Our work will be the analytical backend version of what a PhD student with Prof. Polo Chau’s lab in CSE has done:

<http://haekyu.com/neural-divergence/>

Play around with this to get a feel for changes in activation patterns that occur with adversarial examples. In particular:

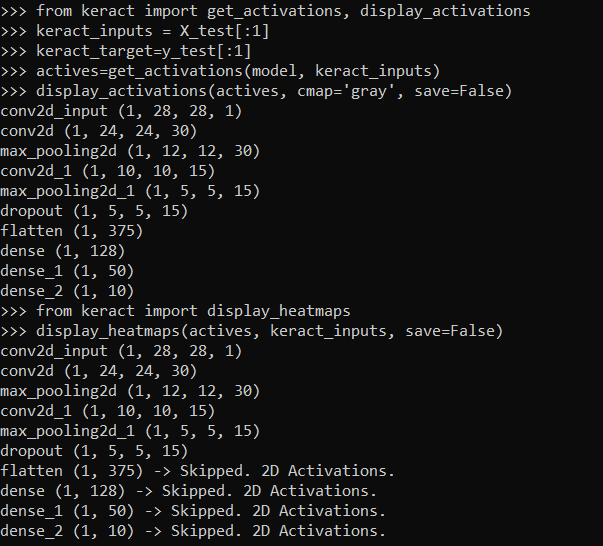
1. Try different pairs of attacked vs benign classes: all classes have been viewed
2. Play around with adding different layers: all layers have been viewed
3. In particular, view the very last ‘activation\_15’ layer. Given this is the logit layer, does the change in activation values correspond with the change in classification? Changing two (or three) of these ten values changed the classification, in the case of cat-dog or cat-airplane. It is possible that changing one value would change the classification, but the 20 observed pictures all changed two or three values.
4. What is the red tick mark in the layers vs the black one? (read Neural Divergence paper if necessary—link in website) The red rick marks are the activation values of the attack image, while the black tick marks are from the benign image
5. Do earlier layers exhibit more or less changes than later ones? Earlier layers exhibit little differences between black and red tick marks- the seventh convolutional layer (21st overall layer) produced visible differences, which were further defines by the eighth convolutional layer. The fifth and final max pooling layer, which is the 45th out of 50 layers, further exacerbated differences.

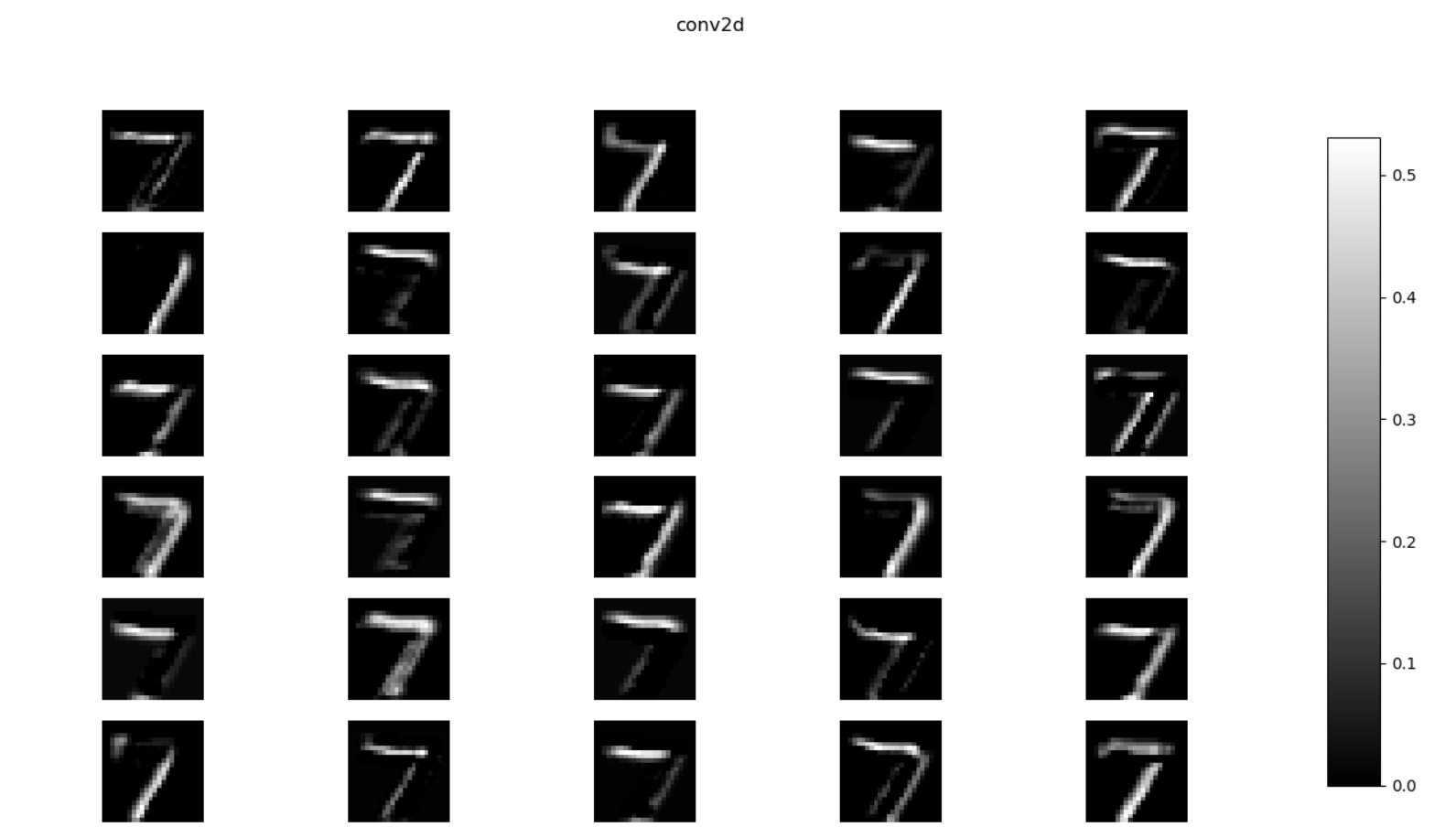
Instead of just visualizing differences, we’ll be analyzing them statistically.

