Decision making: studying the drift-diffusion model and training recurrent neural networks

Sasha Lanine

2024-04-26

Contents

Introduction

A classic approach to studying neuro-computational models of decision-making is the two-alternative forced choice (2AFC) task paradigm. In a 2AFC task, subjects are asked to choose between two alternatives based on perceptual stimuli. An example of this is motion coherence experiments, where participants observe a set of moving dots on a screen and must decide whether the dots are moving predominantly to the left or to the right (Figure 1). The difficulty of the task can be manipulated by varying the proportion of dots moving in the same direction, which we call the coherence of motion. High coherence levels make the task easier, while low coherence levels introduce more noise and difficulty.

In this report, we will explore two models of decision-making in a 2AFC task. The first is the Drift Diffusion Model (DDM), which describes decision-making as an evidence accumulation process. It is a "winner-takes-all" mechanism, where the decision variable integrates noisy sensory inputs over time until it reaches a threshold, at which point a decision is made. DDM has been shown to be effective at modelling and predicting reaction time distributions in motion coherence experiments.

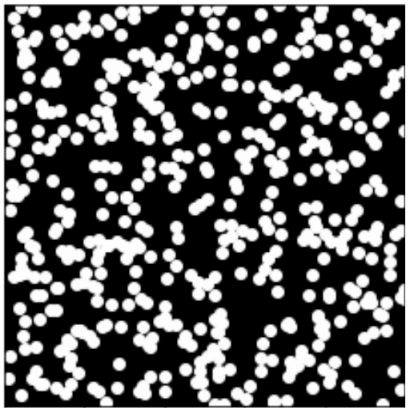


Figure 1: Example of screen used in a motion coherence experiment.

The second model we will consider is an application of Recurrent Neural Networks (RNNs) to perform a 2AFC decision task. We will build an RNN from scratch, detailing its architecture and training procedure. The RNN will be trained using a dataset we generate for the 2AFC task, where the inputs consist of noisy sensory signals and the outputs are the corresponding decision outcomes. After training, we will evaluate the performance of the RNN by analyzing its accuracy on both the training and testing sets. We will then examine the dynamics of the network and apply principal component analysis to the data it generates. This will allow us to visualize the internal representations of the RNN and assess how well it distinguishes between different decision outcomes. Finally, we will compare the results from the RNN with those from the Drift Diffusion Model (DDM).

1 Drift Diffusion

The Drift Diffusion Model (DDM) is a model based on the *accumulation of evidence* over time. In a 2AFC-task, a decision variable x is updated according to the evidence in favor of one or the other decision (A or B) until a threshold is crossed, at which point a decision is made. DDM models this process via the following equation:

$$\frac{dx(t)}{dt} = (I_A - I_B) + \sigma\xi(t), \tag{1.1}$$

where I_A and I_B denote the decision inputs (usually corresponding to intensity of stimulus) in favor of decisions A and B, respectively. In the motion coherence experiment, this corresponds to the level of coherence. The term $\sigma\xi(t)$ is a noise term: σ controls for the magnitude of the noise and $\xi(t)$ is a Gaussian white noise function.

1.1 Dynamics

We simulate the dynamics of the DDM using the following parameters: $\sigma = 7, I_A = 0.95, I_B = 1$, and $x_0 = 0$. Decision thresholds $\mu_A = 20$ and $\mu_B = -20$ are imposed in favor of decisions A and B, respectively. We will assume that $\mu_A = -\mu_B$ in all simulations in this report, and will take $\mu = \mu_A$ from this point on. We ran the simulation for 10000 time steps using Euler's method, with a step size of $dt = 0.1$. The time-evolution of the decision variable over ten trials is summarized in Figure 2 below.

