Bayesian Approach to Deep Learning

Abhishek K. Singh
u6411540@anu.edu.au

Abstract. The optimization of weights in a traditional Neural Network is driven by the, fairly obvious, intention of reducing the error in prediction. This can be visualized as an application of the Maximum Likelihood Estimation for weights estimation. While the point estimates prove good in dealing with certain complex situations [1]., the arguments of why using a frequentist approach still remains. In this paper we use a Bayesian approach to Neural network and a traditional deep learning neural network on a dataset and prove why Bayesian might be a better choice in certain conditions.

Keywords: Deep Learning, Bayesian Neural Network, Posterior Distributions

1 Introduction

Neural networks are revolutionary tools in machine learning which are based on the concept of learning by optimizing the Gradient Descent through back propagation. Gradient descent on its part, generally attributed to Augustine- Louis Cauchy, had been well discussed to minimize error in the parametric space of complex, nonlinear, differentiable, multi-stage, NN-related system in the 1960s [2]. It is no wonder hence, for it to be tried at the first for error minimization, as a natural choice by researchers working in developing an ANN. However, if we dive slightly deep into how we have been treating our traditional ANN or DNN, we might be tempted to say that it is indeed a frequentist approach! There is nothing wrong with it but it implicitly [3] calls for a Bayesian treatment of the same.

In this paper, we will continue [1] our analysis on the forest cover Data which would be implemented on a Traditional DNN and a Bayesian Based DNN (also called as BNN) and compare the results of both with the results of an existing research paper [4]. It is to be noted that we would be using the same methodology [6], for data pre- processing in both the cases and hence the data fed into the function for both the cases would be same.

1.1 Why Bayesian?

Suppose we are watching a game of football. The scores are 6-0 and we are in the final minute of the game. And we see the climax: The losing team scores a goal each on 89 minutes 10 seconds, 89 minutes 20 seconds, 89 minutes 30 seconds. We can also add here is that in all the goals, the touch of a certain player is involved. Would our mind infer, that because the scoring rate in the last minute has been a goal every 10 seconds, so we will have 3 mores goals in the final 30 seconds given 'that' certain player has the ball? And hence the final score would be 6-6 indeed? The realistic answer is a No, our mind won't. The reason is that our mind is guided by "prior beliefs". These beliefs would most likely reinforce our mind with the thought that the 3 goals that were just scored were indeed due to divine momentary luck, but it is difficult to believe that it would be carried to through the remaining length of the game as well. An article [5] published in the *Current Biology* focusses on how the brain makes the decisions. It throws emphasis on the decision making of the brain which is influenced by the prior beliefs of an individual. Hence while replicating a machine version of the brain, it becomes necessary to incorporate the prior beliefs.

The Neural networks we have been dealing with till now lacks this "belief" factor. It is true that we can introduce an L2 regularization factor and sort of give a belief to the network. But this might not be the most appropriate way given we have a plausible Bayesian approach. Further it is reasonable to involve the beliefs somehow because in certain situations a neural network might fail straight away. An easy to look example would be to predict whether a person is atheist or a believer whether he has seen the god or not.

Further Bayesian Neural Networks are sort of imbibed with natural Regularization properties. As we shall see later, this helps them to avoid overfit the data and are particularly helpful in the cases when the distribution of the output class is highly biased towards certain classes. When compared the with the results of the DNN (to be followed soon) and the ANN approach [2], the BNN outperformed both of them reasonably.

1.2 Neural Network as a Frequentist Approach

One interpretation, in the nutshell, might be that the difference between the frequentist and the Bayesian approach is the belief. The frequentist belief is close to truly predictable and the uncertainty in prediction of the future events can hence be attributed to the lack of sampling data. While in Bayesian, we have prior beliefs which shapes are decision in a way.
One of the most popular frequentist technique in parameter estimation is the MLE or the Maximum Likelihood Estimation. In the maximum likelihood approach, the parameters are estimated such that the value of a given likelihood function is maximized. Mathematically this can be written as:

$$\Phi \in \{\arg\max Ⱪ(\theta\ ;\ x)\} \tag{1}$$

where Φ is the estimate of θ and Ⱪ is the likelihood function. So, our estimate of θ would be something that maximizes the function Ⱪ.

