Training Ebru's 17-Dim Dynamic Scheduling Model

Overall plan of improving training results:

- Step 1: Optimize network architecture to reduce loss and improve convergence
  - Order:
    a. layer count (ongoing),

    b. nodes per layer,

    c. activation function
  - Use standard learning rate schedule for now.
- Step 2: Test reference policies to enhance neural network performance.
  - Review values and plots. Fix any issues.
- Step 3: Fine-tune learning rate schedule to maximize performance.

Benchmark: Ebru's Python implementation

Ebru's test instance:

- 17 dimensions, 17 hours, 3060 time steps
- Reference policy: all-zero control

Ebru's benchmark results:

- End loss: ~ 5
- Total steps: 30,000
- Sample average of trained $V^{\mathrm{NN}}(0, X_0)$: between 80 and 120

Experiment Setup

In this experiment, we will use the following hyperparameter settings:

Hyperparameters	**Values**
Neural network architecture	MLP
Number of hidden layers	to be tested
Number of nodes per layers	100
Activation function	ReLU
Precision	float64
Optimizer	Adam
Batch size (training)	256
Batch size (validation)	512
Number of iterations	TBD (manual adjustment)
Learning rate schedule	Piecewise decay (manual)
Learning rates	Starts at 10^{-2}, cut by $1/2$, minimum 10^{-5}

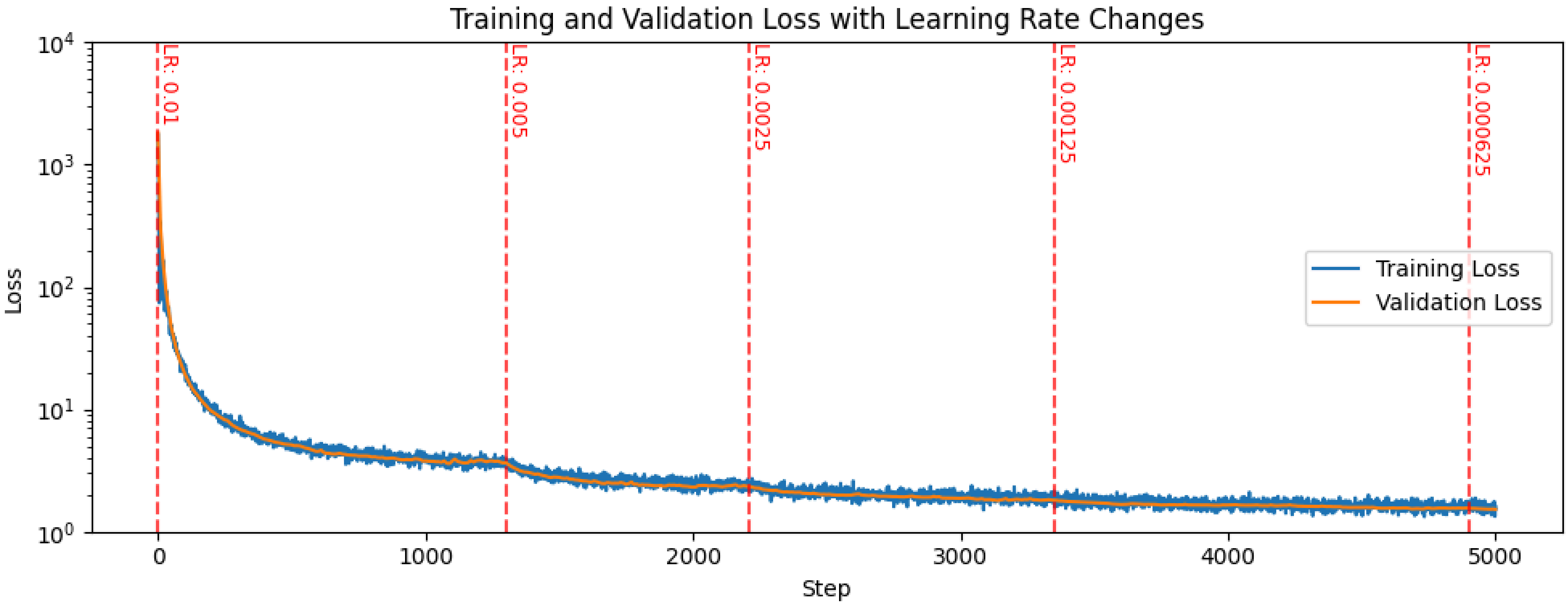
We will test the following layer counts: 2, 3, 4, 5.

