Deep Learning (1470)

Randall Balestriero

Class 6: Optimization and Hyper-parameters

The Full (real) Story

```python
for x_batch, y_batch in loader:
    optimizer.zero_grad()
    # Forward pass: raw logits (no softmax)
    logits = model(x_batch)
    # Functional cross entropy:
    # takes logits and class indices directly
    loss = F.cross_entropy(logits, y_batch)
    loss.backward()
    optimizer.step()
```

The Full (real) Story

```
for x_batch, y_batch in loader:
    optimizer.zero_grad()
    # Forward pass: raw logits (no softmax)
    logits = model(x_batch)
    # Functional cross entropy:
    # takes logits and class indices directly
    loss = F.cross_entropy(logits, y_batch)
    loss.backward()
    optimizer.step()
```

Where is the derivative?

Autodiff!

- All deep learning frameworks implement automatic differentiation

Autodiff!

- All deep learning frameworks implement automatic differentiation

Autodiff!

- All deep learning frameworks implement automatic differentiation

Autodiff!

- All deep learning frameworks implement automatic differentiation

Autodiff!

- All deep learning frameworks implement automatic differentiation

Autodiff!

- All deep learning frameworks implement automatic differentiation

JAX TensorFlow PyTorch theano

- You still need to know about chain-rule and gradients!

Autodiff!

- All deep learning frameworks implement automatic differentiation

TensorFlow PyTorch theano

- You still need to know about chain-rule and gradients!
- The efficient implementation is called “backprop” with vjp (or jvp)

The Elephant in the Room

- Which framework to use?

The Elephant in the Room

- Which framework to use?

The Elephant in the Room

- Which framework to use?

Use the one you are good at!

What are our hyper-parameters so far?

What are our hyper-parameters so far?

- Preprocessing of the data

What are our hyper-parameters so far?

- Preprocessing of the data
- Number of layers (L), width of each layer

What are our hyper-parameters so far?

- Preprocessing of the data
- Number of layers (L), width of each layer
- Initialization of the parameters

What are our hyper-parameters so far?

- Preprocessing of the data
- Number of layers (L), width of each layer
- Initialization of the parameters
- Optimizer, learning rate (+ momentum, …)

What are our hyper-parameters so far?

- Preprocessing of the data
- Number of layers (L), width of each layer
- Initialization of the parameters
- Optimizer, learning rate (+ momentum, …)
- Mini-batch size

What are our hyper-parameters so far?

- Preprocessing of the data
- Number of layers (L), width of each layer
- Initialization of the parameters
- Optimizer, learning rate (+ momentum, …)
- Mini-batch size
- Training steps

Initialization Brainstorming

$$\mathbf{W}^{(3)}\mathrm{ReLU}(\mathbf{W}^{(2)}\mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$

Initialization Brainstorming

$$\mathbf{W}^{(3)}\mathrm{ReLU}(\mathbf{W}^{(2)}\mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$

What would be a good/bad initialization?

Initialization Brainstorming

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}\|_2^2$$

Initialization Brainstorming

$$\mathscr{L} = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}\|_2^2$$

$$\mathbf{W} \leftarrow \mathbf{W} - \lambda\frac{1}{N}\sum_{n=1}^{N} (\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)} - \mathbf{y}_n)\mathbf{x}_n^\top$$

