Decision Making and the Brain

Ciprian Bangu

May 20 2024

Contents

1 Introduction

One of the most salient cognitive funcitons is that of decision making. A popular paradigm for studying decision making in the brain is the 2 Alternative Forced Choice (2AFC) Task. In this task, a subject is presented with a stimuli of moving dots where some proportion are moving randomly, while some other portion are moving, with various levels of coherence, either left or right, depending on the trial. The subject is then asked to decide whether the dots were moving to the left or right by making a saccade in the chosen direction.

It is hypothesized that when presented with this task, the brain comes to a decision in the follwing way: when presented with a stimulus, neurons in the Medio-Temporal visual cortex (MT) fire in response to stimuli in their preferred direction. The firing of these neurons is then integrated over time by neurons in the Lateral Intraparietal area (LIP). The result of this integration is then used as the signal to make the decision in the Frontal eye field, which is responsible for the saccade.

In this report, we will look at two models that aim to capture this process: the Drift Diffusion Model and a Recurrent Neural Network. We will examine the dynamics of these models and how they can be used to approximate decision making in the 2AFC task.

2 The Drift Diffusion Model

The first model we will consider is the Drift Diffusion Model (DDM). This model is given by the following equation:

$$\frac{dx(t)}{dt} = (I_A - I_B) + \sigma\eta(t) \tag{1}$$

where x is a decision variable, I_A and I_B are the neural signals to make decision A and B respectively, σ is the noise magnitude, and $\eta(t)$ is a Gaussian white noise process with zero mean and unit variance. The decision of the subject, either A or B, is represented by the crossing of the decision variable of some threshold. This threshold is μ in the case the subject decides for outcome A, or $-\mu$ in the case the subject decides for outcome B. The difference $I_A - I_B$ can be thought of as the evidence, E, the subject has for deciding in eithe direction. If this difference is positive, the subject is more likely to decide in favor of A, and if it is negative, the subject is more likely to decide in favor of B. Finally, the noise term is meant to capture the noise in the stimulus.

The aim of this model is to capture the process of the integration by neurons in the LIP of signals of motion direction in either the A or B direction as reported by direction sensitive neurons in the MT.

2.1 Simulating the DDM

To get a sense of the dynamics of the stochasitc process produced by the model, we can simulate the model and plot the decision variable x as a function of time. Figure 1 below illustrates 10 trials of the model, given the initial conditions: $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, with possible timesteps = 10000.

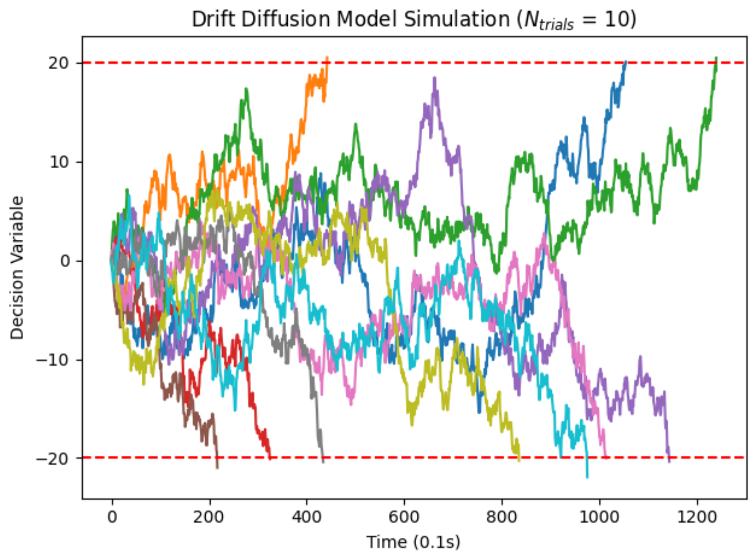

Figure 1: Simulation of the Drift Diffusion Model with $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000. Different trials are indicated by different colors. The dashed red lines indicate the decision variable thresholds.

As we can see from figure 1, the decision variable evolves randomly over time due to the gaussian noise in the model. Moreover, while it does not evolve in the same way every time, the decision variable eventually meets or crosses one of the thresholds, ending the trial, for every trial.

Recall that we introduced the notion of evidence E as the difference between the signals I_A and I_B. In Figure 1, we see that there are 3 trials where the decision variable crosses the threshold for A, and 7 trials where the decision variable crosses the threshold for B. This is consistent with the fact that $I_B > I_A$ in our simulation parameters, meaning the model gets more 'signal' from the B direction. We can see the impact of this difference in evidence by observing the number of trials that end in favor of one decision over the other, over more simulations of the model.

