International Journal of Scientific Research & Engineering Trends
Volume 7, Issue 5, Sept- Oct-2021, ISSN (Online): 2395-566X

Improving the Performance of Neural Networks

Satwik Ram Kodandaram
AI/NLP Scientist
Information Services Group,
Karnataka, India
E-mail: satwikram29@gmail.com

Abstract- Deep Learning is a sub-part of Machine Learning where we exactly mimic the human brain neural network system. Deep Learning models are nonlinear models. They offer increased flexibility and can scale in proportion to the available training dataset. The downside of this flexibility is weights are calculated and updated via a stochastic training algorithm which means that they are sensitive to the training data and may have a different set of weights upon each time they are trained and produce different predictions. Generally, this case is referred to as neural networks with high variance and it will be very difficult to produce a final model for predictions. Deep Learning models often take too much time to train which means we require high computation resources like GPU or TPU. After investing so much time and resources, there is no guarantee that the final model will have low generalization error when performing on the unseen dataset. To overcome this, we need to reduce the variance of the model. A successful approach to reduce the variance is to go for "ensemble learning". In this paper, we will discuss different methods of "ensemble learning" to improve the accuracy of the deep learning model by reducing the variance.

Keywords- High Variance, Low Bias, Low Generalization, Reduce Variance, Ensemble Learning.

I. INTRODUCTION

The concept of "ensemble" is common in machine learning. Different algorithms are combined for single predictions. The most famous Random Forest algorithm is an example where it combines various multiple decision trees to build a model.

For Deep Learning also we can apply some ensemble techniques [1] [2] to improve the accuracy of the model. An increase in a few percent of the accuracy matters a lot when we have a large dataset.

For Deep Neural Networks we can improve the accuracy by changing the architecture of the model and by hyperparameters tuning that is by increasing/decreasing the number of hidden layers, changing weight initialization, changing activation functions, and trying with different combinations of it and picking the best hyperparameters that perform well on unseen data.

However, trying out different hyperparameters requires a lot of time as the model should be trained with all permutations and combinations. And hyperparameter tuning not always gives the best result because tuning the model also requires good knowledge of the subject. To overcome this, we need to reduce the variance of the model.

A successful approach to reduce the variance is to train multiple sub-models instead of a single model and combine all of them to predict a single prediction. This is called ensemble learning and this not only reduces the variance of the predictions but also can result in predictions that are better than a single model prediction. Different "ensemble" techniques can be implemented on Deep Neural Networks and improve the performance of the model.

The advantages of using ensemble learning are as follows:
- Improved Performance
- Improved endurance
- Can Build Robust Model
- Better Generalization compared to Single model

II. ENSEMBLE LEARNING

Suppose we ask a complex question to thousands of random people, and then aggregate their answers. In many cases, we will find that this aggregated answer is better than the expert's answer. This is called "Wisdom of Crowd".

Similarly, if we aggregate the predictions of a group of predictors such as classifier or regressor, we will often get better predictions than the individual best predictor. A group of predictors is called ensemble; thus, this technique is called Ensemble Learning, and an Ensemble Learning algorithm is called as Ensemble method.

An example of Ensemble learning in Machine Learning is the Random Forest algorithm in which trains a group of

Decision Tree Classifiers, each on a different random subset of training set.

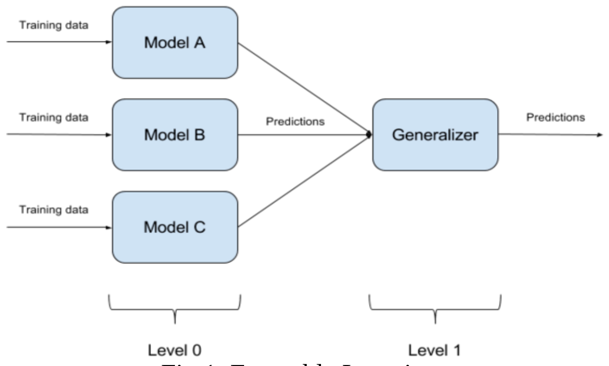


Fig 1. Ensemble Learning.

In Deep Learning we have different ensemble methods that can be implemented to improve the performance of the neural networks. Some of them are discussed in this paper.

