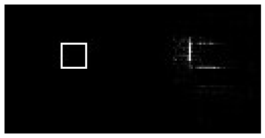
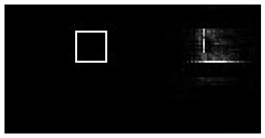
Sensorimotor Representations
Summary 19th September

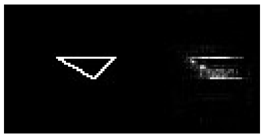
1. Implemented two Autoencoder architectures. In both cases the input and output were set to be a batch of images.

- The architecture of the first Autoencoder consisted of two conv layers with 20 kernels and 4 kernels respectively. Each conv layer was followed by a max pool layer which had a window size of 2. The output from the second max pool layer was flattened and fed into a fully connected layers with 64*64*2 units. The final layer also consisted of 64*64*2 units. The batch size had to be restricted to 2 due to memory issues even when run on the gpu's. The learning rate was set to decay exponentially and the momentum optimizer was used. Furthermore, the loss was defined in a mean square sense. Despite training for 20 Epochs, the Autoencoder did not perform satisfactorily and instead produced images as shown below. There was however, a drop in the loss by a factor of 3 during the course of training.

- The next Autoencoder architecture consisted of a fully connected layer of 64*64*16 units. The second fully connected layer downsampled the output of the previous layer. The result was then passed into a conv layer, a max pooling layer, another conv layer, a max pooling layer and finally a full connected layer. The kernels for the conv layers were 32 and 64 respectively. A fixed learning rate was used and the Adam Optimizer was used with the loss defined in a mean square sense. A sample of results is shown below after training for 20 Epochs.

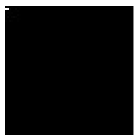






2. Generated controls (angles between links) for two arms with different link lengths. Both set of controls may be used to generate the same shape. Plan is to map controls of one arm onto those of the other. Model setup to this is in variable_seq2seq.py. Model architecture consists of an encoder lstm and decoder lstm. The encoder section of the model is variable length, i.e. it terminates after a certain

number of timesteps specified as a placeholder. Once, the encoder terminates the last state of the encoder is passed on to the decoder, each subsequent input to a decoder timestep is the output of the prior timestep. The output at each time step of the decoder is collected and fed into a fully connected layer. A mean square loss is calculated between the output of the decoder and output placeholder. As of yet this has not trained correctly. The loss fluctuates at each step but does not decrease even after 10 Epochs of training. The output from the model is fed into the two-link arm model and the output image is shown below.



At present I think this may be due to a bug in the code which is why I will be taking a closer look at it. Thanks!