Characterising CNN and LSTM-based dissimilarity measure for monkey neural recordings

Marilena Lemonari[1]

MSc Machine Learning

Primary Supervisor: Prof. Bradley C. Love
Co-supervisor: Dr. Sebastian Bobadilla-Suarez

Submission date: 24 September 2020

Acknowledgements

Firstly, I would like to sincerely express my appreciation to my supervisor, Dr. Bradley Love for his insightful comments and general guidance, which shaped the overall project and guided me towards writing an academic dissertation.

Secondly, I would like to express my gratitude to the post-doc Sebastian Bobadilla for his continuous guidance, advice and support. His enthusiasm and his ideas have inspired me throughout the duration of the project. I also wish to thank Sebastian for his time and patience.

Lastly, I thank Love Lab for pre-processing the empirical dataset, allowing me to use it appropriately for the purposes of this study.

Many thanks also to everyone who gave me tips on setting up and operating the Love Lab's GPU machine.

To my parents

Abstract

Neural coding is a field of neuroscience investigating how neurons process information when the brain reacts to sensory inputs. A fundamental neuroscience question we investigate in this study, is what dissimilarity measure is used by the monkey brain when engaged in a visuomotor task. Having insight into the nature of such measure has practical ramifications in neural data analysis. Recently, deep learning has been receiving growing attention from neuroscientists for its ability to model brain data, achieving high performance. We were therefore inspired to build a neural network producing a dissimilarity measure, in order to discover how well it describes monkey neural recordings and to what extent it captures certain attentional effects. We propose a two-step strategy of developing a CNN and an LSTM which successfully implements the visuomotor task. By incorporating an LSTM into the model strategy, we find that the representation of the same stimulus differs depending on the current task, which is either classifying colour or motion.

We obtain a dissimilarity measure from the proposed neural network by means of distance matrices. We compare dissimilarity measures computed from empirical data of monkey brains with measures from the neural network and two baseline models. Attempting to characterise the dissimilarity measure used by the monkey brain, we find that the neural network performs dimensional stretching in a way similar to what we expect from primates, verifying difference in the representation of same sensory inputs across tasks. Furthermore, we discover that the neural network results are strongly correlated with the empirical data, demonstrating that deep learning can be utilised to model brain information. Additionally, investigating which model describes biological neural recordings best, we conclude that the choice of empirical dissimilarity measure, ISI-distance or SPIKE-distance, affects the choice of best fit model. Finally, the developed neural network enables evaluating whether the representations differ or remain constant across brain regions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Characterising the dissimilarity measure used by the brain is essential in the field of neuroscience for various cognitive tasks such as decision-making, categorisation, memory-retrieval and reasoning ([8], [17] [2], [56]). We explore this basic and important aspect of neuroscience through a machine learning framework. In this study, we are investigating neural network architectures as a way of finding a dissimilarity measure for the brain data obtained from monkeys during a visuomotor task ([50]). On the machine learning side, we build a neural network to describe neural data, utilising the characteristics of the LSTM, by developing a CNN-LSTM model, alternative to the standard DCCN models. The single-unit data derived from the monkey neural recordings offer a new perspective for comparisons with the neural network findings.

The categorisation task involves classifying either motion or colour based on context, by looking at images of coloured random dots moving in a specified direction. The inputs are sequences of ordered images, depicting a fixation point, followed by a cue and then the stimulus of random dots. Cues are shown to discriminate tasks into either colour or motion, setting the context.

This visuomotor task requires the monkeys and the neural network to map similar or even identical sensory inputs to different actions. The primate brain is able to understand and use context to perform such task. In the brain, sensory information is represented by networks of neurons. The sequences of times at which neurons emit electrical pulses known as action potentials ([26]), are called spike trains. Spike trains are of particular interest to neuroscientists as they are considered to be the primary mode of information transmission in the nervous system. There are several spike train distance choices, yielding different dissimilarity measures. Comparing spike train distances is an effective way of estimating

mutual information between stimuli and responses.

In our effort to discover how well ANNs describe neural recordings and to understand the nature of the monkey brain dissimilarity measure, we compare the ANN's activations to the firing rates of neurons in several areas of the rhesus monkey brain. From the monkey data, spike train distances were calculated with the ISI and SPIKE distances (Miller Lab, MIT and Love Lab, UCL). From the model, the Euclidean distance between the pairwise activations is computed to construct distance matrices under the colour and motion conditions.

Artificial neural networks (ANNs) have been receiving increasing attention in the field of neuroscience for their high performance and their ability to model brain information ([21], [20]). There is growing research relating visual ventral stream to successive layers in CNNs, such as in [63] and [64]. As described in [64], goal-driven hierarchical convolutional neural networks (HCNNs) can be used to provide an insight into sensory cortical processing. Deep learning frameworks, like the one proposed in [45], can be beneficial for the theoretical and practical applications of neuroscience. Undoubtedly, deep learning is an effective tool for modelling neural data and therefore, in this study, ANNs are of particular interest when attempting to find the dissimilarity measure of the monkey brain.

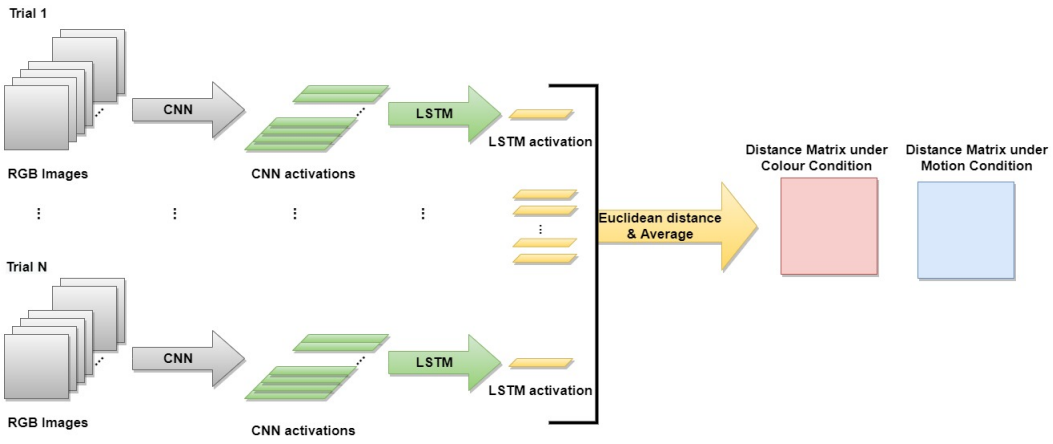


Figure 1.1: CNN-LSTM Model Strategy

Illustrating the strategy for developing an ANN model (CNN-LSTM) which is used to construct two distance matrices (one for the colour and one for the motion condition), by inputting images into a CNN, getting the activations and putting them through the LSTM. The resulting LSTM activations are averaged according to colour or motion. The Euclidean distances of pairwise combinations comprise the distance matrices, derived from the best performing CNN-LSTM based model and used to evaluate dissimilarity measures.

The strategy for developing this neural network involves both CNN and LSTM models, according to Figure 1.1. While recognising CNN's ability to trace brain activity along the ventral stream ([64]), we additionally incorporate an LSTM due to its ability to capture temporal dependencies via memory blocks ([20]). We create and train a CNN to differentiate the images into types, since CNNs are very successful in performing image-based classification tasks ([1], [36]). We obtain the CNN activations corresponding to each image frame and input them into an LSTM, which executes the same categorisation task completed by the monkeys. LSTMs are very popular for their ability to process sequential data and extract features by keeping relevant information for a long duration of time ([27], [9]), inspiring us to utilise their potential. In particular, the LSTM activations are used to create the distance matrices under the colour and motion conditions, acting as another way of measuring dissimilarity. A GPU machine is used to train the CNN and LSTM.

The comparisons between the neural network and the empirical data measures are quantified using statistical tests. Further comparisons with measures designed by other, simple models provide baseline values for the empirical data comparisons, intending to find the best way to model monkey neural dissimilarity. The general strategy for the comparisons is illustrated in Figure 1.2 and Figure 1.3.

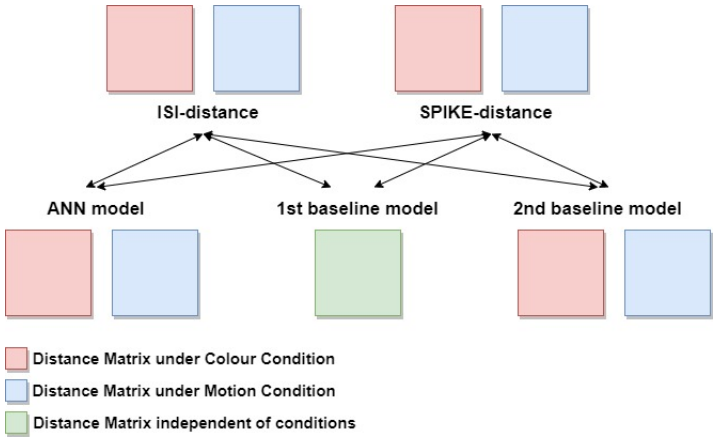


Figure 1.2: Comparisons Strategy I.

The strategy of the study includes developing an ANN model along with two baseline models. This is followed by obtaining the ISI and SPIKE distances and comparing them with the three models, intending to discover how well the measures of the empirical data correlate to the models' distance matrices and whether representations differ across tasks (colour and motion). The better a model describes the empirical data, the larger the correlation should be.

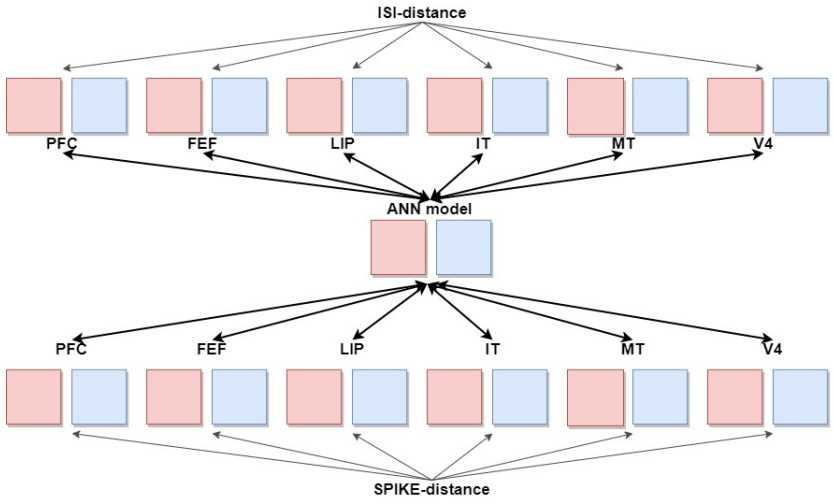


Figure 1.3: Comparisons Strategy II.
Illustrating the comparisons involving the ANN data with six subsets of each of the two empirical data measures corresponding to six brain regions of the monkey (PFC, FEF, LIP, IT, MT and V4). Comparisons are conducted separately for each task (colour and motion) and correlation coefficients are obtained to assess for difference in representations across brain regions. The red and blue colours represent the colour and motion distance matrices respectively.

The objective of this MSc project is to characterise the distance function that the monkeys are using, aiming to shed light on the computational principles of the brain. The structure of the rest of this dissertation is as follows. In Chapter 2, we review related work in literature regarding neural dissimilarity measures, deep learning and its applications, identifying and addressing research gaps. Additionally, we provide a detailed description of the original experiment conducted in [50] and give an overview of CNNs, LSTMs and their parameters. Chapter 3 and Chapter 4 explain the methodology and analyse the results, respectively. Each of the two chapters initially covers the CNN dataset and architecture, followed by the LSTM and finishing up with various comparisons within the neural network findings and between the ANN and empirical data. We draw conclusions and present them in Chapter 5, along with a discussion regarding the results and suggestions for future research.

Chapter 2

Literature Review and Preliminaries

In this chapter, we dive into the previous research on related topics and describe the main ideas and concepts behind our methodology. In Section 2.1, we summarise the current state of research on dissimilarity measures and deep learning applications. We also identify research gaps and describe our contribution. In particular, we contribute to previous research, aligning DCNNs with brain measures, by incorporating an LSTM to a CNN based model, proposing a ground truth dissimilarity measure from the model's distance matrices. Additionally, we are able to consider spike timing codes by using single-unit data from monkeys, going beyond previous work with fMRI where timing information is lost ([5]). Therefore, we investigate the representation of dissimilarity in the monkey brain, using the results derived from these new ideas. In our contribution, we aim to take a step forward in answering key neuroscience questions such as what is the dissimilarity measure used by the brain and whether such dissimilarities differ across tasks. The remaining sections of this chapter, provide a detailed review of the original experiment, the principles of the models (CNN and LSTM) and their parameters. Although these are not directly related to the objective of this study, they constitute the foundations of our methods and are necessary for the development of a model of such complexity (Chapter 3).

2.1 Background

This section contains the main review of literature, discussing topics which are key to this study. The first topic discussed in previous research concerns the several ways of measuring dissimilarity of neural data along with the resulting implications. Another topic is the usefulness of deep learning, various models of which have been utilised to model neural

recordings. The main focus is shifted towards the distance measures and models used in this study, namely ISI and SPIKE distances as well as CNNs and LSTMs. By incorporating an LSTM and utilising single-unit monkey data, we address the research gap. To the best of our knowledge, this will provide new insight into the computational principles of the brain by measuring how well the dissimilarity measures from the empirical data correlate with the model's distance matrices. Detecting dissimilarities reveals how the brain manipulates information and so it is pivotal to a range of neuroscience applications.

2.1.1 Dissimilarity Measures

In this section, we provide a summary of research on dissimilarity measures which deepens our understanding of the ISI and SPIKE distances. These will be used as empirical counterparts of the models' measure and so they comprise a component of the comparisons' analysis in this study. Dissimilarity functions are used to compute distances between spike trains. Previous research has been proposing and exploring such functions attempting to find how to best evaluate dissimilarity measures and representations in neural data.

The proposed approach in [60] emphasised on exploring the temporal structure of neural spikes using the distance between spike trains. Amongst other things, it found that distance is key to discovering the nature and precision of temporal coding, with their metrics identifying stimulus-dependent temporal structure. A simple measure for the distance between two spike trains, having time constant as a parameter, was introduced in [46]. This novel spike distance had a linear dependency with noise, a property which can determine the intrinsic noise of a neuron. Other spike train distances include the van Rossum distance ([46]) and Victor–Purpura distance ([59], [60]). Focusing on fMRI, [5] evaluated different dissimilarity measures such as Pearson correlation, Minkowski and Mahalanobis measures with the former being surpassed by other measures. Selected measures of capturing neural similarity were different across tasks but common across brain regions. In this study, we are looking to determine whether representations change across tasks and brain regions, via ISI and SPIKE distance comparisons with state of the art models.

In the experiment conducted in [50], which is essential to this study and is further discussed in Section 2.2, firing rates of neurons in several areas of the monkey brain were measured. This dataset was further processed to produce spike train distances between the possible colour and motion combinations, using three different measures, namely ISI-distance, SPIKE-distance and SPIKE-SYNCH. However, we concentrate on the ISI-distance and SPIKE-distance, as defined in [31] and [30] (Figure 2.1).

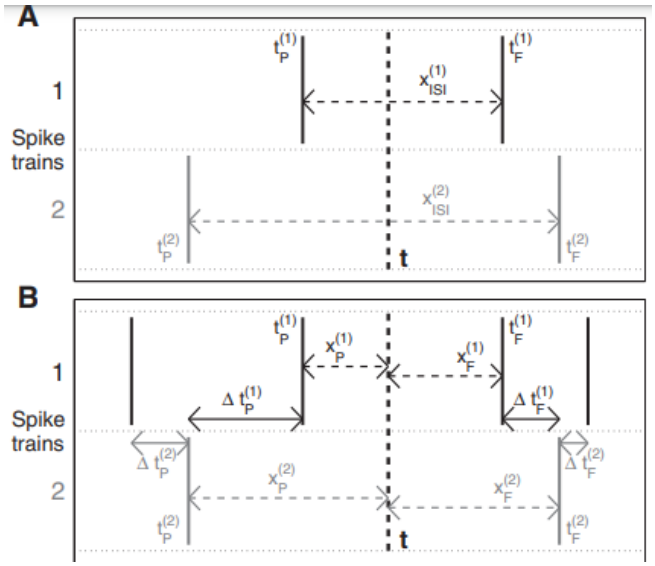


Figure 2.1: ISI and SPIKE Distances

A visual explanation of the two measures, namely ISI (A) and SPIKE (B). (A) illustrates quantities used to define the ISI-distance (dissimilarity profile $I(t)$, see Equation 2.1) for an arbitrary time instant t. (B) illustrates the additional quantities used for the SPIKE-distance (dissimilarity profile $S(t)$, see Equation 2.2). The figures are taken from [32].

The ISI-distance, defined in [31], is a distance function computed from neuronal spikes and is based on the instantaneous rates of interspike intervals (ISIs). It measures the distance between spike trains based on instantaneous rate synchrony (local rate dissimilarity), according to Equation 2.1 ([31]). For spike trains which are identical or have constant and equal interspike intervals with phase shift, the ISI-distance reaches 0. It approaches -1 or 1 if one of the spike trains is faster than the other ([48]).

$$
\begin{aligned}
x_{isi}(t) &= min(t_i^x | t_i^x > t) - max(t_i^x | t_i^x < t), \text{ with } t_1^x < t < t_M^x \\
y_{isi}(t) &= min(t_j^y | t_j^y > t) - max(t_j^y | t_j^y < t) \\
I(t) &= \begin{cases} \dfrac{x_{isi}(t)}{y_{isi}(t)} - 1 & \text{if } x_{isi}(t) \leq y_{isi}(t) \\ -\left(\dfrac{x_{isi}(t)}{y_{isi}(t)} - 1\right) & \text{else} \end{cases} \qquad (2.1) \\
\implies D_I &= \int_{t=0}^{T} |I(t)| dt \text{ and } D_I^s = \sum_{i=1}^{M} |I(t_i)| dt \text{ in the spike-weighted variant.}
\end{aligned}
$$

However, ISI-distance does not account for spike timing differences which is addressed by the SPIKE-distance. SPIKE-distance shares properties with the ISI-distance while taking into consideration spike coincidences, becoming 0 only for identical spikes. It is introduced in [30] and defined with respect to the minimum cost of transforming one spike train into the other, according to Equation 2.2.

$$
\begin{aligned}
t_P^{(n)}(t) &= max(t_i^n | t_i^n \leq t), \text{ and } t_F^{(n)}(t) = min(t_i^n | t_i^n > t) \\
x_{isi}^{(n)}(t) &= t_F^{(n)}(t) - t_P^{(n)}(t) \\
\Delta_{t_P}(t) &= |t_P^{(1)}(t) - t_P^{(2)}(t)|, \text{ and } \Delta_{t_F}(t) = |t_F^{(1)}(t) - t_F^{(2)}(t)| \\
S'(t) &= \frac{\Delta_{t_P}(t) + \Delta_{t_F}(t)}{2 < x_{isi}^{(n)} t >_n} \\
x_P^{(n)}(t) &= t - t_P^{(n)}(t), \text{ and } x_F^{(n)}(t) = t_F^{(n)}(t) - t \\
S_1(t) &= \frac{\Delta_{t_P}(t) x_F^{(1)}(t) + \Delta_{t_F}(t) x_P^{(1)}(t)}{x_{isi}^{(1)}(t)} \\
\implies S(t) &= \frac{S_1(t) x_{isi}^{(2)}(t) + S_2(t) x_{isi}^{(1)}(t)}{2 < x_{isi}^{(n)} t >_n^2} \\
\implies D_s &= \frac{1}{T} \int_{t=0}^{T} S(t) dt
\end{aligned}
\tag{2.2}
$$

Both distances will be used as empirical data dissimilarity measures and compared to an ML model in order to find which measure performed better. The nature of the available data is suited for temporal precision and localisation, even though we only have a few neurons per electrode. In contrast, fMRI which was used in the majority of previous research, has much worse temporal resolution and measures a proxy signal. In this study, we analyse such data, enabling us to answer important neuroscience questions, elucidating the nature of monkey brain dissimilarity measure.

2.1.2 Why Deep Learning

We intend to use deep learning to develop a neural network and relate it to brain data from the original experiment in [50], using both ISI and SPIKE distances for comparisons. There are some initial indications that neural networks develop features similar to neural representations ([65], [63] and[64]). In this section, we give the motivation behind our decision to use deep learning to describe the monkey data and implement the visuomotor task. Deep learning has achieved high performances both with regards to tackling complex tasks

and modelling brain information. The visuomotor task implemented in this study allows us to check for dimensional stretching, investigating the shared computational principles between neural network and empirical data. Also, incorporating motion in the random dots facilitates modelling with LSTMs. Therefore, we chose to develop the neural network using a two-step strategy involving both CNN and LSTM, indenting to better model the brain information.

We use deep learning because of its ability to successfully perform challenging tasks and to describe biological neural recordings. Deep learning models consist of multiple layers and use the backpropagation algorithm ([61]) to update their intrinsic parameters and learn data representations. The inputs to such models can be raw, unstructured, large datasets ([34]). Due to that, on top of the ability to find intricate structures in high-dimensional data, deep learning has nowadays found its way into several scientific fields.

Deep learning and in particular CNNs have been very popular in neuroscience. CNNs have proved to successfully model brain data due to their ability to trace brain activity along the ventral stream (during tasks like object recognition in [64]). The operations of artificial neurons, element-wise multiplications of weight representations with previous layers' patterns, act as similarity operations, allowing for comparisons between the CNN and the brain. In this study, we combine CNN with an LSTM taking advantage of LSTM's ability to process sequential data and keep important information through the sequences via memory blocks, capturing temporal dependencies in the data ([20]). Deep learning architectures, such as artificial neural networks, successfully perform tasks like image classification, object detection and tracking ([1], [67] and [7]). Biological neural networks inspire artificial neural networks (ANNs) in the effort to develop programs which can also solve perceptual problems ([24]). ANNs can be split into two types, the convolutional and recurrent neural networks.