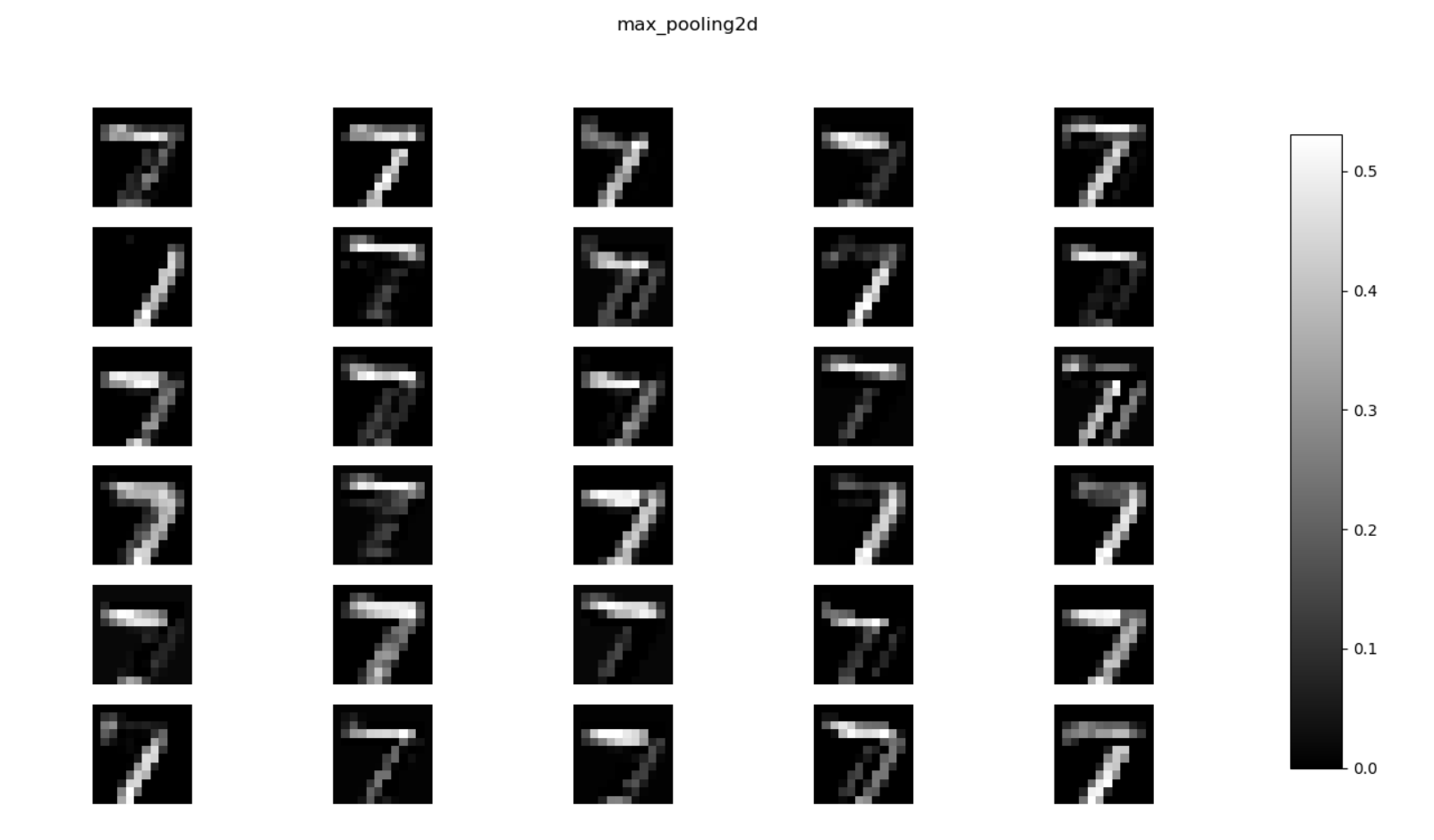
In order to extract activation values, we’ll be using the Keract module:

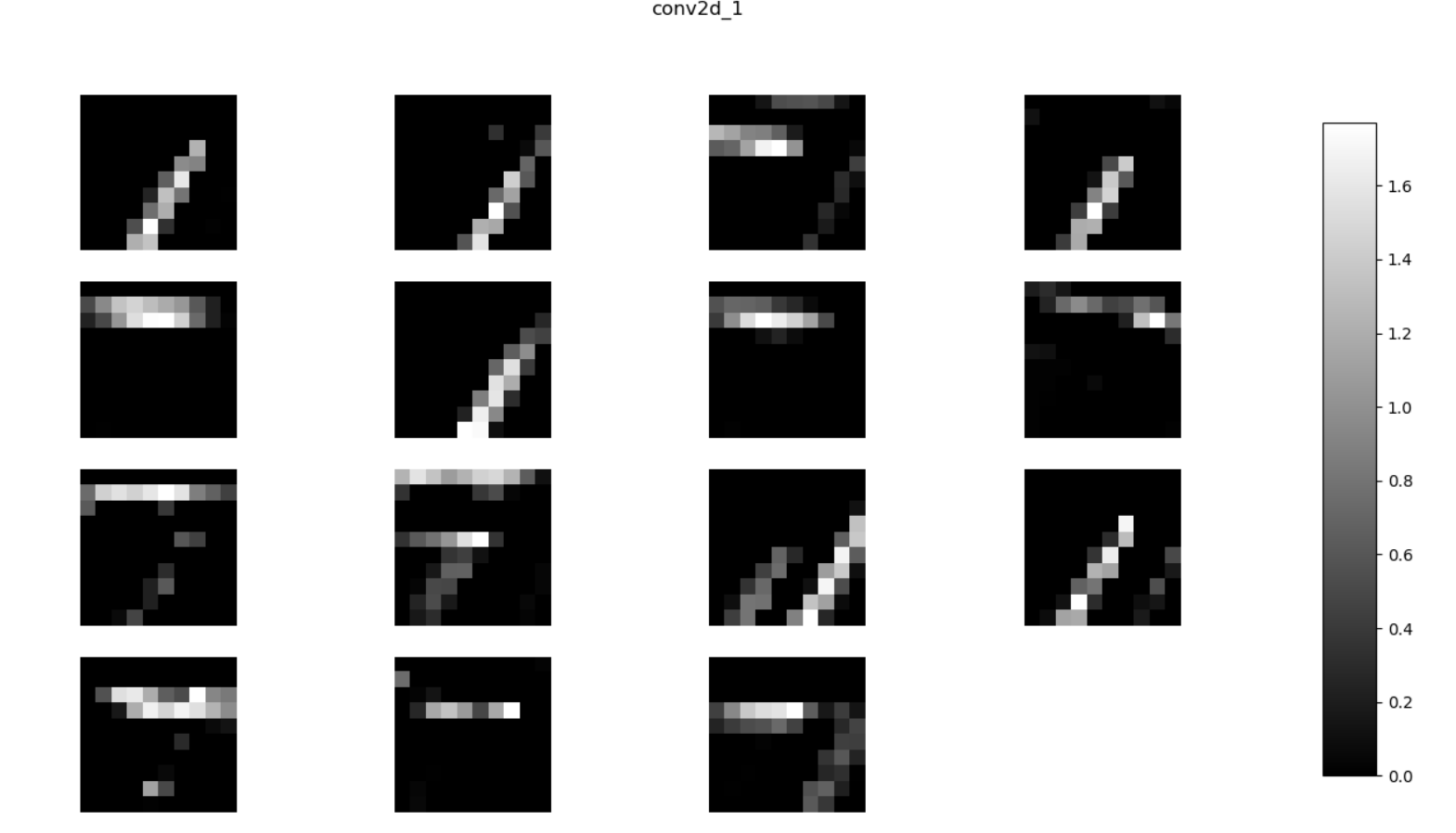
Assignment:

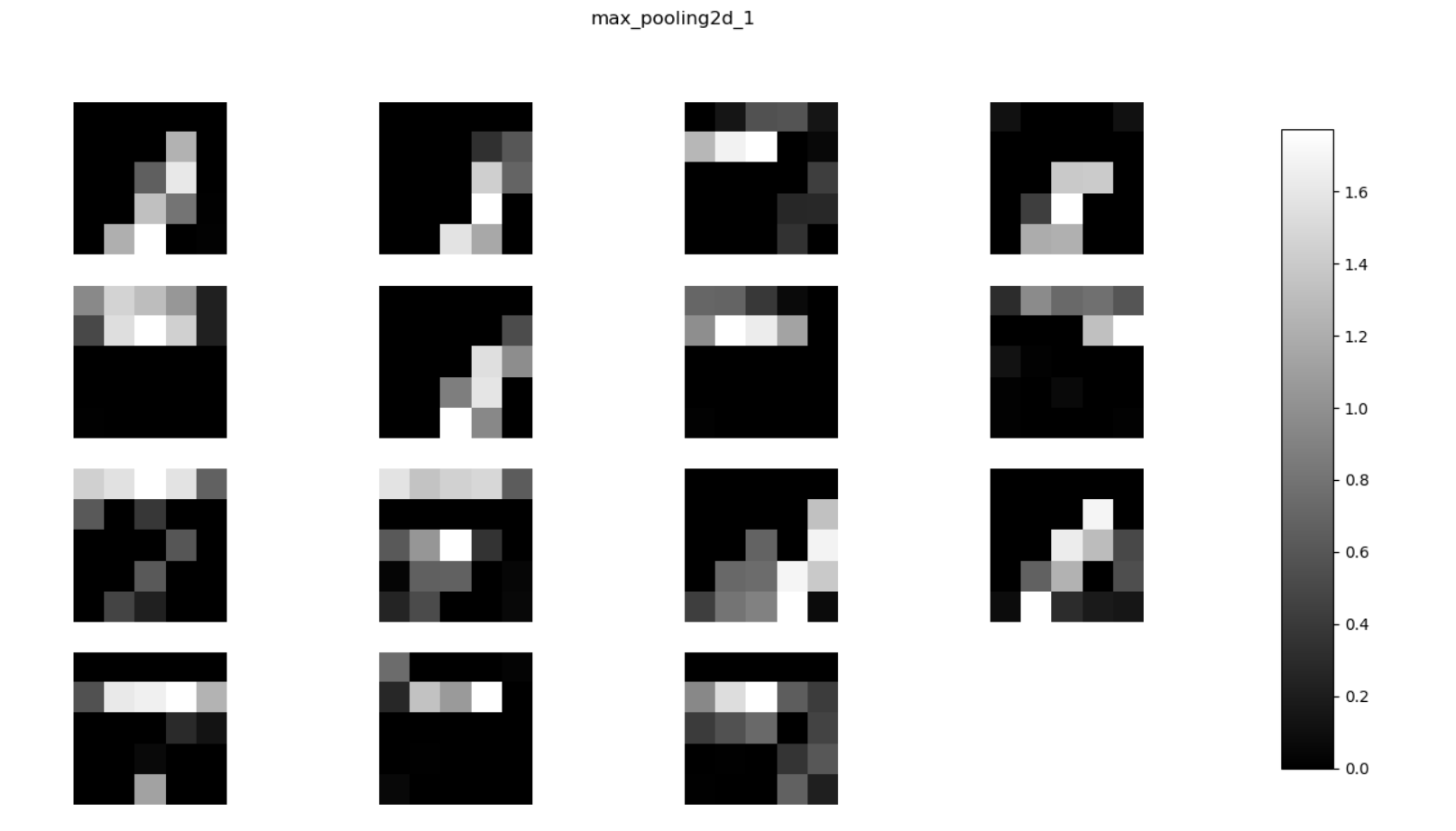
1. Play around with Neural Divergence and answer the questions.
2. Download and install the Keract module:
   1. https://github.com/philipperemy/keract
   2. (earlier versions work with TF 1 if that’s how you’re rolling).
   3. Read through the README to get a sense of what it does and how to use it.
   4. Read up on it: <https://www.machinecurve.com/index.php/2019/12/02/visualize-layer-outputs-of-your-keras-classifier-with-keract/>
3. Try out the MNIST tutorial in that last link (point d above). Either add in the 2 Conv2d layers in the front of your model as in the tutorial, or try to **get heatmaps to work on the dense layers, to see what’s being activated**. It’s especially neat to use with convolutional layers, because each convolution is looking for a particular pattern. This helps to visualize what each convolution finds: a curve? A straight line? What is being triggered in the image? Save a few images of your heat map and send to me or upload to you GitHub.

