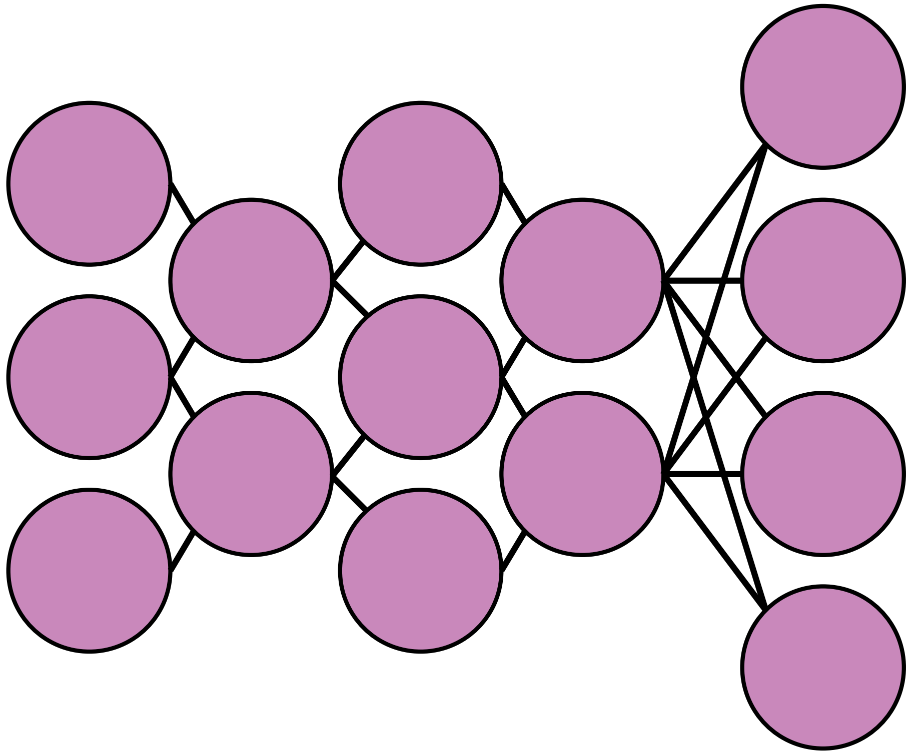
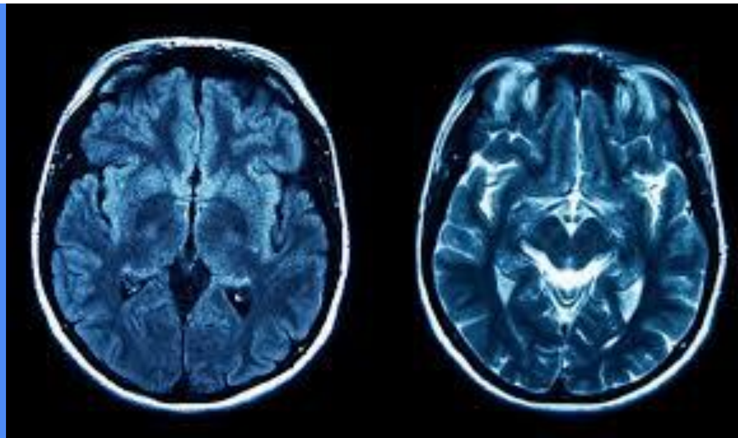
Accelerating Deep Learning with the Biggest Losers

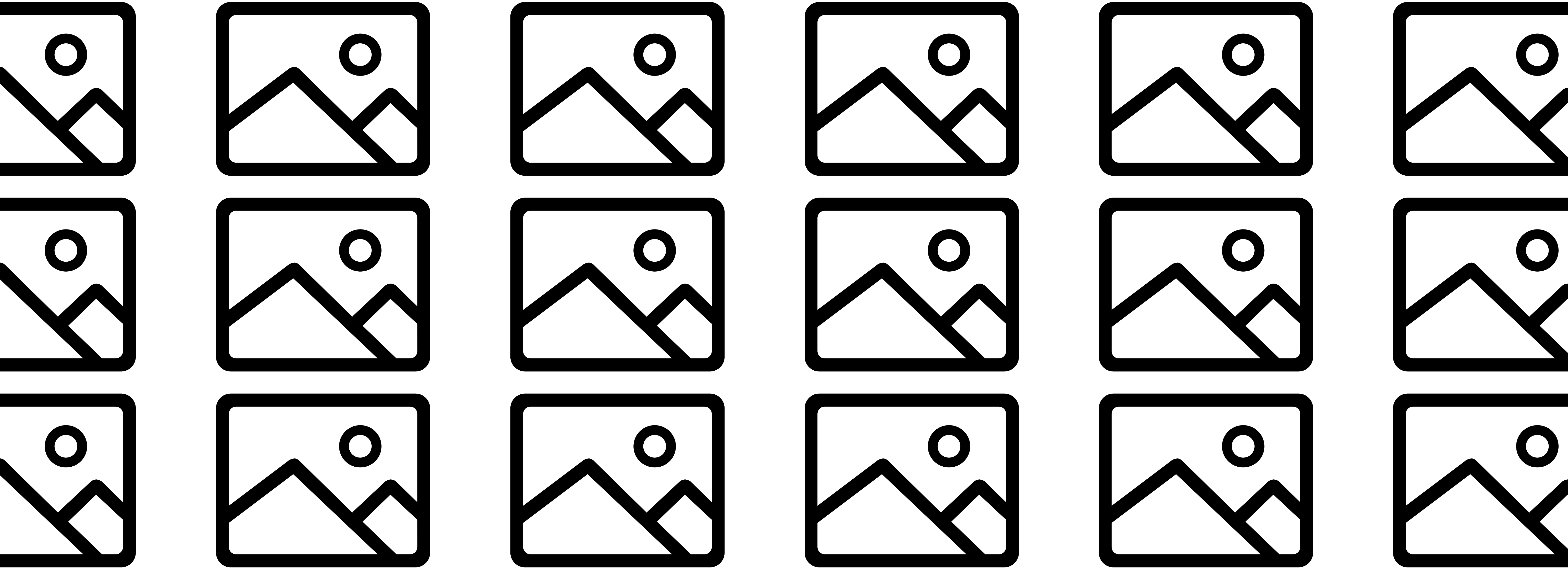
Angela H. Jiang, Daniel L.-K. Wong, Giulio Zhou, David G. Andersen, Jeffrey Dean, Gregory R. Ganger, Gauri Joshi, Michael Kaminsky, Michael A. Kozuch, Zachary C. Lipton, Padmanabhan Pillai

Deep learning enables emerging applications

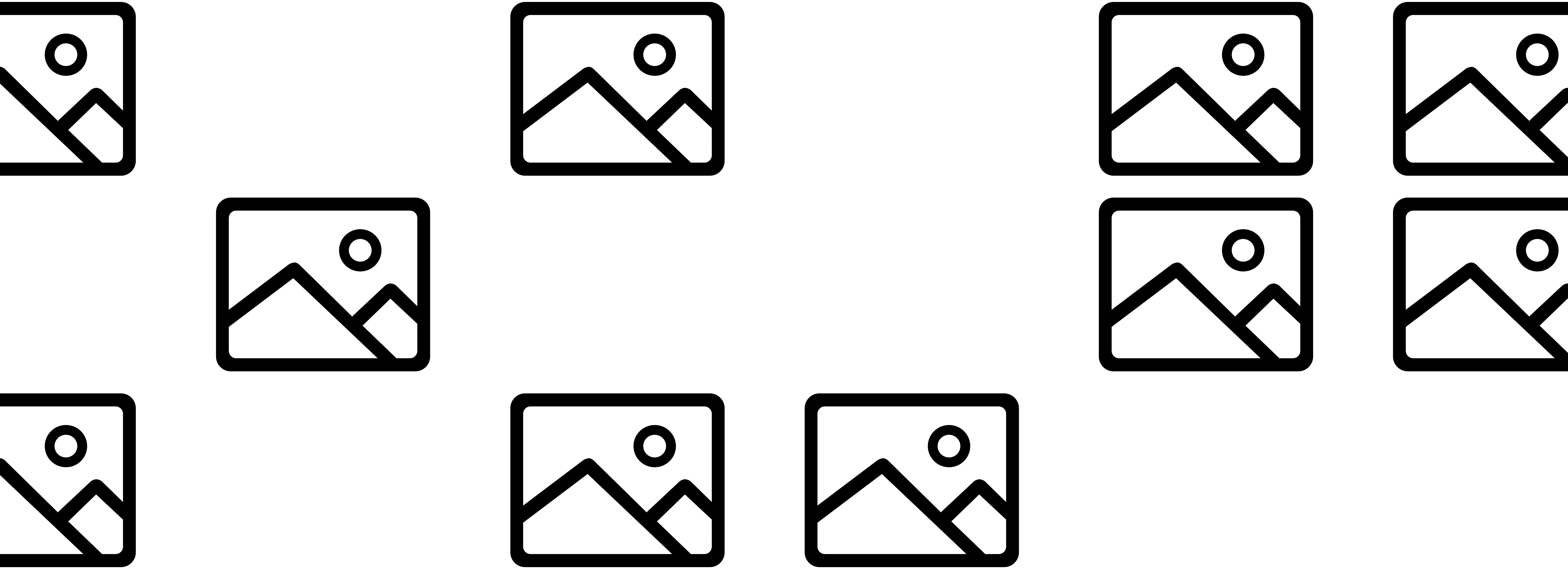




DNN training analyzes many examples



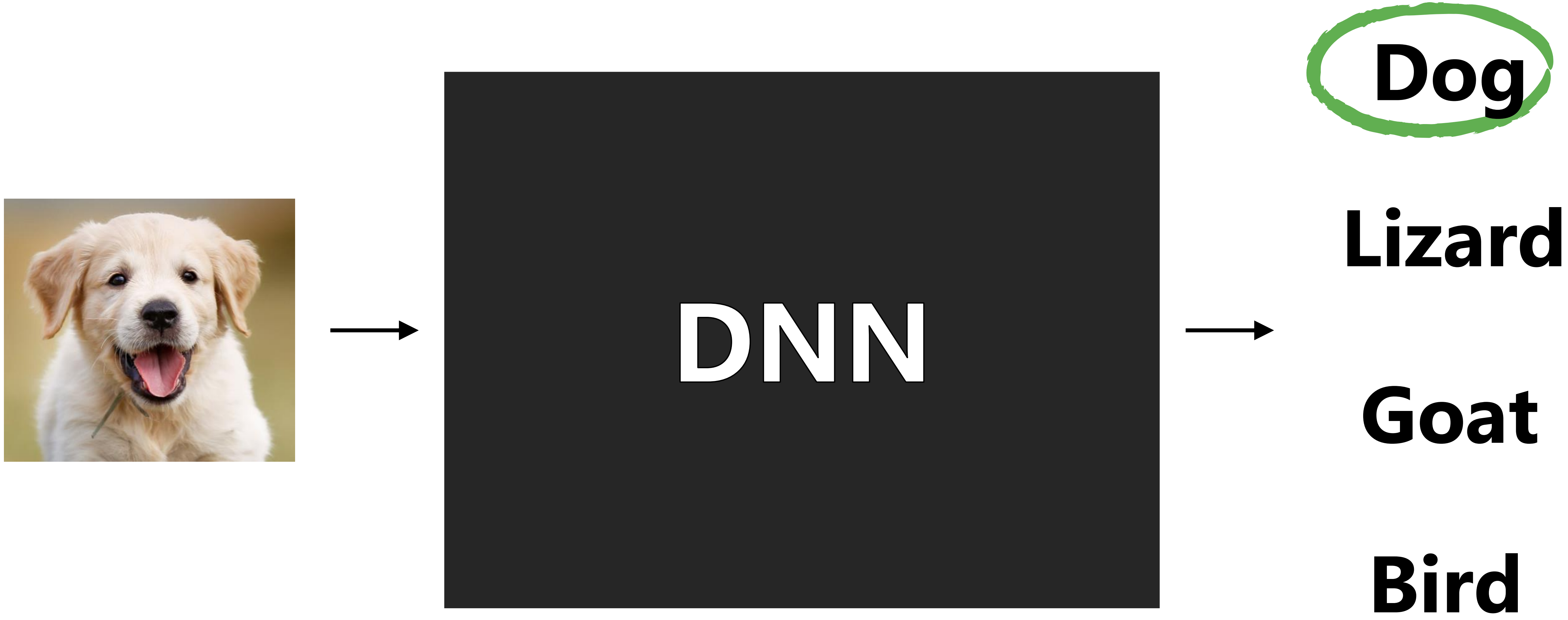
Selective-Backprop prioritizes informative examples



