Problem Set 3

Please submit a typed PDF addressing all problems below. This problem set contains 3 questions and is worth 25 points. **All responses MUST be in *your* own words.** Justification must be provided for **all** written answers. Statements made without any supporting explanation/justification will receive **no credit**. For mathematical derivations and plots, you may insert pictures of handwritten work if you find this easier. The required weekly readings and lecture slides should be helpful in completing the assignment. You can find these on our course website.

1. **Computer Vision Problems [6 points]:**

   (a) Define image classification, object detection, instance segmentation, and semantic segmentation. Then, describe the similarities and differences between these tasks. Your response should include discussion of the inputs and outputs of neural networks used for these tasks.

Term	**Definition**	**Input**	**Output**
Image Classification	The task of assigning a label or class to an entire image. The goal is to identify the main subject of the image from a predefined set of categories.	A 3D matrix representing the image's pixels (height, width, and color channels).	A vector of probabilities, where each element corresponds to the likelihood of the image belonging to a specific class. The class with the highest probability is chosen as the prediction.
Object Detection	Builds upon image classification by not only identifying the objects in an image but also locating them with bounding boxes.	A 3D matrix representing the image's pixels (height, width, and color channels).	A list of bounding boxes, with each box containing the coordinates of the detected object and a class label with an associated confidence score.
Semantic Segmentation	A pixel-level classification task where a class label is assigned to every pixel in an image. This means that all objects of the same class are marked with the same label.	A 3D matrix representing the image's pixels (height, width, and color channels).	A segmentation map, which is an image of the same dimensions as the input, where each pixel's value corresponds to its assigned class.
Instance Segmentation	Goes a step further than semantic segmentation by not only classifying each pixel but also distinguishing between different instances of the same object class.	A 3D matrix representing the image's pixels (height, width, and color channels).	A unique mask for each individual object instance detected in the image, along with a class label and a bounding box for each instance.

Similarities and Differences:

Task	**Goal**	**Output**	**Granularity**
Image Classification	What is in the image?	Single class label for the entire image.	Image-level
Object Detection	What objects are in the image and where are they?	Bounding boxes with class labels	Object-level
Semantic Segmentation	What is the class of each pixel?	Pixel-level classification map	Pixel-level
Instance Segmentation	What is the class and identity of each pixel?	Pixel-level masks for each individual object	Instance-level

A key similarity is that all these tasks can be tackled using deep neural networks, particularly Convolutional Neural Networks (CNNs), which are adept at learning features from image data. The primary difference lies in the level of detail and the nature of the output they provide, moving from a single label for the entire image to a detailed, instance-aware, pixel-level understanding of the scene.

(b) Provide one example real world use-case for each of the four tasks discussed in part (a).

- **Image Classification**
  - **Medical Imaging:** Classifying medical images, such as X-rays or MRI scans, to detect the presence of diseases like cancer of pneumonia.
  - **Automated Quality Control:** In manufacturing, image classification can be used to automatically inspect products on an assembly line and identify defective items.
- **Object Detection**
  - **Autonomous Vehicles:** Self-driving cars use object detection to identify and locate pedestrians, other vehicles, traffic signs, and obstacles to navigate safely.
  - **Retail Inventory Management:** Object detection can be used to monitor shelves in a retail store to track inventory levels and identify out-of-stock items.
- **Semantic Segmentation**
  - **Autonomous Driving:** Identifying drivable areas, lanes, and sidewalks to help a self-driving car understand its environment.

    - **Medical Image Analysis:** Precisely segmenting organs, tissues, or tumors in medical scans to aid in diagnosis and treatment planning.
- **Instance Segmentation**
    - **Self-Driving Cars:** Distinguishing between individual pedestrians or vehicles in a crowded scene for more accurate tracking and prediction of their movements.
    - **Robotics** Enabling robots to identify and grasp specific objects from a cluttered environment.

2. **Model Size [9 points]:** You are designing a neural network to classify inputs of shape $3 \times 224 \times 224$ into 100 classes, and need to investigate the differences between a fully connected and convolutional approach.

    (a) Compute the number of model parameters needed for a fully connected architecture. This model first "flattens" the input, then feeds forward through a single hidden layer with half as many neurons. In other words, the hidden layer down-samples the flattened input by a factor of 2. Then, a second fully connected layer transforms the intermediate representation to the 100 dimensional output required for the task.

    i. Report the number of parameters per layer.
    - **Input Layer:** $3 \times 224 \times 224$
    - **Flattened Input Size:** $3 \times 224 \times 224 = 150,528$
    - **Hidden Layer Size:** $\frac{150,528}{2} = 75,264$
    - **Layer 1 (Input to Hidden)**
        - Weights: $150,528 \times 75,264 = 11,329,339,392$
        - Biases: $75,264$
        - **Total Layer 1 Parameters:** $11,329,339,392 + 75,264 = 11,329,414,656$
    - **Layer 2 (Hidden to Output)**
        - Weights: $75,264 \cdot 100 = 7,526,400$
        - Biases: 100
        - **Total Layer 2 Parameters:** $7,526,400 + 100 = 7,526,500$

    ii. Compute the number of total model parameters (weights and biases).
    - **Total Parameters:** $11,329,414,656$ (Layer 1)$+7,526,500$ (Layer 2) $= 11,336,941,156$

