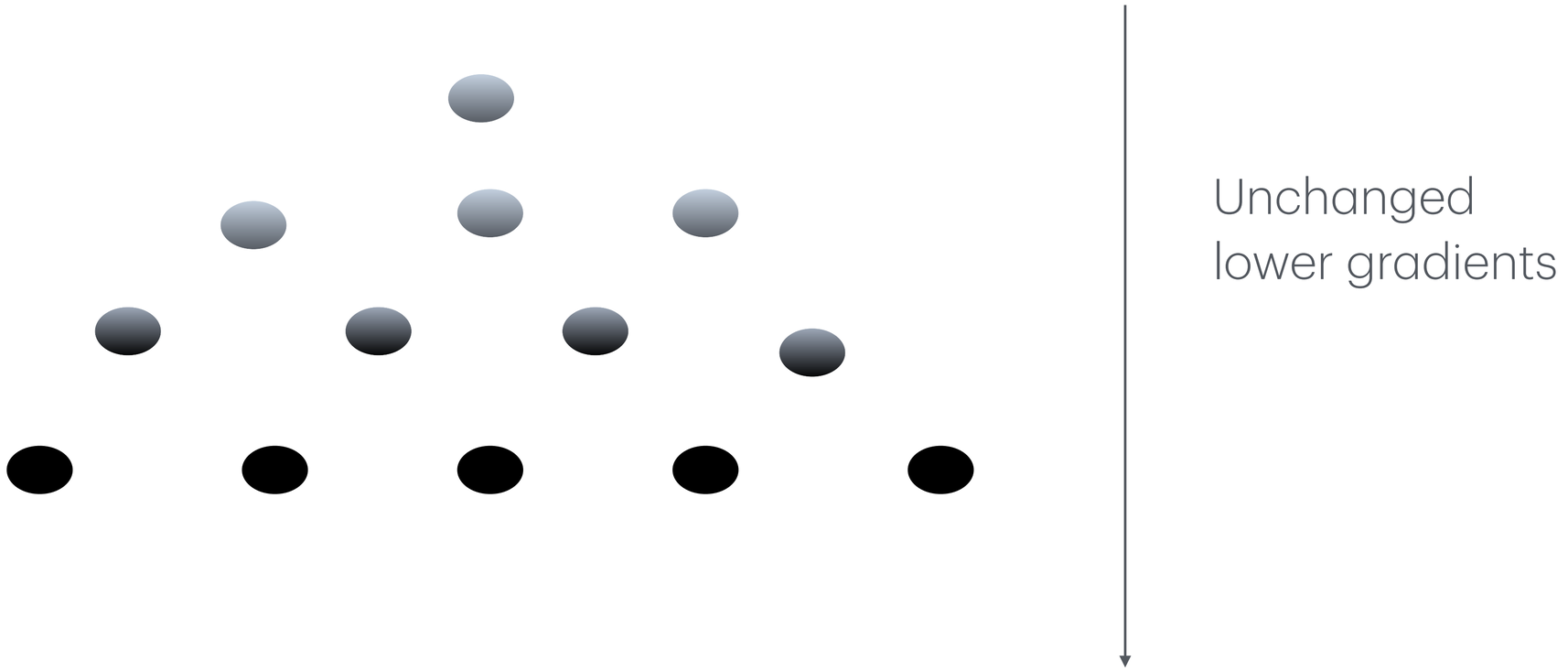
Training Deep Neural Networks

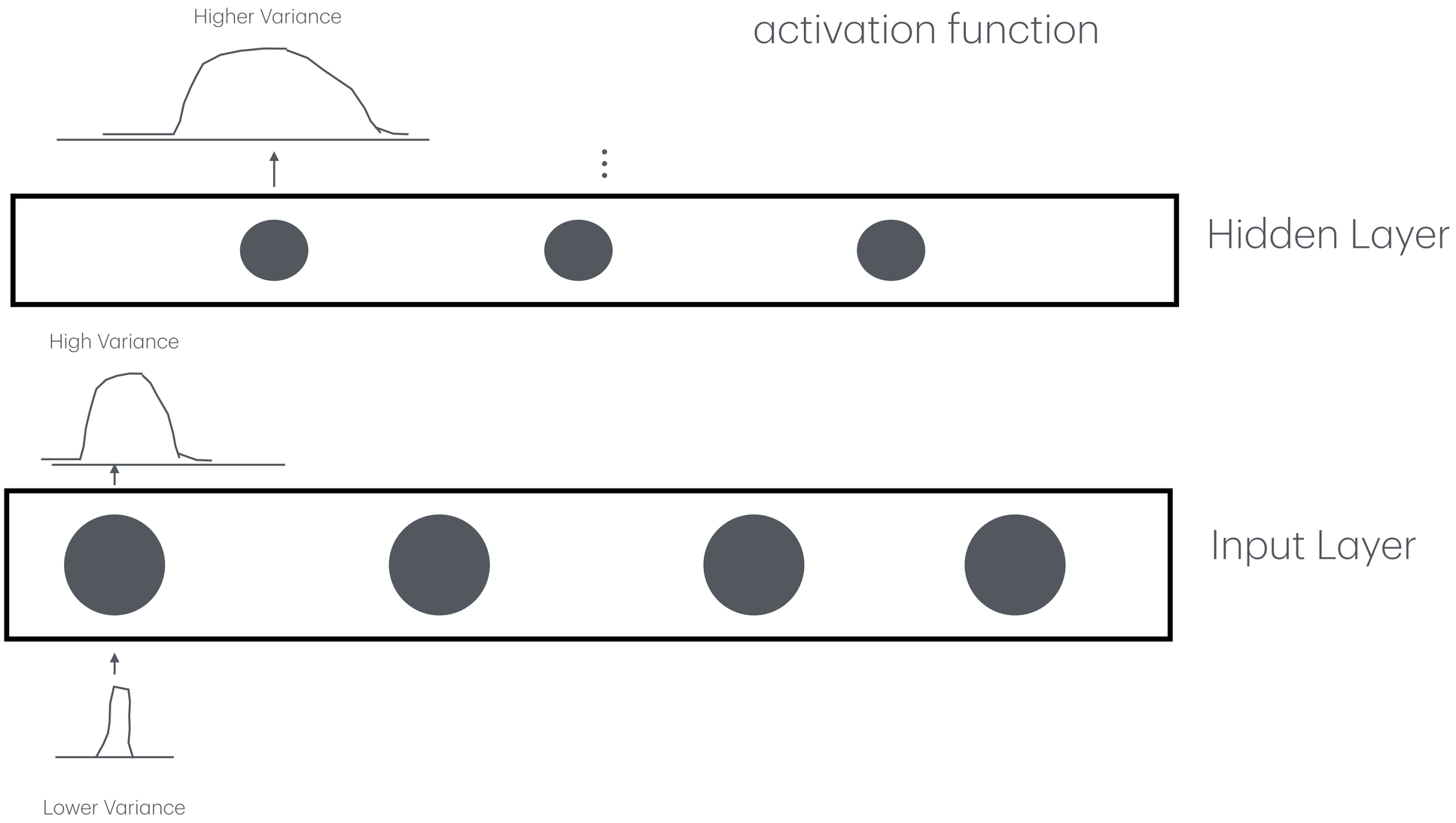
