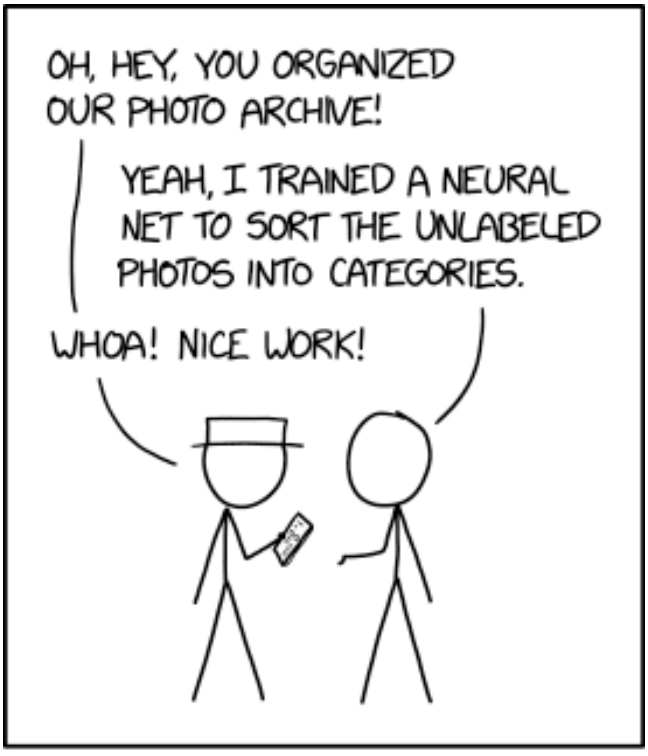
Initial Analysis of Perception and Decision Making Tasks

Amrut, Ankit

January 2021



Not saying I did that for these tasks

1 Overview

What we did:

1. Trained Recurrent Neural Networks for two different tasks using Gradient Descent. We call these tasks the "perception task" and the "decision task".

2. Analyzed the network weight matrices by simulating node placement, plotting the evolution of their principal components during training, and plotting changes in non-normality during training by using the Schur Decomposition.

3. Tested how well the network performed the task it was trained to do as well as how it reacts to inputs that it was not trained on.

4. Observed dynamics and subspace dimensionality when projected onto a latent space of maximal predictive information.

2 Network Training

The neural network used for the perception task contains 128 nodes, while that of the decision task contains 256 nodes. Both networks are fully connected.

Network activity approximates the continuous time equation,

$$\tau \frac{d\mathbf{r}}{dt} = -\mathbf{r} + f(\mathbf{W}_{rec}\mathbf{r} + \mathbf{W}_{in}\mathbf{u} + \mathbf{b} + \sqrt{2\tau\sigma^2}\mathbf{N}(0,1))$$

In our simulations, $\tau = 100$ms was the neuronal time constant. In addition, $\mathbf{u}$ is the input of the network, $\mathbf{b}$ is the bias of each node, $\sigma = .01$ is the strength of the noise applied to each node, and $\mathbf{N}(\mathbf{0},\mathbf{1})$ is a Gaussian white noise process with mean 0 and standard deviation 1. $\mathbf{W}_{in}$, $\mathbf{W}_{out}$ and $\mathbf{W}_{rec}$ are the input, output and internal weights respectively. f is the relu activation function and is defined as $f(x) = \max(0, x)$.

Each output of the network is a linear combination of the node activities and can be written as follows,

$$\mathbf{z} = \mathbf{W}_{out}\mathbf{r}$$

Using Tensorflow, we simulated the first-order Euler approximation of the equation above with a timestep of $\Delta t = 10$ms. The discretized equation was,

$$\mathbf{r}_t = (1-\alpha)\mathbf{r}_{t-1} + \alpha f(\mathbf{W}_{rec}\mathbf{r}_{t-1} + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b} + \sqrt{2\alpha^{-1}\sigma^2}\mathbf{N}(0,1))$$

Where $\alpha = \Delta t/\tau$.

Internal weights were randomly initialized using a Gaussian distribution with mean 0 and variance $1/\sqrt{N}$ where N is the number of nodes in the network. The diagonal of the internal weight matrix was set to 0 so there were no recurrent connections. Output weights were chosen from the same distribution.

Input weights and bias were similarly chosen by Gaussian distributions with standard deviations of $1/\sqrt{3}$ and $1/2$ for the perception and decision tasks respectively. Only the internal weight matrix was trained, but its diagonal values were maintained at 0.