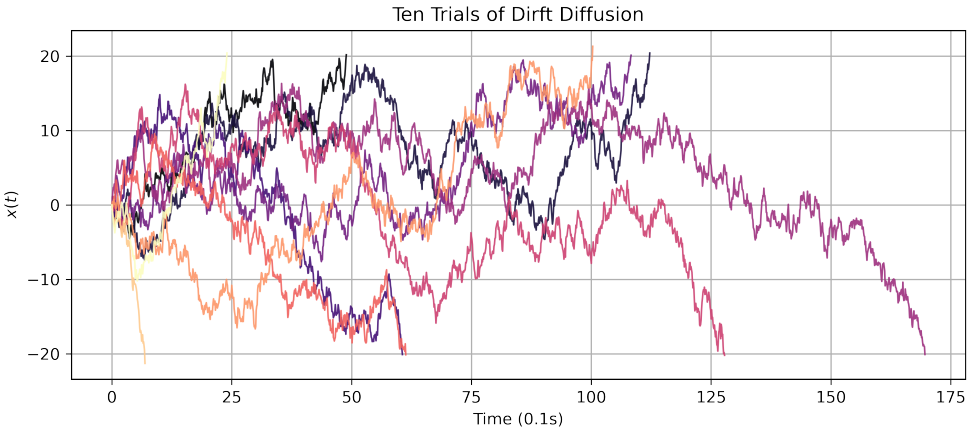


Figure 2: Ten simulations of DDM with parameters $\sigma = 7, I_A = 0.95, I_B = 1$, $x_0 = 0, \mu_A = 20$ and $\mu_B = -20$.

We see that decisions are made long before the end of the simulation period. Moreover, it appears that decisions are more likely to be made in favor of B. We investigate this further, running the simulation for $N = 1000$ trials. The results are shown in Figure 3. Note that if a decision is not made before the end of the simulation period, then we denote the outcome as *No Decision Made*.

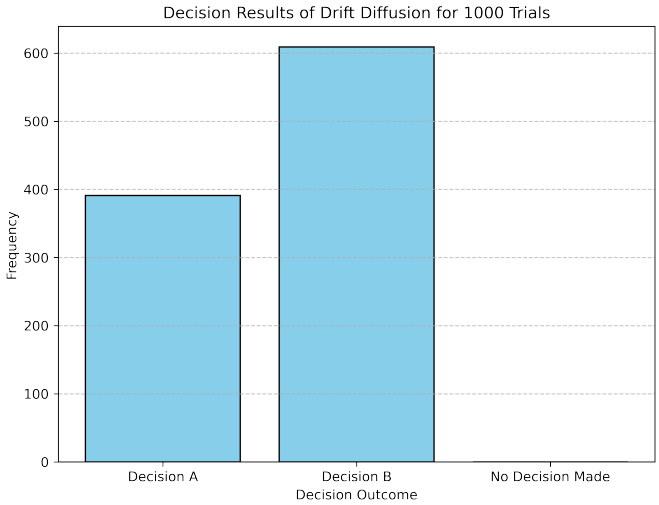


Figure 3: Decision outcomes for 1000 simulations of DDM with parameters $\sigma = 7, I_A = 0.95, I_B = 1$, $x_0 = 0, \mu_A = 20$ and $\mu_B = -20$. A decision in favor of A was made in 391 trials, and a decision in favor of B was made in the remaining 609 trials. There were no trials where no decision was made.

We found that a decision in favor of A and B occurred in 39.1% and 60.9% of the trials, respectively, This a reflection of the fact that the strength of the A stimulus is weaker than the B stimulus ($I_A < I_B$). Moreover, a decision was made in every trial, which is likely a reflection of the relatively large magnitude of $E = I_A - I_B$ and σ compared to μ

1.2 Parameter Space

We now turn our attention to exploring the parameter space of the model. In particular, we will see how decision outcomes are affected by separately varying (1) the decision threshold μ; (2) the difference in stimulus strength $E = I_A - I_B$, which we call the *evidence*; and (3) the magnitude of the noise term σ.

1.2.1 Decision Threshold μ

We ran 100 trials for each of the 100 linearly spaced values of μ in the range $[1, 100]$, holding the remaining parameters fixed at $\sigma = 7, I_A = 0.95$, and $I_B = 1$. Each trial was run for 10000 time steps of size $dt = 0.1$. Figure 4 shows the number of decision made in favor of the possible decision outcomes in our simulations as a function of μ.