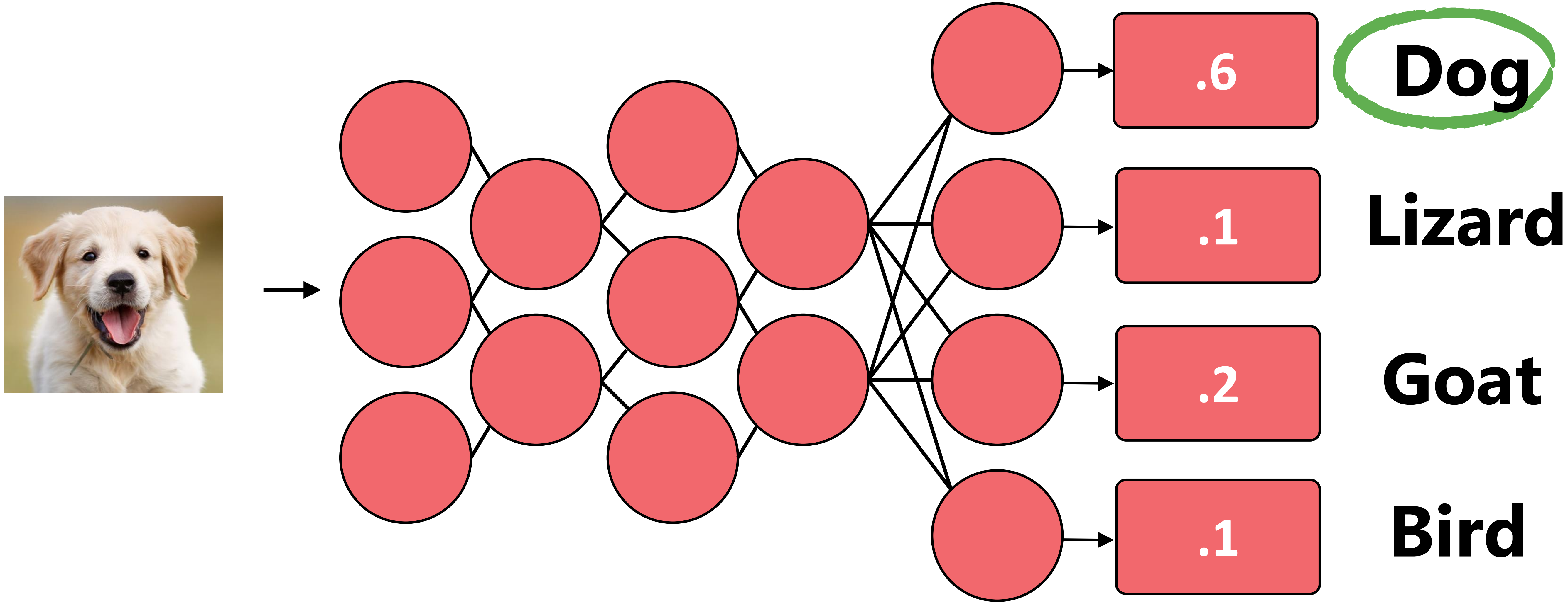
DNN basics

How to use and train a DNN

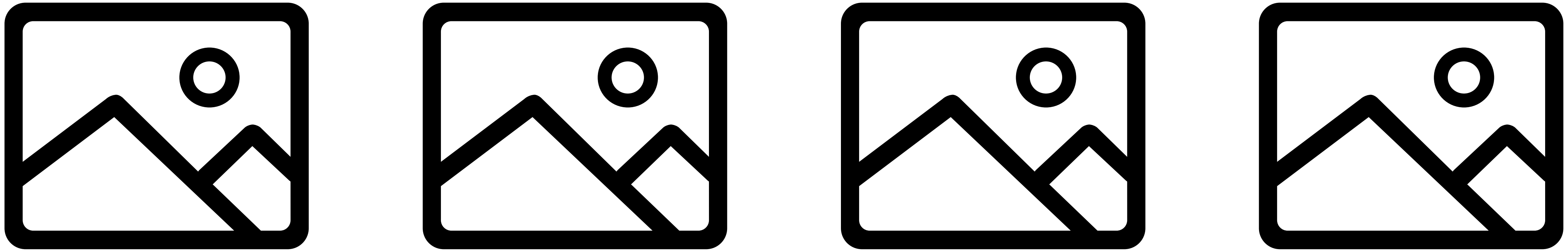
Example task: Image classification



DNN inference: From image to “Dog”