Vanishing Gradients



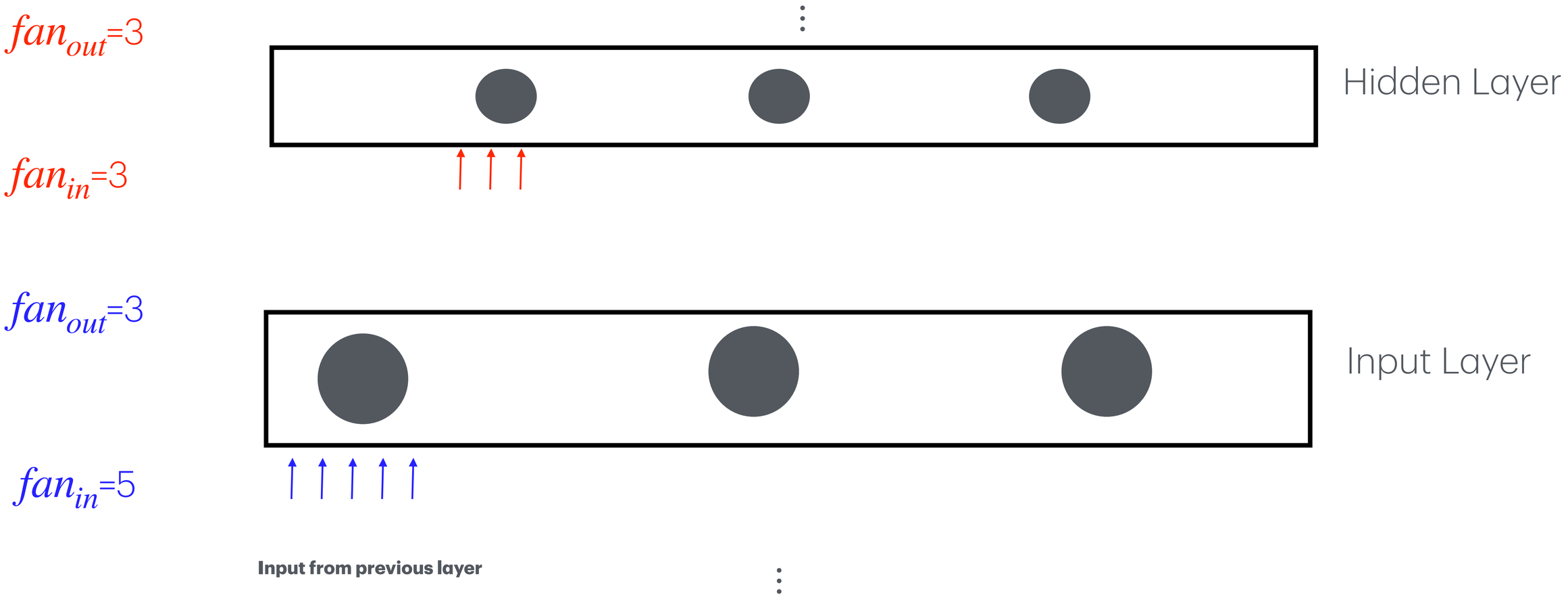
Unchanged
lower gradients

Vanishing/Exploding Gradients: Why?