The network was implemented using Tensorflow and trained using the Adam optimizer (gradient descent) with the default learning rate of 0.001. We used a l2 loss function over time for training.

3 Perception Task

3.1 Task Description

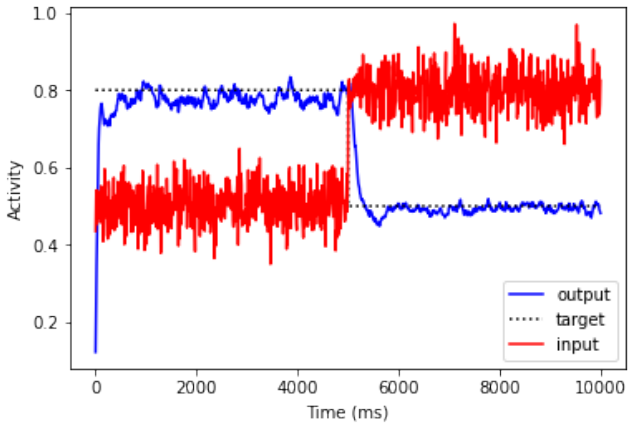


Figure 1: Example of network input, output and expected target output.

This task is the most basic one we could think of and consists of training the network to react in a predictable way to a single given input. More specifically, we trained the network to converge to a steady-state output of 0.5 for an input of 0.8, and an output of 0.8 for an input of 0.5. The input had added Gaussian noise with a standard deviation of 0.05 and mean of 0.

3.2 Weight Matrix Analysis

3.2.1 Visualizing Network Structure

In Figure 2, we observe a change in the structure of the network during training. In particular, we observe the formation of densely connected and more sparse regions in the weight matrix.

In addition to the increased structure seen after training when visualizing the graph structure of the network, Figure 3 shows that the spread of the weights stays centered around 0 but narrows during the training process.

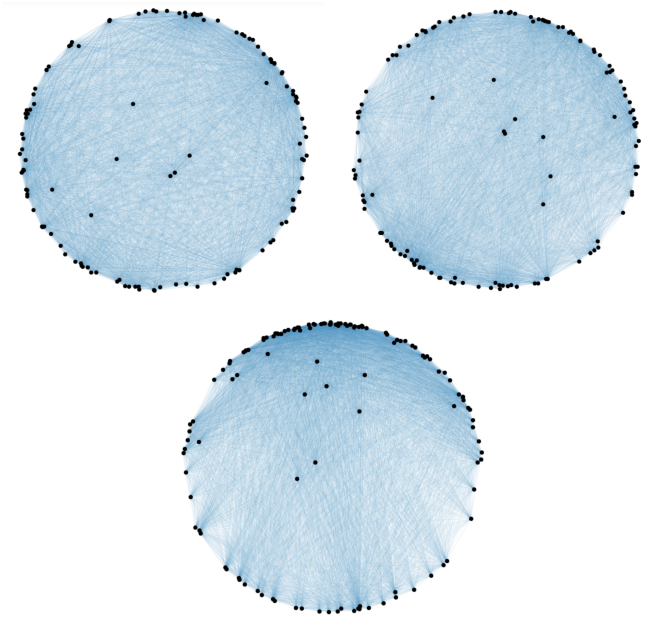


Figure 2: Diagrams of fully connected recurrent neural networks with 128 nodes where the edges are shaded by the strength of the weight and the positions of the nodes are determined using the Fruchterman-Reingold force-directed algorithm. (Top Left) Initial network with randomly intialized weights. (Top Right) Network midway through training, we can begin to see clusters of nodes along the circumference with more densely shaded regions indicating higher connectivity. (Bottom) Network structure post-training. The nodes tend to cluster into a large densely connected region with a few smaller similar highly connected nodes/clusters spread around the circumference.

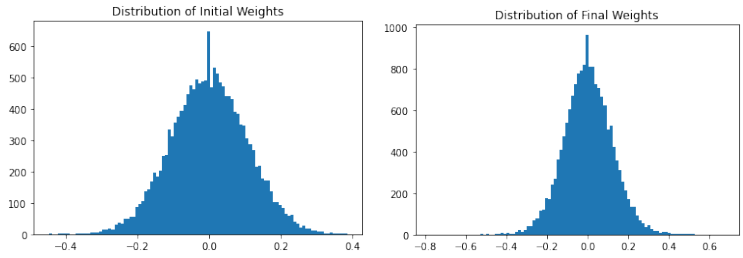


Figure 3: (Left) The initial, untrained distribution of weights in the network. (Right) The distribution of weights in the network after training.

