Artificial Neural Network: Deep or Broad? An Empirical Study

Nian Liu[1] and Nayyar A. Zaidi[1]

Clayton School of Information Technology, Monash University, VIC 3800, Australia
{nliu36,nayyar.zaidi}@monash.edu

Abstract. Advent of *Deep Learning* and the emergence of *Big Data* has led to renewed interests in the study of Artificial Neural Networks (ANN). An ANN is a highly effective classifier that is capable of learning both linear and non-linear boundaries. The number of hidden layers and the number of nodes in each hidden layer (along with many other parameters) in an ANN, is considered to be a model selection problem. With success of deep learning especially on big datasets, there is a prevalent belief in machine learning community that a deep model (that is a model with many number of hidden layers) is preferable. However, this belies earlier theorems proved for ANN that only a single hidden layer (with multiple nodes) is capable of learning any arbitrary function, i.e., a shallow broad ANN. This raises the question of whether one should build a deep network or go for a broad network. In this paper, we do a systematic study of depth and breadth of an ANN in terms of 0-1 Loss, RMSE, Bias, Variance and convergence performance on 72 standard UCI datasets.

1 Introduction

The emergence of *Big Data* and success of deep learning on some problems have sparked a new interest in the study and evaluation of Artificial Neural Networks (ANN). ANN has a long history in the field of Artificial Intelligence. Earliest instance of ANN came in the form of Perceptron algorithm [1]. Perceptron was not capable of handling non-linear class boundaries and, therefore, interest in them remained fairly limited. Breakthrough was achieved by the invention of Backpropagation algorithm and multi-layer Perceptron that could model non-linear boundaries that led to a golden era of ANN. However, results were not as good as one had expected. Attention of Artificial Intelligence (AI) community was soon diverted to more mathematically sound alternative such as Support Vector Machines (with convex objective function) and non-parametric models such as Decision Trees, Random Forest, etc.

With advancements in the computing infrastructure, access to much bigger training datasets and better (for example greedy layer-wised) training algorithms [2] – ANN demonstrated much successes in its application on structured problems[1]

[1] These are mostly the problems in text, vision and NLP where there is a certain structure present in the input features. For example, deep learning performed extremely well on MNIST digit dataset (accuracy improved over 20%) when compared to typical machine learning algorithms.

and, therefore, there has been a renewed interest in the study of ANN mostly under the umbrella term of *deep* learning. However, applicability of deep learning on problems where there is no structure in the input and studying its efficacy over other competing machine learning algorithms remains to be further investigated.

Deep refers to the fact that the network has to be very deep. Why is deep learning effective? We believe that this is due to the feature engineering capability of a deep network. For example, for image recognition, lower-level layers represent the pixels and middle-levels represent the edges made of these pixels and higher-levels represent the concepts made from these edges. An example is the convolutional neural network (CNN), where there is an explicit filtering stage (convolution), followed by max-polling and then a fully-connected network. In an CNN, an image is convolved with a kernel to incorporate the spatial information present in the image, resulting in higher-order features [2]. This is followed by some sampling steps, for example, max-pooling to reduce the size of the output. There can be many convolution and max-pooling layers, which are followed by a fully-connected multi-layer perceptron. What about problems where we can not rely on interactions in the features? In other words, if we cannot rely on convolutional and max-pooling layers, will deep learning be as effective as it is on the structured problems? Well, we argue, that the success of deep learning can be attributed to the fact that it takes into account higher-order interactions among the features of the data. And when faced with extremely large quantities of training data, taking into account these higher-order interactions is beneficial [3].

Therefore, for un-structured problems, there are several directions one could take of which we mention only three in the following:

1. Take the quadratic, cubic or higher-order features and feed them to an ANN with no hidden layers [4].
2. Train a multi-layer ANN with many hidden layers – deep model.
3. Train an ANN with single hidden layer but many nodes – broad model.

We leave option one as an area of future research, and will focus on the second and third option.

To the best of our knowledge, there are no comparative studies of deep and broad ANN in terms of their performance, bias and variance profiles and convergence analysis. We claim our main contributions in this paper are:

– We provide a comparison of 0-1 Loss, RMSE, bias and variance of deep ANNs (with two, three, four and five hidden layers) and broad ANNs (with two, four, eight and ten nodes in one hidden layer). We show that on standard UCI datasets [8], broad ANNs leads to better results than deeper models.
– We compare the convergence analysis of deep ANNs and broad ANNs. We show that broad ANN have far superior convergence profiles than deeper models.

The rest of this paper is organized as follows. In Section 2, we provide a brief overview of ANN and learning algorithm used to train the models. In Section 3,

[2] Earlier applications of multi-layer perceptrons presented only the pixels to the input layer which failed to encode the structure present in the images.

we provide the empirical results. We conclude in section 4 with pointers to future work.