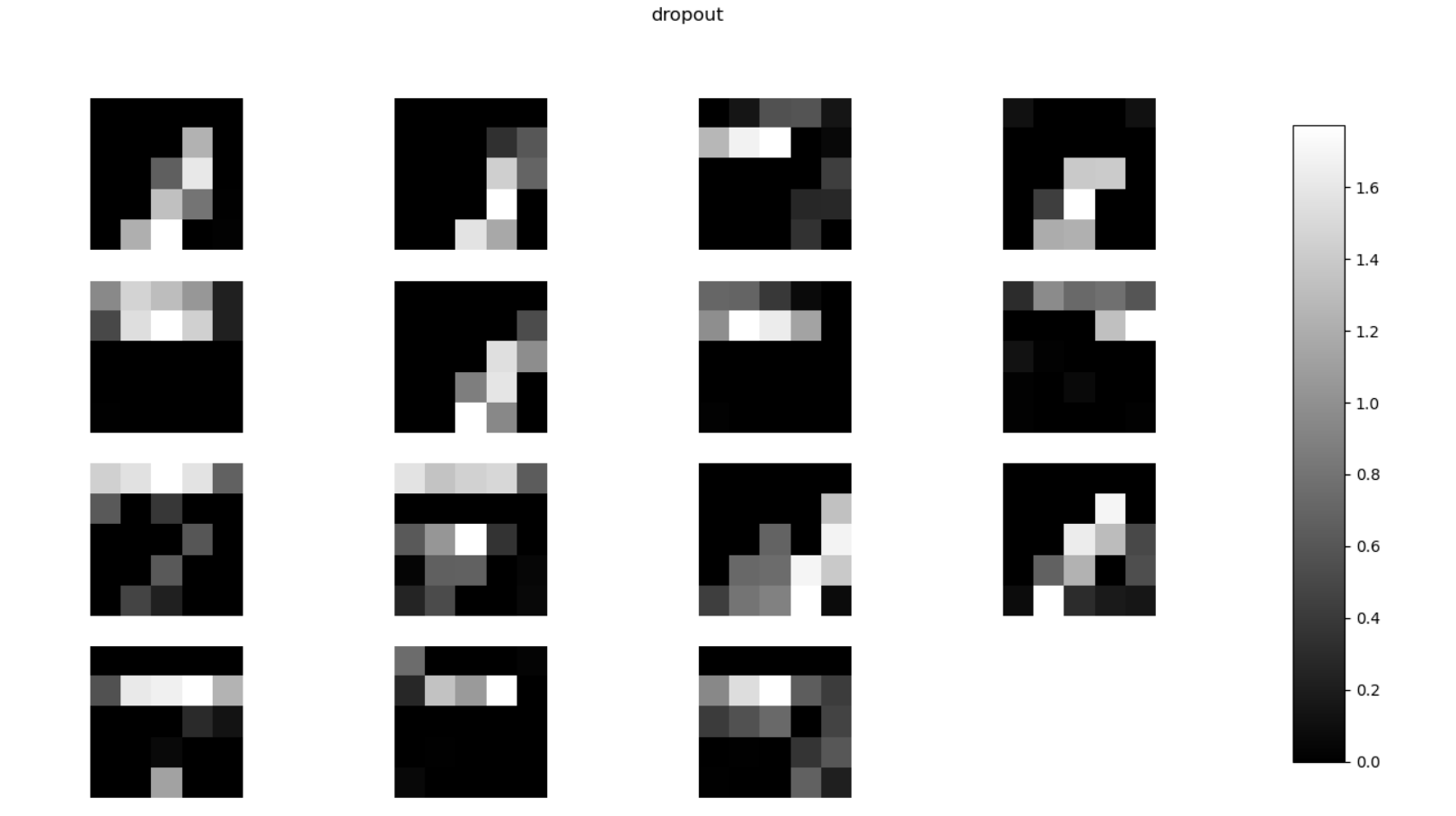


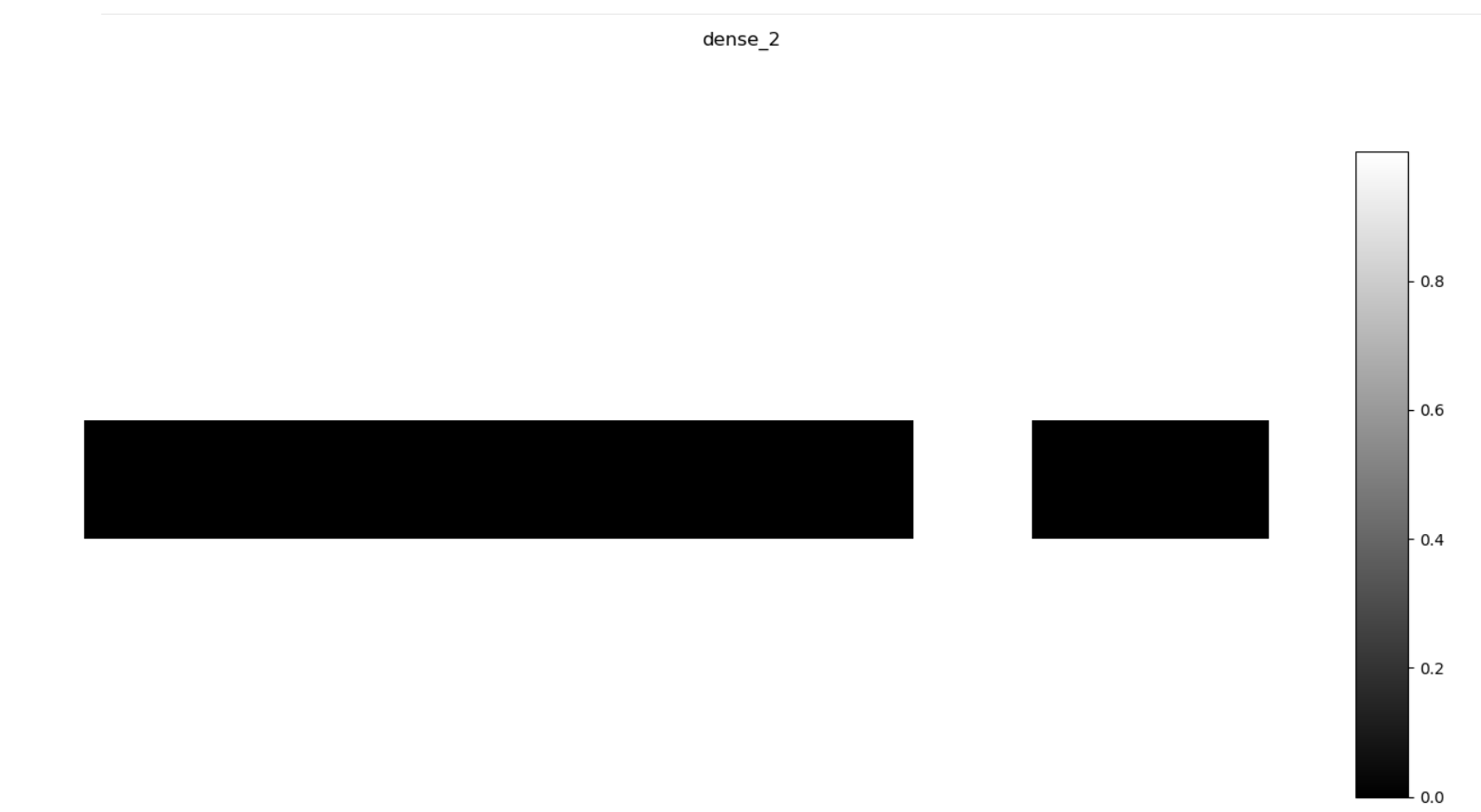