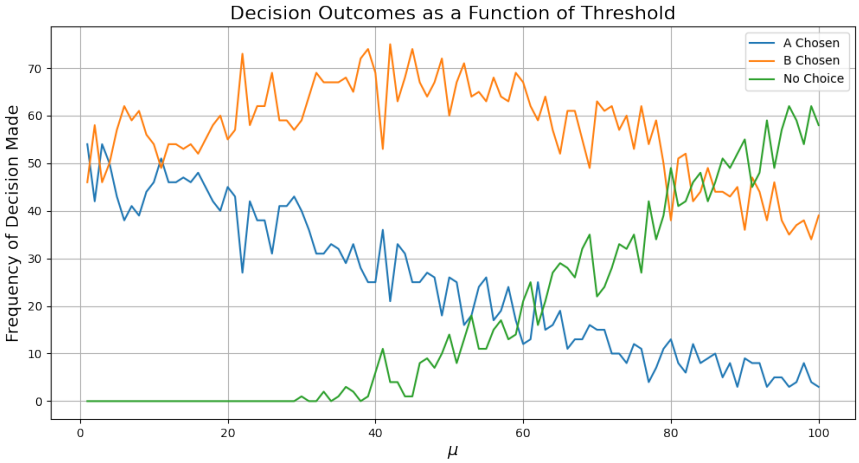


Figure 4: Decision outcomes as a function of the decision threshold μ. Remaining DDM parameters were held fixed at $\sigma = 7, I_A = 0.95$, and $I_B = 1$. The initial condition in each simulation was $x_0 = 0$.

We observe that increasing μ has the effect of making a *No Decision Made* outcome more likely. This is what we should expect. If the distance between μ and x_0 is large, then it should take longer to accumulate sufficient evidence to cross either threshold. Thus a decision is less likely to be made in a finite-time simulation as μ grows.

1.2.2 Noise Magnitude σ

We now consider the effect of varying the noise magnitude parameter σ. For 100 linearly spaced values for $\sigma \in [0, 50]$, we ran 100 trials and recorded the decision outcomes. The remaining parameters were set at $I_A = 0.95$, $I_B = 1$, and $\mu = 20$. Each trial was run for 10000 time steps of size $dt = 0.1$.

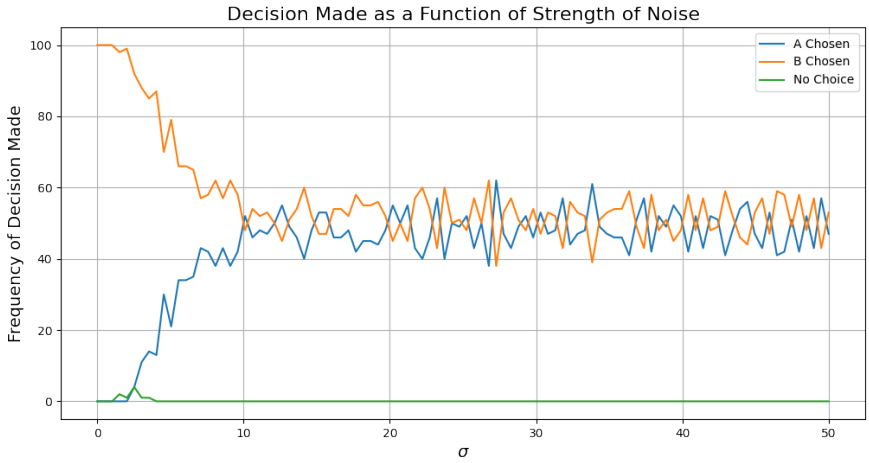


Figure 5: Decision outcomes as a function of the noise magnitude σ. Remaining DDM parameters were held fixed at $\mu = 20, I_A = 0.95$, and $I_B = 1$. The initial condition in each simulation was $x_0 = 0$.

In the absence of noise, a decision is always made in favor of B. This is because Equation 1.1 simplifies to $\dot{x} = (I_A - I_B)$, causing the decision variable to monotonically approach the decision threshold corresponding to the stimulus with the greater strength. However, when noise is introduced, the likelihood of selecting the other decision outcome increases. This probability rises with σ until the noise term becomes sufficiently large relative to E and μ, resulting in decisions being made almost randomly. In our simulations, this random decision-making appeared to occur around $\sigma \approx 10$.

1.2.3 Evidence E

We again ran 100 simulations for 100 linearly spaced values of $E \in [-0.5, 0.5]$. The remaining parameters were held fixed at $\mu = 20$ and $\sigma = 7$. The results are shown in Figure 6 below.

