Knockout training coding problem

Motivation: Imagine you want to model the sensorimotor transformation of a fruit fly (see Cowley et al., 2022). You would also like to know how each neuron in the fruit fly's visual system contributes to behavior. You run an experiment where you observe control flies as well as flies whose neurons have been silenced (16 neurons in total). The idea behind knockout training is to perturb the model in a similar way that we perturbed (silenced) the fly. By doing so, we will get a model (KO network) that not only can predict control behavior but also silenced behavior. Beyond that, our model should be able to predict the activity of the neurons!

Use Google colab with either tensorflow and keras or pytorch.

Goal 1: Generate simulated data of a fruit fly

1. Define a randomly initialized, densely-connected, two-layer linear network with 16 hidden units and 1 output unit. Let the number of input variables be 50.
   This is the ground truth "fruit fly" network.
2. Draw 1k input vectors from a standard normal distribution. Pass this input into the fruit fly network to get the network's output. Add Gaussian noise (with sigma=0.05) to the output to get the fruit fly network's "behavior".
3. Now for the silenced behavior. For each ith hidden unit, draw 1k input vectors from a standard normal distribution, pass this through the network while "knocking out" the ith hidden unit (set its value to 0), and add Gaussian noise (sigma=0.05) to the output.
4. You will now have 17 train data sets, one for control and 16 for each hidden unit. This will be your training data.
5. Also generate test data by repeating steps 2-4 except with 500 input vectors. This will give you 17 test data sets.

Goal 2: Knockout training

1. Define another two-layer linear network with 16 hidden units and 1 output unit. This is the KO network; our goal is to match it to the fruit fly network.
2. Train the KO network (*without* knockout training) with SGD (or any optimizer, like Adam) on MSE to match the control behavior of the fruit fly network. What is your trained model's R^2 on the test control data?
3. Now train the KO network with knockout training. For each training batch, train on one of the 17 training data sets. When training on a knocked-out data set, set the corresponding hidden unit's value to 0.
4. For a relative comparison, train a DO ("dropout") network in the same way as the KO network except for silenced data sets, choose a random hidden unit to knock out.
5. Compute the performance R^2 of the KO network and DO network for each of the 17 test data sets and plot. Note: When predicting knocked-out data for the KO network, make sure you knock out the corresponding hidden units.
6. Is performance greater for the KO network or the DO network?

Goal 3: Predicting neural activity

1. Generate 1k input vectors from a standard normal.
2. Pass these inputs into the fruit fly model, taking the activity of the 16 hidden units.
3. Pass the *same* input into the KO and DO models, taking the activity of the 16 hidden units for each model.
4. Compute the R^2 for each hidden unit between the KO/DO model and the fruit fly model. Plot (x-axis: hidden unit index, y-axis: R^2).
5. Does the KO model better predict neural activity than the DO model?

One tricky part: How do you knock out hidden units? There are different ways to do this. I'd recommend an input mask and the Multiply() layer.

Take your time and enjoy! The solution is pretty open ended. When you feel like you've reached a good point, let me know. I'll have you walk through the results + code with me.