Introduction to Machine Learning

CNNs

Due: Wednesday, April 10, 2024 at 11:00 PM

1) CNN Architectures

In order to get practice with the structural choices in a CNN and understand how parameters are shared, we will consider how many separate weights are in NNs of different architectures.

Hint: For all the questions in this section, if you get stuck, try drawing a picture/diagram of what the network would look like.

1.1)

In a fully-connected feedforward network, the number of weights (including biases) for one layer with 100 inputs and 80 outputs is:

Enter a number: 8080

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

1.2)

Consider a CNN for 1D data. Suppose a convolutional layer receives an input of size 100. This layer has 10 filters, each of size 5. Suppose we use a zero-padding of size 2 on both ends of the input, and a stride of 1.

What are the dimensions of the output of this convolutional layer?

Provide a list of numbers of appropriate length. For example, if the image was 3D, the answer could be [10, 10, 10].

[100, 10]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

1.3)

How many weights (including bias) are needed to specify the map from the input to the output of this convolutional layer?

Enter a number: 60

Check Formatting | Submit | View Answer 100.00%

You have infinitely many submissions remaining.

1.4)

Now consider a zero-padded max pooling layer with an input of size 100, a pooling filter of size 3, and a stride of 2. Assume zero-padding of size 1.

What are the dimensions of the output of this pooling layer?

Provide a list of numbers of appropriate length. For example, if the image was 3D, the answer could be [10, 10, 10].

[50]

Check Formatting | Submit | View Answer 100.00%

You have infinitely many submissions remaining.

1.5)

How many weights (including bias) are needed to specify the max pooling layer?

Enter a number: 0

Check Formatting | Submit | View Answer 100.00%

You have infinitely many submissions remaining.

2) Conversion from Convolutional to Fully Connected

Let's now formalize the relationship between convolutional layers and fully connected layers.

Suppose we have a:

- 1D input x of size 4
- a filter f of size 3: (w_1, w_2, w_3) with stride of 1 and no bias.
- a 1D output z of size 4, produced by convolving f with x. To preserve size, we use padding of size 1 on both sides.

Assume that when doing convolutions, inputs that fall outside the image indices are treated as zeros.

The effect of convolving this filter with the input layer x is actually equivalent to constructing a matrix W of weights between two layers of a fully connected network, i.e.,

$$z = W^\top x \ ,$$

where W is a 4 by 4 matrix (and there is no bias term). Note though that in the conversion, the same weight may occur more than once in the matrix W.

Which of the following weight matrices corresponds to W^T in the equation above? That is, z was produced using a convolution between f and x, but if we want to produce z using W and x, as defined above, what would $W^\top$ be?

Hint: What is the output z, in terms of the elements of w and x?

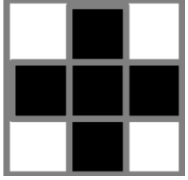
- $$\begin{bmatrix} w_2 & w_3 & 0 & 0 \\ w_1 & w_2 & w_3 & 0 \\ 0 & w_1 & w_2 & w_3 \\ 0 & 0 & w_1 & w_2 \end{bmatrix}$$
- $$\begin{bmatrix} w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \end{bmatrix}$$
- $$\begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \end{bmatrix}$$
- $$\begin{bmatrix} w_1 & w_2 & w_3 & 0 \\ 0 & w_1 & w_2 & w_3 \\ 0 & 0 & w_1 & w_2 \\ 0 & 0 & 0 & w_1 \end{bmatrix}$$

Submit View Answer 100.00%

You have infinitely many submissions remaining.

3) Convoluted Network

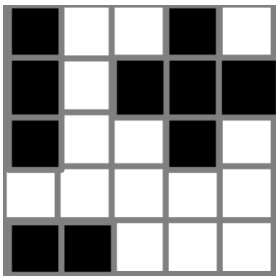
We will explore how convolutional neural networks operate by designing one. Our objective is to be able to locate the pattern



in an image. Throughout this problem, treat dark squares as having value $+1$ and light squares as having value -1.

3.1)

Consider the image that would result from convolving the image below with a filter that is the same as the pattern above. (Use our definition of convolution, in which we slide the **center of** the filter over the image and compute the dot product.) Assume that the edges are padded with -1 and that we use a stride of 1.



Which pixel in the **resulting** image has the largest value, and what is the resulting value?

Provide your answer as a list `[[row, column], value]`, where the first entry is the pixel indices and the second is its value. The pixels are indexed such that the top-left pixel is `[0, 0]`.

[[1,3], 9]

Check Formatting | Submit | View Answer | 100.00%

You have infinitely many submissions remaining.

3.2)

In order to detect this pattern, we could create a very simple network that has

- a convolutional layer with a single filter, corresponding to the desired pattern, applied with a stride of 1,
- no activation function,
- a max-pooling layer with input size equal to the image size, and finally
- a single ReLU unit.

Note that there is no need for additional `flatten` or fully-connected layers because a whole image (result of convolving with our filter) goes into the max pooling and a single value comes out.

Provide a value for the offset W_o on the input to the ReLU that, for any image, would guarantee the output of the ReLU is positive if and only if there is a perfect instance of this pattern in the image.