CNNs have been frequently used for image processing, transforming raw visual input into a complex set of features. The interest in CNNs stems from their ability to include neural computation hallmarks and mimic the hierarchical organisation of cortical systems of mammals ([21]). On the other hand, recurrent neural networks (RNNs) are providing insight into sequential data structures. Both ANN types have outperformed models based on different machine learning approaches. For instance, the CNN developed in [55] gave considerably better results compared to the Support Vector Machine (SVM) for the task of identifying plant species by looking at images. Moreover, when comparing an LSTM with Random Forests and Lasso in a forecasting task, [41] found the best performing model to be

the proposed LSTM. Undoubtedly, neural networks are capable of successfully performing a variety of tasks requiring intelligence. In this study, we go one step further. Apart from just using deep learning to effectively tackle a categorisation task, we are looking to understand to what extent neural networks can be used to model brain information. In biological vision, [33] suggested that deep neural networks can not only address object recognition tasks but also predict high-level neural responses. [49] compared ANN models to discover whose internal representations are more like those of the brain. Although models like DenseNet-169, CORnet-S and ResNet-101 most resembled the primate brain, all performed reasonably.

In conclusion, neural networks have proved to be the best visual stream models currently available, outperforming all previous neural-vision models. They have been utilised towards unravelling the neural mechanisms of the primate brain. Therefore, we develop a deep learning model tackling the task in the original experiment. The developed neural network is a CNN-LSTM based model. Additionally, we test the limits of the developed ANN by comparing behavioural responses of the model and the monkeys.

2.2 Original Experiment

This section summarises the original experiment conducted in [50]. Specifications of this experiment have been used for training the CNN and the LSTM models in our study, even though the description of this experiment does not further contribute to how we measure neural dissimilarity.

2.2.1 Experiment Description

The experiment conducted in [50] constituted the main inspiration for our study, defining the sensory inputs and task we also used for building the neural network. Also, brain activity from the monkeys, participating in the experiment, generated the relevant data for the spike train distances which are used in our study for comparison purposes. The original experiment's goal was to discover how sensory information flows by recording and analysing the firing rates from six regions of the monkey brain when engaged in a categorisation task discriminating colour and motion. In particular, two rhesus monkeys were exposed to a series of sensory inputs and the six monitored brain areas were the middle temporal area (MT), the visual area four (V4), the inferior temporal cortex (IT), the lateral intraparietal area (LIP), the prefrontal cortex (PFC) and frontal eye fields (FEF).

Screen Images

The experiment involved monkeys concentrating on a screen, showing images sequentially in the form of a video of maximum length 4.5s. Once the monkeys focused on the screen, the experiment started. During the sessions, images of fixation point, cue and stimulus were displayed. The input data of this experiment was our reference for creating the datasets for the CNN and LSTM training.

For the first 0.5s, the white fixation point image appeared in the centre of the screen with two additional grey-coloured points on the left and right of the fixation point (the two points were equidistant from the fixation point).

After that, the cue image was displayed for 1s. The cue image depicted one of four possible cue shapes, namely a cross, quatrefoil, circle and a triangle (Figure 2.2). The cue appeared on top of the three points mentioned above and set the context, relating cue shapes to one of two classification tasks (colour or motion).

Figure 2.2: Cue Shapes

Visualising the cue shapes used in the original experiment to discriminate the task. The cross and the quatrefoil indicate the motion task, whereas the circle and the triangle indicate the colour task. Figure taken from [50].

Finally, stimulus images were shown for up to 3s. These were images of random coloured dots, all moving synchronised in a specific direction. The random dots appeared within a circle centred at the fixation point. All stimuli were 100% coherent. An example of the images shown during one particular trial is illustrated in Figure 2.3.

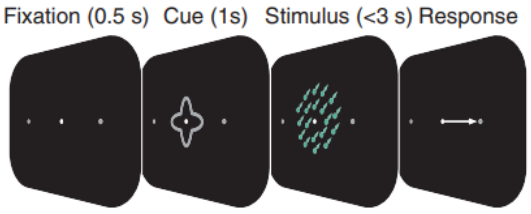
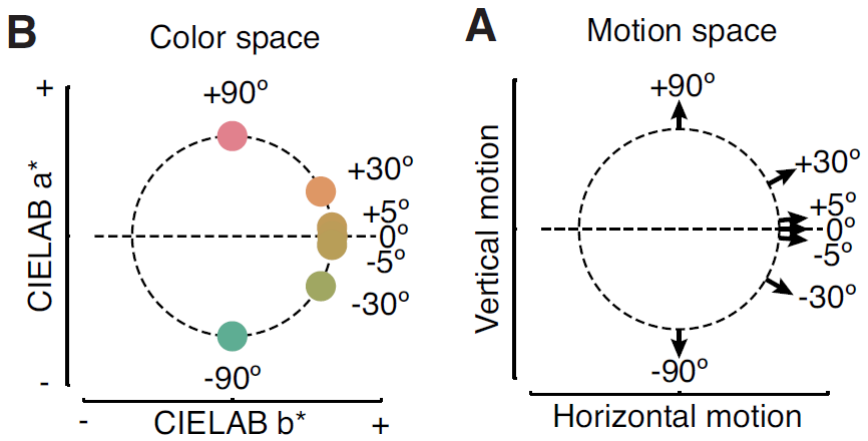


Figure 2.3: Stimulus Example

Displaying an example of the series of image types shown during the original experiment, starting with the fixation point, followed by the task cue and lastly the stimulus images with example choices of colour and motion. Figure taken from [50].

There were seven possible colours and seven possible directions for the random dots. The colour and direction labels are $\{-90°, -30°, -5°, 0°, 5°, 30°, 90°\}$ for each, with $-90°$ being the downward motion and green colour and $90°$ being the upward motion and red colour (Figure 2.4). In total, there were 21 possible colour-motion combinations which are presented in Figure 2.5.



(a) Colour Space (b) Motion Space

Figure 2.4: Colour and Motion Space

Indicating the labels with the corresponding colours and motion directions, given by the original experiment. The boundaries for the conditions were considered to be the respective colour and motion with label $0°$*. Figure taken from Supplementary Material of [50].*

Visuomotor Task

Both monkeys were trained on a visuomotor task, similar to the task we consider for the neural network. The four cues mentioned above, corresponded to a specific task, either classifying motion or colour. Two out of the four cues (cross and quatrefoil) were associated with the motion task, and the remaining two (circle and triangle) represented the colour task. Based on the cue displayed on the screen, the monkeys' response could be interpreted as classification of either motion (downwards or upwards) or colour (red or green). They had up to 3s to give their response with a leftward or rightward saccade, while the stimulus images were shown on the screen. If the required task were classifying motion, a leftward saccade would correspond to an upward motion whereas a rightward saccade would correspond to a downward motion. Similarly, for the colour task, a leftward saccade would mean the monkey classified dot colour as green, whereas a rightward

saccade, as red. The combinations with either motion or colour labelled 0°, are on the category boundary (Figure 2.5) with no clear indication what the correct classification should be. This experiment included a total of five information types, namely cue identity, task, motion direction, colour and choice. The level of complexity of this task, first discriminating and then classifying colour or motion accordingly, allowed us to investigate whether the neural network we develop reflects attentional effects found in the monkey data, like dimensional stretching.

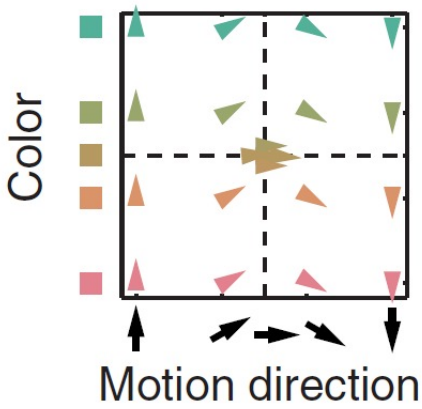


Figure 2.5: Combination Chart
Chart of all 21 possible colour and motion combinations, along with a visualisation of the motion direction and the colour for each of these combination. The black dashed lines represent the category boundaries for colour and motion. Figure taken from [50].

Results

There were several sessions and trials in the experiment. Both the monkeys' responses and the firing rates of neurons in the six cortical regions of their brain were simultaneously recorded. The findings of the original experiment were promising, as the monkeys managed to correctly categorise the two features (colour and motion) with accuracy 94% and 89% for the motion and colour tasks respectively. These results excluded trials with stimuli on the category boundaries. Additionally, they found that every brain region had noteworthy encoding for all the information types. However, Table 2.1 and Table 2.2 show the more in-depth findings in terms of information strength and frequency in each region. In our study, the classification accuracy will be used as a reference point to assess the LSTM's

performance, whereas Table 2.1 can be used to find the information type that the neural network behaves like most.

Information type	Information strength	Information frequency
Cue	V4 & IT	V4 & IT
Task	V4 & IT	All
Motion	MT	-
Colour	V4	-
Choice	LIP, FEF & PFC	LIP, FEF & PFC

Table 2.1: Original experiment findings I
Summarised results from [50]

Information type	Information flow
Cue	Bottom-up, first in MT, then LIP, V4, IT, FEF & PFC
Task	Top-down from frontoparietal to visual cortex
Motion	First in MT, then LIP, V4, IT, FEF & PFC
Colour	First in MT, then V4, LIP, FEF, IT & PFC
Choice	Simultaneously in frontoparietal regions & travelled to FEF and sensory cortex

Table 2.2: Original experiment findings II
Summarised results from [50]

2.2.2 Experiment Dataset

The specifications for the construction of the dataset mainly came from the Supplementary material of [50]. When building the datasets for the CNN and LSTM, the images were created to be as true to the original as possible.

There were different image types throughout the original experiment such as fixation point images, cue images and stimuli images. The cues were grey shapes with 1.5° visual angle diameter and centred at the fixation point. Four cues were used instead of two, in order to differentiate task from cue identity.

The stimuli were random dots of specific colour and motion direction. Stimuli images presented those dot patterns appearing centrally around the fixation point in a circle of 3.2° diameter. Inside the circle, there were 400 dots with dot diameter of 0.08° and random initial position. The speed of movement of the dots was $1.67°/s$ for half trials ("slow") and $10°/s$ for the other half ("fast").

All things considered, this experiment has been our main guide when creating an artificial neural network implementing this categorisation task, trained on similar input data. We constructed the input dataset to resemble the series of images of the original experiment.

2.3 Models

In this section, the CNN and LSTM model principles and architecture characteristics are presented and analysed. This explanatory analysis provides the framework for the development of the CNN-LSTM based model, even though it is not central to what we are testing.

2.3.1 CNN

CNNs are a type of ANNs, inspired by human visual neurons ([40]). CNNs are comprised of three key characteristics, namely convolution, max pool and fully-connected layer ([62]). The CNN architecture includes repetitions of many convolution layers followed by a max pool layer. After these repetitions, there are one or more fully-connected layers. The nature of the human-inspired CNN layers ranks CNNs to the top of the list of models for neuroscience-related tasks, like the one considered in this study.

Convolution Layer

The convolution layer takes a 3D input tensor of shape (c, d, d) and outputs a tensor with lower dimensions. The first input component c denotes the number of channels of the tensor. If the input is an RGB image, there are three input channels, one for each of the red, green and blue colours. The dimensionality reduction in the convolution layer involves a filter called kernel. The kernel is a tensor, usually of shape (k, k), with most common choices the 3 x 3 and 5 x 5 kernels. The kernel is multiplied element-wise with each (k, k)-sized block of the input tensor. The summed up values of each operation are elements of another tensor called feature map. RGB images have three channels and so three feature maps. The resulting feature maps emphasise on the important features, making this 2D convolution operation useful for feature extraction. Figure 2.6 shows an example of such operation. This process is repeated for a predefined number of kernels. Using more kernels will be computationally more expensive, but more features will be found.

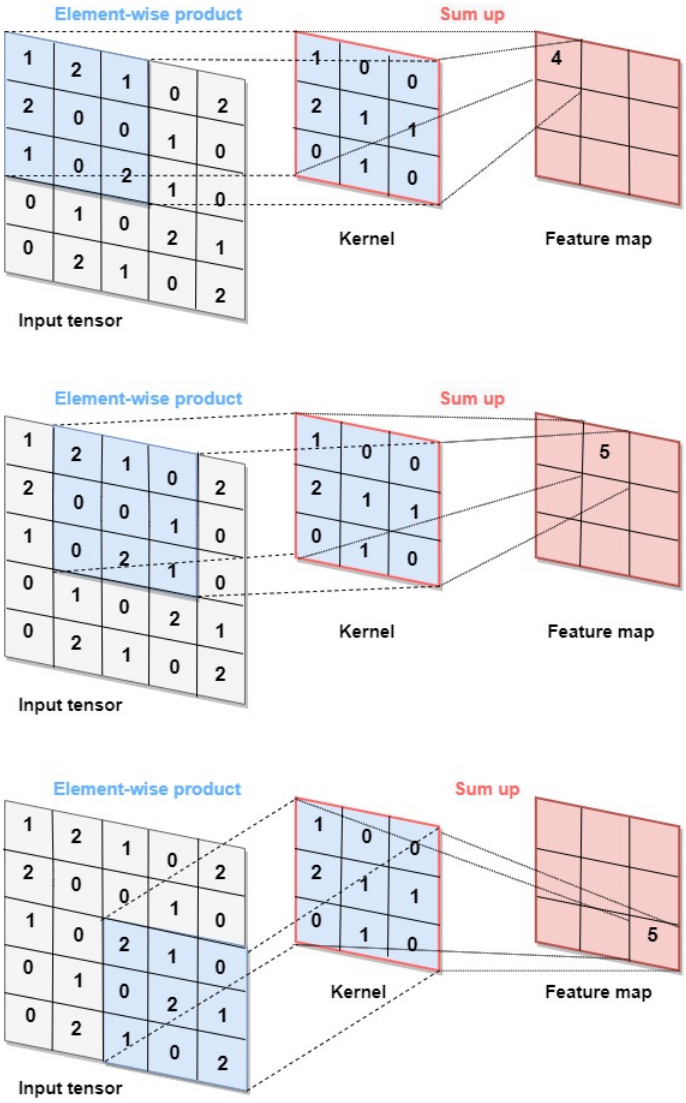


Figure 2.6: Convolution Operation Example (CNN)
Illustrating an example of a convolution layer operation with a 5 x 5 input tensor and a 3 x 3 kernel.

As explained, convolution reduces the input dimensions using a kernel. Depending on the choice of kernel size, the operations may not be compatible with the input size. This problem is addressed by introducing the padding parameter representing the addition of pixels to the edge of the image. An example is shown in Figure 2.7. Padding could also be used in LSTMs ([11]). The most common use for the padding parameter is adding pixels of value 0.

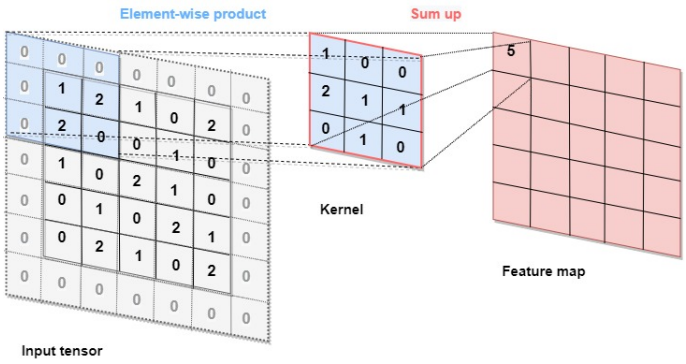


Figure 2.7: Padding Example (CNN)
Illustrating an example of a padding operation, adding zeros to the edges of a 5 x 5 input tensor and performing convolution with a 3 x 3 kernel.

Another parameter in the convolution operation is stride. Stride is defined as the distance that the kernel moves over the inputs at each step. Typically, the stride is 1, meaning the kernel shifts by one pixel between two successive steps.

Max pool Layer

The max pool layer has the same parameters as the convolution layer, namely kernel size, padding and stride. Pooling is introduced as a way of down-sampling, reducing the dimensions of the feature maps. This kind of dimensionality reduction is less computationally expensive and helps in extracting features which are important and invariant to small changes (like position and rotation). For max pooling, the kernel covers a portion of the input and chooses the maximum value at each step. The typical padding value for the max pool layer is 2. An example of such operation is shown in Figure 2.8.

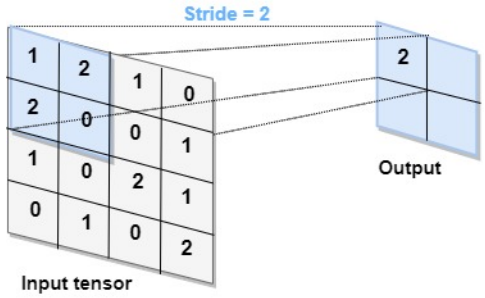


Figure 2.8: Max Pooling Example (CNN)
Illustrating an example of max pooling with input tensor 4 x 4 and stride 2.

ReLu Function

The rectified linear unit function (ReLu) is a nonlinear function defined as $f(x) = max(0, x)$ (Figure 2.9). After each convolution layer, the ReLu function is applied.

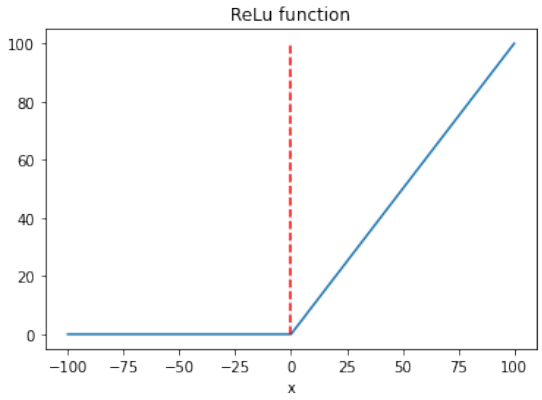


Figure 2.9: ReLu Activation Function

Fully-Connected Layer

The output of the last layer (typically a max pool layer), is a two-dimensional tensor. The first fully-connected layer transforms the 2D input to 1D. Each of these layers consists of several nodes (specified by the user), which carry weights to be learned while training. The last layer has the same number of nodes as the possible classification classes and holds the prediction probabilities for each class.

Applications in Image Classification

CNNs are widely used for binary and multi-class image classification and image recognition. The concept of dimensionality reduction found in CNNs as well as the fact that CNNs are tailored to the spatial structure of images, makes them suitable for these types of input data. CNNs have been used for image-based tasks in various disciplines. A prime example is the field of medicine. CNN architectures succeeded in tasks such as classifying lung image patches with interstitial lung disease and breast cancer histopathological image classification ([36], [53]).

Transfer learning has also been receiving increasing attention, with pre-trained CNNs being proficient in multi-class images classification ([12], [43]). Since we are also looking for

a simple and efficient way to implement multi-label image classification, we were motivated to fine-tune an existing CNN (Section 2.3.2).

2.3.2 VGG-16

The VGG-16 model introduced in [51] is a trained deep CNN, which is applicable to a range of image processing tasks. This particular model was trained on images of shape $(3, 224, 224)$ from the ImageNet challenge dataset. It consisted of 13 convolution layers and three linear layers and was trained for large-scale image classification. The increased depth proved to enhance the model, increasing accuracy.

Apart from the ImageNet experiment, the model has been modified and used in other experiments as well, proving useful in related tasks. The VGG-16 model was utilised in [19] for image-based differentiation of papillary thyroid carcinoma. Exploiting the model was beneficial as it reached high accuracy, outperforming its counterpart (Inception-v3). A similar case in [44], with the VGG-16 model detecting tree species and performing considerably better than the other detection approaches used.

Therefore, transfer learning could prove to be useful for our study. We adjust the model to be suitable for our dataset. Specifics of how this was done can be found in Section 3.2.5.

2.3.3 LSTM

RNNs are also constructed to vaguely simulate the human brain functioning. However, vanilla RNNs suffer from the vanishing gradient problem ([37]). Using LSTM instead, is a way to overcome this problem.

The LSTM was introduced in [22] and is a popular variation of the RNN. Similarly to the RNN, the inputs are processed sequentially, in memory blocks. At each time step t, the LSTM receives the current input x_t and updates the cell state c_{t-1} and hidden state h_{t-1}. This process of obtaining c_t and h_t involves three types of gates, namely the forget, input and output gates. Each gate contains a sigmoid function ([18]). The cell state is a way of carrying important information from the beginning to the end.

An illustration of the LSTM cell, including the three gates and the sigmoid and tanh functions is presented in Figure 2.10.

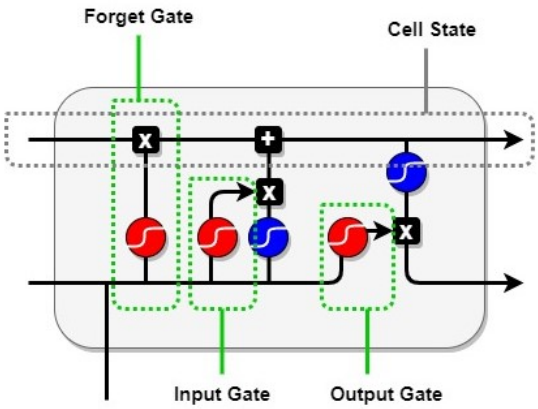


Figure 2.10: LSTM Cell

Visual representation of the information flow through an LSTM cell with the three gates and the cell state, along with the appropriate operations (element-wise multiplication, addition, sigmoid and tanh functions). Sigmoid and tanh functions are represented by the red and blue colours respectively.

Forget Gate

The first gate in the LSTM is the forget gate (introduced in [15]). In the forget gate, the previous hidden state h_{t-1} and current input x_t are concatenated and passed through a sigmoid function, yielding an output f_t. This output is multiplied element-wise with the previous cell state c_{t-1} as explicitly written in Equation 2.3, where σ denotes the sigmoid function and w_f and b_f denote the weight and bias of the respective gate f (forget gate).

$$\begin{aligned} f_t &= \sigma(w_f[h_{t-1}, x_t] + b_f) \\ q_f &= f_t * c_{t-1} \end{aligned} \tag{2.3}$$

Input Gate

The concatenated $[h_{t-1}, x_t]$, is now passed through the input gate. This means that a sigmoid function is applied, giving the i_t output. Additionally, the concatenated vector goes through a tanh function. Its output $\tilde{c_t}$ is multiplied (again element-wise) with the

input gate output i_t. Then, the new cell state c_t is calculated according to Equation 2.4.

$$\begin{aligned} i_t &= \sigma(w_i[h_{t-1}, x_t] + b_i) \\ \tilde{c_t} &= tanh(w_c[h_{t-1}, x_t] + b_c) \\ q_i &= i_t * \tilde{c_t} \\ c_t &= q_f + q_i \end{aligned} \tag{2.4}$$

Output Gate

Lastly, the sigmoid function is again applied to $[h_{t-1}, x_t]$, giving o_t, the output gate's outcome. Having these values, the new hidden state is found by multiplying o_t with $tanh(c_t)$. These operations are mathematically described in Equation 2.5.

$$\begin{aligned} o_t &= \sigma(w_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(c_t) \end{aligned} \tag{2.5}$$

Sigmoid Function

A sigmoid function is key to the LSTM and is used at each gate. Although the domain of the sigmoid is $\mathbb{R}$, its range is $[0, 1]$, since $\sigma(x) = \frac{1}{1+e^{-x}}$ (Figure 2.11). Therefore, the gates output positive values only. The closer a value is to 1, the more information flows through the gate, whereas for values closer to 0, the gate blocks most information. This is a way to keep or discard features as we go through the memory blocks.

