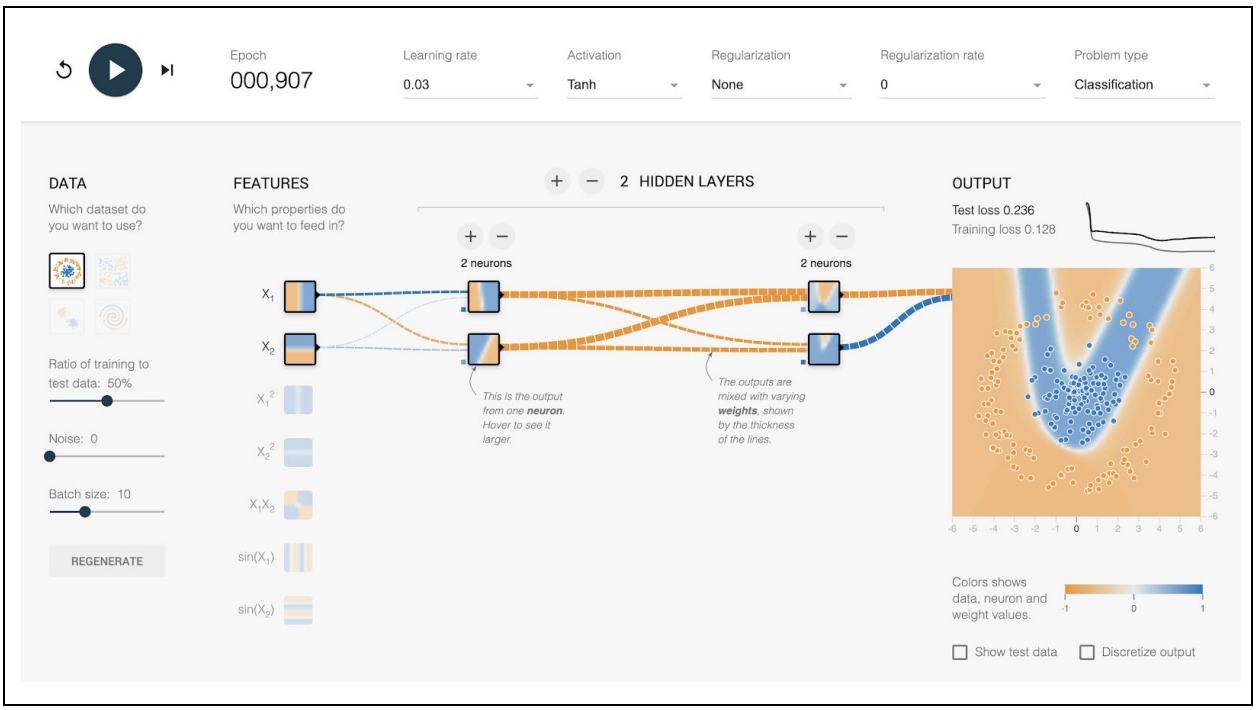
CS 6886: Systems Engineering for Deep Learning

