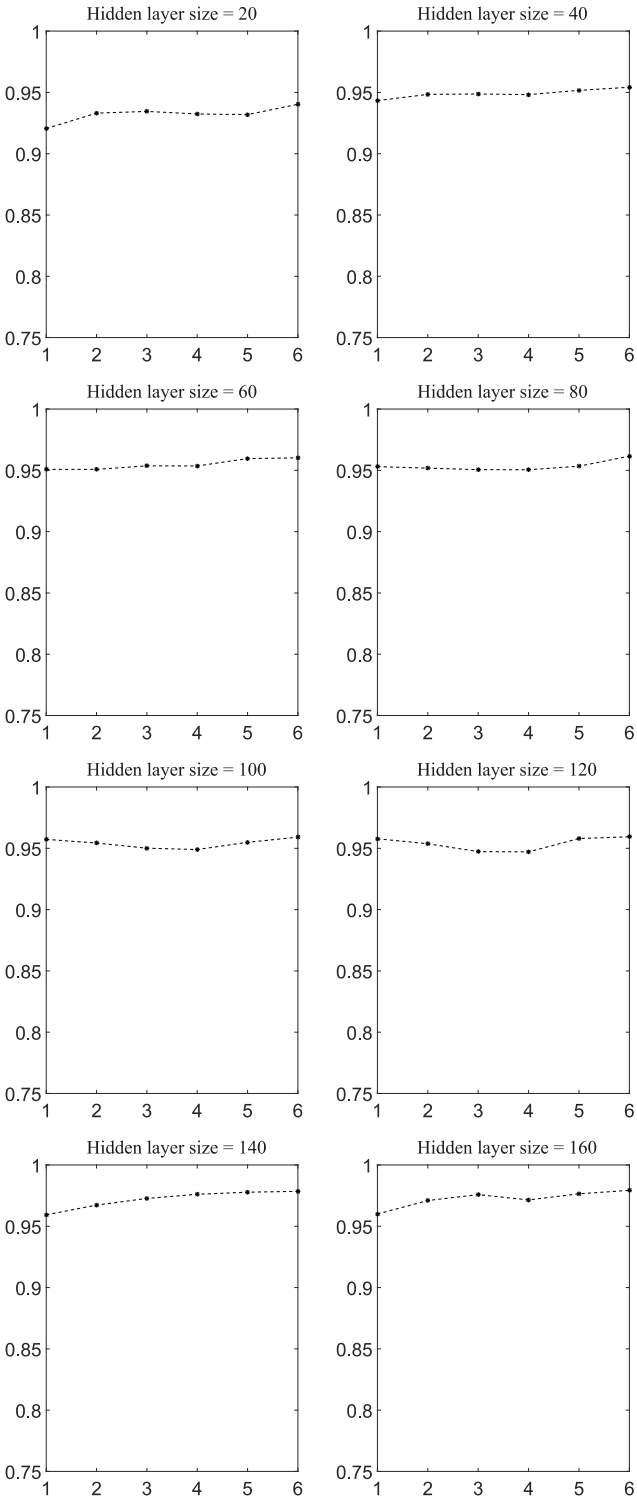


Fig. 8a: Accuracy vs. Training set size

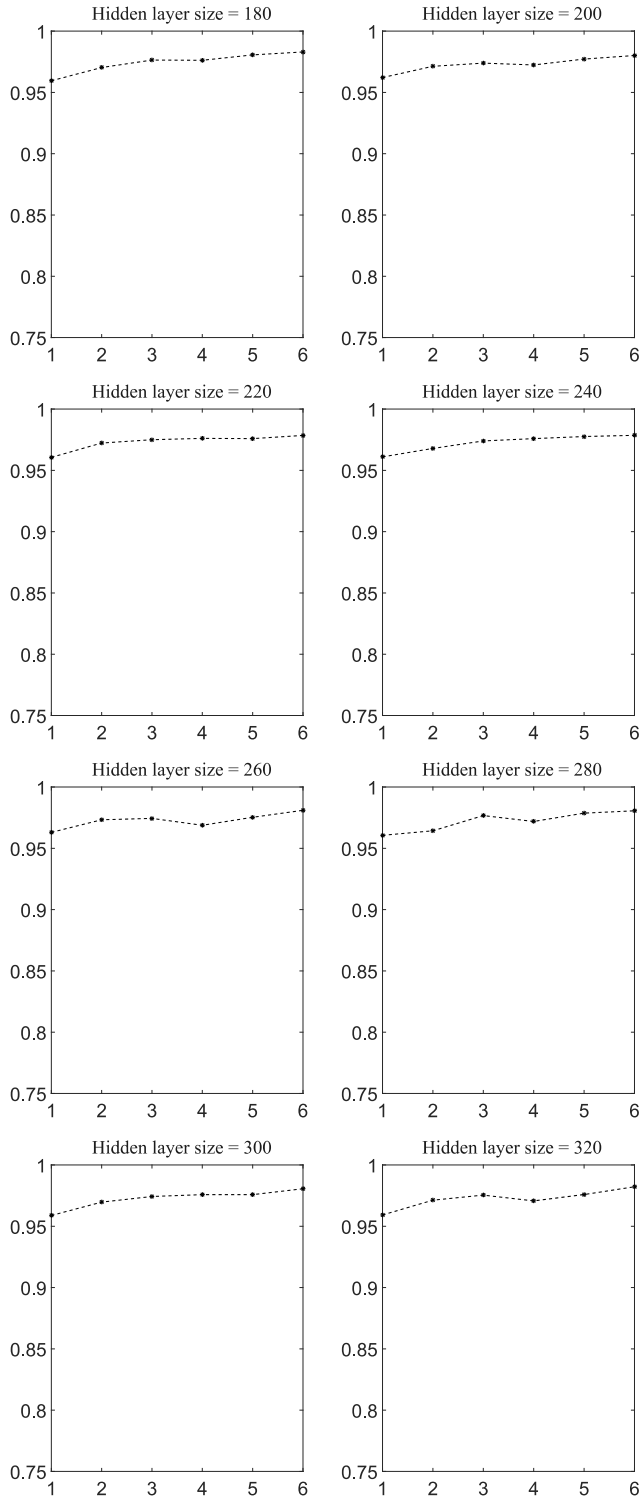


