Sensory Coding Final Paper:

For my 607 Sensory Coding Project, I took a first stab at building a biologically inspired neural network modeling the visual system. As my primary interest is in the computation and perception of motion, I sought to design an architecture that mimics some of the properties of primate area MT.

The models were trained in two tasks: Judgment of object orientation, and judgment object rotation direction. To teach the networks these tasks, I created a synthetic dataset based on an imagined psychophysical task in which the subject would judge rotation direction and average orientation (in degrees) of a short animation of a rotating bar.

I had two specific aims this project: That my models would learn feature detectors analogous to direction-sensitive neurons in MT, and that I could devise a situation where the network learned features that mimicked the structure of cortical columns.

The project became a balancing act between obtaining good classification results and maintaining a relationship to the primate visual system. I began with a 3d CNN architecture that obtained good classification results, but over the course of the project decided that a 3d CNN abstracted the problem to a degree that the learned kernels didn’t represent biological receptive fields adequately. Considering that the retina captures 2-dimensional information, and all prior literature (that I’m aware of) treats receptive fields as 2-dimensional, I opted to switch to a 2d convolutional architecture and add an LSTM layer separately to keep track of differences between contiguous animation frames. My hope was that this would enable the network to learn kernels that resembled those found in MT+.

Unfortunately LSTM is very compute-hungry, and despite getting a working model, I didn’t have enough time to train and test it. However, I was able to port my approach to a much faster model, in which rather than using 3d convolution, I put 2d convolution layers inside a loop, concatenated and flattened the result for each 5-frame stack, then predicted on that output. I feel this simplified model architecture does a good job at representing the processing of 2d stimuli over time. Rather than abstracting the problem to a fully 3d computation, this approach more closely models 2d inputs flowing through the visual stream.

\*\*\*On Loss Functions\*\*\*

I crippled myself by writing a loss function that weighed the accuracy of two predictions equally. When I tried to train the model to predict both rotation direction and global orientation equally, the model was unable to learn. My thought had been that training on both tasks at once would produce receptive fields that learned to specialize in one task or another. At some point it occurred to me that if I were to bias the network to learn one task over the other, it might help. By opting to weigh the rotation classification more