Mini Assignment 2

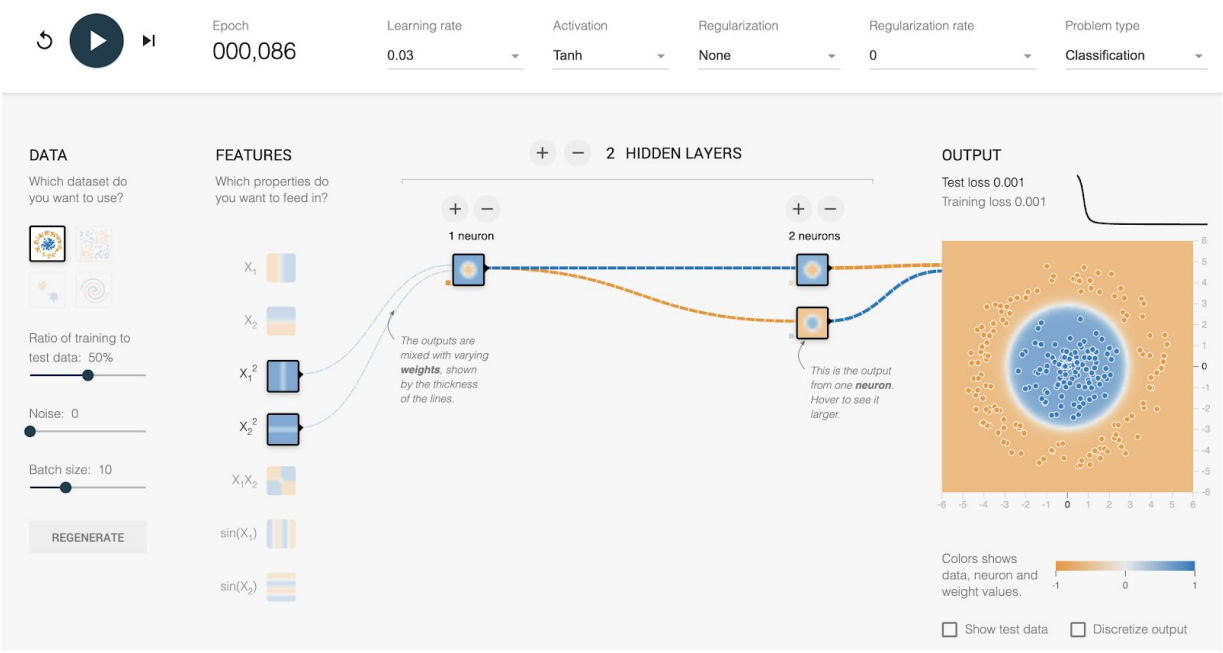
Total Marks: 5

- This assignment is based on the Neural Playground discussed in class.
- In each of the following questions, you are required to illustrate the phenomenon mentioned in the question with configurations of the Neural Playground.
- In each case, you would be required to provide at least *two* examples to contrast scenarios which illustrate the phenomenon. You are welcome to share more examples if they are required to make your point.
- You are required to share a screenshot of the examples in the document (the screenshot should be of only the playground and not of your entire screen).
- You are also required to link the complete URL of the Neural Playground (which will contain all the configurations required to replicate your results)
- You are also required to write a couple of sentences explaining why the examples you have chosen illustrate the phenomenon mentioned in the question.
- An example is provided as Q0 which was covered in the lecture.
- Best 5 out of the 6 will be graded.

Q0. 'Careful feature selection improves accuracy'



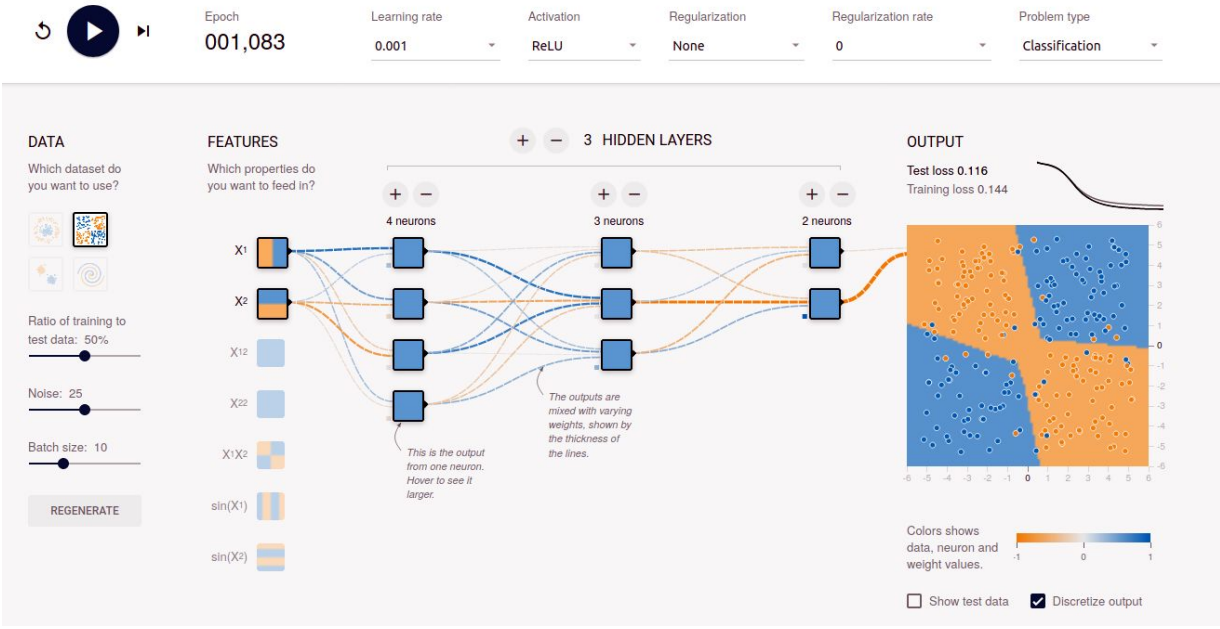
https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=2,2&seed=0.31796&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false



