EECS 182 Deep Neural Networks
Fall 2026 Gireeja Ranade

Homework 6

This homework is due on Oct 14, at 10:59PM.

1. Inductive Bias of CNNs (Coding Question)

In this problem, you will follow the edge_detection.ipynb notebook to understand the inductive bias of CNNs.

1 Overfitting Models to Small Dataset: **Fill out notebook section (Q1).**
   (i) Can you find any interesting patterns in the learned filters?

2 Sweeping the Number of Training Images: **Fill out notebook section (Q2).**
   (i) Compare the learned kernels, untrained kernels, and edge-detector kernels. What do you observe?
   (ii) In this experiment, we initialize the two convolutional layers with edge detectors and freeze both convolutional layers, so only the final layer (classifier) is trained. In the high data regime, this CNN performs worse than the CNN from the previous experiment, which was initialized with random weights and trained end to end. Why do you think this happens?

3 Checking the Training Procedures: **Fill out notebook section (Q3).**
   (i) List every epochs that you trained the model. Final accuracy of CNN should be at least 95% for 30 images per class.
   (ii) Check the learned kernels. What do you observe?
   (iii) (optional) You might find that with the high number of epochs, validation loss of MLP is increasing while validation accuracy increasing. How can we interpret this?
   (iv) (optional) Do hyperparameter tuning. And list the best hyperparameter setting that you found and report the final accuracy of CNN and MLP.
   (v) How much more data is needed for MLP to get a competitive performance with CNN?

4 Domain Shift between Training and Validation Set: **Fill out notebook section (Q4).**
   (i) Why do you think the confusion matrix looks like this?
   (ii) Why do you think MLP fails to learn the task while CNN can learn the task? (Hint: Think about the model architecture.)

5 When CNN is Worse than MLP: **Fill out notebook section (Q5).**
   (i) What do you observe? What is the reason that CNN is worse than MLP? (Hint: Think about the model architecture.)
   (ii) Assuming we are decreasing kernel size of CNN. Does the validation accuracy increase or decrease? Why?
   (iii) What do the learned kernels look like? Explain why.

6 Increasing the Number of Classes: **Fill out notebook section (Q6).**
   (i) Compare the performance of CNN with max pooling and average pooling. What are the advantages of each pooling method?

 All Rights Reserved. This may not be publicly shared without explicit permission.

2. Weights and Gradients in a CNN

In this homework assignment, we aim to accomplish two objectives. Firstly, we seek to comprehend that the weight updates of a convolutional layer are a weighted sum of the input patches in the dataset. This understanding is crucial in answering a commonly asked question: does a CNN memorize images during the training process? Additionally, we will analyze the impact of spatial weight sharing in convolution layers. Secondly, we aim to gain an understanding of the behavior of max-pooling and avg-pooling in backpropagation. By accomplishing these objectives, we will enhance our knowledge of CNNs and their functioning.

Let's consider a convolution layer with input matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$,

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix}, \tag{1}$$

weight matrix $\mathbf{w} \in \mathbb{R}^{k \times k}$,

$$\mathbf{w} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,k} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,1} & w_{k,2} & \cdots & w_{k,k} \end{bmatrix}, \tag{2}$$

and output matrix $\mathbf{Y} \in \mathbb{R}^{m \times m}$,

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,m} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m,1} & y_{m,2} & \cdots & y_{m,m} \end{bmatrix}. \tag{3}$$

For simplicity, we assume the number of the input channel (of $\mathbf{X}$) and the number of the output channel (of output $\mathbf{Y}$) are both 1, and the convolutional layer has no padding and a stride of 1.

Then for all i, j,

$$y_{i,j} = \sum_{h=1}^{k} \sum_{l=1}^{k} x_{i+h-1,j+l-1} w_{h,l}, \tag{4}$$

or

$$\mathbf{Y} = \mathbf{X} * \mathbf{w}, \tag{5}$$

 All Rights Reserved. This may not be publicly shared without explicit permission.

where $*$ refers to the convolution operation. For simplicity, we omitted the bias term in this question. Suppose the final loss is $\mathcal{L}$, and the upstream gradient is $d\mathbf{Y} \in \mathbb{R}^{m\times m}$,

$$d\mathbf{Y} = \begin{bmatrix} dy_{1,1} & dy_{1,2} & \cdots & dy_{1,m} \\ dy_{2,1} & dy_{2,2} & \cdots & dy_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ dy_{m,1} & dy_{m,2} & \cdots & dy_{m,m} \end{bmatrix}, \tag{6}$$

where $dy_{i,j}$ denotes $\dfrac{\partial \mathcal{L}}{\partial y_{i,j}}$.