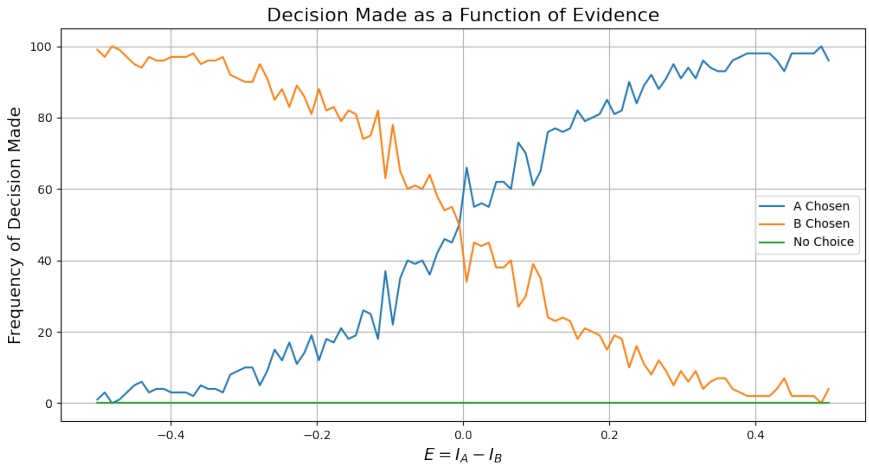


Figure 6: Decision outcomes as a function of evidence $E = (I_A - I_B)$. Remaining DDM parameters were held fixed at $\mu = 20$ and $\sigma = 7$. The initial condition in each simulation was $x_0 = 0$.

In the figure, we observe the frequency of decisions made as a function of evidence $E = (I_A - I_B)$. When E is negative, indicating a stronger stimulus for B, the majority of decisions favor B. As

E increases and becomes positive, indicating a stronger stimulus for A, the decisions gradually shift to favor A. Near $E = 0$, where the evidence for A and B is nearly equal, the frequency of decisions is more balanced.

1.3 Reaction Times

We now consider the distribution of reaction times as a function of E. For each of $E = 0$, $E = 0.01$, and $E = 0.05$, we simulated the dynamics 1000 trials. In all trials, the parameters were set to $I_B = 1$, $\sigma = 7$, and $\mu = 20$. For each value of E, we plot the distribution of reaction times for A and B decisions separately (Figure 7).

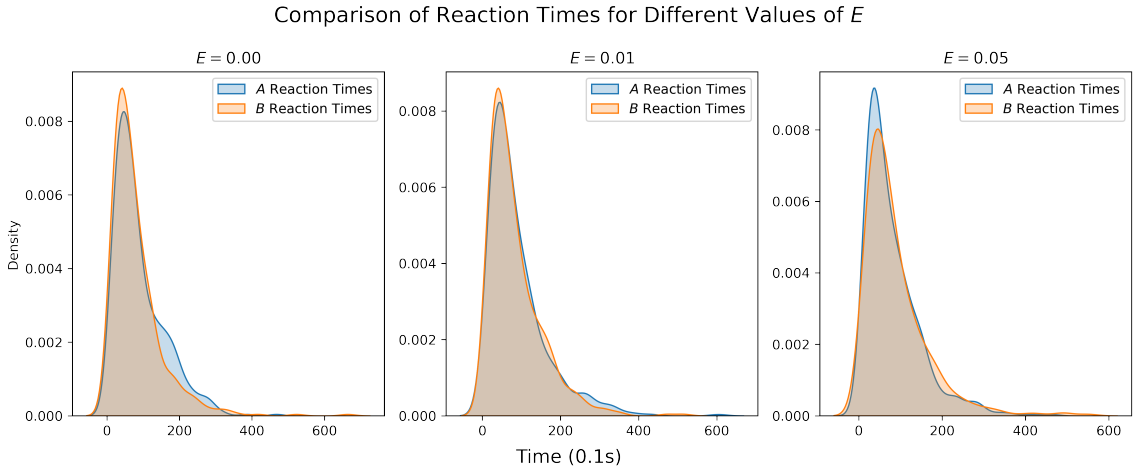


Figure 7: Comparison of reaction times for different values of evidence E. The plots show the distribution of reaction times for decisions favoring A (blue) and B (orange) across 1000 simulations with parameters $I_B = 1$, $\sigma = 7$, and $\mu = 20$.

When $E = 0$, the reaction times for decisions favoring A and B are nearly identical. This is expected because when $E = 0$, Equation 1.1 reduces to $\dot{x} = \sigma\xi(t)$, indicating that decisions are made randomly. The peak density of reaction times is around 100 time units, indicating that decisions are made relatively quickly when the evidence is neutral for these parameters.