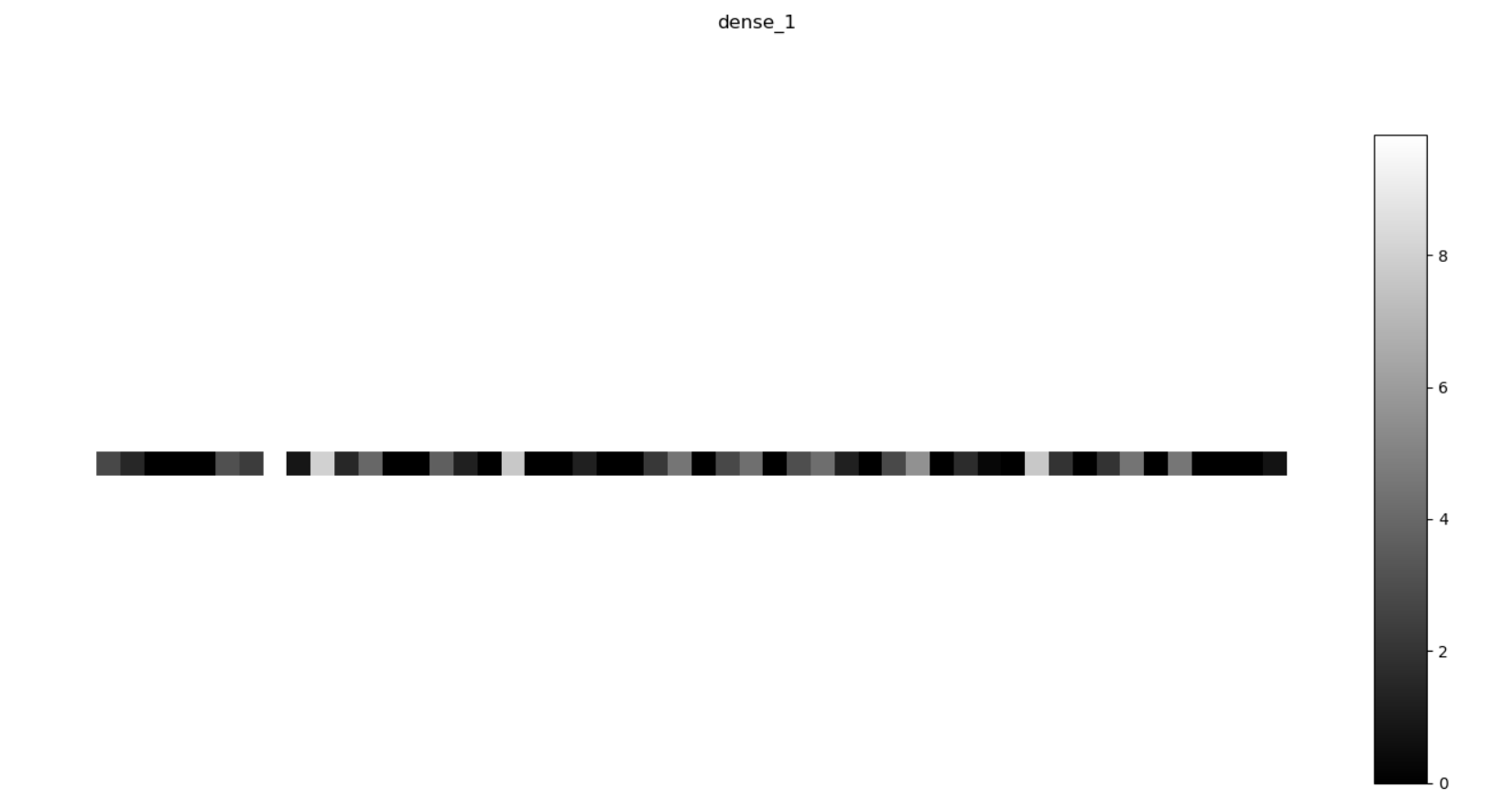
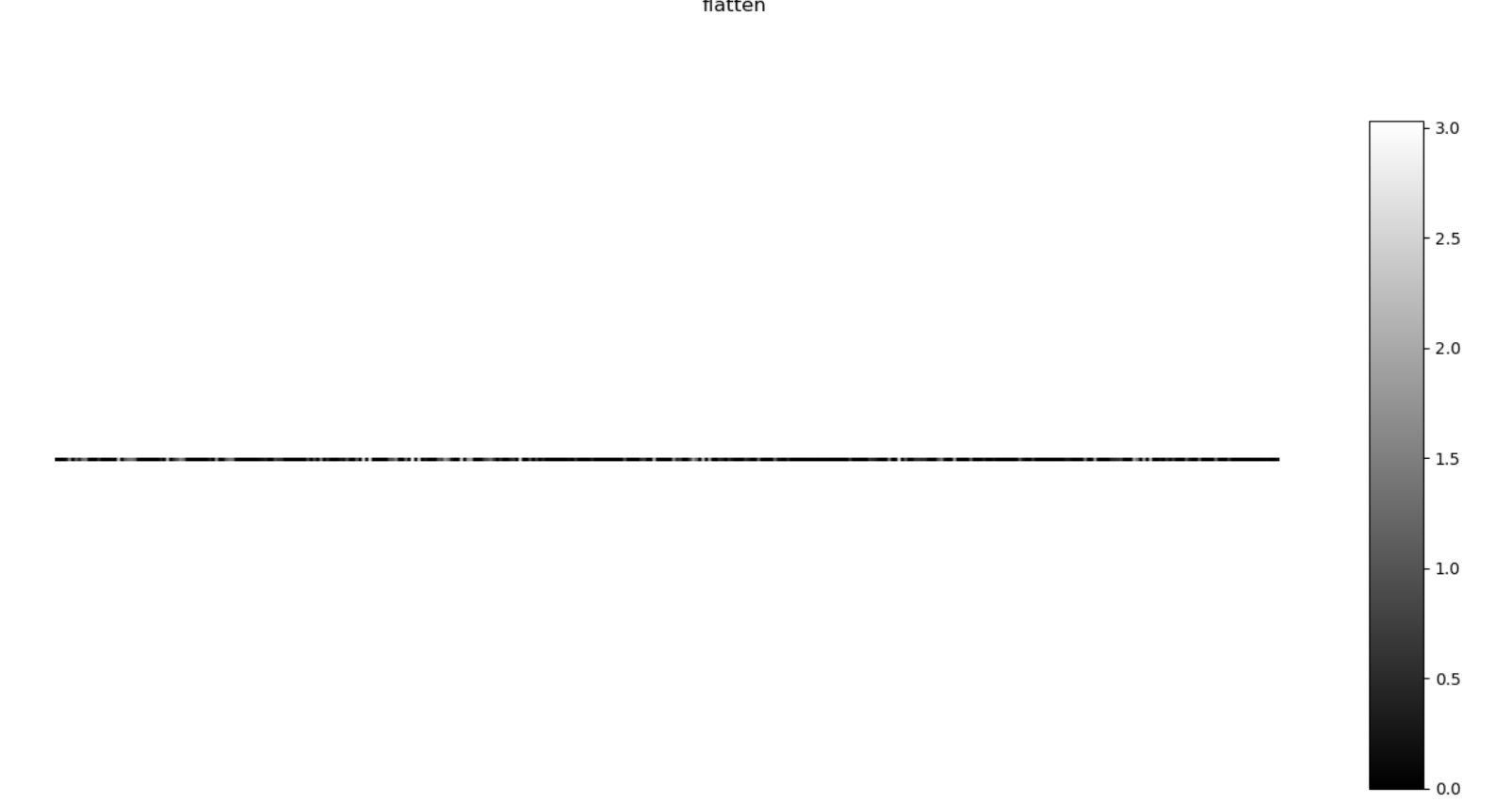