(a) **Derive the gradient to the weight matrix $d\mathbf{w} \in \mathbb{R}^{k\times k}$,**

$$d\mathbf{w} = \begin{bmatrix} dw_{1,1} & dw_{1,2} & \cdots & dw_{1,k} \\ dw_{2,1} & dw_{2,2} & \cdots & dw_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ dw_{k,1} & dw_{k,2} & \cdots & dw_{k,k} \end{bmatrix}, \tag{7}$$

where $dw_{h,l}$ denotes $\dfrac{\partial \mathcal{L}}{\partial w_{h,l}}$. Also, **derive the weight after one SGD step with a batch of a single image**.

(b) The objective of this part is to investigate the effect of spatial weight sharing in convolution layers on the behavior of gradient norms with respect to changes in image size.
For simplicity of analysis, we assume $x_{i,j}, dy_{i,j}$ are independent random variables, where for all i, j:

$$\mathbb{E}[x_{i,j}] = 0, \tag{8}$$
$$\mathrm{Var}(x_{i,j}) = \sigma_x^2, \tag{9}$$
$$\mathbb{E}[dy_{i,j}] = 0, \tag{10}$$
$$\mathrm{Var}(dy_{i,j}) = \sigma_g^2. \tag{11}$$

Derive the mean and variance of $dw_{i,j} = \dfrac{\partial \mathcal{L}}{\partial w_{i,j}}$ for each i, j as a function of n, k, σ_x, σ_g. What is the asymptotic growth rate of the <u>standard deviation</u> of the gradient on $dw_{i,j}$ with respect to the length and width of the image n, with k, $\sigma_x > 0$, and $\sigma_g > 0$ fixed?
Hint: there should be no m in your solution because m can be derived from n and k.
Hint: you cannot assume that $x_{i,j}$ and $dy_{i,j}$ follow normal distributions in your derivation or proof.

(c) Assume stride 2, no padding, and an even input side length. **For a network with only one 2x2 max-pooling layer (no convolution layers, no activations), what will be $d\mathbf{X} = [dx_{i,j}] = [\dfrac{\partial \mathcal{L}}{\partial x_{i,j}}]$? For a network with only one 2x2 average-pooling layer (no convolution layers, no activations), what will be $d\mathbf{X}$?**
HINT: Start with the simplest case first, where $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$. Further assume that top left value is

 All Rights Reserved. This may not be publicly shared without explicit permission.

selected by the max operation. i.e.

$$y_{1,1} = x_{1,1} = \max(x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) \tag{12}$$

Then generalize to higher dimension and arbitrary max positions.

(d) Following the previous part, **discuss the advantages of max pooling and average pooling** in your own words.

Hint: you may find it helpful to finish the question "Inductive Bias of CNNs (Coding Question)" before working on this question

3. Convolutional Networks

Note: Throughout this problem, we will use the convention of NOT flipping the filter before dragging it over the signal. This is the standard notation with neural networks (technically, this operation is cross-correlation)

(a) **List two reasons we typically prefer convolutional layers instead of fully connected layers when working with image data.**

(b) Consider the following 1D signal: $[1, 4, 0, -2, 3]$. After convolution with a length-3 filter, no padding, stride=1, we get the following sequence: $[-2, 2, 11]$. **What was the filter?**

(Hint: Just to help you check your work, the first entry in the filter that you should find is 2. However, if you try to use this hint directly to solve for the answer, you will not get credit since this hint only exists to help you check your work.)

(c) Transpose convolution is an operation to help us upsample a signal (increase the resolution). For example, if our original signal were $[a, b, c]$ and we perform transpose convolution with pad=0 and stride=2, with the filter $[x, y, z]$, the output would be $[ax, ay, az + bx, by, bz + cx, cy, cz]$. Notice that the entries of the input are multiplied by each of the entries of the filter. Overlaps are summed. Also notice how for a fixed filter size and stride, the dimensions of the input and output are swapped compared to standard convolution. (For example, if we did standard convolution on a length-7 sequence with filter size of 3 and stride=2, we would output a length-3 sequence).

If our 2D input is $\begin{bmatrix} -1 & 2 \\ 3 & 1 \end{bmatrix}$ and the 2D filter is $\begin{bmatrix} +1 & -1 \\ 0 & +1 \end{bmatrix}$ **What is the output of transpose convolution with pad=0 and stride=1?**

4. Memory considerations when using GPUs (Coding Question)

In this homework, you will run GPUMemory.ipynb to train a ResNet model on CIFAR-10 using PyTorch and explore its implications on GPU memory.

We will explore various systems considerations, such as the effect of batch size on memory usage and how different optimizers (SGD, SGD with momentum, Adam) vary in their memory requirements.

It is strongly recommended that you start early on this question, since colab daily GPU limits may require you to complete this question over a few days with breaks in between.

(a) Managing GPU memory for training neural networks (Notebook Section 1).