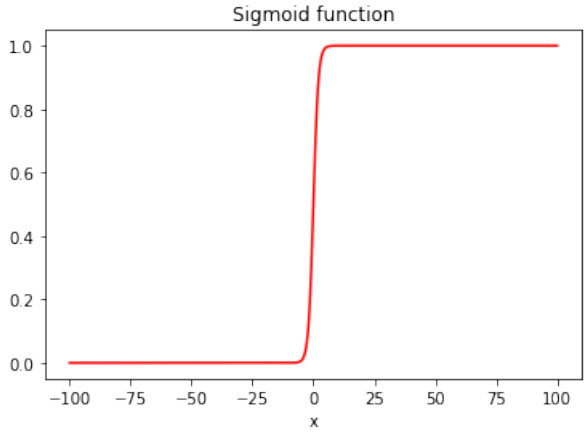


Figure 2.11: Sigmoid Function

Hyperbolic Tan Function (tanh)

The hyperbolic tan function squishes values between $[-1, 1]$ and so regulates the outputs. Its equation is $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (Figure 2.12).

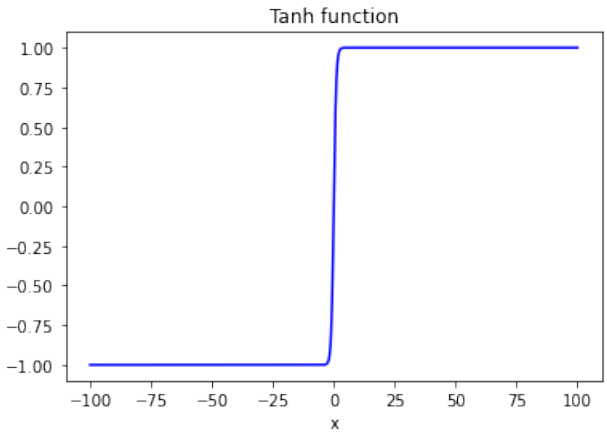


Figure 2.12: Tanh Function

Applications in Sequence Classification

LSTMs process data sequentially, making them very useful when dealing with sequence data such as videos and DNA sequences. This data type is found across numerous fields and so using LSTMs to improve sequence classification results has become quite popular. For the task detecting credit card fraud in [25], the proposed LSTM managed to achieve good results, outperforming the baseline random forest classifier. LSTM achieved high performances in NLP tasks as well, such as sentence and text classification ([10], [47]).

For video classification, end-to-end learning of CNN and RNN can be used. End-to-end learning achieved reasonable results in tasks like speech recognition ([23]) and mispronunciation detection ([35]). Convolutional LSTMs can also tackle tasks like gesture recognition ([68]) and text recognition ([6]).

Alternatively, it is also possible to model the video as an ordered sequence of frames and combine CNN and LSTM ([66], [57]). This inspired our decision to use both CNN (image feature extraction) and LSTM (sequence data) for the task at hand.

2.4 Model Parameters

2.4.1 Hyper-parameters

Machine learning algorithms have several hyper-parameters, the values of which affect the performance of the network. Optimising those hyper-parameter values is typical and important ([42]). For neural networks like LSTMs and CNNs, hyper-parameters such as batch size, learning rate and number of epochs, could help to improve the models' performance.

Batch Size

The batch size defines the number of examples to be used for the training of the neural network. In more detail, let batch size be equal to b. Then, the first b examples are taken, and the neural network is trained on those examples. Then, the network is again trained on the next b examples and so on. After each propagation, the weights of the model are updated.

The batch size is normally less than the total number of samples in the set. As a result of this, training is faster and takes less memory. The study conducted in [39] explored the use of smaller batch sizes and found them to be beneficial. On the other hand, decreasing the batch size could possibly decrease the accuracy of the gradient estimate.

Learning Rate

Learning rate is another neural network hyper-parameter. It refers to the amount by which the model changes at each iteration. Often called step size, the learning rate determines how much the weights of the model changes in response to the estimated loss. The most common value for the learning rate is 0.001.

Choosing too small learning rate values could be very time-consuming, whereas the larger the learning rate, the more unstable the training process could get. Therefore, the choice of learning rate could be challenging. [52] suggested increasing batch size, rather than adjusting the learning rate.

Number of Epochs

Another hyper-parameter having a substantial effect on training accuracy is the number of epochs. The term epoch is used to describe one complete pass of the data through the

algorithm. Consequently, if the number of epochs is n, then the algorithm works through the entire dataset n times. At each epoch, the data is fed into the algorithm in several batches according to the chosen batch size. After each example in the dataset contributed to the update of the model's weights, then an epoch is complete. This process is repeated, each time with different batch order. The number of epochs is usually large, allowing the minimisation of the loss.

Hyper-parameter tuning

There are several ways to tune the hyper-parameters. One of the most intuitive is merely using brute force until we have a satisfactory model performance. Moreover, grid and random search are also convenient hyper-parameter tuning methods. For the former, a grid of hyper-parameters is made, and for each combination, the model is trained. For the latter, the model is trained for a random selection of combinations. Random search and grid search gave similar results in [38] while tuning the hyper-parameters of SVMs. However, random search is often preferred. In [4], it was shown that random search is more efficient for numerous learning algorithms, including neural networks.

There are several other sophisticated methods developed for optimising hyper-parameters. Among them are Bayesian optimisation, early stopping-based and gradient-based optimisation, explained in [13].

For the CNN in our case, no optimisation was required. This is because the model was developed to help boost the LSTM. For the LSTM, manually tuning the hyper-parameters was sufficient due to the exceptional performance of the model.

2.4.2 Model Evaluation

Cross Entropy Loss

The loss criterion used throughout this study is the cross entropy loss. Cross entropy loss was used both as the loss function that the models were trained on and as an evaluation metric (like accuracy and F1 score). The equation of the cross entropy loss for binary classification is $l = -(y\log(p) + (1-y)\log(1-p))$. For multi-class classification ($M > 2$), the respective equation is $l = -\sum_{c=1}^{M} y_c \log(p_c)$.

The reason for choosing cross-entropy as the loss function is because it takes into consideration not only the predicted class of an example (y_c) but also the probabilities of that example belonging to the several classes (p_c). Those probabilities can be interpreted

as the confidence of the model in those predictions.

Accuracy

During training, validation and testing, we also decided to keep track of the accuracy. Accuracy is another metric used to assess the success of the model. It is calculated by counting the number of occasions that the model predicted the correct label and then dividing by the total number of examples in the set.

While accuracy is easy to understand, it can sometimes be misleading especially for imbalanced data. Evidence of this problem, known as the accuracy paradox, was presented in [58]. Thus, though it is useful to record the accuracy, it should not be the only evaluation metric.

F1 Score

An additional measure used for model evaluation is F1 score. The formula for this measure is $2 * \frac{p*r}{p+r}$, where p stands for the precision and r for the recall. The precision is defined to be the ratio of true positives divided by the sum of true and false positives. Recall is the ratio of true positives divided by the sum of true positives and false negatives.

True positives for class c, is the number of correct class c classifications. False negatives are the number of times an example was wrongly classified, while its true label was class c. So, for multi-class classification, precision is $\frac{\sum_c \text{True positives}_c}{\sum_c (\text{True positives}_c + \text{False positives}_c)}$. The recall for multi-class classification is calculated similarly. Then, the F1 score is calculated by taking an average. There are several types of averaging like macro-averaging, micro-averaging and binary-averaging. We decide to use weighted averaging.

Chapter 3

Methods and Experiments

In this chapter, we describe the methods and experiments for the development of the CNN-LSTM based model and the related comparisons. Regarding the CNN-LSTM model strategy, a CNN was developed to classify images into twelve types and then ordered sequences of images were passed through the trained CNN, obtaining activations which were inputs for the LSTM. The LSTM was trained to classify either colour or motion (depending on the cue) into three classes corresponding to upwards, downwards and horizontal for motion and red, green or yellow for colour. The experiments conducted for the ANNs were used for finding the best performing CNN-LSTM model, behaving as similar to primates as possible, tested in terms of confusion matrices. More specifically, we tried three CNNs investigating the effects of different image resolutions and six LSTMs studying the impact of different sequence lengths which correspond to different response times. The chosen neural network architecture produced a dissimilarity measure by means of two distance matrices. The measures from the neural network along with two baseline models were compared to different measures from the empirical data (ISI and SPIKE). We describe the methods used to check for dimensional stretching performed by the neural network, for difference in representations across tasks and brain regions and for correlations between the models and empirical data. Inference drawn from such results provides insight into the dissimilarity function of the monkey brain when engaged in a visuomotor task.

3.1 Visuomotor Task Overview

The complexity of the visuomotor task completed by both the monkeys and the neural network, gave the opportunity for the model to map same inputs to different outputs and

thus representing same stimuli differently (Figure 3.1). In particular, the responses of such task had different meanings, depending on the task cue, which set the context. Specifically, two out of four possible cues indicated that responses correspond to the motion categorisation of the random dot stimuli (upwards or downwards), whereas the other two cues indicated that responses represent the colour classification of the random dots (red or green). The model implementing this visuomotor task created two different dissimilarity measures, one for each condition (colour and motion), enabling us to assess whether representations change across conditions and compare to their counterparts from the empirical data.

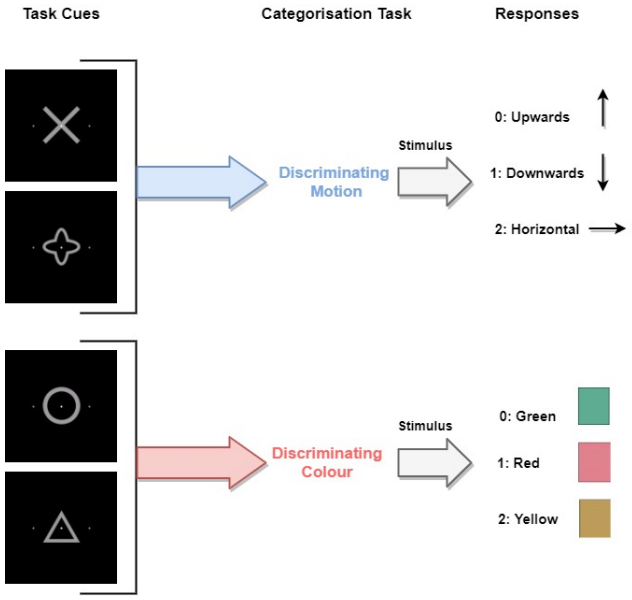


Figure 3.1: Categorisation Task
Illustration of the categorisation task. Task cues indicate either motion or colour discrimination and depending on the pointed task, responses (0,1 or 2) correspond to either motion or colour classification.

3.2 CNN

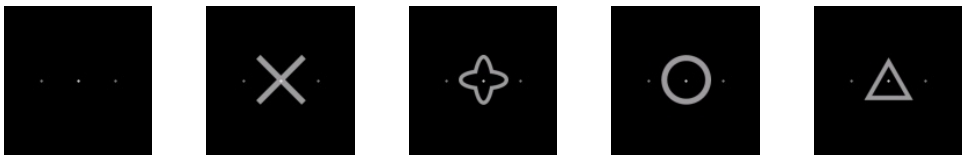
We developed a CNN to distinguish several image types found in the dataset, learning to relate each image to an input type and so encouraging more human-like behaviour. We generated the images representing the fixation point, cues and stimuli from the original experiment (Section 2.2.1). Then, we trained a CNN to classify images into 12 image

categories. These classes represented 12 image types corresponding to the fixation point, the four possible cues and seven possible dot colours. For the training of the CNN, we used the pre-trained VGG-16 model introduced in [51]. We carried out three experiments, training three CNNs with different image resolution, investigating the trade-off between computational complexity and imitation of the original images. The activations from the chosen CNN were used as inputs to the LSTM.

3.2.1 CNN Dataset

The dataset used for the training of the CNN was created from scratch, according to the image specifications in Section 2.2.2. The dataset consisted of images belonging to one of twelve possible image types, and the CNN was trained to classify the inputs into one of twelve classes representing each image type. In particular, one class represented the fixation point image, four classes represented each of the four cue shapes and seven classes represented stimuli images for each of the seven dot colours. By construction, the images resembled those in the original experiment, so that the results given by the complete CNN-LSTM model were comparable to the empirical data.

The first image type was the fixation point image. The distance between the fixation point and either of the grey dots was equal to $\frac{1}{4}$ of the horizontal length of the image. This ratio was kept constant for the rest of the image types. The following four image types are the cues, namely the cross, the quatrefoil, the circle and the triangle cue. The shapes of these cues were identical to the ones in [50]. We imported the monochrome cue images and changed the cue colour to grey, the background colour to black, adding the three dots also appearing in the fixation point image. Figure 3.2 shows these first five image types corresponding to classes 0 to 4, with shape of each image being $(3, 128, 128)$. For training, each of these images was repeated 500 times, giving a total of 2500 training examples.



(a) Fixpoint (b) Cross (c) Quatrefoil (d) Circle (e) Triangle

Figure 3.2: Fixation point and Cue Images
Displaying five out of the twelve image types used in the dataset for the CNN training. These types are the fixation point, the cross, the quatrefoil, the circle and the triangle.

The seven different choices for the colour of the random dots, all presented in [50], were described with labels $\{-90°, -30°, -5°, 0°, 5°, 30°, 90°\}$. We extracted the colours from the original experiment, using an online colour picker program, uploading the picture of the different colours (Figure 2.4 taken from [50]) and getting the HTML code of the coloured pixels. Figure 3.3 is a visualisation of the final colours used in the dataset along with their corresponding labels.

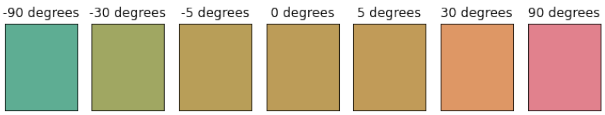


Figure 3.3: Colour Chart

Visualising the seven possible different colours of the random dots which were used in the dataset, along with their corresponding labels. The colour green has label $-90°$*, red has label* $90°$ *and the colour of the category boundary has label* $0°$*. These colours are identical to the ones presented in the original experiment in [50].*

These seven image types are stimuli images with 400 random dots of a specific colour appearing randomly inside a circle. The ratio of the diameter of this circle and the length of the side of the image was equal to $\frac{1}{3}$. For each colour choice, there were 500 stimuli images in the dataset used for the training of the CNN. These images differed from each other due to the random position of the 400 coloured dots. Figure 3.4 shows examples of these stimuli images corresponding to classes 5 to 11 (again with shape $(3, 128, 128)$). The random dots of the resulting images were more dense compared to the original study. However, the high density should not affect the networks' performance.

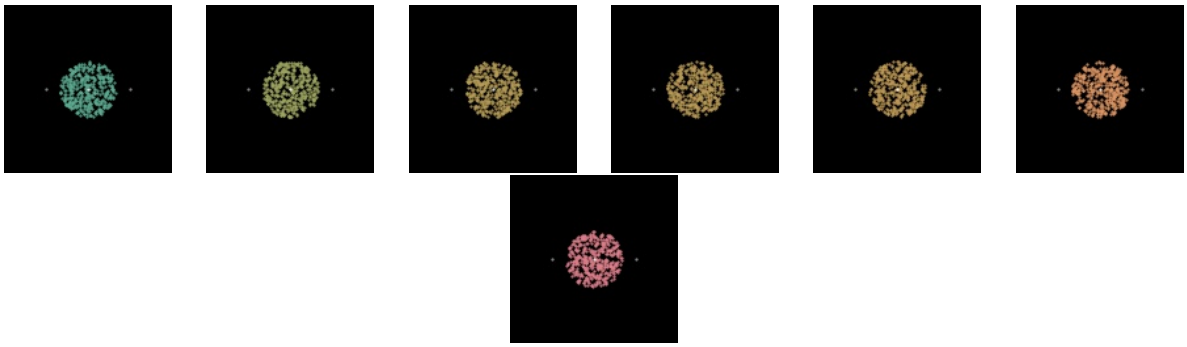


Figure 3.4: Coloured Stimuli Images

Example images of the additional seven image types used in the dataset, representing stimulus images with different colour for the random dots. The coloured random dots are initialised randomly within the circle.

3.2.2 CNN Pre-processing

The dataset used for training the CNN consisted of 500 images from each of the twelve classes. Therefore there was a total of 6000 training examples. By design, each class was represented equally in the set, and therefore there was no class imbalance.

3.2.3 CNN Evaluation

There were three sets concerning the development of the CNN, namely the training, validation and test sets. The training and validation sets were created by randomly splitting the dataset described in Section 3.2.2, with the validation set comprising 10% of the data. Conversely, the test set was constructed after the CNN training. All three sets are useful during the training and testing process.

The training set is used to fine-tune the CNN, whereas the validation set is used to check whether the model is over-fitting. The phenomenon of over-fitting refers to the case when a model perfectly fits the training data but fails to generalise out of sample. As a result, this has a very low training error but a considerably higher validation error. Therefore, keeping track of the model's performance on the validation set helps prevent over-fitting. Additionally, the validation set is used to tune any hyper-parameters of the model. By looking at the validation loss for different values of the hyper-parameters, we can choose the ones yielding the smallest error.

The test set is used to evaluate the performance of the trained model on unseen data. In this case, we generated a test set consisting of 300 examples for each class, yielding a total test set size of 3600 examples.

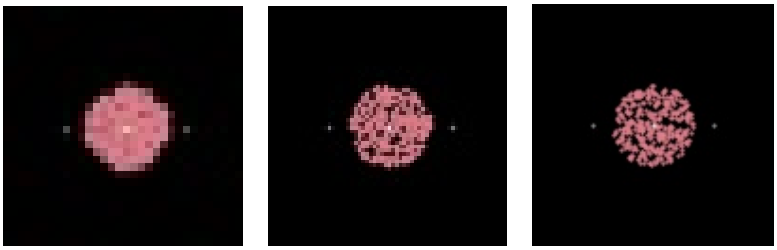
While training the CNN, we picked cross-entropy loss as the loss criterion and stochastic gradient descent (introduced in [54]) as the optimiser.

3.2.4 CNN Experiments

We carried out three experiments, each with different image shapes. Having three trained CNNs allowed for comparisons in terms of performance, computational complexity and image resolution. The three CNNs came from different choices for the image dimensions d, while creating the CNN dataset, with $d = \{32, 64, 128\}$. The produced RGB images have three channels, and therefore, they have shape $(3, d, d)$.

Input data with smaller dimensions speeds up training. On the other hand, increasing the dimensions provides higher resolution images. Stimuli images with dimensions 32, 64

and 128 respectively are shown in Figure 3.5, illustrating this difference in resolution. We trained the CNN for the three cases and measured the performance.



(a) 32 x 32 (b) 64 x 64 (c) 128 x 128

Figure 3.5: Different Resolution Stimuli Images
Showing stimulus images of an example colour 90° for three different image shapes (32 x 32, 64 x 64 and 128 x 128). Visualising the different resolution of these different image sizes.

3.2.5 CNN Fine Tuning

The CNN we trained is a modification of the VGG-16 pre-trained model described in [51] and explained in Section 2.3.2. We loaded the VGG-16 model and froze some of its weights so that they will not change during training. More explicitly, only the parameters of the linear layers were updated in the training loop. The VGG-16 has kernel size 3, stride 1 and padding also 1.

In our experiments, the input dimensions of the first convolution layer was $(3, d, d)$ with $d = \{32, 64, 128\}$. Therefore, we modified the pre-trained model accordingly.

Modifying the VGG-16, as illustrated in Figure 3.6, required an understanding of the layers' operations. Regarding the convolution operation, if the input of the 2D convolution ("conv2d") has shape $(C_{in}, d_1^{in}, d_2^{in})$ then the output has shape $(C_{out}, d_1^{out}, d_2^{out})$, with d_1^{out} and d_2^{out} according to Equation 3.1 and C_{out} the second input argument of "conv2d". Therefore, the output of the first convolution layer (and so input of the second one) was $(64, d, d)$. The "maxpool2d" layers have kernel of size 2, padding 0 and stride 2. The output dimensions of a max pooling operation are given again according Equation 3.1, where the parameters take these new values specific to the "maxpool2d". So, for the first max pool layer, the input dimensions were $(64, d, d)$ and the output dimensions $(64, d/2, d/2)$.

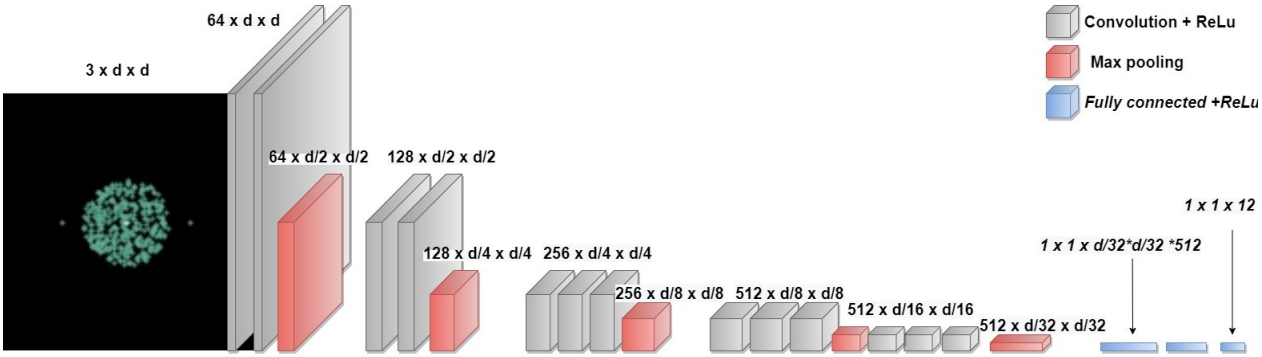


Figure 3.6: VGG-16 Modification
Visualising the VGG-16 architecture (13 convolution and 3 linear layers), along with the appropriate modification of the model according to the input shapes of our experiments.

$$
\begin{aligned}
d_1^{out} &= \frac{d_1^{in} - (kernel\_size[0] - 1) + 2\text{x}(padding[0])}{stride[0]} + 1 \\
d_2^{out} &= \frac{d_2^{in} - (kernel\_size[1] - 1) + 2\text{x}(padding[1])}{stride[1]} + 1
\end{aligned} \tag{3.1}
$$

The output of the last max pool layer had dimensions $(512, d/32, d/32)$. Therefore, the first linear layer had $\frac{d}{32} * \frac{d}{32} * 512$ input nodes. Since there were twelve classes, the number of output nodes was twelve. The choice of the number of nodes for the intermediate layers is shown in Table 3.1.