Initialization Brainstorming

$$\mathscr{L} = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}\|_2^2$$

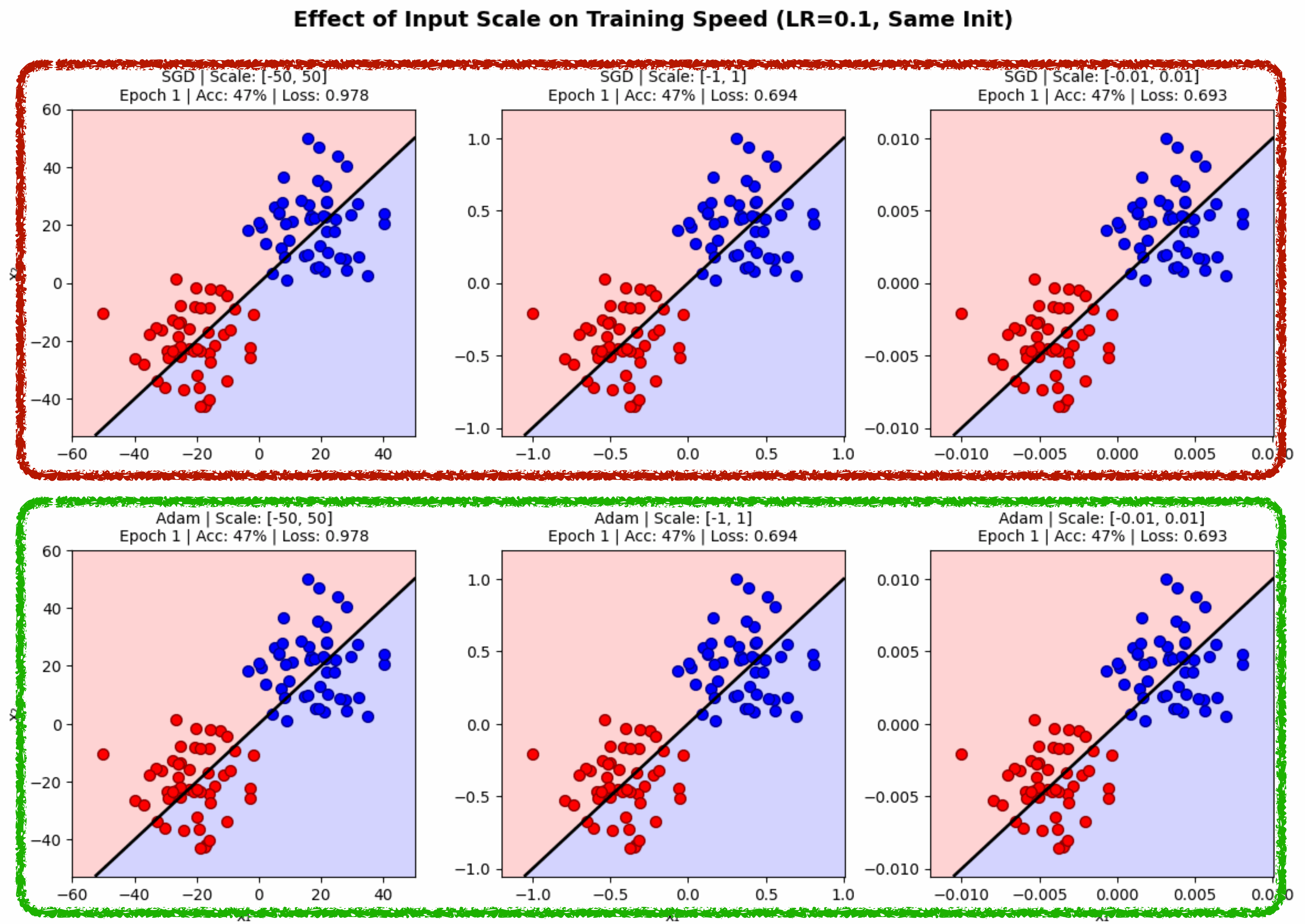
Data scaling impacts training dynamics!

$$\mathbf{W} \leftarrow \mathbf{W} - \lambda \frac{1}{N}\sum_{n=1}^{N} (\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)} - \mathbf{y}_n)\mathbf{x}_n^\top$$

