# SEMTM0011: Cognitive AI Coursework

Ben Hallett, ck21706@bristol.ac.uk, Student No. 2101829

November 22, 2024

Github Repository

## 1 Imposing Brain Like Constraints on an artificial neural network

Many Artificial Neural Network (ANN) architectures use bio inspired learning rules and architectures. However, there are often many simplifying assumptions and certain physical restraints seen in the brain are not present in common ANN architectures. Below we will discuss various ways in which ANNs can be adapted to impose brain like constraints on their behavior.

### 1.1 Architecture

The first major difference between ANNs and the brain, is that a single artificial neuron is not equivalent to a single human neuron. In fact, it has been shown that the firing rate of a pyrimidal neuron can be accurately modelled by a two layer fully connected neural network, with sigmoid nonlinearities [1], as shown in figure 1. The paper showed a correlation of $r^2 = 0.94$ fo firing rate predicted by a two layer ANN, against the firing rate predicted by a SOTA realistic model of neuron dynamics.

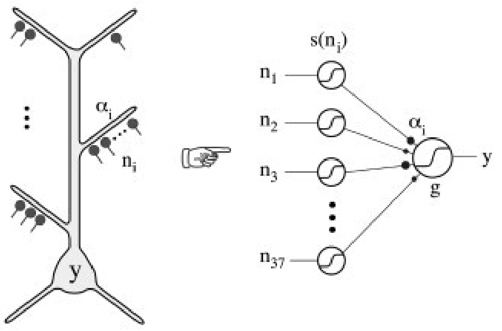

Figure 1: Correspondence between simple schematic of a neuron and the NN model [1]

The implication of this finding is that the output of a single neuron is roughly equivalent to a simple non-linear regression of its inputs. If we wanted to model a small recurrently connected region of neurons in the brain using an artificial RNN, we might represent each biological neuron with a two layer ANN forming a "unit". If we had a connectivity matrix, perhaps from real brain data, we might use this to form the inputs to each unit.

To implement this for a vanilla RNN architecture, first let's consider the equations that govern a RNN

$$\boldsymbol{h}_t = f(W_{hx}\boldsymbol{x}_t + U_h\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \tag{1}$$
$$\boldsymbol{y}_t = g(W_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{2}$$

where $\boldsymbol{h}_t$ is a vector of the hidden state activations at timestep t, $\boldsymbol{y}_t$ is a vector of the output activations at timestep t and $\boldsymbol{x}_t$ is the input state at timestep t. $W_{hx}$ is the weight matrix from the input values $\boldsymbol{x}_t$ to the hidden layer activations $\boldsymbol{h}_t$, $U_h$ is the recurrent weight matrix linking $\boldsymbol{h}_{t-1}$ to $\boldsymbol{h}_t$, and $\boldsymbol{b}_h$ is the bias term for the hidden layer activations. $f$ and $g$ are both nonlinearity functions.

If we replace the artificial neurons, whose activations at time t form the vector $\boldsymbol{h}_t$, with individual 2 layer ANNs, we end up with the recurrent network structure in figure 2

The layers of the network shown in figure 2, are related by the equations

$$\hat{\boldsymbol{h}}_t = f(W_{\hat{h}x}\boldsymbol{x}_t + \boldsymbol{b}_{\hat{h}}) \tag{3}$$
$$\boldsymbol{h}_t = f(W_{h\hat{h}}\hat{\boldsymbol{h}}_t + U_h\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{h}}) \tag{4}$$
$$\boldsymbol{y}_t = g(W_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{5}$$

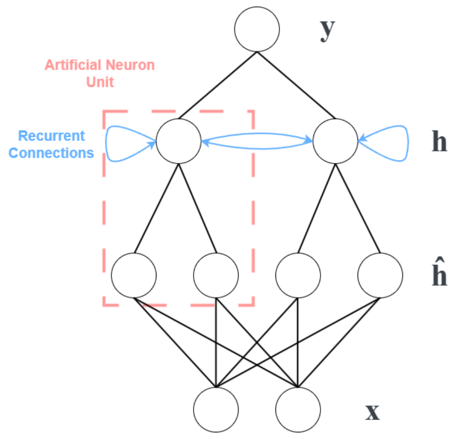

Figure 2: Simple Architecture of a small RNN with biologically realistic recurrent units, represented by the two layer networks highlighted in red. The recurrent layer is largely the same, except each node in the reccurent layer takes a different embedding of the input, $\boldsymbol{x}$ as input.

where $W_{h\hat{h}}$ is the sparse weight matrix which in the case of the small system in figure 2 will take the form

$$W_{h\hat{h}} = \begin{pmatrix} w_1^1 & w_2^1 & 0 & 0 \\ 0 & 0 & w_1^2 & w_2^2 \end{pmatrix} \tag{6}$$

In general, columns of $W_{h\hat{h}}$ can only contain one non-zero entry (linearly independent rows). The rest of the weight matrices are fully connected. In a biological context, we could think of this RNN as representing a brain region, with input $x_t$ coming from connections to neurons in primarily nearby brain regions, with perhaps a small number of long distance connection. the output of the RNN might be output to other brain regions or recurrently connected to the current brain region, forming part of its input. Interestingly, this sort of model would be consistent with a study of connectivity across the cortex [2], which found high local intrinsic connectivity, high input from neighbouring areas and weak long range connectivity.

Another architectural difference between ANNs and the brain is that most ANNs contain neurons which act in both an excitatory (positive output weights) and inhibitory manner (negative output weights), depending on what neuron they are connected too. In the brain, most neurons follow Dale's law, which states that neurons tend to be either excitatory or inhibitory [3]. With this in mind Song et al adapted the RNN architecture to follow Dales principle. They do this by enforcing consistent positivity or negativity of the recurrent weight matrix $U_h$ in the vanilla RNN model given in equation 1 and 2. This can be achieved by rectifying $U_h$ using a matrix mask of the form

$$U_{\text{rec}} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} \cdot & + & - \\ + & \cdot & - \\ + & + & \cdot \end{pmatrix} \tag{7}$$

where positive columns represent excitatory neurons and negative columns inhibitory neurons.

## 1.2 The Cost Function

When training ANNs for supervised learning tasks, backpropagation is typically used to generate loss gradients for each parameter in the network with respect to some cost function, which is based on the output prediction mismatch. A naive way of doing this for a task such as regression, would be to use the squared error between our target output and actual output after each forward pass of the network as our cost. The problem with this approach is that it does not constrain artificial neuron activations or weights to be biologically realistic, as in the brain. In the brain there is implicit energetic cost associated with connections between neurons and there is also limited available space. These two factors have likely driven the brain to grow with sparse connectivity and prioritise short distance over long distance connections which has been observed in cortical brain regions [2, 4].

We can impose brain like constraints through a number of regularisation techniques. Consider that our naive cost function, $L_{\text{tot}}$ is given by

$$L_{\text{tot}} = L_{\text{task}} \tag{8}$$

where $L_{\text{task}}$ is the loss associated with the output/prediction of the ANN. If we want to penalise large artificial neuron activations, to emulate the natural limit on firing rate in the brain, we can impose an additional condition

$$L_{\text{tot}} = L_{\text{task}} + L_{\text{activations}} = L_{\text{task}} + \beta_2 \|\boldsymbol{a}\|_2 \tag{9}$$

where $\|\boldsymbol{a}\|_2$ is the l-2 norm of the vector $\boldsymbol{a}$ containing the activation value of each neuron in the ANN and $\beta_2$ is a hyper parameter which controls the contribution of the norm to the total loss.

If we want to enforce sparsity in the network we can add the condition

$$L_{\text{tot}} = L_{\text{task}} + L_{\text{dense}} = L_{\text{task}} + \beta_1 \|\boldsymbol{w}\|_1 \tag{10}$$

where $\|\boldsymbol{w}\|_1$ is the l-1 norm of the vector $\boldsymbol{w}$ containing all of the weight values in the ANN and $\beta_1$ is a hyperparameter for the same purpose as $\beta_2$.