(b) Compute the number of model parameters needed for a convolutional neural network architecture. This model uses 2-D convolution layers with 7×7 filters, no padding, and 1×1 stride. The layers do not change the number of channels, i.e., the output from the convolutions should also have 3 channels, same as the input. This type of convolutional layer is repeatedly applied until the features are of shape $3 \times 8 \times 8$. Then, a flatten layer is used and the result is fed into a fully connected layer with an output size of 100.

i. Find the number of parameters for a single 7×7 convolutional layer which takes an input with 3 channels and returns an output with 3 channels.

- **Filter Size:** 7×7
- **Input Channels:** 3
- **Output Channels:** 3
- **Weights:** $(7 \times 7 \times 3) \times 3 = 441$
- **Biases:** 3
- **Total Parameters per Convolutional Layer:** $441+3 = 444$

ii. Using convolutional layers of the type described in (i), compute how many of these layers are required to reduce the input to the shape $3 \times 8 \times 8$.

- **Initial Dimension:** 224
- **Filter Size:** 7
- **Stride:** 1
- **Output Size Formula:** $\frac{\text{Input Size}-\text{Filter Size}}{\text{Stride}} + 1$
- **Layer 1 Output:** $\frac{224-7}{1} + 1 = 218$
- **Layer 2 Output:** $\frac{218-7}{1} + 1 = 212$
- $\ldots$
- Let n be the number of layers. The reduction in size per layer is $(\text{Filter Size} - 1) = 6$
- **Final Size:** $224 - 6n = 8$
- $6n = 216$
- **n = 36 layers**

iii. Compute the number of parameters required for a fully connected layer which takes an input of shape $3 \times 8 \times 8$, flattens it, and then produces an output of size 100.

- **Input Shape:** $3 \times 8 \times 8$
- **Flattened Input Size:** $3 \times 8 \times 8 = 192$

- **Output Size:** 100
- **Weights:** $192 \times 100 = 19,200$
- **Biases:** 100
- **Total Parameters:** $19,200 + 100 = 19,300$

iv. Compute the total number of parameters for the entire architecture.

- **Convolutional Layers:** 36 layers $\times\ 444 \frac{\text{parameters}}{\text{layer}} = 15,984$
- **Fully Connected Layer:** 19,300
- **Total Parameters:** $15,984 + 19,300 = 35,284$

3. **Convolutional Operations [10 points]:** This is **NOT** a programming exercise. Parts (a) and (b) both refer to the following data:

Data for Question 3

Input

2	0	-1	-1
0	1	0	2
-2	0	1	1
1	1	0	-1

Filter

1	-1	1
0	1	0
1	-1	1

(a) Compute and report the output after performing a convolution of the input with the filter. Use a 1×1 stride and no padding. For full-credit, you **MUST** show your mathematical derivations.

The output size of a convolution with a 4×4 input and a 3×3 filter with 1×1 stride and no padding is $\frac{4-3}{1} + 1 = 2 \times 2$. Let O be the resulting 2×2 matrix:

$$O_{1,1} = (1)(2) + (-1)(0) + (1)(-1) + (0)(0) + (1)(1) + (0)(0) + (1)(-2) + (-1)(0) + (1)(1) = 1$$
$$O_{1,2} = (1)(0) + (-1)(-1) + (1)(-1) + (0)(1) + (1)(0) + (0)(2) + (1)(0) + (-1)(1) + (1)(1) = 0$$
$$O_{2,1} = (1)(0) + (-1)(1) + (1)(0) + (0)(-2) + (1)(0) + (0)(1) + (1)(1) + (-1)(1) + (1)(0) = -1$$
$$O_{2,2} = (1)(1) + (-1)(0) + (1)(2) + (0)(0) + (1)(1) + (0)(1) + (1)(1) + (-1)(0) + (1)(-1) = 4$$

Thus,

$$O = \begin{bmatrix} 1 & 0 \\ -1 & 4 \end{bmatrix}$$

(b) Compute and report the output after performing a *transposed* convolution of the input with the filter. Use a 1×1 stride and no padding. For full-credit, you **MUST** show your mathematical derivations.

For a transposed convolution with a 4×4 input and 3×3 filter with 1×1 stride and no padding, the output size is $(4-1)*1+3 = 6 \times 6$. A direct way to compute it is to multiple each element of the input matrix by the filter and sum the overlapping regions. Let O be the resulting 6×6 matrix and $IF_{i,j}$ represent the segment of O where the input element $I_{i,j}$ of the input matrix is transpose convoluted by the filter:

$$
\begin{aligned}
O =& OF_{1,1} + OF_{1,2} + OF_{1,3} + OF_{1,4} + \\
& OF_{2,1} + OF_{2,2} + OF_{2,3} + OF_{2,4} + \\
& OF_{3,1} + OF_{3,2} + OF_{3,3} + OF_{3,4} + \\
& OF_{4,1} + OF_{4,2} + OF_{4,3} + OF_{4,4}
\end{aligned}
$$

$$
= \begin{bmatrix} 2 & -2 & 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & -2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} +
$$