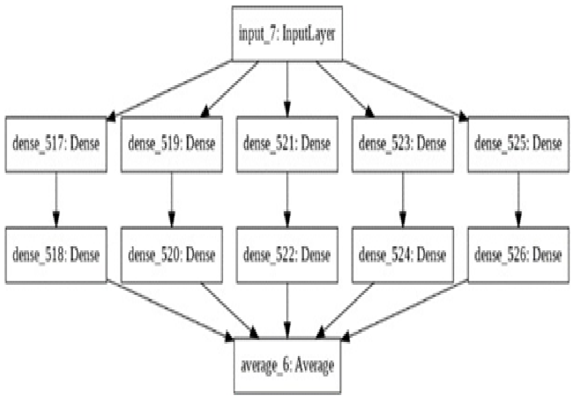


Fig 2. Neural Networks.

III. STACKING GENERALIZATION ENSEMBLE FOR DEEP LEARNING

A model averaging is an ensemble technique where it combines the prediction of multiple trained sub- models to a single prediction. Here all the sub-model is contributed equally to the combined single prediction. A limitation of this approach is here we are just combining the predictions of each sub- model regardless of its performance.

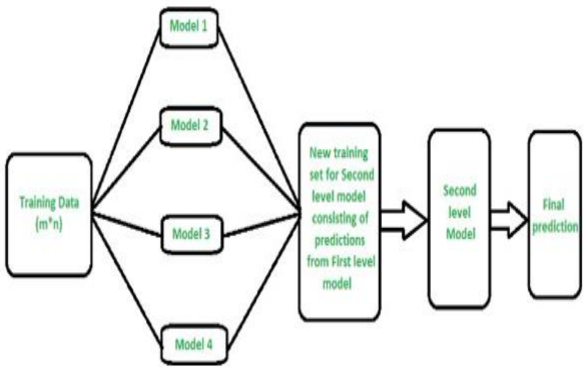
A slight variation of this version that is averaging the weights of the sub-models is called the weighted average ensemble. The weighted average is a technique, where we will have multiple sub-models which were trained on some train dataset and we average these models' weights to get single predictions.

This technique is called Model averaging. This technique can be improved by taking more weights by the model which is performing well on unseen data.

In simple terms, we take more contribution from the well-performing model on unseen data and less contribution from the model which is not performing well on unseen data. This method would significantly improve the accuracy of the model across test data.

A further approach to this is we can build a learning algorithm that can learn how to take the contributions of each trained sub-models. This technique is called Stacking Generalization or simply stacking [3] and can result in better prediction compared to single sub-model prediction.

This is something we train a completely new model to take best contributions from each sub-model. This method will reduce the variance of the model as we train the model to take contributions and thus increase the accuracy over the unseen data.


Model 1
Model 2
Model 3
Model 4
Training Data (m*n)
New training set for Second level model consisting of predictions from First level model
Second level Model
Final prediction
Fig 3. Stacking.

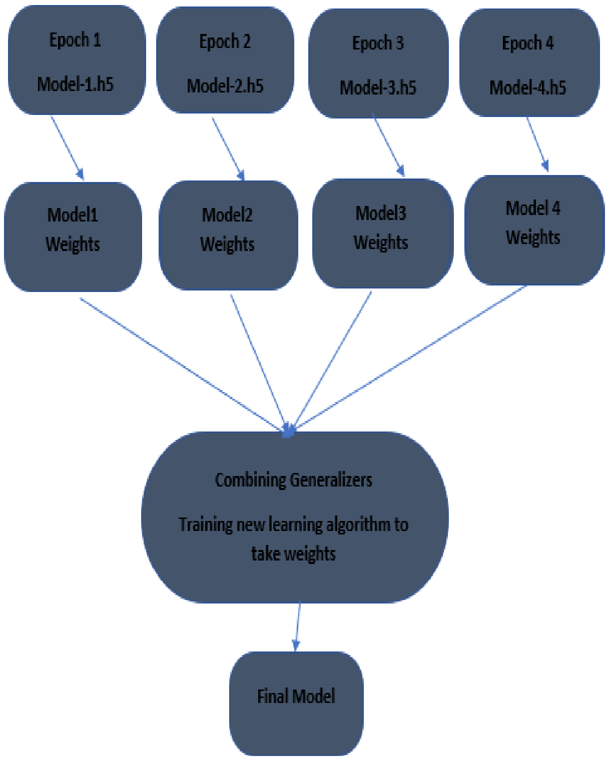


Fig 4. Stacking Generalization.

