TO DO:

Weight Initialization

Entering the initial state of all the weights in the neural network. Initial weights are used to prevent the exploding / vanishing gradient problem. The exploding / vanishing gradient problem causes the neural network to update with too small / too large of values to effectively train the network. They used an N(0, 0.1) distribution in the first layer, N(0, 0.05) in all hidden layers, and N(0, 0.001) in the output layer. Since we recommend a ReLU activation function, we recommend using Kaiming Initialization. This initialization will help avoid the exploding / vanishing gradient problem, and may increase the speed in which the neural network trains.

Activation Functions

An activation function is an equation that determines the value of a neuron using the previous layer’s values. The activation function itself is a transformation that takes (input \* neuron weight + bias) of the previous layer, inputs that into the function, and uses the output of the function as the “activation” of the neuron. This “activation” is used as input for the next layer in the network, or is the output of the network itself if the layer that the activation occurs at is the output layer. They use a tanh activation function for hidden layers and sigmoid for the output layer. We recommend using Leaky ReLU instead. Problems with the sigmoid and tanh functions include the vanishing gradient and that they are computationally intensive. Leaky ReLU is more computationally efficient than sigmoid and tanh, which would result in faster training of the model. We recommend Leaky ReLU instead of regular ReLU as some inputs are close to 0 and some are negative, which is not compatable with regular ReLU.

Neuron / Hidden Layer Count

Increasing neurons would increase the number of neurons in a specific hidden layer. Increasing the number of hidden layers would increase the number of layers between the input and output layer. Increasing hidden layers and neurons would allow for more non-linear behavior to be modeled by the neural network. They use 300 neurons per hidden layer and four hidden layers. Although they mentioned in the paper that adding more hidden layers and neurons increased training time without noticeably increasing performance, we recommend adding more hidden layers and hidden neurons. Adding extra hidden neurons and hidden layers could discover more non-linear trends in the data when paired with the other changes that we recommend. These extra trends discovered by the neural network could potentially increase model accuracy and allow for even more accurate predictions. Advancements in computational power since this paper was written could reduce the fears of extra computational time.

Learning Rate

The learning rate controls how much the weights of the network should change with each iteration. The learning rate controls the trade off between convergence speed and overshooting the target. A learning rate that is too large will overshoot equilibrium, while a learning rate that is too small will take an extremely long time to reach equilibrium. They used an initial learning weight of 0.05, decaying by a factor of 1.0000002 each batch update until a minimum of 10^-6. We recommend using an adaptive gradient descent algorithm such as Adam. This algorithm stores the past gradients and past momentums and takes the weighted average (more recent = larger weight) in order to get a more appropriate learning rate for the data. Larger changes in the past result in larger changes in the present, while smaller changes in the past result in smaller changes in the present. This could cause the network to converge faster, without risking overshooting its target.

Batch Size

The batch size is the number of samples that will be used to calculate the error gradient. The batch size controls the accuracy of the “step” the neural network takes to minimize the loss function. The larger the batch size, the more accurate the step, but the longer the network takes per step. They use a batch size of 100, and we recommend doubling the batch size to 200. With modern computational advancements, we have fewer concerns about speed per batch, so increasing the batch size could result in a more accurate gradient, which could result in higher accuracy and quicker training.

Most Useful Evaluation Metric

In comparing their models, they use the area under the curve (AUC). This metric gives us a sense of how well our model performs in terms of Background Rejection and Signal Efficiency, or how well our model does at avoiding false positives and hitting true positives. In order to directly compare our models to theirs, we should also use this metric. It wouldn’t make sense to score our models using a metric that they didn’t use; we want an apples-to-apples comparison. They also list the standard deviation of the AUC, so we should do the same.

Since they use discovery significance to show that small increases can be the result of large increases in discovery significance, we don’t need to calculate this as well. Discovery significance is inherent in AUC increases.

1) Weight Initialization

a) What is it?

I) Entering the initial state of all the weights in the neural network.

b) What does it do?

I) Initial weights are used to prevent the exploding / vanishing gradient problem. The exploding / vanishing gradient problem causes the neural network to update with too small / too large of values to effectively train the network.

c) What do they use?