3.2.2 Principal Components Analysis

Almost all principal components of the weight matrix increased during training up to around component 100. Therefore, instead of looking at individual singular values, is more interesting to look at change in the proportion of variance in the weight matrix explained by each component during the training process as seen in Figure 4.

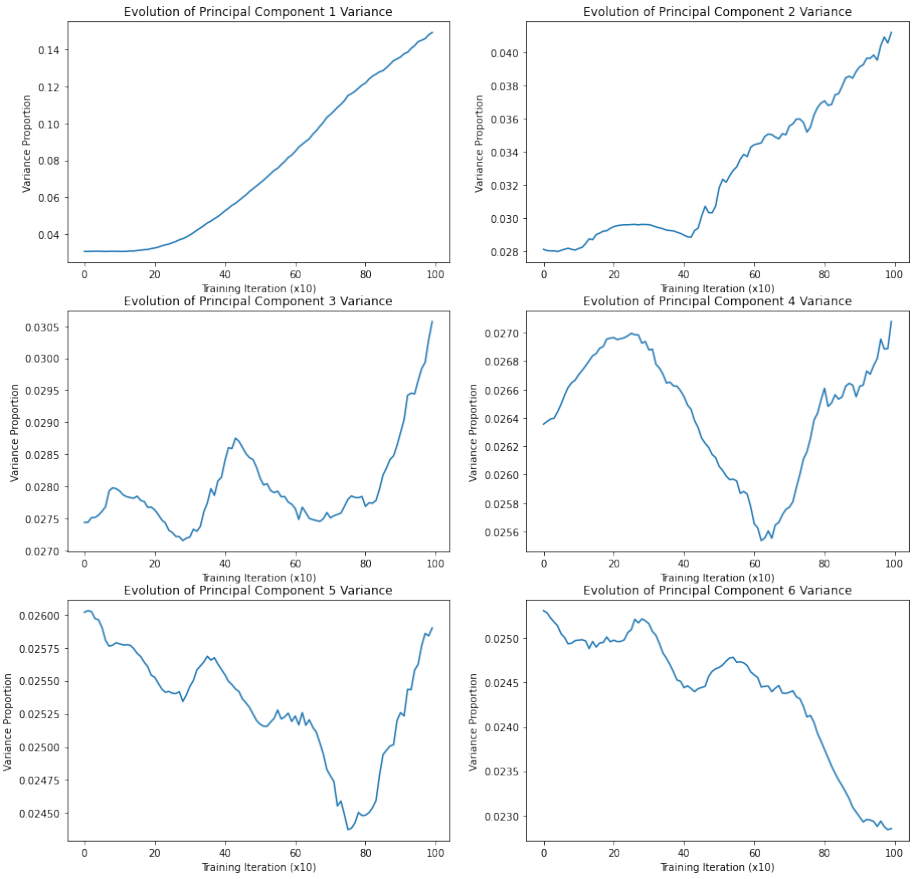


Figure 4: Change in proportion variance explained by each principal component over the course of training. The weight matrices were saved every 10 iterations. The first 5 principal components increase in explained variance during training, but components 6 and higher show a more steady decrease.

The evolution of the variance explained by the first five principal components generally starts with a flat or even downwards trend before a large increase in value. Starting from principal component 6 and higher, there is only a downward trend. It stood out to me that the large increases for each of the principal components tended to happen "in order". Meaning, component 1 began to

increase before component 2 which began to increase before component 3 and so on. I didn't plot the summed evolution of the first few components during training because that curve would largely follow the curve of component 1 which is overwhelmingly large compared to the others.

3.2.3 Schur Decomposition

Here, We plotted the change in the 2-norm of the triangular matrix in the schur decomposition. We see a very constant increase in this value during training which also implies an increase in the non-normality of the weight matrix. This is a very interesting result because it is also repeated for the other task.

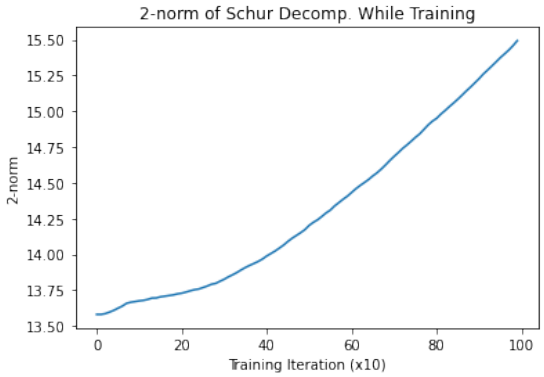


Figure 5: The 2-norm of the triangular matrix in the schur decomposition increases throughout training.