Let X be the independent variable and Y be the target variable or labels

Let D = {(X, Y)} be the training dataset given to the base-models.

Let B be the base models and M be the Meta models

Algorithm 1: Stacking Generalization
Input: Training dataset D = {(X,Y)}
Output: prediction – Stacked generalized prediction from the model

```
• Step 1: Train the base-level models
• for each t in Train
• Train a base model Bt ∈ B based on D
• end for
• Step 2: Train the meta-level models by
• for each t in Train
• Train a meta-model Mt ∈ M by giving the base models B as the input
• end for
• Step 3: Obtain the stacked ensemble prediction and return it.
• Step 4: return prediction.
```

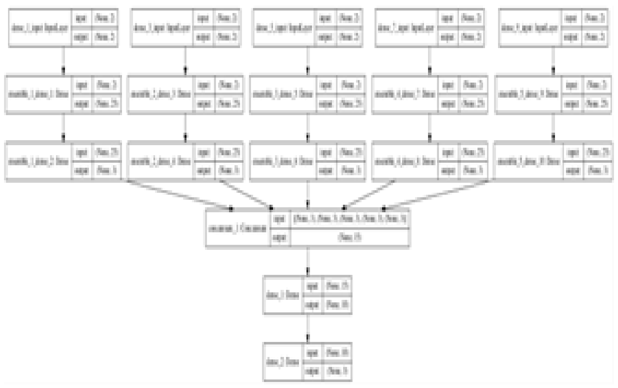


Fig 5. Stacking Generalization Visualization.

IV. HORIZONTAL FOR CLASSIFICATION PROBLEMS

In Deep learning modeling problems, in most of the case, it is very tough to improve the performance of the model. Also, if we train for some number of epochs model often will have high variance.

It is very challenging to decide which model is performing well on unseen data at the end of training and we don't have clarity also which model will do well on all the cases.

Horizontal voting is one of the best ensemble methods to overcome this problem. Suppose if we train a model for 100 epochs, save the last 20 models. These last 20 models can be ensembled to get a single prediction. In simple terms, horizontal voting is a technique where we take votes from each sub-model and finalize the class which got the maximum votes.

This final ensemble prediction will do better compared to the randomly saved model. This method can be implemented when we have less data and can significantly improve the performance of the model. One key thing here is, we have to use an odd number of sub-models as we should not get an even number of votes for the class which will lead to a tie.

Let epoch be the number of iterations to train the model where e ∈ E. Let threshold be the threshold value greater than which the corresponding model will be saved.

Let model be the neural network models being trained on the training data Train (X, Y) such that m ∈ model.

Let 'save' be the function for saving the module in the location 'models/' directory.

Algorithm 2: Horizontal saving.
Input: threshold and epoch
Output: The models file with .h5 as an extension after the threshold epochs will be saved in the directory

```
• Step 1: Create a class that inherits the TensorFlow call back module
• Step 2: Save model after each epoch in the h5 format
• for each e in epoch
• if epoch > = threshold, then
• filename ← "model"+str(e)+".h5"
• save(filename)
• end for
• Step 3: end
```

Here we don't save the model after each epoch, because initially the model will have high variance and the loss will be more. As the model is trained for certain epochs, the loss function will go on decreases. So, after a certain threshold, the model will be saved. For example, if we are training the model for 500 epochs, saving all the models at each epoch will lead to 500 different models.

The models say the first 100-200 model performance will be less compared to the last 20 models. So, we can save the last 20 or 50 models so that these models will have less variance compared to early models and on ensemble, the model averaging or voting can give the best result compared to the random saved model.

Here in the example for 500 epochs, the threshold is 480. We are saving the models with epochs greater than or equal to 480 for 500 epochs. Ensemble of last 50 models will give good results compared to the randomly saved model.