Now consider, if we have in general a data with

1.) Number of instances represented as **N**
2.) Target/output variable represented by **y** and
3.) Input variable represented by **x,** where x the given d – dimensional vector.

Extending the above notation for **y** and **x** to a complete set, we write **Y** and **X** respectively. Now assuming[1] **y** to have a Normal Distribution with its mean being dependent on **x** and **ω**, we can write

$$P(y_i|\ (x_i, \omega)) = Ɲ((x_i, \omega), \sigma^2) \tag{2}$$

(to be noted here is that the equation on the R.H.S is not written in conditional form just for simplicity and to emphasize that we are not going to expand the R.H.S by the conditional – multiplicative rule of Probability, rather the Normal Distribution is the destination.)
Now the corresponding likelihood function would be: (Iterating over all the elements.)

$$P(y_i|\ (x_i, \omega)) = \ \prod_1^N Ɲ((x_i, \omega),\ \sigma^2) \tag{3}$$

Taking the negative logarithm and adding (in the last step) from 1 to *N*, we get, for the R.H.S

$$\begin{aligned} &= -\ln\left(\frac{1}{(\sigma\times\sqrt{(2\pi)}}\times e^{-\left(\frac{y-(x_i,\omega)}{\sigma^2}\right)^2}\right) \\ &= -ln\frac{1}{(\sigma\times\sqrt{(2\pi)}} + \left(\frac{y-(x_i,\omega)}{\sigma^2}\right)^2 \\ &= \ N\times\ ln\sigma + N\times\ln\sqrt{(2\pi)}\ +\ N\times\left(\frac{y-(x_i,\omega)}{\sigma^2}\right)^2 \end{aligned} \tag{4}$$

Now the last term can be visualized as the Mean squared error! Recall, that we took the negative logarithm on both the sides, this means for the L.H.S to increase R.H.S must decrease; hence we should have a small Mean – Squared error.
So, the idea is that talking about minimizing the mean squared error is the other way of saying: Maximizing the likelihood estimator or a frequentist approach to the neural Networks. It might be argued that if we take the cross – entropy function to visualize the error, then the approach should not be termed a frequentist. This is however not true, recall that we assumed that y had a gaussian distribution and with more general case[2], we can establish that cross entropy error can also be visualized as an MLE approach.

2 Bayesian Approach to Neural Network

The Bayesian inference in this case gets complex in its mathematics although its representation in its most raw form is easy to understand. The way a Bayesian Neural network works is based on MAP - Maximum A Priori. In the text to follow forward, we would be using quite a few terminologies which form the basis of our discussion. The full Bayesian treatment for the weight estimation requires some descent amount of mathematics which is not covered here to entirety and would require to introduce a lot of new terms. It is thus tried, that the use of mathematics be limited to the bare essential.

[1] The generalized case where y does not need to be a Gaussian is also possible, just for simplicity it is avoided here.
[2] See "*Pattern Recognition and Machine Learning*"- Christopher M. Bishop, does not cover the case exclusively, but provides a good background for investigation. Pg. 278 -282.

Bayes' theorem and the terminologies associated:

The conditional probability of an event X_i with respect to the outcome of an event Y is given as:

$$P\left(\frac{X_i}{Y}\right) = \frac{p\left(\frac{Y}{X_i}\right) \times P(X_i)}{P(Y)} \tag{5}$$

Where:

$P\left(\frac{X_i}{Y}\right)$ is called the Posterior Distribution.
$P(X_i)$ is the Prior Belief / A Priori
$P\left(\frac{Y}{X_i}\right)$ is called the class conditional Probability.
Now in the BNN (Bayesian Neural Network), weights are modelled as conditioned upon both the Input Training Data and the Classification output. Also "*it makes use of the posterior inference of the weight, along with the weights", to* make predictions about class y_i corresponding to a given x_i. So, this makes the weights to be given as (Using the Bayes theorem equation 5):

$$P(\omega/(\mathbf{X}_{Train}, Y_{Train})) = \frac{P(Y_{Train}|X_{Train}, \omega) \times P(\omega)}{Z} \tag{6}$$

Where:

Z = Normalization Constant
P(ω) = Prior/Prior Belief/ A priori
$P\ (Y_{Train} \mid X_{Train}, \omega)$ = Likelihood Function

The likelihood distribution can be evaluated by various methods (both parametric and Non-Parametric). The way it is calculated in the Edward Library[3], used to implement Bayesian Neural Network on python, is based on Gibbs sampling which is the subset of the larger Markov Chain Monte Carlo Methods.
Next, we estimate the value of occurrence of a general class y_i for a given x_i marginalized over the weight ω This can be written as:

$$\int P(y|(x, \omega) \times P(\omega|Y_{Train}, X_{Train})\ d\omega \tag{7}$$

An intuition to this equation can be thought of as follows:
To estimate an unknown, the neural network would estimate the probability of occurrence of that unknown by using all possible weights, which would be multiplied by the probability of occurrence of such weights. So, this means, that while summing over the weights, we might expect that a highly unlikely weight would be offset by its low probability of occurrence.
To evaluate this integral, we first divide the data into small batch sizes and then infer the conditional Probability for the weights. We then multiply the above obtained density function with the output of an ANN $P(y|(x, \omega)$ and evaluate the integral which is our final output. The process of evaluating the inference $P(\omega|Y_{Train}, X_{Train})$ is now done with these new weights and the cycle is repeated till the required epochs.
The M.A.P in turn can be thought of as the general case, whose subset is the MLE when the distribution of the weights is uniform.

2.2 Moving from ANN to DNN

DNNs are powerful classifiers which are capable of creating nonlinear decision boundaries around the data points. This makes them also prone to overfitting. In the analysis to follow, we implement a DNN with 2 hidden layers over the forest cover data. Here we continue our Analysis on the same lines as before, we use selective sampling [1] and pre- process the data as per done in the paper [5].

[3] More about it can be read in the ReadMe.txt

3 Method

3.1 Data Set Information

For this analysis, we modelled a real data set, which contained information associated with 7 types of forest convertype. This data was collected by USFS and the link to this data on the UCI website can be found in reference. This data consisted of 54 attributes for each of the covertype class. Of these 54 variables 10 were quantitative variable, 4 binary and 40 for each of the 40-soil type. (1 signifies presence, 0 the opposite). The technique used for prediction is DNN and BNN which has been implemented on Python. The data has been pre- process following the techniques used in a certain paper[6].

3.2 Overview of the Raw Data

Below is the distribution of output data[4]. The percentage distribution of the output variable in the data is:

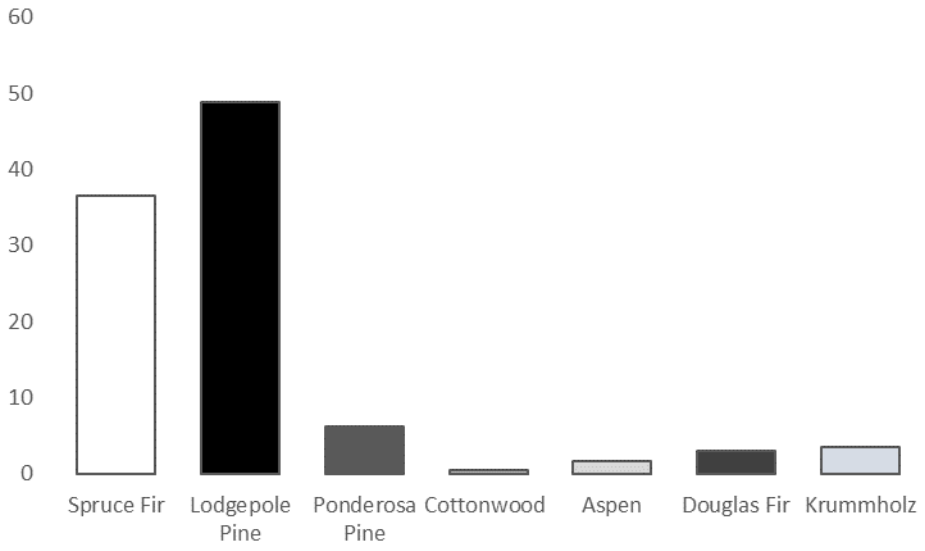

Fig.1. % Distribution of forest cover type

From the histogram, it can be seen that the top 2 most frequently occuring class have the total occurrence of more then 80%, combined together. As such this data is highly biased in the forest types Spruce- Fir , and Lodgepole Pine.

3.3 Data Processing :

Here we use a simple paramteric outlier removal technique for near similar gaussian distributions. From statistics we know that 99.73% of the data in a normal distribution is with 3 standard deviations from the mean. Keeping this , in mind, the values above or below 3 standard devations have been replaced with the mean. This technique has been applied to the the following variables :

1.) Elevation
2.) Slope
3.) Hillshade_9Am
4.) HillShade_noon
5.) Hillshade_3pm
6.) Horizontal_Distance_to_fire_points

[4] The distribution of the other variables have been covered previously (refer to 1 in reference) and hence not been added here to avoid redundancy.

3.3a Variable Grouping

Aspect
The attribute Aspect has been coded to remove the problem of continuity [6] . Post its bucketing , its statistical significance in predicting the output through a logistic model has been found to be fairy high. To code the aspect values , the values are classified into 8 directions North , North-East , East and so on.

Soil Types:
The information about the different soil types is fairly staraight forward, as obtained from the UCI website. The 40 subclasses of the soil type can be merged into 8 broad soil types .The need to do it arose from the very unsymemetric distribution of some of the soil stypes.

3.3b Normalising quanitiative variables

The quantitative variables : Elevation , Horizontal distance to hydrology, Horizontal Distance to roadways, Hillshade (all possible values) have been normalised , this is in line with the recommended practices.[7]

3.4 Initial Observations

The brute application of neural network on raw data gave an acurracy of ~52% on the Training Data and ~48% on the testing data.
A variable reduction technique was used subsequently which was based on the t-scores obtained while this data was run on a logistic regression model on R .In simple logistic regression, the output is modelled to the input variables , by a linear equation of the form :

$$y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + \cdots + a_nx_n \quad (8)$$

The general logistic regression model gives an output of z scores and the corresponding probabilty for all the input attributes x_i.The probability , in this case , is the probability that the null Hypothesis : $a_i = 0$, is true. Or the other way of saying this – it is the probabilty that the input variable is not significant . So , lower the probability means , the input variable is , in fact significant.
The t scores and the corresponding p-values when run on the processed are shown for the insignificant variables

Table 1 . Result for variable significance

Coefficients	t- value	$P(>t)$
Vertical_Distance	0.36	0.72
East	0.53	0.59
South East	0.69	0.59
North West	0.18	0.62
Mo. Sandstone	1.8	0.76

With the above result, the respective columns were removed from the data set and this gave the final processed data on which all the analysis has been done. The data was divided into training and testing samples in the ratio of 7: 3
At the first , the model was run on this processed training data using the DNN and the BNN . The results of the DNN showed a flaw in model learning because of the high biasing in the output data. The model failed to learn to predict 4 out of the 7 variables which had the least frequncy of occurene. Although the model gave accuracy of more than 50% which can be attributed to the fact that more then 50 % of the testing data was dominated by the 1st – 3 forest covertype and the model predicted one of those 3 covertype for every set of input variable. The confusion matrix of the model looks like as follows:

Table 2. Percentage correct vaule detected with DNN

Spruce-Fir	Loge.pin	Pond.Pin	Cottonwood	Apen	Doug.Fir	Krummhol
2.17	97.82	0.47	0.00	0.00	0.00	0.00
0.16	**99.65**	0.19	0.00	0.00	0.00	0.00
0.00	91.34	**8.66**	0.00	0.00	0.00	0.00
0.00	92.86	7.14	**0.00**	0.00	0.00	0.00
0.00	100.0	0.00	0.00	**0.00**	0.00	0.00
0.00	92.06	7.94	0.00	0.00	**0.00**	0.00
14.55	85.41	0.03	0.00	0.00	0.00	**0.00**

Now comparing the results with the ANN model [1] , we get very similar results , with the model accuracy in predicting 4 of the 7 variables is 0% upto 2 decimal places.
The same data was run on a Bayesian Neural Netork which gave an accuracy of ~54% on the Training set ans ~48-50% on the Test set . Investigating further, we realised that the predictions were relatively unbiased, as hypothesized earlier . Following has been the confusion matrix of the BNN.

Table 3. Percentage correct vaule detected with BNN

Spruce-Fir	Loge.pin	Pond.Pin	Cottonwood	Apen	Doug.Fir	Krummhol
53.28	45.63	0.77	0.00	0.00	0.01	0.25
45.00	**39.40**	12.45	0.27	0.23	0.13	2.50
15.76	10.22	**70.45**	0.66	0.50	0.19	2.18
Rest.[5]	Rest.	Rest.	Rest.	Rest.	Rest.	Rest.
31.65	46.97	20.46	0.00	**0.00**	0.00	0.91
26.02	33.55	20.98	2.66	1.57	**1.63**	13.57
64.25	35.01	0.73	0.00	0.00	0.00	**0.01**

It can be clearly seen that BNN is less biased than the DNN.. It would have been very intereseting to see the even better prediction that the BNN could have provided[6] with the selective sampling , explained in the following sections.

[5] There is no data for the Cottonwood class because it just so happened that all of the Observations with CottonWood class fell under the Training Set while we did the sampling.It is to be noted that the total no of observations in the data set was nearly 500000 while the Cottonwood class had 2747 appearences

[6] The further analysis with the Bayesian Networks could not be done , one of the following reasons is mentioned in the ReadMe file.

3.5 Selective Sampling

Referring to **Figure1** , we have previously talked about the biasness in the data.With selective sampling we aim to reduce the biasnees in the data . To do this , we create 3 different set of Training- Testing Data as follows. Data as follows(In tall the 3 cases the Testing would be simply the compliment.)

Data is randomly selected in pre – decided proportions of the various output classes to obtain a less skewed output class. These predecided proportions can be looked in the sample code provided. Below we give , how the 3 distributions look graphically with there results in the results section.

4 Results

4a Sample1

The Data Distribution obtained throught the Selective sampling is as follows :