(i) **How many trainable parameters does ResNet-152 have**? What is the **estimated size of the model in MB**?

(ii) Which GPU are you using? **How much total memory does it have**?

 All Rights Reserved. This may not be publicly shared without explicit permission.

(iii) After you load the model into memory, **what is the memory overhead (MB) beyond the model parameters**?

(b) Optimizer memory usage (Notebook Section 2).

(i) **What is the total memory utilization during training with SGD, SGD with momentum and Adam optimizers**? Report in MB individually for each optimizer.

(ii) **Which optimizer consumes the most memory**? Why?

(c) Batch size, learning rates and memory utilization (Notebook Section 3)

(i) **What is the memory utilization for different batch sizes (4, 16, 64, 256)? What is the largest batch size you were able to train?**

(ii) **Which batch size gave you the highest accuracy at the end of 10 epochs?**

(iii) **Which batch size completed 10 epochs the fastest (least wall clock time)? Why?**

(iv) Attach your **training accuracy vs wall time plots** with your written submission.

5. Denoising using U-Nets

U-Nets are one of the most influential architectures in deep learning for tasks requiring precise spatial localization—such as image segmentation and denoising. The U-Net is composed of two main components:

Encoder: aims to decrease the spatial dimensions of the image into a feature map that captures relevant information about the image.

Decoder: aims to upsample the feature map and produce a relevant segmentation map using the patterns learnt in the contracting path.

Importantly, U-Nets add **skip connections** such that every block in the decoder incorporates the feature map from its corresponding block in the encoder. Its architecture is visualized in the below figure:

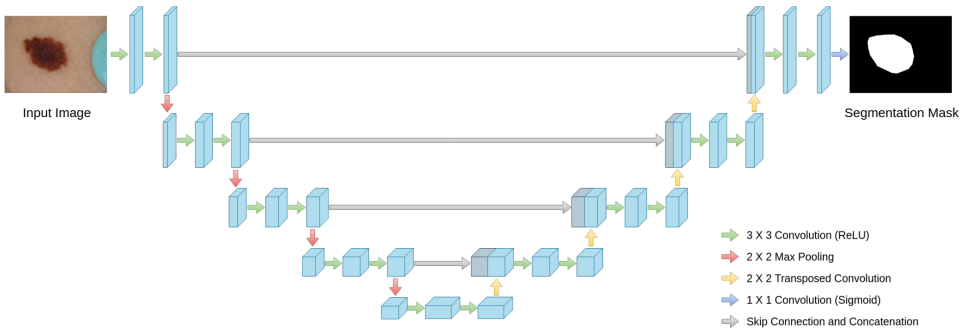


Figure 1: Diagram for U-Net

To understand why U-Nets are so effective, it helps to study their components in a simplified setting of denoising a 1D signal.

Let a clean signal be $\mathbf{x} = [0, 0, 0, 9, 9, 9, 9, 9]^T$. As input, we observe $\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$, where each ϵ_i is independent Gaussian noise with variance $\mathrm{Var}(\epsilon_i) = \sigma^2$. Our goal is to predict $\mathbf{y}$ such that it is as close to the true signal $\mathbf{x}$ as possible.

(a) Consider a simple U-net architecture with 2 levels where each encoder block performs a convolution with kernel $\mathbf{k} = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]^T$ using stride 1 and same padding. In between the two blocks is max-pooling with filter size 2 and stride 2. Let $\mathbf{e}$ be the output of the encoder applied to $\hat{\mathbf{x}}$. **What is the**

 All Rights Reserved. This may not be publicly shared without explicit permission.

expected output $\mathbb{E}[\mathbf{e}]$? What is the variance of each element e_i (in terms of σ)? You can assume that elements of the input to each convolution are independently and identically distributed.

(b) In the same U-net, the decoder performs a transpose convolution with kernel $\mathbf{k} = [1, 1]^T$ with stride 2. Then, the reconstructed signal is the element-wise average of the upsampled signal and the output of the first encoder block (via a skip connection). **What is the expected reconstructed signal $\mathbb{E}[\mathbf{y}]$? What is the variance of each element y_i Why is the skip connection important?** You can assume that the upsampled signal and the output of the encoder block are independent.

(Hint: The smoothing operation in the encoder, though useful for dampening noise, destroys one crucial detail about the signal. The skip connection is important to bring this detail.)

Contributors:

- Matteo Guarrera.
- Mert Cemri.
- Sizhe Chen.
- Suhong Moon.
- Kumar Krishna Agrawal.
- Peter Wang.
- Anant Sahai.
- Qiyang Li.
- Linyuan Gong.
- Long He.
- Saagar Sanghavi.
- Romil Bhardwaj.
- Joey Hong.
- Gireeja Ranade.

 All Rights Reserved. This may not be publicly shared without explicit permission.