Let model be the neural network models being trained on the training data Train (X, Y), such that m ∈ model

Let all_models be a list containing all the models. The location 'models/' directory.

Algorithm 3: Loading all the saved models

```
• Input: the models present in the location 'models/' to all_models
• Output: list of all the models
• Step 1: Initialize all_models as an empty list
• all_models = [ ]
• Step 2: Append all the models present in the location 'models/' to all_models
• Step 3: for each m in model, do
• all_models ← append(m)
• Step 4: return all_models
```

Let model be the set of neural network models being trained on the training data (X, Y), such that m ∈ model. Let yprediction be the predictions obtained by all the models on the test data Train (X', Y').

Let 'array' be the function for converting lists to arrays

Algorithm 4: Horizontal Voting for classification problem

```
• Input: models, test set Test' (X ', Y '), and empty y prediction list
• Output: predictions – final prediction obtained
• Step 1: Obtain the predictions of each model
• for each i in range(X)
• for each model in models, do
• y prediction [i] ← predict(X[i])
• Calculate highest number of votes for ith test data and append
• Y prediction [i] ← highest voted class
• end for
• end for
• Step 2: Convert list into an array
• Y prediction ← array(y prediction)
• Step 3: return y prediction
```

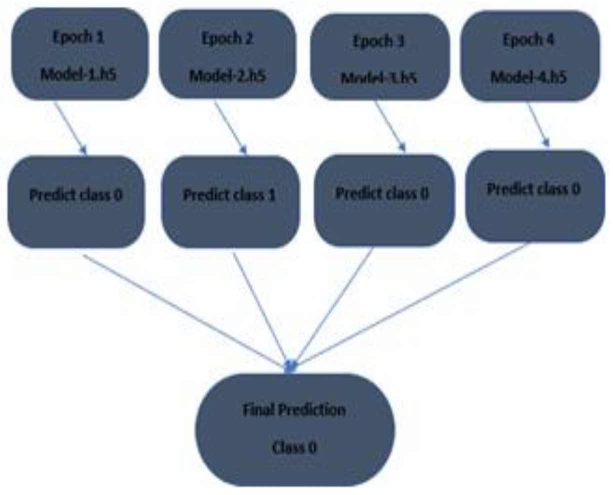


Fig 6. Horizontal voting for Classification.

Here is an example of a classification problem. In the above example we can see, we have saved the model after every epoch and we are loading the model. Each model will give its predictions; we can see 4 models with its prediction. 3 models are predicting with class 0 and 1 model prediction with class 1. With Horizontal Voting, the final prediction is class 0 as it has more votes.

V. HORIZONTAL SAVING AND MODEL AVERAGING FOR REGRESSION PROBLEMS

Model averaging is an ensemble technique where we take each model's weights for prediction. Each model will be equally contributed to the final prediction. In the regression problem, we will predict the continuous value.

Here model will be trained and saved as usually mentioned above based on certain epochs. We are not saving the model for all the epochs because, at the beginning epoch, the mean squared error and mean the absolute error will be high.

So, saving the model for each and taking it for prediction and averaging it, will have high error as initially, epochs models will have high mean squared and mean absolute error.

On averaging these models, the final prediction will be affected by these models, so we will save the last 50-100 epochs so that on averaging our prediction will be best compared to the randomly saved model.

Let model be the set of neural network models being trained on the training data Train (X, Y), such that m ∈ model. Let yprediction be the predictions obtained by all the models on the test data Test (X, Y).

Let 'array' be the function for converting lists to arrays

Algorithm 5: Horizontal Voting for a regression problem

```
• Input: models, test set Test (X , Y)
• Output: predictions – final prediction obtained
• Step 1: Load all the saved models and obtain the predictions of each model and append into the list yprediction
• for each model in models, do yprediction ← predict(X) yprediction ← append(yprediction)
• end for
• Step 2: Convert list into an array
• yprediction ← array(yprediction)
• Step 3: Find the sum of the arrays
• result ← sum(yprediction)
• Step 4: Calculate the average of each prediction and return the average
• average ← (result)/(total number of models)
• Step 5: return average
```