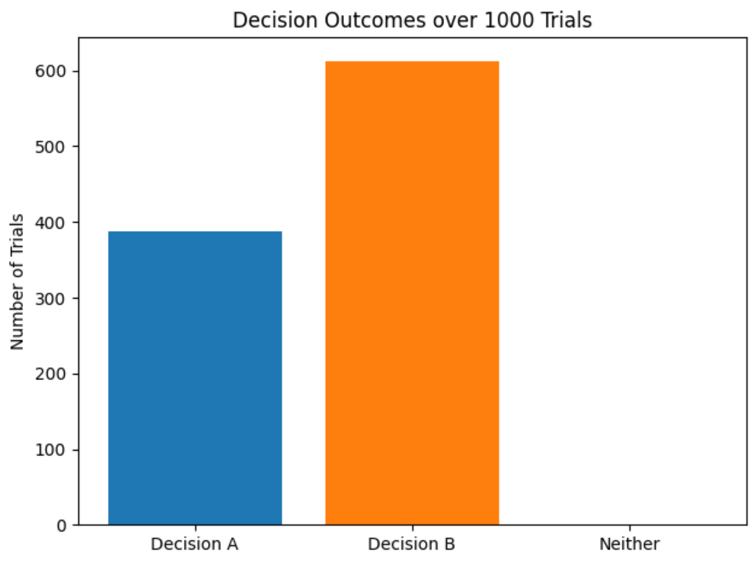

Figure 2: Outcomes of the Drift diffusion model over 1000 simulations. $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$. The number of trials that end in favor of decision A is plotted in blue, and the number of trials that end in favor of decision B is plotted in orange.

Figure 2 above illustrates the proportion of outcomes for each decision after 1000 simulations of the model. From Fig. 2, we can see that the split between decisions A and B is about 60:40 in favor of decision B. Thus, even a small difference in the evidence in favor of one decision over another can lead to a significant difference in the number of trials that end in favor of one decision over the other. In this case a 0.05 evidence advantage for B led to a 50% increase in the number of trials that ended in favor of B over A.

2.2 Varying other parameters

We have seen how a small differential in the evidence can impact the distribution of outcomes for the model. However, there are more parameters we can vary to get differential behavior. In this section, we will vary μ, σ and E to observe how the distribution of outcomes changes. Something to note here is that this collection of parameters contains both subject internal and external parameters. Namely, while μ and E are subject internal parameters, i.e., they are provided by the subject, σ is an external parameter, i.e., it would be provided by the experimenter in their varying of the motion coherence.

2.2.1 Varying μ

First, let us vary the threshold parameter of the model, μ. In Figure 3 below, we keep the same intial conditions as before, but vary μ from 1 to 100. Given the small amount of simulations, the difference in the number of trials that end in favor of B rather than A is initially small, and in some cases negative. However, as μ increases, there emerges a clear advantage for decision B over A. This is because the evidence imablance combined with the higher threshold makes it less likely that x will reach or cross μ on any given trial. From the figure, we see that this decrease in A decisions is linear as μ increases. However, the relationship between increases in μ (or, rather, increases in the *magnitude* of μ, since it is negative for decision B) is parabolic. Initially, increases in μ have a positive imapct on the number of trials that end in favor of B. However, this effect flattens out around $\mu = 50$, after which the number of trials that end in favor of B begins to decrease (although it remains higher than the number in favor of A). This is because the model is less likely to cross any threshold as μ increases, since it only has a finite amount of time to reach μ, which will take longer as μ increases.

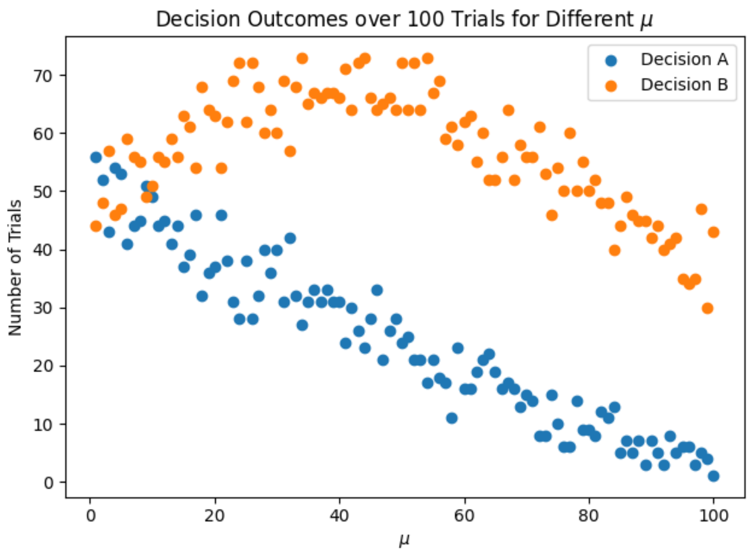

Figure 3: Number of trials that end in favor of decision A and B over 100 simulations of the DDM, varying μ from 1 to 100. $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000.