Increase variance at layers causing saturation on the activation function



Vanishing/Exploding Gradients

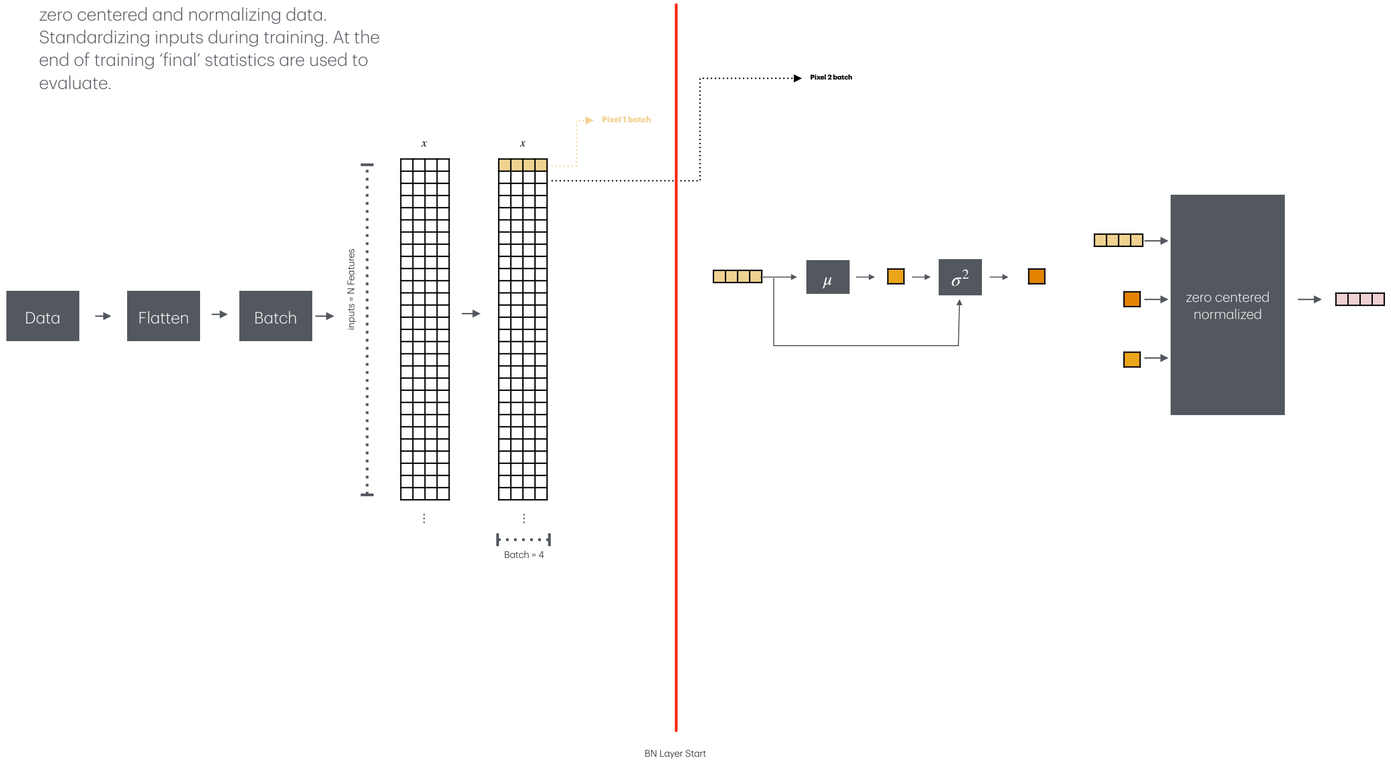


Vanishing/Exploding Gradients

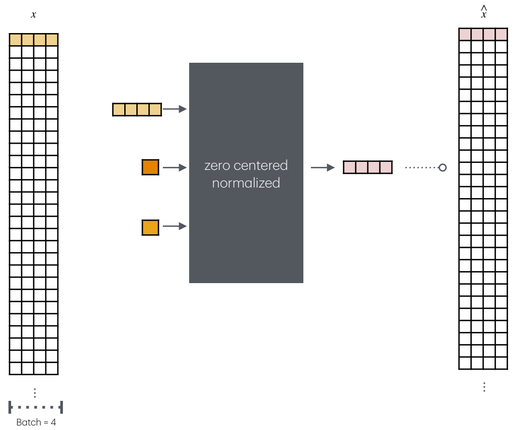
Glorot Uniform	Logistic Activation Softmax Tanu	• Normal Distribution • Uniform Distribution	
LeCun Normal	SELU		Outputs of each layer self-normalize (mean=0, std=1) Must initialize hidden layer weights with LeCun normal Neural Network(NN) architecture must be sequential
He	RELU ELU		ELU is slow, but coverages quicker than other functions
	Leaky RELU		Never dies’
	RRELU Randomized		‘alpha’ variable acts as regularizer reducing overfitting
	Parametric Leaky RELU		‘alpha’ variables learns during training

