CSCI 5922 problem set 3

Jake Krol

February 2025

1 Q1

1.1 Q1a

Image classification receives as **input** an image (and class label if training example) and **outputs** a probability distribution over the k possible classes the image belongs to.

Object detection receives as **input** an image (and object class label plus boundary coordinates if training example) and **outputs** 1) the coordinates of a boundary box encompassing the image plus 2) a probability distribution of the object belonging to k possible output classes.

Semantic segmentation receives as **input** an image (and pixel-wise class labels if training example) and **outputs** one-hot encoded matrices for each of the k possible classes indicating which class each pixel belongs to.

Image segmentation receives as **input** an image (and pixel-wise labels of 1) object label and 2) class label if training example) and **outputs** 1) pixel-wise object labels and 2) pixel-wise class labels to indicate both the boundaries of each individual object and the class of the object.

The key **similarites** and **differences** are the following

- All approaches receive an image as input.
- All approaches output some form of classification with differing resolution: image-wise, object-wise, pixel-wise.
- Both segmentation approaches are pixel-wise classification while other approaches are not.
- Semantic segmentation is different than image segmentation since separate objects are not delineated, only classes of objects are delineated.

1.2 Q1b

- An example of **image classification** is to answer the question of whether an image of a banking check if fraudulent, yes or no.

- An example of **object detection** is to outline the boundaries of objects in a room, say for robot navigation.
- An example **semantic segmentation** is in rotoscoping to delineate foreground objects from background objects. Where, the background can be substituted while the foreground object(s) stays the same.
- An example of **image segmentation** is in urban infrastructure planning to understand exactly where each type of building and other object is located in a city, based on images.

2 Q2

2.1 Q2a

i) The **parameters per layer** are $\boldsymbol{\theta_{h_1} = 11,329,414,656}$ and $\boldsymbol{\theta_{out} = 7,526,500}$. Where, θ_{h_1} are parameters used to compute the representation for hidden layer 1, and θ_{out} are parameters used to compute the output layer representation.

Given that,

- $d_{in} = 3 \cdot 224 \cdot 224 = 150,528$
- $d_{h_1} = \frac{d_{in}}{2} = 75,264$
- $d_{out} = 100$

The parameters per layer of a fully connected neural networks is the product of the previous layer dim and the current layer dim (weights) plus the current layer dim (biases).

$$\theta_l = (d_{l-1} \cdot d_l) + d_l \tag{1}$$

Therefore, θ_{h_1} and θ_{out} are computed by

$$\begin{aligned}\theta_{h_1} &= (d_{in} \cdot d_{h1}) + d_{h_1} = 11,329,414,656 \\ \theta_{out} &= (d_{h_1} \cdot d_{out}) + d_{out} = 7,526,500\end{aligned}$$

ii) The **total number of parameters** is $\theta_{h_1} + \theta_{out} = 11,336,941,156$

2.2 Q2b

2.2.1 Q2bi

The parameters for a single convolutional layer with a 3 channel input, 3 channel output, and $7x7$ kernel is **444**. A convolutional layer is parameterized by the number of input channels (l), the kernel dimensions (m, n), and the number of output channels (k).

$$\textbf{\# of parameters} = (l \cdot m \cdot n + 1)k =$$
$$(3 \cdot 7 \cdot 7 + 1)3 = (7^2) \cdot (3^2) + 3 = \mathbf{444}$$

The plus 1 accounts for the bias term for each output channel.

2.2.2 Q2bii

The number of convolutional layers required to go from a $3 \cdot 224 \cdot 224$ input to a $3 \cdot 8 \cdot 8$ output is **36**. Output dimensions of a convolutional layer are determined by input height (h_{in}), input width (w_{in}), kernel dimension (k), stride (s), and padding (p).

$$(h_{out}, w_{out}) = (\frac{h_{in} - k + 2p}{s} + 1, \frac{w_{in} - k + 2p}{s} + 1)$$

Since the stride is 1, padding is 0, and $h_{in} = w_{in}$. The number of layers (l) required to get $h_{out} = w_{out} = 8$ is

$$\begin{aligned} h_{in} - 6l &= 8 \\ 224 - 6l &= 8 \\ l &= \mathbf{36} \end{aligned}$$