3.3 Network Functionality

3.3.1 Network Output

Figure 6 shows an interesting asymmetry in the two attractors near our desired input/output values. These "attractors" can be identified as the flat regions in the plots. We call these regions attractors because the output remains constant for varying input. The attractor for an input of 0.8 and output of 0.5 is concave up while the other is a saddle point.

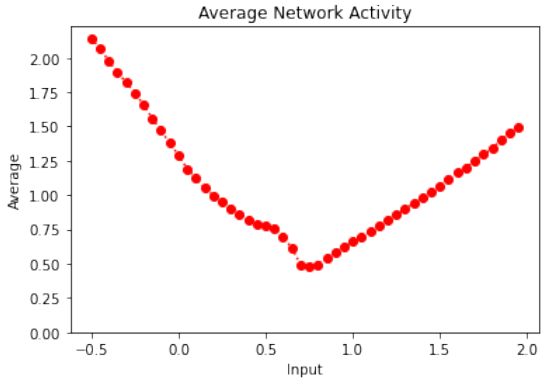


Figure 6: This plot shows the average network output activity for various inputs. We can see attractors (the flat regions) near the input training values of 0.5 and 0.8. However, there is an asymmetry in the concavity of both attractors.

3.4 DCA

We used Dynamical Components Analysis to project network activity for different inputs into low-dimensional subspaces. It is interesting to look at the latent space trajectories, angles between the subspaces and the change in predictive information with respect to dimensionality.

3.4.1 Latent Space Trajectories

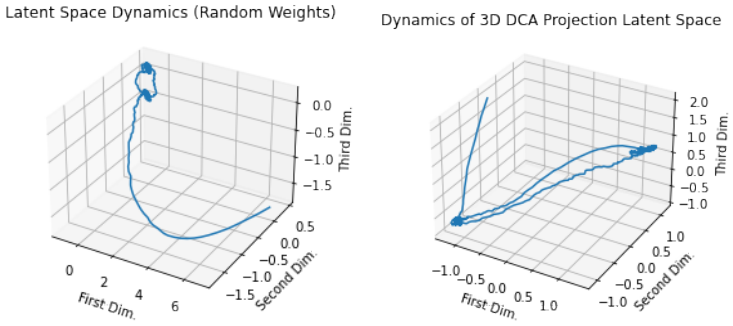


Figure 7: (Left) DCA Projection onto latent space with maximally predictive information for an untrained network with random weights and input values switching between 0.5 and 0.8. (Right) DCA Projection onto latent space with maximally predictive information for a trained network with random weights and input values switching between 0.5 and 0.8.

Figure 7 shows an interesting result when comparing the latent space dy-

namics of the network pre vs post training. In both cases, the input causes the dynamics of the networks to start at 0, then transition to a steady state equilibrium for an input of 0.5, then transition to a steady state for an input of 0.8 and back to 0.5 again. In both cases, we see both steady-state values reflected in the 'blobs' in the dynamics while the transitions are reflected in the curves between the 'blobs'. The initial transient activity is the curve coming in from outside.

The training process caused the dynamics between the two 0.5 and 0.8 outputs to take up quite a but more space in the final latent space when compared to that of the network with randomly initialized weights. The size of the transient curve from 0 can be used as a 'ruler' for the dynamics. This is interesting because it reflects that the network is more precisely tuned to values between 0.5 and 0.8. In addition, in the subspace with the randomly initialized weights, we see that the initial transient curve from 0 is nearly in the same 2D plane as the trajectory between the 0.5 and 0.8 inputs. However, when the weights are trained, the transient dynamics is orthogonal to the dynamics between the two steady state values. This orthogonality also reflects a more precise tuning of the trained network to values within the output range.

As a side, it is also interesting how the added noise is mainly visible during the steady-state activity and not during the transitory periods.

3.4.2 Subspace Angles

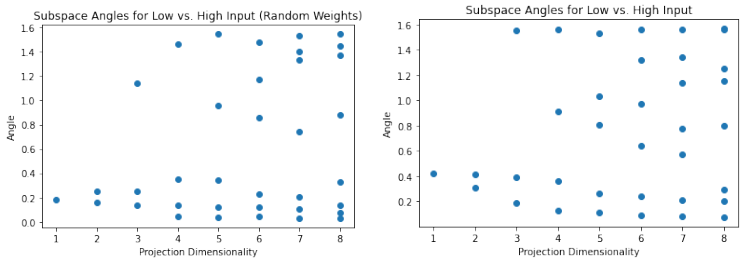


Figure 8: A scatter of the subspace angles between the n-dimensional DCA projections for an input of 0.8 and an input of 0.5. Two n dimensional projections have n subspace angles between them. (Left) Subspace angles for randomly initialized weights. (Right) Subspace angles for trained weights.