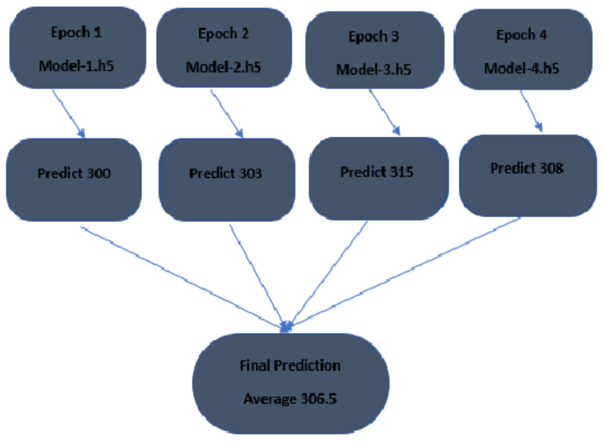


Fig 7. Horizontal voting for Regression.

VI. HORIZONTAL VOTING DEEP LEARNING ENSEMBLE WITH DIFFERENT MODELS

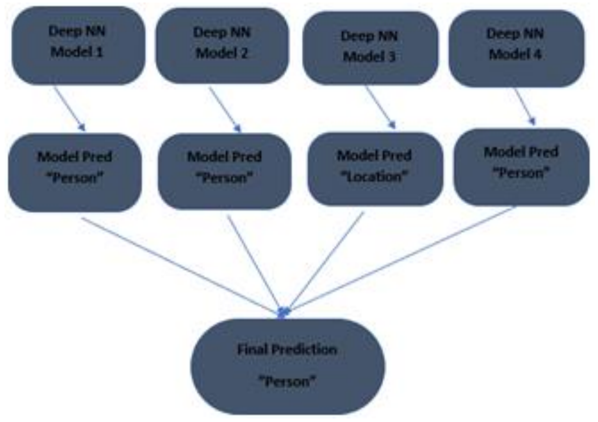


Fig 8. Horizontal voting with different models.

Horizontal voting with different models is the same as the above-explained ensemble technique. In the above technique, we have saved models after every epoch and took a final prediction with more votes. Here 90% of the concepts remain the same but here instead of saving the model with each epoch and loading to predict and taking votes, we train multiple Deep Neural Networks and combine the prediction of all the models and make a final prediction based on the maximum votes.

VII. EXPERIMENTAL STUDY

We have taken some common machine learning and deep learning datasets to test the ensemble learning algorithms. The result we found out was amazing. Without any hyper-parameters tuning etc; we could able to achieve good benchmark results.

For regression problems we tested the ensemble algorithms with the metrics mean squared error (mse), root mean squared error (rmse), mean absolute error (mae) and, r2 score. For classification problems, we tested the ensemble algorithms with the metrics accuracy, precision, recall, and f1 score.Some of the key observations made during the experimental study are:

- Neural networks are non-linear and have high variance.
- This can be frustrating while preparing the final model for predictions.
- Ensemble learning methods combine the model weights from different models to reduce the variance of predictions.
- Ensemble learning can also reduce generalization error.

Table 1. Horizontal Voting

Models	Precision	Recall	F1 Score	Accuracy
Random	0.68	0.61	0.64	0.83
Best 1	0.78	50.59	0.67	0.84
Best 2	0.76	0.60	0.67	0.85
Ensemble	0.78	0.61	0.68	0.87

Table 2. horizontal Voting (Averaging)

Models	MSE	RMSE	MAE	R2 Score
Random	2989449448.8	54675.8580	35326.6222	0.76629
Best	2889449448.8	53753.599	33326.6222	0.76.21
Ensemble	2748419021.2	52425.3662	322236.66	0.78514

Table (1) shows the experimental result on the classification dataset. The random model is the model which was saved using the model. Save () method using TensorFlow. The random model is not the best. Best 1 model is the model which was saved using callbacks. This model is the best model through the epochs on which the model was trained, which means it has the low validation loss and highest validation accuracy.

The best 2 model is another model with different neural network architecture and this was saved using callbacks with low validation loss and highest validation accuracy.
Here all 3 models came out with accuracy score around 83% to 85%.