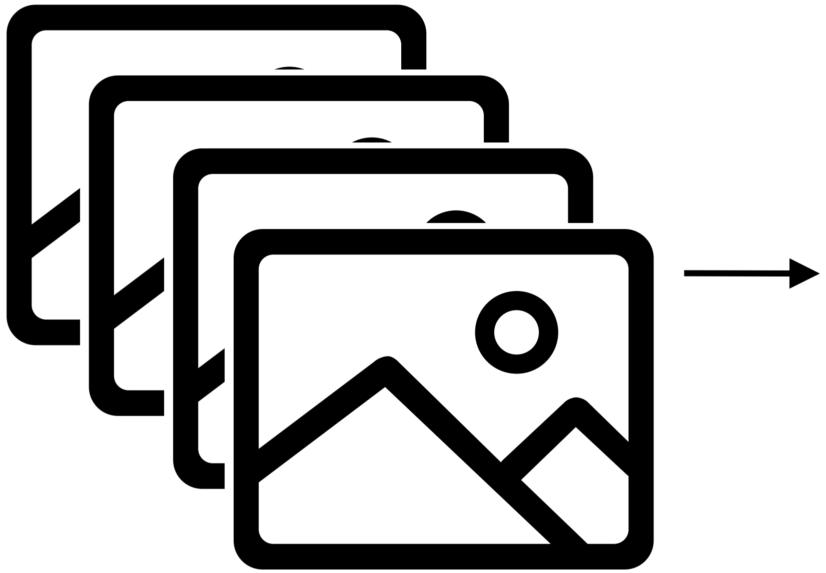


Training DNNs relies on a labeled dataset

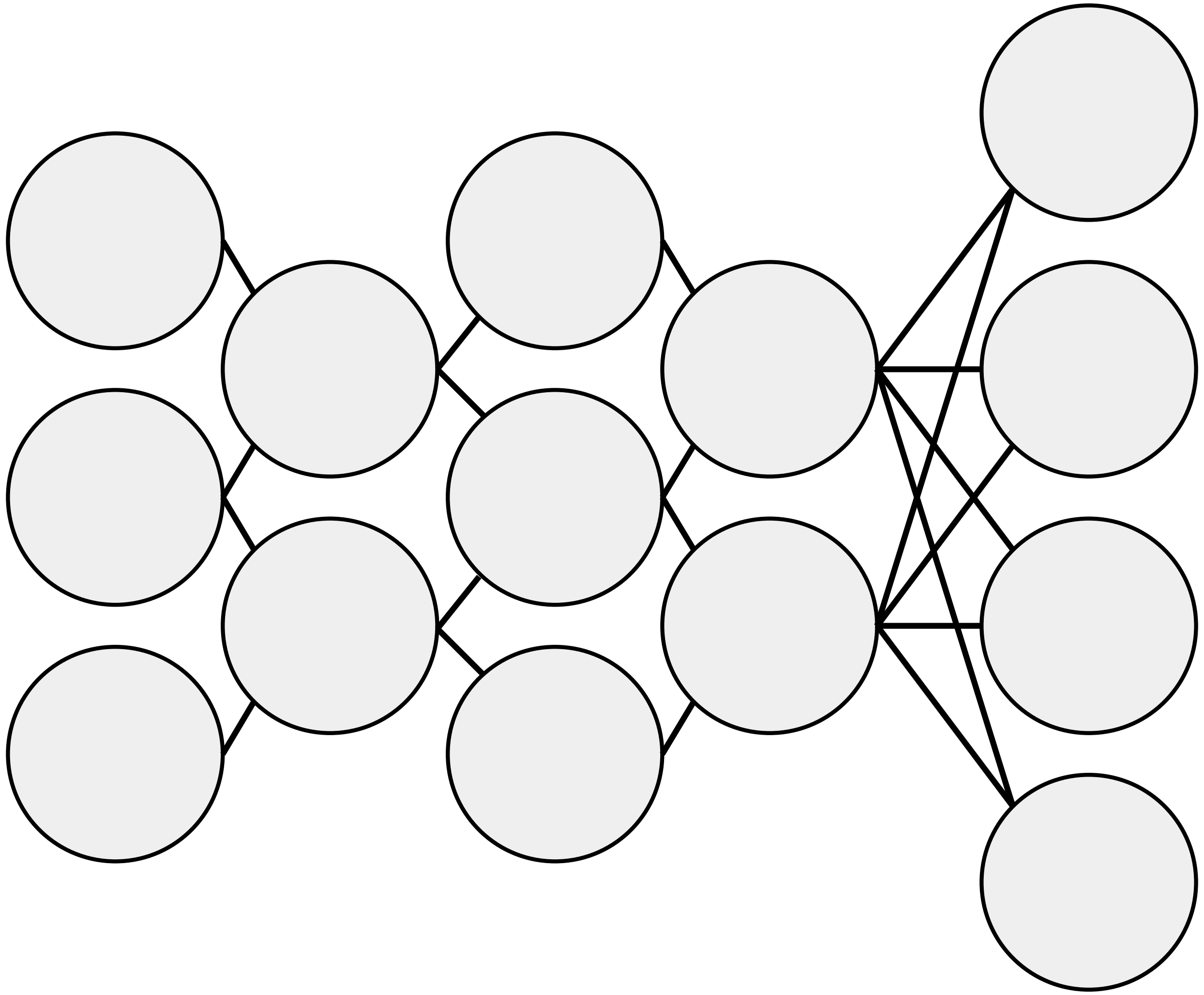


Class: Dog | **Class: Bird** | **Class: Lizard** | **Class: Dog**

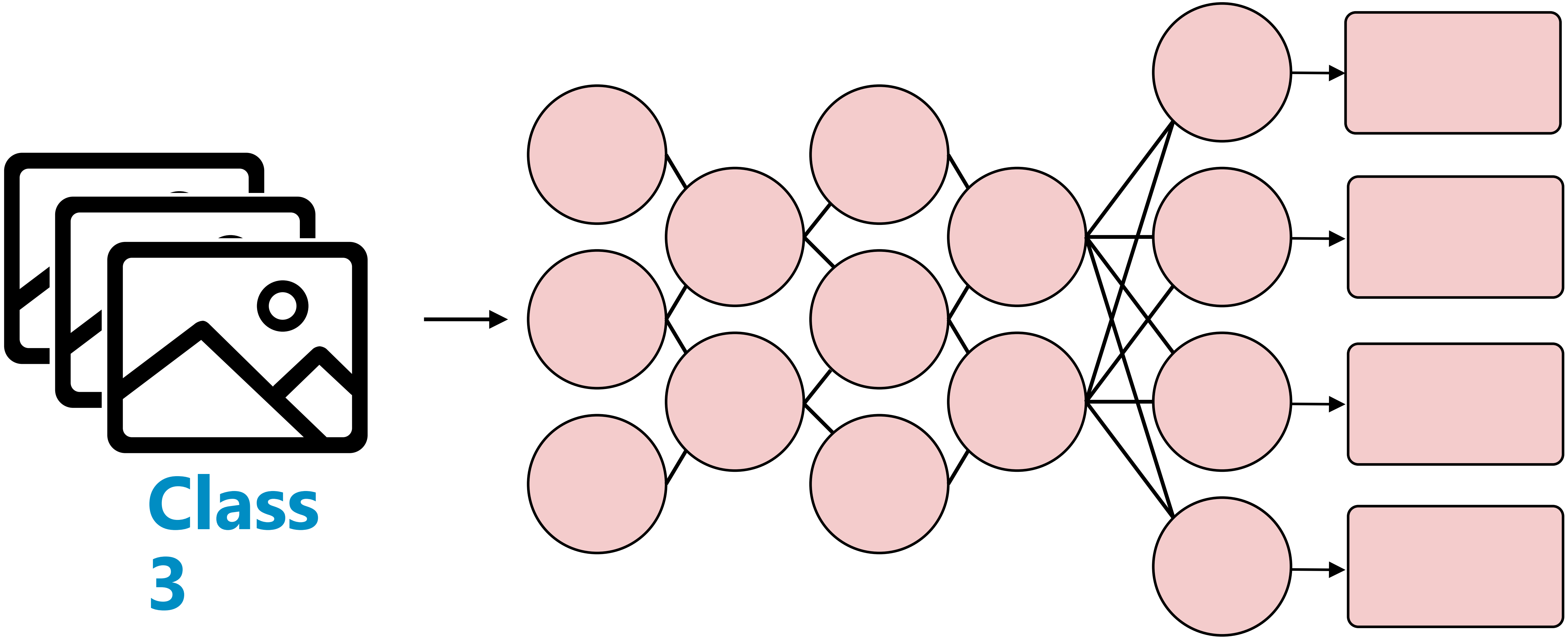
DNN training: Determining the weights



Class 1

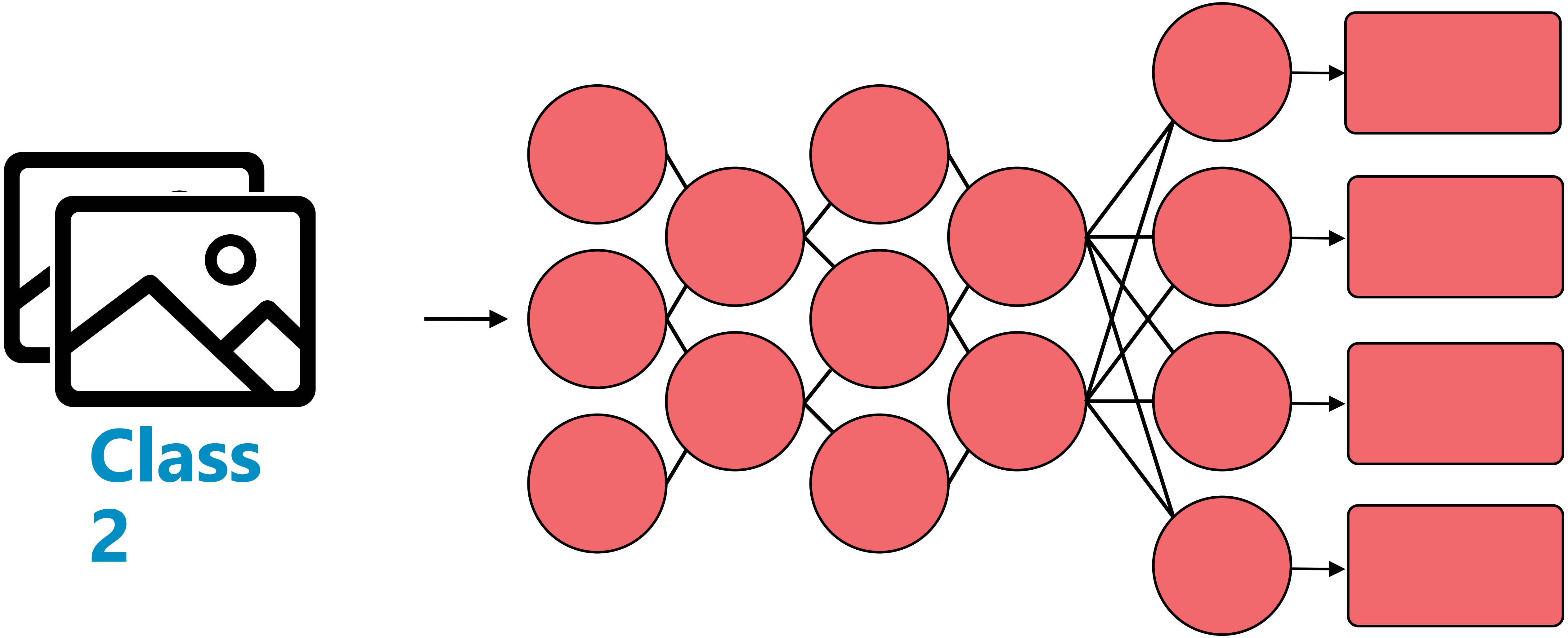


DNN training: Determining the weights via backpropagation

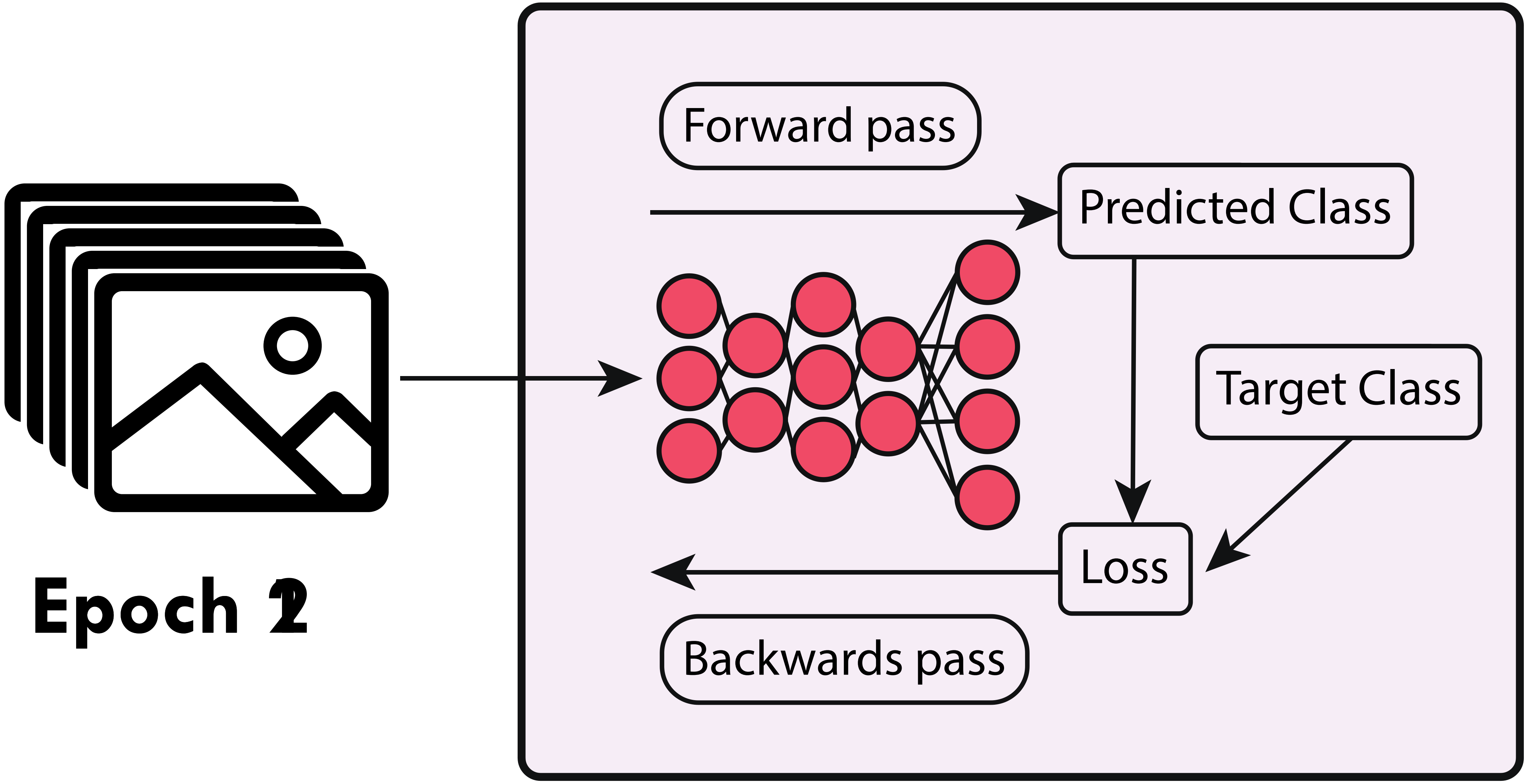


Class 3

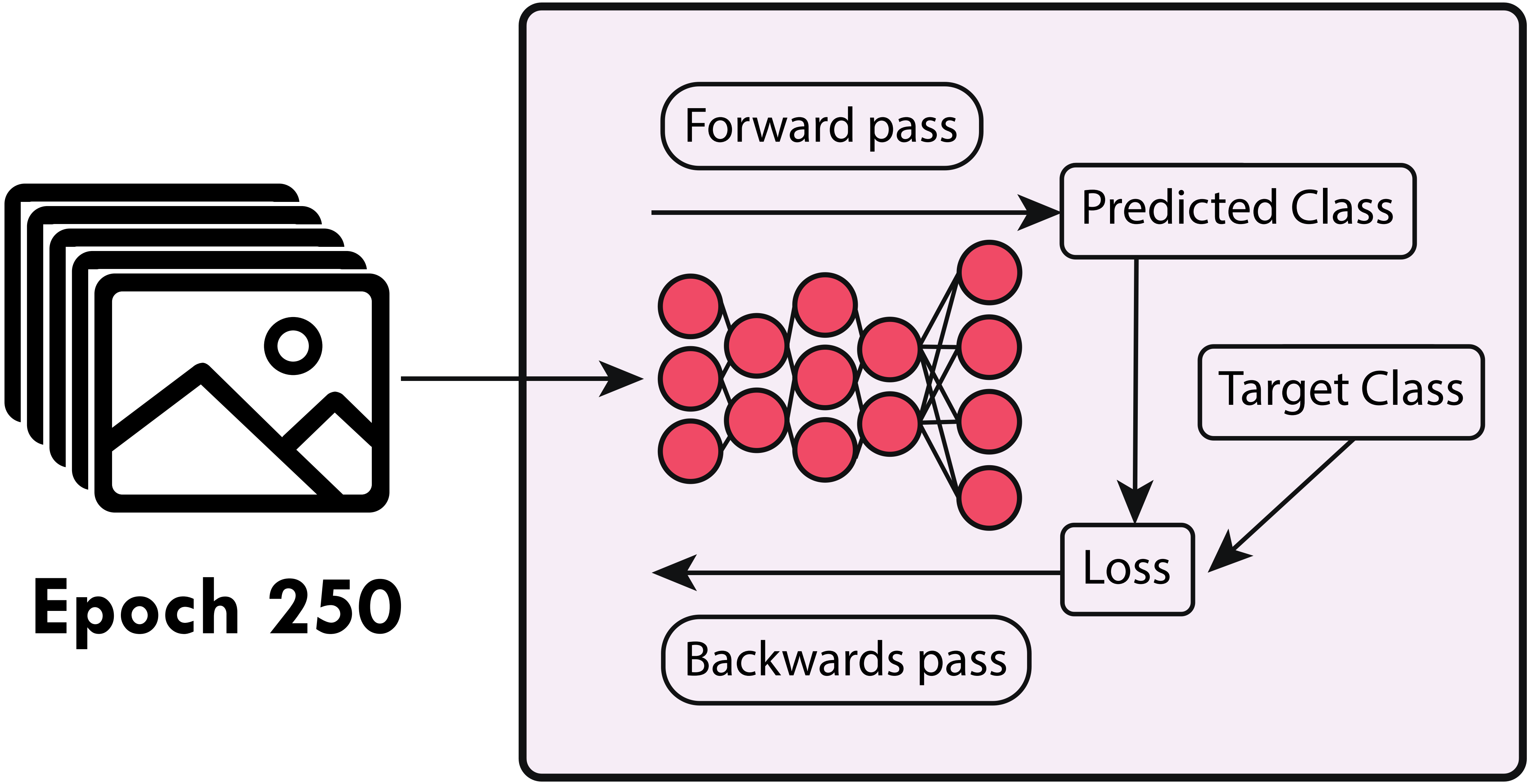
DNN training: Determining the weights via backpropagation