Fig. 8b: Accuracy vs. Training set size

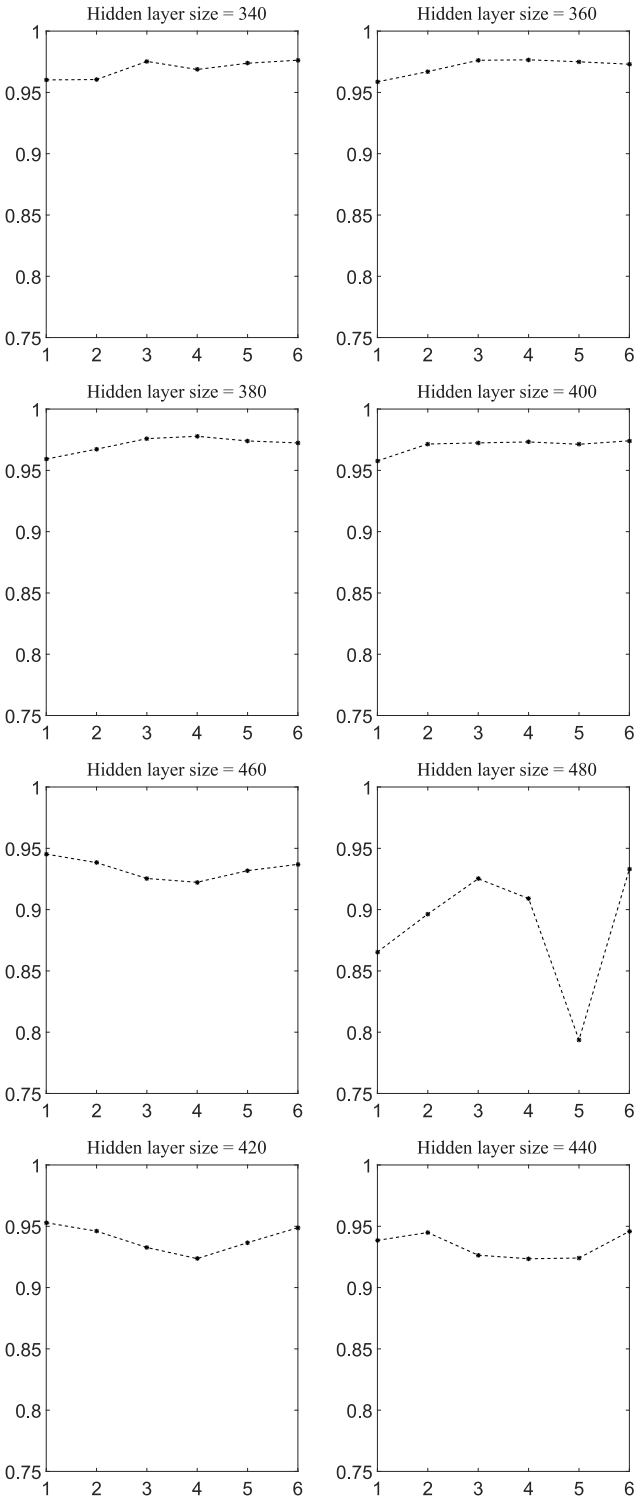


Fig. 8c: Accuracy vs. Training set size