The obvious question that arises from these two regularisation terms is why does l-2 regularisation minimize scale and l-1 regularisation lead to sparsity? We can intuit this by looking at figure 3, which shows that the l-1 norm scales linearly with number of dimension (as you would expect) whilst the l-2 norm scales logarithmically. So for the l-1 weight regularisation, the best way to reduce the loss is to drive unimportant weight values towards zero. More specifically, if you imagine a connection between two neurons with weight $w_i$ which has does not contribute meaningly to the output of the network, so that $\partial L_{task}/\partial w_i \approx 0, \forall w_i \in \Re$, then the optimal value of $w_i$ satisfies $\partial L_{tot}/\partial w_i = \partial L_{dense}/\partial w_i \approx 0$ which requires that $w_i \approx 0$. If $\partial L_{task}/\partial w_i$ approaches zero for a large value of $w^*$ at a rate greater than 1 in the region of $w^*$ then the task loss will dominate and the weight will not be driven to zero.

Returning to the l-2 norm, we know that it approximates distance from the origin (magnitude) in 2 dimension, and we can see that since it scales logarithmically with the dimensionality, it approximates magnitude in higher dimensions too. So the best way to minimize the l-2 activation norm is to minimize the magnitude of each activation at the same rate. In other words, the gradient of the l-2 activation norm will always act to reduce all values approximately equally. If we wanted this to be exactly the case, we should use the l-n norm instead, where n is the number of units in the ANN, but higher order norms are more computational expensive to compute.

In the brain, distance also contributes to the energetic cost of neuron connections, therefore another way in which we might impose regularisation to the cost function is by giving each neuron in the network a physical location and adding a l-2 distance loss term to the cost function as in the paper by Achterberg et al. [5].

## 1.3 The learning Rule

By far the most common learning rule for the training of ANN in recent years has been the guided backpropogation algorithm (BP for short), in which output errors following a forward pass of a ANN are propogated back through the network through repeat application of the chain rule. The end result being loss gradients with respect to each parameter in the network, quantifying each parameters contribution

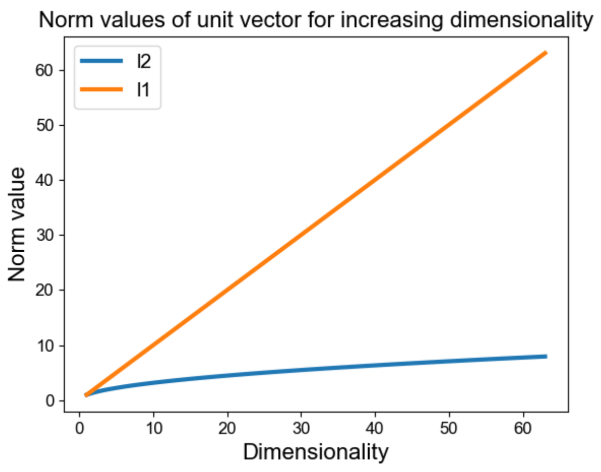

Figure 3: Visualisation of the l-2 and l-1 norm against increasingly high dimensional unit vectors

to the error and facilitating incremental updates. Historically, there has been some scepticism of the biological plausibility of this of this algorithm [6, 7]. This primarily because, in its standard applications it requires informational symmetry of weights and biases during the forward and feedback (backward) pass, which is biologically implausible. Another issue, is that BP is non-autonamous, since it requires loss information which is external to the network being trained. It is difficult to conceive of a way this would function with neural circuits alone.

In recent years there have been a number of papers offering biologically plausible implementations of the guided BP algorithm [8–10]. In the paper by Lillicrap et al [8], instead of using the forward pass weight matrices for the backward pass, each forward pass matrices is assigned a fixed randomly initialised backward pass matrix. During BP, the error is backpropogated through these random matrices instead of through the actual weight matrices. They show that, for a number of test cases this produces results that perform roughly equivalently to guided BP in terms of convergence, whilst solving the problem of information symmetry during feedback. More Recently, Lillicrap et al. explored how the cortex might realistically implement BP like algorithms, proposing the NGRAM hypothesis, which posits that the cortex uses local differences in activity states to drive synaptic changes, effectively approximating gradient descent [10]. They explore a number of existing learning algorithms that rely on NGRAMS and they note that many of these algorithms either approximately, or in some cases exactly reproduce gradients computed by BP. Supporting evidence of the NGRAM hypothesis appears shortly afterwards in the work of Song et al. which develops a biologically realistic BP algorithm that results in identical weight updates to guided BP, whilst employing only local plasticity of neurons and relying on an NGRAM based learning rule. Their FA-Z-IL algorithm is based on Predictive Coding networks trained with inference learning and has only slight computational overhead when compared to BP, whilst being equivalent to BP, requiring only local synaptic plasticity and being fully autonomous. They suggest that this mechanism could be implemented in the brain and also that it could provide inspiration for future development of neuromorphic computing [11].

### 1.4 Additional Modifications

When training an ANN using supervised learning, sometimes the loss function or architecture can mean that loss gradients primarily flow through a small number of neurons and exploding gradients may become an issue. This can result in numerical issues, such as huge weight updates, leading to poor convergance. Moreover it is also not biologically realistic, since in the brain there is a physical limit to the strength of synaptic connections. We can impose constraints on the size of gradients throughout the backpropogation process using gradient clipping, in which we simply clamp any gradient values above a physically realistic threshold to the threshold value.

In the brain there is evidence that sleep is important for consolidation and insight following learning [12]. Participants were tasked with performing cognitive tasks, which they could improve upon gradually or by gaining insight into a hidden rule. The study found that, subjects that slept between trails gained insight at a rate twice as high than subjects that did not sleep between trails. The Implication of these results is that during sleep the brain consolidates information stored during waking hours and may be able to strengthen existing connections and form new connections. This hypothesis is supported by evidence that the hippocampus facilitates memory replay during sleep [13, 14]. All of this put together implies that, the simulation of events experienced during the day during sleep, might possibly facilitate additional learning in the brain. If we were to relate this to the design of training datasets for ANN, we might posit that the creation of augmented data from existing data, by various transformations, might simulate some of this additional learning that the brain accomplishes during sleep.

## 2 Training networks with brain inspired changes on a perceptual decision making task

For this section I have selected the DualDelayMatchSample-v0 task from neurogym. See Figure 4 for an illustration of the task.

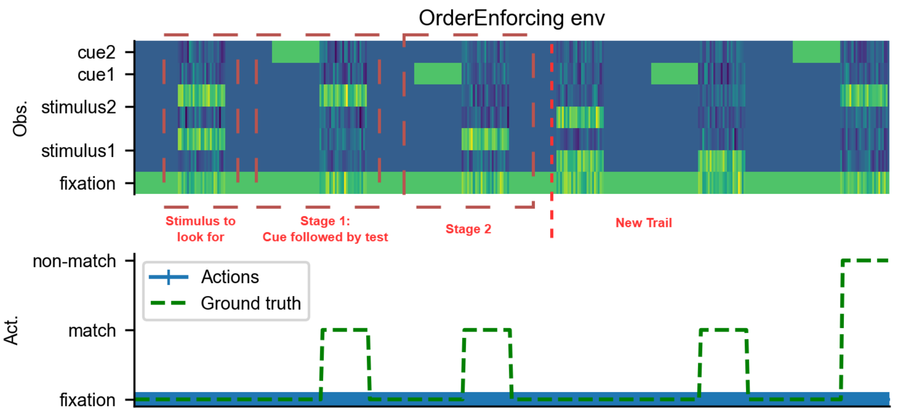

Figure 4: Example run of the Neurogym DualDelayMatchSample task. Each trail consists of two stimulus being displayed which the network should remember for the whole trail. Then there are two stages in which a cue indicates which of the initial stimuli to look for, followed by a test stimulus, which the network must either match to the cued initial stimulus or reject.