2.2.3 Q2biii

As stated in Q2ai, the parameters used in a fully connected layer is

$$\text{\# of parameters} = (d_{in} \cdot d_{out}) + d_{out}$$

So, the number of parameters of a fully connected layer receiving a flattened $d_{in} = 3 \cdot 8 \cdot 8$ input and outputting $d_{out} = 100$ is

$$(192 \cdot 100) + 100 = \mathbf{19{,}300}$$

2.2.4 Q2biv

To compute the total number of parameters in the architecture, we combine the results from Q2bi, Q2bii, and Q2biii to get **35284**.

In more detail, we have an architecture with $l_{conv} = 36$ convolutional layers, with $\theta_{conv_i} = 444$ parameters per layer, and a fully connected component with $\theta_{fc} = 19,300$ parameters. In total,

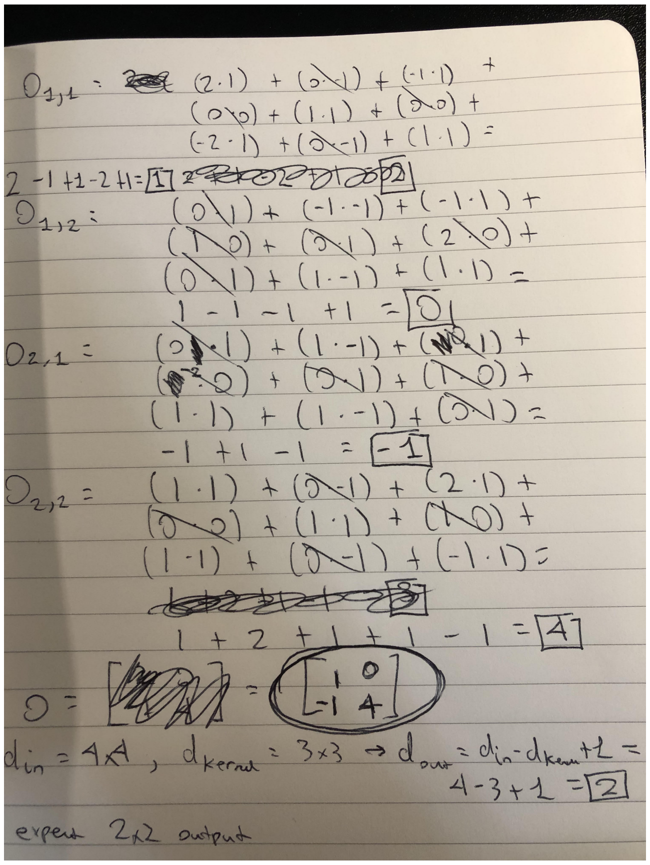
$$\theta_{total} = (\theta_{conv_i} \cdot l_{conv}) + \theta_{fc} = 444 \cdot 36 + 19,300 = \mathbf{35284}$$

3 Q3

3.1 Q3a

The output of the convolution, O, is

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



$$O_{1,1} = [illegible] (2 \cdot 1) + \cancel{(0 \cdot 1)} + (-1 \cdot 1) +$$
$$\cancel{(0 \cdot 0)} + (1 \cdot 1) + \cancel{(0 \cdot 0)} +$$
$$(-2 \cdot 1) + \cancel{(0 \cdot -1)} + (1 \cdot 1) =$$
$$2 - 1 + 1 - 2 + 1 = \boxed{1} \quad [illegible] \boxed{2}$$

$$O_{1,2} = \cancel{(0 \cdot 1)} + (-1 \cdot -1) + (-1 \cdot 1) +$$
$$\cancel{(1 \cdot 0)} + \cancel{(0 \cdot 1)} + \cancel{(2 \cdot 0)} +$$
$$\cancel{(0 \cdot 1)} + (1 \cdot -1) + (1 \cdot 1) =$$
$$1 - 1 - 1 + 1 = \boxed{0}$$