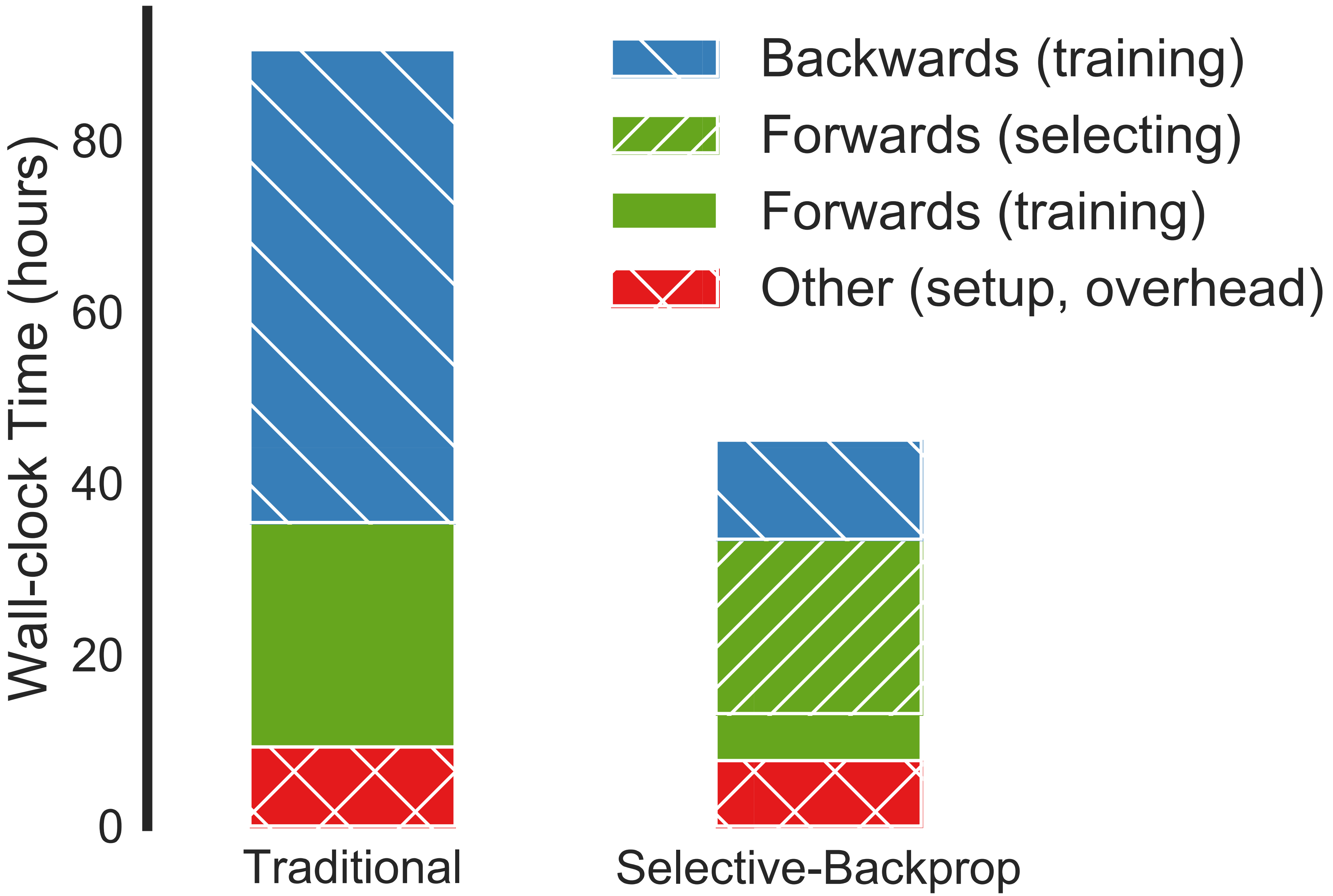
DNN training analyzes an example many times



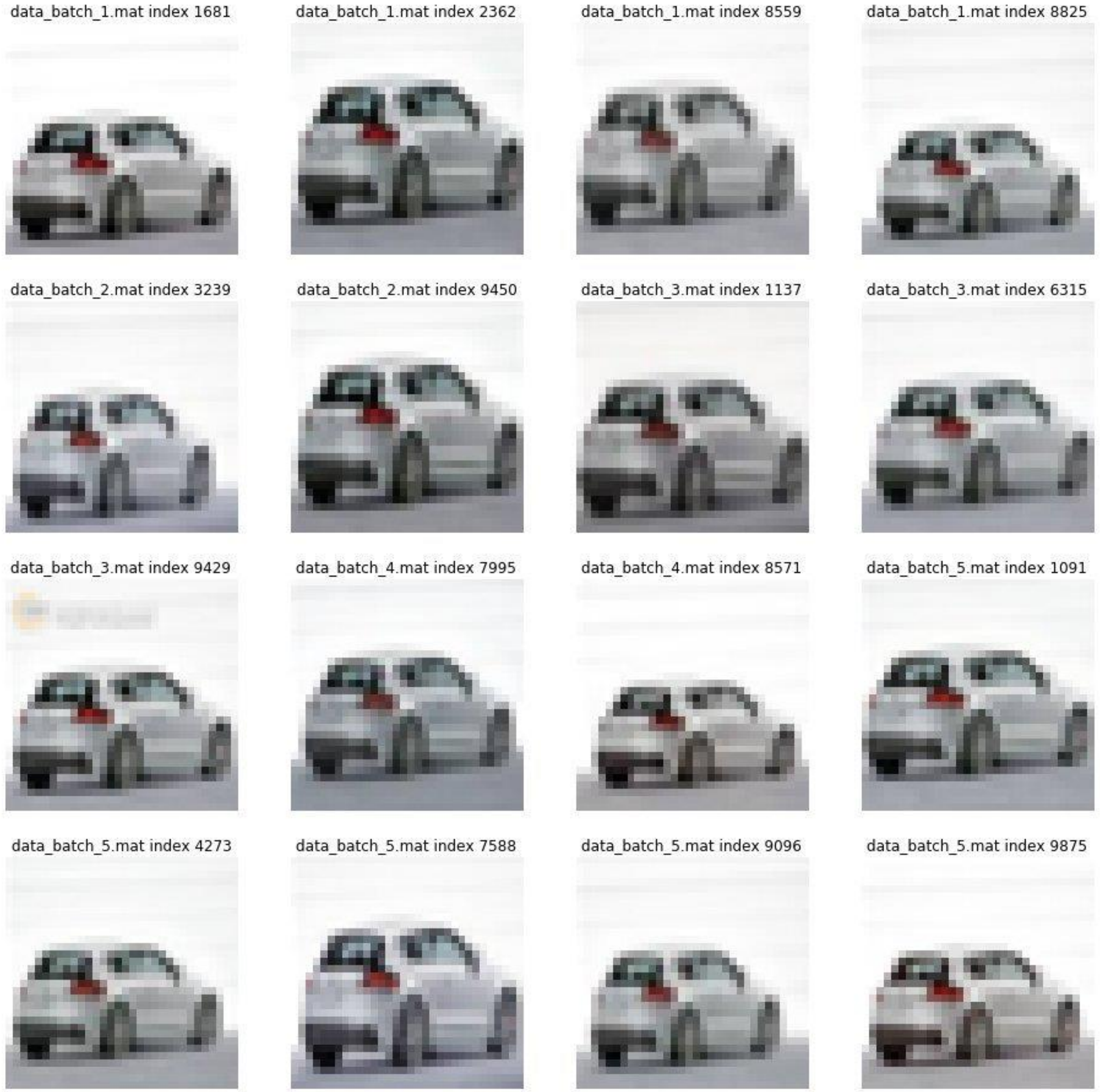
DNN training analyzes an example many times



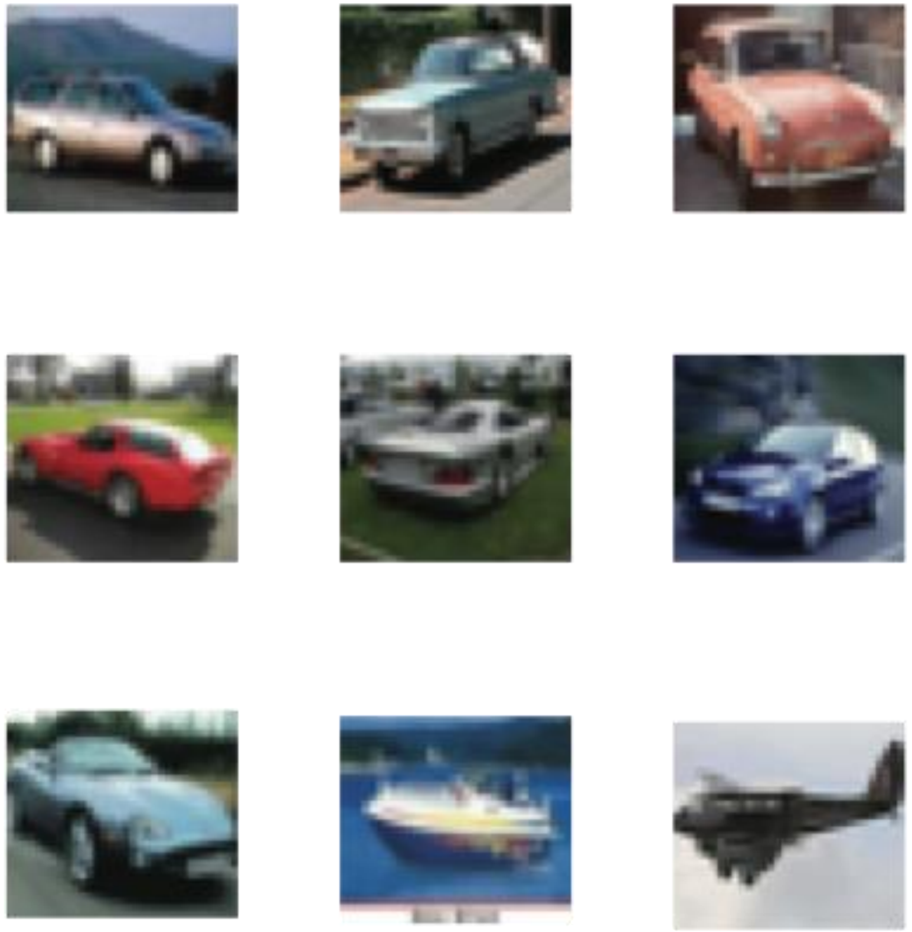
SelectiveBackprop targets slowest part of training