### 2.1 Training Three different RNN architectures on the task

If we consider the task in figure 4 further. the initial stimuli for each trail has to be remembered for the whole trail, whilst the cues and test stimuli only have to be remembered for as long as it takes for the network to converge to a stable prediction output. For this reason an architecture like the light-GRU might be the most appropriate here, as we can imagine one unit (or more) in the hidden layer encoding the initial stimuli and other units encoding the cue and test states for a shorter period. This would result in a learned gate vector with a longer time window for the initial stimuli units and shorter time window for the other units. I have chosen the light-GRU over the standard GRU for simplicity and to make it easier to make modifications to the architecture.

## Light-GRU Architecture

The recurrent update rule of the original light-GRU Architecture [15] is described by the following system of equations

$$\boldsymbol{z}_t = \sigma\big(BN(W_{zx}\boldsymbol{x}_t) + U_z\boldsymbol{h}_{t-1}\big) \tag{11}$$

$$\tilde{\boldsymbol{h}}_t = \text{ReLU}\big(BN(W_{hx}\boldsymbol{x}_t) + U_h\boldsymbol{h}_{t-1}\big) \tag{12}$$

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1-\boldsymbol{z}_t)\odot \tilde{\boldsymbol{h}}_t \tag{13}$$

where W terms denote feed forward weights and U terms denote recurrent weights. Equation 11 describes the update rule for the gate value of each hidden unit, which in turn controls the proportion of each hidden unit activation from the previous timestep is "remembered" in eq. 13. Equation 12 is the recurrent update rule for the proposed new hidden layer activations, $\tilde{\boldsymbol{h}}_t$ which are combined with the previous hidden layer activations in eq. 13. The $BN$ terms denotes the batch normalisation function, given by

$$BN(a) = \gamma \odot \frac{a-\mu_b}{\sqrt{\sigma_b^2+\epsilon}} + \beta \tag{14}$$

where $\mu_b$ and $\sigma_b^2$ are the minibatch mean and variance, respectively. The terms $\gamma$ and $\beta$ are trainable scaling parameters and the term $\beta$ removes the need for a bias term in equations 11 and 13. At evaluation time running estimates of the mean and variance are used instead of batch values. There is an element of biological realism to this computation, at least at evaluation time, as it has been observed that normalisation is a canonical neural computation found throughout the brain [16].

## ENU-Light-GRU Architecture

We can adapt the light-GRU architecture to be more biologically realistic by replacing the hidden layer artificial neurons with 2 layer ANNs, which I will coin Embedded Neural Units (ENUs), each of which closely emulates the behaviour of biological pyramidal neuron in the brain [1]. using the simple recurrent ENU network derived in equations 3-5, we have that the inputs to each ENU are given by

$$\hat{\boldsymbol{h}}_t = \text{ReLU}\big(W_{\hat{h}x}\boldsymbol{x}_t + \boldsymbol{b}_{\hat{h}}\big) \tag{15}$$

where $W_{\hat{h}x}$ maps input values to independent higher dimensional embeddings in $\hat{\boldsymbol{h}}_t$. We can incorperate this into eq. 12 in the following manner

$$\tilde{\boldsymbol{h}}_t = \text{ReLU}\big(BN(W_{h\hat{h}}\hat{\boldsymbol{h}}_t) + U_h\boldsymbol{h}_{t-1}\big) \tag{16}$$

where $W_{h\hat{h}}$ is defined as in eq. 6 in block diagonal form, so that units are independent. An additional note here is that during training it is important to mask the zero entries of $W_{h\hat{h}}$ so that they are not adjusted during backpropogation. This looks something like $W_{h\hat{h}} = \hat{W}_{h\hat{h}} \odot W_{\text{rec}}$ where $\hat{W}_{h\hat{h}}$ is a learnable parameter and

$$W_{\text{rec}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{17}$$

shown here for the case of two ENUs with length 3 input embeddings

## EI-Light-GRU with L2 activation regularisation

Another way in which we can adapt the light-GRU architecture to be more biologically realistic is by imposing Dales law on the hidden layer units, using an Excitatory-Inhibitory structure, as described in section 1.1. Additionally we can adjust the cost function to an L2 regularisation term on the hidden unit activations, penalising large and biologically unrealistic values during training, as described in section 1.2.

To implement EI recurrent connections in each recurrent step, the update rule for hidden activations in eq. 12 can be modified so that

$$\tilde{\boldsymbol{h}}_t = \text{ReLU}\big(BN(W_{hx}\boldsymbol{x}_t) + (U_h \odot U_{\text{rec}})\boldsymbol{h}_{t-1}\big) \tag{18}$$

where $U_{\text{rec}}$ is a fixed rectification matrix as defined in eq. 7. implemented in PyTorch, this looks like

```
    def create_ei_mask(hidden_size, e_size):
        mask = torch.ones(hidden_size, hidden_size) - torch.eye(hidden_size)
        # Making inhibitory columns negative
        mask[:, e_size:] *= -1
        # making a parameter in case we want to access it later and to explicitly
    state that it is not a learnable parameter
        mask = nn.Parameter(mask, requires_grad=False)

        return mask
```

with the reccurent update rule given by

```
    def recurrent_step(self, input, hidden):
        """light GRU timestep with E-I linear transformation"""
        z_t = self.gate_act(self.bn_z(input @ self.Wzx.T) +  hidden @ self.Uz.T)
        h_ungated = self.hidden_act(self.bn_h(input @ self.Whx.T) + hidden @ (self.Uh*
    self.mask).T)
        h_t = z_t * hidden + (1 - z_t) * h_ungated
        return h_t
```

To penalise large hidden unit activations, we can modify the cost function as in eq. 9, for use during model training. Implemented in Pytorch this looks like

```
    # cross entropy with l2 penalty on hidden layer activations
    def criterion(output_logits, labels, hidden_activations):
        cross_entropy = nn.CrossEntropyLoss()
        l2_penalty = l2_activation_coef * torch.mean(torch.linalg.vector_norm(
    hidden_activations, ord=2, dim=1))
        return cross_entropy(output_logits, labels) + l2_penalty
```

where we have modified the forward call to the RNN so that it returns hidden unit activations along side the output from the fully connected layer

```
    def forward(self, input, hidden=None, return_hidden=False):
        hidden_timeseries, _ = self.gru(input, hidden)
        logits = self.fc(hidden_timeseries)

        if return_hidden:
            return logits, hidden_timeseries
        else:
            return logits
```

## 2.2 Comparison of Model Performance

All three models were trained for 1000 epochs. During each training epoch, the model inputs and labels consisted of 32 batches of time series inputs corresponding to 2 independent trials. For all models an Adam optimizer was used, with an initial learning rate of 0.01 and an exponential learning rate scheduler, so that the final learning rate at the 1000th epoch was 0.0001. Figure 5 shows the convergence of the training loss and training accuracy for the different models. We can see from the figure that the training loss and accuracy both converged similarly for the different models, with the exception of the EI-light-GRU with l2 regularisation, which seems to have converged to a slightly worse minima. I did not feel the need to perform any validation during training for this task as the input data is low dimensional and each training batch is generated on the fly, so there is little risk of over fitting. The oscillations in the training loss and accuracy, may have arisen due to the fact that the majority of time steps have a“no action” label and there are only 4 periods in each training batch that require a “reject” or “accept” response, effectively biasing the cost function towards the “no action” state.