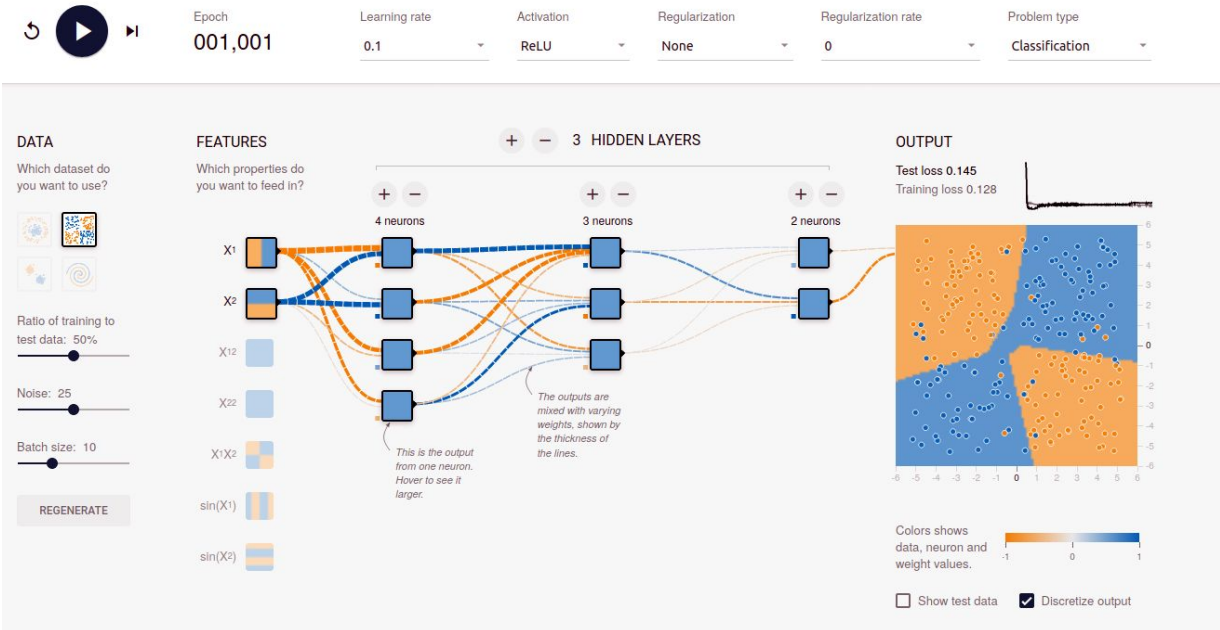
https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=1,2&seed=0.31796&showTestData=false&discretize=false&percTrainData=50&x=false&y=false&xTimesY=false&xSquared=true&ySquared=true&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

In the first case, with features X_1 and X_2, even after 900 epochs we have a substantial training loss. But in the second case, with features X_1^2 and X_2^2, the training loss is much lesser as early as epoch 86 with just a single neuron in the first hidden layer. This is because the features selected in the second case are better suited to the decision boundary (which is a circle).

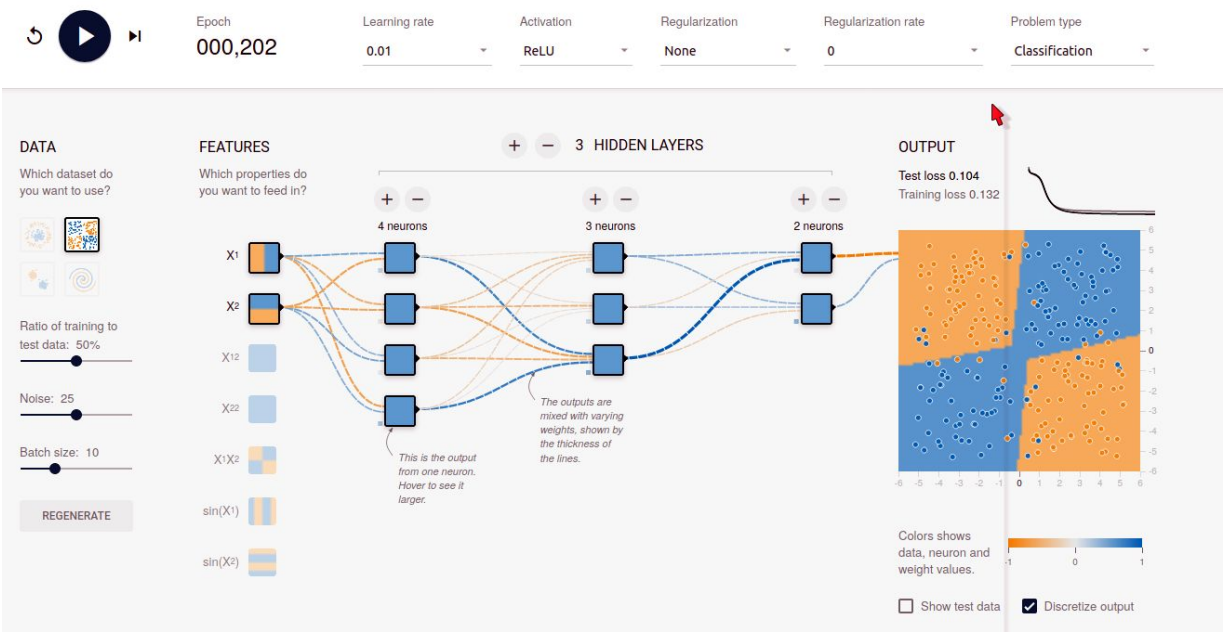
Q1. ‘Learning rate needs to be sensitively changed to avoid very slow learning or divergence’



https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.001&regularizationRate=0&noise=25&networkShape=4,3,2&seed=0.04536&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false



https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.1&regularizationRate=0&noise=25&networkShape=4,3,2&seed=0.04536&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