Figure 8 provides a comparison of the subspace angles between maximally predictive latent spaces of the low input dynamics and high input dynamics for a trained network and untrained, randomly initialized network. We can conclude that the angles for the trained network are more structured than that of the untrained network. In addition, in the trained network, orthogonality between the subspaces begins at dimension 3. The subspace angles for the untrained weights have a similar structure where orthogonality begins at dimension 3, but in a less dramatic way. For the first two projection dimensions, subspace angles

are low in both cases which implies the low input and high input dynamics occupy roughly the same latent space.

3.4.3 Predictive Information

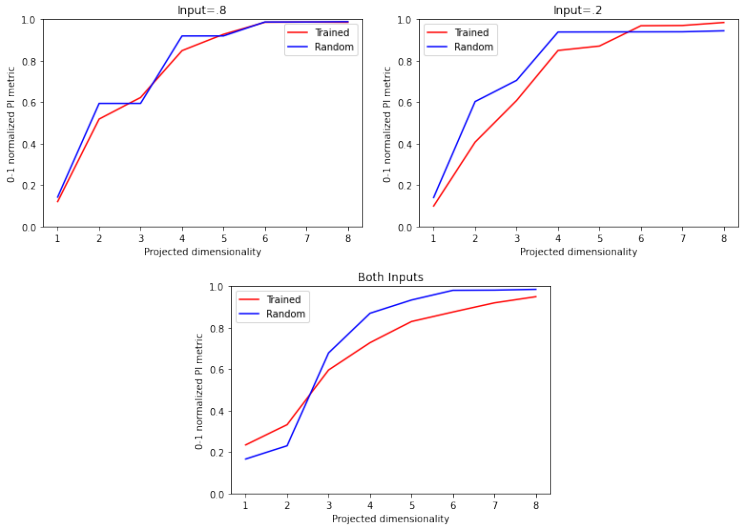


Figure 9: Predictive Information versus latent space dimensionality for different inputs. Comparing the PI for a trained network and an untrained network.

The PI curves look fairly similar for the trained and the untrained networks for the high and low inputs in Figure 9. However, when looking at dynamics where both inputs were present, note that the untrained network starts off with a lower predictive information but ends up with a higher PI.

In addition, there is low dimensionality both before and after training the network. Since a stable, linear, randomly initialized network would have a linear PI curve that does not saturate, the low dimensionality of the untrained network is likely due to the RELU nonlinearity.

4 Decision Task

4.1 Task Description

For the Decision task, a fully connected network with 256 nodes was given two inputs between 0 and 2, and was asked for one output. The network would output 0.5 if input1 was greater than input2 and 0.8 otherwise. Input noise standard deviation was 0.01 for this task.

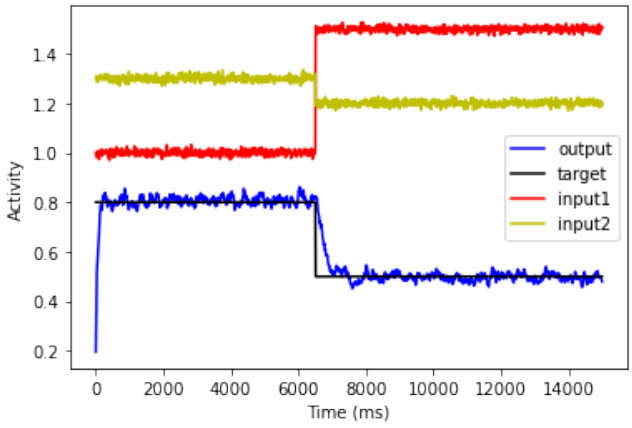


Figure 10: Example of network inputs, output and expected target output.

4.2 Weight Matrix Analysis

4.2.1 Visualizing Network Structure

Similar to the perception task, Figure 11 shows that the spread of the weights for the decision task remains centered around 0 after training but does not narrow very much.