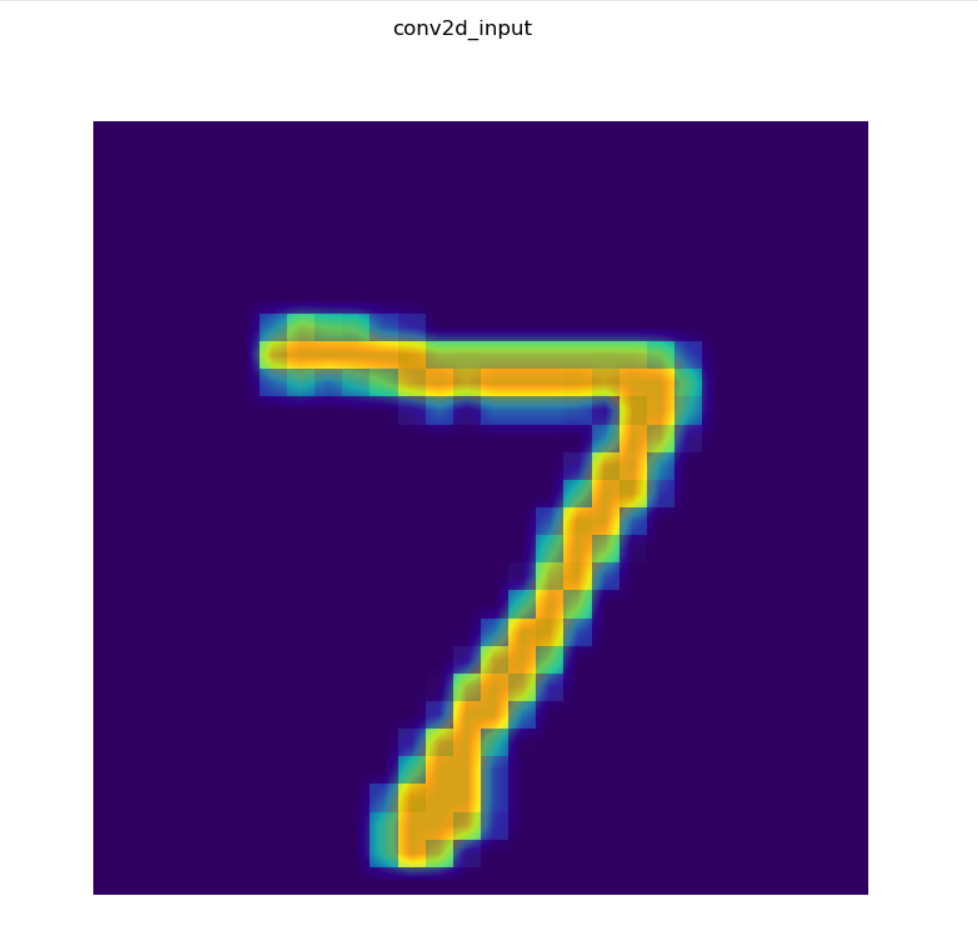


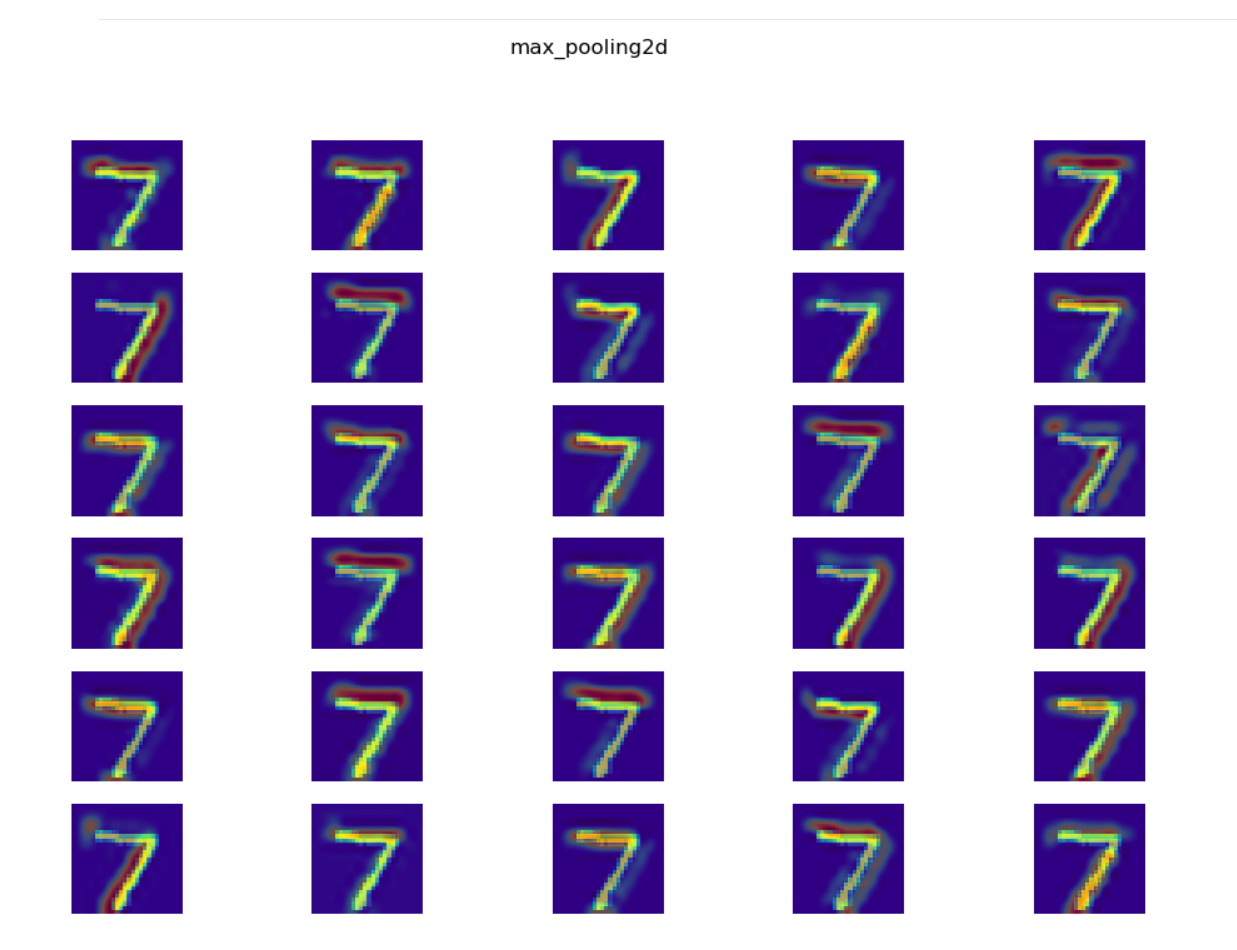