Experiment Result Summary

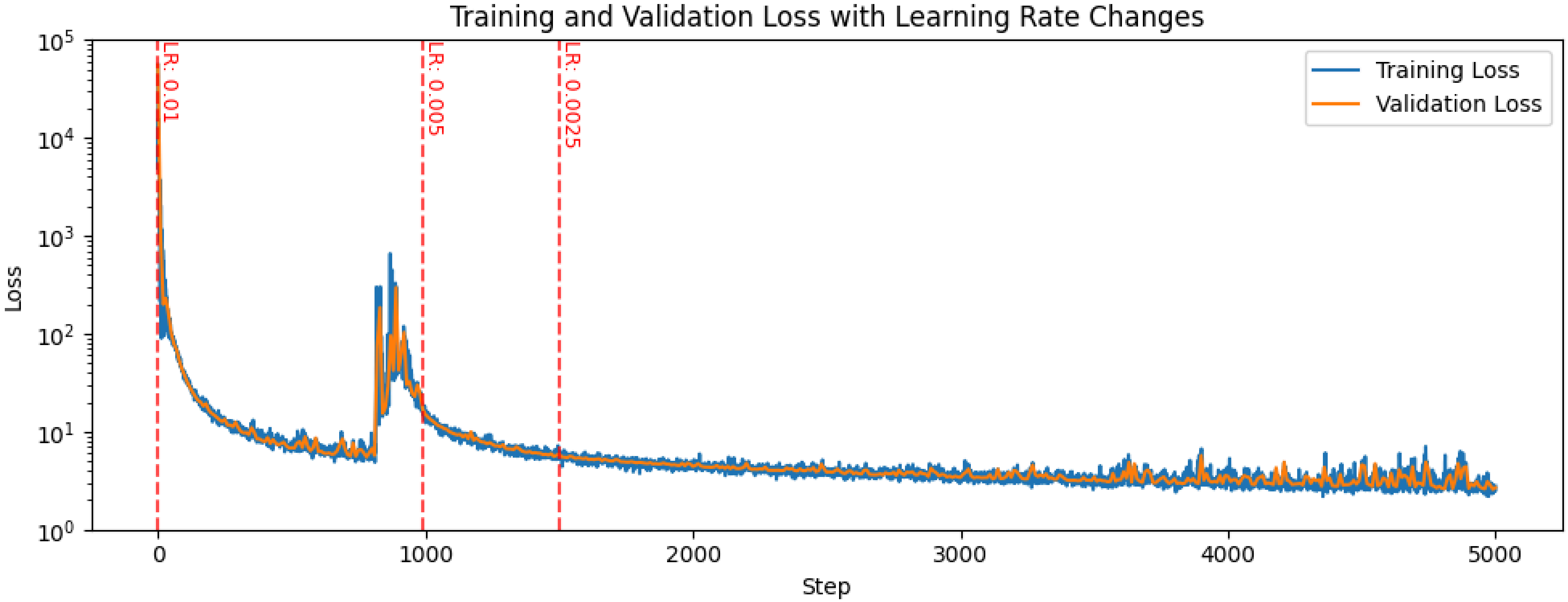
Only partial results are collected. More results will be added later.

Number of layers	**End loss**	**Total steps**	**Average of trained $V^{\mathrm{NN}}(0, X_0)$**	**Notes**
2				waiting for results
3				waiting for results
4	~ 1.5	5000 (running for more)	~ 12.1	waiting for more results
5	~ 2.7	5000 (running for more)	~ 2.5	waiting for more results

Loss Curve - 4 Layers



Loss Curve - 5 Layers


Training and Validation Loss with Learning Rate Changes
Loss
10^5
10^4
10^3
10^2
10^1
10^0
Step
LR: 0.01
LR: 0.005
LR: 0.0025
Training Loss
Validation Loss