UTRECHT UNIVERSITY

MASTER'S THESIS ARTIFICIAL INTELLIGENCE

# Predictive Coding as an Emergent Phenomenon

*Author:*
Elgar DE GROOT *(6557260)*

*Supervisors (Utrecht University):*
dr. David TERBURG
dr. Chris JANSSEN
*Supervisors (Radboud University):*
dr. Tim KIETZMANN
dr. Nasir AHMAD
prof. dr. Marcel VAN GERVEN

August 24, 2020

# Abstract

Predictive coding is a prominent theory in neuroscience for understanding neural processing in the brain. The theory states that the brain constantly predicts incoming sensory stimuli and improves its predictions by calculating the error between the predictions and the actual stimuli. It is still an open question whether this theory requires hard-wired neural circuitry or whether it could be an emergent phenomenon. Here, we explore whether predictive coding emerges in recurrent neural networks that receive predictable stimuli in the form of image sequences, when they are trained in an unsupervised way to reduce network activity. We show that these networks can successfully learn to predict future stimuli. Without imposing architectural constraints we see that a natural distinction emerges between units that calculate prediction errors and units that generate predictions. These findings suggest that these properties of predictive coding may be emergent phenomena in neural networks that efficiently encode predictable stimuli.

# Contents

# List of Figures

# 1 Introduction

Neuroscience strives to find mechanistic explanations for neural processes. Advances in deep learning have enabled a new way of investigating neural models of the brain. However, they most often rely on supervised learning from labelled data which is not thought to be a biologically plausible learning method. As infants for example, humans learn to distinguish different objects and sounds in their surroundings without supervision. How exactly the brain picks up on the statistical regularities of its environment is still an open question.

One prevalent theory in neuroscience that could explain this is "predictive coding"[1–10]. Predictive coding posits that the brain is constantly trying to predict future sensory signals[11,12]. By comparing the prediction with actual sensory stimuli the brain can calculate the prediction error and in turn improve the prediction. In order to generate an accurate prediction the brain will need to learn about the objects and features in the world. Recent computational models have shown the efficacy of predictive coding[8,10]. There are several models of how the brain could implement such a strategy[4,7,13], although it is uncertain whether the structures necessary for the implementation of predictive coding have to be explicitly incorporated and have been found by nature via evolutionary processes, or whether they can emerge from other more elementary elements such as efficient coding and recurrent processing.

Another influential theory that is related to predictive coding is "efficient coding". This theory can perhaps best be substantiated by two observations: natural signals are highly redundant[14,15], and neural processing is metabolically very expensive[16,17]. The theory postulates that one of the key tasks of sensory processing is redundancy reduction[18,19]. Predictive coding can also be understood as a redundancy reducing principle[3], specifically in a dynamic environment, because it is removing the predictable and therefore redundant information from the input signal and only forwarding the error of prediction. Here, we ask whether a neural system that optimises for efficient coding in a dynamical environment will exhibit features that are often associated with predictive coding, such as error calculating units and prediction units. The key idea is that in order to effectively reduce the redundancy in a temporal signal, the network will need to use past input to anticipate the activity from future input.

We train recurrent[1] artificial neural networks to minimise neural activity. As stimuli we use predictable sequences of handwritten digits. The caveat here is that while minimising activity our model is not allowed to ignore the input activity and can only affect it through recurrent connections, either through inhibition or excitation. We observe that our models learn to inhibit the patterns of activity that are predicted by the statistical structure of the input, based on previous input. We show that even though we do not explicitly model prediction and error units, our population of artificial neurons naturally divides into two distinct groups performing either prediction or error calculations. This supports the notion that predictive coding is an emergent phenomenon in recurrent networks that optimise for efficient processing.

[1] As opposed to feedforward networks where the signal flows from input to output, recurrent networks can influence their own internal state in a temporal sequence.

# 2 Background

## 2.1 Supervised vs. Unsupervised Learning

The field of artificial intelligence has made great strides in recent years, and deep learning in particular has been very successful. Examples are the success of deep convolutional networks on image classification tasks such as ImageNet, or the deep learning and reinforcement learning based AlphaGo by DeepMind that beat some of the top Go players[20–22]. This success can be attributed for a large part to the effectiveness of stochastic gradient descent to propagate errors through layers of artificial neurons to update their weights and improve performance, also called backpropagation. Training a neural network in this manner usually requires a vast amount of labelled data. While acquiring labelled data is one problem, another problem is that this type of learning is not very biologically plausible. Although there have been several attempts to find a biologically plausible implementation of backpropagation[23–25], a remaining issue is that labels provide only a small amount of information as error signal. Even for a categorisation task with 1000 categories (e.g. ImageNet) this is still less than 10 bits. Consequently a lot of samples are needed in order to learn effectively. Reinforcement learning, or trial and error, might naively seem like a more natural learning method, but suffers from the same problem as it provides merely one bit of information every trial (e.g. success or failure). Reinforcement learning, on its own, requires a huge amount of trials (e.g. millions to learn drive a simulated car[26]), of which a lot will be failures. This is not what we expect from humans, or indeed find acceptable. In learning to drive a car for example, failure can lead to death. There might certainly be a biological justification for both methods. However, they alone do not seem to be enough to explain the learning capabilities of the brain. For instance, infants learn to distinguish different objects in the images they see and learn to recognise words in the sounds they hear[27], without obvious supervision. Assuming this knowledge is not innate, but learned (which we will not further debate here as it is out of the scope of this work), how could the brain achieve this?