2 A Simple Feed-Forward ANN

An Artificial Neural network (ANN) mimics how the neurons in a biological brain work. In the brain, each neuron receives input from other neurons. The effect and magnitude of the input is controlled by synaptic weight. The weight will change over time so that the brain can learn and perform tasks. We will denote the input layer of an ANN as X, each hidden layer as H_k, and the output layer will be denoted as Y. In this paper, we only considered sigmoid nodes with the activation function of form: $a_k = f(h_k \cdot w_k)$, where w_k represent the weighting of each layer as mentioned earlier. Also, w_0 is the weighting for the input layer. Here, h_k represents the input of each layer and a_k represents the output of each layer. Since the output of the previous layer is the input of the current layer, the following equality holds:

$$a_{k-1} = h_k \tag{1}$$

Note that the output of the last hidden layer is input of the output layer. Hence a_s, where s represents the index of the final hidden layer, will be the prediction given by the NN.

The objective function of an ANN is to minimize the Mean Squared Error (MSE) that is:

$$J(w) = \frac{1}{2}(a_s - Y)^2.$$

The weights for all layers are randomized initially. Using the input value h_0, all activations functions are calculated layer by layer allowing for an output a_s. This process is called forward propagation. Using this output value, back propagation can be implemented through gradient descent to minimize the MSE.

In order to use gradient descent, we need the partial derivatives of each w_k: $\triangle w_k = \frac{\delta J}{\delta w_k}$. Let us show the derivation of partial derivatives of the last two layers and the pattern will be apparent. For the output layer s:

$$\frac{\delta J}{\delta w_s} = \frac{\delta J}{\delta a_s}\frac{\delta a_s}{\delta w_s}.$$

The first part is the plain derivative of the cost function with respect to a_s.

$$\frac{\delta J}{\delta a_s} = (a_s - y).$$

The second part is the derivative of the sigmoid function with respect to w_s. Since both h_s and w_s are variables, chain rule must be used:

$$\frac{\delta a_s}{\delta w_s} = f'(h_s \cdot w_s)h_s.$$

Putting it altogether:

$$\triangle w_s = (a_s - y)f'(h_s \cdot w_s)h_s = \gamma_s h_s. \tag{2}$$

Note that we have introduced a new notation of γ_k. For the second last layer $s-1$, we have:

$$\frac{\delta J}{\delta w_{s-1}} = \frac{\delta J}{\delta a_{s-1}} \frac{\delta a_{s-1}}{\delta w_{s-1}}.$$

Let us focus on the first part:

$$\frac{\delta J}{\delta a_{s-1}} = \sum \frac{\delta}{\delta a_{s-1}} \left[\frac{1}{2}(a_s - y)^2 \right].$$

Note, the above sum is over all the nodes in hidden layer $s-1$. We have removed the subscript indexing each node in layer $s-1$ for simplicity purposes. In the following equations, all the summations are over the number of nodes in the hidden layer unless explicitly stated otherwise. We expand above Equation as:

$$\begin{aligned} \frac{\delta J}{\delta a_{s-1}}, &= \sum (a_s - y) \frac{\delta a_s}{\delta a_{s-1}}, \\ &= \sum (a_s - y) \frac{\delta a_s}{\delta h_s}, \\ &= \sum (a_s - y) f'(h_s \cdot w_s) w_s = \sum \gamma_s w_s. \end{aligned}$$

The jump from the second line to the third line above is valid because of Equation 1. The summation sign is needed because each node in layer $s-1$ provides an input to each node in layer s, hence the partial derivatives need to be added together. The second part is the same as before:

$$\frac{\delta a_{s-1}}{\delta w_{s-1}} = f'(h_{s-1} \cdot w_{s-1}) h_{s-1}.$$

Putting it together:

$$\triangle w_{s-1} = \sum \gamma_{s-1} h_{s-1} = \sum (a_s - y) f'(h_s \cdot w_s) w_s \times f'(h_{s-1} \cdot w_{s-1}) h_{s-1}.$$

It can be seen that:

$$\gamma_{s-1} = \sum \gamma_s w_s \cdot f'(h_{s-1} \cdot w_{s-1}).$$

Using this, we have a general formula for each w_k:

$$\triangle w_{k-1} = \sum \gamma_{k-1} h_{k-1} = \sum \gamma_k w_k f'(h_{k-1} \cdot w_{k-1}) h_{k-1},$$

where $\gamma_{k=s}$ is provided in Equation 2.