2.2.2 Varying σ

Now, let us vary σ, or the magnitude of the noise in the model. In Figure 4 below, we keep the same initial conditions as before, but vary σ from 0 to 50, simulating the model for 100 trials.

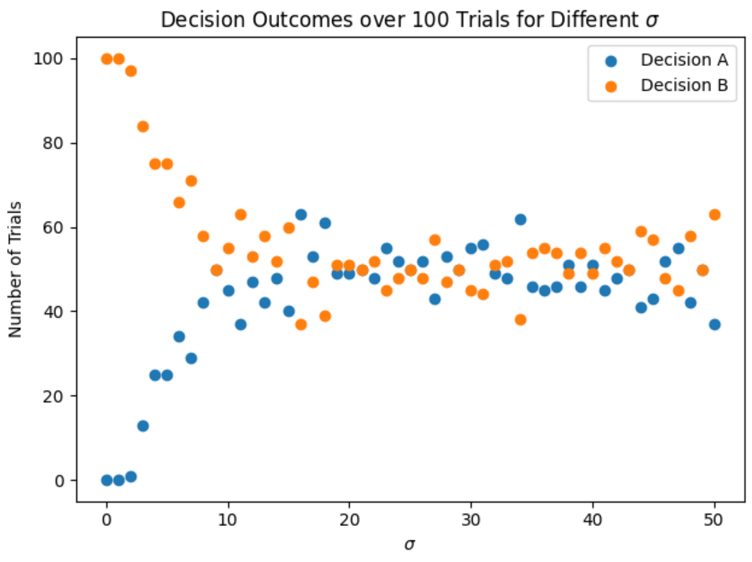

Figure 4: Decision outcomes over 100 simulations of the ddm for varying values of σ. $I_A = 0.95$, $I_B = 1$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000. σ varies from 0 to 50.

In Figure 4 above, we can observe the effect the magnitude of the Gaussian noise has on the subject's decision outcomes. Recall that, given our intial conditions, there is a slight evidential advantage for Decision B. When the $\sigma = 0$, this advantage is fully borne out, in that the model always decides in favor of B. However, as the noise level increases, the number of trials ending in B decreases, and the number of trials ending in A increases. As σ approaches 50, the number of trials for each decision becomes about equal. This is because for high levels of σ, the noise term in the model dominates the signal term, so x's crossing of a threshold is governed by chance rather than evidence. Essentially, the subject can no longer decide based on the evidence so it decides randomly.

2.2.3 Varying E

Finally, let us vary the evidence E in the model. Previously, we noted how an imbalance in E lead to a slight preference for one outcome over another. In Figure 5 below, we keep the same initial conditions as before, but vary E from -0.5 to 0.5. This larger magnitude of values allows us to see this effect more clearly.

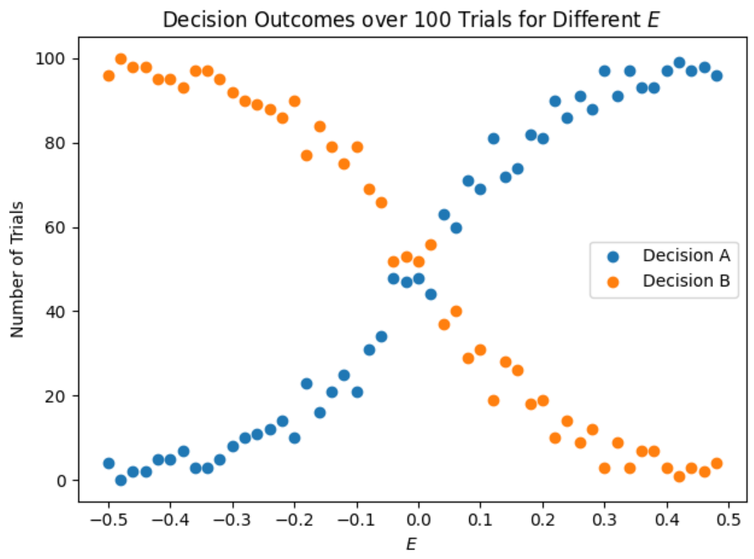

Figure 5: Decision outcomes over 100 simulations of the DDM for varying values of E. $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000. E varies from -0.5 to 0.5.

From Figure 5, we can see that as we vary E, the decision outcomes follow a sigmoidal pattern, and are symmetrical around $E = 0$. When E is negative and large, the model nearly always decides in favor of B. As E goes to 0 from the left, the model begins to decide for A more often, eventually becoming random. Conversely, when E is positive and large, the model nearly always decides in favor of A. Similarly, as E approaches 0 from the right, the model begins to decide for B more often, eventually becoming random. The more compelling the evidence for one decision over another, i.e., the stronger(weaker) the firing of MT neurons encoding for one direction, the more(less) likely the model is to decide in favor of that decision.