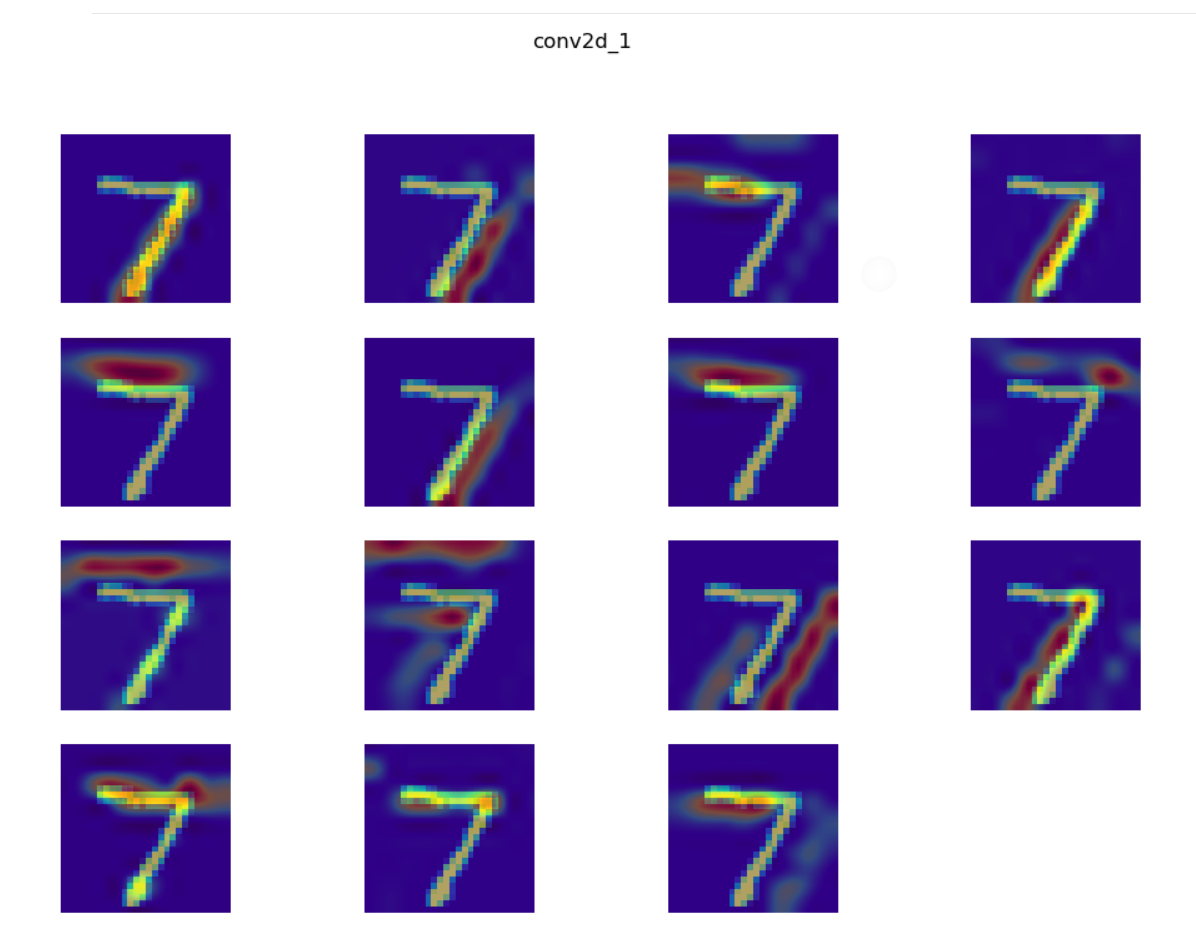
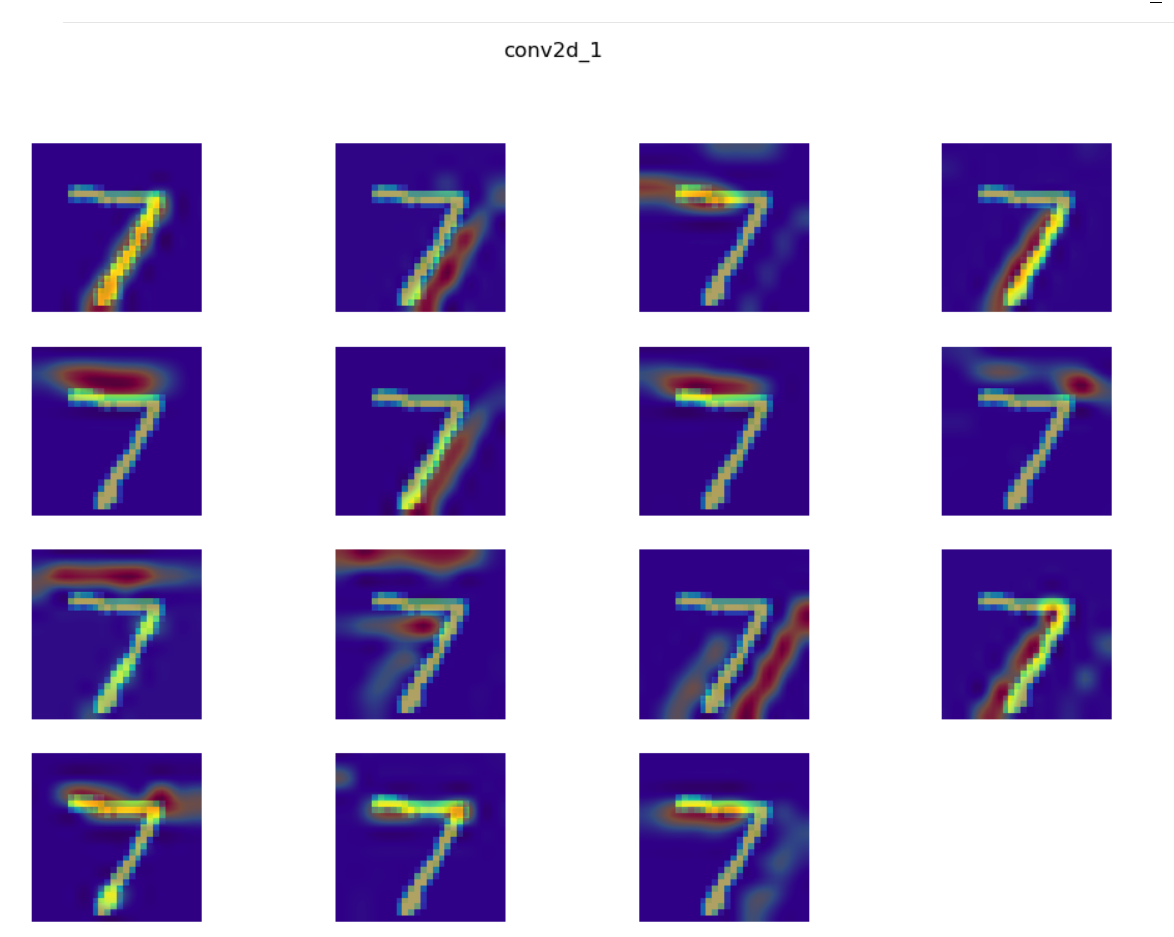