All the above derivation was for a single class of a single instance. To generalize it, we only need to sum over all instances and over all classes hence the following new cost function – with n_c classes and n features, we have:

$$J(w) = \sum^{n_c} \sum^{n} \frac{1}{2}(a_s - y)^2.$$

Luckily, derivatives of sums are the sum of the derivatives. Hence all the derivation above is valid in the general case.

$$\triangle w_{k-1} = \sum \gamma_{k-1} h_{k-1} = \sum \gamma_k w_k f'(h_{k-1} \cdot w_{k-1}) h_{k-1},$$

with

$$\gamma_s = \sum^{n_c} \sum^{n} (a_s - y) f'(h_s \cdot w_s).$$

In this work, we used Stochastic Gradient Descent (SGD) for optimization. We leave investigation of batch-optimization methods such as quasi-Newton (L-BFGS) and conjugate Gradient methods for future work [3].

3 Experimental Results

In this section, we compare and analyze the performance of our proposed algorithms and related methods on 72 natural domains from the UCI repository of machine learning datasets [8]. The experiments are conducted on the datasets described in Table 1. 40 datasets have fewer than $1,000$ instances, 20 datasets have between $1,000$ and $10,000$ instances and 12 datasets have more than $10,000$ instances. These datasets are shown in bold font in Table 1. Each algorithm is tested on each dataset using 3 rounds of 2-fold cross validation. We compare four different metrics, i.e., 0-1 Loss, RMSE, Bias and Variance[4]. There are a number of different bias-variance decomposition definitions. In this research, we use the bias and variance definitions of [9] together with the repeated cross-validation bias-variance estimation method proposed by [10]. Kohavi and Wolpert [9] define bias and variance as follows: $\text{bias}^2 = \frac{1}{2} \sum_{y \in \mathcal{Y}} \Big(\mathrm{P}(y|\mathbf{x}) - \hat{\mathrm{P}}(y|\mathbf{x}) \Big)^2$, and $\text{variance} = \frac{1}{2} \Big(1 - \sum_{y \in \mathcal{Y}} \hat{\mathrm{P}}(y|\mathbf{x})^2 \Big)$. We report Win-Draw-Loss (W-D-L) results when comparing the 0-1 Loss, RMSE, bias and variance of two models. A two-tail binomial sign test is used to determine the significance of the results. Results are considered significant if $p \leq 0.05$ and shown in bold.

The datasets in Table 1 are divided into three categories. When discussing results, we denote all the datasets in Table 1 as *All*. We denote the following datasets as *Big* – `Poker-hand`, `Covertype`, `Census-income`, `Localization`, `Connect-4`, `Shuttle`, `Adult`, `Letter-recog`, `Magic`, `Sign`, `pendigits`. The remaining datasets (i.e., {*All* - *Big*}) are denoted as *Little* in the results.

Numeric features are discretized by using the Minimum Description Length (MDL) supervised discretization method [11]. A missing value is treated as a separate feature value and taken into account exactly like other values.

[3] As the comparison is between a deep and broad model of the same ANN – the exact implementation of the ANN is less consequential, as long as the implementation is consistent between the two, we believe that the results should remain valid.

[4] The reason for performing bias/variance estimation is that it provides insights into how the learning algorithm will perform with varying amounts of data. We expect low variance algorithms to have relatively low error for small data and low bias algorithms to have relatively low error for large data [6].

Domain	**Case**	**Att**	**Class**	**Domain**	**Case**	**Att**	**Class**
Poker-hand	1175067	11	10	Annealing	898	39	6
Covertype	581012	55	7	Vehicle	846	19	4
Census-Income(KDD)	299285	40	2	PimaIndiansDiabetes	768	9	2
Localization	164860	7	3	BreastCancer(Wisconsin)	699	10	2
Connect-4Opening	67557	43	3	CreditScreening	690	16	2
Statlog(Shuttle)	58000	10	7	BalanceScale	625	5	3
Adult	48842	15	2	Syncon	600	61	6
LetterRecognition	20000	17	26	Chess	551	40	2
MAGICGammaTelescope	19020	11	2	Cylinder	540	40	2
Nursery	12960	9	5	Musk1	476	167	2
Sign	12546	9	3	HouseVotes84	435	17	2
PenDigits	10992	17	10	HorseColic	368	22	2
Thyroid	9169	30	20	Dermatology	366	35	6
Pioneer	9150	37	57	Ionosphere	351	35	2
Mushrooms	8124	23	2	LiverDisorders(Bupa)	345	7	2
Musk2	6598	167	2	PrimaryTumor	339	18	22
Satellite	6435	37	6	Haberman'sSurvival	306	4	2
OpticalDigits	5620	49	10	HeartDisease(Cleveland)	303	14	2
PageBlocksClassification	5473	11	5	Hungarian	294	14	2
Wall-following	5456	25	4	Audiology	226	70	24
Nettalk(Phoneme)	5438	8	52	New-Thyroid	215	6	3
Waveform-5000	5000	41	3	GlassIdentification	214	10	3
Spambase	4601	58	2	SonarClassification	208	61	2
Abalone	4177	9	3	AutoImports	205	26	7
Hypothyroid(Garavan)	3772	30	4	WineRecognition	178	14	3
Sick-euthyroid	3772	30	2	Hepatitis	155	20	2
King-rook-vs-king-pawn	3196	37	2	TeachingAssistantEvaluation	151	6	3
Splice-junctionGeneSequences	3190	62	3	IrisClassification	150	5	3
Segment	2310	20	7	Lymphography	148	19	4
CarEvaluation	1728	8	4	Echocardiogram	131	7	2
Volcanoes	1520	4	4	PromoterGeneSequences	106	58	2
Yeast	1484	9	10	Zoo	101	17	7
ContraceptiveMethodChoice	1473	10	3	PostoperativePatient	90	9	3
German	1000	21	2	LaborNegotiations	57	17	2
LED	1000	8	10	LungCancer	32	57	3
Vowel	990	14	11	Contact-lenses	24	5	3
Tic-Tac-ToeEndgame	958	10	2				