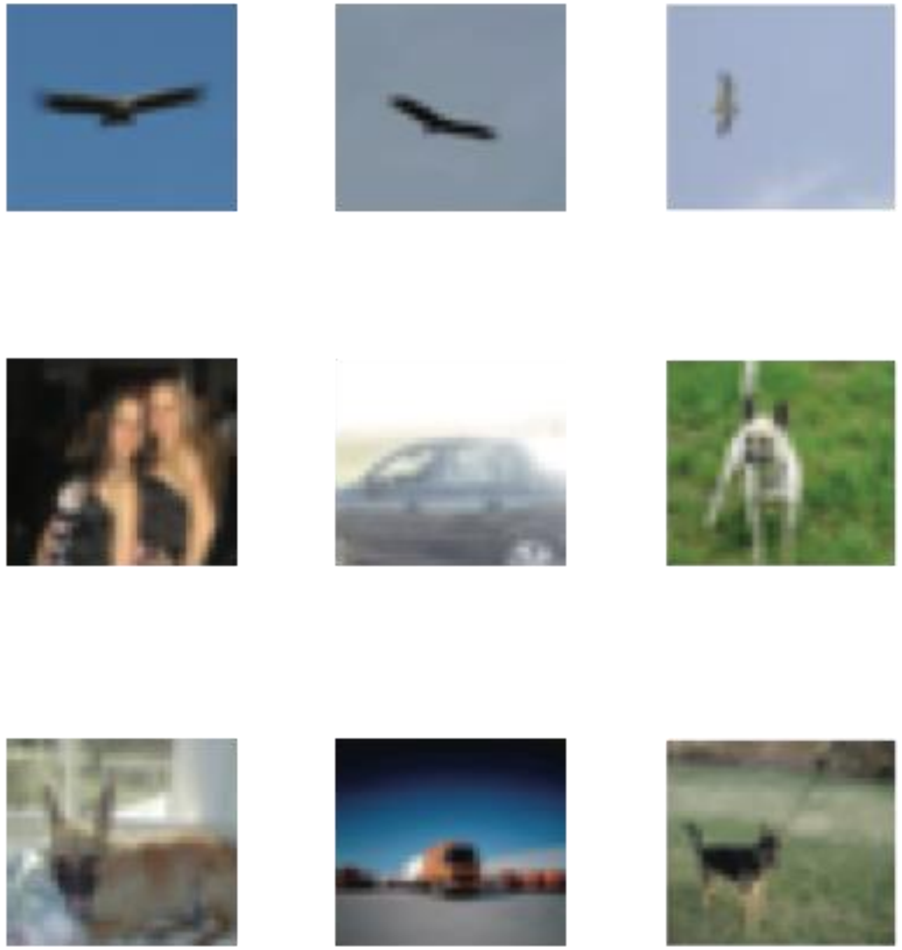
Not all examples are equally useful



Prioritize examples with high loss



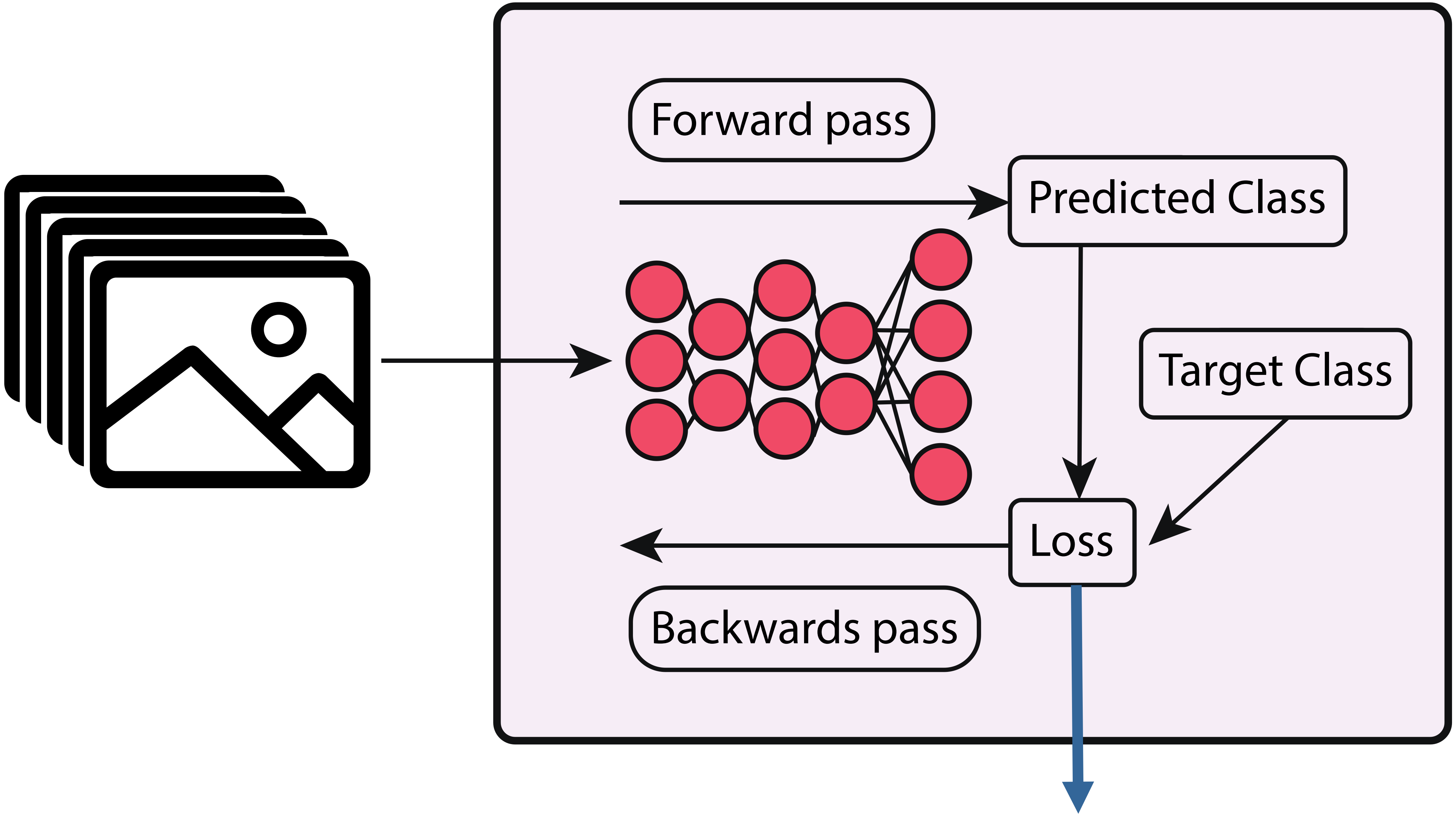
Examples with low loss



Examples with high loss

Selective Backprop algorithm

DNN training analyzes an example many times



Bad idea #1:
Deciding with a hard threshold

if loss > threshold: backprop()

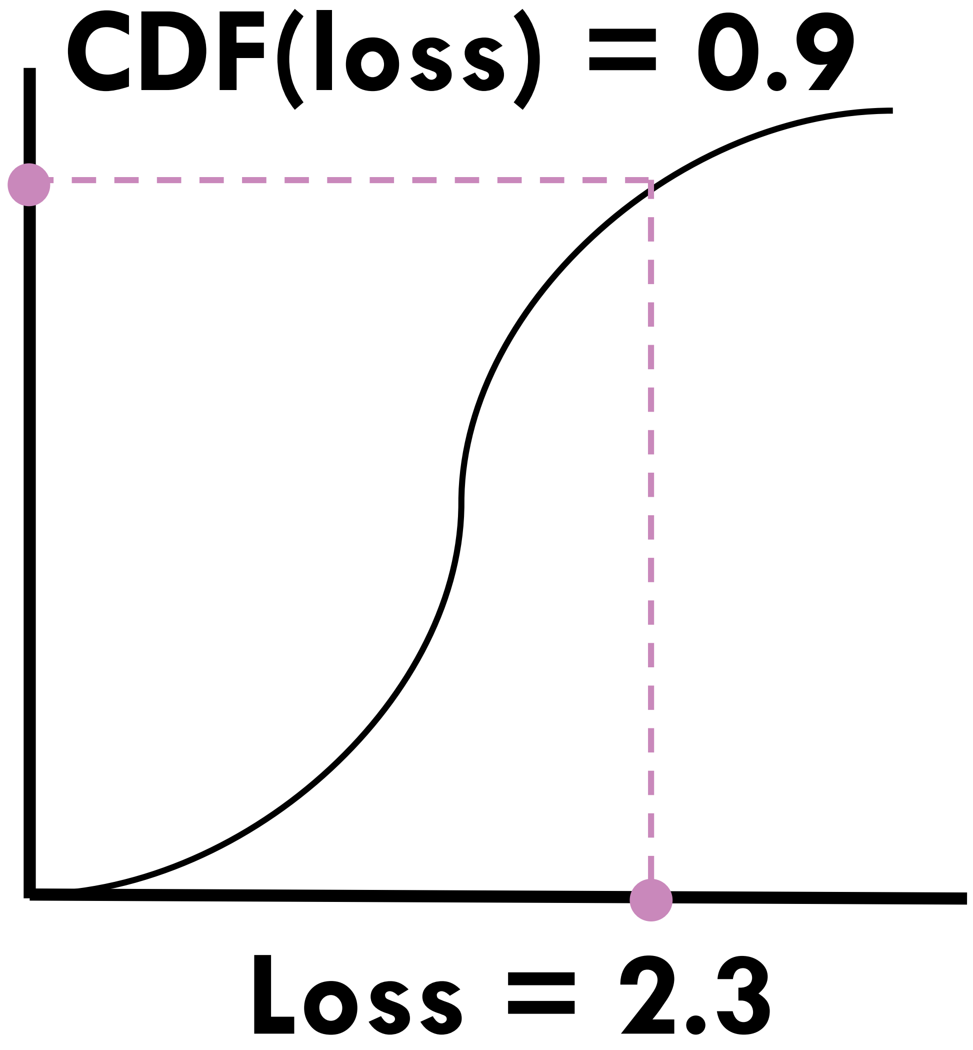
Bad idea #2:
Deciding probabilistically with absolute loss

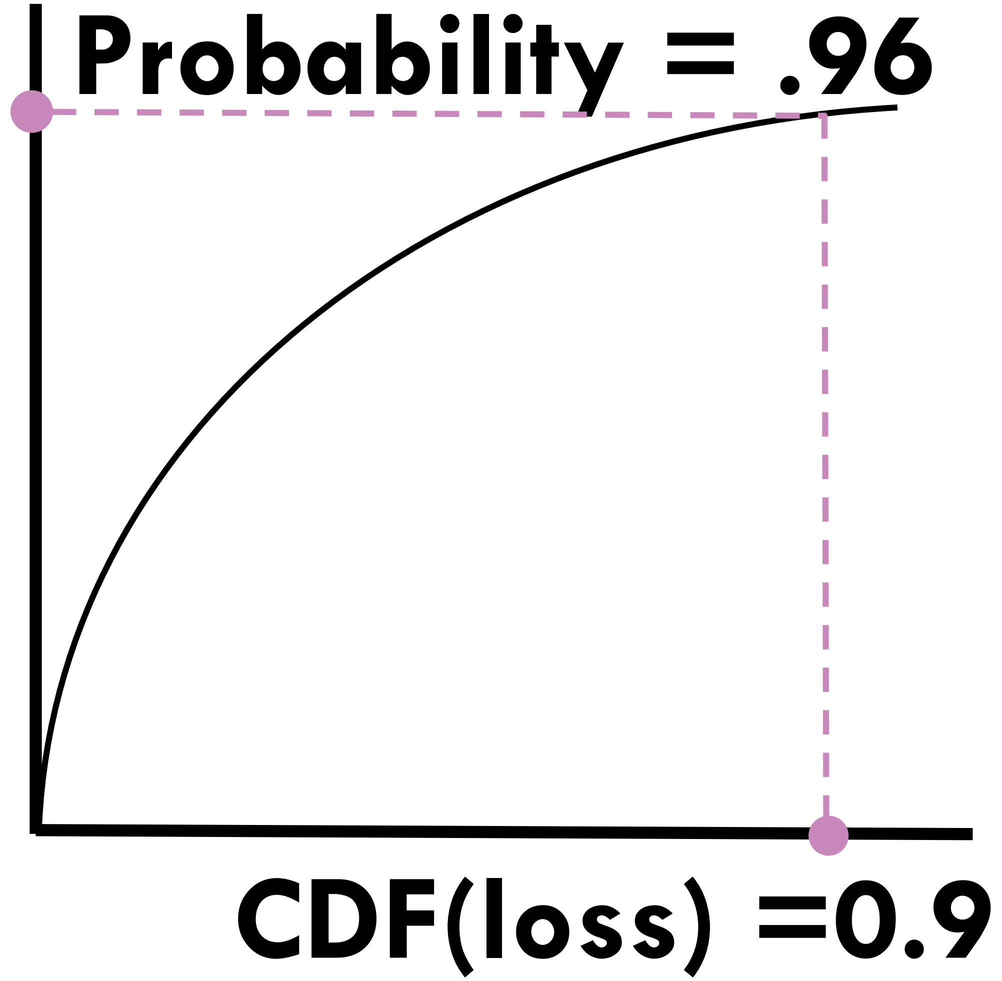
$$P(backprop) = normalize(loss, 0, 1)$$

Good idea:
Use relative probabilistic calculation

$$P(backprop) = Percentile(loss, recent\ losses)^B$$

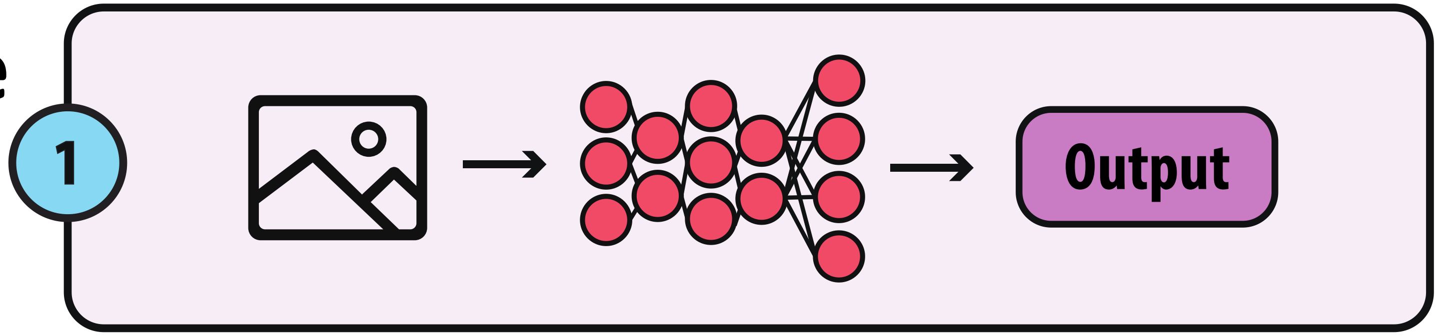
Example of probability calculation



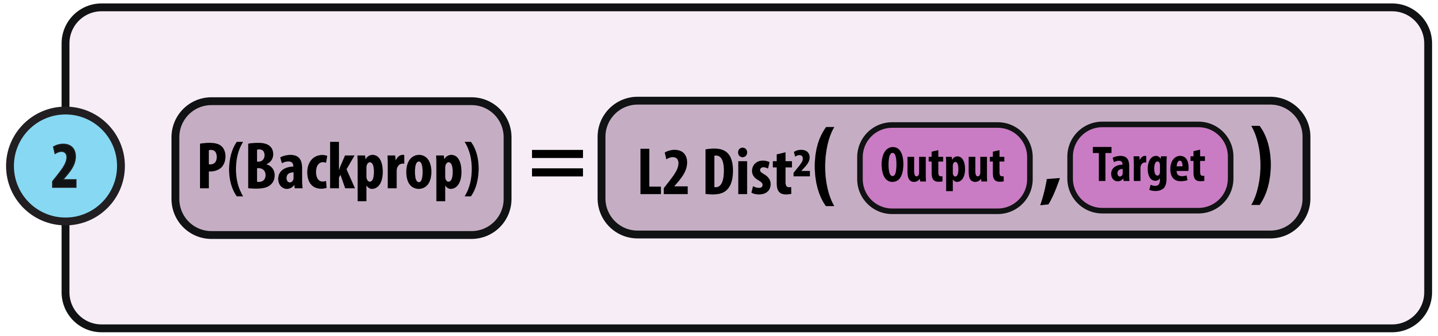


Selective-Backprop approach

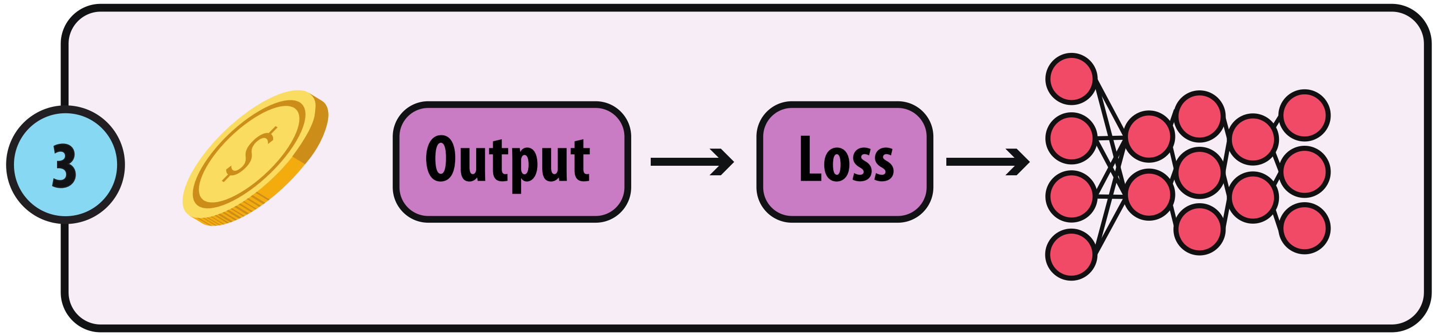
Forward propagate example through the network