I) They used an N(0, 0.1) distribution in the first layer, N(0, 0.05) in all hidden layers, and N(0, 0.001) in the output layer.

d) What do we recommend?

I) Since we recommend a ReLU activation function, we recommend using Kaiming Initialization.

e) How might it be impactful?

I) This initialization will help avoid the exploding / vanishing gradient problem, and may increase the speed in which the neural network trains.

An activation function is an equation that determines the value of a neuron using the previous layer’s values. The activation function itself is a transformation that takes (input \* neuron weight + bias) of the previous layer, inputs that into the function, and uses the output of the function as the “activation” of the neuron. This “activation” is used as input for the next layer in the network, or is the output of the network itself if the layer that the activation occurs at is the output layer. They use a tanh activation function for hidden layers and sigmoid for the output layer. We recommend using Leaky ReLU instead. Problems with the sigmoid and tanh functions include the vanishing gradient and that they are computationally intensive. Leaky ReLU is more computationally efficient than sigmoid and tanh, which would result in faster training of the model. We recommend Leaky ReLU instead of regular ReLU as some inputs are close to 0 and some are negative, which is not compatable with regular ReLU.

2) Activation Functions

a) What is it?

I) An activation function is an equation that determines the value of a neuron using the previous layer’s values.

b) What does it do?

I) The activation function itself is a transformation that takes (input \* neuron weight + bias) of the previous layer, inputs that into the function, and uses the output of the function as the “activation” of the neuron. This “activation” is used as input for the next layer in the network, or is the output of the network itself if the layer that the activation occurs at is the output layer.

c) What do they use?

I) They use a tanh activation function for hidden layers and sigmoid for the output layer.

d) What do we recommend?

I) We recommend using Leaky ReLU instead.

e) How might it be impactful?

I) Problems with the sigmoid and tanh functions include the vanishing gradient and that they are computationally intensive. Leaky ReLU is more computationally efficient than sigmoid and tanh, which would result in faster training of the model. We recommend Leaky ReLU instead of regular ReLU as some inputs are close to 0 and some are negative, which is not compatable with regular ReLU.

Increasing neurons would increase the number of neurons in a specific hidden layer. Increasing the number of hidden layers would increase the number of layers between the input and output layer. Increasing hidden layers and neurons would allow for more non-linear behavior to be modeled by the neural network. They use 300 neurons per hidden layer and four hidden layers. Although they mentioned in the paper that adding more hidden layers and neurons increased training time without noticeably increasing performance, we recommend adding more hidden layers and hidden neurons. Adding extra hidden neurons and hidden layers could discover more non-linear trends in the data when paired with the other changes that we recommend. These extra trends discovered by the neural network could potentially increase model accuracy and allow for even more accurate predictions. Advancements in computational power since this paper was written could reduce the fears of extra computational time.

3) Increased neurons / hidden layers

a) What is it?

I) Increasing neurons would increase the number of neurons in a specific hidden layer. Increasing the number of hidden layers would increase the number of layers between the input and output layer.

b) What does it do?

I) Increasing hidden layers and neurons would allow for more non-linear behavior to be modeled by the neural network

c) What do they use?

I) They use 300 neurons per hidden layer and four hidden layers.

d) What do we recommend?

I) Although they mentioned in the paper that adding more hidden layers and neurons increased training time without noticeably increasing performance, we recommend adding more hidden layers and hidden neurons.

e) How might it be impactful?

I) Adding extra hidden neurons and hidden layers could discover more non-linear trends in the data when paired with the other changes that we recommend. These extra trends discovered by the neural network could potentially increase model accuracy and allow for even more accurate predictions. Advancements in computational power since this paper was written could reduce the fears of extra computational time.

4) Normalize / Scale Data

a) What is it?

I) Normalizing / scaling data brings all the data onto the same scale.

b) What does it do?

I) Since big numbers typically result in larger weights, it reduces the

c) What do they use?

I) Standardized training / test set so variables were N(0, 1) except for features without negative values, which were scaled to N(1,1).

d) What do we recommend?

I) We recommend normalizing based on the training set, then applying that normalization to the test set.

e) How might it be impactful?

I) The way that the paper is worded, it seems like they created the standardizer using both the training and testing data set. This could result in data leakage, which could artificially increase accuracy by allowing information about the training data set to leak into the testing data set. This could result in the model performing better for the training and testing set than it would with a data set that it has never seen before.