Table 1. Details of Datasets (UCI Domains)

3.1 Comparison in terms of W-D-L and Geometric Averages

Let us start by comparing five broad models with five deep models in terms of their Win, Draw and Loss results on standard datasets. We denote broad models as NN2, NN4, NN6, NN8 and NN10. This denotes ANN with one hidden layer with two, four, six, eight and ten nodes in the hidden layer.

We denote deep models as NN2, NN22, NN222, NN2222, NN22222. This denotes ANN with one hidden layer with two nodes, two hidden layers with two nodes each, three hidden layers with two nodes each, four hidden layers with two nodes each and five hidden layers with two nodes each, respectively.

We also include an ANN with no hidden layer. This is denoted as NN0. This is actually similar to Logistic Regression except that it is using Mean-square-error instead of Conditional Log-Likelihood (CLL) [4].

Broad Models Let us start with comparing the bias and variance of broad models. We compare the results in Table 2. A systematic trend in bias, as expected can be seen. A broader model is lower-biased than the less broader model. NN10 being lowest biased winning significantly in terms of W-D-L when compared to other lesser broad models. Variance results are slightly surprising. An NN10, though has (non-significant) higher variance than NN0, has significantly lower variance than NN2, NN4 and non-significantly lower variance than NN6 and NN8.

	vs. NN0		vs. NN2		vs. NN4		vs. NN6		vs. NN8	
	W-D-L	*p*	W-D-L	*p*	W-D-L	*p*	W-D-L	*p*	W-D-L	*p*
					All Datasets - Bias					
NN2	35/3/34	1								
NN4	45/4/23	**0.010**	49/7/16	**<0.001**						
NN6	47/4/21	**0.002**	47/5/20	**0.001**	37/7/28	0.321				
NN8	48/3/21	**0.002**	44/5/23	**0.014**	37/7/28	0.321	36/11/25	0.200		
NN10	52/3/17	**<0.001**	47/5/20	**0.001**	41/9/22	**0.023**	43/10/19	**0.003**	40/15/17	**0.003**
					All Datasets - Variance					
NN2	20/2/50	**<0.001**								
NN4	21/2/49	**0.001**	38/6/28	0.268						
NN6	27/3/42	0.091	43/7/22	**0.013**	40/8/24	0.060				
NN8	32/2/38	0.550	42/7/23	**0.025**	44/8/20	**0.004**	36/9/27	0.314		
NN10	30/3/39	0.336	42/7/23	**0.025**	43/9/20	**0.005**	34/13/25	0.298	33/10/29	0.704

Table 2. A comparison of Bias and Variance of broad models in terms of W-D-L on *All* datasets. p is two-tail binomial sign test. Results are significant if $p \leq 0.05$.

As discussed in Section 1, lower-bias of broad models translates into better 0-1 Loss and RMSE performance on *Big* datasets. This can be seen in Table 3. It can be seen that on *Big* datasets, NN10 leads to much better performance than NN8, NN6, NN4, NN2 and NN0. Some of the results are not significant statistically, but a general trend can be seen that a broader ANN leads to better performance than less broader ANN.