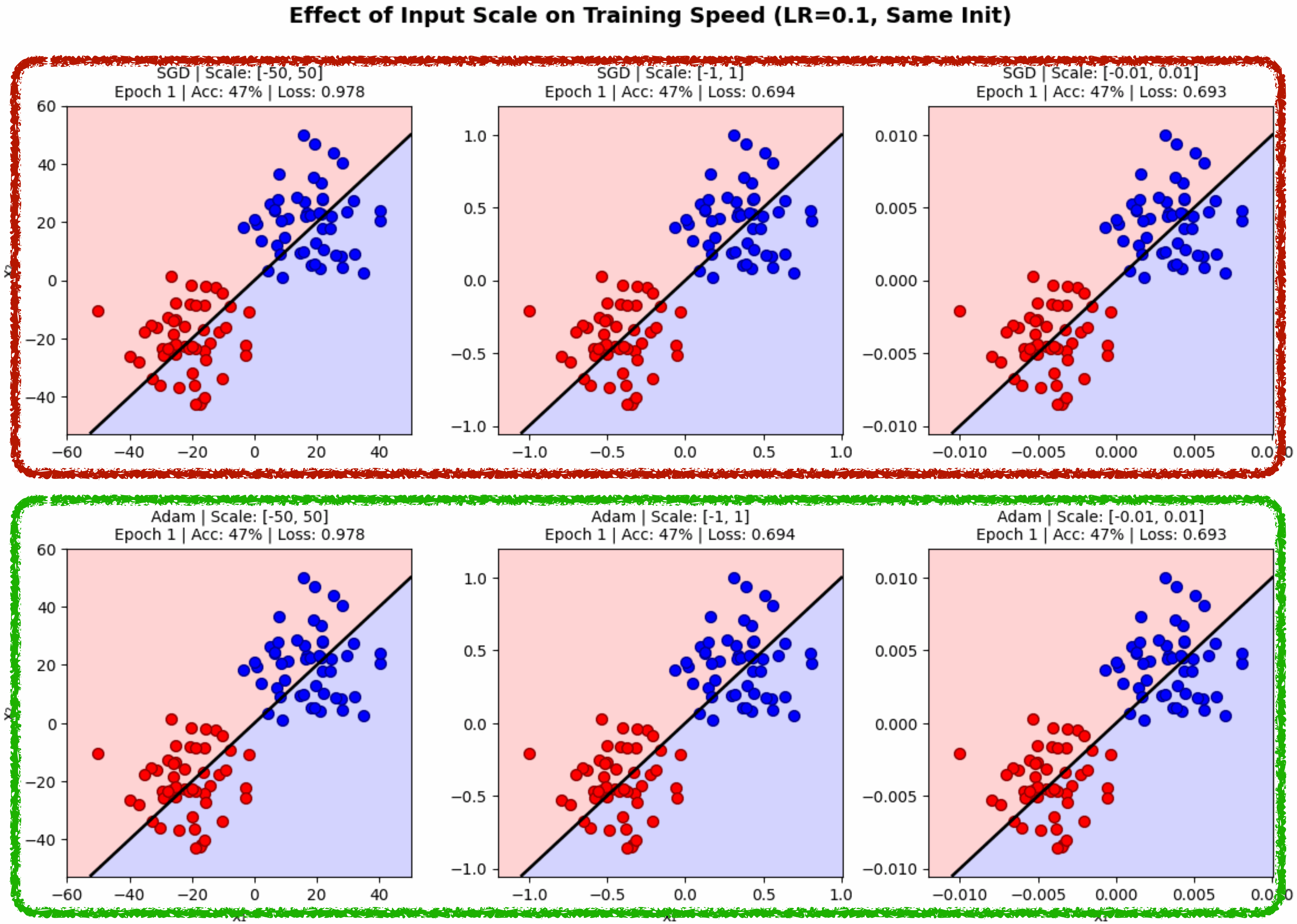
Initialization Brainstorming

Optimizer	Update idea (per step)	Strengths	Weaknesses / Notes
SGD	`θ ← θ − lr * g`	Simple, stable; often good generalization	Requires tuning `lr`; can be slow without momentum
SGD + Momentum	`v ← μv + g`; `θ ← θ − lr * v`	Faster convergence; smooths noisy gradients	Extra hyperparameter (momentum μ)
SGD + Nesterov	Look ahead before gradient step	Often converges faster than vanilla momentum	Slightly more complex; similar tuning issues
Adam	Per-parameter adaptive lr using 1st & 2nd moment estimates	Usually works "out of the box"; good for sparse grads	Can generalize worse than SGD; more hyperparameters
AdamW	Adam + *decoupled* weight decay	Better weight decay behavior; often preferred default	Slightly more to configure (weight_decay)
RMSprop	Scales lr by running avg of squared grads	Good for non-stationary problems; common in RNNs	Can be sensitive to hyperparameters
Adagrad	Accumulates squared grads, shrinking lr over time	Good for sparse features; no manual lr schedule	Learning rate can become too small over long training

Initialization Brainstorming



Initialization Brainstorming



We can’t cross-validate everything…

Initialization Brainstorming

$$\mathbf{W}^{(3)}\mathrm{ReLU}(\mathbf{W}^{(2)}\mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$

Initialization Brainstorming

$$\mathbf{W}^{(3)}\text{ReLU}(\mathbf{W}^{(2)}\text{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$

What would be a good/bad initialization?