	Input size	Layer 1 output size	Layer 2 output size	Layer 3 output size
CNN128x128	8192	2000	400	12
CNN64x64	2048	200	40	12
CNN32x32	512	60	30	12

Table 3.1: Linear Layers Input and Output Sizes (CNN)

3.2.6 CNN Hyper-parameters

The hyper-parameters we considered for the CNN model, also explained in Section 2.4.1, were the batch size, the learning rate and the number of epochs. Since the focus of this study is the LSTM model architecture, the hyper-parameter values were chosen using

brute-force, and further optimisation was not deemed necessary. As shown by the results in Section 4.1, the performance of the model using these hyper-parameters was sufficiently good, discouraging additional analysis. The final hyper-parameter values used while training the CNN, are given in Table 3.2.

Parameter Name	Value
Batch Size	4
Num. Epochs	15
Learning Rate	1e-3
Criterion	Cross Entropy Loss
Optimiser	SGD

Table 3.2: Hyper-parameters (CNN)

3.3 LSTM

The trained CNN was used to produce activations which were inputted into an LSTM. We developed a bidirectional LSTM which we trained to distinguish two tasks, namely motion and colour. Then based on the indicated task, the model classified either colour into red or green (or yellow) or motion into upwards or downwards (or horizontal). The LSTM activations were used to create distance matrices constituting an engineered dissimilarity measure and utilised to elucidate the nature of the monkey brain dissimilarity.

3.3.1 LSTM Dataset

The LSTM training set consisted of the activations obtained when passing sequences of ordered images, resembling the sessions in the original experiment, through the trained CNN. In this section, we explain the dataset which includes these image sequences. The processing of these sequences, prior to inputting them into the LSTM, is described in Section 3.3.2.

This dataset represented the sequence of images shown to the monkeys in the original experiment, with each sequence of images being one trial. Each trial was constructed by choosing one out of the four possible cues, one out of two possible dot speeds and one out of the 21 possible combinations of colours and directions. These colour-motion combinations comprising the stimulus space are shown in Figure 3.7.

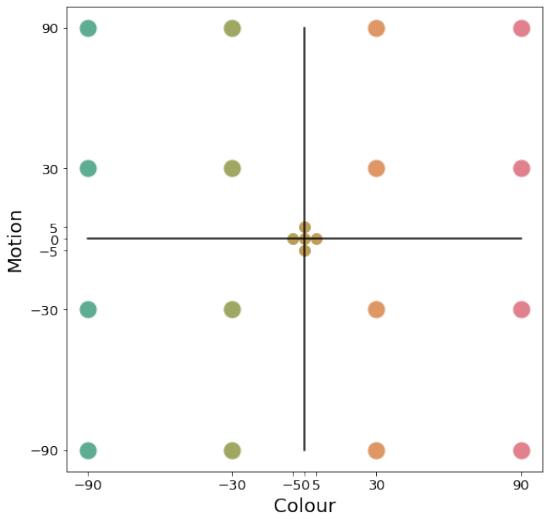


Figure 3.7: Stimulus Space

Visualising the 21 colour and motion combinations in the form of 2D points. The black straight lines represent the category boundaries for colour and motion. The distances between the trained LSTM activations of all pairwise combinations are used to create the distance matrices later on.

We chose a frequency of 60 frames per second. In the LSTM, this is translated as the number of images shown for each type of image (fixation point, cue and stimulus). More precisely, the fixation point image was shown for 0.5s in the original experiment, and therefore, there were 30 images of the fixation point in each trial. Accordingly, there were 60 images of the cue chosen for each trial (corresponding to 1s) and 180 images of the stimulus (corresponding to 3s). So, for an individual trial, there were a total of 270 images with the 180 stimulus images representing both colour and motion. The first stimulus image was an image of 400 random dots within a circle, with specific dot colour for that particular trial. The images that came next represented the chosen direction of motion since each dot moved towards that specified direction.

The speed of the dots was specified to be either "slow" or "fast". This speed corresponded to the amount of displacement (horizontally and vertically) of the dots between two successive frames, with every dot moving towards the specified motion direction. In half trials, we specified the speed choice to be "slow" and in the other half to be "fast".

The response for each trial depended on the cue, the dot colour and the dot motion. If the cue was either cross or quatrefoil, then the task of LSTM was to classify whether the random dots are moving upwards or downwards (or horizontally). Otherwise, the focus

was shifted on classifying the colour into red or green (or yellow).

If the random dots were moving in directions $\{-90°, -30°\}$, then this was considered downward motion, and if they were moving in directions $\{90°, 30°\}$, it was considered upward motion. Similarly, if the dot colour was labelled $\{90°, 30°\}$ and $\{-90°, -30°\}$ it was considered red and green respectively. So in the case of the cross or quatrefoil cue, the response was 0 if the random dots were considered to be moving upwards and 1 if they were moving downwards. Similarly, if the cue was a circle or a triangle, the response was 0 if the random dots were considered to be green and 1 if they were red. Figure 3.8 illustrates the above.

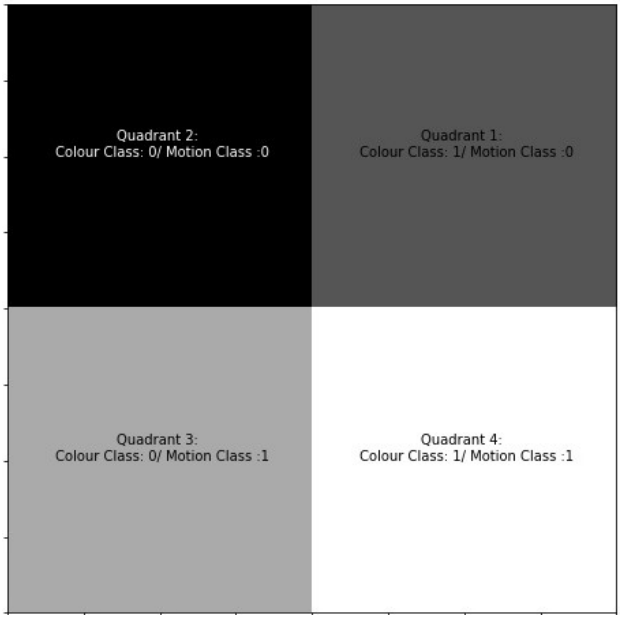


Figure 3.8: Response Space (LSTM)
Showing the correct classification of the points in Figure 3.7, belonging to each quadrant. There are two classes and the correct response depends on the indicated task of the particular trial. The boundary points are classified as class 2.

There were 16 stimulus points which fall into one of the four quadrants and five additional points which fall onto the category boundaries, as can be seen in Figure 3.7. The points on the boundaries corresponded to class 2 (neither 0 nor 1), regardless of the cue shape. This extra class was not an option for the monkeys in the original experiment. However, the focus was on the results excluding the points on the category boundary. Therefore, we added an extra class in order to shift importance away from these five points.

The set was constructed by creating 30 trials for each combination of speed (2 possible

choices), cue (4 choices) and stimulus (21 choices). Therefore, the size of the resulting dataset was $2 * 4 * 21 * 30 = 5040$.

3.3.2 LSTM Pre-processing

The CNN processed the dataset described in Section 3.3.1 to create the LSTM training set. Each of the 270 images from each trial in the dataset, was sequentially passed through the trained CNN. For each image inputted into the CNN, we obtained the activations from the second to last layer of the model. This way, we reduced the dimensionality of the data and so the dimensions of each trial changed from $(270, 3, d, d)$ to $(270, m)$, where m is the size of the activations of the second to last layer of the CNN. In the experiment with image shape $(3, 128, 128)$, the dimensions for each trial were $(270, 400)$. This comprised the dataset used for the training of the LSTM.

The responses corresponding to each training example (trial) were according to Section 3.3.1. Class 2 had a fewer number of examples compared to classes 0 and 1. This imbalance in classes was not alarming since there was no massive difference and class 2 was not of interest when considering the scope of this study.

3.3.3 LSTM evaluation

The dataset had shape $(5040, 270, 400)$. This number came from the fact that there were 21 combinations of colour and motion, four possible cue shapes and two possible choices for the dot speed. These gave 168 $(21 * 4 * 2)$ different training sequence types. Each type was repeated 30 times, each time with random initialisation of the dots' position, yielding a total of 5040 training examples. The training examples were then split into training and validation sets, with the latter comprising 10% of the total data. Likewise, we created the test set with 10 repetitions of each of the 168 data types, and therefore the test set size is 1680.

Similarly to the CNN evaluation in Section 3.2.3, we kept track of the performance of the model using both a validation and a test set. The validation set was used in the hyper-parameter optimisation process. We compared the validation losses, accuracies and F1 scores obtained while training the LSTM, for a range of hyper-parameter values. The values giving the best results were chosen to be used in the final LSTM. At testing time, the trained LSTM was used on unseen data. Its results were utilised when constructing the distance matrices.

Upon learning the weights of the model, we set the loss criterion to be the cross-entropy loss and the optimiser to be the Adam algorithm (proposed in [29]).

3.3.4 LSTM experiments

We conducted two types of experiments regarding the LSTM. In particular, we trained an LSTM on the dataset which only contained stimuli images of different motions to verify whether the motion of the random dots was clearly represented and understood by the model, as well as to discover which motion directions arwere misclassified most frequently, by observing the confusion matrix. This was an explanatory experiment and it is not discussed any further throughout this study. Furthermore, we trained the LSTM on the full data for different sequence lengths which correspond to different response times. This is the main LSTM experiment we focus on in Section 4.2.

In our attempt to find the LSTM architecture describing the monkey data best, we conducted experiments for different response times. As discussed in Section 2.2.1, the monkeys had to give their response within an interval of 3s, which corresponds to 180 image frames (since the chosen frequency is 60 frames per second). We decided to sample response times in this 3s window. We trained the LSTM using dataset dimensions $(5040, seq\_len, 400)$ for several seq_len values. The parameter seq_len denotes the number of image frames including the frames representing the fixation point and the cue. Specifically, we investigated sequence lengths $seq\_len = \{120, 160, 200, 230, 250, 270\}$, which are equivalent to $\{30, 70, 110, 140, 160, 180\}$ stimuli images since the first 90 frames were fixation point and cue images. The respective response times in seconds are $\{0.5, 1.167, 1.833, 2.333, 2.667, 3\}$.

Additionally, prior to conducting these experiments, we trained a simple LSTM to classify the motion of the sequential images as one of the seven discussed motion directions. The results of this seven-class categorisation enabled us to examine whether the LSTM could differentiate those seven motions without having to pay attention to cues and colours. So, instead of training on the full dataset with dimensions $(5040, 270, 400)$, we decided to train this simpler LSTM on a smaller dataset only containing stimuli images of different motions. The input images were reshaped into a 1D tensor in order to be fed into the LSTM. The produced confusion matrix is found in Appendix A. It shows that the LSTM was competent in discriminating motion, reaching 88.80% training accuracy. This LSTM mostly confused motion directions $\{-5°, 0°, 5°\}$, which is what we expect from primates as well. Since the results of the simple LSTM were sensible, we were confident that motion was represented clearly in the sequence of images and so the proper LSTM would most likely

be capable of differentiating motion appropriately. The simple LSTM is an explanatory model and will not be further discussed.

3.3.5 LSTM model

For this classification task, we trained a bidirectional LSTM. The input parameters of such model are of the form $(input\_size, hidden\_size, num\_layers, num\_classes)$. Since each trial of our input data was a sequence of one-dimensional tensors of size 400, then the $input\_size$ was 400. We chose a hidden size of 500 and num_layers to be two. Lastly, it followed from the experimental setup that there were three classes (0,1 and 2). Because our model is bidirectional, h_0 (initial hidden state) and c_0 (initial cell state) had shapes $(num\_layers * 2, batch\_size, hidden\_size)$ each.

The outputs are of the form $output, (h_n, c_n)$. The last hidden and cell states (h_n, c_n) have dimensions analogous to h_0 and c_0.

Then, there were two fully connected layers with a ReLu activation function before the first layer. The output of the model was the output of the last linear layer, which had size three and corresponded to the prediction probabilities for each class. For the construction of the distance matrices during testing, the activations of the first linear layer were used. The shape of each activation tensor was 250.

3.3.6 LSTM hyper-parameters

The LSTM was trained using a range of values for its hyper-parameters, in order to find those for which the model performed best. The hyper-parameters we considered are the batch size, the learning rate and the number of epochs. For this process, we used the full sequence of images with length 270 and we documented validation loss, accuracy and F1 score. The hyper-parameter values chosen after several trials, results of which are presented in Section 4.2, are displayed in Table 3.3.

Parameter Name	Value
Batch Size	12
Num. Epochs	15
Learning Rate	1e-3
Criterion	Cross Entropy Loss
Optimiser	Adam

Table 3.3: Hyper-parameters (LSTM)

3.4 Comparisons

We created a neural network (CNN-LSTM) capable of performing the described categorisation task. Performing a series of comparisons between quantities of the neural network activations, we sought to discover certain attentional effects like dimensional stretching or contraction. Additionally, comparing the neural network's activations with the firing rates of neurons in several areas of the rhesus monkey brain, we sought to evaluate the dissimilarity functions and representations in neural data. In this section, we describe how such comparisons were made.

3.4.1 ANN data

Comparisons between the produced distances matrices under the colour and motion conditions were made in order to obtain stretching ratios and discover dimensional stretching and contraction. Statistical t-tests were used to check for statistically significant stretching along the appropriate dimensions. This section explains the distance matrices, states the experiments and gives an overview of t-tests.

Distance matrix

The distance matrices were created from the LSTM activations derived at testing time, under the colour and motion conditions separately. The activations for each colour-motion combination were recorded and then the Euclidean distance between the pairwise averages, corresponding to the 16 combinations of interest, was calculated. Therefore, each of the two distance matrices had shape $(16, 16)$. One contained the distances obtained from examples when the task was motion classification and the other when the task was colour classification.

In more detail, the testing examples were passed through the trained LSTM, and the activation tensor for each one was recorded. For the examples with cues indicating the motion classification task (motion condition), the activation tensors for each combination were averaged. The five points on the category boundary are not of interest, and so they were excluded. Therefore, this gave 16 tensors, instead of 21, denoting the average activation values. Then, we calculated the pairwise Euclidean distance between all of 16 combinations, giving a 16 by 16 distance matrix. Similarly, we computed the distance matrix under the colour condition.

The matrices are symmetric and have zeros in the diagonals. Symmetric because the Euclidean distance between a and b is equal to the one between b and a. The diagonal elements are zero since the distance between a and a is zero. Also, in this case, distance matrix is a synonym of dissimilarity matrix.

Experiments

Having those matrices, we performed a series of comparisons using statistical tests. In the beginning, we compared them with each other in order to unveil stretching or contraction of dimension (Figure 3.9).

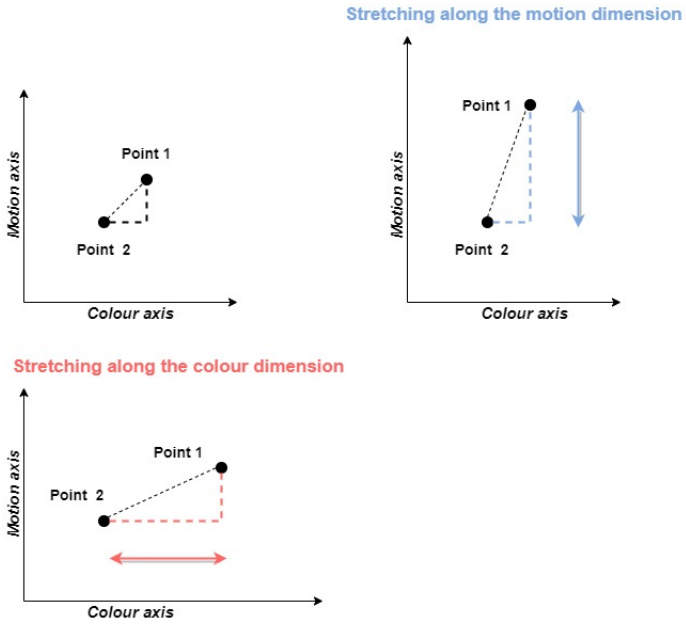


Figure 3.9: Dimensional Stretching Example
Illustrating the notion of dimensional stretching, with example stretching ratios along the colour and motion dimensions.

In particular, we conducted two types of experiments, namely "within conditions" and "between conditions".

"Within conditions" refers to comparing quantities computed using a single distance matrix, while "between conditions" refers to comparing quantities calculated using values from both matrices.

Then we compared the neural network distance matrices with the empirical data from the monkey brain, attempting to correlate the two.

Statistical *t*-test

We used statistical t-tests in order to draw inference from the ANN data. Those tests come from the Student's t distribution. The two-sample t-test is used to compare equality between two population means and the one-sample t-test to compare a population mean with a fixed value.

Depending on the alternative hypothesis, the t-test could be either one-sided or two-sided. The former refers to an alternative hypothesis with a strict inequality sign $(>, <)$, whereas the latter refers to a not equal sign $(\neq)$.

We performed such tests by computing the test statistic along with the corresponding degrees of freedom in order to obtain the p-value. Having the p-value, we either accepted or rejected the null hypothesis according to Table 3.4. More details about the t-test can be found in Appendix B.

p-value Range	**evidence Type**
p-value <= 0.01	Strong evidence against H_0
0.01 < p-value <= 0.05	Evidence against H_0
0.05 < p-value <= 0.01	Weak evidence against H_0
p-value > 0.1	No evidence against H_0

Table 3.4: P-value Significance Chart

3.4.2 Empirical Data

In this section, we provide details of the empirical data along with the empirical data dissimilarity measures we use in this study. These measures were correlated with dissimilarity measures from three models, namely the neural network and two baselines, using the Spearman's rank test. Such correlation coefficients reveal whether representations change across tasks and brain regions and also serve as a model and empirical measure selection process.

During the original experiment, the parameters recorded included which monkey participated, the task (colour/motion), the lobe and region of the brain and the corresponding neuronal information in spiking activity. There were two monkeys involved in the experiment, one female (Paula) and one male (Rex). The cued task was colour in half the trials and motion in the other half. There were three recorded brain lobes (frontal, parietal and temporal) and six brain regions (PFC, FEF, LIP, IT, MT and V4). The histogram, corresponding to the brain regions, indicated that there were eight categories instead of

six since some neurons were not localised in a specific region and so they were labelled with the brain lobe. Those data points were excluded from the dataset when comparing between brain regions.

The raw spikes from the original data were sorted and provided by Earl Miller's Lab at MIT and then Dr. Sebastian Bobadilla of Love Lab pre-processed that data to get the distances for this project. The analysed neural data included the pairwise distances of all the colour-motion pairs. The dataset consisted of columns with the values of the three distance functions. For the comparisons with the ANN data, we use the distances measured with the ISI-distance function and SPIKE-distance function, denoted by isi_full and $spike\_full$ respectively. The "$\_full$" refers to a 200ms window after random dot onset, meaning that both monkeys had enough information to make a decision within that window for most trials.

Pre-processing

Prior to using the empirical data for comparisons, we found that 60 values of the isi_full distance and 10465 values of the $spike\_full$ distance were "nan" and so they were removed. Moreover, the dataset was split into two smaller sets, one corresponding to the colour condition and the other to the motion condition.

Experiments

The dataset was used to construct another set of distance matrices which come from the monkey data. There were two such matrices for each function, each one representing the distances obtained either under the colour or motion condition. We compared these matrices to their counterparts from the LSTM and baseline models. We quantified these comparisons with the Spearman's rank test, which gave the correlation coefficients between the data.

Spearman's Rank Test

The Spearman's rank test is a non-parametric test which checks for an association between two sets of data. This is done by ranking the data and computing a statistic using these ranks. This test is suitable in our case, since the distances in the datasets we compared, have very different scaling.

The null hypothesis accompanying this test is that there is no association between

the two inputs. The test's output is a correlation coefficient between -1 and 1, and the p-value. The closer the coefficient is to 0, the less association there is. On the contrary, coefficient values close to the endpoints indicate either a positive or negative correlation. Although there are a few ways for deciding whether to reject or accept the null hypothesis, we computed p-values and use them according to Table 3.4.

Chapter 4

Results and Evaluation

In this chapter, we present results of the CNN and LSTM experiments as well as the results from the comparisons. The CNN and LSTM experiments acted as preliminaries in order to choose the final CNN-LSTM based model. In particular, we evaluated the performance of three CNNs with different input image shapes. We chose the CNN128x128 to be used for the LSTM inputs. We also compared LSTM models with different sequence lengths and chose the one corresponding to the model with the best results ($seq\_length = 250$). Altogether, the neural network (CNN-LSTM) reached very high accuracy, effectively implementing the task at hand. Having this neural network, along with the accompanying colour and motion distance matrices, we performed a series of comparisons and drew inference relating back to the goal of this study.

Firstly, we present stretching ratios and draw conclusions on dimensional stretching, discovering significant stretching along the appropriate dimensions. Secondly, we compared the dissimilarity measures from the neural network with the empirical data measures from the six brain regions and the two tasks, discovering that representations change across regions and tasks. Finally, we compared the dissimilarity measures obtained from the empirical data (ISI-distance and SPIKE-distance) with the measures from three models (neural network and two baselines). Statistical results reveal which model best describes the empirical data in each case (second baseline and neural network for ISI and SPIKE distance respectively), characterising the distance function that the monkeys are using.

4.1 CNN

In this section, we exhibit the results of the experiments described in Section 3.2.4, regarding different image shapes. Also we analysed the explanatory figures, collectively leading to the selection of the CNN model, which was combined with the best-performing LSTM and produced the distance matrices, providing a deeper understanding of the model behaviour. We highlight aspects of such behaviour which are similar to the functions of the monkey brain.

The performance of the three CNN experiments is shown in Table 4.1 and Table 4.2. The values in the tables are shown in four significant figures (4 s.f.) for both the cross-entropy loss and the accuracy.