2.3 Reaction Time Distributions

In the previous sections, we examined the distribution of the decisions made by the model. In this section, we will examine the distribution of the reaction times of the model, i.e., how long it takes the model to make a decision given some initial conditions. Specifically, we will investigate how the distribution of reaction times changes as we vary E in the model.

Keeping the initial conditions the same as before, Figure 6 below illustrates the differences in reaction time distributions. From Fig. 6 we can see that the reacion times are normally distributed with a rightward skew, indicating that for each condition, most of the trials end before 100 seconds. Moreover, we can see that when the evidence is equal, or just slightly greater, for either option, the distributions overlap: it takes the model about the same amount of time to make a decision either way. However, when the evidence is skewed towards one outcome, in this case A, a greater proportion of trials where A is the outcome end quickly than those where B is the outcome. The area of the B distribution is also smaller in this case, indicating less trials end in favor of B when the evidence is skewed towards A. This is because $\frac{dx(t)}{dt}$ is larger and positive when E is larger and positive. Thus, the pressure towards the A threshold is greater, leading to quick decisions in favor of A, and slower decisions in favor of B: it will only go towards the B threshold if it is forced to by the noise term.

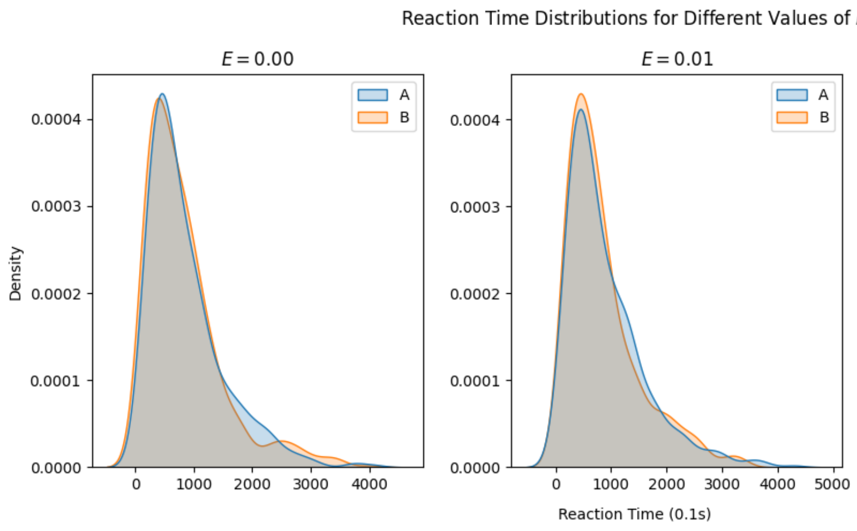

Figure 6: Reaction time distributions for the DDM for varying values of E. $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000. E varies from 0.00 to 0.05.

2.4 Summary

In this section, we have examined the Drift Diffusion Model as a way of modeling the decision making process undergone by a subject's brain in a 2AFC task. We have seen how the model can be used to approximate the integration of informtation over by time by different brain regions, and lead to decisions in favor of one outcome over another based on both subject-internal parameters (μ and E), and external parameters (σ). Moreover, we have seen how variation in these parameters can lead to different patterns of decision making for subjects, both in terms of number of A vs B decisions, and the amount of time taken to make these decisions.

In the next section, we will look at the same task through the lens of a different model of decision making, namely a Recurrent Neural Network.

3 The Reccurent Neural Network Model

In the previous section, we examined the Drift Diffusion Model as a way of modeling the neural decision making process in a 2AFC task. However, the DDM we described did not actually *perform* the task in any meaningful sense. In this section, we will train a Recurrent Neural Network (RNN) to perform the 2AFC task itself, and then analyze its performance and dynamics.

3.1 The Representing the 2AFC Task

In order to train the network, we first need to simulate the task itself. We will represent the task stimulus as a vector $\mathbf{u}$ of length T, where T is the length of the trial, and each element of $\mathbf{u}$ is a scaler u_t representing the type of stimulus: positive for motion rightward, or negative for motion leftward; 0 if there is no motion. u_t's magnitude represents the *coherence* of the stimulus. Formally, u_t is defined as:

$$u_t = v + \xi_t \; \mathcal{N}(0, \sigma) \tag{2}$$

where v is velocity of the stimulus (coherence and direction), and ξ_t is some gaussian noise with mean 0 and standard deviation σ, representing the noise in the stimulus.

We will also define an intended output, which represents the true direction of the stiumulus, and is the value we want our RNN to be able to predict. It is defined as a vector $\mathbf{z}^*$ of length T, where:

$$\begin{cases} \mathbf{z}^*{}_x = 0 & x \in [0, T-1] \\ \mathbf{z}^*{}_T = \text{sign}(\bar{\mathbf{u}}) \end{cases} \tag{3}$$

where $\bar{\mathbf{u}}$ is the average velocity of the stimulus over the trial. Thus, the network will need learn to output 1 when the average velocity is positive, and -1 when the average velocity is negative.