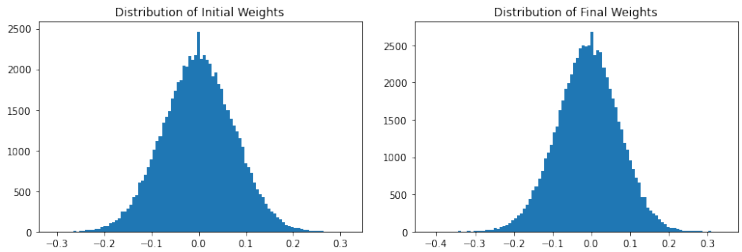


Figure 11: (Left) The initial, untrained distribution of weights in the network. (Right) The distribution of weights in the network after training.

In Figure 12, we observe a change in the structure of the network during training. In particular, the nodes are clustering around the circumference of the graph.

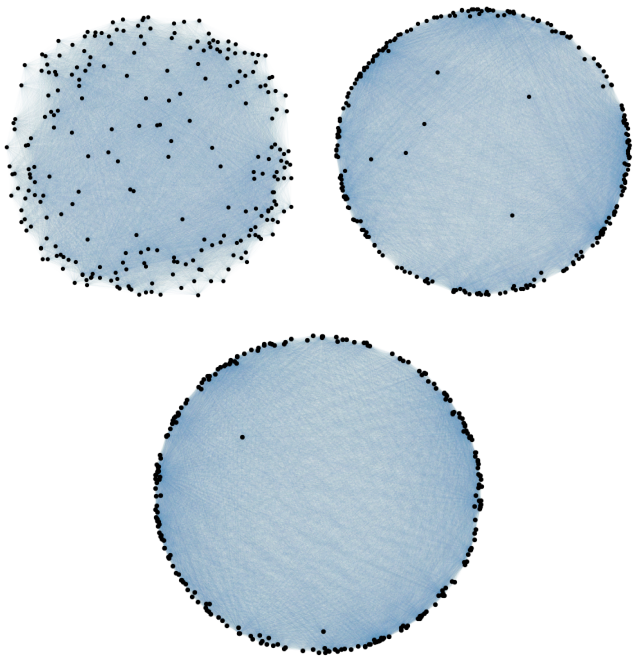


Figure 12: Diagrams of fully connected recurrent neural networks with 256 nodes where the edges are shaded by the strength of the weight and the positions of the nodes are determined using the Fruchterman-Reingold force-directed algorithm. (Top Left) Initial network with randomly intialized weights. (Top Right) Network after 10 iterations of training, there is significant structure forming where the nodes are moving to the outside of the graph. (Bottom) Network structure post-training. The nodes are mostly around the circumference of the graph and form strongly connected clusters with weaker connections between each other.

4.2.2 Principal Components Analysis

Almost all principal components increased during training up to around component 100. Like before, instead of looking at individual singular values, is more interesting to look at change in the proportion of variance explained by each component during the training process as seen in Figure 13.

We see that the variance proportion explained tends to increase with training up to dimension 6. Therefore, the weight matrix for this task can be considered "higher-dimensional" than that of the previous task. This can be explained by the intuition that the decision making task is "more complex" than our first perception task. The variance begins to decrease for principal component 8 and higher.