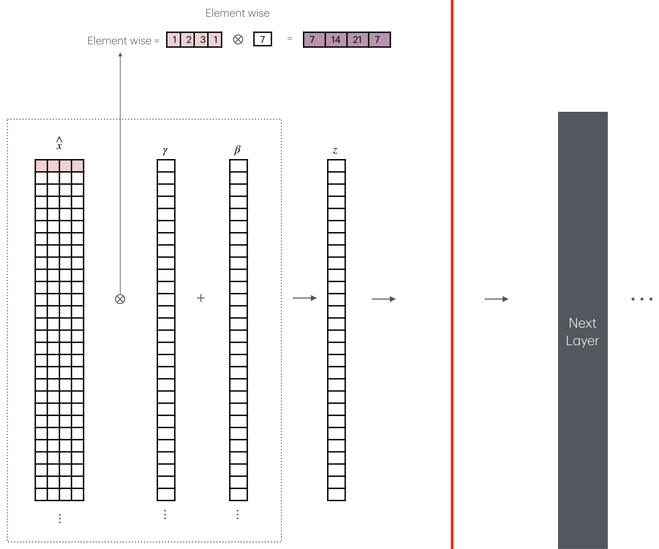
BatchNormalization

Addresses gradients growth/drop problem by zero centered and normalizing data. Standardizing inputs during training. At the end of training ‘final’ statistics are used to evaluate.



BatchNormalization





γ
β Trainable elements during back propagation

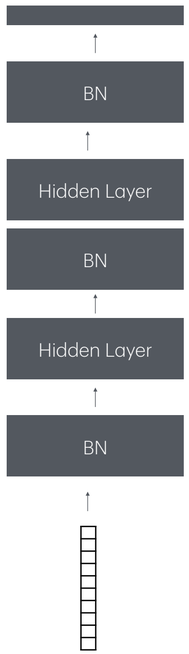
μ
σ Moving average updates during training

BN Layer End

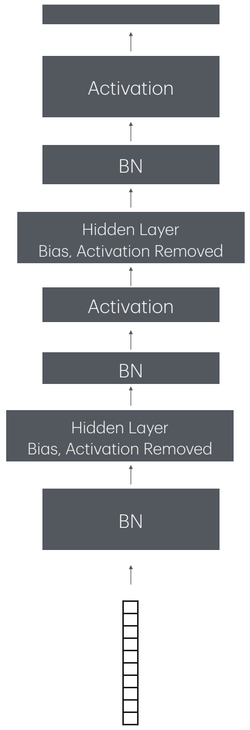
Note: Acts as a regularizer, reducing overfitting