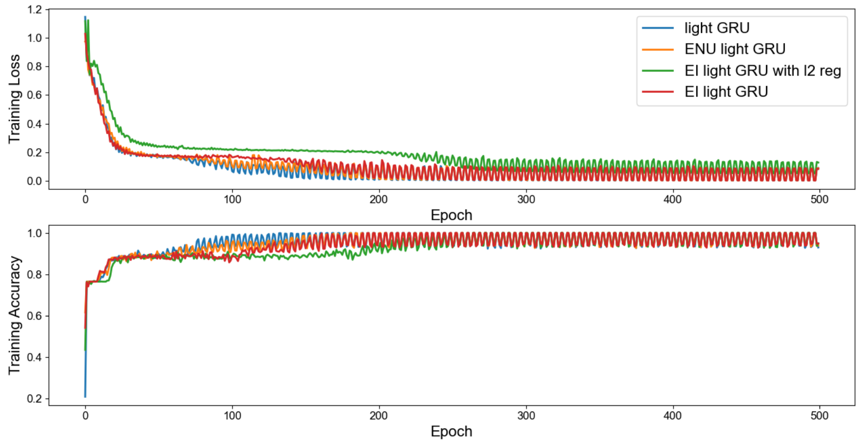

Figure 5: Training Loss (top) and accuracy (bottom) for the different GRU architectures against the number of epochs.

Having trained each of the models and achieved comparable performance metrics on the training data, we can visualise their response to a test trail by reproducing the task visualisation from figure 4. The results of this are shown in figure 6, which shows that accross the board, models are good at predicting the “no action” state, which makes sense given it contributes most to the loss function at train time. We see however, that models are not quite as good at predicting correct response during each test cue, especially the EI-light-GRU, which visibly fails to respond to half of the test cues in figure 6.

The output from the hidden state of each model is produced by a single fully connected linear layer, meaning we hope during training that the hidden state will produce recurrent embeddings that are linearly separable for each state. We can visualise the hidden layer activations in various ways to attempt to understand the dynamics of the models and to compare their behaviour.

First we can plot the maximum activation of each of the 64 neurons in the hidden layers in each model, throughout a test timeseries. If we do this for a sequence length of 1000 time steps, we get the stem plot shown in figure 7. We can see from this, that there appear to be some units in the light-GRU and ENU-light-GRU models that do not activate at all throughout the dynamics, suggesting that perhaps the hidden layer could be smaller for this task without negatively impacting performance. Furthermore, we can see that the effects of l-2 activation regularisation on the units of the EI-light-GRU model, as the maximum activations are lower than the other two unregularised models.

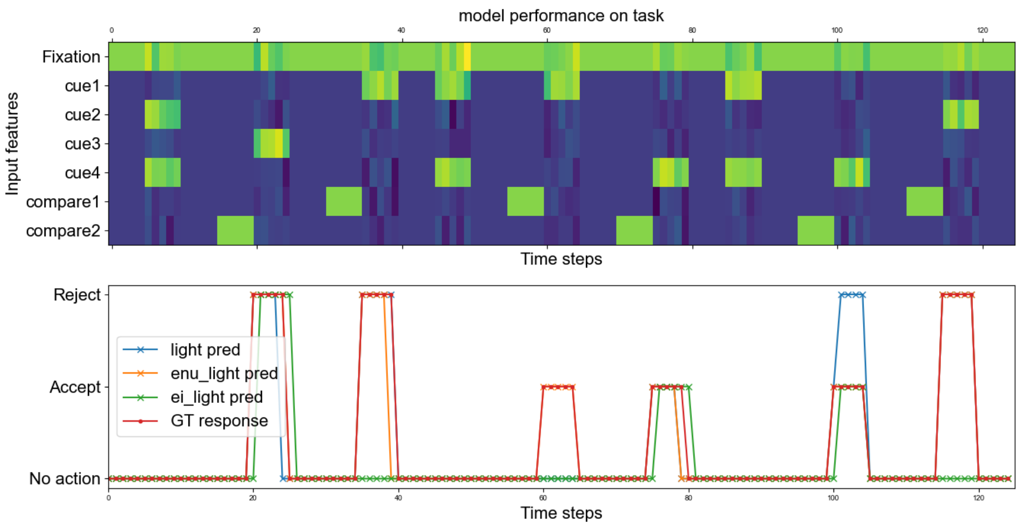

Figure 6: Comparison of trained model performance on the “DualDelayMatchSample-v0” neurogym task.

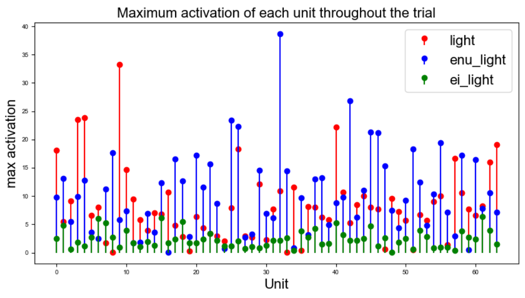

Figure 7: A comparison of the maximum activations over the full dynamics of the different network architectures.

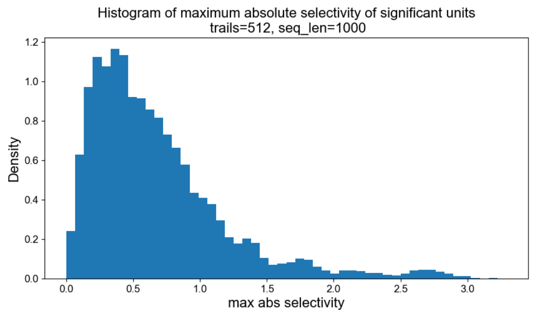

Figure 8: Maximum absolute selectivity histogram arising from a Monte Carlo simulation with 512 trials and sequence length 1000 ($\approx$ 20 task repeats). For each trial, the active/inactive and accept/reject selectivity is calculated, and the maximum absolute value for each unit across both metrics is taken. This figure indicates the proportion of units highly selective for a specific state.

Next, we can look at the selectivity of each of the units. Since selectivity is a binary comparison, we need to make two comparisons to cover the three output states. We can divide the actions into active (accept/reject) and inactive (do nothing) and then select the hidden layer activation corresponding to active predictions, giving $\boldsymbol{h}_{\textbf{active}}$ alongside those corresponding to inactive predictions $\boldsymbol{h}_{\textbf{inactive}}$. The sensitivity of each unit can then be calculated by

$$s = \frac{2(\mu_{\text{active}} - \mu_{\text{inactive}})}{\sqrt{\sigma^2_{\text{active}} + \sigma^2_{\text{inactive}}}} \tag{19}$$

which effectively quantifies the separation between the assumed normally distributed activations. Since activations can only be positive, a large positive selectivity in this case indicates the unit preferentially fires for active predictions, whilst large negative selectivity would correspond to inactive predictions. Notably, a unit could have a high mean active and inactive activation, but a low selectivity.

The results of calculating the active/inactive selectivity of each hidden unit in the light-GRU model, alongside the results of calculating the accept/reject selectivity, can be seen in figure 9. We can see from the figure that there are a small number of highly action and inaction selective units and about half of the units are not highly selective for action or inaction. However, if we look at the corresponding accept/reject selectivity, we see that many of the units that are not highly action selective are in fact highly accept/reject selective. We can also see that the units that were not activated in figure 7 are not selective in either comparison, which makes sense.