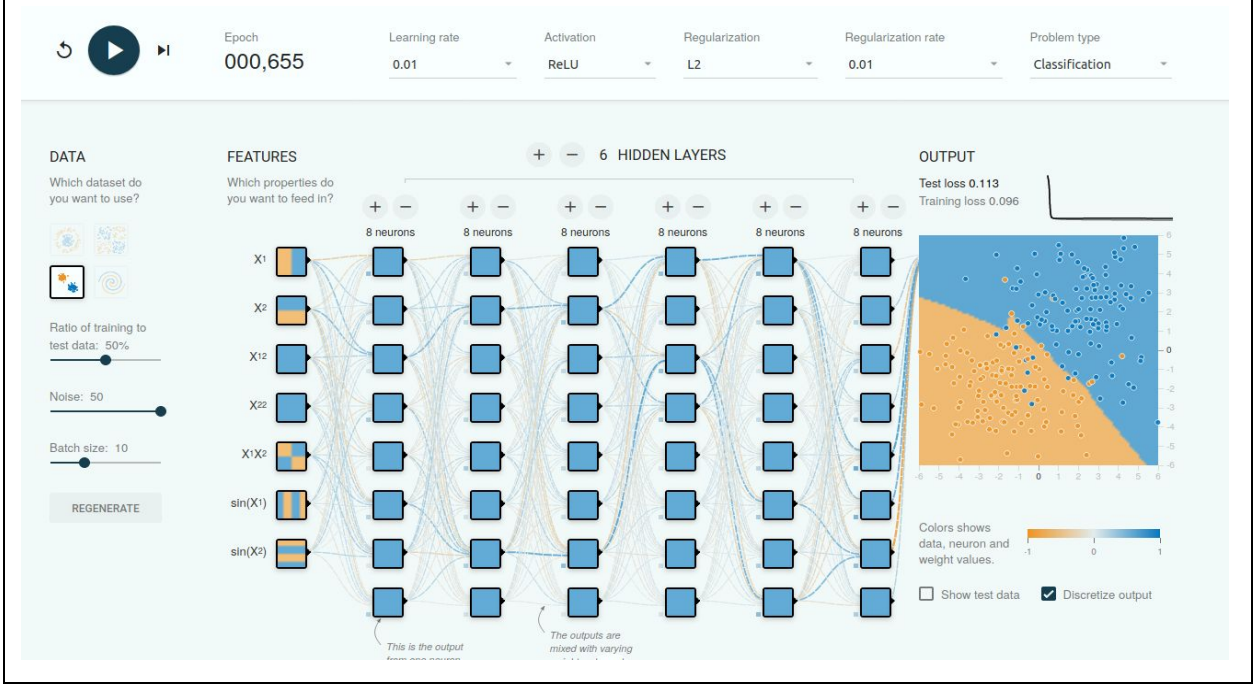


https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.01&regularizationRate=0&noise=25&networkShape=4,3,2&seed=0.04536&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

In all the cases, the learning rates differ by only a factor of 10, which is usually the rate at which we change hyperparameters while fine-tuning them. In the first case, we see the problem of choosing a relatively small learning rate which causes the model to take around 1000 epochs to finally reach the minima, indicating the problem of slow learning. In the second case, we see that choosing a relatively large learning rate causes the model to never converge to the minima. Instead, it keeps oscillating around the minima and in this case, also seems to be slightly diverging away from the minima to indicate the divergence problem of large learning rate. Now in the third case, we see that the model quickly converges and also stays relatively stable, indicating the learning rate worked for this model. From the facts above, and also from the fact that the learning rates were close enough in a practical sense, we see that sensitive tuning of learning rate is highly required to mitigate the slow learning or divergence problems. (Note: There are better features which can be used for this case, but this representation illustrates the above point clearly since usually, we will not directly have the best features in our hand).