BatchNormalization

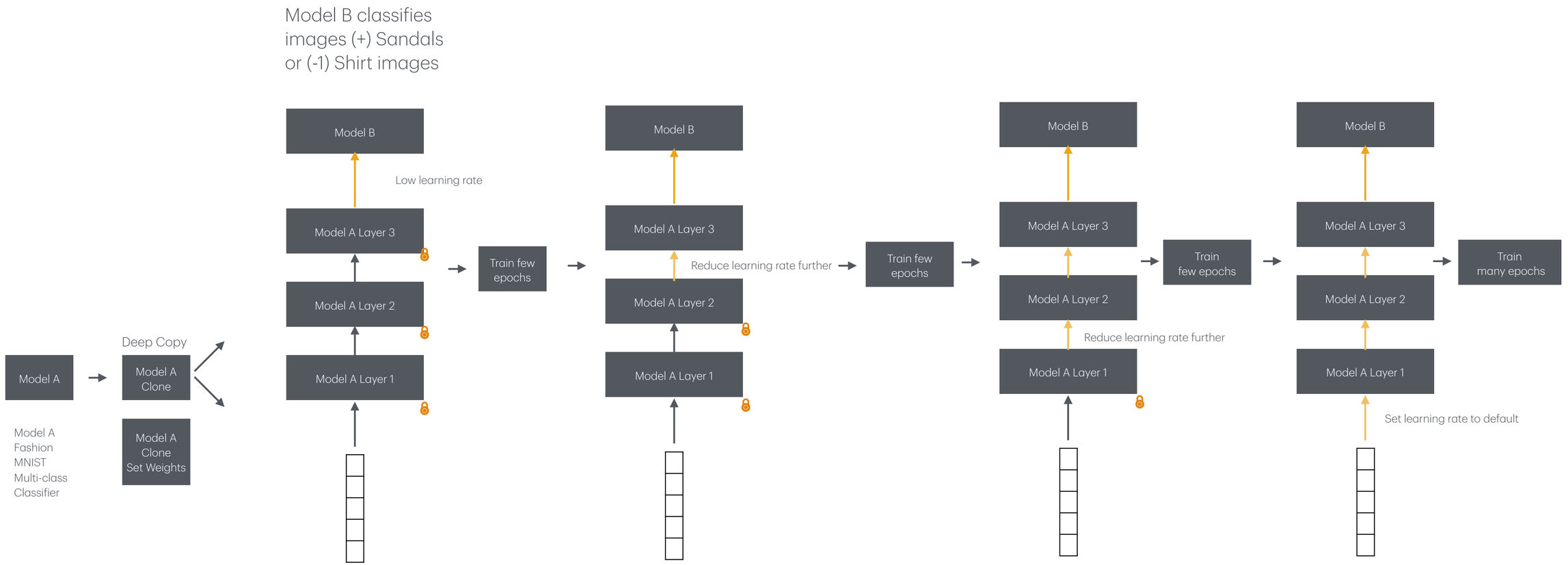
Option 1



Option 2

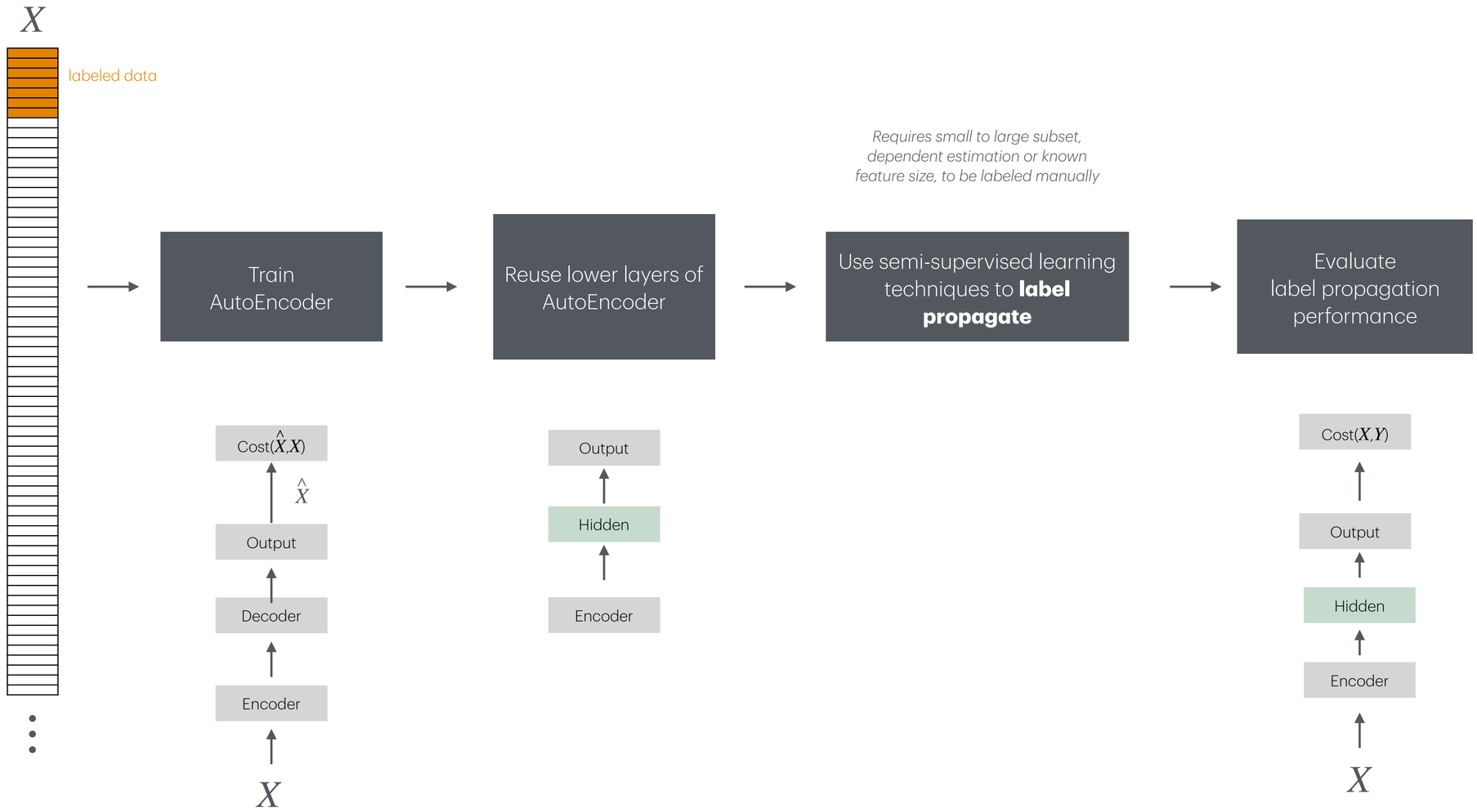


Transfer Learning

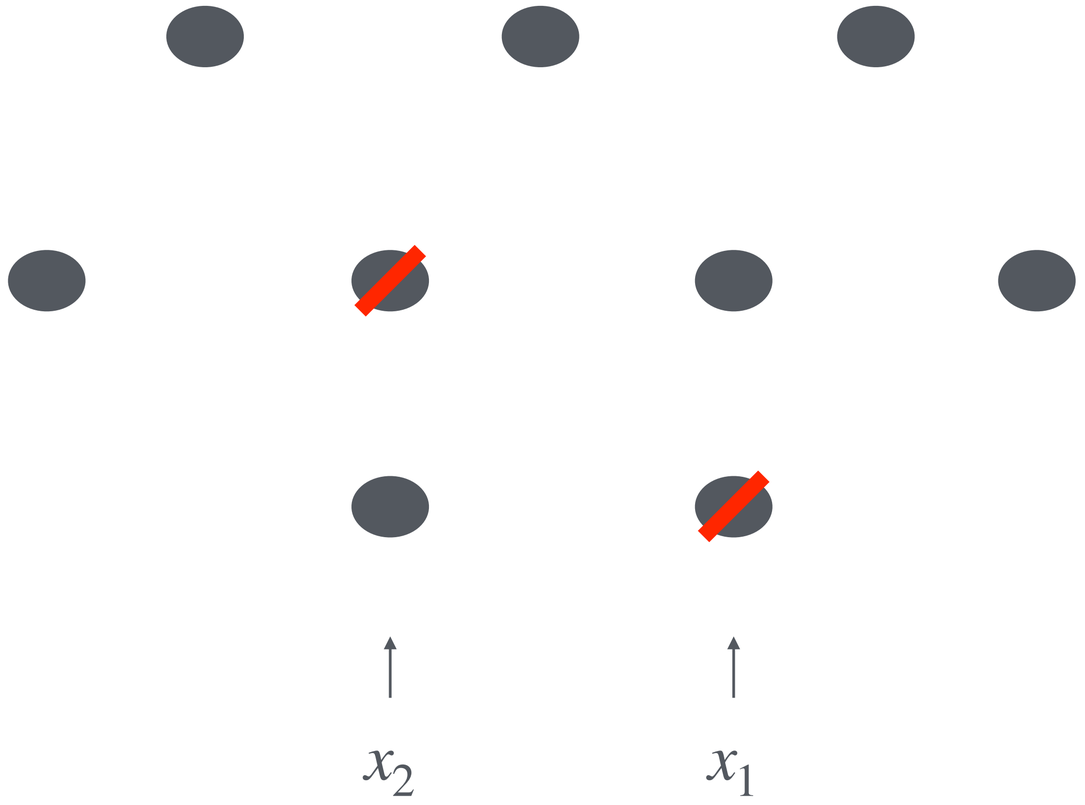


Transfer learning works on deep convolutional neural networks.
Not a good candidate for TL as small dense networks learn specific ‘patterns’