	vs. NN0		vs. NN2		vs. NN4		vs. NN6		vs. NN8	
	W-D-L	*p*	W-D-L	*p*	W-D-L	*p*	W-D-L	*p*	W-D-L	*p*
					All Datasets – 0-1 Loss					
NN2	27/2/43	0.072								
NN4	31/6/35	0.712	50/9/13	**<0.001**						
NN6	33/3/36	0.801	49/3/20	**<0.001**	45/7/20	**0.003**				
NN8	37/1/34	0.813	50/5/17	**<0.001**	44/8/20	**0.004**	31/14/27	0.694		
NN10	40/2/30	0.282	51/4/17	**<0.001**	49/5/18	**<0.001**	38/9/25	0.130	40/8/24	0.060
					Big Datasets – 0-1 Loss					
NN2	6/0/6	1.226								
NN4	7/0/5	0.774	12/0/0	**0.011**						
NN6	7/0/5	0.774	12/0/0	**0.001**	11/0/1	**0.006**				
NN8	8/0/4	0.388	12/0/0	**<0.001**	9/0/3	0.146	8/0/4	0.388		
NN10	8/0/4	0.388	12/0/0	**<0.001**	10/0/2	**0.039**	9/0/3	0.146	9/0/3	0.146
					All Datasets – RMSE					
NN2	40/1/31	0.3425								
NN4	46/0/26	**0.025**	51/0/21	**<0.001**						
NN6	45/0/27	**0.044**	46/2/24	**0.012**	43/2/27	0.07				
NN8	49/0/23	**0.003**	47/0/25	**0.012**	44/2/26	**0.041**	38/2/32	0.550		
NN10	52/0/20	**<0.001**	48/1/23	**0.004**	47/2/23	**0.006**	39/3/30	0.336	38/6/28	0.268
					Big Datasets – RMSE					
NN2	8/0/4	0.388								
NN4	9/0/3	0.146	12/0/0	**<0.001**						
NN6	9/0/3	0.146	12/0/0	**<0.001**	10/0/2	**0.039**				
NN8	10/0/2	**0.039**	11/0/1	**0.006**	10/0/2	**0.039**	9/0/3	0.146		
NN10	10/0/2	**0.039**	10/1/1	**0.011**	9/0/3	0.146	9/0/3	0.146	8/0/4	0.388

Table 3. A comparison of 0-1 Loss and RMSE of broad models in terms of W-D-L on *All* and *Big* datasets. p is two-tail binomial sign test. Results are significant if $p \leq 0.05$.

Finally, let us compare the geometric averages of broad models on standard datasets in Figure 1. The results corroborates W-D-L results. It can be seen that

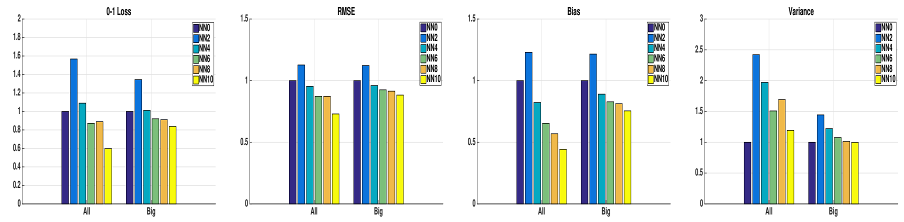

Fig. 1. Comparison (geometric average) of 0-1 Loss, RMSE, Bias and Variance for broad models on *Little* and *Big* datasets. Results are normalized w.r.t NN0.

a broader model leads to a low-biased classifier, which leads to low 0-1 Loss and RMSE. Similarly variance is higher than NN0, but one can not infer a trend in the growth of variance as model becomes broader and broader.

Deep Models Let us start by comparing the bias-variance profile of deep models in Table 4. It can be seen that unlike broad models, a deep model does not lead to low bias at all. In fact, bias increases as the model is made deeper and deeper. It can be seen that only NN2 wins to NN0 (a bare win over one dataset), whereas all other deeper models lose to NN0 in terms of the bias. Similarly, all deeper models except NN22222 loses significantly to NN0 in terms of variance. N22222 (non-significantly) wins to NN0 in terms of the variance.

	vs. NN0		**vs. NN2**		**vs. NN22**		**vs. NN222**		**vs. NN2222**	
	W-D-L	p	W-D-L	p	W-D-L	p	W-D-L	p	W-D-L	p
					All Datasets – Bias					
NN2	35/3/34	1								
NN22	30/3/39	0.336	28/4/40	0.182						
NN222	26/1/45	**0.032**	21/3/48	**0.002**	24/4/44	**0.021**				
NN2222	5/0/67	**<0.001**	3/1/68	**<0.001**	3/2/67	**<0.001**	4/9/59	**<0.001**		
NN22222	0/1/71	**<0.001**	0/1/71	**<0.001**	1/2/69	**<0.001**	1/9/62	**<0.001**	0/61/11	**<0.001**
					All Datasets – Variance					
NN2	20/2/50	**<0.001**								
NN22	20/1/51	**<0.001**	27/6/39	0.175						
NN222	24/1/47	**0.009**	34/3/35	1	32/4/36	0.905				
NN2222	34/1/37	0.813	34/1/37	0.813	36/2/34	0.905	32/9/31	1		
NN22222	40/2/30	0.282	38/1/33	0.6353	39/2/31	0.403	35/9/28	0.450	8/61/3	0.227

Table 4. Bias W-D-L on *All* and *Big* datasets. p is two-tail binomial sign test. Results are significant if $p \leq 0.05$.

