Homework 1 Written Questions

CSCI 1430

Template Instructions

This document is a template with specific answer regions and a fixed number of pages. Given large class sizes and limited TA time, the template helps the course staff to grade efficiently and still focus on the content of your submissions. Please help us in this task:

- Make this document anonymous.
- Questions are in the orange boxes. Provide answers in the green boxes.
- Use the footer to check for correct page alignment.
- **Do NOT remove the answer box.**
- **Do NOT change the size of the answer box.**
- **Extra pages are not permitted unless otherwise specified.**
- **Template edits or page misalignment will lead to a 10 point deduction.**

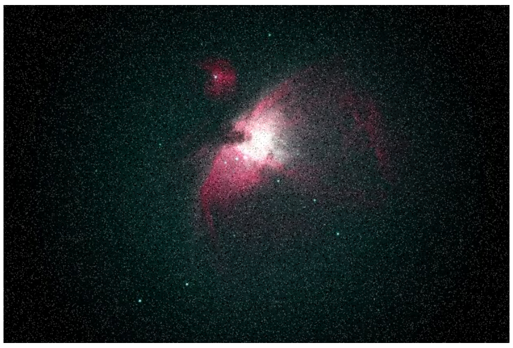
Gradescope Submission

- Compile this document to a PDF and submit it to Gradescope.
- Pages will be automatically assigned to the right questions on Gradescope.

This Homework

- 5 questions **[12 + 8 + 10 + 6 + 14 = 50 points]**.
- Include code, images, and equations where appropriate.

Q1 — [12 points]
We have been given special permission to use the telescope on the roof of Barus and Holley. Unfortunately, our fantastic image of the Orion nebula has noise: (orion-noise.png)



One way to deal with this noise is with image convolution. Convolution is a type of image filtering that is a fundamental image processing tool.

Explicitly describe the input, transformation, and output components of 2D discrete convolution. Please be precise; define variables as need.

Q1.1.1 — [2 points]

Input **[2–4 sentences]**

Q1.1.2 — [2 points]

Transformation (how is the image transformed?) **[2–4 sentences]**

Only Q1 and Q1.1.1 and Q1.1.2 should be on this page

Q1.1.3 — [2 points]

Output **[2–4 sentences]**

Q1.2 — [4 points]

Describe two filter kernels that we may use with convolution, and give an example computer vision application that each enables. **[4–8 sentences]**

Q1.3 — [2 points]

What kind of filter might we use to de-noise our image of the Orion nebula, and why? **[2–3 sentences]**

Q2 — [8 points]

Now that we've de-noised our image of the Orion nebula, let's explore filtering techniques more closely. Two kinds of linear filtering are correlation and convolution.

Q2.1 — [3 points]

What is different between convolution and correlation? Include differences in their algebraic properties. When might we use each? **[5–6 sentences]**

To solidify our understanding of the distinction between correlation and convolution, we will process another image.

Q2.2 — [4 points]

Devise a scenario in which the output of correlation and convolution differ.
Write code that loads an image and produces two distinct images, one from convolution and one from correlation on some kernel of your choice. Then, compute the difference of the two images (the order in which you subtract the images should not matter) and display it as well.
Specify your kernel, and provide the input image and two output results. Then, use your understanding of convolution and correlation to explain the outputs. **[2–4 sentences]**

Consider scipy.signal.convolve2d and scipy.signal.correlate2d to experiment!

Only Q2 and Q2.1 and Q2.2 should be on this page

TODO_orig_image.png	TODO_difference.png
TODO_conv_result.png	TODO_corr_result.png

TODO: Your explanation here.

Q2.3 — [1 point]

Consider a situation where we apply two different filters sequentially to an image. How will the output image change depending on the order in which we apply the filters? Will the behavior be different for convolution versus correlation? **[1–2 sentences]**

Q3 — [10 points]

Differentiating convolution. Convolution as an operation is defined only by multiplications and additions of the values within the filter and the image inputs. When interpreting these values as real $\mathbb{R}$ numbers (floating point) and not integers $\mathbb{Z}$, addition and multiplication are smooth functions. Thus, we can analytically differentiate convolution to compute a derivative with respect to its inputs and outputs, such that we can use continuous optimization methods to minimize an objective defined in those terms.