Unsupervised Learning



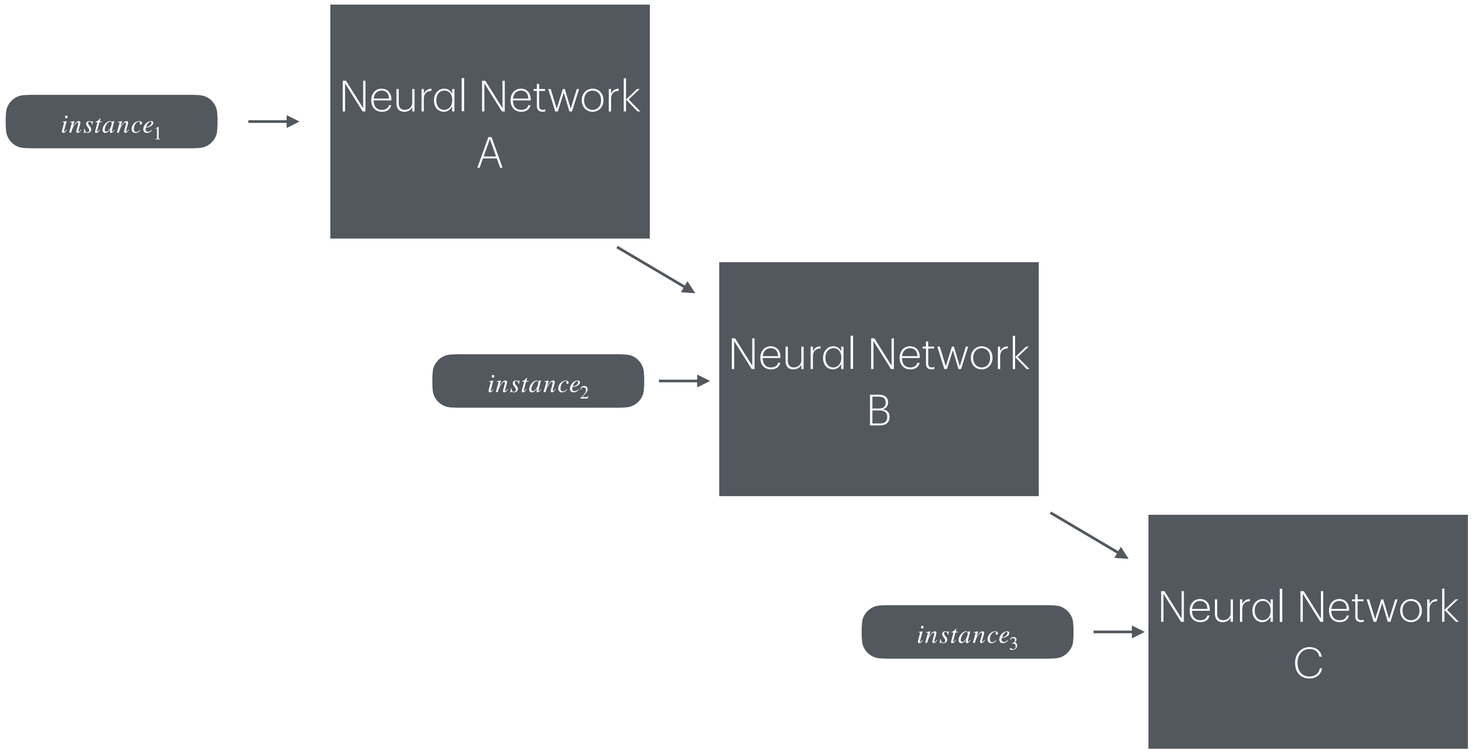
Dropout



Neurons must be learn ‘well’ on their own as inputs will vary in size

Neurons less sensitive to input changes, sustaining learned behavior from few inputs, equating to a manageable variance