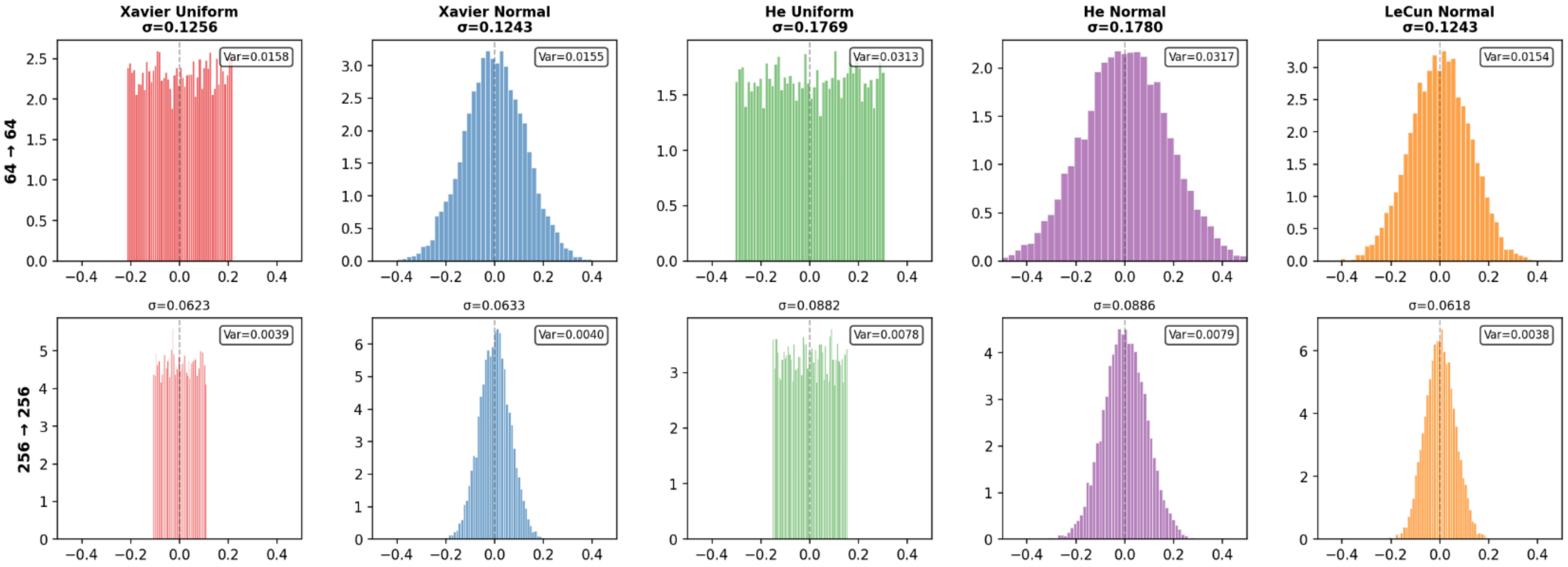
Initialization Brainstorming

$$\mathbf{W}^{(3)}\mathrm{ReLU}(\mathbf{W}^{(2)}\mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \mathbf{b}^{(3)}$$

What would be a good/bad initialization?

Assume $\mathrm{Var}(\mathbf{x}) = 1$, find σ so that with $\mathbf{W}^{(1)}_{i,j} \sim \mathcal{N}(0,\sigma)$ we have $\mathrm{Var}\left(\mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{x})\right) = 1$

Initialization Brainstorming



What are our hyper-parameters so far?

- Number of layers (L), width of each layer
- learning rate
- Mini-batch size
- Training steps

What are our hyper-parameters so far?