A comparison of W-D-L of 0-1 Loss and RMSE values of deep models is given in Table 5. It can be seen that NN2 and NN22 are competitive with NN0 in terms of RMSE on *Big* datasets, deeper models NN222, NN2222 and NN22222 are not very competitive to simple model such as NN0. A comparison of the average 0-1 Loss, RMSE, Bias and Variance of different deep models is given in Figure 2. It can be seen that deeper the model, higher its bias is. The performance in terms of the variance is very surprising, as it shows that the deep models leads to lower-variance than simple models such as NN0.

	vs. NN0		vs. NN2		vs. NN22		vs. NN222		vs. NN2222	
	W-D-L	p	W-D-L	p	W-D-L	p	W-D-L	p	W-D-L	p
					All Datasets – 0-1 Loss					
NN2	27/2/43	0.072								
NN22	28/1/43	0.096	24/5/43	**0.027**						
NN222	24/1/47	**0.009**	25/5/42	**0.050**	28/3/41	0.148				
NN2222	7/0/65	**<0.001**	4/2/66	**<0.001**	4/2/66	**<0.001**	3/9/60	**<0.001**		
NN22222	7/1/64	**<0.001**	5/1/66	**<0.001**	4/2/66	**<0.001**	3/9/60	**<0.001**	1/61/10	**0.012**
					Big Datasets – 0-1 Loss					
NN2	6/0/6	1.226								
NN22	5/0/7	0.774	4/0/8	0.388						
NN222	4/0/8	0.388	2/0/10	**0.039**	4/0/8	0.388				
NN2222	2/0/10	**0.039**	0/0/12	**<0.001**	1/0/11	**0.006**	1/1/10	**0.012**		
NN22222	1/1/10	**0.012**	0/0/12	**<0.001**	0/0/12	**<0.001**	0/1/11	**<0.001**	0/6/6	**0.031**
					All Datasets – RMSE					
NN2	40/1/31	0.343								
NN22	41/0/31	0.289	24/0/48	**0.006**						
NN222	35/0/37	0.9063	19/0/53	**<0.001**	24/2/46	**0.011**				
NN2222	15/0/57	**0.044**	4/0/68	**<0.001**	3/2/67	**<0.001**	3/8/61	0.072		
NN22222	14/0/58	**<0.001**	5/0/67	**<0.001**	4/2/66	**<0.001**	3/8/61	**<0.001**	2/58/12	**0.013**
					Big Datasets – RMSE					
NN2	8/0/4	0.388								
NN22	9/0/3	0.146	3/0/9	0.146						
NN222	8/0/4	0.388	1/0/11	**0.006**	4/0/8	0.388				
NN2222	5/0/7	0.774	0/0/12	**<0.001**	0/0/12	**<0.001**	1/1/10	**0.023**		
NN22222	3/0/9	0.146	0/0/12	**<0.001**	0/0/12	**<0.001**	0/1/11	**<0.001**	0/6/6	**0.031**

Table 5. 0-1 Loss W-D-L on *All* and *Big* datasets. p is two-tail binomial sign test. Results are significant if $p \leq 0.05$.

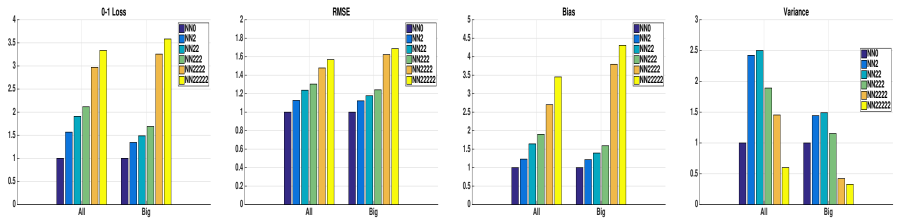

Fig. 2. Comparison (geometric average) of 0-1 Loss, RMSE, Bias and Variance for deep models on *Little* and *Big* datasets. Results are normalized w.r.t NN0.

3.2 Deep vs. Broad Models – Discussion

From the comparative results on deep and broad models, one can conclude that broader models have better performance than deeper models. However, we stress, that we are comparing very different models in terms of the number of parameters being optimized [5], and, hence, we didn't gave a direct comparison of broad and deep models. For example, with 10 numeric features and three classes, NN10 will learn 143, whereas NN22222 will learn 61 parameters. Therefore, we used NN0 as the base model against which we compared the performance. We showed that

[5] If n is the no. of attributes and n_c is the number of classes, then, broad model has $((n+1)*N(H_1)) + ((N(H_1)+1)*n_c)$ number of parameters, where $N(H_1)$ denotes the number of nodes in hidden layer H_1. A deep model, on the other hand, optimizes $((n+1)*N(H_1)) + \sum_{k=1}^{K}(N(H_k)+1*N(H_{k+1})) + ((N(H_K)+1)*n_c)$ number of parameters.