On using the horizontal voting ensemble method for these 3 models, the accuracy of the model was jumped to 87 %. Although a 2% increase looks very low in reality, it is a very good improvement when we deal with large datasets. Also, to make a note, both recall and f1 scores improved.

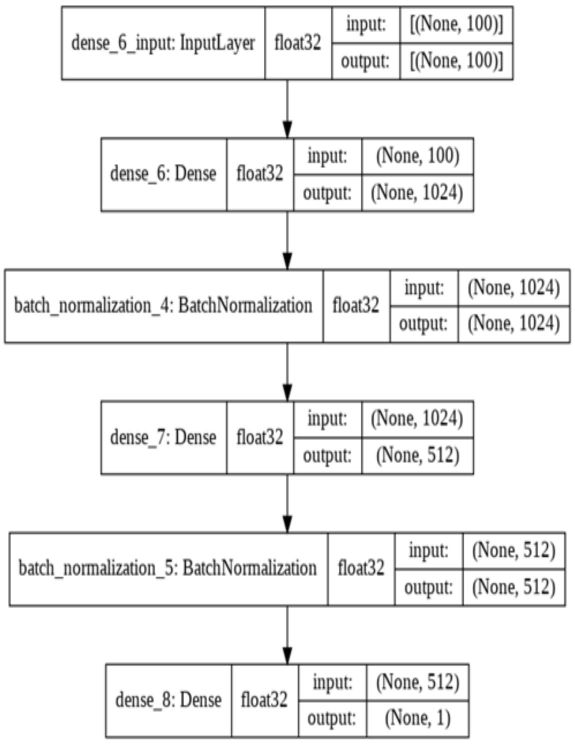
Table (2) shows the experimental result on the regression dataset. Both random and best models were saved using the same neural network architecture. During the last 10 to 20 epochs, the model was saved at every epoch. And these models were used for horizontal averaging. The final model after averaging these 10 to 20 models had a low mean squared error and high R2 score which was found out to be 2748419021.1 and 0.78 respectively.

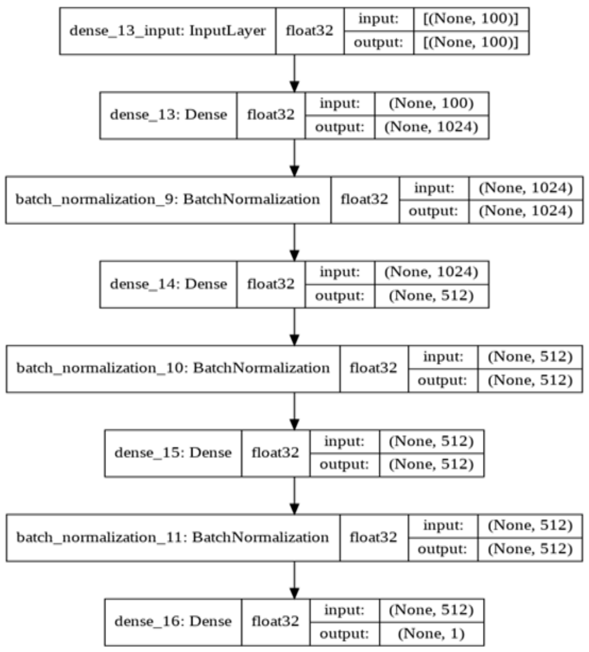
Table 3. Stacking Generalization

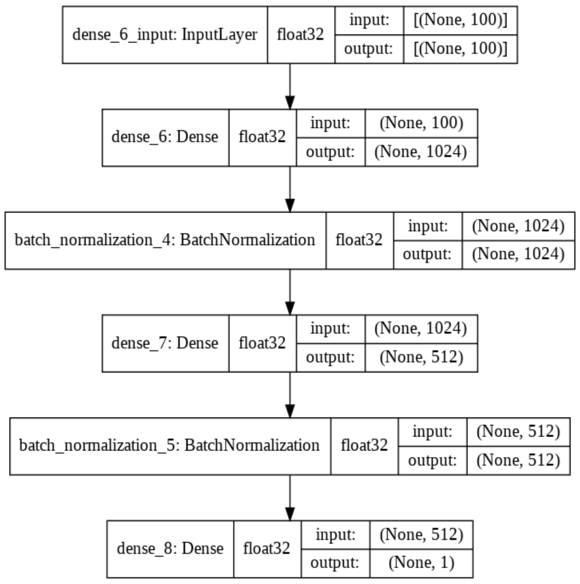
Models	Accuracy
Model 1	0.817
Model 2	0.819
Stacked Model	0.83

Table (3) shows the experimental result on a classification dataset. Model 1 and Model 2 were the two different neural network architectures that were trained with an accuracy of 81.7% and 81.9 % respectively.

