

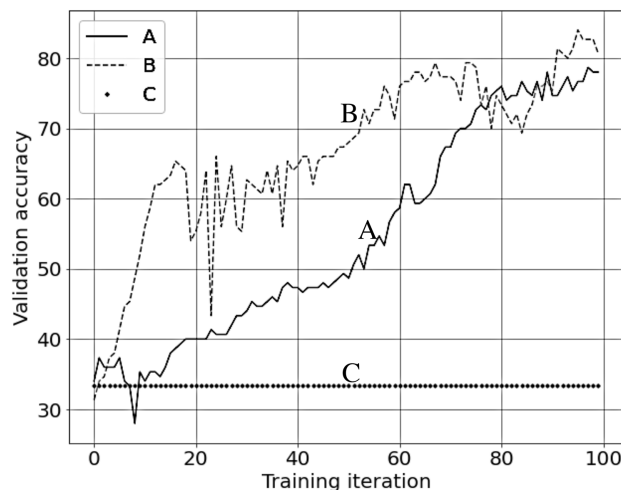
EECS 182 Deep Neural Networks

Fall 2026 Gireeja Ranade

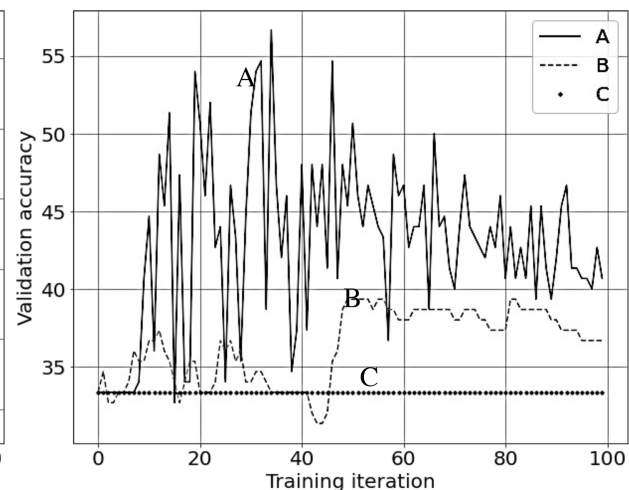
Discussion 4

1. Debugging neural networks

- (a) Recall that we learned three initialization methods in lecture: Zero (all weight values are set to 0 in the beginning), Xavier, and He. Suppose we train two different deep CNNs to classify images: (i) using ReLU as the activation function, and (ii) using sigmoids as the activation function. For each of them, we try initializing weights with the three different initialization methods aforementioned, while the biases are always initialized to all zeros. We plot the validation accuracies with different training iterations below:



(i) CNN with ReLU as the activation



(ii) CNN with sigmoid as the activation

Which initialization methods are A, B, and C, and very briefly explain your reasoning below it.

A is ☐ Zero ☐ Xavier ☐ He

B is ☐ Zero ☐ Xavier ☐ He

C is ☐ Zero ☐ Xavier ☐ He

Solution: A is Xavier, B is He initialization. From the plot we observe that B performs better for ReLU while A performs better for sigmoid. Recall that Xavier is designed for sigmoid activations, whereas He initialization considers ReLU. He initialization assumes the inputs to the ReLU function are half non-positive and half positive and change the standard deviation of the normalization distribution to compensate for that property.

C is zero parameter initialization. This method causes neurons to perform the same calculation in each iteration and produces the same outputs. Thus the accuracy is the lowest and not updating for different iterations.

- (b) You are designing a neural network to perform image classification. You want to use data augmentation to regularize the training for your model. You write the following code:

```
import random
from torch.utils.data import DataLoader, Subset

train_dataset = load_train_dataset()

augmented_dataset = apply_augmentations(train_dataset)

num_data = len(augmented_dataset)
indices = list(range(num_data))
random.shuffle(indices)
split = int(0.8 * num_data)
train_idxes, val_idxes = indices[:split], indices[split:]

train_data = Subset(augmented_dataset, train_idxes)
val_data = Subset(augmented_dataset, val_idxes)
test_data = load_test_dataset()

train_loader = DataLoader(train_data, batch_size=32)
val_loader = DataLoader(val_data, batch_size=32)
test_loader = DataLoader(test_data, batch_size=32)

# Train the model
for epoch in range(10):
    for images, labels in train_loader:
        # Training code here
    # Validation code here

# Testing code here
```

On running this code, you find that you get a *high training accuracy and validation accuracy, but a significantly lower testing accuracy as compared to the validation accuracy*.

Your TA suggests that there is a bug in your code that is causing this behavior. **Identify the bug, briefly explain why it causes the observed behavior, and show how to fix the bug.**

Solution: In our buggy code, we perform data augmentation before we create train/validation splits. This causes training data to leak into the validation set and vice-versa. To correct this, we must perform data augmentation after creating the training/validation splits.

```
train_dataset = load_train_dataset()

train_idxes, val_idxes = get_train_val_idxes(train_dataset)

train_data = apply_augmentations(Subset(train_dataset, train_idxes))
val_data = Subset(train_dataset, val_idxes)
test_data = load_test_dataset()

train_loader = DataLoader(train_data, batch_size=32)
```

```
val_loader = DataLoader(val_data, batch_size=32)
test_loader = DataLoader(test_data, batch_size=32)
```

2. Bias-Variance Tradeoff

Say that we are trying to model a function whose true value is f . We train a predictor $f_{\mathcal{D}}$ on a dataset $\mathcal{D}$, and we want to find the expected squared-error of our function, given a distribution over datasets

$$\mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}} - f\|^2]$$

Recall that the bias and variance of a predictor can be written as

$$\begin{aligned}\text{bias} &= \|\bar{f}(x) - f(x)\| \\ \text{variance} &= \mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - \bar{f}\|^2]\end{aligned}$$

where $\bar{f}(x) = \mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(x)]$ represents the average value of a trained predictor.

Decompose the expected squared-error of our learned function in terms of the bias and variance.

Hint: Add and subtract the average predictor $\bar{f}(x)$ inside the norm.

Solution: The trick is to add and subtract the average trained predictor $\bar{f}(x)$ inside the norm.

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - f(x)\|^2] &= \mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - \bar{f}(x) + \bar{f}(x) - f(x)\|^2] \\ &= \mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - \bar{f}(x) + \bar{f}(x) - f(x)\|^2] \\ &= \mathbb{E}_{\mathcal{D}}[\|(f_{\mathcal{D}}(x) - \bar{f}(x)) + (\bar{f}(x) - f(x))\|^2] \\ &= \mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - \bar{f}(x)\|^2] + \mathbb{E}_{\mathcal{D}}[\|\bar{f}(x) - f(x)\|^2] + 2\mathbb{E}_{\mathcal{D}}[(f_{\mathcal{D}}(x) - \bar{f}(x))^T(\bar{f}(x) - f(x))]\end{aligned}$$

Cross-term is 0 since $\bar{f}(x) = \mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(x)]$. Also, notice that the second term doesn't depend on dataset $\mathcal{D}$. So we finally have

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - f(x)\|^2] &= \mathbb{E}_{\mathcal{D}}[\|f_{\mathcal{D}}(x) - \bar{f}(x)\|^2] + \|\bar{f}(x) - f(x)\|^2 \\ &= (\text{variance}) + (\text{bias})^2\end{aligned}$$

Contributors:

- Romil Bhardwaj.
- Peter Wang.
- Long He.
- Naman Jain.
- Matteo Guarrera.
- Mert Cemri.
- Sizhe Chen.
- Saagar Sanghavi.
- Sergey Levine.