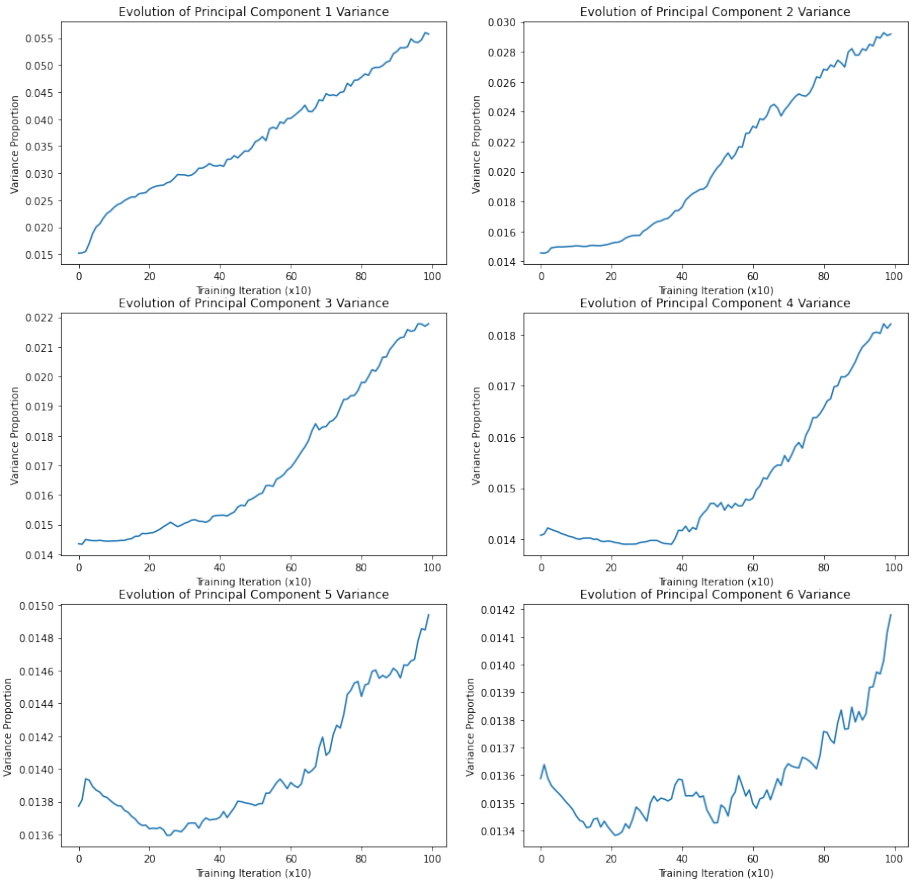


Figure 13: Change in proportion variance explained by each principal component over the course of training. The weight matrices were saved every 10 iterations. The first 6 principal components increase in explained variance during training, but with increased fluctuation

4.2.3 Schur Decomposition

Figure 14: The 2-norm of the triangular matrix in the schur decomposition increases throughout training.

We plotted the change in the 2-norm of the triangular matrix in the schur decomposition in Figure 14. We see a very constant increase in this value during training which also implies an increase in the non-normality of the weight matrix. This is the same trend seen in the Schur Decomposition of the perception task

4.3 Network Functionality

4.3.1 Network Output

Figure 15 provides a visualization for when the network makes mistakes. We see that the network is biased towards guessing that input2 is larger than input1 if it is unsure of the actual result (assuming initial node activations are 0). Therefore, it is able to correctly identify when input2 is larger with perfect accuracy. However, it makes mistakes when input1 is larger than input2. The network requires a large difference (about 0.15) between input1 and input2 in order to correctly classify small input values. However, the network becomes significantly better at discriminating between the values for larger inputs.

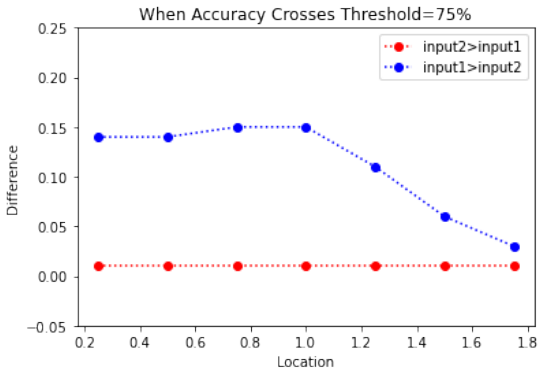


Figure 15: Shows the difference between the two values required for the network to correctly classify which input is greater than 75% of the time. The x-axis is the value of input1 during the discrimination task while the y-axis is the difference between input1 and input2. The y-axis has a resolution of 0.01.

4.4 DCA

4.4.1 Latent Space Trajectories

In the trajectories where the inputs switched between two different values, there was similar latent space equilibrium as in the previous task. I found it most illuminating to have the network classify results throughout the entire space of possible input values with a resolution of 0.05 and then plot the input1 and input2 isocontours in the latent space dynamics which can be seen in Figure 16. The isocontours for inputs 1 and 2 were plotted in different latent spaces. These plots gave me a structural picture of 'where', relative to each other, each computation was taking place.

We also plotted the input1 isocontours for dynamics with the untrained random weight matrix. We did not plot the input2 isocontours for the random weights because the network was untrained so there would be no difference between the two.

It is interesting to note that the input1 and input2 isocontours for the trained network have a layered structure, where the layer of the contour most depends on which input was greater than the other. This structure is not seen in the dynamics of the untrained network.