Calculate usefulness of backpropping example based on its accuracy

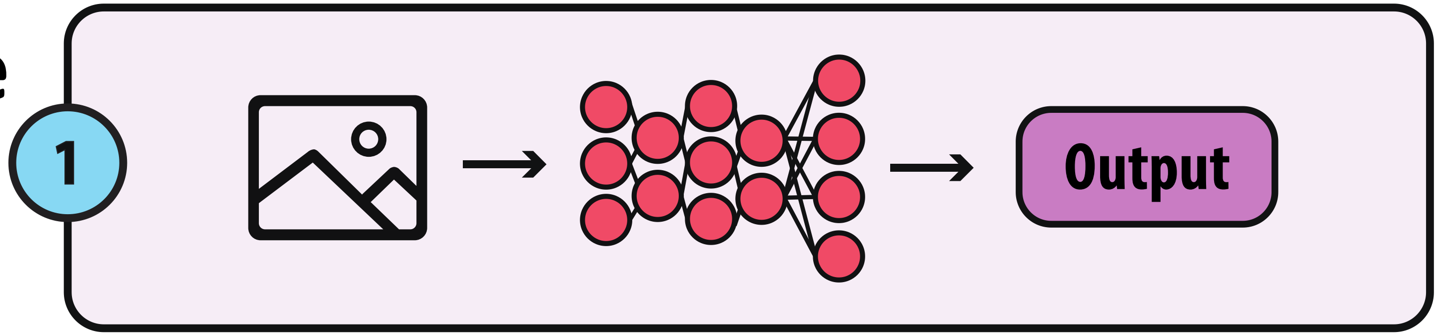


Decide probabilistically if we should backprop

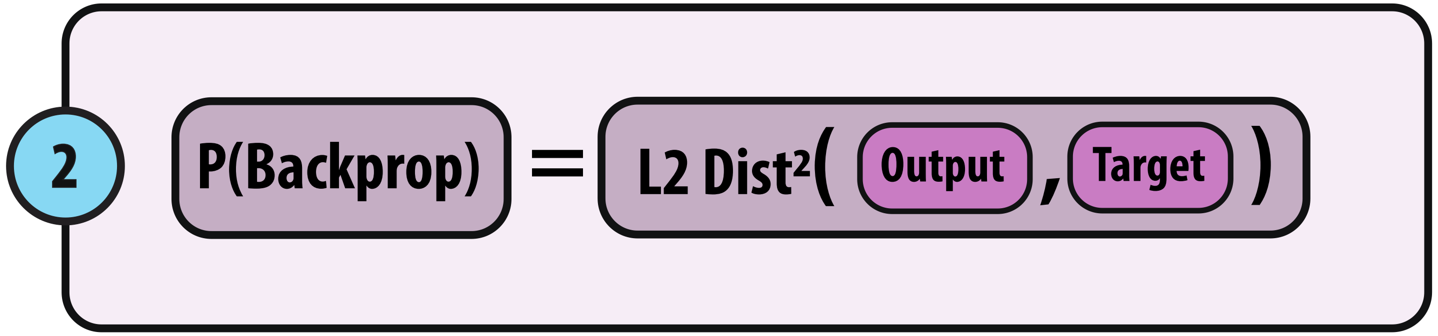


StaleSB reduces forward passes

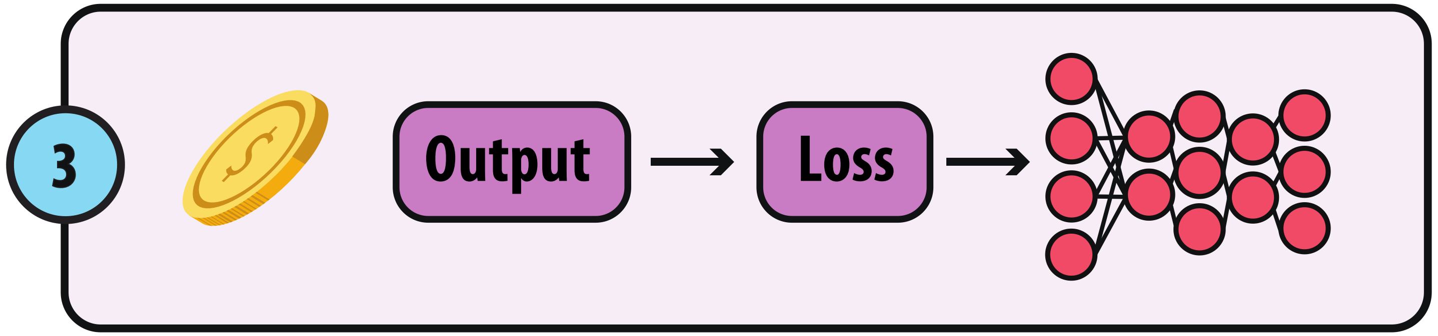
Forward propagate example through the network ***every n epochs***