To train our network, we need to provide it a mask to indicate we only want it to learn the final decision. This is defined as a vector $\mathbf{m}$ such that:

$$\mathbf{m} = \begin{cases} 0 & x \in [0, T-1] \\ 1 & x = T \end{cases} \tag{4}$$

Finally, we need to create a training set for the network, i.e., a collection of simulated stimuli, intended outputs, and mask values that we will train the model on, as well as a test set, i.e., a seperate set of stimuli on which to test the model's performance. We will do this by perfroming the above mentioned steps for 500 simulated trials. The simulations for the training set will randomly choose an initial v between (integers) -5, 5 (exlcuding 0), and a random level of noise $\sigma \in \{0, 0.1, 0.05\}$. The test set will have a random initial v between -5, 5 (excluding 0), and a random level of noise $\sigma \in \{0.5, 1\}$.

Figure 7 below illustrates a sample of the input vectors in the training set and, and their corresponding true output vectors.

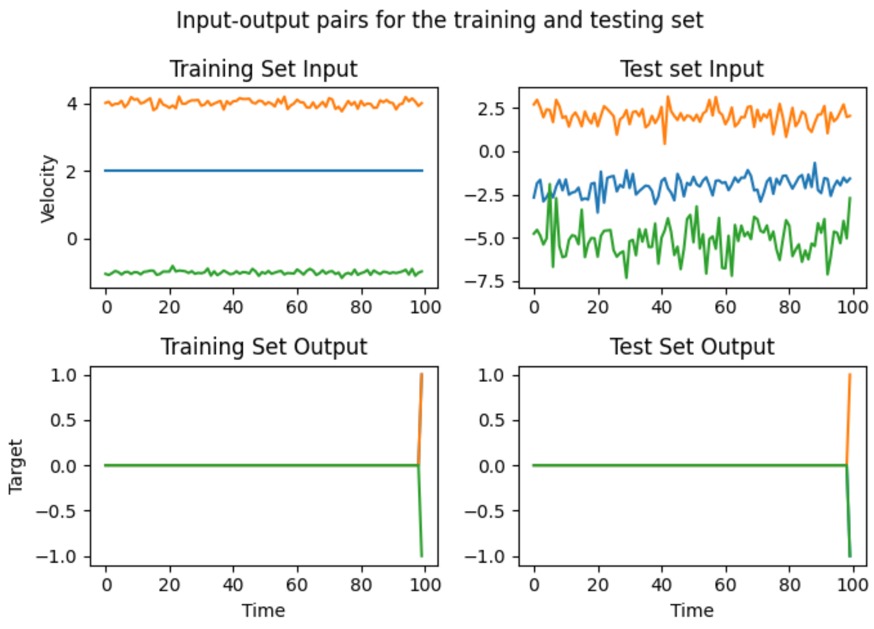

Figure 7: Sample of input output pairs for the 2AFC task. The top row represents the input vectors, and the bottom row represents the true output vectors.

From Figure 7 we see that when the initial input vector is positive, the velocirty stays positive, and the tru output is 1. Conversely, when the initial input vector is negative, the velocity stays negative, and the true output is -1.

3.2 Designing an RNN for the 2AFC Task

Now that we created our task, we can design the RNN that will learn from the training set to predict the output of the test set.

A Recurrent Neural network is a set of inter-and-self connected nodes that can be used to model the input output pairs of a sequence. In our model, each neuron x_i will have a firing rate given by $\phi(x_i(t))$, where $\phi = \tanh$. The parameters of the model are J_{ij}, the synaptic weights between neurons, B_{ik} which are the weights between the input to the network and the neurons, and W_{kj} which are the weights of the output of the neurons.

Taken together, we have that the dynamics of a neuron, given P inputs of signal u_k are given by:

$$\dot{x}_i = -x_i(t) + \sum_{j=1}^{N} J_{ij}\phi(x_j(t)) + \sum_{k=1}^{P} B_{ik}u_k(t) \tag{5}$$

and the output of the network is given by:

$$z_k(t) = \sum_{j=1}^{N} W_{kj}\phi(x_j(t)) \tag{6}$$

Training this network amounts to finding the optimal values of J_{ij}, B_{ik}, and W_{kj} that minimize the error between the output of the network and the desired output. This is done through backpropgation and gradient descent, where the error from a predicition is shared throughout the network and the weights are updated accordingly.