	Training Loss	Validation Loss	Test Loss
CNN128x128	0.3232	0.3804	0.4041
CNN64x64	0.3982	0.4128	0.4044
CNN32x32	**0.2965**	0.2735	0.2992

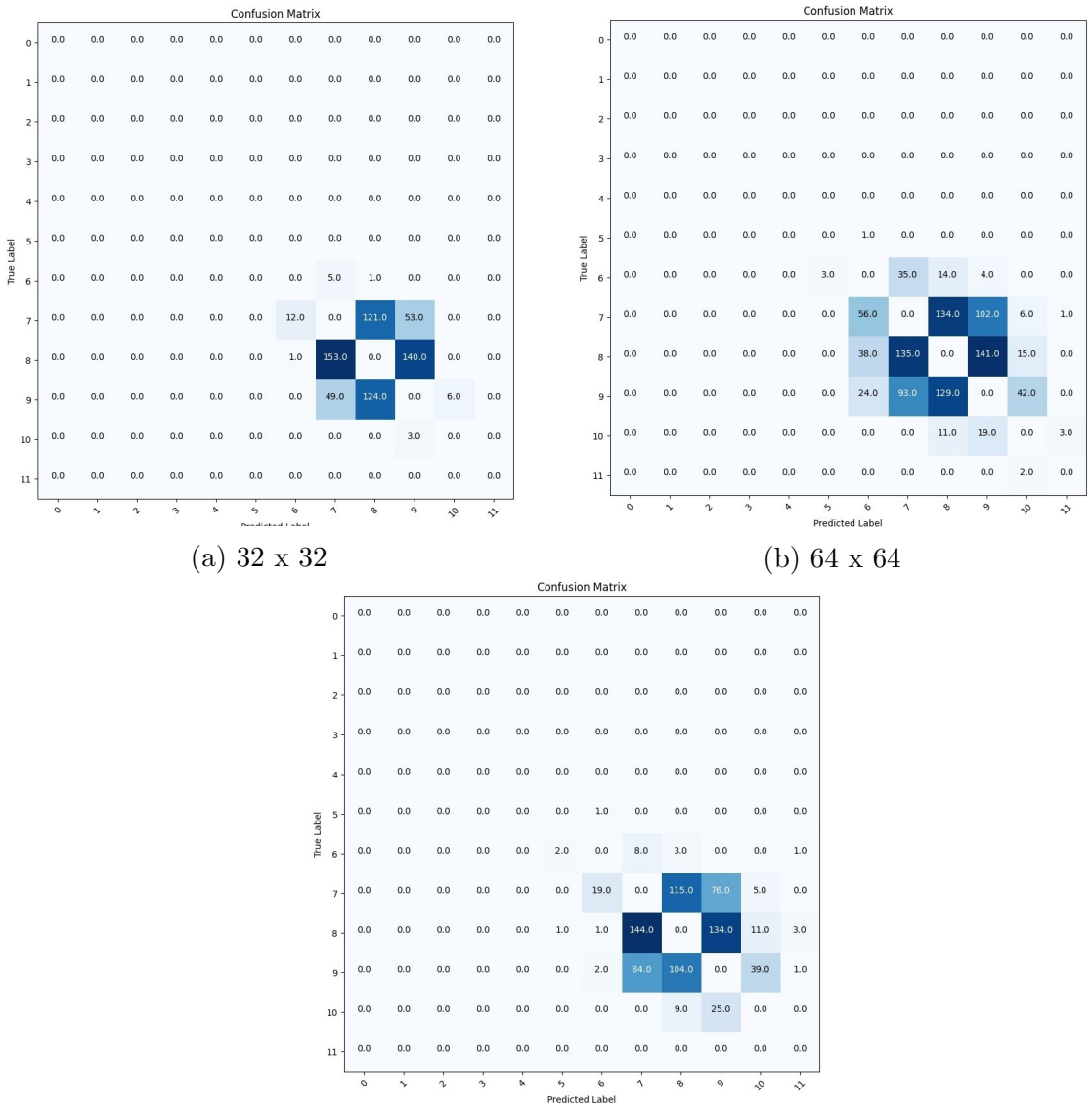
Table 4.1: CNN Losses for the three Image Shapes

	Training Accuracy	Validation Accuracy	Test Accuracy
CNN128x128	85.41%	80.67 %	81.58 %
CNN64x64	81.33 %	82.33 %	81.53 %
CNN32x32	**87.63 %**	89.17 %	88.97 %

Table 4.2: CNN Accuracies for the three Image Shapes

All three CNNs achieved high performance with accuracies exceeding 80%. So we can deduce that regardless of the input image dimensions, the CNN was competent in differentiating the images of the fixation point, cross, quatrefoil, circle, triangle and different coloured stimuli images. From the results, we can see that the model with image dimensions $(3, 32, 32)$ (i.e. "CNN32x32") gave the highest accuracy and the smallest loss. The second best performing model was "CNN128x128" and the worst was "CNN64x64". The time it took for each epoch to be executed while training, was roughly 1.5 minutes for "CNN128x128", 0.37 minutes for "CNN64x64" and 0.28 minutes for "CNN32x32". Although the models' performance was not the same throughout the three experiments, the differences laid in slight increase or decrease in the number of mistakes made while classifying colours.

The confusion matrices in Figure 4.1, constructed while training the different CNNs, reveal the patterns of mistakes of each CNN.



(a) 32 x 32

(b) 64 x 64

(c) 128 x 128

Figure 4.1: Confusion matrices of CNN32x32, CNN64x64 and CNN128x128
Visualising the confusion matrices of the CNN experiments. The confusion matrix is a way of studying the performance of the model in more depth, providing insight into the mistakes that the model makes. The value at the i^{th} row and j^{th} column of the confusion matrix, is equal to the number of times the model wrongly predicted that an input belongs to class j instead of the correct class i. The bottom right values refer to confusions between the colour classifications. Larger numbers are visualised with a darker shade of blue.

From Figure 4.1, we observe that all CNNs exhibited roughly the same pattern, eventually making no mistakes while classifying fixation point and cue images. Conversely, the CNNs mostly confused the colours corresponding to the category boundary with labels $\{-5°, 0°, 5°\}$. This is a significant finding suggesting a similarity between the behaviour of the model and the monkey, since it is rational to confuse such similar colours.

Consistent with the results, we expected "CNN32x32" to be the best performing model, given that the resolution of the image was lower and therefore there were more coloured pixels. So, the model would more easily differentiate the seven colours. This difference in resolution between the three image sizes can be seen in Figure 3.5. However, we decided to proceed using "CNN128x128" taking into consideration the insignificant difference in performance values. This decision stemmed from our effort to mimic the images shown to the monkeys, with the $(3, 128, 128)$ images being more similar to the those in the original experiment. At the same time, their higher resolution would intuitively be beneficial when determining the motion direction in the LSTM. Higher resolution implies clearer dot positions, defining the displacement of each dot through the sequences of images more accurately.

The learning curves for the training and validation loss and accuracy of the chosen model are presented as a function of epochs in Figure 4.2. In both graphs, we can see that training had a lower loss and higher accuracy than validation, with the training curve being smoother. In general, the loss and accuracy followed the same patterns for training and validation, gradually improving the performance.

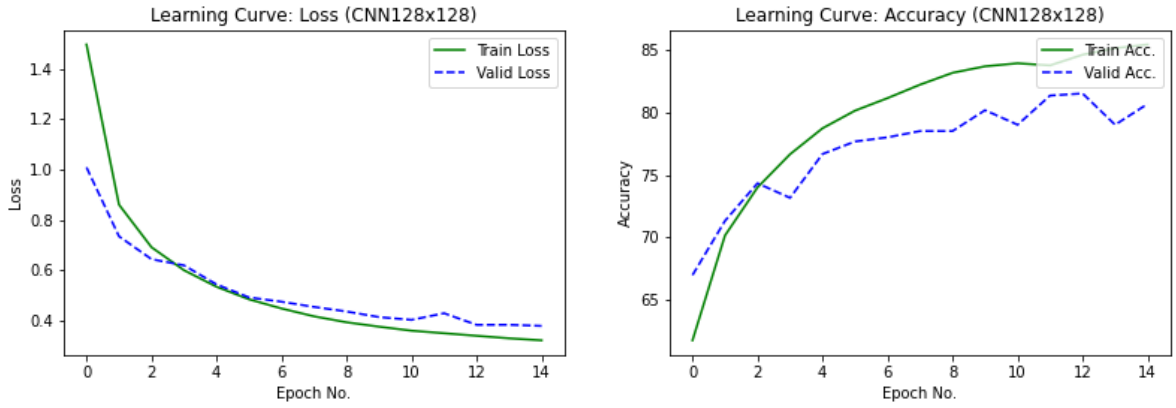
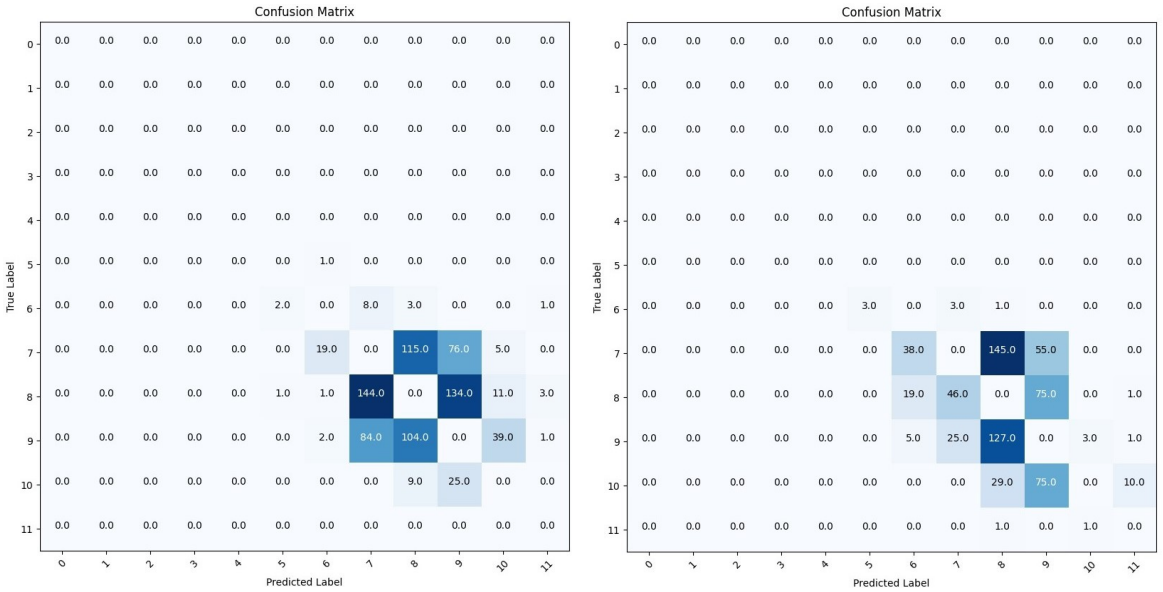


Figure 4.2: Learning curves (CNN)
Graphs of the training and validation loss (left) and accuracy (right) of the "CNN128x128" against the number of epochs.

We created testing confusion matrices and confusion tables from the "CNN128x128" outputs, to understand the model's limitations. The confusion matrices produced from the training and test data are shown in Figure 4.3. From the confusion matrices displayed in Figure 4.3, we notice that the classes with considerably more mistakes are classes 7, 8 and 9, which correspond to dot colours labelled $\{-5°, 0°, 5°\}$. A visualisation of those colours can be found in Figure 3.3.

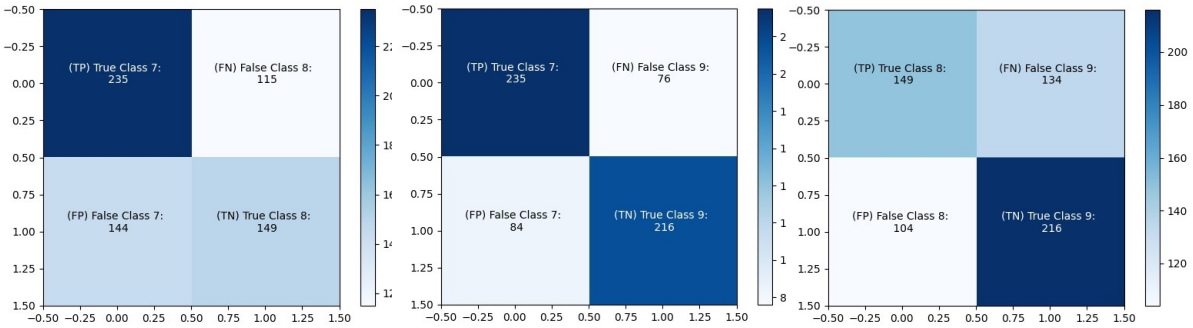


(a) Confusion Matrix (Training) (b) Confusion Matrix (Testing)

Figure 4.3: Confusion matrices (CNN)
Training and testing confusion matrices of "CNN128x128". Showing the number of mistakes at training and testing.

Figure 4.4 shows the tables of confusion for classes 7 and 8, classes 7 and 9 and classes 8 and 9, respectively. The first quadrant of each table presents the number of times the first class was misclassified as the second (false "negatives"). This is known as Type II error. Quadrant 3 represents Type I error, displaying the number of times the second class was misclassified as the first (false "positives"). Therefore, the false-negative rates are $\frac{23}{70}, \frac{76}{311}\&\frac{134}{283}$ and the false-positive rates are $\frac{144}{293}, \frac{7}{25}\&\frac{13}{40}$.

The results presented in Figure 4.3 and Figure 4.4 helped to understand how the model works. The main flaw of the trained CNN lies in its inability to correctly classify classes 7,8 and 9 with high accuracy. However, when considering the LSTM, all three belong to the same class (class 2), by design. Thus, there is no reason for further optimising the CNN.



(a) Classes: 7 & 8 (b) Classes: 7 & 9 (c) Classes: 8 & 9

Figure 4.4: Tables of Confusion
Visualising the amount of correct and incorrect classifications in terms of False Positives, False Negatives, True Positives and True Negatives, for each pairwise combinations of classes 7,8 and 9 (which correspond to colour labels $-5°, 0°$ and $5°$).

This trained CNN, achieving the above results, was used to obtain the LSTM input data. The data was generated by inputting images into the CNN and obtaining the corresponding activations of the second to last layer. Each training example of the LSTM was a sequence of activation tensors of specified length, representing the series of images shown to the monkeys.

4.2 LSTM

In this section, we present the results of the trials for the hyper-parameter selection along with the results regarding the LSTM experiments with different sequence lengths (response times). The best performing LSTM combined with "CNN128X128" produced the distance matrices of the neural network, which we compared with each other and with the empirical data, characterising the monkey brain dissimilarity measure.

The first step towards creating this LSTM architecture was to tune the hyper-parameters. We carried out several trials and show the relevant results in Table 4.3. For the batch size, we trained the LSTM using six epochs and $1e-3$ learning rate and changing the batch size to $4, 8$ and 12. From Table 4.3, we decided to use a batch size of 12 since it had a lower loss, higher accuracy and larger F1 score. Accordingly, we tried two values for the learning rate (0.001 and 0.0001) and re-trained the LSTM while keeping the epoch number and the batch size constant to 6 and 12, respectively. The best choice for this parameter was 0.001. Finally, using batch size 12 and learning rate 0.001, we trained the model for 15 epochs and reported the performance for every five epochs.

Batch size	**batch_size= 4**	**batch_size= 8**	**batch_size= 12**
Validation Loss	0.01480	0.02811	**0.001736**
Validation Accuracy	99.80 %	99.01 %	**100.0** %
Validation F1 score	0.9980	0.9900	**1.0**
Learning rate	**lr=0.0001**	**lr=0.001**	
Validation Loss	0.1309	**0.001736**	-
Validation Accuracy	96.83 %	**100.0** %	-
Validation F1 score	0.9694	**1.0**	-
Number of epochs	**n_epochs=5**	**n_epochs=10**	**n_epochs=15**
Validation Loss	0.08695	0.006558	**0.001169**
Validation Accuracy	97.42 %	99.80 %	**100.0** %
Validation F1 score	0.9739	0.9980	**1.0**

Table 4.3: Hyper-parameter tuning (LSTM)

It was also interesting to look at the produced learning curve since [3] suggested that the optimal number of epochs is the point of divergence of training and validation loss. Both Table 4.3 and Figure 4.5 indicate that the best choice was 15 epochs.

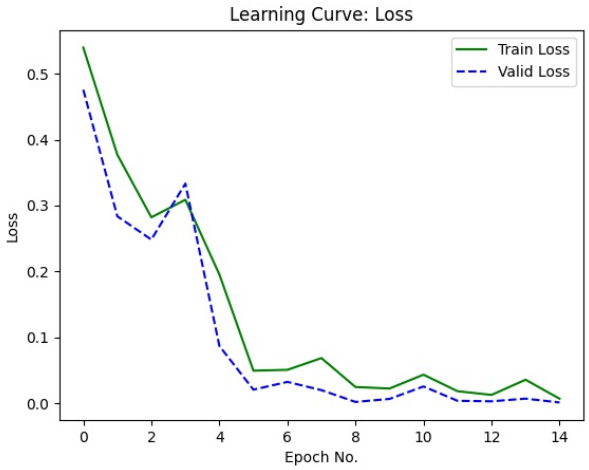


Figure 4.5: Training and Validation Loss
Plotting the training and validation loss of LSTM with batch size 12 and learning rate 0.001 as a function of epochs (total of 15 epochs).

In conclusion, the values for the three hyper-parameters, namely batch size, learning rate and number of epochs, used throughout the LSTM experiments were $batch\_size = 12$, $lr = 0.001$ and $num\_epochs = 15$.

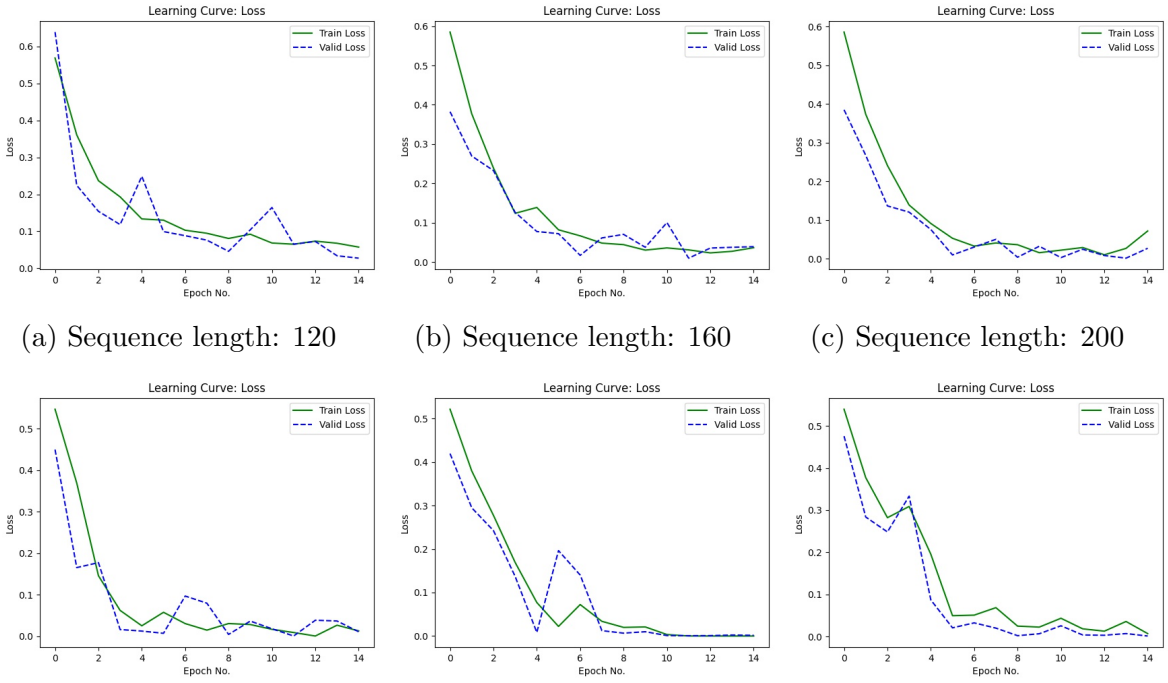
Following the hyper-parameter selection, we trained six LSTMs with the best hyper-parameters, conducting the experiments described in Section 3.3.4, regarding the different sequence lengths. The training and validation results of those experiments are presented in Table 4.4 (to 4 s.f.).

seq_len	120	160	200	230	250	270
Train. Loss	0.05713	0.03626	0.07151	0.01272	**3.437e-05**	0.006896
Valid. Loss	0.02718	0.03889	0.02672	0.01059	0.001649	**0.001169**
Train. Acc.	98.02 %	98.96%	97.42 %	99.65 %	**100.0 %**	99.85 %
Valid. Acc.	98.81%	98.81 %	99.21 %	99.60 %	**100.0 %**	**100.0 %**
Train. F1score	0.9802	0.9897	0.9744	0.9966	**1.0**	0.9983
Valid. F1score	0.9889	0.9879	0.9918	0.9959	**1.0**	**1.0**

Table 4.4: Losses, Accuracies & F1 scores (LSTM)

All models were successful in tackling this classification task, reaching accuracies higher than 95%. The LSTMs with sequence lengths 250 and 270 were the two best performing models. However, looking at the training loss, accuracy and F1 score, the best results were achieved using the first 250 frames of images (out of the full 270 which corresponds to the maximum response time of 3s). The differences in the performances were not large enough to conclude with certainty which point within the response window gives the best outcome. It could be the case that different set of hyper-parameter values could alter the results, increasing or decreasing some models' scores. Comparing with findings from the original experiment, the average response latency of the monkeys was 1 to 1.270s and the LSTMs with sequence lengths 120 and 160 correspond to response times 0.5s (below the interval) and 1.167s (within the interval) respectively. Both LSTMs manage to achieve high performance comparable to the monkeys' results (94% for the motion and 89% for the colour task).

Figure 4.6 shows the training and validation loss against the number of epochs for the experiments regarding the six sequence lengths. The losses of the LSTM with sequence length 250 were stabilised sooner than the rest (epoch 11), a result which is compatible with the findings in Table 4.4. For the other experiments, increasing or decreasing the number of epochs would lead to optimal losses. All things considered, we chose to proceed using the trained LSTM with sequence length of 250. Additional analysis of the chosen LSTM can be found in Appendix C, where we display the error matrices through the training process. During testing, the chosen model gave a loss of 0.0002288 (to 4 s.f) and an accuracy of 100%.