Q2. ‘When training and test data distributions vary, then regularization improves generalization’

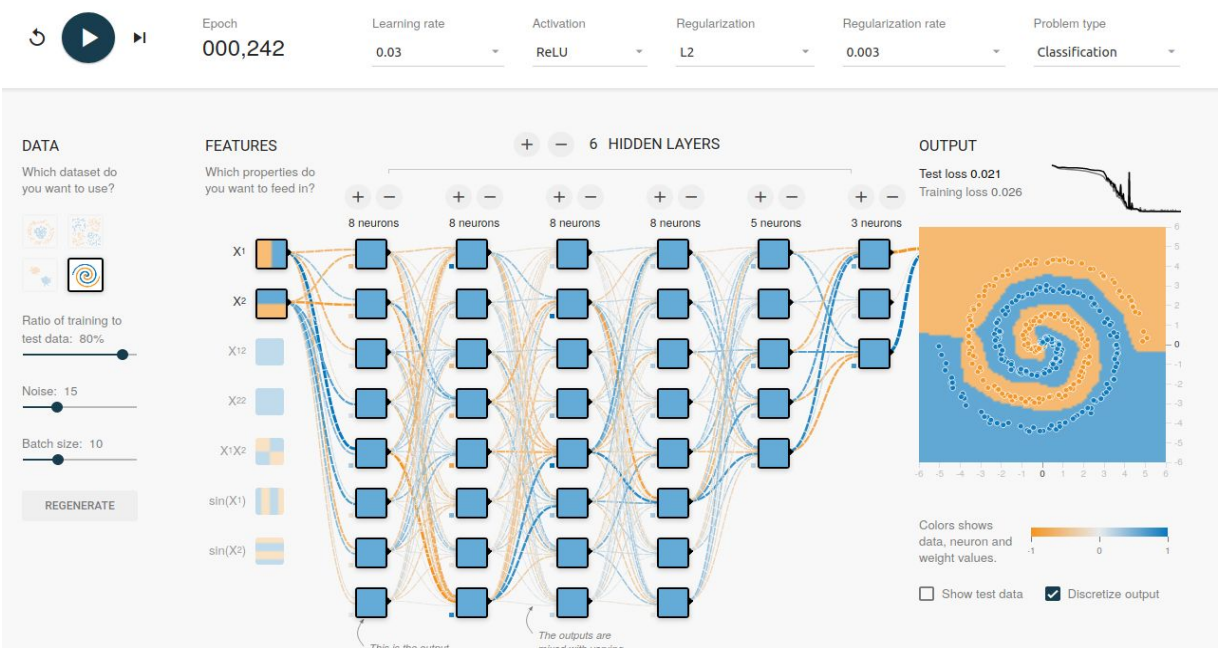
https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.01&regularizationRate=0&noise=50&networkShape=8,8,8,8,8,8&seed=0.50876&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false



https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.01&regularizationRate=0.01&noise=50&networkShape=8,8,8,8,8,8&seed=0.50876&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false
Regularization is a good tool to make sure training proceeds in the right direction. In the dataset we see that the dataset is extremely noisy which makes the training and test dataset distribution appear, although they are a linearly separated class. In this case, we use a deep network and all the features to show the point about what happens in absence of regularization. In the first case we see that in the absence of regularization, the model begins to fit the training distribution extensively that the test loss starts to increase after some time. But in the second case where we have applied regularization, we see that the effect of overfitting on the training data is reduced as compared to the initial case. Thus when we are not sure of the distribution of training and test data and when we are using deep neural networks and lots of features, it helps to apply regularization.

Q3. 'ReLU can be a more effective activation function than tanh, especially for deeper networks'

https://playground.tensorflow.org/#activation=tanh&regularization=L2&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0.003&noise=15&networkShape=8,8,8,8,5,3&seed=0.86475&showTestData=false&discretize=true&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