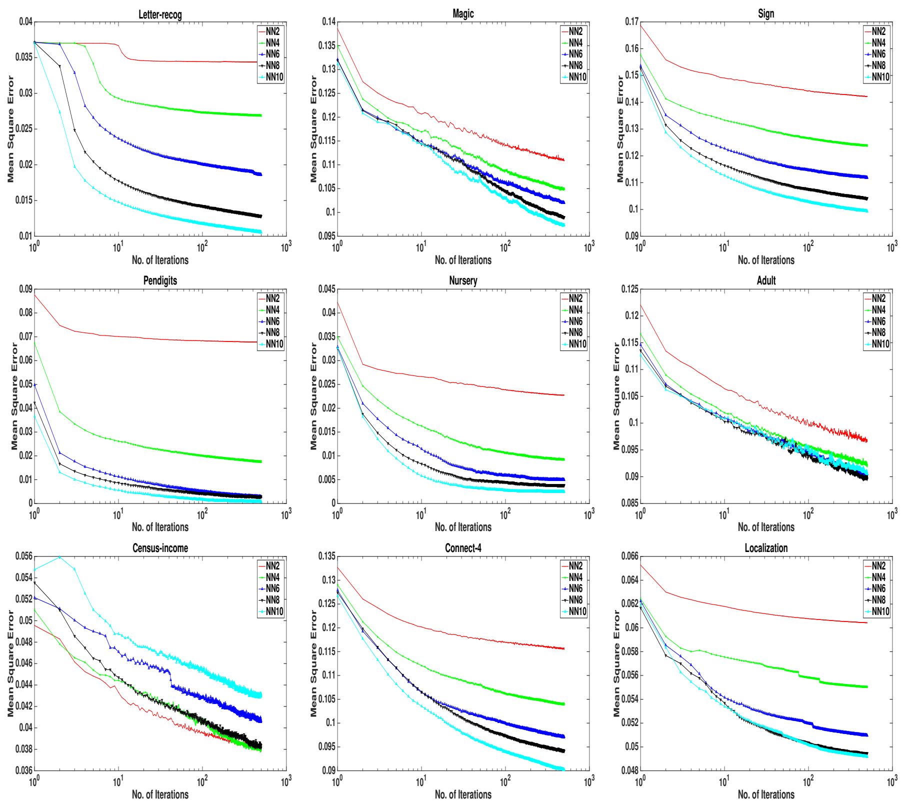

Fig. 3. Variation in Mean Square Error of NN2, NN4, NN6, NN8 and NN10 with increasing number of (optimization) iterations on sample datasets – `letter-recog`, `magic`, `sign`, `pendigits`, `nursery`, `adult`, `census-income`, `connect-4`, `localization`.

broader models results in decreasing the bias of NN0, whereas, deeper models were not successful in doing that. For example, NN4 and NN22222 will learn (almost) the same number of parameters with 10 numeric features and 3 classes, NN4 wins in terms of Bias with NN0 45 times, draws 4, and loses 23 times. NN22222 loses to NN0 in terms of bias, 71 times.

In Figures 3 and 4, we compare the convergence profiles of broad and deep models respectively on sample datasets. Note, what we plot is the variation in mean-square-error on training data. Of course, the results are averaged over three rounds of two-fold cross validation. As just mentioned, we do not compare broad and deep models directly, but instead used NN2 as the reference point between the two paradigms. As we expected, a broader model results in better convergence profile than less broader model. On all except one dataset (`census-income`) that is shown in Figure 3, NNk results in better convergence than NN(k-1). For deep models, it can be seen that most models are worst than NN2 (red line) except on `census-income` datasets, where NN22 (green line) leads to the best convergence. We argue, that based on these results, broader models not only lead to better results, but also have better convergence profile. Of course, this property will lead to faster training and better efficiency overall.