(a) Sequence length: 120 (b) Sequence length: 160 (c) Sequence length: 200

(d) Sequence length: 230 (e) Sequence length: 250 (f) Sequence length: 270

Figure 4.6: Learning Curves (LSTM)
Plotting the training and validation losses of LSTM with different sequence lengths (120, 160, 200, 230, 250 and 270) with batch size 12, learning rate 0.001 over the epoch number (15 epochs in total).

We derived the colour and motion distance matrices from the neural network. Each matrix was created by taking the Euclidean distance between the averaged activation tensors of the pairwise colour and motion combinations. We excluded combinations on the category boundaries, yielding 16 x 16 distance matrices.

From the distance matrix under the colour condition in Figure 4.7, we note that the top right and bottom left square held the larger distances (distance matrices are symmetric). Those values correspond to distances between points of positive colour labels (i.e. $\{90°, 30°\}$) and negative colour labels (i.e. $\{-90°, -30°\}$). The biggest is the distance between combinations $\{90°, -30°\}$ and $\{-90°, 30°\}$ representing colours $90°$ (red) and $-90°$ (green) respectively. These results are sensible because different sign for the colour is equivalent to different class labels under the colour condition and so we expected this distance matrix pattern. Observing this matrix suggests that, under the colour condition, there is stretching in the colour dimension, implying that the neural network reflects the dimensional stretching we expect to find in the monkey data.

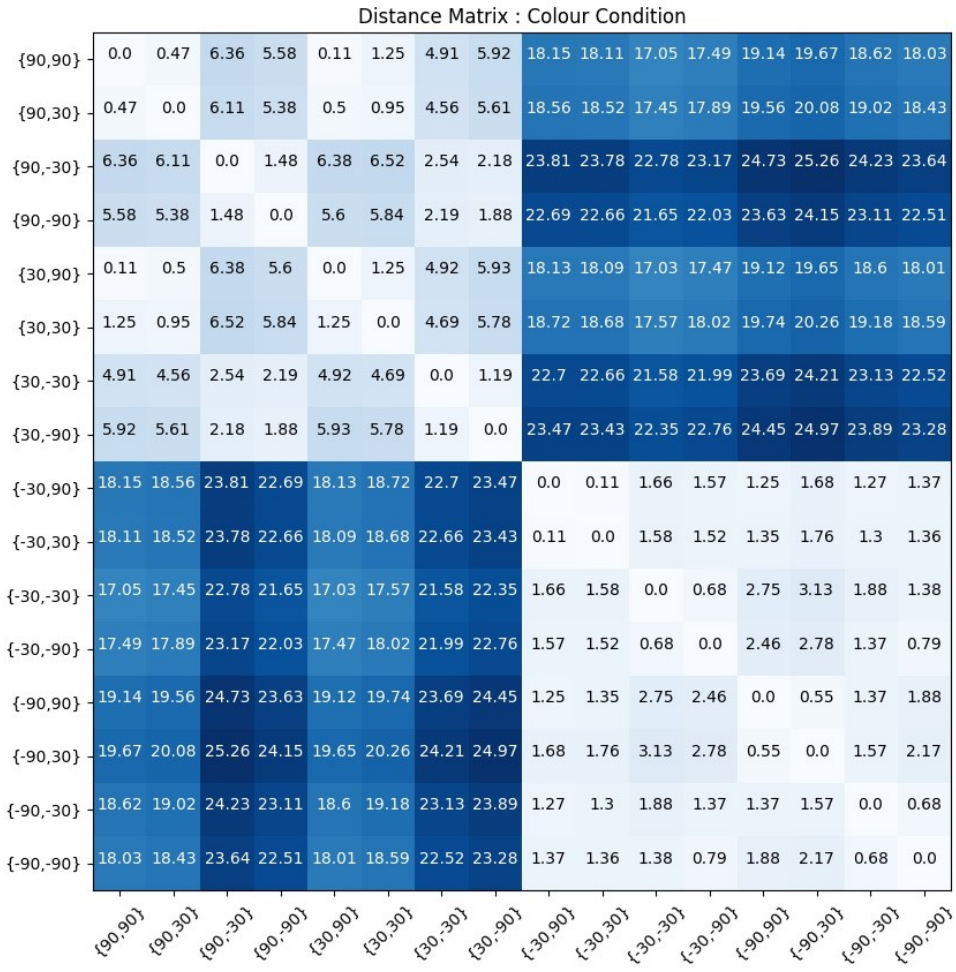


Figure 4.7: Distance matrix (task: colour)
Illustrating the 16 x 16 distance matrix derived from the activations of trials under the colour condition. Darker shades of blue indicate larger distances.

Similarly, we look at the distance matrix under the motion condition, shown in Figure 4.8. The largest distance is between combinations $\{90°, -90°\}$ and $\{-30°, 90°\}$ and between $\{30°, -90°\}$ and $\{-30°, 90°\}$, which correspond to motion directions $-90°$ (downwards) and $90°$ (upwards). In general, we deduce that the larger values are distances between combinations with positive and negative motion directions. Again, since different direction signs mean different classification under the motion condition, this is the matrix architecture we anticipated. Initial observation suggests stretching of the motion dimension, as we also expect from primates, giving a first indication that this neural network can be used to model brain data.

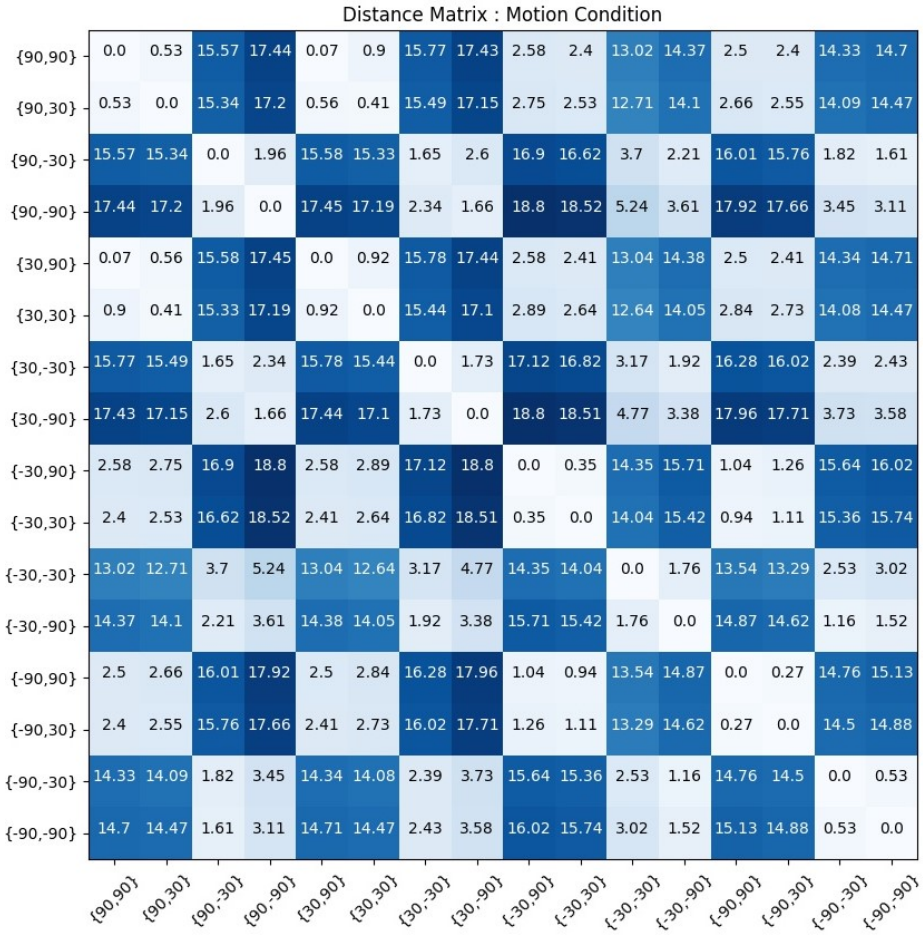


Figure 4.8: Distance matrix (task: motion)
Illustrating the 16 x 16 distance matrix derived from the activations of trials under the motion condition. Darker shades of blue indicate larger distances.

Additionally, further analysing the two matrices, we deduce that the distance matrix representing the colour task appeared to recognise changes in motion. This can be seen from the fact that the distances between combinations with different motion directions are larger than the distances between the same motion direction. Analogously, the motion distance matrix recognised changes in colour. In both cases, the matrices captured changes in the condition alternative to the one specified by the matrix, hinting that the model gave some attention to changes in both colour and motion, regardless of the specified condition. This model behaviour is sensible since the brain is able to attain this form of knowledge as well.

Concluding, the trained LSTM with sequence length 250, successfully implemented the

required categorisation task, showing signs of behaviour similar to the monkeys. Also, the outcome of distance matrices showed that representation changed as a function of task, in a similar way that we would expect from primates.

4.3 Comparisons

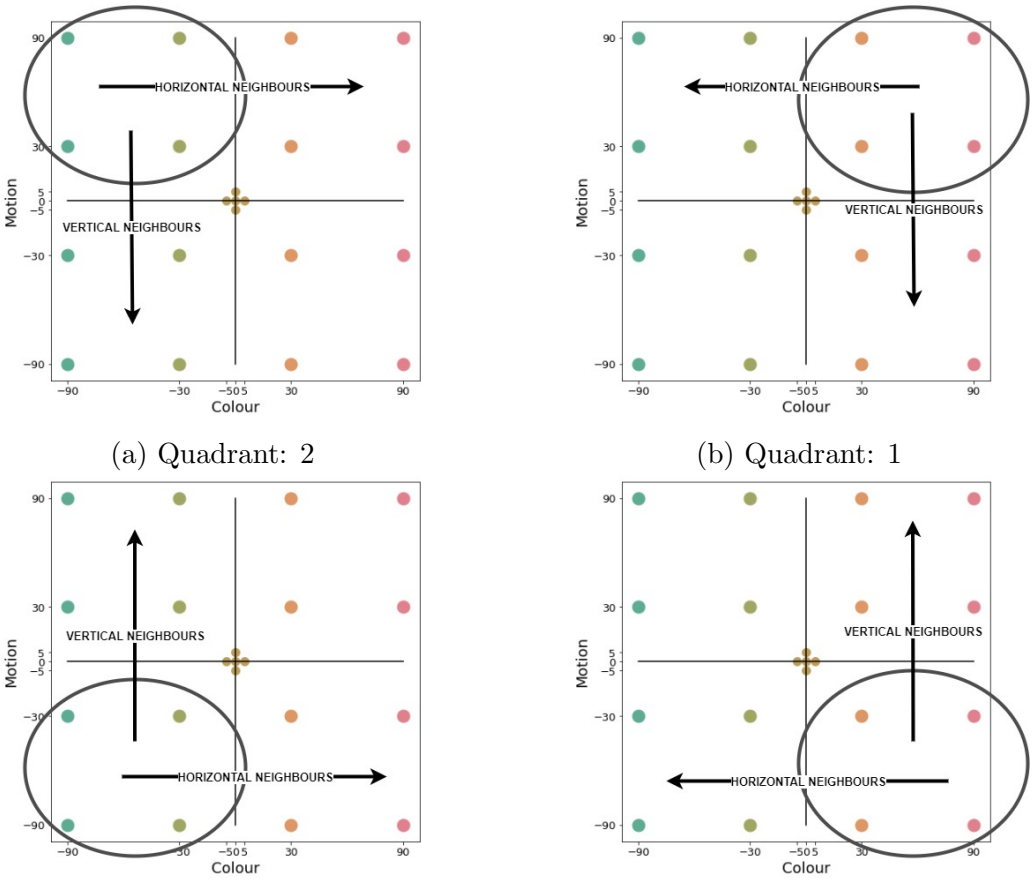
This section includes the main results used to infer the nature of the dissimilarity measure of the monkey brain. So far, we created a CNN and used it to develop an LSTM which was trained on a categorisation task, discriminating colour and motion. The activations of the resulting LSTM were used to form two distance matrices which provided an initial indication of dimensional stretching and association with monkey data. In this section, we present the comparisons made between the distance matrices to formally check for stretching or contraction of the colour and motion dimensions. Additionally, we show the correlation coefficients of the ANN data with the relevant empirical data, used to discover which distance function best reflects the monkey brain. The ANN data was further compared to the empirical data from each of the six brain regions. Although there is growing literature around the function of each region, the purpose of these results is to reveal whether representations change across regions without further analysis. Additionally, we correlated the empirical data with baseline models, which provides a baseline correlation coefficient. Comparing the empirical data with both the ANN data and baselines, we intended to shed light on the computational principles of the brain.

4.3.1 ANN data

Comparing the distance matrices with each other, computing stretching ratios and performing$t$-tests, we searched for stretching or contraction of dimensions within and between conditions.

In this paragraph, we explain the concept of neighbouring combinations which is important when investigating dimensional stretching. The 21 combinations were 2D points with the first dimension representing colour and the second representing motion (Figure 3.7). These 2D points took up a grid with lowest value -90 and highest value 90 in both dimensions. This grid was divided into four quadrants (Figure 3.8). Each quadrant corresponds to two classification values, one indicating the correct class for colour and one for motion. There were four points in each quadrant and five points on the category boundaries. For our analysis, we used the sixteen combinations inside the four quadrants and we excluded

the points on the boundaries. We introduce the notion of horizontal and vertical neighbours of a point. The horizontal neighbours of a point are the points belonging to the left or right adjoining quadrant, whereas the vertical neighbours are the points in the quadrant above or below. Figure 4.9 illustrates these neighbour types. Horizontal neighbours have different classification for the colour, whereas vertical neighbours have different motion classification.



(a) Quadrant: 2

(b) Quadrant: 1

(c) Quadrant: 3

(d) Quadrant: 4

Figure 4.9: Quadrant Neighbours
Diagram explaining the horizontal and vertical neighbouring points for each quadrant.

Within Conditions

We examined whether the LSTM stretched the colour or motion dimensions within conditions, by finding the ratio of the average distance of the horizontal and vertical neighbours of each combination.

Under the colour condition, we defined d_2 to be the distance between two horizontal neighbours (change of colour class) and d_1 the distance between two vertical neighbours (change of motion class). Similarly, defining d_2' and d_1' to denote the distances under the motion condition. All these distances were taken directly from the colour and motion distance matrices respectively (Figure 4.7 and Figure 4.8). Under the colour condition, we expected d_2 to be larger than d_1 (so the ratio $r = \frac{d_2}{d_1} > 1$), while under the motion condition we expected $r' = \frac{d_2'}{d_1'} < 1$.

We took all the points in quadrant i and found the distances d_2, d_1, d_2' and d_1' for all their horizontal and vertical neighbours. Then we computed the ratios r and r' and calculated the means ($mean_1$ and $mean_2$). We repeated this for all four quadrants. We present these stretching ratios computed from the ANN data to 4 d.p. in Table 4.5. As expected, $\bar{r} > 1$ and $\bar{r'} < 1$ implying that under the colour condition there is stretching along the colour dimension and under the motion condition along the motion dimension.

	$\bar{r}$	$\bar{r'}$
Quadrant 1	13.3783	0.2131
Quadrant 2	11.0432	0.1743
Quadrant 3	3.4158	0.1581
Quadrant 4	4.0857	0.1889
Average of Quadrants	7.9807	0.1836

Table 4.5: Stretching Ratios Within Conditions

Presenting the two types of ratios, namely $\bar{r}$ and $\bar{r'}$, computed from the ANN distance matrices, for each quadrant and overall. Quantity $\bar{r}$ denotes the ratio of change in colour over change in motion, under the colour condition and so is expected to be larger than 1. Quantity $\bar{r'}$ denotes the ratio of colour over motion change under the motion condition and so is expected to be lower than 1.

In order to test for statistically significant difference between the pairs of the computed means for each quadrant (Table 4.5), we used a two-sample two-sided t-test. The null and alternative hypothesis we tested is as follows.

$$H_0 : \bar{r_{Qi}} = \bar{r'_{Qi}} \text{ vs } H_1 : \bar{r_{Qi}} \neq \bar{r'_{Qi}}$$

Next, we performed a similar t-test to check for equality between the two averages of the means of all four quadrants. The average of the means under the colour condition is called "Average CC" and under the motion condition "Average MC". Therefore, we tested the following null and alternative hypothesis.

$$H_0 : \text{Average CC=Average MC vs } H_1 : \text{Average CC} \neq \text{Average MC}$$

The results of these statistical tests are presented in Table 4.6 and reveal that in every case, the means from the two distances matrices are significantly different. Therefore, the dimensions are stretched differently, depending on the cued task, as expected.

	t-statistic	degrees of freedom	p-value	Evidence
Quadrant 1	28.2877	126	2.0104e-56	Strong evidence against H_0
Quadrant 2	32.3365	126	7.3155e-63	Strong evidence against H_0
Quadrant 3	64.4453	126	2.5396e-98	Strong evidence against H_0
Quadrant 4	77.9957	126	1.6473e-108	Strong evidence against H_0
Average of Quadrants	25.4679	510	6.2965e-93	Strong evidence against H_0

Table 4.6: Within Conditions t-test I

Showing the t-tests results assessing significant difference between the two types of ratios ($\bar{r}, \bar{r'}$) for each quadrant and for the average of all quadrants. Here the null hypothesis H_0 represent equality between the ratios.

Additionally, we carried out one-sided t-tests to test whether the average ratios of the ANN data are significantly better or the same as a fixed stretching or contraction ratio. The fixed ratio was constant and equal to 2 and 1/2 for colour and motion conditions, respectively. For the LSTM to do better, the ratios needed to be larger than 2 or smaller than 1/2 according to the condition. Thus, the null and alternative hypotheses are,

$$H_0 : \text{Average CC} = 2 \text{ vs } H_1 : \text{Average CC} > 2$$

$$H_0 : \text{Average MC} = 1/2 \text{ vs } H_1 : \text{Average MC} < 1/2$$

	t-statistic	degrees of freedom
Average CC	19.5361	255
Average MC	98.6720	255

Table 4.7: Within Conditions t-test II

Displaying the statistical results of the one-sided tests assessing the amount of stretching along the colour and motion dimension under each condition. In particular, one t-test checked whether the average ratio $\bar{r}$ of all quadrants under the colour condition (CC) was significantly larger than 2 and another t-test checked whether the average ratio $\bar{r'}$ under the motion condition (MC) was smaller than 1/2.

The computed test statistics in Table 4.7 have $p - value << 0.01$ and so the null hypotheses were rejected, implying that the LSTM stretched dimensions more than the baseline.

In conclusion, there was stretching of the expected dimensions within conditions. The model achieved significant stretching since the stretching factors were better than the baseline values. These findings verify that the model captured such attentional effects in a similar manner to primates, showing the potential of neural networks in the field of neuroscience and the modelling of the brain.

Between Conditions

Following the within conditions comparisons, we defined ratios calculated using values from both distance matrices of the LSTM, intending to check for stretching between conditions.

We calculated quantities r_BC1 and r_BC2 representing between conditions (BC) stretching ratios. More specifically, we set $r\_BC1 = \frac{d'_1}{d_1}$ denoting the change in the motion direction under the motion condition over the change in motion direction under the colour condition. Analogously, $r\_BC2 = \frac{d'_2}{d_2}$ represented the ratio of colour change when the task is motion over the colour change when the task is colour. We expected $r\_BC1 > 1$ and $r\_BC2 < 1$ since the change along the motion dimension should be larger when the cued task is motion and the change along the colour dimension should be larger under the colour task.

Again, we calculated the means of those ratios for each quadrant and then found the average of the means of all quadrants. The stretching ratios between conditions, calculated using the ANN data, are shown in Table 4.8. The means of the ratios of changes in motion and colour, are indeed considerably higher and lower than 1, respectively.

	Change in Motion ($\bar{r}BC1$)	Change in colour ($\bar{r}BC2$)
Quadrant 1	8.7744	0.1382
Quadrant 2	8.7744	0.1362
Quadrant 3	2.9460	0.1362
Quadrant 4	2.9460	0.1382
Average	8.7744	0.1382

Table 4.8: Stretching Ratios Between Conditions
Presenting the two types of ratios $r\bar{B}C1$ and $r\bar{B}C2$, computed from the ANN distance matrices, for each quadrant and overall. Quantity $r\bar{B}C1$ denotes the ratio of the motion direction change under the motion condition over the colour condition. Similarly, quantity $r\bar{B}C2$ denotes the colour change ratio of the motion condition over the colour condition. So, we expected $r\bar{B}C1 > 1$ and $r\bar{B}C2 < 1$.

We performed a one-sided t-test to assess whether the stretching along either dimension is at least 2. The hypotheses we tested are the following.

$$H_0 : \bar{r}BC1 = 1/2 \text{ vs } H_1 : \bar{r}BC1 < 1/2 \iff \bar{d_1} > 2 * \bar{d_{1'}}$$

$$H_0 : \bar{r}BC2 = 2 \text{ vs } H_1 : \bar{r}BC2 > 2 \iff \bar{d'_2} > 2 * \bar{d_2}$$

The results of such tests are presented in Table 4.9. Both computed test statistics yielded $p - value << 0.01$. Thus, there was strong evidence against H_0 in both cases, and so the factor of stretching was larger than 2.

	t-statistic	degrees of freedom
Change in colour	174.2882	255
Change in motion	24.0061	255

Table 4.9: Between Conditions t-test
Displaying the statistical results of the one-sided tests assessing the amount of stretching under the motion and under the colour condition for each dimension. In particular, one t-test checked whether the average ratio $r\bar{B}C1$ of all quadrants was significantly less than 1/2 and another t-test checked whether the average ratio $r\bar{B}C2$ was larger than 2.

Similarly to within conditions comparisons, we found significant stretching along the correct dimensions. Therefore, the model stretched the colour and motion dimensions according to intuition, again proving the neural network's ability to learn such patterns by itself.

4.3.2 Empirical Data

In this section, we compared the ANN data with empirical data from the monkeys. By performing these comparisons, we aim to characterise the distance function of the monkey brain. The empirical data represented the findings of the original experiments. As described in Section 3.4.2, the dataset containing the empirical data included pairwise distances computed using several distance functions. For comparison purposes, we used two of them, namely the isi_full and $spike\_full$ distances.

Distance function: isi_full

The distance matrix under the colour condition, created using the isi_full distances from the empirical data, is displayed in Figure 4.10. Visually comparing this matrix with its counterpart from the ANN data, we can see some similarities in terms of which distances are larger.