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 & -2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} +
$$

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 2 & -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 \\ -2 & 2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} +
$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -2 & 1 & 0 & 0 & -1 \\ 0 & 3 & -1 & 2 & -3 & 2 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 1 & -1 & -1 & 4 & 0 & 1 \\ -2 & 3 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

4. **Extra Credit [2.5 points]:** You are designing a fully convolutional architecture for an image segmentation task. The model consists of two stages: a series of 2-D convolution layers for down-sampling followed by a series of 2- D transposed convolution layers for up-sampling. The inputs are of shape $3 \times 224 \times 224$, the desired smallest *latent* representations are of shape $256 \times 16 \times 16$, and the final outputs are of shape $3 \times 224 \times 224$. The down-sampling side uses filters of size 3×3, and the number of channels is given by:

$$[3, 16, 32, 64, 128, 256, 256, 256...] \tag{1}$$

where the channels are kept at 256 until the desired latent representation shape is achieved by continual application of 2-D convolution layers. After the features reach shape $256 \times 16 \times 16$, transposed convolution layers with 9×9 filters are used to up-sample the features back to their original spatial resolution. The number of channels for this stage is given by:

$$[256, 256, ..., 256, 128, 64, 32, 16, 3] \tag{2}$$

In other words, the channels are kept at 256 until the final 5 layers, which follow the same pattern as the down-sampling stage in reserve. For all layers, a 1×1 stride is used with no padding. Compute the number of parameters required for: (1) the down-sampling stage, (2) the up-sampling stage, and (3) the entire model.

Down-Sampling Stage:
The number of parameters for a convolutional layer is (Filter Height$\times$Filter Width$\times$ Input Channels + 1) $\times$ Output Channels.

The channel progression for down-sampling is $[3, 16, 32, 64, 128, 256, 256, \ldots]$.
The filter size is 3×3.

1. $3 \to 16$: $(3 \times 3 \times 3 + 1) \times 16 = 448$
2. $16 \to 32$: $(3 \times 3 \times 16 + 1) \times 32 = 4640$
3. $32 \to 64$: $(3 \times 3 \times 32 + 1) \times 64 = 18496$
4. $64 \to 128$: $(3 \times 3 \times 64 + 1) \times 128 = 73856$
5. $128 \to 256$: $(3 \times 3 \times 128 + 1) \times 256 = 295168$

Total reduction in spatial dimension per 3×3 convolution with no padding is $3 - 1 = 2$.
Initial size: 224×224. Target size: 16×16.
Total reduction needed: $224 - 16 = 208$.
Number of layers $= \frac{208}{2} = 104$

We have 5 layers with changing channels. The remaining $104 - 5 = 99$ layers will have $256 \to 256$ channels.
Parameters for a $256 \to 256$ layer: $(3 \times 3 \times 256 + 1) \times 256 = 590080$.
Total parameters for these 99 layers: $99 \times 590080 = 58417920$.

Total parameters for the down-sampling stage: $448 + 4640 + 18496 + 73856 + 295168 + 58417920 = 58810528$

Up-Sampling Stage:
The number of parameters for a transposed convolutional layer is the same as a standard convolutional layer with the input and output channels swapped.
The filter size is 9×9.
The channel progression is $[\ldots, 256, 256, 128, 64, 32, 16, 3]$.

The up-sampling per 9×9 transposed convolution is $9 - 1 = 8$.
Initial size: 16×16. Target size: 224×224.
Total increase needed: $224 - 16 = 208$.
Number of layers $= \frac{208}{8} = 26$.

There are 5 layers with changing channels at the end. The first $26 - 5 = 21$ layers will have $256 \to 256$ channels.
Parameters for a $256 \to 256$ layer: $(9 \times 9 \times 256 + 1) \times 256 = 5308672$.
Total parameters for these 21 layers: $21 \times 5308672 = 111482112$.

Parameters for the last 5 layers:

1. $256 \to 128$: $(9 \times 9 \times 256 + 1) \times 128 = 2654336$
2. $128 \to 64$: $(9 \times 9 \times 128 + 1) \times 64 = 663616$

3. $64 \to 32$: $(9 \times 9 \times 64 + 1) \times 32 = 165920$
4. $32 \to 16$: $(9 \times 9 \times 32 + 1) \times 16 = 41448$
5. $16 \to 3$: $(9 \times 9 \times 16 + 1) \times 3 = 3891$

Total parameters for the up-sampling stage: $111482112 + 2654336 + 663616 + 165920 + 41448 + 3891 = 115011323$

Total Model Parameters:
Total Parameters $= 58810528 + 115011323 = 173821851$

Collaboration versus Academic Misconduct: Collaboration with other students (or AI) is permitted, but the work you submit must be your own. Copying/plagiarizing work from another student (or AI) is not permitted and is considered academic misconduct. For more information about University of Colorado Boulder's Honor Code and academic misconduct, please visit the course syllabus.