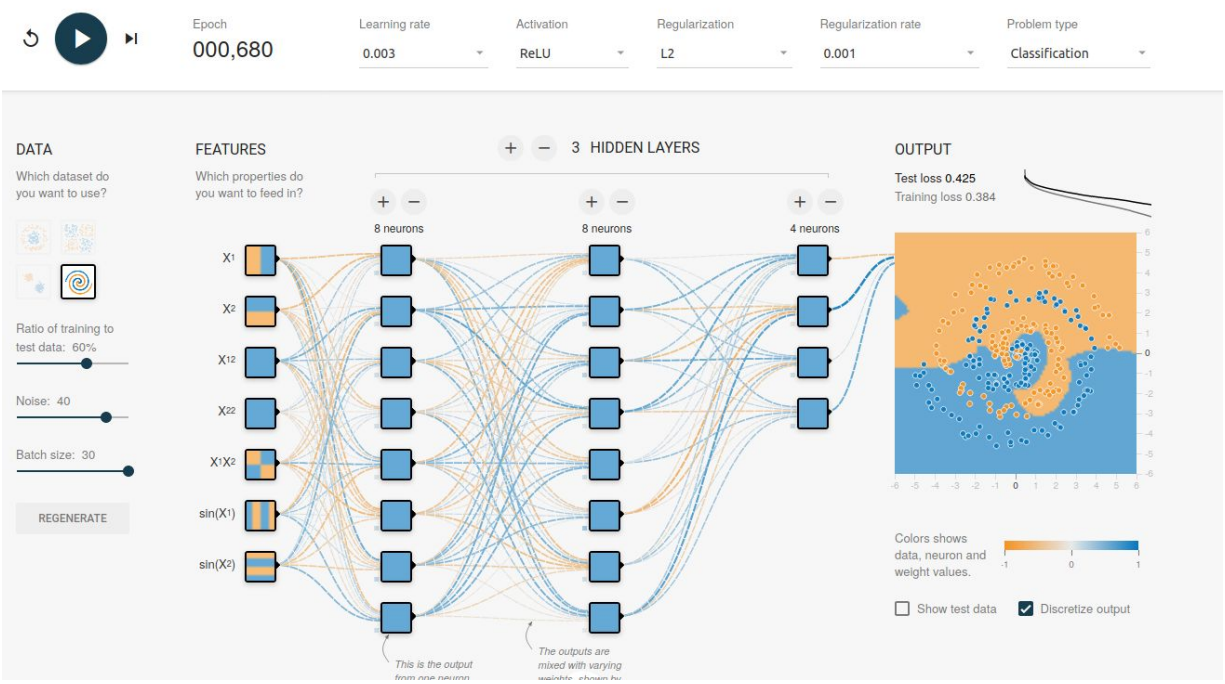


https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0.003&noise=15&networkShape=8,8,8,8,5,3&seed=0.86475&showTestData=false&discretize=true&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

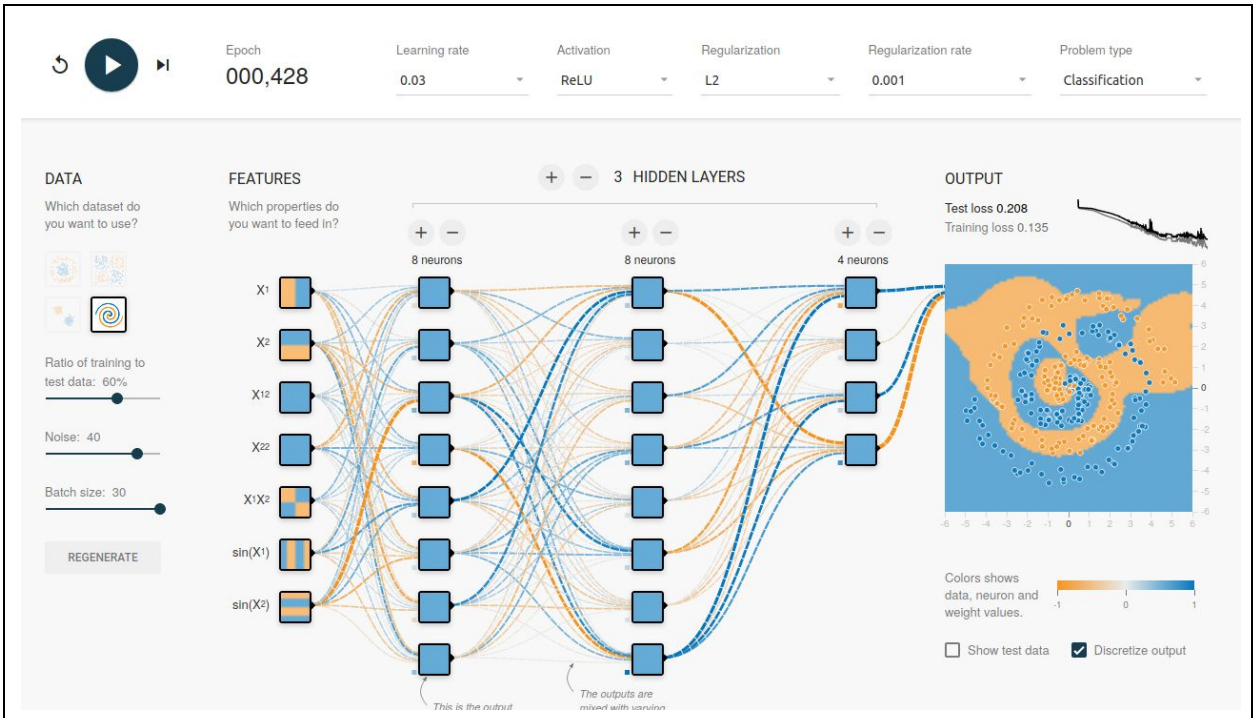
ReLU works much better than tanh, especially in deeper neural networks, due to the vanishing gradient problem. In the first case we see that most of the weights(lines connecting the circles) are dull, which indicates no real training is happening. This occurs due to the nature of the tanh gradient, which drops off to 0 at either ends of the graph (vanishing gradient), and it creates a problem of not giving good gradient values to train the model which effectively stops the model from learning. But in the second case, as we can see the training is done quickly, due to the nature of ReLU gradient, which does not have vanishing gradient, thus enabling the training to continue and finish successfully. Thus ReLU proves to be fruitful compared to tanh in this case, and generally as well. (Note : The calculation of both the function and gradient for ReLU is much simpler than tanh, adding to its advantage.)