While the above is the general makeup of an RNN, we will have to do some additional work to adapt it to the 2AFC task. Firstly, we will need to discretize the neuronal dynamic for simulation. This can be done using Euler's method resulting in the following:

$$x_{i,t+1} = (1 - \Delta t)x_{i,t} + \Delta t(\sum_{j=1}^{N} J_{ij}\phi(x_{j,t}) + \sum_{k=1}^{P} B_{ik}u_{k,t}) \tag{7}$$

which can be vectorized as:

$$\mathbf{x}_{t+1} = (1 - \Delta t)\mathbf{x}_t + \Delta t(\mathbf{J}\phi(\mathbf{x}_t) + \mathbf{B}\mathbf{u}_t) \tag{8}$$

Finally, we need to define the loss function the network will use to learn the task. For this network, we will use the mean squared error loss function, which is defined as:

$$\mathcal{L} = \frac{1}{P}\sum_{t=1}^{T} m_{p,t,k}(z_{p,t,k} - z^*_{p,t,k})^2 \tag{9}$$

where P is the number of nuerons in the network, p is the trial number, t is the timestep, k is the input dimension, and $m_{p,t,k}$ is the mask for the network. The network will learn to minimize this function by adjusting its parameters via gradient descent, i.e., to make its output $z_{p,t,k}$ as close to $z^*_{p,t,k}$ as possible.

3.3 Training and assessing the RNN

Having generated our data, and designed our network, we can now feed it the training set so that it can learn the task. Our network will consist of 128 neurons, and will consider $T = 100$ timesteps. Figure 8 illustrates the decrease in the learning loss over 50 training epochs, with a learning rate of $lr = 0.001$.

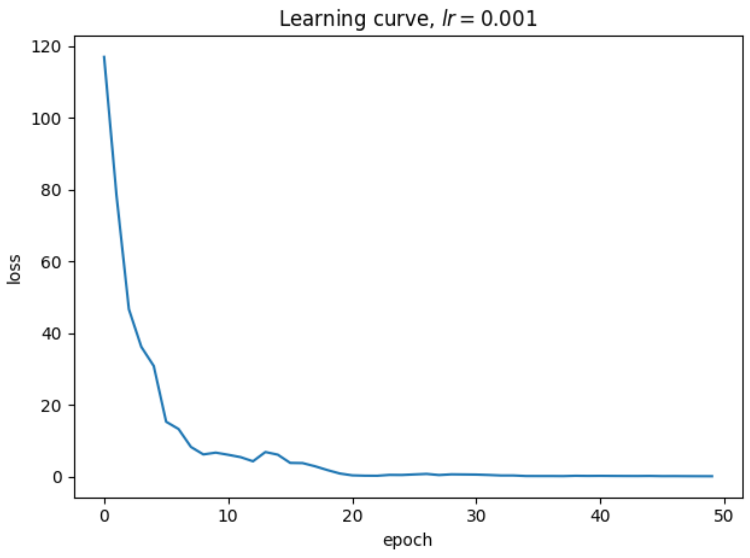

Figure 8: The learning curve of the RNN over 50 epochs, minimizing the Mean Squared Error with a learning rate of 0.001

We can assess the performance of our network by inspecitng it's accuracy, i.e., how well it did on categorizing the input vectors in each set. We define accuracy as

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{the total number total predictions}}. \tag{10}$$

Assessing our network, we find that it has a 100% Accuracy on the training set, and a 94.44% accuracy on the test set. This means that our model has perfectly learned the training set: given a sequence from the set it will always produce the correct result. However, it has not learned the test set perfectly: it will only be correct around 94% of the time. This may be due to the higher level of noise in the trials in the test set, which makes it harder for the network to predict the correct output.

3.4 Analyzing the RNN's Dynamics

Now that we have trained and asssessed our model, we can visualize the network dynamics to get a sense of the decision making process it carries out. Namely, we can inspect the model's guesses over time, to see how the network's decision evolves over time.

3.4.1 The Training Set

Figure 9 below illustrates the the model's readout trajectory over time on the training set, along with the mean readout trajectory, seperated into trials where the true output was 1 (Positive runs), and trials where the true output was -1 (Negative runs).

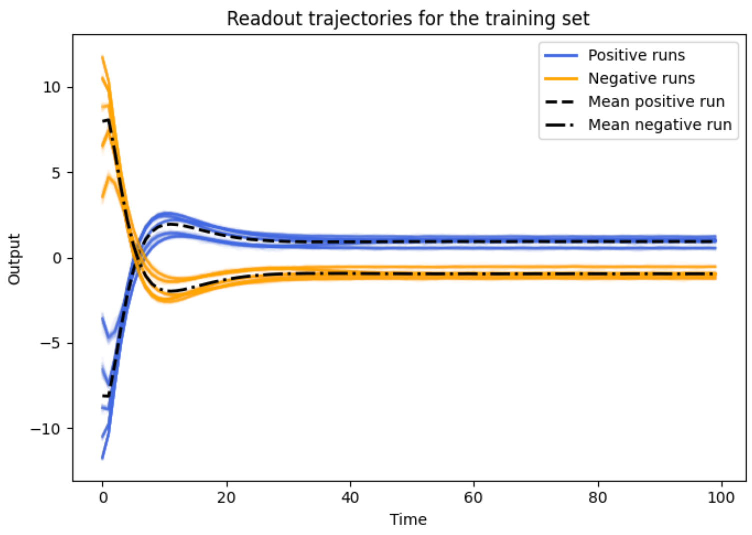

Figure 9: Model readout trajectories for the training set, over 500 runs. Orange trials are those where the true output was 1, and blue trials are those where the true output was -1. The dashed black lines are the mean readout trajectories.