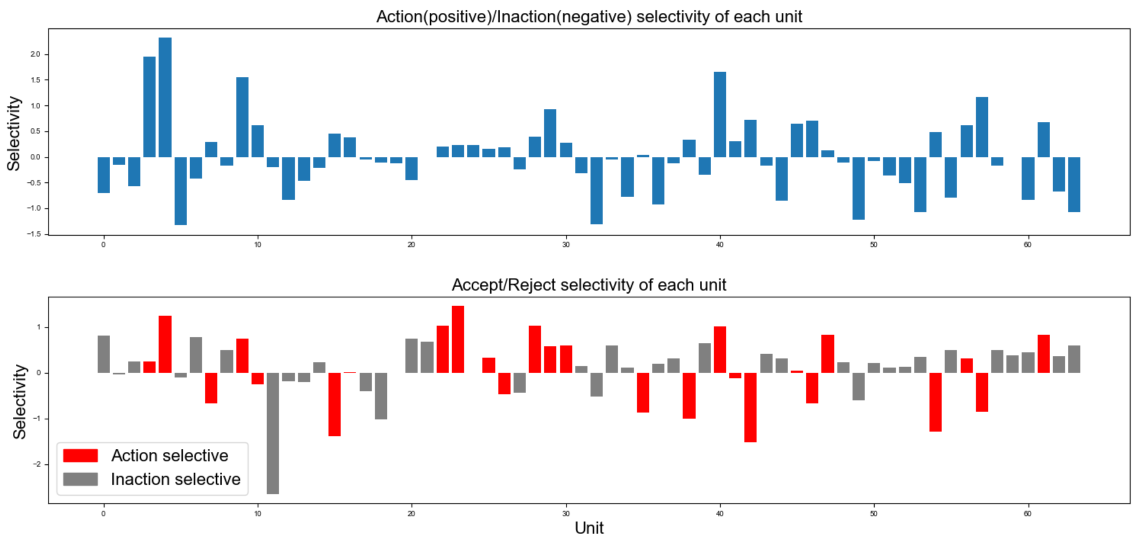

Figure 9: Selectivity of the hidden layer units of the light-GRU for two binary cases: Active or inactive and accept or reject

To get a better idea of the proportion of hidden layer units that are highly selective for a specific action state we can run a monte carlo simulation, by generating repeat selectivity results as in figure 9. To do this, for each mote carlo trial we can calculate the maximum of the absolute selectivity for each unit across the two selectivity metrics in fig 9. We can also remove any units that have a low maximum activation throughout the dynamics, as seen in fig 7, since they are either redundant, or don't activate in the given trail. The results of doing this for the light-GRU model with 512 monte carlo trials can be seen in figure 8. We can see from the resulting distribution, that whilst a large portion of the units are highly selective for a given action, approximately half have a max abs selectivity below 0.5, suggesting that their activation does not strongly correspond to a particular output state. Additionally, we can observe two humps at around 1.75 and 2.7 abs max selectivity, which suggest the presence of a small number of highly selective units that are consistent throughout the trials.

The next obvious step is to look at the time series activity of the most highly action selective hidden layer unit in each of the models. The results of doing this can be seen in figure 10, which reveals some very patterned activation dynamics. Spikes in the EI and standard light-GRU activations, seem to correspond to test cues, which are cues which require an immediate response of accept or reject. So it makes sense that these would be the most selective for changes from inaction to action. The ENU light-GRU activations seem to spike for the duration of the first compare cue pair in each trail, potentially encoding trial timing and again serving as a cue for accept/reject in response to the first test. Additionally, the effects of L2 regularisation can again be seen in the EI-light-GRU dynamics, which have smaller activation magnitude. The similarity between maximally selective units seen here, indicates that perhaps the models use similar encodings to solve the task, despite their varying architecture. The slight misfiring observed also indicates further room for improvement in the training.

The final thing we can do to visualise the dynamics is to perform dimensionality reduction on the hidden layer of each model using PCA and then visualise the trajectories in 2 dimensions. To do this, I generated a single batch of training data with a long sequence length, alongside a single batch of test data lasting a single task trail, each PCA model was fit to the hidden layer activations resulting

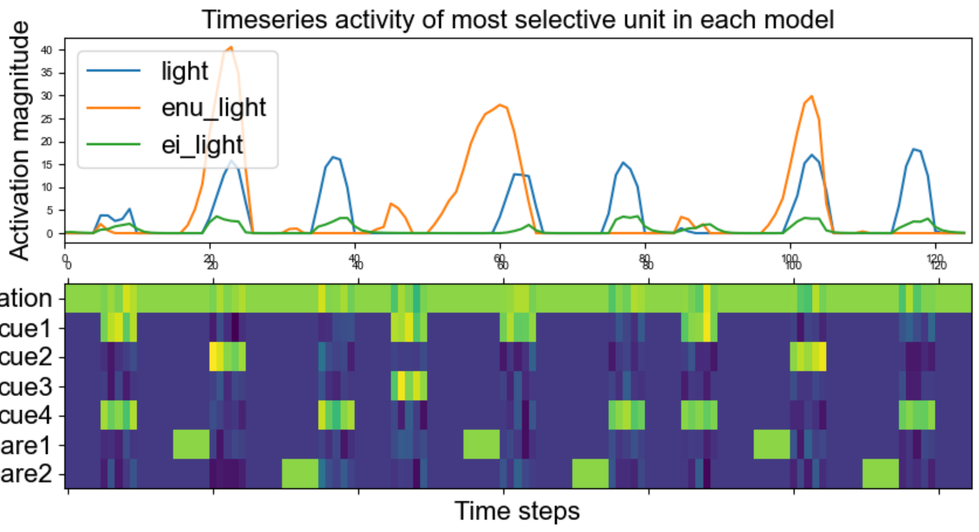

Figure 10: Timeseries activations of the most selective unit from each model (top) against the corresponding input vector timeseries (bottom). The label "activation magnitude" is potentially a little misleading, in fact all activations are positive or zero.

from the training data and then the hidden activations resulting from the test data were compressed and visualised in figure 11. We can quite clearly see that for each model, there is a region, primarily consisting of states, corresponding to inactive predictions and then at test time the state moves away from this region in notably distinct trajectories. The fact that this behaviour is clearly separable in the PCA compressed data, is a good indication that it is linearly seperable in the higher dimensions, as is often observed in the kernel trick [17], for example, and indicated by the fact that the first two principle components only account for between 37-44% of the variance in the high dimensional activation states. Interestingly, there is some clear variation in the behaviour of the compressed trajectories of each model, which I observed to be consistent across tests. Since PCA does not explicitly preserve the structure of the data, it is hard to draw conclusions from this, but it is certainly an indication that the different models have slightly different emergent dynamics.

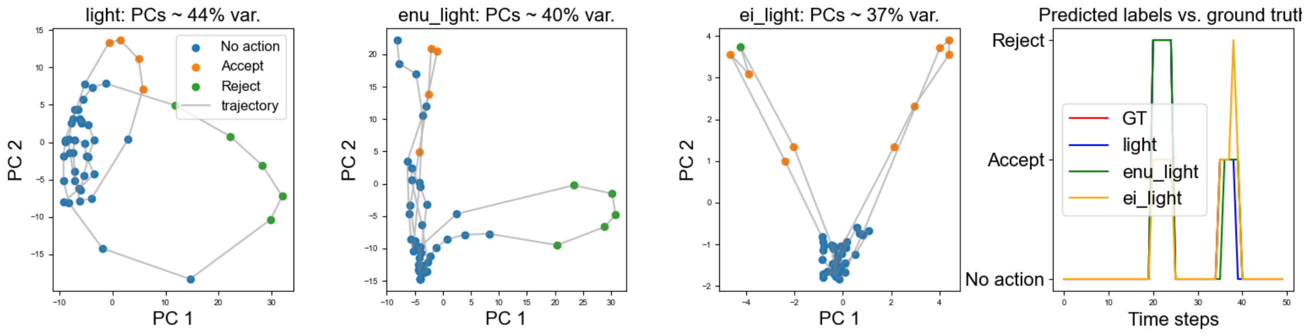

Figure 11: A comparison of PCA compressed hidden layer activation trajectories for the light-GRU, ENU-light-GRU and EI-light-GRU with l2 Reg. Each PCA model was fit to an task input sequence of 1000 timesteps and then shorter trajectories consisting of a single trail, were transformed to the first 2 PCs to produce the visualisations.