As E increases, indicating stronger evidence for A, the reaction times for decisions favoring A become faster and more frequent, while those for B become less frequent and slower. This demonstrates the sensitivity of the drift diffusion model to changes in evidence. The results are consistent with the model's predictions, where increased evidence accelerates decision-making towards the favored choice.

This concludes our exploration of the dynamics of the drift diffusion model. We now turn our attention to the model of decision making based on RNNs.

2 RNN Models of Decision Making

In this section, we explore the application of Recurrent Neural Networks (RNNs) to model decision-making processes within the framework of a two-alternative forced choice (2AFC) task. The objective is to build and train an RNN to perform the 2AFC task, subsequently analyzing its performance and the underlying dynamics.

To do this, we must first decide on the architecture of the RNN. We will then define the task and training procedure. Once this is done, we will have a closer look at the dynamics of the

system. Finally, we will see conclude by exploring how principal component analysis (PCA) can shed light on the underlying structure of network's representations.

2.1 Formal Structure

2.1.1 Architecture

The architecture of the RNN used in this experiment is shown in Figure 8. We assume that it is composed of N recurrently connected neurons, whose activity at time t is denoted by $\mathbf{x}(t) \in \mathbb{R}^N$. The strength of the connections between the neurons is described by a matrix $\mathbf{J} \in \mathbb{R}^{N \times N}$. At each point in time, the network receives m external input signals described by $\mathbf{u}(t) \in \mathbb{R}^m$. These are then translated into the network via an input connection matrix $\mathbf{B} \in \mathbb{R}^{N \times m}$. The dynamics of the network are governed by

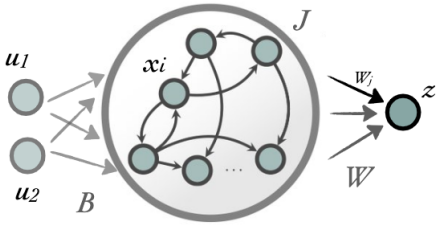


Figure 8: Architecture of RNN used in modelling decision making within the framework of a 2AFC task.

$$\frac{d\mathbf{x}}{dt}(t) = -\mathbf{x}(t) + \mathbf{J}\phi(\mathbf{x}_t) + \mathbf{B}\mathbf{u}(t), \tag{2.1}$$

where ϕ is a non-linear activation function. In our case, we will always take ϕ to be the hyperbolic tangent function. The network also provides output to a readout value $z(t) \in \mathbb{R}$ at each point in time via an output weight vector $\mathbf{w} \in \mathbb{R}^N$. That is,

$$z(t) = \mathbf{w}\phi(\mathbf{x}_t). \tag{2.2}$$

We now must define the task we want the network to perform, as well as the training procedure we will use.

2.1.2 The Task

The first step to modelling the 2AFC task with the RNN described by Equation 2.1 is to decide on the input signals. For our experiment, we will use a one dimensional noisy input

$$u(t) = v + \xi(t), \tag{2.3}$$

where $\xi(t) \sim \mathcal{N}(0, \sigma)$ and $v \in \mathbb{R}$ is a value fixed in time. In the case of the motion coherence task, v representing the velocity of the stimulus. On the other hand, σ inversely represents the coherence of motion, with lower values corresponding to greater coherence.

The task is then to train the RNN to produce the desired output in a trail of length T. The desired output corresponds to detecting the direction of motion, which can be captured by

$$z^* = \text{sign}(v). \tag{2.4}$$

For convenience, we will only set the last output at the very end of the trial at time T. Next we must decide on a procedure to train the RNN to perform this task.

2.1.3 Training

To train the RNN, we must define a loss function to capture the error in its outputs. We will use mean squared error. Since we only consider the output at the final time step, this is given by

$$\mathcal{L} = \frac{1}{P}\sum_{p=1}^{P}(z_p(T) - z_p^*(T)), \tag{2.5}$$

where P is the number of trials and p is the corresponding index.

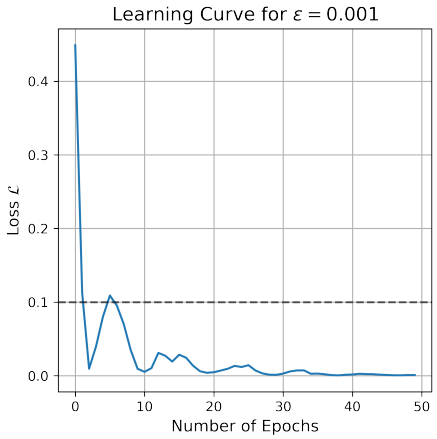


Figure 9: Loss as function of epochs elapsed for a simulation with learning rate $\varepsilon = 0.001$.