~~Q4. 'Adding momentum speeds up learning especially when individual batches generate noisy parameter updates'~~

Q5. ‘Learning rate may have to be scheduled or changed as learning progresses to achieve high accuracy’



https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=30&dataset=spiral&regDataset=reg-gauss&learningRate=0.003&regularizationRate=0.001&noise=40&networkShape=8,8,4&seed=0.32129&showTestData=false&discretize=true&percTrainData=60&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false



https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=30&dataset=spiral&regDataset=reg-gauss&learningRate=0.03&regularizationRate=0.001&noise=40&networkShape=8,8,4&seed=0.32129&showTestData=false&discretize=true&percTrainData=60&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false

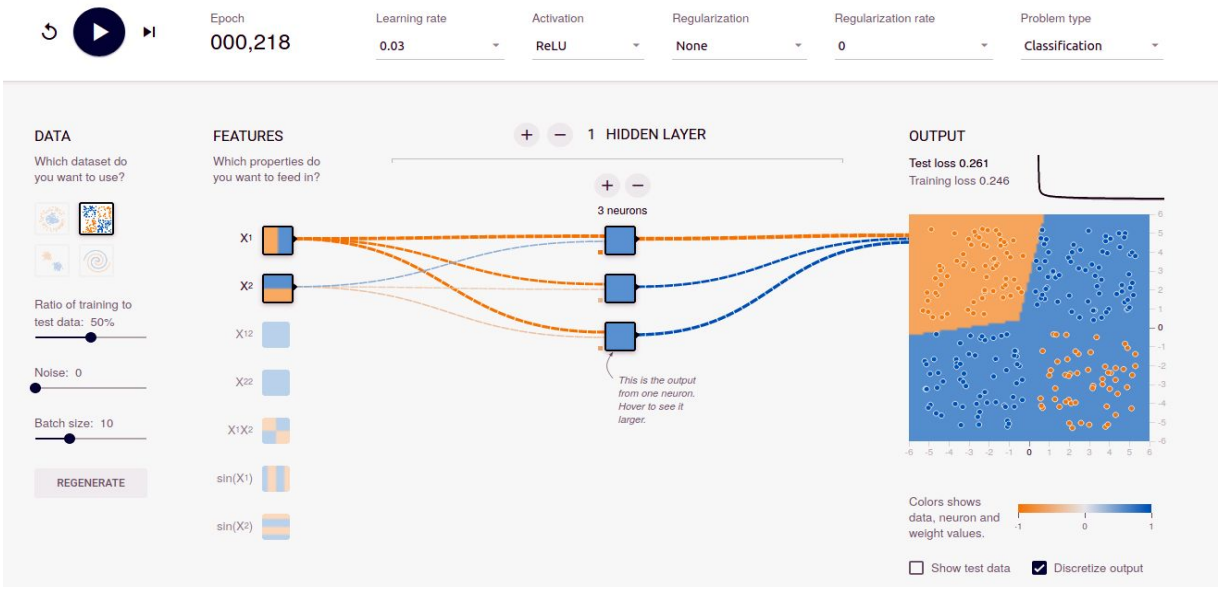
https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=30&dataset=spiral&regDataset=reg-gauss&learningRate=0.03&regularizationRate=0.001&noise=40&networkShape=8,8,4&seed=0.32129&showTestData=false&discretize=true&percTrainData=60&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false

AND

https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=30&dataset=spiral&regDataset=reg-gauss&learningRate=0.003&regularizationRate=0.001&noise=40&networkShape=8,8,4&seed=0.32129&showTestData=false&discretize=true&percTrainData=60&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false