Dropout

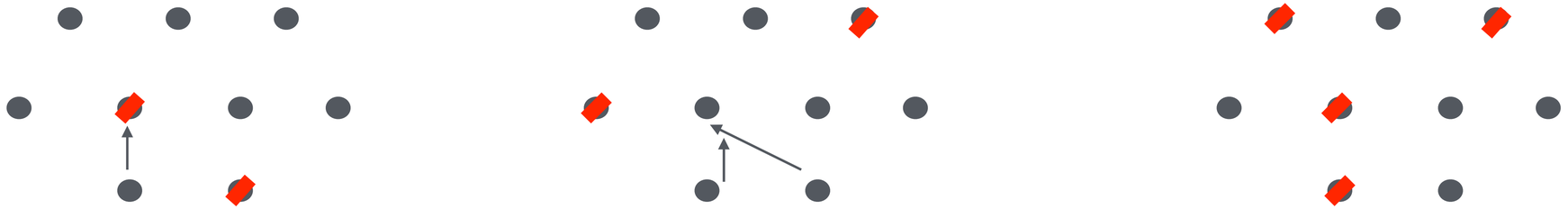


At every step (new instance) a
new training network is trained

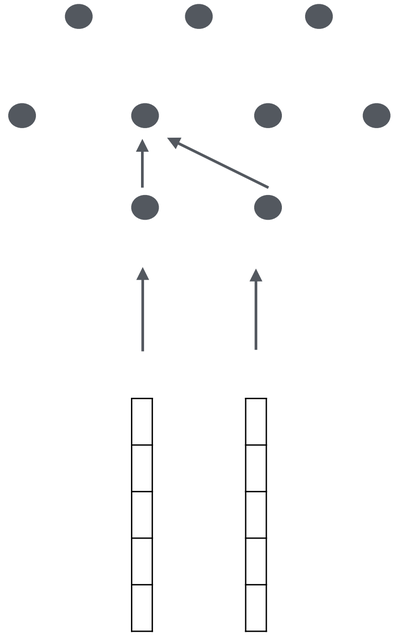
10000 training steps = 10000 training networks

Dropout

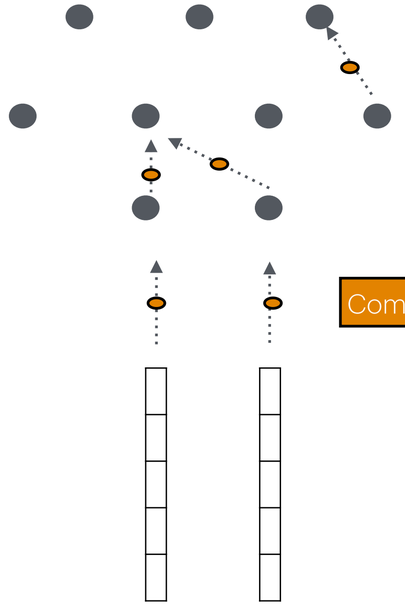
Training



Non-Training



During testing a neuron could be connected to twice as many inputs

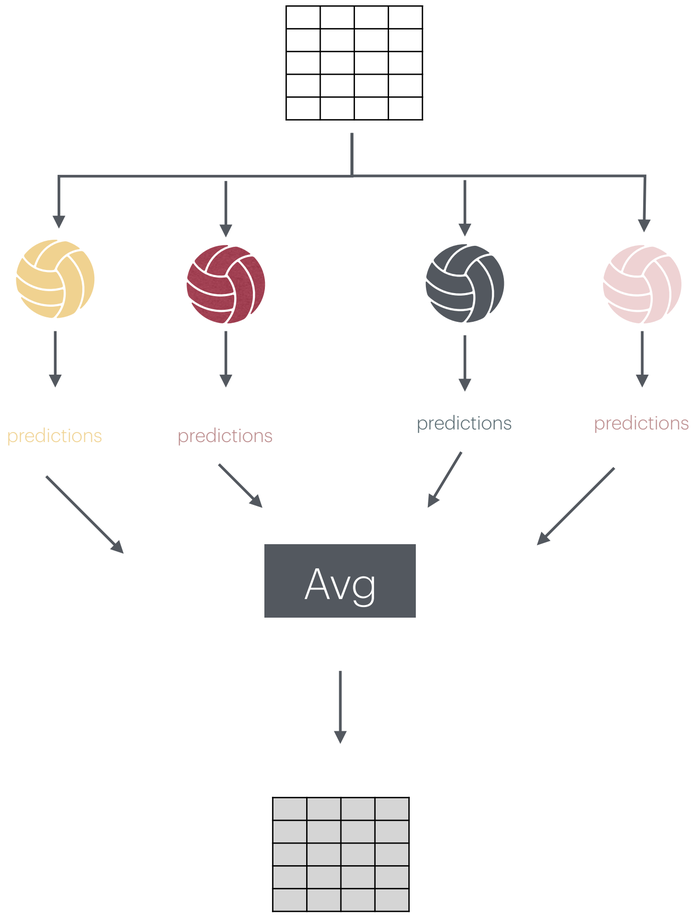


Multiply each neuron input connection weight neuron by 0.5

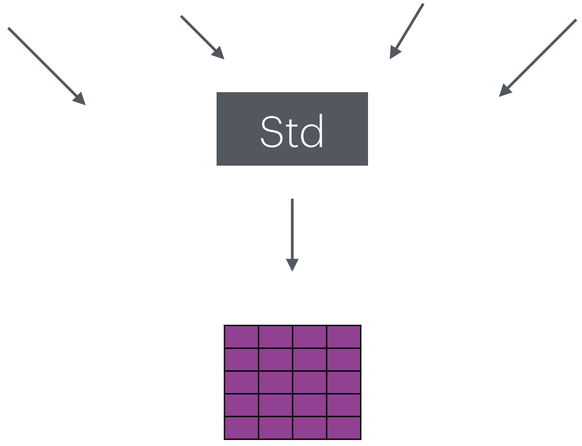
Compensator(keep probability) = 0.50

Or Divide neurons outputs by 0.5 during training

Monte Carlo (4 prediction states)