Using the Stacking generalization ensemble method these two models were stacked and the accuracy of this model was jumped to 83 %. There are many ways in improving the performance of neural networks. Ensemble technique is one of them.Here are the snapshots of few different Neural Networks architecture used for ensemble.


dense_6_input: InputLayer | float32 | input: [(None, 100)] | output: [(None, 100)]
dense_6: Dense | float32 | input: (None, 100) | output: (None, 1024)
batch_normalization_4: BatchNormalization | float32 | input: (None, 1024) | output: (None, 1024)
dense_7: Dense | float32 | input: (None, 1024) | output: (None, 512)
batch_normalization_5: BatchNormalization | float32 | input: (None, 512) | output: (None, 512)
dense_8: Dense | float32 | input: (None, 512) | output: (None, 1)
Fig 9. Neural Architecture 1.


dense_13_input: InputLayer | float32 | input: [(None, 100)] | output: [(None, 100)]
dense_13: Dense | float32 | input: (None, 100) | output: (None, 1024)
batch_normalization_9: BatchNormalization | float32 | input: (None, 1024) | output: (None, 1024)
dense_14: Dense | float32 | input: (None, 1024) | output: (None, 512)
batch_normalization_10: BatchNormalization | float32 | input: (None, 512) | output: (None, 512)
dense_15: Dense | float32 | input: (None, 512) | output: (None, 512)
batch_normalization_11: BatchNormalization | float32 | input: (None, 512) | output: (None, 512)
dense_16: Dense | float32 | input: (None, 512) | output: (None, 1)
Fig 10. Neural Architecture 2.


dense_6_input: InputLayer | float32 | input: [(None, 100)] | output: [(None, 100)]
dense_6: Dense | float32 | input: (None, 100) | output: (None, 1024)
batch_normalization_4: BatchNormalization | float32 | input: (None, 1024) | output: (None, 1024)
dense_7: Dense | float32 | input: (None, 1024) | output: (None, 512)
batch_normalization_5: BatchNormalization | float32 | input: (None, 512) | output: (None, 512)
dense_8: Dense | float32 | input: (None, 512) | output: (None, 1)
Fig 11. Neural Architecture 3.

VIII. CONCLUSION

Neural Networks is a concept where it exactly replicates our human brain's neural network system. It is often very difficult to build State of the Art (SOTA) models. We have so many different methods to improve the performance [4] [5].

This might take time to come to a conclusion which method would work out for the particular problem. But for ensemble suppose if we have 2 to 3 models performing decently on the unseen dataset, the ensemble method can improve the performance of the model on an unseen dataset.

And if we don't have 2 to 3 models, we can save the model at each epoch and ensemble it. In this, we should different methods of ensemble techniques that can improve the performance, yet there is still research going on and better algorithms can be applied and build a strong generalizer.

REFERENCES

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, Oct. 1990, doi: 10.1109/34.58871.
[2] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks, Volume 12, Issue 10,1999, Pages 1399-1404, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(99)00073-8.
[3] David H. Wolpert, Stacked generalization, Neural Networks, Volume 5, Issue 2, 1992, Pages 241-259, ISSN0893-6080, https://doi.org/10.1016/S0893-6080(05)80023-1.
[4] MacKay D.J.C. (1995) Developments in Probabilistic Modelling with Neural Networks — Ensemble Learning. In: Kappen B., Gielen S. (eds) Neural Networks: Artificial Intelligence and Industrial Applications. Springer, London. https://doi.org/ 10.1007/978-1-4471-3087-1_37
[5] Polikar R. (2012) Ensemble Learning. In: Zhang C., Ma Y. (eds) Ensemble Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978- 1-4419-9326-7_1