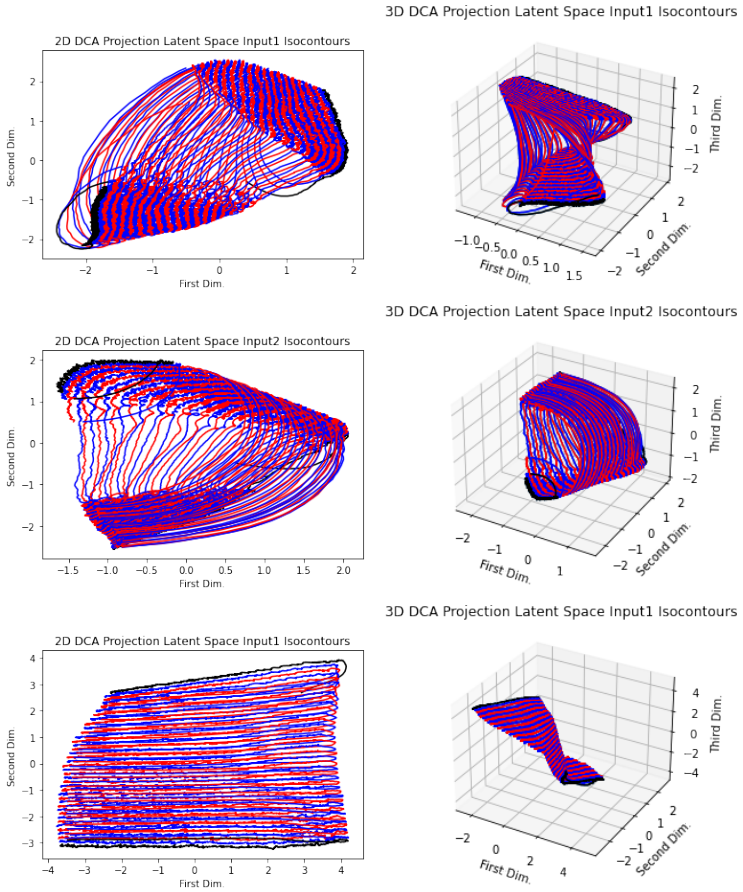


Figure 16: The alternating blue and red lines reflect the isocontours where one input was changing and the other was held fixed. The black lines represent the isocontours where the fixed input takes on its smallest or largest value. Note that there is some variability when the projection was repeated, but the major features are always there. (Top Row) The Input1 isocontours for the trained network. (Middle Row) The input2 isocontours for the trained network. (Third Row) Input1 isocontours for the initial, untrained network with random weights.

4.4.2 Predictive Information

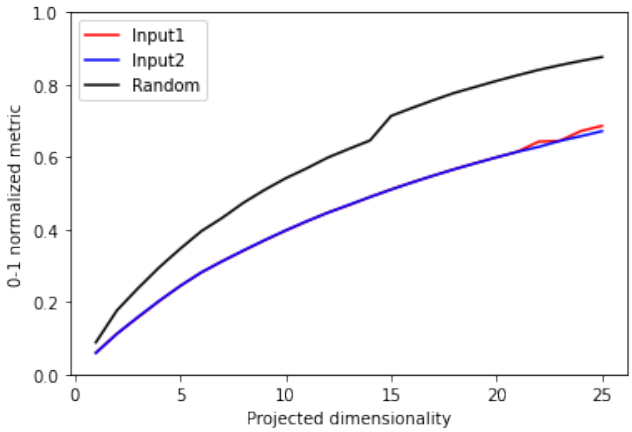


Figure 17: Predictive Information versus dimensionality for the three dynamical conditions plotted in Figure 16. The red and blue curves represent the predictive information when simulating the input1 and input2 isocontours respectively while the black curve is the input1 isocontours for the untrained weights. Note that the projection dimensionality goes up to 25 which is larger than that of the previous task. This is because this simulation had far more varied input than before.

As seen in Figure 17, the predictive information of the untrained network increases more rapidly with dimensionality than that of the trained network. This is an interesting result and somewhat unexpected.

5 Conclusion

1. We do see evidence of the network optimizing itself for computations by looking at its latent space dynamics. Namely, the layered structure in Figure 16 and the orthogonality in Figure 7.

2. We also see structure begin to form in the networks during training as seen in Figures 2, 8, 12.

3. The non-normality of weight matrices trained for both tasks increased with training which is shown in Figures 14, 5

4. During training, the first 1-2 principal components of the weight matrix are most important and increase rapidly but higher principal components also increase later on when there is more fine tuning going on. This is visualized in Figures 4, 13.

5. In Figure 17, it is apparent that predictive information increases more rapidly for the untrained weight matrix than the trained weight matrix for the decision task. This is not as apparent, but can still be seen for the PI curve for the perception task in Figure 9.

6. The networks have asymmetry, biases and make mistakes when learning tasks (Figures 15, 6)