The RNN weights are then updated according to the loss on each trial via back propagation. The size of weight updates during back propagation is controlled by the learning rate ε. We found that a learning rate of $\varepsilon = 0.001$ was sufficient to ensure that the loss attained a value less than 0.1 in a reasonable number of epochs. Figure 9 shows that this was achieved in just under 10 epochs for one such simulation.

The data used for training was generated by running 500 trials of the 2AFC. At the start of each trial, the parameters v and σ were chosen uniformly at random from the sets $\{-5, -4, -3, -2, -1, 1, 2, 3, 4, 5\}$ and $\{0, 0.1, 0.05\}$, respectively.

To evaluate the performance of the RNN post-training, we must produce a second novel data set for testing and define a measure of accuracy. As with the training set, the test was generated by simulating 500 trials of the 2AFC task. The only difference is that σ was instead chosen uniformly at random from the set $\{0.5, 1\}$.

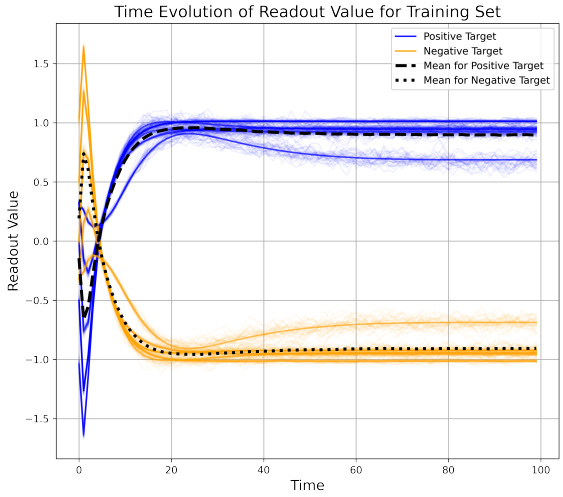
The measure of accuracy we use is the proportion of correct decisions. The RNN was found to have a respective accuracy of 100% and 99.6% on training and testing sets. Note that it is not surprising that the RNN was 100% accurate on the training set. This simply entails that, given the simplicity of the task, the RNN was sufficiently powerful to learn the exact mapping of inputs to outputs. While this does not necessarily imply that the RNN will perform well on unseen data, its 99.6% accuracy score on the testing set is an encouraging indicator that it will generalize well.

2.2 Dynamics

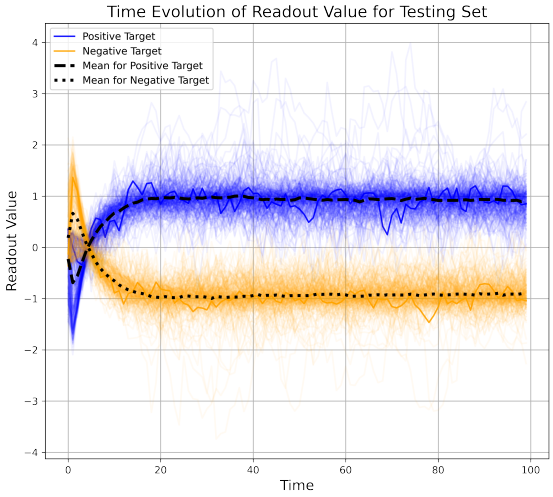
We now consider the dynamics of the RNN. We will first take a closer look at the time evolution of $z(t)$ for different decision outcomes. We will then use principal component analysis to analyze the neural trajectories.

2.2.1 Readout Trajectories

The readout trajectories for the training and test sets are both shown in Figure 10 below.



(a) Time evolution of readout values for the training set in the 2AFC task.



(b) Time evolution of readout values for the testing set in the 2AFC task.

Figure 10: Time evolution of readout values for training and test set. The trajectories represent the decisions made by the RNN, with blue lines indicating the positive target (+1) and orange lines indicating the negative target (-1). The solid lines show individual readout trajectories, while the dashed lines represent the mean response for each target.