Figure 9 shows visually what our training-set accuracy score reported numerically: the network perfectly categorizes the trails. Not only are the mean readout trajectories correct, but each individual trajectory correctly catagorizes the trial. That is to say: when the true output is 1, the network's final readout trajectory is always positive, and when the true output is -1, the network' final readout trajectory is always negative.

Figure 10. illustrates the principle components of the network's readout trajectories, and the projection of the test set readout trajectories onto the first two components.

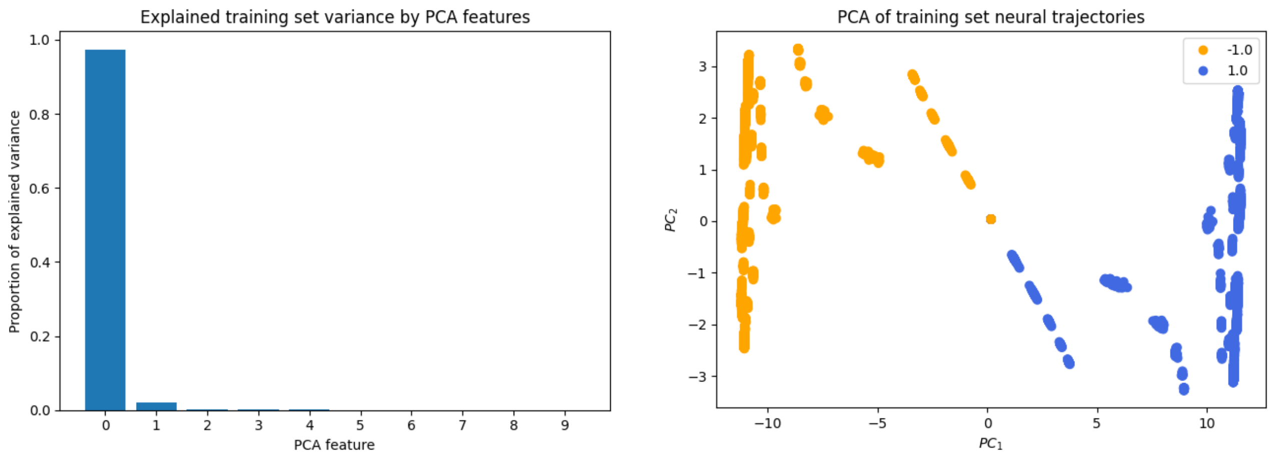

Figure 10: The explained variance of the principle components of the network's readout trajectories (left), and the projection of the train set readout trajectories onto these components (right), colored by the true trajectory. We see that almost 100% of the variance is explained by the first two components. Projecting onto these components, we can see the clean seperation between the two classes by the network

From Figure 10, we can ascertain that the dynamics of the network can be nearly fully captured by the first 2 principle components. Moreover, we can see the network's 100% accuracy in categorizing the trials, in that the two classes are cleanly sepearted by the network.

3.4.2 The Training Set

Figure 11 below illustrates the the network's readout trajectories over time on the test set, along with the mean readout trajectory.

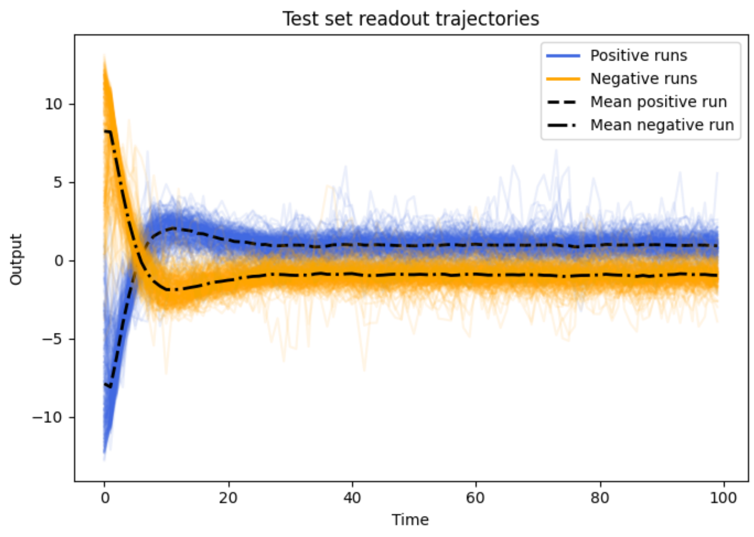

Figure 11: Model readout trajectories for the test set, over 500 runs. The dashed black lines represent the mean readout trajectories.