Of course, even though we might use continuous values within, discrete convolution is still defined upon the discrete pixel grid—we cannot analytically differentiate with respect to grid size terms, e.g., the kernel width or height, which are still integers. We can only analytically differentiate with respect to the values within the kernel.

So, given a target image and an original image, suppose we wish to find the kernel that produced the target image. Let $T \in \mathbb{R}_{M\times N}$ be the target image, $I \in \mathbb{R}_{M\times N}$ be our original image, and our kernel $f \in \mathbb{R}_{K\times L}$. As a reminder, convolution is defined as:

$$H = f * I = \sum_{k,l} f[k,l]I[m-k,n-l]$$

Q3.1 — [1 point]

Write down a mean-squared error (MSE) loss function $\mathcal{L}$ between the convolved original image and the target, where the mean is over all pixels. Include summation terms over M, N and index pixels with m, n.

$$\mathcal{L} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}((f*I)[m,n] - T[m,n])^2$$

$$\mathcal{L} = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}(H[m,n] - T[m,n])^2$$

Only Q3 and Q3.1 should be on this page

Q3.2 — [7 points]

Next, find an equation for the partial derivative of the MSE loss with respect to one of the filter weights:

$$\frac{\partial \mathcal{L}}{\partial f[k,l]}$$

1. This will require an application of the chain rule as this derivative depends upon the convolution output H. Start here.
2. As $f[k,l]$ contributes to every pixel in the output H and so every pixel contributes to $\mathcal{L}$, your application must sum the gradients over all output pixels M, N.
3. When differentiating the loss term wrt. H, recall that the loss itself is defined as a sum over all pixels—use dummy iterators, e.g., m', n' to keep track. But, we're taking the derivative with respect to a specific pixel location, so what happens at those other locations?

Apply the chain rule:

$$\frac{\partial \mathcal{L}}{\partial f[k,l]} = \sum_{m=1}^{M}\sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial H[m,n]} \cdot \frac{\partial H[m,n]}{\partial f[k,l]}$$

Consider the first term (outer derivative), and substitute the loss into the equation. Note that we must introduce two sets of m, n variables: m, n index the specific pixel we are differentiating with respect to, and m', n' are iterators that cover the whole image to compute the loss.

$$\frac{\partial \mathcal{L}}{\partial H[m,n]} = \frac{\partial}{\partial H[m,n]}\left[\frac{1}{MN}\sum_{m'=1}^{M}\sum_{n'=1}^{N}(H[m',n'] - T[m',n'])^2\right]$$

For all entries where $m \neq m'$ and $n \neq n'$, the derivative will be 0—the error at some other pixel does not affect the error at the pixel we care about. Thus,

$$\frac{\partial \mathcal{L}}{\partial H[m,n]} = \frac{\partial}{\partial H[m,n]}\left[\frac{1}{MN}(H[m,n] - T[m,n])^2\right]$$

$$\frac{\partial \mathcal{L}}{\partial H[m,n]} = \frac{2}{MN}(H[m,n] - T[m,n])$$

—

Now consider the second term (inner derivative), and substitute in the equation for convolution. We introduce dummy variables k', l' similarly.

$$\frac{\partial H[m,n]}{\partial f[k,l]} = \frac{\partial}{\partial f[k,l]}\sum_{k',l'} f[k',l']I[m-k',n-l']$$

Again, for all entries where $k \neq k'$ and $l \neq l'$, the derivative is 0. Thus,

$$\frac{\partial H[m,n]}{\partial f[k,l]} = \frac{\partial}{\partial f[k,l]} f[k,l]I[m-k,n-l]$$

Since $I[m-k,n-l]$ acts as a constant coefficient with respect to $f[k,l]$, the derivative of the product is simply the coefficient:

$$\frac{\partial H[m,n]}{\partial f[k,l]} = I[m-k,n-l]$$

—

Substituting both terms back in:

$$\frac{\partial L}{\partial f[k,l]} = \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\frac{2}{MN}(H[m,n] - T[m,n]) \cdot I[m-k,n-l]\right)$$

Q3.3 — [2 points]

Given what we've learned about filtering so far, provide an interpretation of your definition of $\frac{\partial L}{\partial f[k,l]}$. What is it?