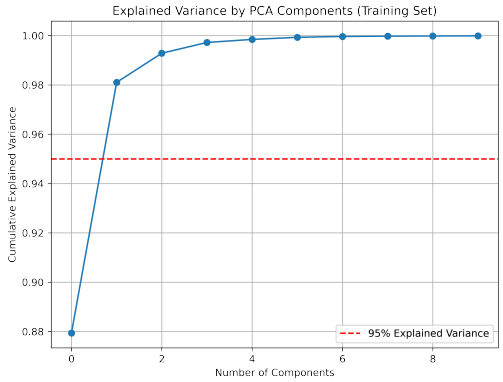
The RNN shows a consistent pattern of decision-making for both training and test sets. The time-evolution of the readout values are distinctly separated for positive and negative targets, indicating that the network is able to distinguish between the two effectively.

We observe that the testing set trajectories are noisier when compared to the training set. This is expected as the network encounters unseen data and since the training set was generated with larger values of σ. This indicates that there is still a (albeit small) level of uncertainty when the RNN makes predictions on new data. Nonetheless, these plots further indicate that the RNN successfully learned to distinguish between the two types of motion in both training and testing sets, as reflected in the accuracy testing.

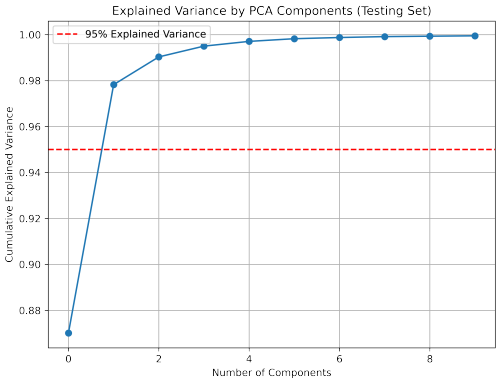
2.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a useful tool for dimensionality reduction. In this section, we perform PCA on the neural trajectories obtained from the RNN in both training and testing. The goal is to then project the high-dimensional data onto the a lower-dimensional subspace defined by the leading principal components. This will enable us to better understand the structure of the neural trajectories.

Figure 11 shows the cumulative variance explained by principal components for both the training and testing data (11a and 11b). For the training set, the first two components explain 98.1% of the variance. For the the testing set, the first two components explain 97.8% of the variance. This rapid accumulation suggests that the decision-making process can be effectively captured by a the first two principal components in both cases.



(a) Cumulative explained variance by PCA components for the training set.



(b) Cumulative explained variance by PCA components for the testing set.

Figure 11: Each plot shows the cumulative explained variance as a function of the number of principal components for the first ten components obtained from PCA. The red dashed line represents the threshold for 95% explained variance.

Figure 12 depicts the projection of the neural trajectories in the training set onto their two leading principal components. The different colors represent the two targets: blue for the positive target (+1) and red for the negative target (-1). We see that the trajectories form distinct clusters, suggesting that the RNN has successfully learned to delineate between the positive and negative trajectories. The structure of the clustering also suggests that the ten distinct branches may correspond to the ten values of v used in training. Each branch may indicate a specific velocity value, demonstrating how the RNN's internal representations vary with the input signal.

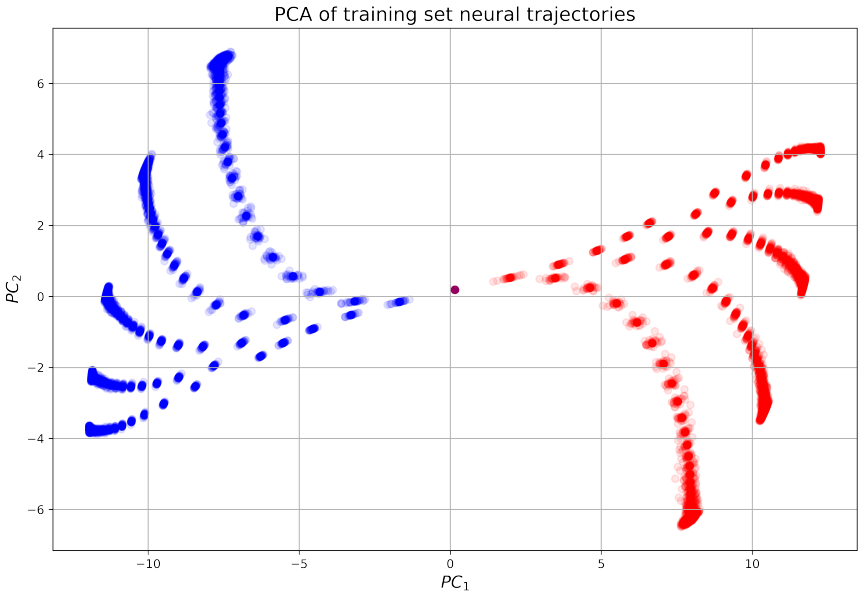


Figure 12: Projections of network activity onto the two leading principal components obtained by PCA for the training data. Blue points represent trajectories for the positive target (+1), and red points represent trajectories for the negative target (-1).