Calculate usefulness of backpropping example based on its accuracy

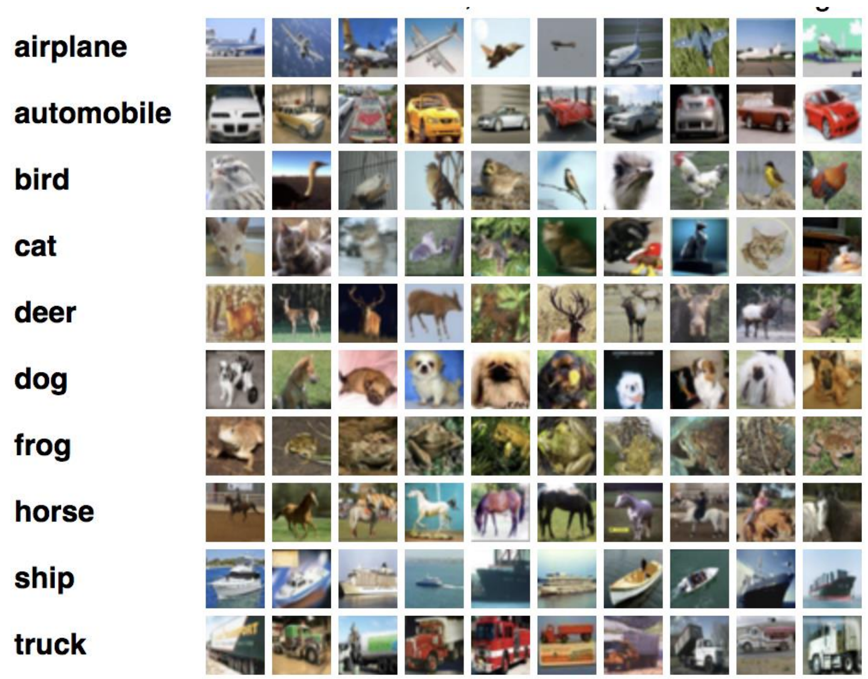


Decide probabilistically if we should backprop



Evaluation of Selective Backprop

Datasets



CIFAR10

60,000 Training Images

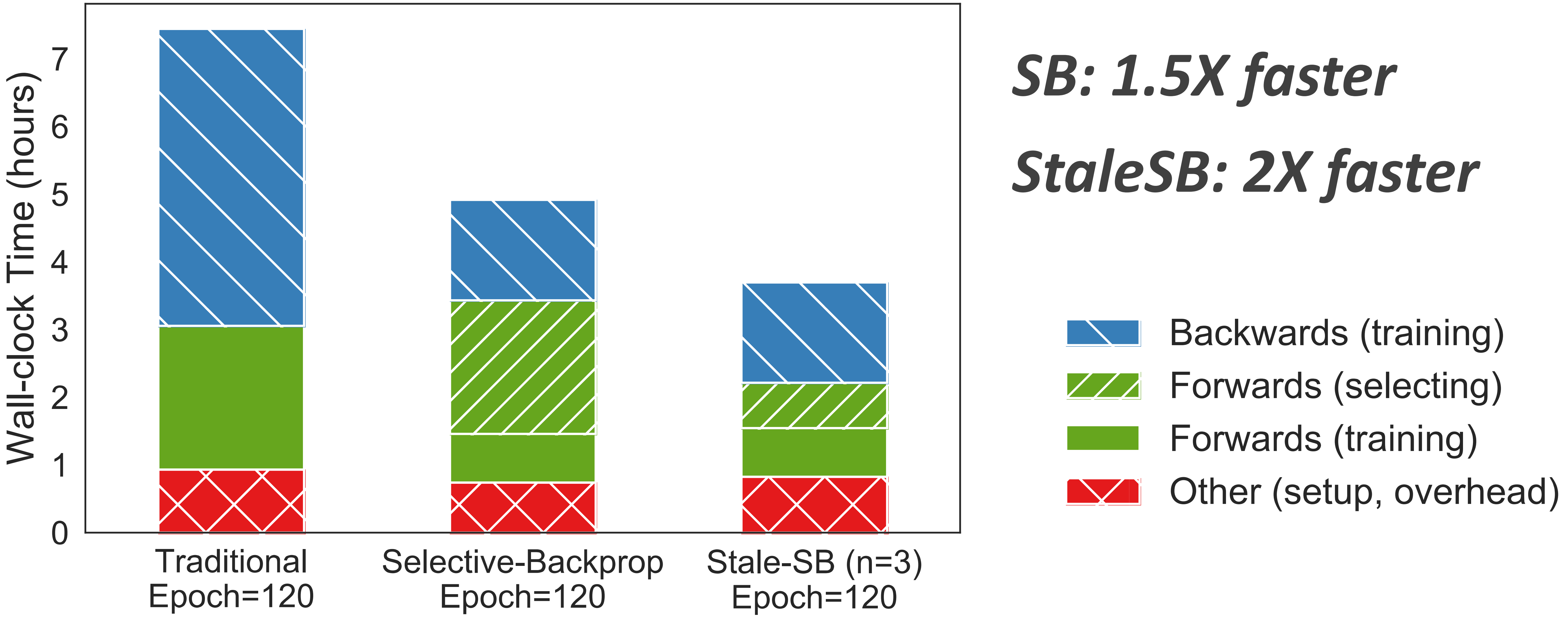
CIFAR100

60,000 Training Images

SVHN

604,388 Training Images

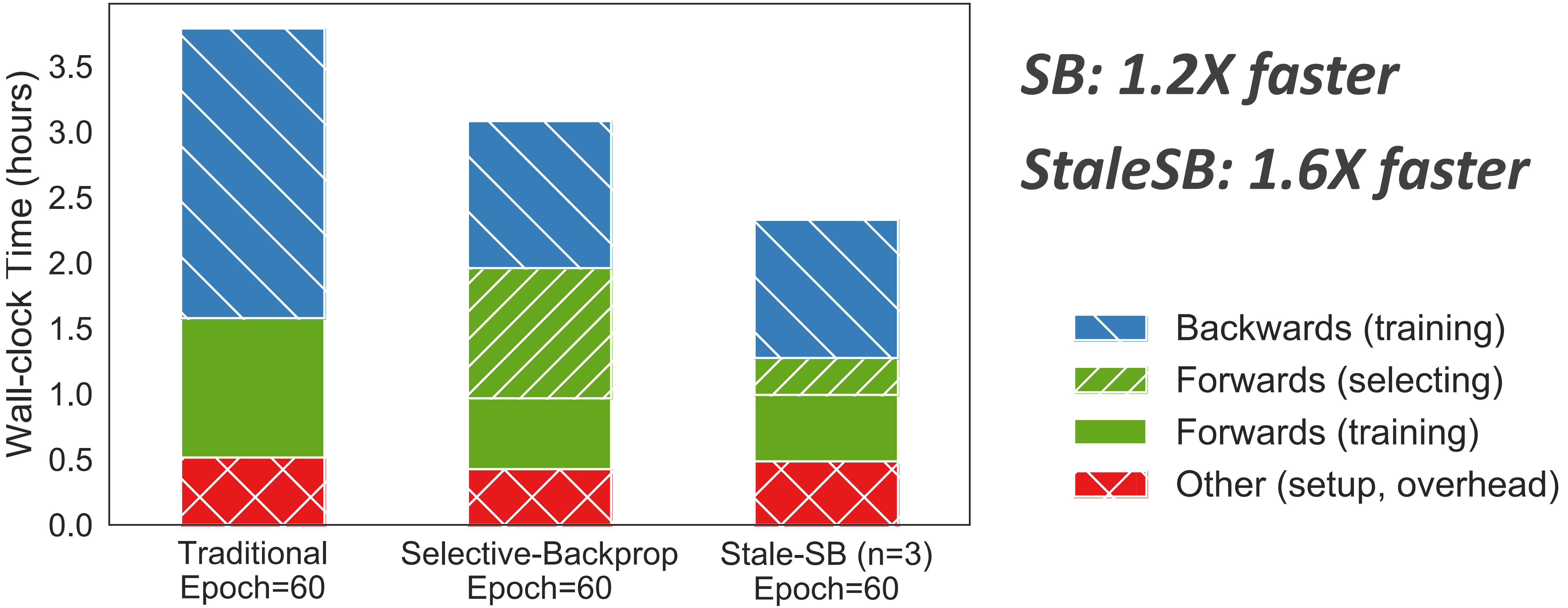
Train CIFAR10 to 4.14% (1.4x Traditional's final error)



SB: 1.5X faster

StaleSB: 2X faster

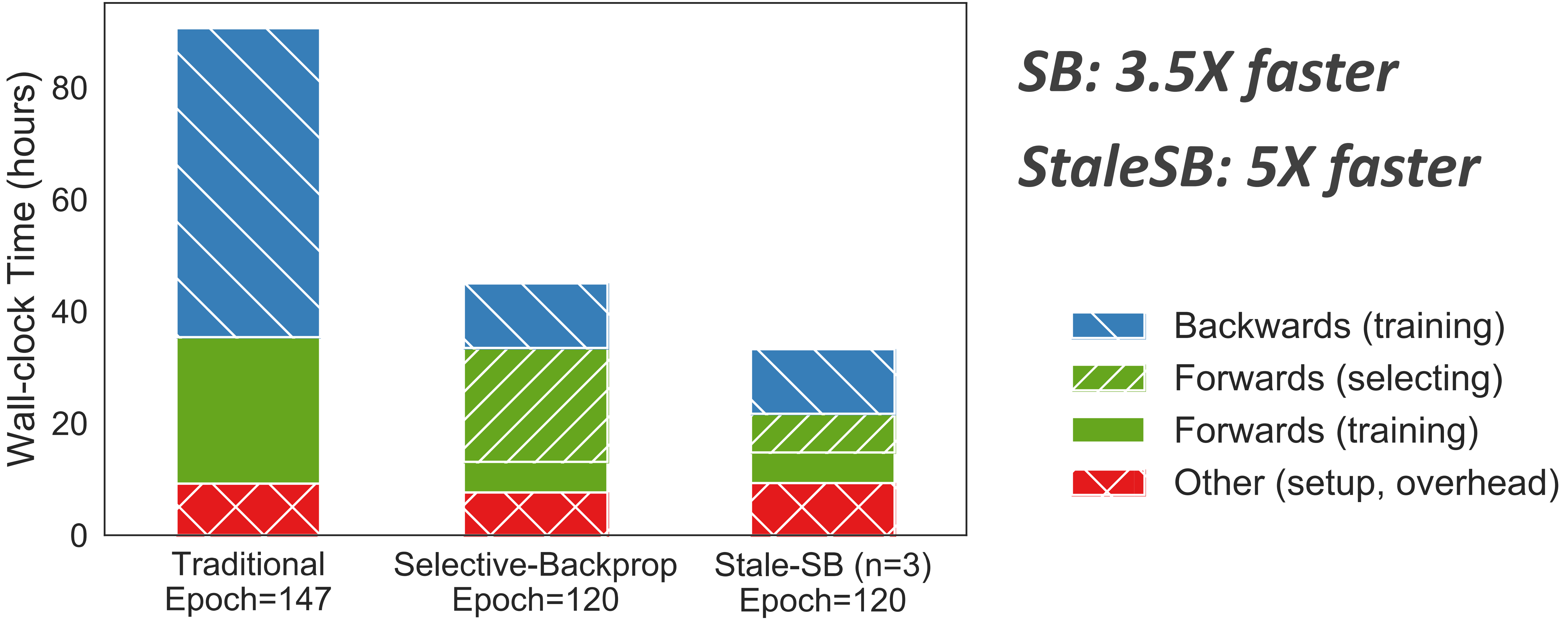
Train CIFAR100 to 25.5% (1.4x Traditional's final error)



SB: 1.2X faster

StaleSB: 1.6X faster

Train SVHN to 1.72% (1.4x Traditional's final error)



SB: 3.5X faster

StaleSB: 5X faster

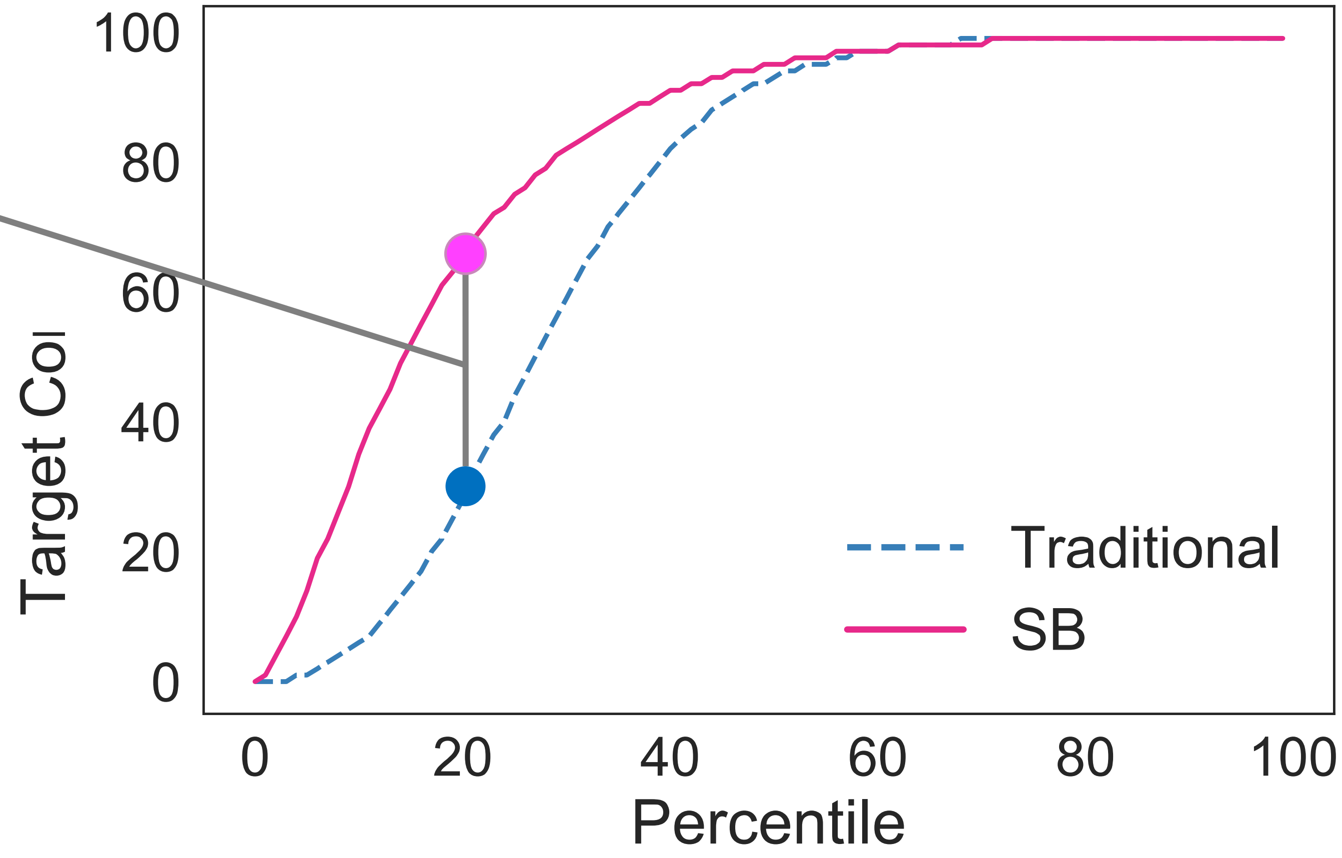
SB on CIFAR10 targets hard examples

3% correct w/ Traditional

29% correct w/ SB

Output = [0.1, 0.3, 0.6]

Target confidence = 0.3



✓ Selective-Backprop accelerates training

Reduces time spent in the backwards pass by prioritizing high-loss examples

✓ SelectiveBackprop outperforms static approaches

Trains up to 3.5x faster compared to standard SGD

Trains 1.02-1.8X faster than state-of-the-art importance sampling approach

✓ Stale-SB further accelerates training

Trains on average 26% faster compared to SB

www.github.com/angelajiang/SelectiveBackprop

Compared approaches

Traditional

Classic SGD with no filtering

Katharopoulos18

State of the art importance sampling approach

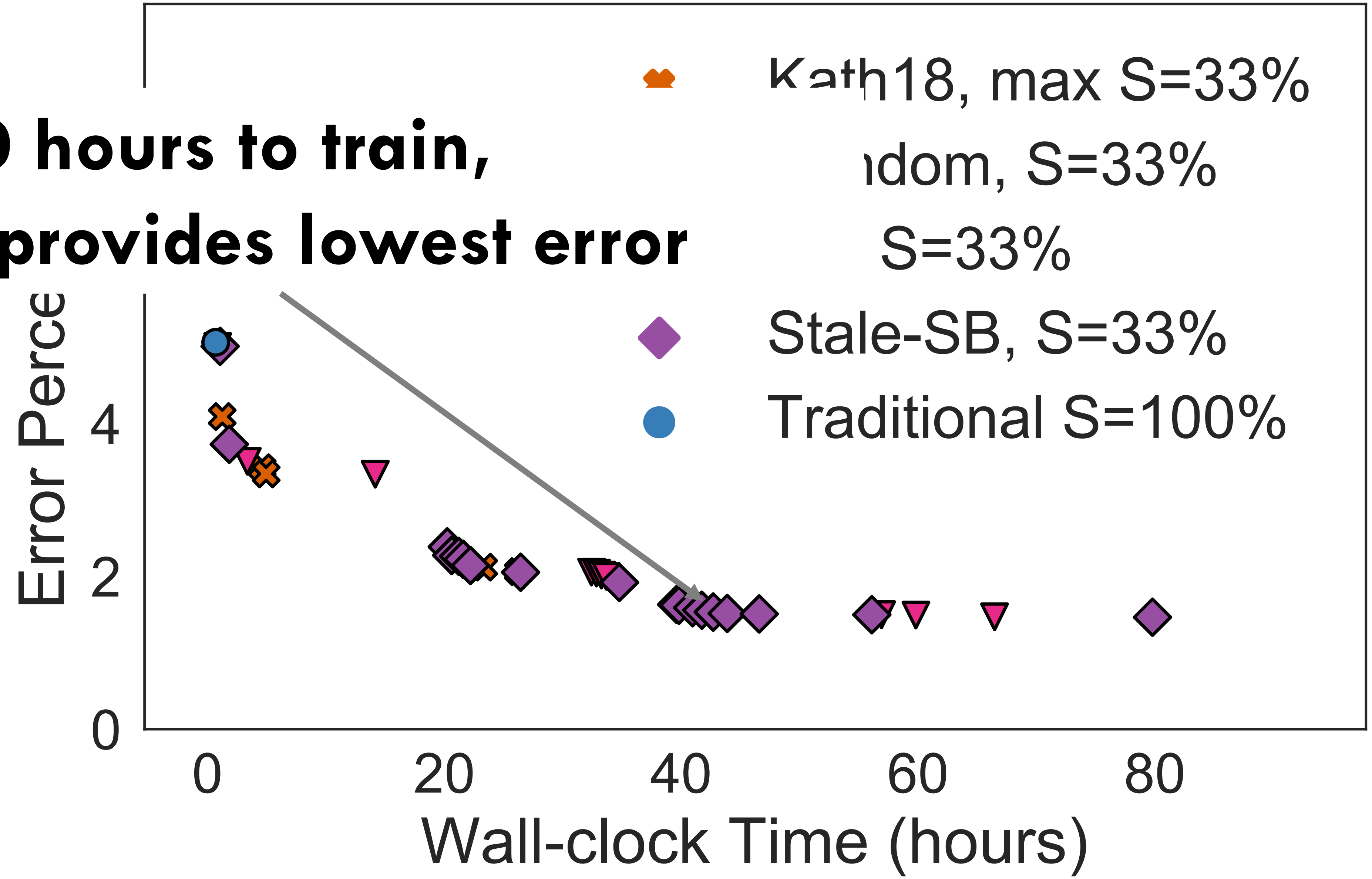
Random

Random sampling approach

Selective-Backprop (Us)