Figure 11 paints a different picture of model activity in the test set than the training set. While the trajectories in Fig 9 are smooth and well seperated, the trajectories in figure 11 are more erratic and closer together. Moreover, while the mean readout trajectories are correct (the network is still good at the task), there are some individual trials that are miscategorized. This behavior is caused by the novelty of the test set and the increased noise: the model has never seen the data so its predicitons are less accurate, and the noise in the test set makes it harder for the model to predict the correct output.

Figure 12 illustrates the proportion of explained variance of the principle components of the network's readout trajectories, and the projection of the test set readout trajectories onto the first two components.

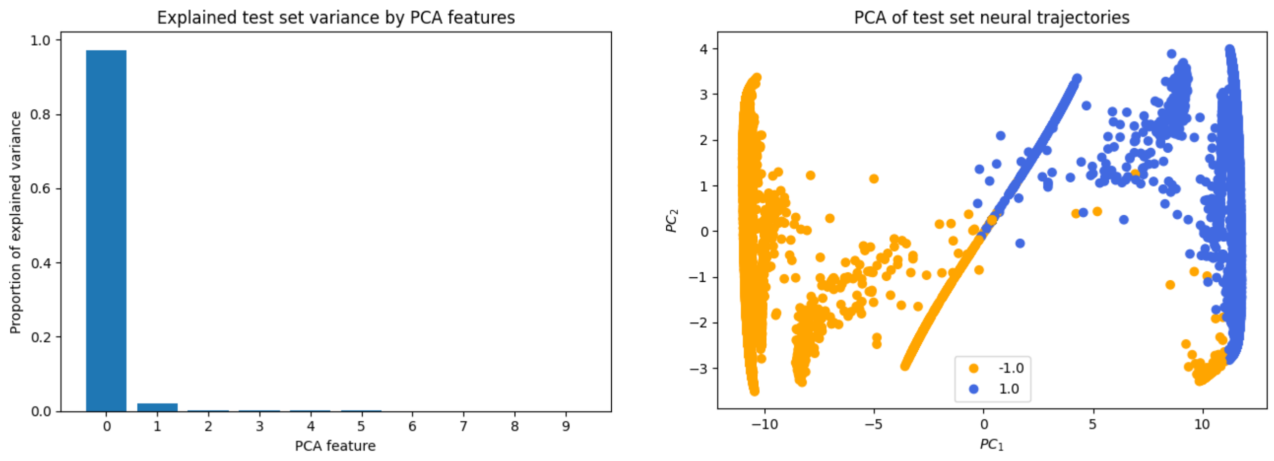

Figure 12: The explained variance of the principle components of the network's readout trajectories (left), and the projection of the test set readout trajectories onto these components (right), colored by the true trajectory. Again, 100% of the variance is explained by the first two components. However, there is some overlap in the clustering, reflecting the model's 94% accuracy on the test set.

From the figure we can see that, again, the first two principle components capture almost all of the variance. Moreover, our accuracy score for the test set is reflected in the projection of the trajectories. In this case, there is a somewhat clean sepeartion between the two classes, but there is some overlap in the clustering. This reflects the model's 94% accuracy on the test set: while it is mostly correct, there are some erroneous trials. Specifically, the network seems to miscategorize some trials where the true output is -1 as 1, but not the other way around. It may be that these trials were by chance more noisy, and thus harder for the network to predict.

3.5 Summary

In this section, we designed and trained a reccurent neural network to perform a simulacrum of a 2AFC task. We saw that by training the network to minimize the mean squared error loss function, we were able to construct an RNN that could perfectly categorize the training set, and achieve a high degree of accuracy on the test set. Moreover, we saw that the network's dynamics could be captured by the first two principle components, and that the network's accuracy was reflected in the clustering of the trajectories in this space. Thus, we saw how a network of interconnected nodes, aiming to behave similarly to a network of real neurons, could perform the 2AFC task.

4 Conclusion

Given the complexity of the brain, explicating every element of the neural circuitry that is involved in decision making is impracticle. However, we can create models that approximate this process, and use them to gain insight into the process. In this report, we examined two such models: the Drift Diffusion Model, and the Reccurent Neural Network model. With the DDM, we have a way of giving a descriptive analysis of the decision making process, and how both internal and external parameters can impact the decision making process. With the RNN, we developed a predictive model that was capable of 'perfoming' the task, giving us a glimpse of how a network of neurons could perform the task. We saw that even given a complex input, the task could be reduced to two intrinsic dimensions, allowing the network to categorize the trials with a high degree of accuracy.