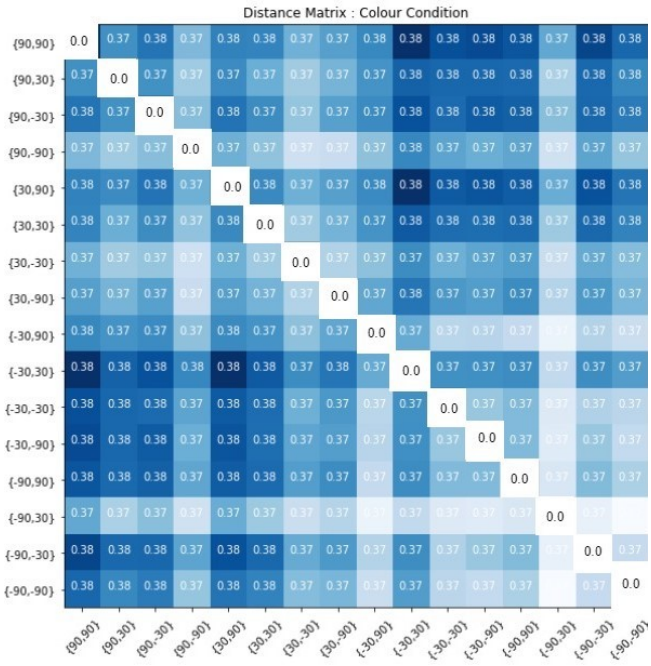


Figure 4.10: Empirical Data ISI-Distance Matrix (task: colour)
Colour condition distance matrix derived from the ISI-distances computed from the empirical data.

We tested for correlation between the two matrices, using the Spearman's rank test with H_0 corresponding to no association between the two matrices. On top of that, we further analysed the monkey data by splitting it into the six brain regions and comparing each of them with the ANN colour condition distance matrix. Table 4.10 shows the outputs of these tests.

	Correlation	p-value	Evidence
All regions	0.3417	0.0001339	Very strong evidence against H_0
PFC	0.2834	0.001706	Very strong evidence against H_0
FEF	0.4345	7.120e-07	Very strong evidence against H_0
LIP	0.1030	0.2630	No evidence against H_0
IT	-0.01276	0.8900	No evidence against H_0
MT	0.06797	0.4607	No evidence against H_0
V4	0.09974	0.2784	No evidence against H_0

Table 4.10: ANN and Empirical Data Comparisons under the Colour Condition (isi_full)
Presenting the Spearman's correlation coefficients indicating association under the colour condition, between the ANN and the ISI-distance empirical data as well as the subsets of ISI-distance data corresponding to a specific brain region. H_0 represents no association.

We conclude that the empirical and ANN data are correlated under the colour condition (0.3417). The ANN data resembled brain regions PFC and FEF with the latter having the strongest correlation. No association was found between the ANN data and the other four brain regions. This suggests that representations change across brain regions, under the colour condition.

Similarly, the empirical distance matrix under the motion condition, which was also created from the monkey data with the isi_full distances, is shown in Figure 4.11. Again, just a visual comparison of the patterns of the matrices in terms of the magnitude of the distances, suggested that there is a correlation between ANN and empirical data.

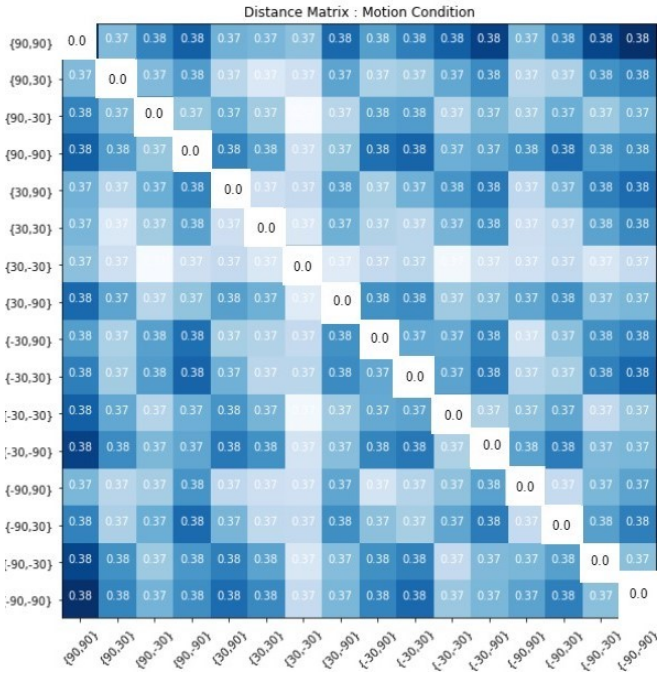


Figure 4.11: Empirical Data ISI-Distance Matrix (task: motion)
Motion condition distance matrix derived from the ISI-distances computed from the empirical data.

Then, we tested for such correlation using the full dataset as well as the subsets corresponding to each brain region. The results in Table 4.11 reveal a strong association between the ANN data and the empirical data (0.5060). Brain regions PFC, LIT and LIP, seem to have a positive correlation with the ANN data while no association is found for IT, MT and V4, verifying the change in representations across brain region under the motion condition as well. Another thing to note is that the correlation between the two datasets is stronger under the motion condition (0.5060) rather than the colour condition (0.3417).

	Correlation	p-value	Evidence
All regions	0.5060	3.722e-09	Very strong evidence against H_0
PFC	0.3988	6.444e-06	Very strong evidence against H_0
FEF	0.4725	5.084e-08	Very strong evidence against H_0
LIP	0.2032	0.02605	Strong evidence against H_0
IT	-0.04342	0.6377	No evidence against H_0
MT	0.1923	0.0354	No evidence against H_0
V4	-0.08015	0.3842	No evidence against H_0

Table 4.11: ANN and Empirical Data Comparisons under the Motion Condition (isi_full)
Presenting the Spearman's correlation coefficients indicating association under the motion condition, between the ANN and the ISI-distance empirical data as well as the subsets of ISI-distance data corresponding to a specific brain region.

Furthermore, we checked these results against the findings of the original experiment. In the original experiment, there were five types of information recorded (cue identity, task, colour, motion and choice) and as can be seen from Table 2.1, in regions PFC and FEF the strongest type of information was "choice". This implies that data from the LSTM is more compatible with the "choice" information, even though all types information were incorporated into the neural network.

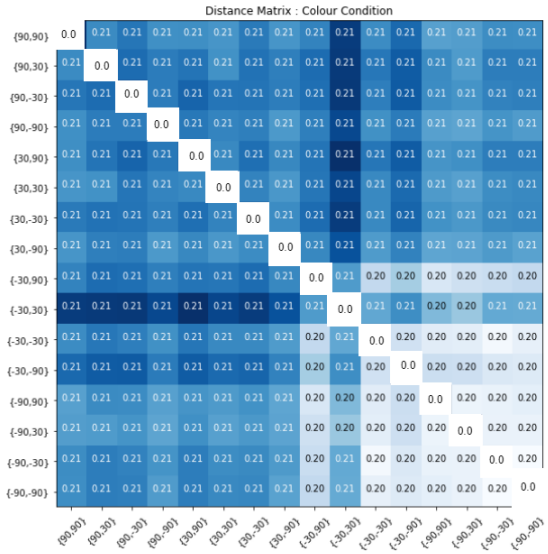
Distance function: $spike\_full$

We repeated the process using the $spike\_full$ distances and obtained the distance matrices in Figure 4.12. A visual comparison of the matrices with the respective ones from the ANN, indicated a possible correlation between the two sets of distance matrices (LSTM and $spike\_full$).

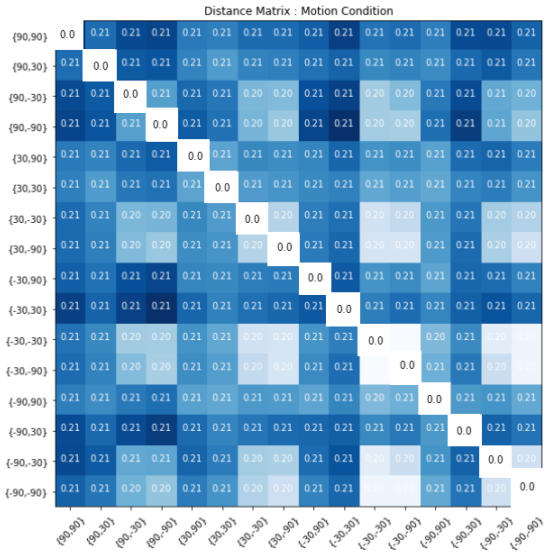
We checked for correlation more explicitly, using the Spearman's rank test. Table 4.12 and Table 4.13 present the outputs for the colour and motion distance matrices respectively. There is a positive correlation between the empirical and ANN data under both conditions, with 0.3550 and 0.4349 being the correlation coefficient corresponding to the colour and the motion conditions, respectively. Further analysing, we broke it down to the six brain regions. We deduce that, under the colour condition, regions PFC, FEF, IT and MT are strongly associated with the ANN data (Table 4.12).

Under the motion condition, Table 4.13 relates the ANN data with regions PFC, FEF and MT, while no association is found for LIP, IT and V4.

Therefore, the brain regions for which the empirical data (evaluated using $spike\_full$) is mostly correlated with the ANN data, are PFC, FEF and MT (under both conditions).



(a) Condition: Colour



(b) Condition: Motion

Figure 4.12: Empirical Data SPIKE-Distance Matrices
Illustrating the distance matrices derived from the SPIKE-distances of the empirical data, under the colour and the motion conditions.

	Correlation	p-value	Evidence
All regions	0.3550	6.9237e-05	Very strong evidence against H_0
PFC	0.3475	0.0001009	Very strong evidence against H_0
FEF	0.6110	1.257e-13	Very strong evidence against H_0
LIP	0.07048	0.4443	No evidence against H_0
IT	0.1985	0.02979	Strong evidence against H_0
MT	0.2051	0.02460	Strong evidence against H_0
V4	-0.006987	0.9396	No evidence against H_0

Table 4.12: ANN and Empirical Data Comparisons under the Colour Condition (spike_full)

	Correlation	p-value	Evidence
All regions	0.4349	6.935e-07	Very strong evidence against H_0
PFC	0.3701	3.176e-05	Very strong evidence against H_0
FEF	0.6164	6.645e-14	Very strong evidence against H_0
LIP	0.05486	0.5517	No evidence against H_0
IT	-0.01806	0.8447	No evidence against H_0
MT	0.3062	0.0006722	Very strong evidence against H_0
V4	0.1254	0.1725	No evidence against H_0

Table 4.13: ANN and Empirical Data Comparisons under the Motion Condition (spike_full)

In the original experiment findings, PFC and FEF hold "choice" information. Also, region MT, contains the strongest "motion" information. Thus, the ANN data is more consistent with "choice" and "motion" information in the $spike\_full$ case.

In conclusion, correlating the neural network with the empirical data leads to different coefficients depending on the choice of empirical dissimilarity measure. Although the correlation coefficients are similar to the corresponding values computed using the isi_full distance, the colour correlation is larger and motion correlation is smaller in this case.

4.3.3 Baseline correlation

We defined two types of baseline distance matrices. We compared the empirical data with these baselines, leading to two baseline correlation coefficients. Ranking the correlation coefficients between the empirical data (ISI and SPIKE) and the three models (ANN and two baselines) under each condition, revealed which model best described the monkey neural recordings subject to the empirical dissimilarity measure.

For the first baseline model, we created a distance matrix by just calculating the Euclidean distances between each pair of 2D points. For example, the element of the distance matrix corresponding to dissimilarity between points $\{90°, 90°\}$ and $\{90°, 30°\}$ was $\sqrt{(90-90)^2 + (90-30)^2}$. The resulting matrix is shown in Figure 4.13.

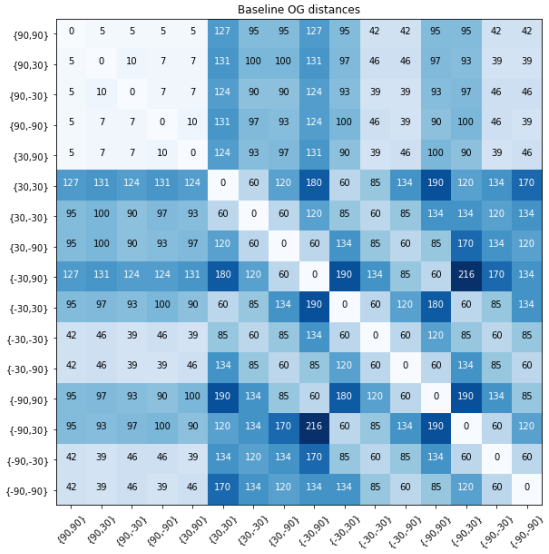
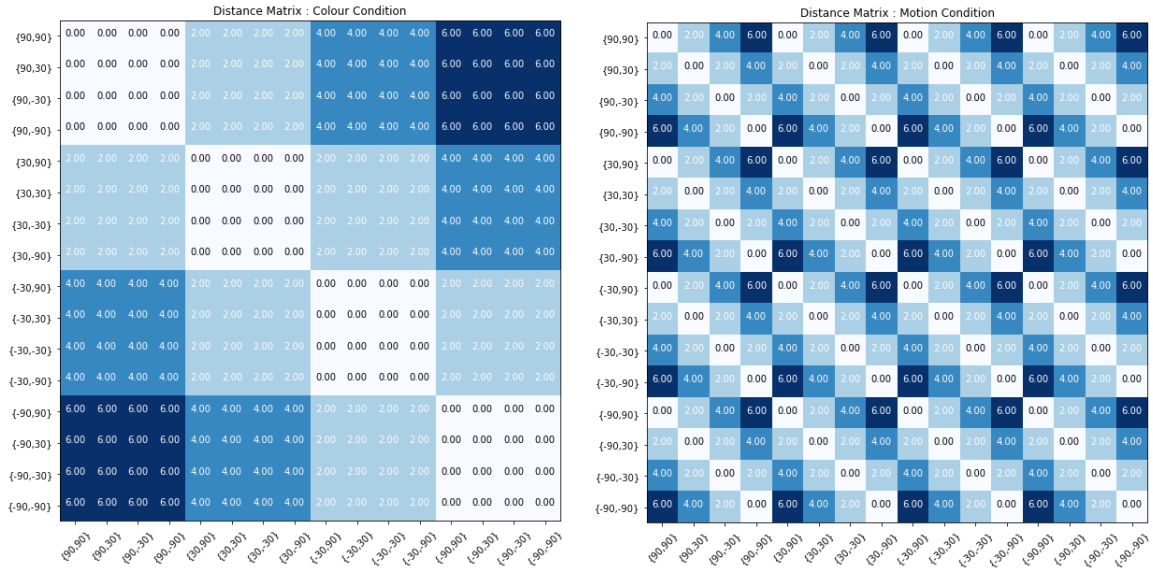


Figure 4.13: First Baseline Distance Matrix
Illustrating the distance matrix acting as the first baseline, computed from the Euclidean distance of the labels of the pairwise combinations.

The second type of baseline represented constant stretching and depended on the condition. More explicitly, under the colour condition, when the two points $\{x_1, y_1\}$ and $\{x_2, y_2\}$ were different in the first coordinate (colour changes), the distance is equal to $|x_1 - x_2|$ times a factor of 2. Using a factor of 2 was just a heuristic but should not make a significant difference when performing the Spearman's test. The analogous was applied for the motion condition with the second coordinates y_1 and y_2. A visualisation of these matrices is shown in Figure 4.14.



(a) Condition: Colour

(b) Condition: Motion

Figure 4.14: Second Baseline Distance Matrices

Illustrating the distance matrices derived from the second baseline model, corresponding to constant stretching for either the colour or motion conditions.

The results of the Spearman's rank tests are presented in Table 4.14 and Table 4.15 for the isi_full and $spike\_full$ distances respectively.

	Correlation	p-value
Monkey data & baseline (Colour)	-0.2349	0.009824
Monkey data & Constant stretching (Colour)	**0.3941**	8.457e-06
Monkey data & ANN data (Colour)	0.3417	0.0001339
Monkey data & baseline (Motion)	-0.1679	0.06682
Monkey data & Constant stretching (Motion)	**0.6235**	2.840e-14
Monkey data & ANN data (Motion)	0.5060	3.722e-09

Table 4.14: Monkey Data Correlations (isi_full)

	Correlation	p-value
Monkey data & baseline (Colour)	-0.2288	0.01197
Monkey data & Constant stretching (Colour)	0.1580	0.08473
Monkey data & ANN data (Colour)	**0.3550**	6.9237e-05
Monkey data & baseline (Motion)	0.03344	0.7170
Monkey data & Constant stretching (Motion)	0.3533	7.546e-05
Monkey data & ANN data (Motion)	**0.4349**	6.935e-07

Table 4.15: Monkey Data Correlations (spike_full)

In both cases, the first type of baseline has the weakest association with the empirical data. The data corresponding to the isi_full distance function has the strongest correlation with the second type of baseline. This is the case both under the colour and under the motion conditions. Conversely, the correlation coefficient between $spike\_full$ empirical data and the ANN data is the largest correlation coefficient under both conditions. These findings suggest that different distance functions produce different outcomes and so how we measure dissimilarity matters both on the modelling and the data side.

Chapter 5

Conclusions and Future work

The fundamental question we wish to answer in this study is how to best measure dissimilarity for monkey neural recordings. This is a basic neuroscience question, critical to a range of cognitive tasks. Characterising the dissimilarity measure helps advance our understanding on the computational principles of the brain. One of the main results found in this study is that the dissimilarity measure obtained from the developed CNN-LSTM model performs dimensional stretching following the pattern we expect from primates. This implies that neural networks share computational principles with primate brains. Additionally, we found that dissimilarity computations change as a function of task (colour and motion). Also, we investigated the way the neural network interacts with the empirical dissimilarity measure by comparing the ANN and empirical data. Such comparisons revealed that the choice of empirical dissimilarity measure (ISI-distance or SPIKE-distance) affects the amount of correlation with the neural network. We deduce that SPIKE-distance data was more strongly correlated with the neural network data.

In this dissertation, we have presented a strategy utilising a CNN and an LSTM to implement the simple categorisation task also performed by two monkeys in [50], hence describing the monkeys' biological neural recordings. We developed a neural network which reads a sequence of images showing a fixation point, followed by a cue and finally a stimulus with random dots of specific colour and motion direction. We trained a CNN to discriminate images of the fixation point, the different cue shapes and the different dot colours. We then proposed an LSTM which recognised a categorisation task (colour or motion), based on the indicated cue, and classified colour as red, green or yellow, or motion as downwards, upwards or horizontal, based on the task cues. We added an extra class, which was not an option in the original experiment, to correspond to the

two category boundaries, namely the boundary between red and green, and the boundary between upwards and downwards. The addition of a third class has made the results more robust, since even monkeys could not correctly classify the stimuli corresponding to this extra class. The proposed CNN-LSTM strategy involved taking the activations of the trained CNN to be the LSTM inputs, taking advantage of the CNN for image-based classification and the LSTM for sequence data processing. The resulting LSTM was used to produce distance matrices, measuring dissimilarity between activations, and compared to distance function values computed from empirical data.

The first component of the neural network is the CNN. We trained a CNN for three image shapes, namely $(3, 32, 32), (3, 64, 64)$ and $(3, 128, 128)$, attempting to discover, via analysing the confusion matrices, which models behave similarly to monkeys as well trade-off between similarity with the original dataset and computational time. We found that all three CNNs succeed in differentiating the various image types, exceeding 80% accuracy with order of decreasing performance "CNN32x32", "CNN128x128" and "CNN64x64". All three classify the fixation point and cue images perfectly, mostly confusing colours on the category boundaries. These results are consistent with our intuition, with "CNN32x32" performing best since the images have lower resolution, making the coloured dots to appear more dense, which accommodates colour discrimination. However, $(3, 128, 128)$ images are closer to the image dataset we are imitating (from the original experiment) and also have more distinct dots due to their higher resolution, aiding motion discrimination (in the LSTM). For the reasons described, and also since the differences in the performance were insignificant, we chose to proceed with "CNN128x128".

The second step of the neural network architecture was building an LSTM which uses memory blocks to capture long-range temporal dependencies. We trained an LSTM with six choices for the sequence length, representing different response times, attempting to reflect the fact that in the original experiment, monkeys had to give their response within an interval of 3s (after stimulus onset). We indented to determine at which point the LSTM was most competent at differentiating between the two tasks (context) and categorising colour or motion (classification). The LSTM's inputs represented the sequences of ordered images similar to the ones shown during the original experiment. For each trial, the corresponding sequence of images was passed through the "CNN128x128" one by one, obtaining the activation from the second to last layer of the CNN model. We used the produced activations as inputs to the LSTM and we investigated the six LSTM architectures. All architectures give outstanding results, exceeding 95% accuracy, with the LSTM of se-

quence length 250 (corresponding to 2.667s) outperforming the others. This result may come from the fact that LSTM processes data sequentially, and so showing more stimulus images would affect how well it remembers the cue shape, while showing fewer stimulus images would not enable the model to understand the colour or the motion of the stimulus.

We obtained the activations derived from the trained LSTM and partitioned them into groups according to task (colour or motion condition). These activations were averaged based on the colour-motion combination of the random dots in each stimulus. Then, we created two distance matrices by taking the Euclidean distance between the average activations of pairwise combinations. The distance matrices reveal patterns showing that representation changes as a function of task, close to how we would expect from humans and monkeys. Therefore, the neural network reflects the dimensional stretching found in the monkey data.

Also, we formally checked for the existence and the amount of such stretching by computing stretching ratios and carrying out statistical t-tests . We found that, under the colour condition, the colour dimension is stretched by a factor considerably larger than 2 and similarly, under the motion condition, there is significant stretching along the motion direction with stretching factor larger than 2 (Within Conditions comparisons). Additionally, we found that the colour dimension is stretched by a larger amount when there is a change in the colour classification under the colour condition rather than under the motion condition. Similarly, there is more stretching along the motion dimension when motion classification changes under the motion rather than the colour condition (Between Conditions comparisons). Interestingly, the network learns something that primates do regardless since these results show significant stretching along the dimensions we expected. Although such stretching is expected, it is not trivial.