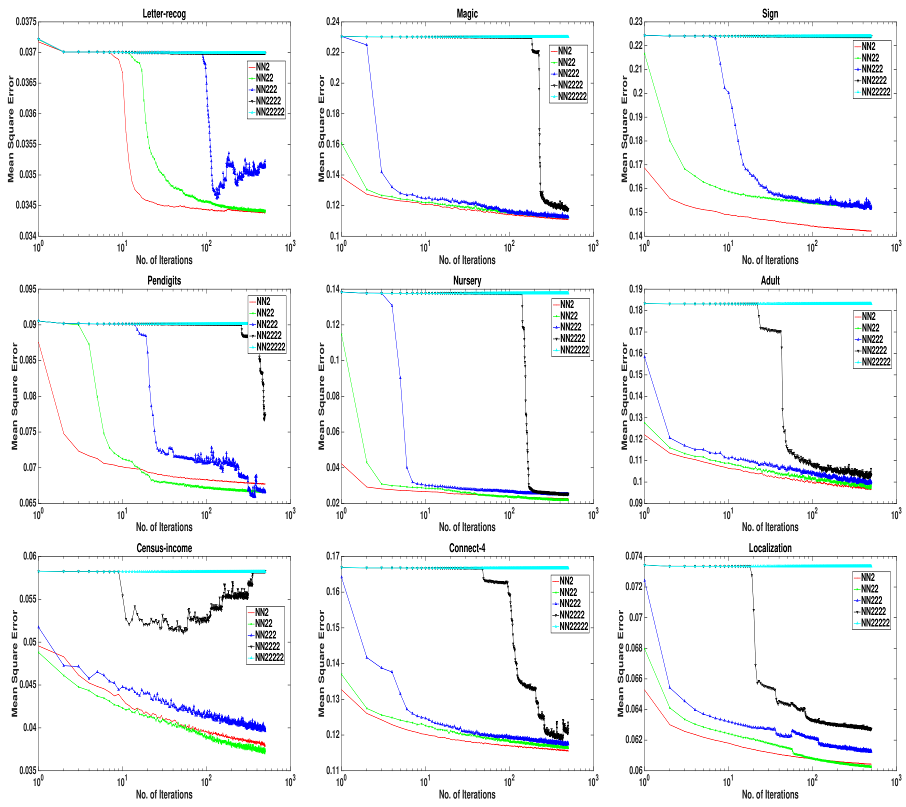

Fig. 4. Variation in Mean Square Error of NN2, NN22, NN222, NN2222 and NN22222 with increasing number of (optimization) iterations on sample datasets – `letter-recog`, `magic`, `sign`, `pendigits`, `nursery`, `adult`, `census-income`, `connect-4`, `localization`.

4 Conclusion and Future Works

Deep artificial neural networks have proven to be very effective for extremely large datasets on structured problems. However, on non-structured, problems, it is not very clear, how good the performance will be. An obvious question is the expansion of the breadth of ANN instead of depth. In this paper, we studied empirically the effect of altering the breadth and depth of ANN on it performance. We analysed performance in terms of 0-1 Loss, RMSE, bias and variance. Since success of deep learning is attribute to creating higher-order features, we expected low-bias classifiers for deep ANN as well as for broad ANN. However, our results showed that only broader ANN led to a lower-biased model. Deeper models led to increasing the bias.

The results we have presented in this paper are preliminary and warrants further investigation. There are several areas which we needs to be explored:

– The number of nodes in deeper model are set to two. Since the datasets in our experiments are not hugely big (biggest is the `poker-hand` with 1.17 million instances), we assumed that five hidden-layer neural network with two hidden

nodes in each layer covers the essence of a deep network. Obviously, the number of nodes in each layer effects the performance and a systematic study is needed to assess the effect on performance.
- The maximum number of optimization iterations are set to 500. It can be the case that a deeper network just converge slowly, therefore, variations with more number of iterations needs to be tested.
- In this work, we have constrained ourselves to (feed-forward) multi-layer perceptrons – further experimentations needs to be conducted to check if results will hold on other ANN types such as Deep Belief Networks or Restricted Boltzmann machines, etc.

5 Acknowledgment

Authors would like to thank Francois Petitjean, Geoff Webb and Reza Haffari for helpful discussion during the course of this work. Nian Liu was supported by ECR 2015 seed grant by the Faculty of Information Technology, Monash University, Australia.

References

1. F. Rosenblatt, "The perceptron–a perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Tech. Rep. 85-460-1, 1957.
2. G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Compuation*, 2006.
3. X. Zhang and Y. LeCun, "Text understanding from scratch," arXiv:1502.01710, 2015.
4. N. A. Zaidi, F. Petitjean, and G. I. Webb, "Preconditioning an artificial neural network using naive bayes," in *Advances in Knowledge Discovery and Data Mining*, 2016, pp. 341–353.
5. N. A. Zaidi, G. I. Webb, M. J. Carman, F. Petitjean, and J. Cerquides, "ALR^n: Accelerated higher-order logistic regression," *Machine Learning*, pp. 1–44, 2016.
6. D. Brain and G. I. Webb, "The need for low bias algorithms in classification learning from small data sets," in *PKDD*, 2002, pp. 62–73.
7. N. A. Zaidi, M. J. Carman, J. Cerquides, and G. I. Webb, "Naive-Bayes inspired effective pre-conditioners for speeding-up logistic regression," in *IEEE International Conference on Data Mining*, 2014, pp. 1097–1102.
8. A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml
9. R. Kohavi and D. Wolpert, "Bias plus variance decomposition for zero-one loss functions," in *ICML*, 1996, pp. 275–283.
10. G. I. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, no. 2, pp. 159–196, 2000.
11. U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Machine Learning*, vol. 8, no. 1, pp. 87–102, 1992.