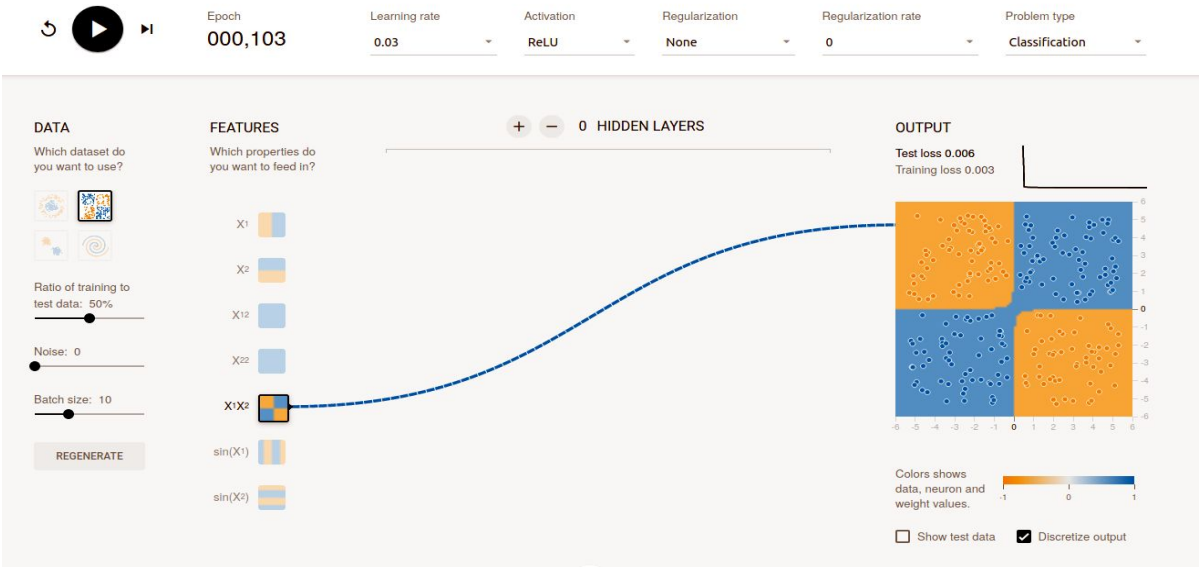
Scheduled learning rate helps in improving the accuracy of the model as compared to only a single model and helps in speeding up an otherwise long training process. In the first image we see that the model is taking too long to converge, and in the second case we see that although the model comes close to converging very fast, the convergence is not happening. To combine the best of both effects, we see that in the third case we train the model using higher learning rate and when it comes close to converging, we lower the training rate to make the model converge.

Q6. 'There can be very clever ways of lowering accuracy, but one simple approach of using a complex network can work (if you are rich enough to collect data, train the models)'

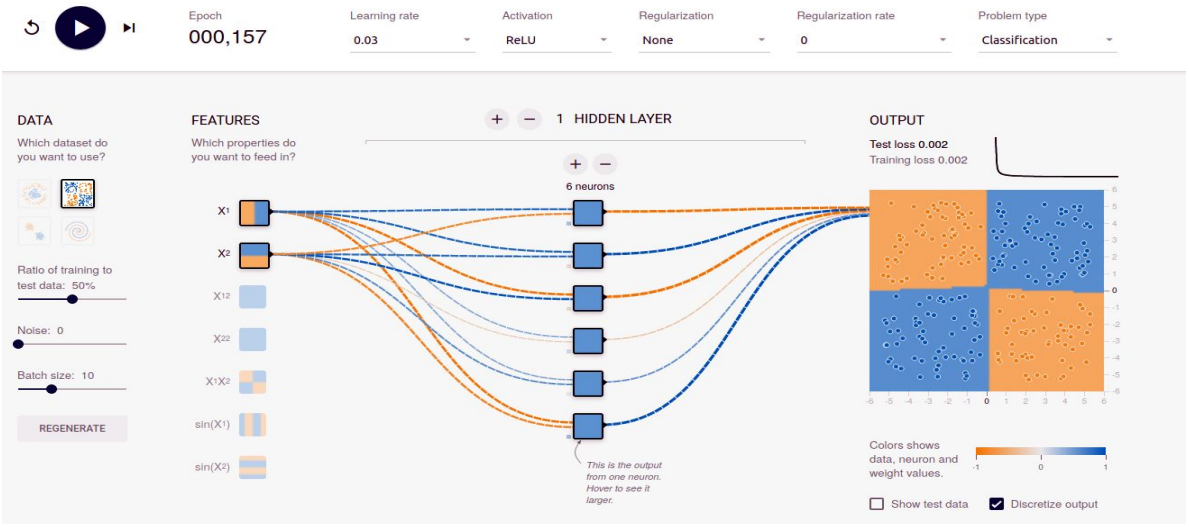


https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=3&seed=0.26535&s

howTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false



https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=&seed=0.26535&showTestData=false&discretize=true&percTrainData=50&x=false&y=false&xTimesY=true&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false



https://playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=6&seed=0.26535&showTestData=false&discretize=true&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false

&problem=classification&initZero=false&hideText=false
The idea of using machine learning is to allow the computer to learn complex functions by itself without much human intervention. The learning complexity of deep networks grows with the increase in complexity of the neural network. While there may be some clever method to capture the complexity in data ourselves, the easiest way is to let the network capture the complexity by increasing the size. In the first case we see that the neural network is unable to learn the function by itself. In the second case, we can see that the clever method would be to supply a different set of features to the neural network which will altogether remove the need for a network itself. But, as we see in the third case, just by increasing the complexity of the network, we are able to fit the data without taking much effort ourselves and allowing the neural network to figure out the required features. The latter approach is particularly useful when we are in a position to not visualize the data, which makes it harder for us to generate better features. Thus we increase the complexity of the neural network and hope the network will learn the required features on its own.