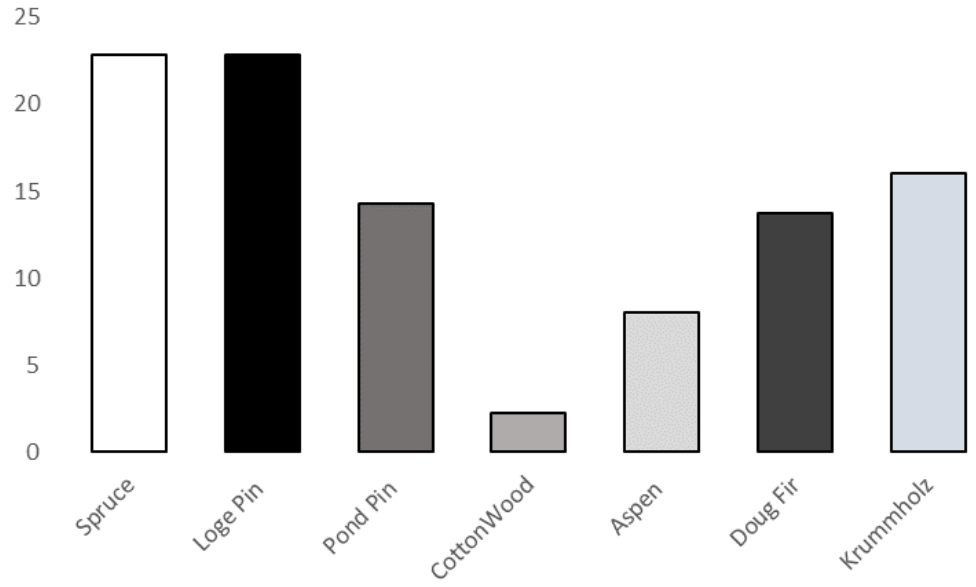

Fig.2. % Distribution of forest cover type, after doing Sampling1

With this data we achieved an accuray of **~55%** on the Training data and **~58%** on the Testing data. The confusion matrix in this case looks like :

Table 4. Percentage correct vaule detected with DNN , using 1st kind of sampling

Spruce-Fir	Loge.pin	Pond.Pin	Cottonwood	Apen	Doug.Fir	Krummhol
82.22	10.23	0.08	0.00	0.00	0.02	7.45
58.24	**37.95**	2.64	0.00	0.00	0.81	0.37
0.43	0.70	**98.39**	0.00	0.00	0.48	0.00
10.05	3.20	83.95	**0.00**	0.00	2.80	0.00
22.67	68.66	7.54	0.00	**0.00**	1.13	0.00
12.90	22.85	53.43	0.00	0.00	**10.83**	0.00
23.41	0.36	0.39	0.00	0.00	0.00	**75.84**

It can be seen that post removing the bias of the data , the DNN model has performed better then it did in the previous case. Comparing the results with the ANN model of the paper [cite your paper] , the model has made significant improvement for the same case.

4b Sample2

The Data Distribution obtained throught the Selective sampling is as follows :

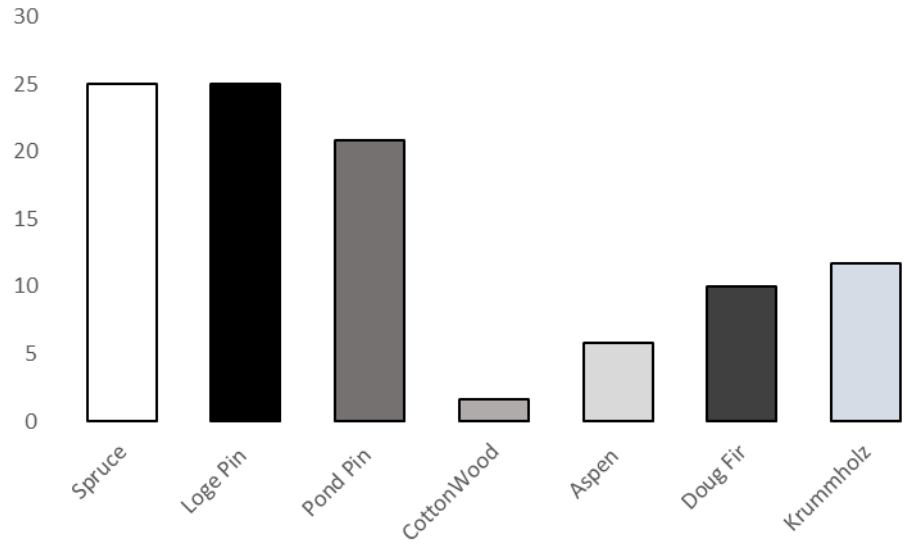

Fig.6. % Distribution of Data

The results in this case turns out to an improved accuacy of **~57%** on the Trainset and an accuracy of about **~50%** on the TestSet. The confusion matrix in this case turns out to be :