## 3 Repeated analysis with a different task

For the Second task I have chosen ContextDecisionMaking-v0, since this task requires the integration of slightly more complex input information, perhaps benefitting from the deeper structure of the ENU-light-GRU model. I learned from the previous task that the loss function was skewed towards the inactive state, since this was the dominant state in the training data. I also saw that the fluctuation in the training accuracy after the plateau in training was potentially caused by the short sequence length. To mitigate this, this time around I have increased the sequence length to incorporate more trails in each epoch and I have weighted the cross entropy loss based on the proportions of each label in the training data. In the brain you might think of this as a strong positive feedback being associated with a correct action and a small, or even non existent positive feedback being associated with correct inaction. I have also reduced the hidden layer size to 32, due to the observation that there were many units with low selectivity and/or that didn't activate in the previous task. Finally, I reduced the l-2 activation penalty coefficient by a factor of ten for the EI-light-GRU, so that it does not hamper training, as seen in the previous task.

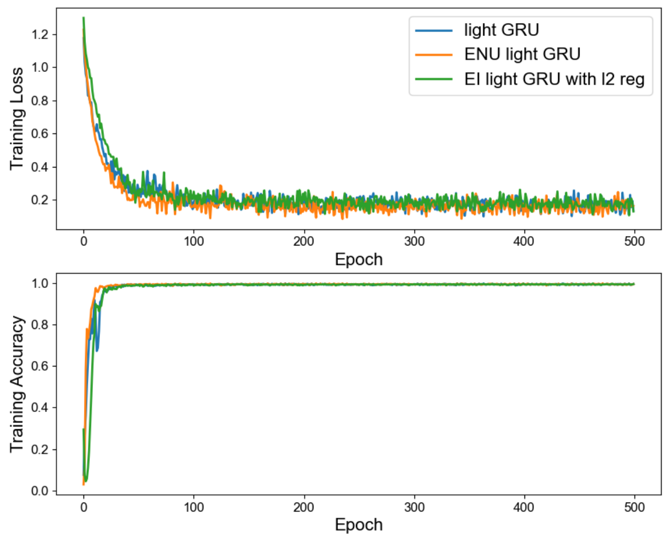

Figure 12: Training Convergence for task 2 over 500 epochs.

The training loss and accuracy results for each epoch can be seen in figure 12. It appears from this that my theories were correct, as the training accuracy plateaus in a much more stable manner this time. A test run of the task with associated model predictions can be seen in figure 13. This time we see much better choice accuracy when compared to the results from the previous results in figure 6, with all models producing ground truth predictions for this timeseries. Looking at the maximum unit activation in figure 14, we see a similar picture to the previous task, but notably this time the variance of maximum activation strengths is lower. The effects of the l2 activation regularisation can again be seen in the lower EI-light-GRU activations. As in the previous task, all three models exhibit similar training progress, with the ENU-light-GRU perhaps converging to an optimum slightly faster, although I wouldn't say this is outside of the margin of uncertainty associated with the random weight initialisations.

Looking at the maximum unit activations for each of the models in figure 14, we see a similar picture to the previous task, with the effects of l-2 regularisation seen to a lesser extent in the max EI-light-GRU unit activations. For this task though, there are notably fewer inactive units for the light-GRU and EI-light-GRU models, likely as a result of the reduced dimensionality of the hidden layer. We can still see quite a few inactive units in the ENU, light architecture however. This could be an indication that

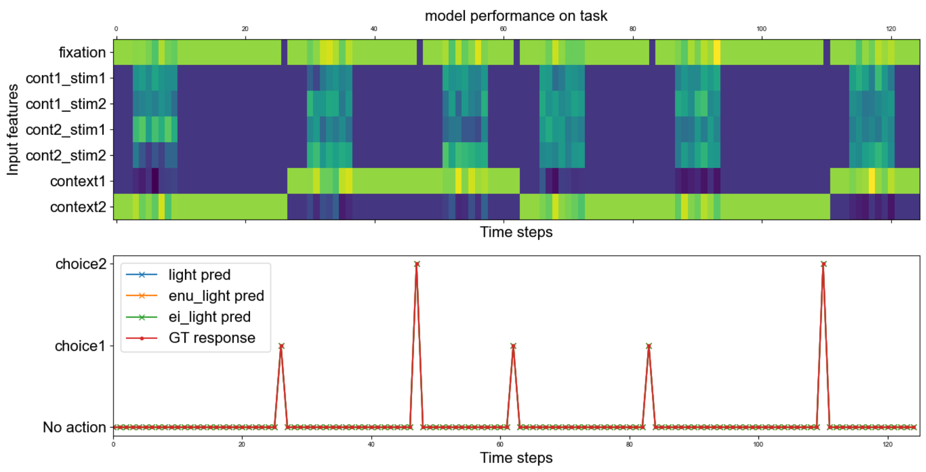

Figure 13: Performance comparison of the three trained models for the “ContextDecisionMaking-v0” neurogym task.

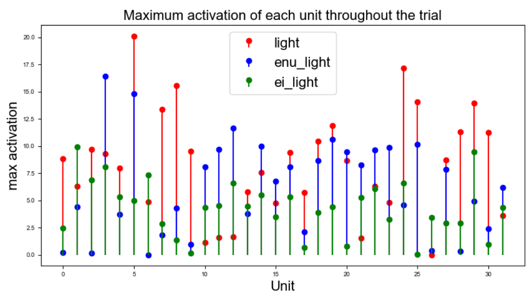

Figure 14: Maximum hidden unit activations for the three models for the second task.

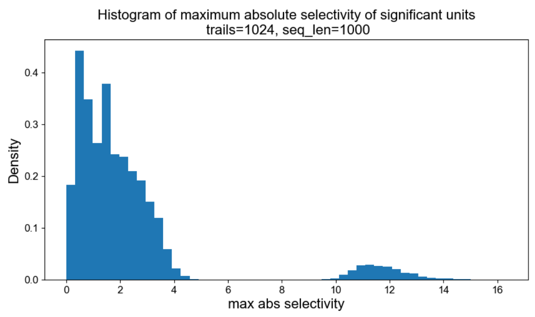

Figure 15: maximum absolute selectivity histogram for task 2, arising from a monte carlo simulation with 1024 trials. Double that of the previous task because of the reduced hidden layer size. See fig. 8 for an explanation of max abs selectivity.

the deep feed forward layer in the ENU model are compressing the input space slightly so that less of the non recurrent computation has to occur on the hidden layer. For example, the input contains two dimensions relating to context cues, which could easily be represented by one hidden unit, where high activations indicate the first context and low/zero activation indicate the second context. However, this trend was not observed in the first task, so the evidence is quite week for the ENU-light-GRU significantly compressing the hidden layer embedding.

Looking at the selectivity of the different units in the light-GRU model in figure 16, we see that there are just two units which strongly select for action and there are many more that select for choice 1 or choice 2. This makes sense, when you consider that action is prompted by gaps in the fixation in this task, whereas in the previous task, action was prompted by context dependent cues, which is slightly more complex. Looking at the distribution of maximum absolute selectivity in figure 15, we see a similarly left skewed distribution to the first task. The presence of an even more pronounced selectivity hump in this distribution confirms the reasoning from the previous trail as it is clear in this case that the hump is caused by the two highly action selective units in figure 16. The observation that even with the number of hidden units halved, still roughly half of the units in the distribution exhibit low selectivity suggests a deeper pattern might be at play than simply redundancy. Perhaps some recurrent hidden units are not used during output prediction, but instead are used in recurrent computations, such as the integration of stimulus information in this task, or comparison of cues in the previous task.