The learning rate controls how much the weights of the network should change with each iteration. The learning rate controls the trade off between convergence speed and overshooting the target. A learning rate that is too large will overshoot equilibrium, while a learning rate that is too small will take an extremely long time to reach equilibrium. They used an initial learning weight of 0.05, decaying by a factor of 1.0000002 each batch update until a minimum of 10^-6. We recommend using an adaptive gradient descent algorithm such as Adam. This algorithm stores the past gradients and past momentums and takes the weighted average (more recent = larger weight) in order to get a more appropriate learning rate for the data. Larger changes in the past result in larger changes in the present, while smaller changes in the past result in smaller changes in the present. This could cause the network to converge faster, without risking overshooting its target.

5) Learning Rate

a) What is it?

I) The learning rate controls how much the weights of the network should change with each iteration.

b) What does it do?

I) The learning rate controls the trade off between convergence speed and overshooting the target. A learning rate that is too large will overshoot equilibrium, while a learning rate that is too small will take an extremely long time to reach equilibrium.

c) What do they use?

I) Initial learning weight of 0.05, decaying by a factor of 1.0000002 each batch update until a minimum of 10^-6.

d) What do we recommend?

I) We recommend using an adaptive gradient descent algorithm such as Adam.

e) How might it be impactful?

I) This algorithm stores the past gradients and past momentums and takes the weighted average (more recent = larger weight) in order to get a more appropriate learning rate for the data. Larger changes in the past result in larger changes in the present, while smaller changes in the past result in smaller changes in the present. This could cause the network to converge faster, without risking overshooting its target.

6) Ensemble

a) What is it?

I) An ensemble model is a model that combines the results of multiple models to create a single prediction.

b) What does it do?

I) There are different ways to ensemble models: boosting, bagging, Bayes optimal classifier, among others. We recommend bagging, where each model has a single vote: either “not Higgs,” or “Higgs.”

c) What do they use?

I) None. They create 3 models, but compare them to each other. There is no ensembling in this paper.

d) What do we recommend?

I) Since we recommend creating new models with different activation functions, batch sizes, etc, it might be worth looking into taking the average of the outputs of these models in order to create a single super model. Since we are doing classification, we could create a model that takes the “votes” from each model, and if the number of “Higgs” is above a certain threshold, we classify that point as “Higgs.”

e) How might it be impactful?

I) An ensemble model could create a “best of both worlds” scenario where a neural network fit using certain parameters may perform better for certain data and worse for others. Ensembling could increase accuracy by bringing in insight from multiple models.

The batch size is the number of samples that will be used to calculate the error gradient. The batch size controls the accuracy of the “step” the neural network takes to minimize the loss function. The larger the batch size, the more accurate the step, but the longer the network takes per step. They use a batch size of 100, and we recommend doubling the batch size to 200. With modern computational advancements, we have fewer concerns about speed per batch, so increasing the batch size could result in a more accurate gradient, which could result in higher accuracy and quicker training.

7) Batch size

a) What is it?

I) The batch size is the number of samples that will be used to calculate the error gradient.

b) What does it do?

I) The batch size controls the accuracy of the “step” the neural network takes to minimize the loss function. The larger the batch size, the more accurate the step, but the longer the network takes per step.

c) What do they use?

I) 100

d) What do we recommend?

I)

e) How might it be impactful?

I) With modern computational advancements, we have fewer concerns about speed per batch, so increasing the batch size could result in a more accurate gradient, which could result in higher accuracy and quicker training.

8) What evaluation metric would the most useful?

a) AUC

I) In comparing their models, they use the area under the curve (AUC). This metric gives us a sense of how well our model performs in terms of Background Rejection and Signal Efficiency, or how well our model does at avoiding false positives and hitting true positives. In order to directly compare our models to theirs, we should also use this metric. It wouldn’t make sense to score our models using a metric that they didn’t use; we want an apples-to-apples comparison. They also list the standard deviation of the AUC, so we should do the same.

b) Discovery Significance

I) Since they use discovery significance to show that small increases can be the result of large increases in discovery significance, we don’t need to calculate this as well. Discovery significance is inherent in AUC increases.