This has a clean interpretation: the gradient for convolution MSE at a kernel pixel location *is* the correlation (or cross-correlation) between the (scaled) error map and the input image shifted by that kernel pixel location offset. This can be more easily seen if we think about H, T, I as whole matrices 'outside' the summation rather than as individual pixels inside it. I is offset by k, l, and is correlated with the (scaled) error $H - T$.

Only Q3.3 should be on this page

Convolution with an MSE loss is convex: MSE is a sum of squares (parabola, with a single valley), and since convolution is a linear operator it can only stretch or rotate the parabola. Convolution can never create a ‘second valley’ within this parabola.

Seeing convolution as a linear operator provides additional interpretations. Suppose we again wish to recover a kernel f given a target and original image, but our input or target image had noise. We could form linear systems like $Ax = b$ to solve least squares regression problems that find a kernel of best fit.

$$f * I = \sum_{k,l} f[k,l] I[m-k, n-l] = Ax = T$$

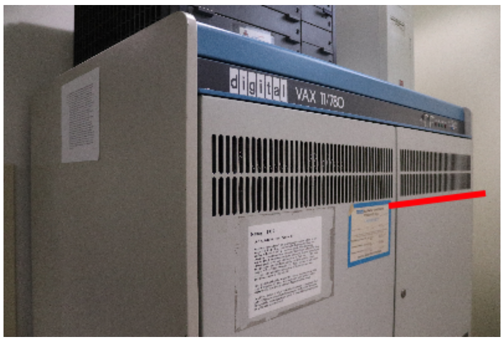
Each row of A would be a 2D patch of I that have been flattened, such that $A \in \mathbb{R}^{(m \cdot n) \times (k \cdot l)}$. x would be the flattening of f into a column vector, and b would be the corresponding flattening of T into a vector, such that multiplication Ax produces the result of the convolution T in its corresponding vector of elements b.

Q3.4 — [2 points]

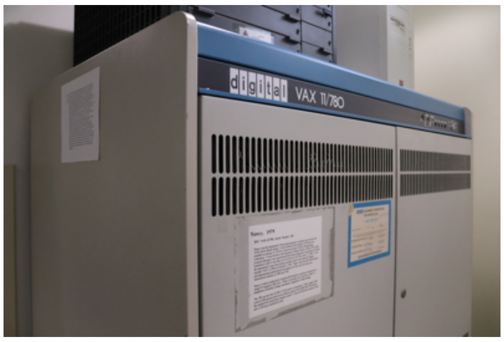
Write pseudocode that creates matrix A.

```
for m in range(H):
    for n in range(W):
        # flatten patch
        patch = I[m:m+K, n:n+L].flatten()
        # set row to equal the patch
        A[row, :] = patch
        row += 1
```

Q4 — [6 points]
While exploring Brown CS's history in the halls of CIT, we happen upon Nancy: a DEC VAX 11/780. So struck by its beauty, we decide to take an artful photo with our camera. Modern digital sensors have many megapixels, so we resize it to make the file smaller.



Resized image



Depiction of original (*not original file*)

Oh no! What happened to Nancy? There are weird artifacts in the vents (above red line)—these definitely weren't there in the original photo. Plus, if we look closely, the white label text is less smooth and there are jagged lines.

Q4.1 — [3 points]

What is this phenomenon called, and why did it happen? **[2–4 sentences]**

Q4.2 — [3 points]

How might we fix this issue with filtering? Describe the process, and explain why it works. **[2–4 sentences]**

Q5 — [14 points]
With filtering, we can create *hybrid images* that depict different objects when viewed at different distances. They are inauthentic images of the natural world.

As technology advances, evaluating the authenticity of images becomes increasingly difficult. Please read this article by photography critic Andy Grundberg in the *New York Times* from August 1990.