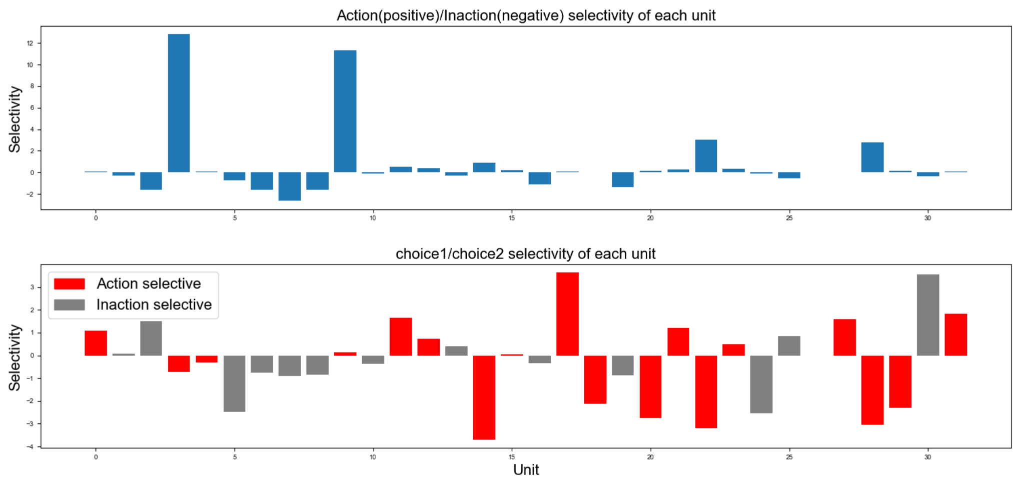

Figure 16: Hidden unit selectivity of the light-GRU for task two.

To conduct a more in depth look into the hidden unit behaviour we can select the most action selective unit for each model, as in the previous task. The results of this can be seen in 17, which clearly shows that the most action selective unit in each model spikes during gaps in the fixation, confirming the interpretation of the selectivity in figure 16. These spikes in activity almost certainly drive recurrent dynamics which result in a choice embedding and consistency in behaviour is observed both here and in the previous task. An additional observation from figure 17 is that the effects of l-2 activation regularisation can clearly be seen in the EI-light-GRU activations, which only increase significantly during fixation gaps. The other models seem to be a little bit more sensitive to noise during the stimulus periods.

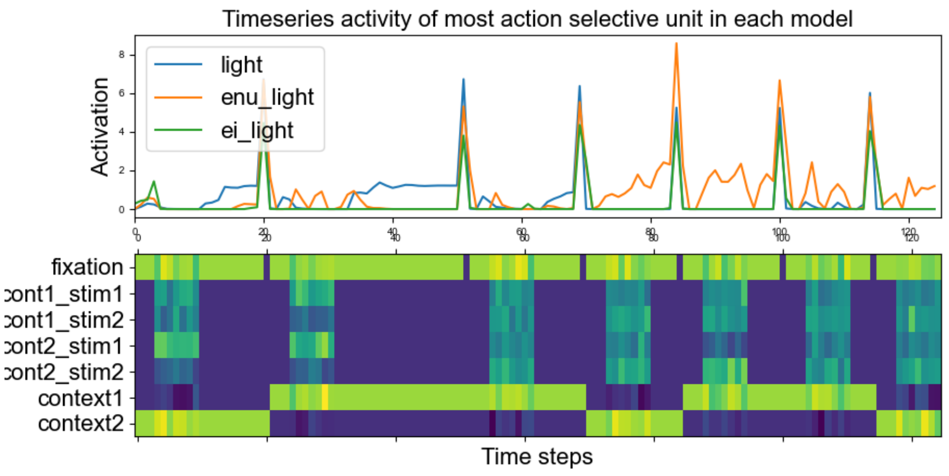

Figure 17: A comparison of the the timeseries activations for the most action selective units in each model (top), against the task2 input series, where each column corresponds to an input vector.

To extend the analysis beyond that of the first trial, we can plot the hidden unit activation timeseries for the most action selective, choice 1 selective and choice 2 selective units for each of the models, tested on the same trail. The results of this can be seen in figure 18. For this figure I have added prediction indicators as dotted lines to elucidate the behaviour of the units. We can see that for all three models the behaviour is similar. The action selective unit performs as in figure 17 and the c1 selective and c2 selective units have a kind of switching behaviour. This is most easily observed in the EI-light-GRU timeseries (bottom), for which the c1 selective unit (orange line) is always active in the dynamics and fixation prior to a choice 1 prediction. Likewise c2 selective unit spikes just prior to and during choice two predictions. This behaviour is more exaggerated in the dynamics of the other two models, likely due to the absence of L2 regularisation. Interestingly the c2 selective unit associated with the ENU-light-GRU model seems to be more active prior to predictions and actually dips during predictions, which is perhaps a kind of reset mechanism emerging from backpropogation. These results are very much consistent with our analysis for the first task, demonstrating consistency in the dynamics, between the models. This is interesting as it highlights the fact that is often the case in deep learning problems, data quality, data quantity and appropriate learning rules are often more important for learning than architectural changes.

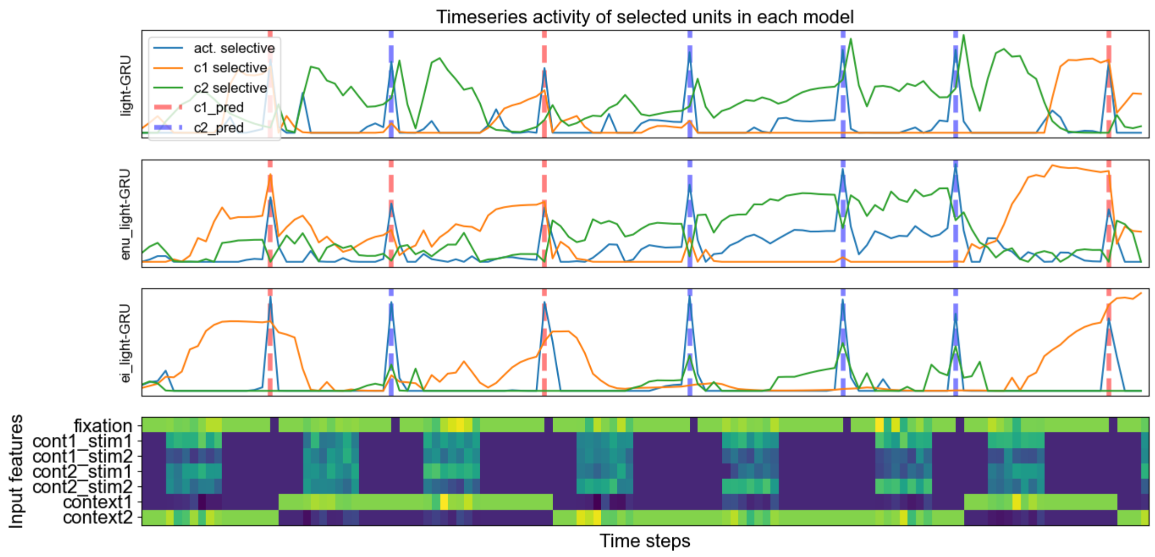

Figure 18: Hidden unit activation timeseries comparison, this time corresponding to the units that are most selective for each possible output prediction.

Finally, we can repeat the PCA analysis from figure 11 in the previous task. This time I had to compress the hidden activations to 3 dimensions instead of 2 to see proper separation in the projections. The results of this can be seen in figure 19. Again we can see clear separation between the inactive model states and the choice states, as in the first task. Notably, the percentage hidden layer variance accounted for in the first 3 PCs is higher in this task than in the previous, which was ≈ 40%. This could be simply due to the fact that these models only have half the dimensionality in the hidden layers, but it might also indicate that the latent space of this problem is slightly simpler than the previous. The idea that PCA compression might tell you something about the complexity of the task is evidenced by the fact that both in the previous task and in this task the percentage of accounted for variance in the principle components were fairly consistent across the different models.