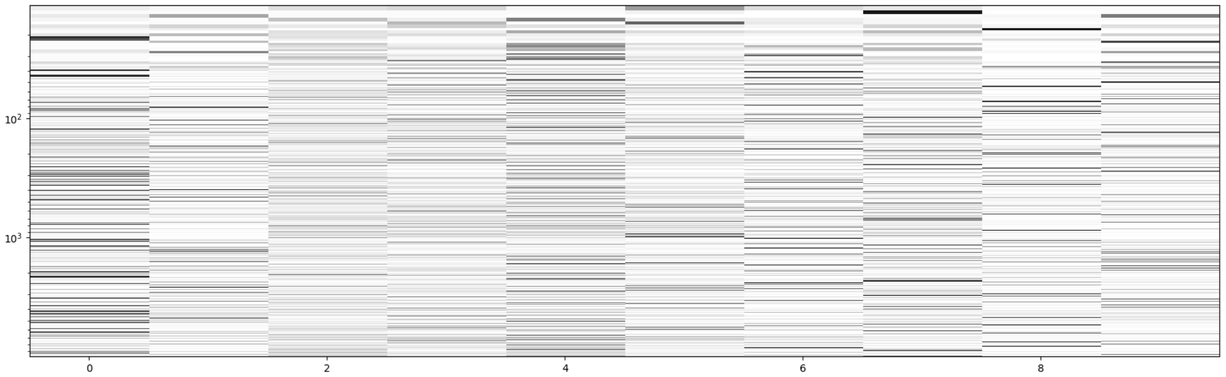


Predictions made on
4 different networks

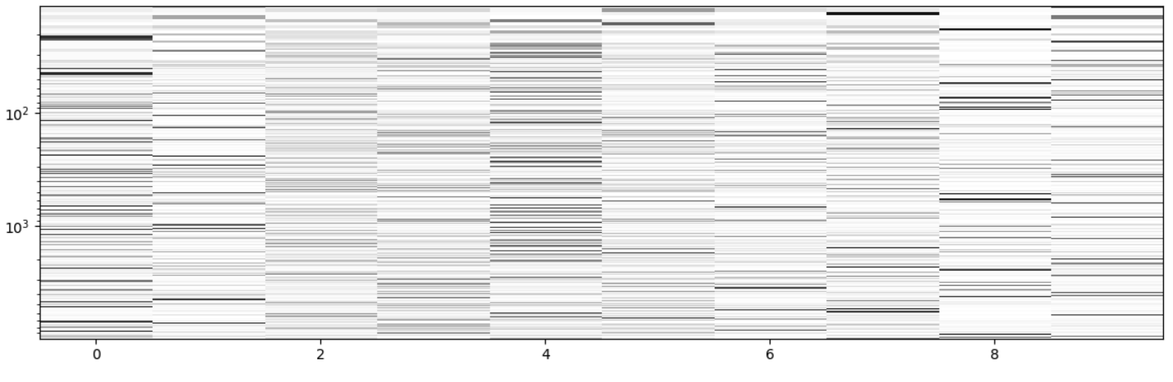


Predict, MC Predictions, MC STD Dev

Model Prediction

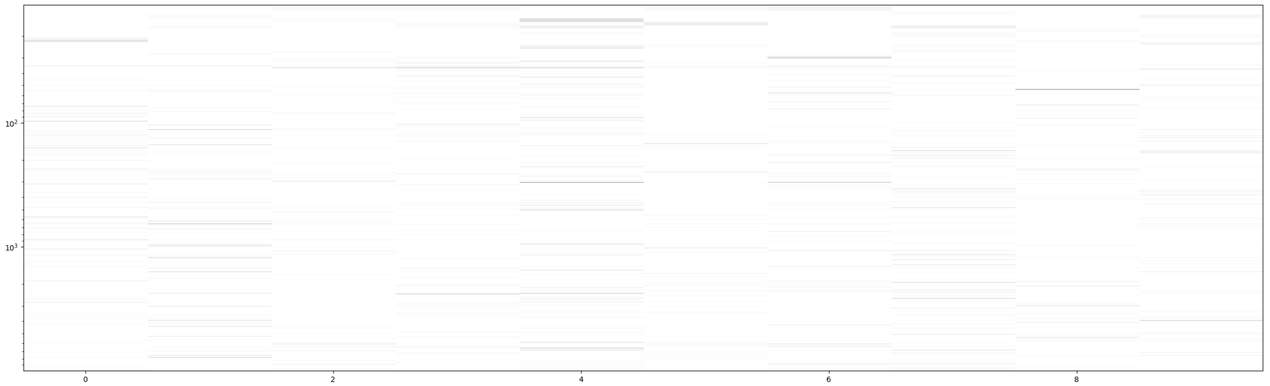


Model + Monte Carlo Prediction



Poor accuracy across both

Residuals(Model, Model+MonteCarlo)



Dark - error

White - no error

ICycle Learning Schedule