SVHN

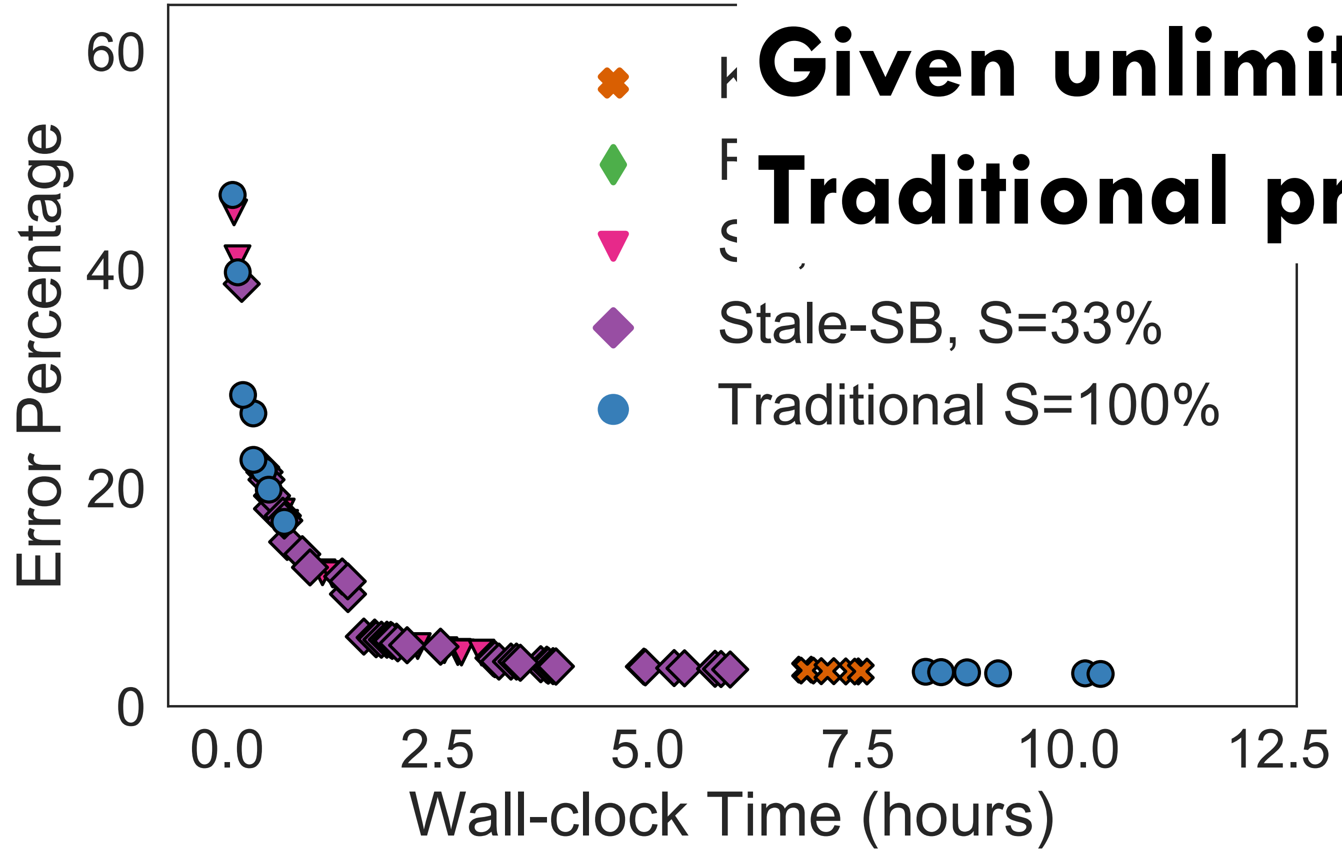
Given 40 hours to train,
Stale-SB provides lowest error



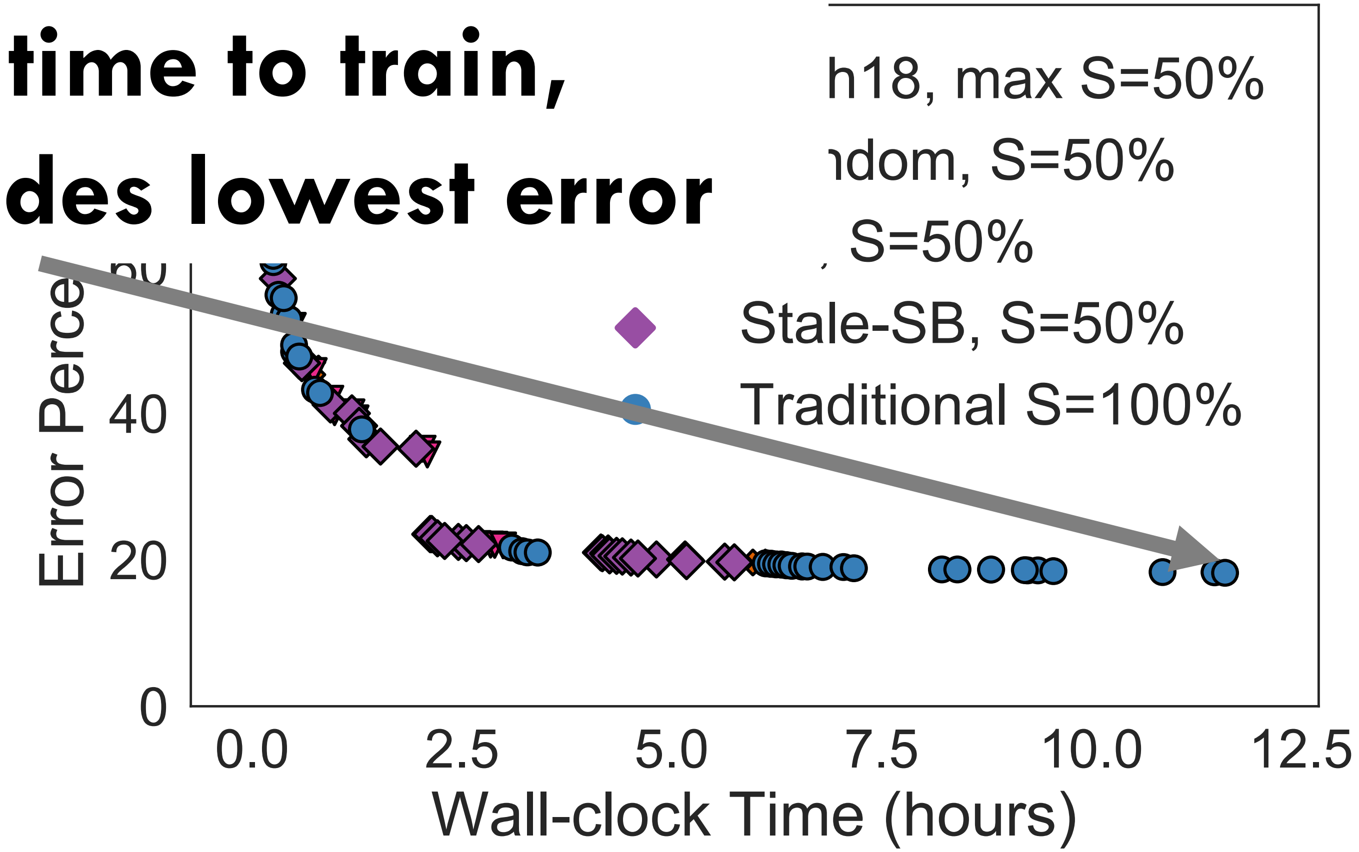
Most Pareto optimal points are SB or StaleSB

Given unlimited time to train, Traditional provides lowest error

CIFAR10

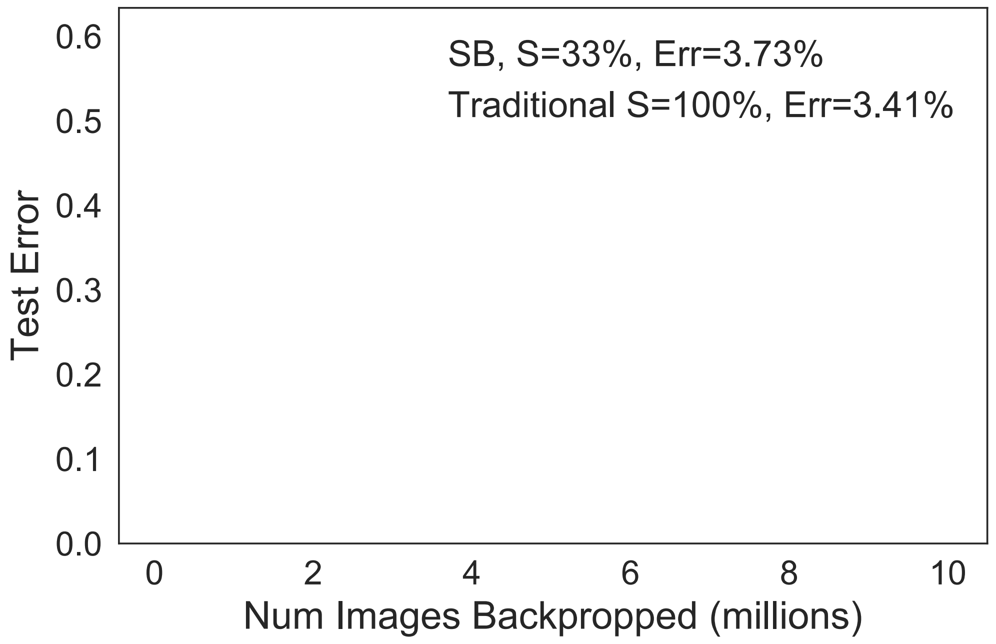


CIFAR100

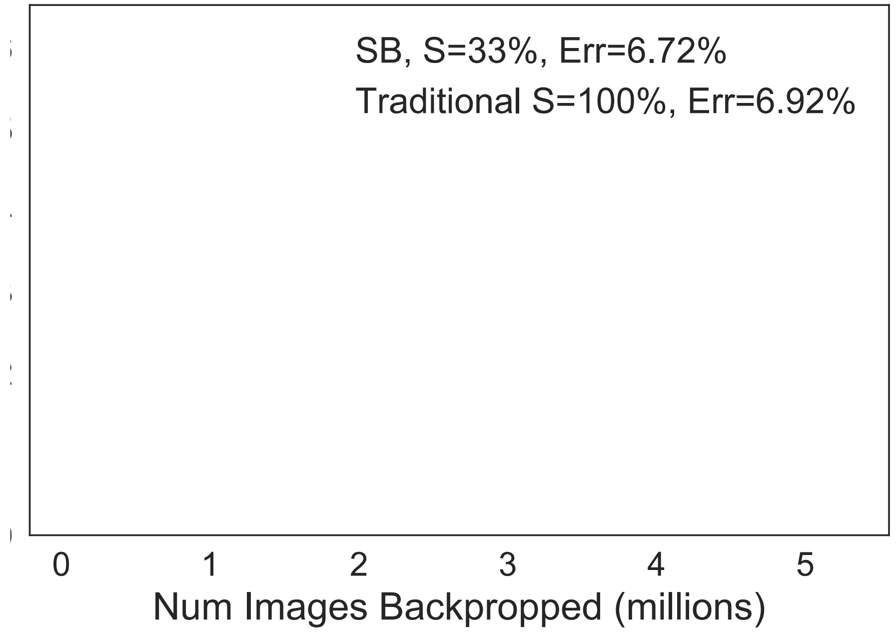


SB is robust to modest amounts of error

0.1% Randomized



10% Randomized



20% Randomized