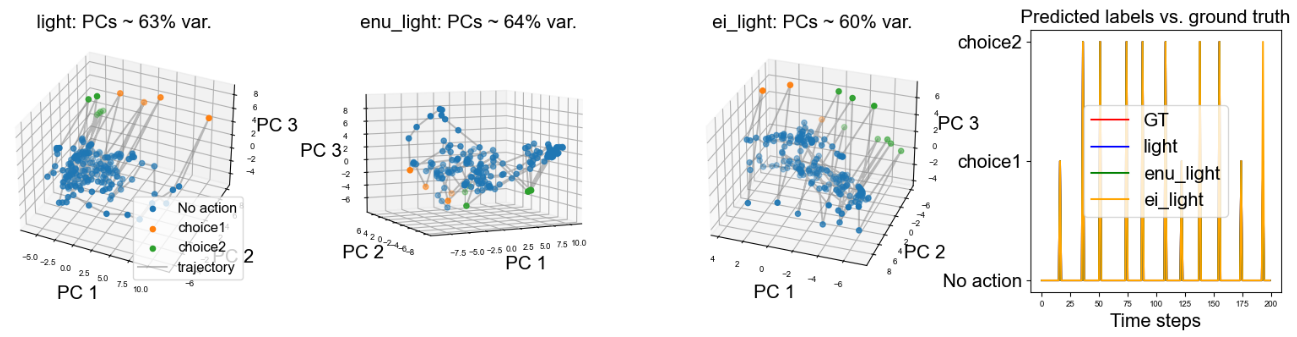

Figure 19: A Comparison of PCA compressed hidden layer activation trajectories for task 2. This time, each of the trajectories has been projected onto the first three principle components.

## 4 Conclusion

The main finding from this report, is that each of the models tested performed roughly equivalently across tasks. Additionally, learning rules and data quality during training seemed to have a much greater effect on the model performance than slight changes in model architecture. The biological implication of this is that neural networks are generally robust to architectural changes, supporting the idea of canonical local circuits in the brain, since very simple canonical computations can be repeatedly applied with appropriate feedback and learning mechanisms to achieve good performance on a variety of tasks [18]. We might also conclude that all of the biologically realistic changes we made are plausible precisely because they did not hamper performance. For example, in line with existing research we did not observe significant negative effects of imposing Dales law in the EI-light-GRU network [19], suggesting that the chemical limitation of synaptic firing that causes neurons to be either excitatory or inhibitory does not pose any constraints on the ability of the brain to learn effectively.

Additionally, we observed from selectivity histograms that a consistent proportion of units were selective, whilst the rest were non selective, implying that some units represent events/states, whilst others perhaps drive recurrent computation CITATION NEEDED. This pattern would be in line with the observation that certain neurons in the brain encode specific objects and rarely fire [20, 21], whilst others fire regularly, but appear to serve a computational purpose [22].

Considering the ENU-light-GRU architecture, I came to the conclusion throughout testing that the addition of an extra feedforward layer had little effect on performance for the tasks tested. This could be because both tasks had primarily time dependent behaviours and the input space was low dimensional. Additionally, one might reason that the light-GRU architecture already represents the superposition of a region of biological neurons. This is because, if we consider the weighted connections between inputs and individual hidden units, we end up with a distinct two layer feedforward network for each unit, which look like the ANN in figure 1. a vanilla RNN, with a hidden layer size of 32, could therefore be thought of as roughly approximating a brain region containing 32 neurons, each of which receiving one or more inputs from the input vector, which represents all of the inputs coming into that region.

Finally, we observed that l-2 regularisation in the EI-light-GRU model, acted to reduce the erroneous spiking of the hidden units tested in figures 10 and 17. One can imagine that in the brain this is extremely important, as erroneous spiking carries an intrinsic energy cost and synchronous erroneous spiking also occurs during seizures [23]. Perhaps in the brain some form of l-2 regularisation occurs during learning and is controlled by neuromodulator(s), which appears as the l-2 activation regularisation coefficient hyperparameter in the EI-light-GRU model. This would be inline with the speculation that neuromodulators in the brain are roughly equivalent to hyper parameters in many deep learning schemes [24].

## References

[1] Panayiota Poirazi, Terrence Brannon, and Bartlett W Mel. Pyramidal neuron as two-layer neural network. *Neuron*, 37(6):989–999, 2003.

[2] Nikola T Markov, P Misery, Arnaud Falchier, C Lamy, J Vezoli, R Quilodran, MA Gariel, Pascale Giroud, Maria Ercsey-Ravasz, LJ Pilaz, et al. Weight consistency specifies regularities of macaque cortical networks. *Cerebral cortex*, 21(6):1254–1272, 2011.

[3] John Carew Eccles. From electrical to chemical transmission in the central nervous system: the closing address of the sir henry dale centennial symposium cambridge, 19 september 1975. *Notes and records of the Royal Society of London*, 30(2):219–230, 1976.

[4] Xiaolong Jiang, Shan Shen, Cathryn R Cadwell, Philipp Berens, Fabian Sinz, Alexander S Ecker, Saumil Patel, and Andreas S Tolias. Principles of connectivity among morphologically defined cell types in adult neocortex. *Science*, 350(6264):aac9462, 2015.

[5] Jascha Achterberg, Danyal Akarca, DJ Strouse, John Duncan, and Duncan E Astle. Spatially embedded recurrent neural networks reveal widespread links between structural and functional neuroscience findings. *Nature Machine Intelligence*, 5(12):1369–1381, 2023.

[6] Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

[7] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[8] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1): 13276, 2016.

[9] Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566–22579, 2020.

[10] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[11] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[12] Ullrich Wagner, Steffen Gais, Hilde Haider, Rolf Verleger, and Jan Born. Sleep inspires insight. *Nature*, 427(6972):352–355, 2004.

[13] Daoyun Ji and Matthew A Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature neuroscience*, 10(1):100–107, 2007.

[14] Pierre Maquet. The role of sleep in learning and memory. *science*, 294(5544):1048–1052, 2001.

[15] Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018.

[16] Matteo Carandini and David J Heeger. Normalization as a canonical neural computation. *Nature reviews neuroscience*, 13(1):51–62, 2012.

[17] Eric Kim. Everything you wanted to know about the kernel trick. *URl: http://www. eric-kim. net/eric-kim-net/posts/1/kernel_ trick. html*, 2013.

[18] Rodney J. Douglas and Kevan A. C. Martin. 15canonical cortical circuits. In *Handbook of Brain Microcircuits*. Oxford University Press, 08 2010. ISBN 9780195389883. doi: 10.1093/med/9780195389883.003.0002. URL https://doi.org/10.1093/med/9780195389883.003.0002.

[19] Jonathan Cornford, Damjan Kalajdzievski, Marco Leite, Amélie Lamarquette, Dimitri M Kullmann, and Blake Richards. Learning to live with dale's principle: Anns with separate excitatory and inhibitory units. *bioRxiv*, pages 2020–11, 2020.

[20] R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

[21] Gabriel Kreiman, Christof Koch, and Itzhak Fried. Imagery neurons in the human brain. *Nature*, 408(6810):357–361, 2000.

[22] Eric I Knudsen, Sascha du Lac, and Steven D Esterly. Computational maps in the brain. *Annual review of neuroscience*, 10:41–65, 1987.

[23] PA Robinson, CJ Rennie, and DL Rowe. Dynamics of large-scale brain activity in normal arousal states and epileptic seizures. *Physical Review E*, 65(4):041924, 2002.

[24] Kenji Doya. Metalearning and neuromodulation. *Neural networks*, 15(4-6):495–506, 2002.