-8

Check Formatting | Submit | View Answer 100.00%

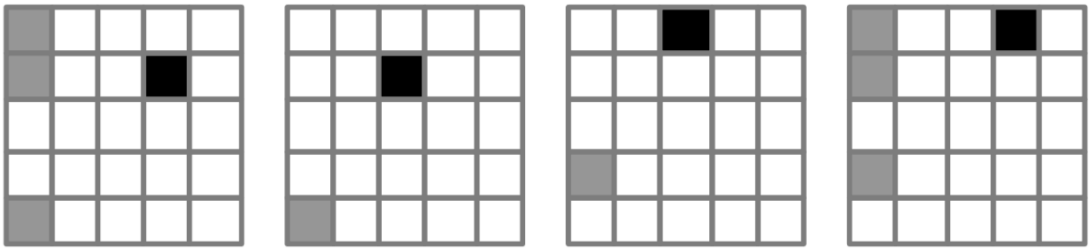
You have infinitely many submissions remaining.

3.3)

Kan Volution thinks that instead of having this single convolution layer with a single filter matching the whole desired pattern, it would be better to start with a convolutional layer with four smaller filters, shown below (with black squares having value +1/4 and white squares having value -1/4):

It is slightly unusual to have 2×2 filters (usually they have odd dimension). When we apply them, we change the convention a little, and place the upper-left pixel (instead of the usual center) of the filter on top of the image pixel whose value we are computing. We use -1 padding for one column on the right and one row on the bottom of the input image.

The following images are the result of convolving the original image with these 4 simple filters and running through a ReLU. Black squares have value $+1$, grey squares have value $+0.5$, and the rest have value 0. Try to convince yourself that these 4 resulting images (and their values) make sense.



Now, the next layer of Kan's network takes in this 3D tensor (of shape $5 \times 5 \times 4$, with 4 channels) and applies a single $2 \times 2 \times 4$ filter.

Specify a filter on this next layer that will generate a final image with a high value at the pixel located at the upper left corner of the original 3x3 pattern we are trying to detect, and lower values elsewhere.

Provide your answer as a (nested) list for the four filters. For example `[[[-1, 1], [1, 1]], [[1, -1], [1, 1]], [[1, 1], [-1, 1]],[[1, 1], [1, -1]]]` describes four 2-by-2 filters where each filter has one light pixel. In particular, the first filter `[[-1, 1], [1, 1]]` has a light pixel at the top left; the second filter has a light pixel at top right; the third bottom left; and the fourth bottom right.

Your list should follow the same shape/location pattern as the example, and each of the weights in your filters should be either 1 or -1.

[[[-1, -1], [-1, 1]], [[-1, -1], [1, -1]], [[1, -1], [-1, -1]], [[-1, 1], [-1,

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4) Kat's Cat

4.1)

In this section, we will train CNNs to classify images. As a warmup, we first look at a CNN with one convolutional layer and one activation layer with a one-dimensional input.

- Assume there is a sign activation function. The activation is $\text{sign}(z) = 1$ if $z > 0$, and $\text{sign}(z) = 0$ if $z < 0$. For this problem we will not allow boundary cases, so an error will be raised if $z = 0$ is the input to any $\text{sign}(z)$.
- Assume we will use zero-padding of size 1 to handle the image boundary.
- Assume a stride of 1.

4.1.1)

Provide a 3 x 1 filter and offset that produces the output when applied to the input:

```
input:  [0 0 1 1 0 0 1 0]
output: [0 0 0 0 0 0 1 0]
```

Enter the weights of a 3 x 1 filter and the bias as a list: $[w_1, w_2, w_3, b]$.

[-1, 1, -1, -.5]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4.1.2)

Now, imagine that we take two images:

```
[0 0 0 0 0 0 1 0]
[0 0 0 0 0 1 0 0]
```

and treat them as two channels that are the outputs of a convolutional layer. We convolve them with a (1 x 1 x 2) tensor filter `[[[1, 1]]]` to obtain a 1-channel 1D image. What is the resulting image?

Enter the resulting image as as a list:

[0, 0, 0, 0, 0, 1, 1, 0]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4.2)

Kat trains a CNN classifier to recognize their cat, but the cat is always in the lower left quadrant of the 100 x 100 training images, and it usually takes up a 20 x 20 image patch and there is no sub-image of size less than 20 x 20 that will reliably recognize the cat.

4.2.1)

For the rest of this problem, we will be talking about PyTorch. Please this view this note on PyTorch for more information about how to use it.

Kat makes a CNN in Pytorch with layers

```
[
    nn.Conv2d(1, 1, kernel_size = (20, 20), stride = 1, padding = 10),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size = (100, 100)),
    nn.Flatten(),
    nn.Linear(1, 1)
]
```

and `CrossEntropyLoss`. It has 100% training accuracy. Will it have high accuracy on test images with the cat in a 20-pixel window in the upper right quandrant?

