AI Explains: Neural Network Scripts and rev.02 Changes

# 03a\_simple1hiddenlayer

## Original Script:

- A simple neural network with one hidden layer (H=2) is trained to classify 1D data (X in [-20, 20]).  
- Uses SGD optimizer, 1000 epochs, and a custom cross-entropy loss.  
- The model learns to map a non-linearly separable region to a linearly separable one.  
- Plots show the model's output and activations during training.

## rev.02 Changes:

- Data range is expanded (X in [-30, 30]), and the hidden layer size is increased (H=4).  
- Uses Adam optimizer instead of SGD, and trains for 800 epochs.  
- Random seed is changed for different initialization.

## What the Changes Mean:

- Increasing the hidden layer size allows the model to learn more complex patterns.  
- Adam optimizer adapts the learning rate, often leading to faster or more stable convergence.  
- Changing the data range and seed means the model sees a different problem and starts from different weights, so the results and learned boundaries will differ.

# 03b\_multiple\_neurons

## Original Script:

- Similar to 03a, but focuses on the effect of more hidden neurons.  
- Trains on X in [-20, 20], H=2, SGD, 1000 epochs.  
- Shows how increasing neurons can improve the model's ability to fit the data.

## rev.02 Changes:

- Data is denser (X in [-10, 10], step 0.5), H=5 (more neurons), Adam optimizer, 600 epochs.  
- Random seed changed for new initialization.

## What the Changes Mean:

- More neurons = more model capacity, so the network can fit more complex shapes.  
- Adam optimizer and denser data can lead to different learning dynamics and potentially better fits.

# 03c\_xor\_v2

## Original Script:

- Trains a network to solve the noisy XOR problem with 1, 2, and 3 hidden neurons.  
- Data: 100 samples, SGD, 500 epochs, BCELoss.  
- Shows how more neurons are needed to solve XOR.

## rev.02 Changes:

- Data: 200 samples (more noise), batch size 4, Adam optimizer, 300 epochs.  
- Random seed changed.

## What the Changes Mean:

- More data and noise make the problem harder, so the model's ability to generalize is tested.  
- Adam optimizer and batch training can help with convergence and stability.  
- Fewer epochs may mean less overfitting, but also less time to learn.

# 03d\_one\_layer\_neural\_network\_MNIST

## Original Script:

- Classifies MNIST digits with a single hidden layer (H=50), SGD, 2 epochs.  
- Shows basic image classification with a shallow network.

## rev.02 Changes:

- Hidden layer increased to H=128, Adam optimizer, 3 epochs, new seed.

## What the Changes Mean:

- More hidden units = more capacity to learn digit features.  
- Adam optimizer can speed up and stabilize training.  
- More epochs = better accuracy, but risk of overfitting if too many.

# 03e\_activationfuction\_v2

## Original Script:

- Visualizes activation functions: Sigmoid, Tanh, ReLU.  
- Shows their shapes and how they transform input data.

## rev.02 Changes:

- Adds LeakyReLU, expands data range to [-10, 10], new seed.

## What the Changes Mean:

- LeakyReLU helps with the "dying ReLU" problem (neurons stuck at zero).  
- Wider data range shows more of each function's behavior.

# 03f\_mist1layer\_v2

## Original Script:

- Tests different activation functions on MNIST with H=50, SGD, 2 epochs.  
- Compares Sigmoid, Tanh, ReLU.

## rev.02 Changes:

- H=64, Adam optimizer, 2 epochs, new seed.  
- Tries ReLU, Sigmoid, and Tanh in a new order.

## What the Changes Mean:

- More hidden units and Adam can improve learning.  
- Different activations affect how the network learns features: ReLU is often best for deep nets, Sigmoid/Tanh can saturate.  
- Changing order lets you see how each activation impacts accuracy and loss.

# General Notes for Students

- Changing the random seed changes the initial weights, so results may vary.  
- More hidden units = more model capacity, but also more risk of overfitting.  
- Adam optimizer is usually faster and more robust than SGD, but can sometimes overfit.  
- More data or more noise makes the problem harder, but can help generalization.  
- Try running the scripts yourself and experiment with different values to see how the model behaves!