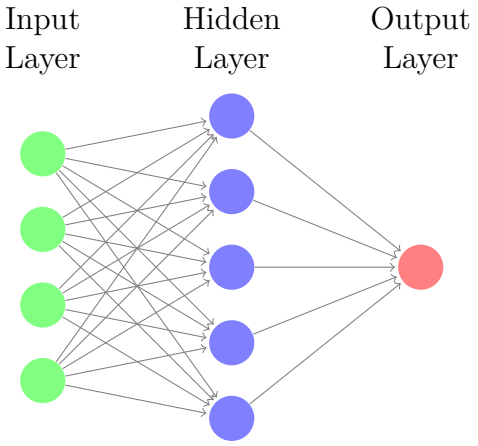
1. Suppose we are using a neural network for the task of recognizing handwritten digits (0-9). The input images are grayscale and are 10 pixels by 10 pixels and there are 10 possible digits. If we use a neural network with one layer of 50 hidden neurons, how many weights will need to be learned for the neural network? Write your answer in the space below. You should show your work for partial credit.

   As an example, we show a very small network with 4 inputs, one hidden layer of 5 neurons, and one output. This is just an example of what the network generally looks like, but these are NOT the numbers we want you to use for the calculation.



2. For this problem we will be writing code to handle image processing.

   (a) Write a function called `separate` that takes a color image (`(n, m, 3)` `numpy` array) and returns a new color image `numpy` array that has each color channel separated. To do this, we will create a new image that is 3 times as wide horizontally as the original image and then copying each channel to separate part of the result image.

   For example, if we were given an image of a puppy, this function should return an image that is 3 times as wide that has the contents

   RED channel puppy, GREEN channel puppy, BLUE channel puppy

   In each section, there should only be red, green, or blue pixels from the original image respectively. Note that, even though each of the sections contain one color channel, the resulting image should be colored so that each section can be seen in its respective color.

   For full credit, your solution should not modify the input image and should not have any loops that loop over the image.

(b) Write a function called `magnitude` that takes a gray-scale image (`numpy` array with shape `(n, m)`) and an integer patch size named `size`. The function should create a square window with width and height `size` that it moves over the image in a sliding-window algorithm computing the "magnitude" of the pixel values at each point. To compute the magnitude of a window, should compute each pixel squared, sum those values, and then take the square root of that number using `np.sqrt(v)`. For each position, the result should be stored in a `numpy` array which will be returned from the function. For example, if we had the following array named `img`

1	2	3
2	1	2
2	2	1
2	2	1

If we call `magnitude(img, 2)`, it would return a `numpy` array with the following values (we show the square roots for simplicity rather than the actual float values):

$\sqrt{10}$	$\sqrt{18}$
$\sqrt{13}$	$\sqrt{10}$
$\sqrt{16}$	$\sqrt{10}$

You should make no assumption about the size of the image except that it is larger than the patch size. You may assume the patch size is at least 1 and will be smaller than the width/height of the image. Your method should not modify the given array, but instead should return a new one. For full credit, your solution should only have two loops.