- (●) Yes
- ( ) No

Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4.2.2)

If Kat's CNN were, instead

```
[
    nn.Conv2d(1, 10, kernel_size = (3, 3), stride = 1, padding = 1),
    nn.ReLU(),
    nn.Flatten(),
    nn.Linear(100000, 1)
]
```

would we expect it to be possible to train it to reliably recognize the cat in the training data?

- ○ Yes
- ◉ No

Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4.2.3)

If Kat's CNN were, instead

```
[
    nn.Conv2d(1, 10, kernel_size = (3, 3), stride = 1, padding = 1),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size = (3, 3), stride = 3),
    nn.Conv2d(10, 10, kernel_size = (3, 3), stride = 1, padding = 1),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size = (3, 3), stride = 3),
    nn.Conv2d(10, 1, kernel_size = (3, 3), stride = 1, padding = 1),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size = (3, 3), stride = 3),
    nn.Flatten(),
    nn.Linear(9, 1)
]
```

would we expect it to be possible to train it to reliably recognize the cat in the training data?

- ◉ Yes
- ○ No

Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4.2.4)

Kat's housemate Kit thinks a fully-connected network with many layers would be a better idea. So Kit makes each 100×100 image into a single input vector of size 10,000 and runs it through a standard fully-connected network with 3 hidden layers with non-linear activations and a single output. Would it be possible for Kit to train their network to have high training accuracy on Kat's training data?

- (●) Yes
- ( ) No

Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

4.2.5)

Would Kit's network have high accuracy on test data with cats in the upper right quadrant?

- ( ) Yes
- (●) No

Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

5) Image Classification

For the rest of this assignment, we'll make use of the following **CNN colab notebook.**

We'll be working with the MNIST handwritten digit image dataset. This is a collection of 70,000 black and white images of handwritten digits (although the pixel values of the images themselves are greyscale). The original dataset has integer pixel values from `[0, 255]`, but we will scale our pixel values to `[0. , 1.]` floats. Each image is (28 x 28) pixels. The labels are integers 0-9, so our output layer will always have 10 units. Because we are doing classification in this problem, we will use softmax activation on the output layer and negative log-likelihood loss.

Examples of handwritten digits from the MNIST dataset

5.1) No hidden layers

With one epoch of training, what is the validation accuracy of a network **with no hidden units** on this data?

Enter a number between 0 and 1 (to three decimal places): .851

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

5.2) One fully-connected hidden layer

Using one epoch of training, try a single hidden layer, followed by a ReLU activation layer before the final output layer, and gradually increase the units; specifically, try (128, 256, 512, 1024) units and observe the results. What are the accuracies on the validation set?

Enter a Python list of 4 numbers between 0 and 1 (to three decimal places):

[.900, .913, .921, .930]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

5.3) Two fully-connected hidden layers

Now, try a network with two hidden layers (for 1 epoch):

- A fully-connected layer with 512 hidden units
- A ReLU activation layer
- A fully-connected layer with 256 hidden units
- A ReLU activation layer
- A fully-connected layer with 10 output units

What is the accuracy on the test set?

Enter a number between 0 and 1 (to three decimal places): .922

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

Now run the model for 20 epochs. What is the validation accuracy after the 1st, 5th, 10th, and 15th epochs?

Enter a Python list of numbers between 0 and 1 (to three decimal places):

[.968,.973, .969,.975]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

5.4) Convolutional architectures

Finally, build a convolutional network with the following structure:

- A convolutional layer (without padding) with 32 filters of size 3 × 3
- A ReLU activation layer
- A max pooling layer with size 2 × 2
- A convolutional layer (without padding) with 64 filters of size 3 × 3
- A ReLU activation layer
- A max pooling layer with size 2 × 2
- A flatten layer (will flatten to size 1600 = 5 × 5 × 64; try to figure out why!)
- A fully-connected layer with 128 units
- A ReLU activation layer
- A dropout layer with drop probability 0.5
- A fully-connected layer with 10 output units

To define Convolutional and max pooling layers in PyTorch use the following syntax:

```
c = Conv2d(in_channels=i, out_channels=o, kernel_size=filter_size)
m = MaxPool2d(kernel_size=filter_size)
```

where `i` and `o` are integers and `filter_size` can either be an integer (for square filters) or a tuple (for non-square filters i.e. `(2, 3)` for 2x3 filter).

For all convolutional layers, you can assume that there is no padding and that stride=1.

Train the model for one epoch and report your test accuracy.

Enter a number between 0 and 1 (to three decimal places): .954

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

Now, assume that the accuracy of getting one digit correct is 0.9985. Again assume that predictions of digits are independent from each other. What is the probability of reading a full 5-digit zip code correctly?

Enter a number between 0 and 1 (to *four* decimal places): .9925

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

This is why people care about dropping the error rates to very low values.

Survey

(The form below is to help us improve/calibrate for future assignments; submission is encouraged but not required. Thanks!)

How did you feel about the **length** of this homework?

- ○ Too long.
- ◉ About right.
- ○ Too short.

How did you feel about the **difficulty** of this homework?

- ○ Too hard. We should tone it down.
- ◉ About right.
- ○ Too easy. I want more challenge.

Do you have any feedback or comments about any questions in this homework? Anything else you want us to know?

Submit