$$O_{2,1} = \cancel{(0 \cdot 1)} + (1 \cdot -1) + \cancel{(0 \cdot 1)} +$$
$$\cancel{(-2 \cdot 0)} + \cancel{(0 \cdot 1)} + \cancel{(1 \cdot 0)} +$$
$$(1 \cdot 1) + (1 \cdot -1) + \cancel{(0 \cdot 1)} =$$
$$-1 + 1 - 1 = \boxed{-1}$$

$$O_{2,2} = (1 \cdot 1) + \cancel{(0 \cdot -1)} + (2 \cdot 1) +$$
$$\cancel{(0 \cdot 0)} + (1 \cdot 1) + \cancel{(1 \cdot 0)} +$$
$$(1 \cdot 1) + \cancel{(0 \cdot -1)} + (-1 \cdot 1) =$$
$$[illegible] \boxed{3}$$
$$1 + 2 + 1 + 1 - 1 = \boxed{4}$$

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

$$d_{in} = 4 \times 4, \quad d_{kernel} = 3 \times 3 \rightarrow d_{out} = d_{in} - d_{kernel} + 1 =$$
$$4 - 3 + 1 = \boxed{2}$$

expect 2x2 output

Figure 1: Q3a convolution of input and kernel.

3.2 Q3b

Algorithmically, the transposed convolution is produced by iterating over the input matrix elements, multiplying the kernel by the I_{ij}th element, and filling the output matrix with a running sum.

Algorithm 1 Transposed convolution

```
1: Input1: Matrix $I \in \mathbb{R}^{I_h \text{ x } I_w}$
2: Input2: Matrix $K \in \mathbb{R}^{K_h \text{ x } K_w}$
3: Output: Matrix $O \in \mathbb{R}^{(I_h-1)+K_h \text{ x } (I_w-1)+K_w}$. Initialized with zeros.
4: Note: inclusive start index, exclusive stop index, and 1-indexed
5: for $i = 1$ upto $I_h$ do
6:     for $j = 1$ upto $I_w$ do
7:         $O[i : j + K_h \ , j : j + K_w] += I[i, j] \cdot K$
8:     end for
9: end for
10: return $O$
```

For example, the first step is $O[1 : 1 + 3, 1 : 1 + 3] += 2 * K$ to get

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

. . . this continues for 16 total steps and a final matrix of

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

4 Q4

4.1 Q4 downsampling parameters

The total number of downsampling parameters is $\theta_{down} = 58,810,528$.

First, find the total number of downsampling layers by using the formula

$$(h_{out}, w_{out}) = (\frac{h_{in} - k + 2p}{s} + 1, \frac{w_{in} - k + 2p}{s} + 1)$$

Simplifying with $k = 3, s = 1, p = 0$ reveals that each downsampling layer reduces both height and width by 2. The number of layers can be solved for using the input dimension, 224, and desired latent dimension, 16. Let L_{down} be the total number of layers in the downsampling section of the architecture

$$\begin{aligned} 224 - 2L_{down} &= 16 \\ L_{down} &= 104 \end{aligned}$$

Next, use the following formula to compute the number of parameters per downsampling layer.

$$\#\text{ of parameters} = (l \cdot m \cdot n + 1)k$$

Based on the given feature map count during downsampling, the 6th ... 104th layer (99 layers total) have all the same parameters.

$$\theta_{down,6:104} = 99[(256 \cdot 3^2 + 1)256] = 58,417,920$$

For layers 1 to 5 of downsampling, the parameter counts are

$$\begin{aligned} \theta_{down,1} &= (3^3 + 1)16 = 448 \\ \theta_{down,2} &= (16 \cdot 3^2 + 1)32 = 4640 \\ \theta_{down,3} &= (32 \cdot 3^2 + 1)64 = 18496 \\ \theta_{down,4} &= (64 \cdot 3^2 + 1)128 = 73856 \\ \theta_{down,5} &= (128 \cdot 3^2 + 1)256 = 295168 \end{aligned}$$

Also, there are $L_{down} = 104$ bias terms.

In total, there are $\theta_{down} = 58,810,528$ parameters for the downsampling step.