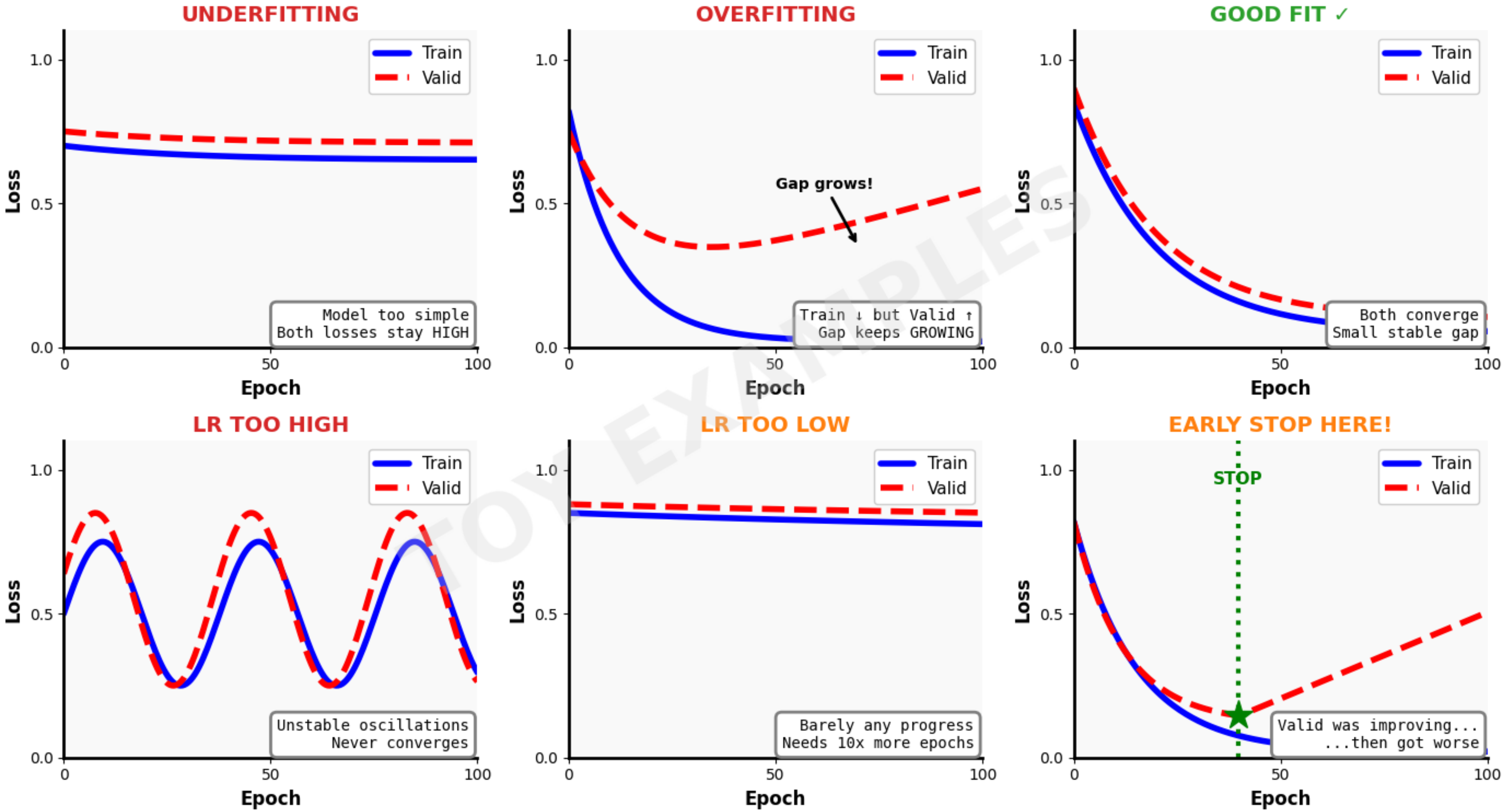
- Number of layers (L), width of each layer
- learning rate
- Mini-batch size
- Training ste

Cross-validate with your train/valid/test split!

General Tips

- Start with some random exploration
- Once you have a “okay” solution:
  - Change one thing at a time
  - Think carefully about ranges of hparams and inter-play (normalization, lr)
  - Keep in mind that many hparams don’t transfer

General Tips



Announcement

https://world-model-mila.github.io/

World Modeling Workshop

Date: 4-6 February 2026
Location: Agora, Mila - Quebec AI Institute, 6666 Rue Saint-Urbain, Montréal, Canada
Online: Public streaming (freely available to all) (link TBA)

Free stream on YouTube and X!

Yoshua Bengio
LawZero, Mila, UdeM

Yann LeCun
AMI Labs, NYU

Sherry Yang
GDM, NYU

Shirley Ho
Polymathic, Flatiron, NYU

Jürgen Schmidhuber
The Swiss AI Lab, KAUST

Amir Zadeh
Lambda

Questions?