The projection of the testing data onto their two leading principal components is shown in Figure

13. It also shows clear separation for neural trajectories for the two targets. This suggests that, while the RNN is able to generalize well to new data, there exists variability in the neural responses that leads to less distinct clustering. On the other hand, this variability could also be in part explained by the larger values of σ used to build the testing set. We recommend further research to disentangle these effects to better understand the nature of low-dimensional representations of novel data in neural activity.

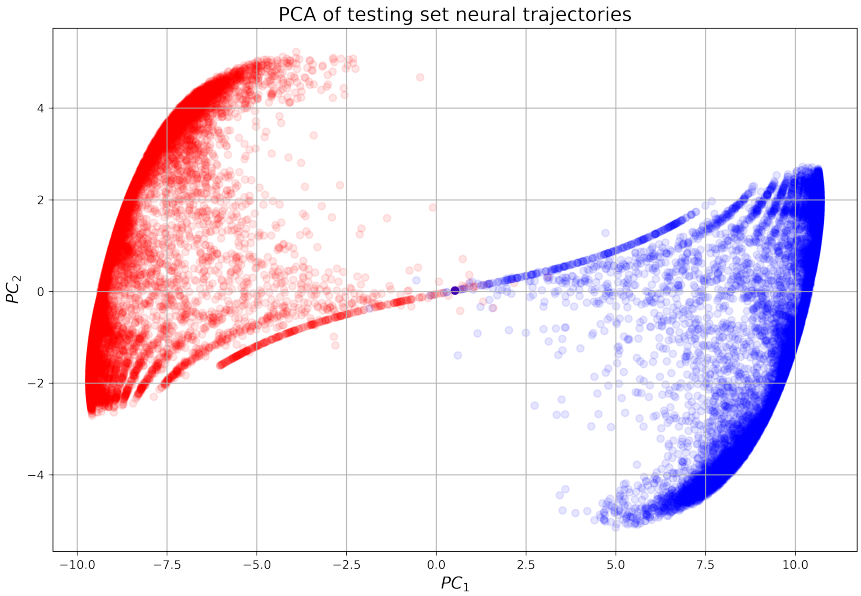


Figure 13: Projections of network activity onto the two leading principal components obtained by PCA for the testing data. Blue points represent trajectories for the positive target (+1), and red points represent trajectories for the negative target (-1).

All told, PCA has revealed that there is distinct clustering for the two different targets in the first two principal components in both the training and testing case. This indicates that the RNN was able to effectively learn to predict the direction of motion, and that it will be able to generalize.

3 Conclusion

In this report, we examined two models of decision-making in a 2AFC task: the Drift Diffusion Model (DDM) and Recurrent Neural Networks (RNNs). Our simulations of DDM demonstrated how varying parameters such as decision thresholds, noise magnitude, and evidence values can significantly influence decision outcomes and reaction times in evidence accumulation model.

For RNNs, we built and trained a network to perform the 2AFC task. It then showed a robust capability in distinguishing between positive and negative targets, with high accuracy on both training and testing sets. The readout trajectories highlighted the RNN's ability to distinguish the different targets over time. Furthermore, Principal Component Analysis (PCA) revealed that the RNN's internal representations formed distinct clusters corresponding to different decision outcomes in a low-dimensional subspace.

This report highlights the relative strengths of the Drift Diffusion Model (DDM) and Recurrent

Neural Networks (RNNs). The DDM is mathematically simplistic, reducing decision-making processes to a single decision variable. However, it also offers an intuitive explanation of the evidence accumulation process. The effects of varying its parameters are clear and easily interpretable. In contrast, the RNN, though less transparent in its representations and parameter meanings, provides a more flexible framework that can be adapted to model a wide variety of tasks. Additionally, RNNs are more capable of modeling biophysically plausible neural dynamics, allowing for better representations of cognitive processes.