Grundberg stated that: "In the future, readers of newspapers and magazines will probably view news pictures more as illustrations than as reportage, since they can no longer distinguish between a genuine image and one that has been manipulated."

Q5.1 — [4 points]

When is Grundberg's future, and why? **[4–6 sentences]**

Q5.2 — [4 points]

For a news picture, are any digital manipulations permissible? If so, how do we decide which ones? Use at least one ethical framework from the ethics primer to explore this. **[4–6 sentences]**

The Coalition for Content Provenance and Authenticity (C2PA) has designed a technical specification to attach history to an image. Please watch this video for an overview; stop at 4 minutes and 40 seconds.

For context, a stakeholder of a system is identified as a person or group who has an explicit interest or concern in the system itself, its operations, and its consequences.

Q5.3 — [4 points]

Describe one situation in which the C2PA system helps us in determining authenticity and identify one stakeholder that benefits. Then, describe a second situation in which C2PA does not help to determine authenticity and one stakeholder that is still at risk. Please explain why C2PA helps or not in each case. **[4–6 sentences]**

Q5.4 — [2 points]

Grundberg's article is titled "Ask It No Questions: The Camera Can Lie." Does the C2PA system weaken Grundberg's argument? **[1–2 sentences]**

Q6:

Technical practice (optional). In computer vision, each image is a matrix of pixels. The `numpy` library provides fast computation with large multi-dimensional vectors and matrices.

To familiarize yourself with the library, read through the following scenarios and complete the exercises. It is possible to use *one* `numpy` function to complete each of the following tasks.

With `numpy` imported as

```
    import numpy as np
```

we can call functions with

```
    np.function_name(<arguments>)
```

Test out your answers by creating your own python program. Some functions you might find useful are np.squeeze, np.expand_dims, np.clip, np.pad, and np.zeros.

Use operators like `[]` and `:`, but remember that each prompt can be completed with only *one* function/operator shorthand.

Q6.1:

Create a black image `img` with all values in this matrix equal to 0, where `np.shape(img) == (320,640)`.

```
# TODO: Your expression here
```

Q6.2:

Assume we have a 2D matrix `img` with values range [-1.0, 1.0]. Clip `img` so that all its values lie within the range [-0.5, 0.5].

```
# TODO: Your expression here
```

Q6.3:

A malformed filter operation has messed up the output dimensions, producing a variable `img_out` where `np.shape(img_out) == (1, 1, 320, 640)`. Remove all 1-sized dimensions. Convert `img_out` to a new matrix `img_fixed` where `np.shape(img_fixed) == (320, 640)`.

```
# TODO: Your expression here
```

Color images are represented with a three dimensional matrix, where often the third dimension represents spectral information. The presence of a third dimension or the size of the color dimension could help us to identify whether an image is color (RGB) or grayscale.

Q6.4:

Say you have a grayscale image `img` where `np.shape(img) == (320, 640)`. Convert this to a new image `img_expanded` where `np.shape(img_expanded) == (320, 640, 1)`. In other words, add a 1-sized dimension to `img`.

```
# TODO: Your expression here
```

Q6.5:

Suppose we have an RGB image matrix, `img`, of shape (320, 640, 3). Retrieve the third blue channel of the image while preserving all of `img`'s dimensions and intensity values.

```
# TODO: Your expression here
```

Q6.6:

Suppose we have a second RGB image matrix, `img2`, also of shape (320, 640, 3). Retrieve the red and blue channels of `img2` within a single variable of shape (320, 640, 2).

```
# TODO: Your expression here
```

Q6.7:

Padding is a useful operation to help us produce a convolved image equal in size to an input image. Given an RGB image, `img` of (320, 640, 3), pad it with two columns of zeros on the left and right edges of the image, and three rows of zeros on the top and bottom edges of the image. Do not add zero padding to the color dimension.

```
# TODO: Your expression here
```

Feedback? (Optional)

We appreciate your feedback on how to improve the course. You can provide anonymous feedback through this form which can be accessed using your Brown account (your identity will not be collected). If you have urgent non-anonymous comments/questions, please email the instructor.

Only feedback should be on this page