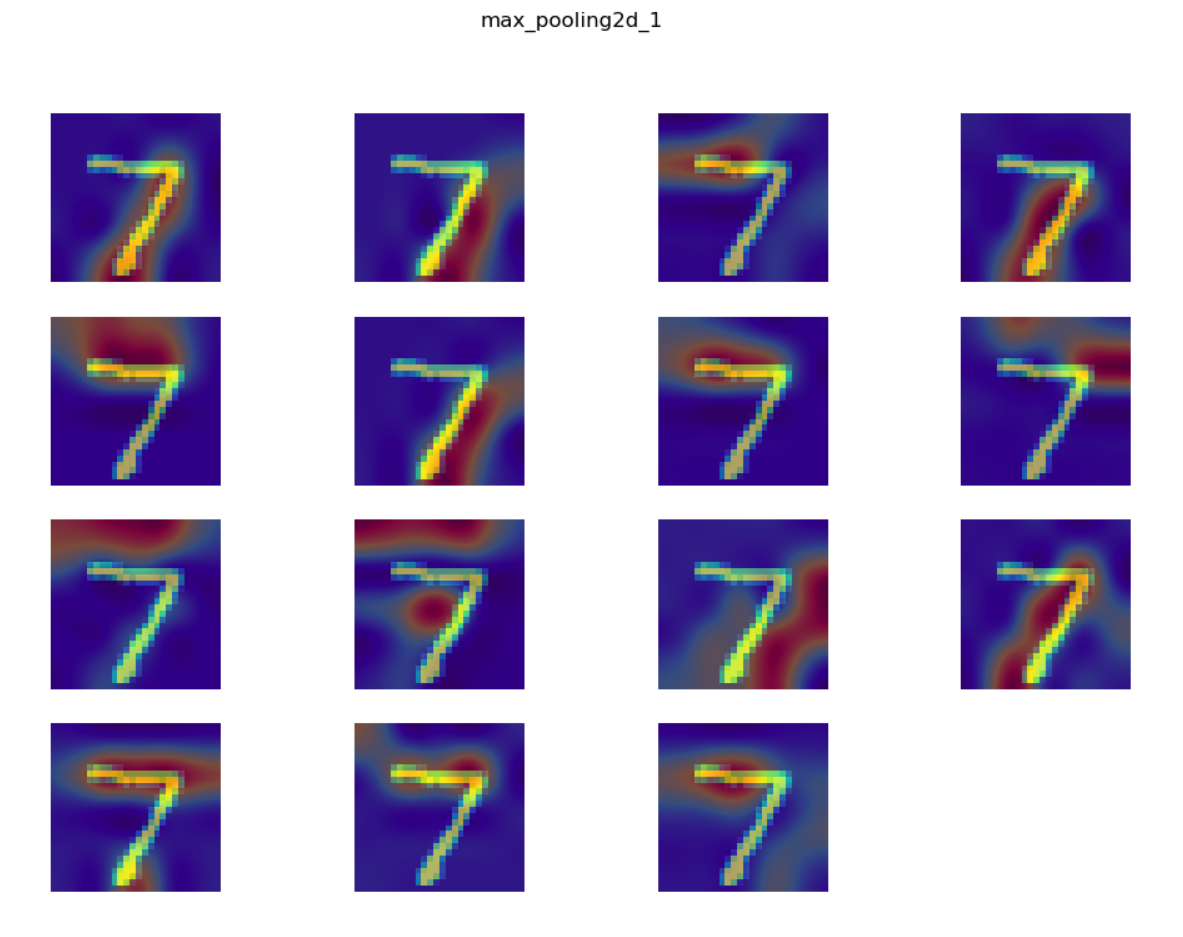


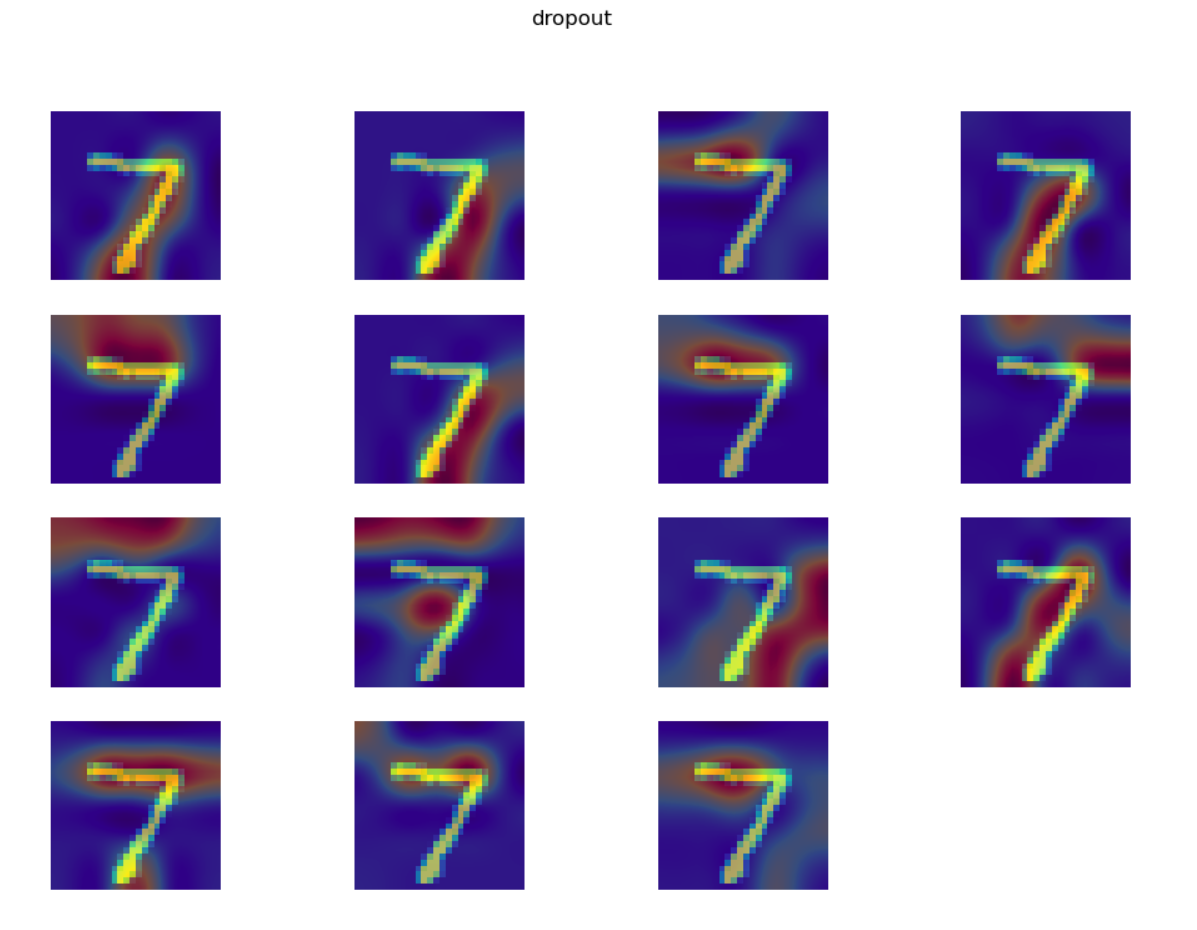












1. Try to visually see differences in the heatmap activations of a benign sample vs its adversarial example. Use the adversarial examples you generated last week. **What changes in the model’s detection? How does what it detects help determine the ultimate classification?**
2. Let’s formalize this difference a little more with some quantitative analysis:
   1. Group MNIST examples by label (numpy’s where() function may be useful, as in zero\_indices = np.where(labels == 0)[0]; for ex in dataset[zero\_indices]…)
      1. I created an array with as many rows as classes, and as many columns as neurons
      2. As I looped through the first 5000 images, I used determined the image’s classification with img\_class = int(np.where(keract\_targets == 1)[0][0])
   2. For each class, run all examples through Keract and record the 128 values of the Dense layer for each.
      1. They have been recorded in the active\_sum, where each row is a class and each column is a neuron
   3. Calculate the mean and standard deviation for each neuron of the Dense layer for each class, e.g. for neuron 1 of the Dense layer for class ‘4’, the mean activation value is 3.6 with a stdev of 1.2.
      1. Calculations have been completed and stores in active\_mean and active\_stdev arrays, where each row is a class and each column is a neuron
   4. What’s the average number of neuron activations of a normal sample that fall outside of one standard deviation from the mean? E.g. on average for all the benign training samples, each sample can expect to have 3 neurons out of 128 of the dense layer whose activation does not occur within 1 stdev of the mean.
      1. Let’s focus in on the first neuron:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | No. Obs outside one st dev | Total Obs of Class | Portion outside one st dev |
| 0 | 158 | 479 | 0.330 |
| 1 | 279 | 563 | 0.496 |
| 2 | 36 | 488 | 0.0738 |
| 3 | 16 | 493 | 0.0324 |
| 4 | 28 | 535 | 0.0523 |
| 5 | 7 | 434 | 0.0161 |
| 6 | 36 | 501 | 0.0719 |
| 7 | 198 | 551 | 0.359 |
| 8 | 17 | 462 | 0.0368 |
| 9 | 50 | 495 | 0.101 |

Therefore, for this particular neuron, with a labeled image of a 2, 3, 4, 5, or 6, the neuron experiences very consistent activation (less than 10% are more than a standard deviation above or below the mean activation value experienced from all 5000 images). Interestingly, this neuron was tested 5001 times, and only 16.5% of those activations were beyond a standard deviation away from the mean. For a normal distribution, approximately 32% of the values would be a standard deviation away from the mean, suggesting that heavily skewed values have increased the variance.

Of the 5001 benign images (with each image activating each of the 128 neurons), 14.8% were calculated as being beyond one standard deviation away from a neuron’s specific mean and standard deviation, which equates to 19 out of 128 neurons per image.

* 1. Now do this for your adversarial examples. Make at least 100 adversarial examples for MNIST (10 for each of the digits, any attack method is fine), and find the average number of neurons whose activation is outside of 1 standard dev of the mean, for BOTH the original class and the attacked class. In other words, if an image was originally ‘5’ but gets attacked to be classified as ‘3’, use both the original ‘5’ mean and stdev, plus the attacked class ‘3’ mean and stdev. Does the adversarial example have activations outside the norm more often for both classes? Just one? By how much? Does this number change with smaller perturbations (eg L2 norms?)
  2. Bonus: implement the MNIST model in the tutorial in 2.d above, and also calculate mean/stdev values for the Maxpooling layer in addition to the dense layer right before the 10 value logit dense layer at the end.