Further analysis included comparing the distance matrices of the neural network and empirical data, in order to find what dissimilarity measure suits the monkey brain best. We computed two sets of distance matrices using the empirical data, each representing one of the two chosen distance functions, namely ISI and SPIKE. We correlated the empirical with the ANN data using the Spearman's rank test and got two sets of results, for ISI-distance and SPIKE-distance. For each distance function we obtained two correlation coefficients, one for the colour condition (0.3417 for the ISI-distance and 0.3550 for the SPIKE-distance) and one for the motion condition (0.5060 and 0.4349). We found that the ANN and empirical data are strongly correlated under both conditions, with the motion condition data exhibiting a stronger correlation than the colour condition data. Also, we

further correlated the ANN data with subsets of empirical data corresponding to each of the six brain regions of the monkeys. The resulting correlation coefficients, under both conditions and both distance functions, suggest that the "choice" is the strongest type of information in the neural network and that representations change across regions.

Finally, we compared the empirical data with the two baseline models on top of the neural network model, ranking the correlation coefficients of the three, in order to select the most suitable model for describing each of the two dissimilarity measures (ISI and SPIKE). When we propose a model, either the neural network or one of the baselines, we guess the ground truth (the true dissimilarity measure) and expect representations to resemble those of the proposed models. The first baseline model gave a single distance matrix constructed from the Euclidean distances between the combination labels. The second baseline, which depends on the condition, was characterised by a fixed stretching factor. The outcome of the Spearman's tests with regards to the ISI-distance revealed that the constant stretching baseline model describes the empirical data best, followed by the neural network and the first baseline. Conversely, the findings regarding the SPIKE-distance indicated that ANN data have the strongest association with the empirical data, followed by the constant stretching data. The rank of these correlations acted as a way to select the model providing the best description of the empirical data, given the set of models at our disposal.

The above results, provide an insight into the dissimilarity measure that the neural network invents and an understanding of how to model the neural behaviour of monkeys. Mainly, we deduce that both on the modelling and the data side, the way we measure dissimilarity is very important. On the modelling side, the three models correlate differently with the empirical data. On the data side, for different distance functions we select a different model (for ISI we select constant stretching baseline model and for SPIKE we select the neural network model).

Future work could experiment with getting the activations from different layers, both for the CNN and the LSTM. The distance matrices for each layer choice could be recorded and visualised, providing a better understanding as to how these distances evolve as we go through the models. Also, we would ideally like to have data derived from responses of more than two monkeys as well as responses for more complicated tasks.

An interesting experiment would be to use different methods for measuring the distance between the average activations from the LSTM and compare the results to the Euclidean distance method we used in this study. Moreover, examining the effects of different neural

network architectures would provide insight as to what attributes are necessary for the network to perform the kind of stretching we saw. An alternative approach that could improve the results is end-to-end learning. Instead of first building a CNN and then an LSTM, end-to-end learning approaches, like convolutional LSTMs, train parameters jointly by inputting images through a CNN-type architecture and then through an LSTM, learning all gradients ([16]).

Further research could also look at model classes of interest, encouraging the neural network itself to incorporate more cognitive phenomena exhibited by monkeys or humans. For instance, instead of average dissimilarity from the network, it may be worth encouraging the LSTM to pick up on context from the previous trial, relating the model to the data at a trial-by-trial level. An LSTM capturing these long-range dependencies may give a better fit, even though we would have to deal with noise in the spike trains.

Lastly, investigating other kinds of measures apart from ISI-distance and SPIKE-distance would give a better understanding of how to best measure dissimilarity in the monkey brain. Other measures are the Victor–Purpura distance and the van Rossum distance. Metric learning can also be used to construct task-specific distance metrics, automatically. Alternatively, we could construct our own dissimilarity measure for both modelling and data. Additionally, further research could focus on data-driven approaches like in [5], which looked at the data and uses a classifier to get accuracy on all stimulus. Then, it used the confusion matrix of the classifier as the ground truth to infer dissimilarity measure. It would also be interesting to examine how non-timing measures performed for a qualitative comparison with ISI and SPIKE.

In summary, the CNN and LSTM-based model successfully accomplishes a visuomotor task, performing as good as the trained monkeys. The neural network dissimilarity measure is competent in describing the monkey neural recordings. The neural network picks up certain attentional effects like dimensional stretching in a similar manner as primates. Regarding the contribution to machine learning, our method of combining a CNN and an LSTM via the activations of intermediate layers, has proved to be of significant importance as it achieves outstanding performance in terms of accuracy and loss. It provides opportunities to model video-type data for a broader range of applications and more complex tasks. The findings of this study contribute to neuroscience as well, revealing that the choice of dissimilarity measure is crucial, having practical implications in neural data analysis. In essence, machine learning and neuroscience can be mutually benefit. Neural networks can be utilised to model brain activity, enabling manipulation of neural data as-

pects via neural network architectures. Additionally, neuroscience questions inspire more complex machine learning models and tasks, investigating in depth the extent to which machine learning can be turned to advantage to incorporate certain effects of neural activity and make those models more intelligent, capturing sophisticated patters apart from just implementing tasks. Although further work could give more depth into the characteristics of the dissimilarity measure used by the monkey brain, the findings of this study are very promising.

As a final note, the code of this project is available at
https://www.dropbox.com/sh/8ri5o2w2n0gi3o3/AABjHoGRIRcXM1s5-P8csHMaa?dl=0

Bibliography

[1] Shohel Ali Ahmed, Snigdha Dey, and Kandarpa Kumar Sarma. Image texture classification using artificial neural network (ann). In *2011 2nd National Conference on Emerging Trends and Applications in Computer Science*, pages 1–4. IEEE, 2011.

[2] Mariam Aly, Charan Ranganath, and Andrew P Yonelinas. Detecting changes in scenes: The hippocampus is critical for strength-based perception. *Neuron*, 78(6):1127–1137, 2013.

[3] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[4] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[5] Sebastian Bobadilla-Suarez, Christiane Ahlheim, Abhinav Mehrotra, Aristeidis Panos, and Bradley C Love. Measures of neural similarity. *Computational Brain & Behavior*, pages 1–15, 2019.

[6] Thomas M Breuel. High performance text recognition using a hybrid convolutional-lstm implementation. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 11–16. IEEE, 2017.

[7] G Chandan, Ayush Jain, Harsh Jain, et al. Real time object detection and tracking using deep learning and opencv. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 1305–1308. IEEE, 2018.

[8] Marc N Coutanche and Sharon L Thompson-Schill. Creating concepts from converging features in human cortex. *Cerebral cortex*, 25(9):2584–2593, 2015.

[9] Muhammad Najam Dar, Muhammad Usman Akram, Sajid Gul Khawaja, and Amit N Pujari. Cnn and lstm-based emotion charting using physiological signals. *Sensors*, 20(16):4551, 2020.

[10] Zixiang Ding, Rui Xia, Jianfei Yu, Xiang Li, and Jian Yang. Densely connected bidirectional lstm with applications to sentence classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 278–287. Springer, 2018.

[11] Mahidhar Dwarampudi and NV Reddy. Effects of padding on lstms and cnns. *arXiv preprint arXiv:1903.07288*, 2019.

[12] WA Ezat, MM Dessouky, and NA Ismail. Multi-class image classification using deep learning algorithm. In *Journal of Physics: Conference Series*, volume 1447, page 012021. IOP Publishing, 2020.

[13] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, Cham, 2019.

[14] Banda Gerald. A brief review of independent, dependent and one sample t-test. *International Journal of Applied Mathematics and Theoretical Physics*, 4(2):50–54, 2018.

[15] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[16] Tobias Glasmachers. Limits of end-to-end learning. *arXiv preprint arXiv:1704.08305*, 2017.

[17] Robert L Goldstone. The role of similarity in categorization: Providing a groundwork. *Cognition*, 52(2):125–157, 1994.

[18] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE, 2005.

[19] Qing Guan, Yunjun Wang, Bo Ping, Duanshu Li, Jiajun Du, Yu Qin, Hongtao Lu, Xiaochun Wan, and Jun Xiang. Deep convolutional neural network vgg-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: a pilot study. *Journal of Cancer*, 10(20):4876, 2019.

[20] Umut Güçlü and Marcel AJ van Gerven. Modeling the dynamics of human brain activity with recurrent neural networks. *Frontiers in computational neuroscience*, 11:7, 2017.

[21] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan. Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. *arXiv preprint arXiv:1706.02737*, 2017.

[24] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[25] Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100:234–245, 2018.

[26] Eric R Kandel, James H Schwartz, and Thomas M Jessell. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

[27] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[28] Tae Kyun Kim. T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540, 2015.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Thomas Kreuz, Daniel Chicharro, Conor Houghton, Ralph G Andrzejak, and Florian Mormann. Monitoring spike train synchrony. *Journal of neurophysiology*, 109(5):1457–1472, 2013.

[31] Thomas Kreuz, Julie S Haas, Alice Morelli, Henry DI Abarbanel, and Antonio Politi. Measuring spike train synchrony. *Journal of neuroscience methods*, 165(1):151–161, 2007.

[32] Thomas Kreuz, Mario Mulansky, and Nebojsa Bozanic. Spiky: a graphical user interface for monitoring spike train synchrony. *Journal of neurophysiology*, 113(9):3432–3445, 2015.

[33] Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1:417–446, 2015.

[34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[35] Wai-Kim Leung, Xunying Liu, and Helen Meng. Cnn-rnn-ctc based end-to-end mispronunciation detection and diagnosis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8132–8136. IEEE, 2019.

[36] Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. Medical image classification with convolutional neural network. In *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 844–848. IEEE, 2014.

[37] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[38] Rafael G Mantovani, André LD Rossi, Joaquin Vanschoren, Bernd Bischl, and André CPLF De Carvalho. Effectiveness of random search in svm hyper-parameter tuning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. Ieee, 2015.

[39] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[40] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.

[41] Elisa Mussumeci. *A machine learning approach to dengue forecasting: comparing LSTM, Random Forest and Lasso*. PhD thesis, 2018.

[42] Haidar Osman, Mohammad Ghafari, and Oscar Nierstrasz. Hyperparameter optimization to improve bug prediction accuracy. In *2017 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE)*, pages 33–38. IEEE, 2017.

[43] Shivarudhrappa Raghu, Natarajan Sriraam, Yasin Temel, Shyam Vasudeva Rao, and Pieter L Kubben. Eeg based multi-class seizure type classification using convolutional neural network and transfer learning. *Neural Networks*, 124:202–212, 2020.

[44] Mohammad Rezaee, Yun Zhang, Rakesh Mishra, Fei Tong, and Hengjian Tong. Using a vgg-16 network for individual tree species detection with an object-based approach. In *2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*, pages 1–7. IEEE, 2018.

[45] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

[46] MCW van Rossum. A novel spike distance. *Neural computation*, 13(4):751–763, 2001.

[47] Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6940–6948, 2019.

[48] Eero Satuvuori and Thomas Kreuz. Which spike train distance is most suitable for distinguishing rate and temporal coding? *Journal of neuroscience methods*, 299:22–33, 2018.

[49] Martin Schrimpf, Jonas Kubilius, Ha Hong, Najib J Majaj, Rishi Rajalingham, Elias B Issa, Kohitij Kar, Pouya Bashivan, Jonathan Prescott-Roy, Kailyn Schmidt, et al. Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, page 407007, 2018.

[50] Markus Siegel, Timothy J Buschman, and Earl K Miller. Cortical information flow during flexible sensorimotor decisions. *Science*, 348(6241):1352–1355, 2015.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[52] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[53] Fabio Alexandre Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN)*, pages 2560–2567. IEEE, 2016.

[54] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[55] Bálint Pál Tóth, Márton József Tóth, Dávid Papp, and Gábor Szücs. Deep learning and svm classification for plant recognition in content-based large scale image retrieval. In *CLEF (Working Notes)*, pages 569–578, 2016.

[56] Lorraine K Tyler, Helen E Moss, MR Durrant-Peatfield, and JP Levy. Conceptual structure and the structure of concepts: A distributed account of category-specific deficits. *Brain and language*, 75(2):195–231, 2000.

[57] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6:1155–1166, 2017.

[58] Francisco J Valverde-Albacete and Carmen Peláez-Moreno. 100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox. *PloS one*, 9(1):e84217, 2014.

[59] Jonathan D Victor and Keith P Purpura. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of neurophysiology*, 76(2):1310–1326, 1996.

[60] Jonathan D Victor and Keith P Purpura. Metric-space analysis of spike trains: theory, algorithms and application. *Network: computation in neural systems*, 8(2):127–164, 1997.

[61] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[62] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[63] Daniel L Yamins, Ha Hong, Charles Cadieu, and James J DiCarlo. Hierarchical modular optimization of convolutional networks achieves representations similar to macaque it and human ventral stream. In *Advances in neural information processing systems*, pages 3093–3101, 2013.

[64] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[65] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

[66] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[67] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[68] Guangming Zhu, Liang Zhang, Peiyi Shen, and Juan Song. Multimodal gesture recognition using 3-d convolution and convolutional lstm. *Ieee Access*, 5:4517–4524, 2017.

Appendix A

LSTM for Motion

	Training	Validation
Loss	0.2924	0.2831
Accuracy	88.80 %	86.51 %

Table A.1: Losses & Accuracies (LSTM for Motion)

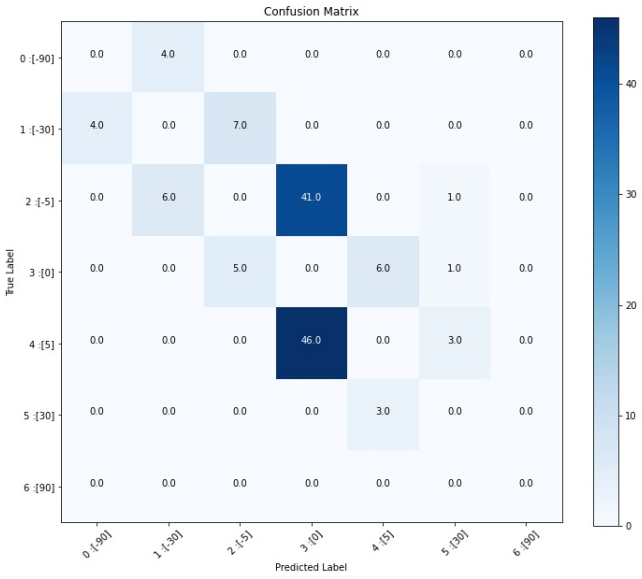


Figure A.1: Confusion matrix (LSTM for motion)

Appendix B

t-test

One of the ways to check for statistically significant difference between two quantities is to perform the t-test ([28]). If we want to check whether a mean is equal to a fixed value, we use the one-sample t-test (an explanation found in [14]), whereas if we are going to check for equality between two means, we use the two-sample t-test. In both cases, in order to test for statistical significance, we need the null and the alternative hypothesis, denoted by H_0 and H_1 respectively.

Student's t-distribution

The Student's t distribution is derived from the normal distribution. In particular, if $X_1, X_2, ..., X_n$ are independent and identically distributed (iid) from a normal distribution with mean μ and variance σ^2 ($\sim N(\mu, \sigma^2)$), then

$\frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \sim N(0,1)$,where $\bar{X} = \frac{\sum_{i=1}^{N} X_i}{n}$ is the sample mean.

Then, the random variable $\frac{\bar{X}-\mu}{\hat{\sigma}/\sqrt{n}} \sim t(n-1)$, where $\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(X_i-\bar{X})^2}{n-1}$ is the sample variance.

The probability density function of the Student's t distribution is illustrated in Figure B.1

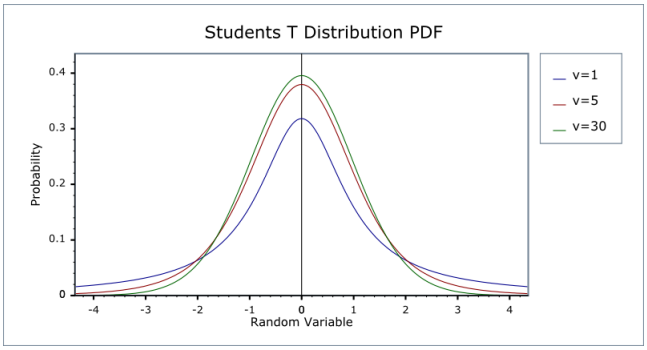


Figure B.1: Student's t distribution pdf

Assumptions

The assumptions under which we can apply the t-test are,

$$\begin{gathered}\bar{X} \sim N(\mu, \sigma^2/n) \\ \hat{\sigma}^2(n-1)/\sigma^2 \sim \chi^2(n-1) \\ (\bar{X} - \mu) \text{ and } (\hat{\sigma}/\sqrt{n}) \text{ are independent}\end{gathered}$$

One-sample

In the one-sample case, the hypothesis are as follows

$$H_0 : \text{mean} = \mu \text{ vs } H_1 : \text{mean} \neq \mu$$

Then the test statistic is found according to Equation B.1 (also explained in [14]), where n is the sample size, $\bar{X}$ the sample mean and $\hat{\sigma}$ the sample standard deviation. The degrees of freedom of the test statistic are $df = n - 1$.

$$\begin{aligned} t &= \frac{\bar{X} - \mu}{\hat{\sigma_X}} \\ \bar{X} &= \frac{\sum_{i=1}^{n} x_i}{n} \\ \hat{\sigma_X} &= \frac{\hat{\sigma}}{n} \\ \hat{\sigma}^2 &= \frac{\sum_{i=1}^{n} \left(x_i - \bar{X}\right)}{n-1} \end{aligned} \tag{B.1}$$

Under the assumptions, the t-statistic

$$t \sim t(n-1)$$

where $t(n-1)$ is the Student's t distribution with $n-1$ degrees of freedom.

Having the test statistic t and the degrees of freedom df, we estimate the range in which the p-value belongs. Then, the decision to either accept or reject H_0 is based on the p-value, as shown in Table B.1.

p-value range	**evidence type**
p-value <= 0.01	Strong evidence against H_0
0.01 < p-value <= 0.05	Evidence against H_0
0.05 < p-value <= 0.01	Weak evidence against H_0
p-value > 0.1	No evidence against H_0

Table B.1: P-value Significance Chart

Two-sample

In the two-sample case, there are two means, namely mean_A and mean_B of the samples A and B respectively. Then the test statistic is calculated according to Equation B.2, where $\hat{\sigma_A}$ and $\hat{\sigma_B}$ is the sample standard deviation of samples A and B. The degrees of freedom are $df = (n_A - 1) + (n_B - 1) = n_A + n_B - 2$.

$$\begin{aligned}
t &= \frac{\bar{X_A} - \bar{X_B}}{\hat{\sigma}\sqrt{\frac{1}{n_A} + \frac{1}{n_B}}} \\
\bar{X_A} &= \frac{\sum_{i=1}^{n} x_{Ai}}{n} \\
\bar{X_B} &= \frac{\sum_{i=1}^{n} x_{Bi}}{n} \\
\hat{\sigma}^2 &= \frac{(n_A - 1)\hat{\sigma_A}^2 + (n_B - 1)\hat{\sigma_B}^2}{n_A + n_B - 2} \\
\hat{\sigma_A}^2 &= \frac{\sum_{i=1}^{n} \left(x_{Ai} - \bar{X_A}\right)}{n_A - 1} \\
\hat{\sigma_B}^2 &= \frac{\sum_{i=1}^{n} \left(x_{Bi} - \bar{X_B}\right)}{n_B - 1}
\end{aligned} \tag{B.2}$$

Under the mentioned assumptions and the additional assumption that the variances of the independent populations are the same, the t-statistic

$$t \sim t(n_A + n_B - 2)$$

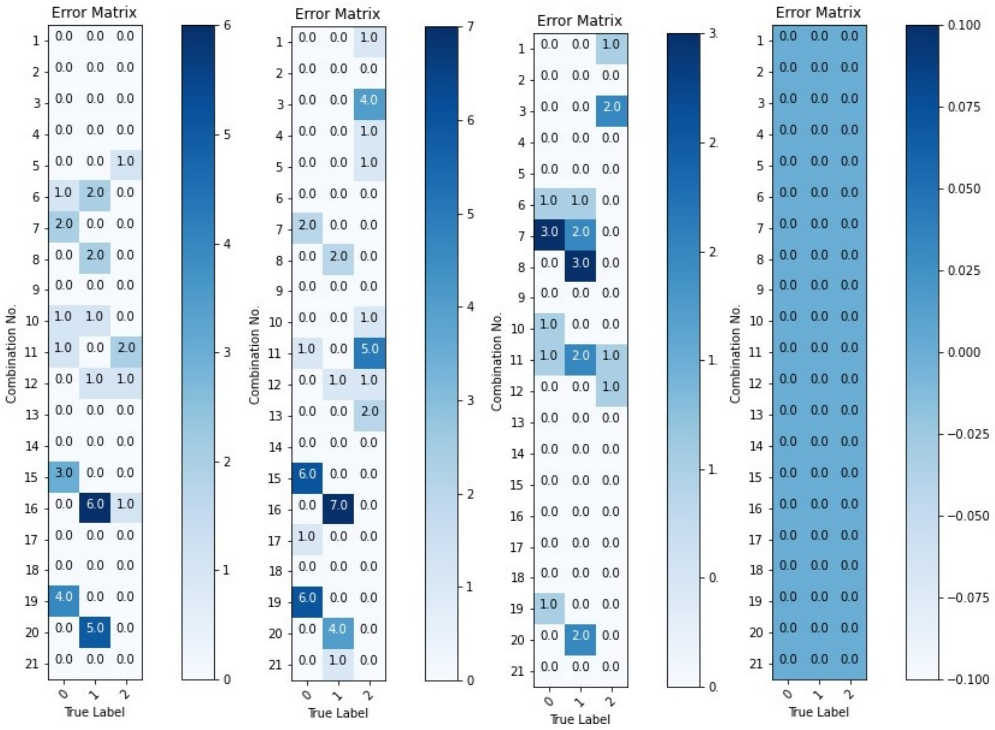
where $t(n_A + n_B - 2)$ is the Student's t distribution with $n_A + n_B - 2$ degrees of freedom.

Again, rejecting or accepting H_0 according to Table B.1.

Appendix C

LSTM Error Matrices

Further analysis of the LSTM results involved plotting the error matrices throughout training. This additional analysis is conducted to better understand how the chosen model works and where it makes mistakes.



(a) Epoch: 5 (b) Epoch: 7 (c) Epoch: 9 (d) Epoch: 11

Figure C.1: Error Matrices (LSTM)
Visualising how the classification errors are distributed across the colour/motion combinations, during training epochs (5,7,9 and 11).

Figure C.1 shows the error matrices at epochs $5, 7, 9$ and 11. Initially, mistakes are made for combinations several combinations such as $15, 16, 19$ and 20 which correspond $\{-30°, 30°\}, \{-30°, -30°\}, \{-90°, 30°\}$ and $\{-90°, 30°\}$. However, the model learns, eventually repairing all mistakes.