Table 4. Percentage correct vaule detected with DNN , using 1st kind of sampling

Spruce-Fir	Loge.pin	Pond.Pin	Cottonwood	Apen	Doug.Fir	Krummhol
45.03	5.83	35.12	0.00	0.00	0.25	13.79
16.60	**52.37**	27.98	0.00	0.00	2.66	0.39
0.02	1.33	**90.33**	0.00	0.00	8.32	0.00
0.00	7.10	90.75	**0.00**	0.00	2.15	0.00
4.69	50.14	40.43	0.00	**0.00**	4.74	0.00
1.72	5.49	74.09	0.00	0.00	**18.70**	0.00
7.62	11.11	6.01	0.00	0.00	0.00	**75.25**

Theoretically this model has been slightly inferior then the previous one, but in practice it might turn out to be more efficient as it has clearly lesser number of false positives and it predicts all the classes better then before,except for one class. We skip here the 3rd model which had near similar details. Further while comparing the results[4] with an existing paper, we find that results are similar in terms of which covertype has been predicted with good accuracy.The overall results of the paper[4] , has been 70% which is much more then we could produce. This can be attributed to many reasons which might include, but not limited to , number of points on which data was tested . This can be a goo reason because in our case, a low training and low testing has done better in trial runs. But this case was avoided, because it involved a lot of information loss. The takeaways are that with good data pre – processing and selective sampling we cas still make the neural networks predict well.

4 Conclusions

The DNN , as compared to the ANN, was able to handle a biased data slightly well .The results further confirmed that the biological inspiration of the Neural Network would be incomplete without encluding the beliefs . We saw that the BNN showed potential and capacity to learn promising on one such data . However if the data is fairly well distributed then the DNNs might offer better prospects in terms of reducing the computational complexity. Further , there are not much resources avaible as of now to utilise variational inferences. The one such used here is derived from the edward library[7], which has its own issues relating to it. It thus becomes very important to be pro – fluent in a language like python to implement ideas in a customised library.

5 Future Work

There is a great potential in utilisng the different distributions as seen in fields as distant as quantum physics and nuclear engineering.It would be interesting to introduce weight decay like the one in nuclean engineering , to the DNN or BNN weights after a certain life cycle of learning based on certain conditions. As discussed in the paper, Gibbs sampling is ones such way of dealing with the calculation of class conditional probablities. It would be interesting to see how the other MCMCM works .The author in the immeditate future, plans to work on specific optimisation techiniques specific to the problem of machine vision. While the data set in hand might differ but a yet another way of looking at things, that we have used in this paper and the previous would be of importance to consider.

[7] More about it in the code files

References

[1] A. Singh, "Bias Handling in ANN with comparison to a Bayesian Based classifier", Assignment, Australian National University, 2018.

[2] J. Schmidhuber, "Who Invented Backpropagation?", *People.idsia.ch*, 2018. [Online]. Available: http://people.idsia.ch/~juergen/who-invented-backpropagation.html. [Accessed: 28- May- 2018].

[3]. M. Nisen, "Why People Often Do The Exact Opposite Of What They're Told", *Business Insider Australia*, 2018. [Online]. Available: https://www.businessinsider.com.au/why-people-dont-follow-directions-2013-8. [Accessed: 28- May- 2018]

[4] J. Blackard and D. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables", Computers and Electronics in Agriculture, vol. 24, no. 3, pp. 131-151, 1999.

[5] M. d'Acremont and P. Bossaerts, "Decision Making: How the Brain Weighs the Evidence", *Current Biology*, vol. 22, no. 18, pp. R808-R810, 2012

[6] R. Bustos and T. Gedeon, "Decrypting Neural Network Data: A GIS case study", Artificial Neural Nets and Genetic Algorithms, pp. 231-234, 1995.

[7] M. Rafiq, G. Bugmann and D. Easterbrook, "Neural network design for engineering applications", *Computers & Structures*, vol. 79, no. 17, pp. 1541-1552, 2001.