## 2.2 Predictive Coding

A prevalent theory originating from neuroscience that could be used to explain this is predictive coding[1,11,12]. This theory postulates that the brain actively constructs an internal representation of the world it inhabits by generating predictions and testing them against what is perceived. In this way, the input from the senses effectively serves as the training signal. Perhaps the biggest problem for unsupervised learning is how to extract useful information from the incoming signal. And before one can answer that we first need to know what is meant by 'useful'. In the predictive coding theory this is simply the information that is predictive of future stimuli. It is for instance useful to recognise objects since their appearance will remain the same in the future and is thus predictable, even if the object is in another place. It is still unclear how predictive coding might be implemented by the brain. It is a very general theory, and there are many models that implement it in various ways[4,5,7–9,11]. However, most share the same important elements: internal representation units, which can generate predictions, and prediction error units, which calculate the error between the prediction and the actual input signal[8,11,12,28]. The question remains that if the brain implements some form of predictive coding, whether the necessary architecture is hard-wired (e.g. through evolutionary processes) or whether it is an emergent phenomenon of recurrent neural systems in a dynamic environment, that might be optimised for other objectives, such as energy efficient processing.

### 2.3 Efficient Coding

Neural processing is very metabolically expensive. While the brain accounts for only 2% of the body mass, it uses about 20% of the resting metabolism[16,17]. It is therefore of importance that the brain processes information efficiently. Furthermore, sensory signals are highly redundant[3,14]. This is because nearby events in time and space tend to correlate. For instance, nearby pixels in natural images are often similar in colour, and pixels don't quickly change colour from moment to moment[3]. Information about one event will often tell you a lot about another event, or in other words are statistically dependent on one another. By removing this redundancy, one can encode the same information with less bits of information. For our sensory processing this is advantageous as it saves energy. There are several information theoretic methods for reducing the redundancy in a signal. Principal component analysis (PCA) reduces the correlation in a signal by removing pairwise covariances and thereby extracting the 'principal components' of the signal. However, there is no strong evidence to suggest that this is the strategy employed by our brain[29]. A stronger decorrelating approach is independent component analysis (ICA)[30–32]. This method aims to maximise the statistical independence between the units that represent the input signal, by minimising the mutual information between units. The results seem to match well with the observed response patterns of cells in the early visual cortex suggesting that the brain maximises for statistical independence between representational neurons[33]. A related method is sparse coding which assumes that a signal can be represented by only a small number of units[34–36]. It works by finding a set of units that together can represent incoming signals effectively, while each unit is as sparsely active as possible. The two objectives of efficient coding are very clear in this method. The first is maximising the information from the input signal, the other is minimising the activity of the outgoing signal. Predictive coding can also be interpreted as an efficient coding method[3,37]. Statistical dependencies between events mean that they are predictive of one another. So by removing the predictable, one is removing statistical dependencies. What is left, that which is not predictable, is the prediction error. Regardless of the method, the brain appears to be capable of efficiently encoding incoming signals. Does this happen by means of a hard-wired predictive coding strategy? Or is it capable of prediction, because it uses an efficient coding strategy?

### 2.4 Excitatory-Inhibitory Balance

Related work by Denève and collaborators[38,39] shows that a biologically plausible method of efficient coding can learn to predict an incoming sequence suggesting that the latter option has some weight. They showed that a spiking neural network that keeps a tight excitatory-inhibitory balance (E-I balance) of signals between neurons can represent an input sequence with high precision and with minimal neural activity. Excitatory signals coming from the senses convey information from the outside world to our brains. As noted before these signals are highly redundant and therefore metabolically costly. Inhibitory interneurons suppress neural activity and thereby reduce the metabolic cost. However, this also reduces the amount of information in the signal. By keeping a tight balance between these excitatory and inhibitory signals the interneurons are prevented from *over*-inhibiting. In other words they should be precise about it. In our work we use a similar training objective. We minimise the presynaptic activity (i.e. the activity before the application of an activation function), which can be interpreted as a way of keeping a tight E-I balance in a recurrent rate-based[2] network. In this work we investigate whether these principles will allow the emergence of properties otherwise often explicitly modelled in predictive coding

[2] In rate-based neural networks neural activity models the average firing rate of a biological neuron (or a population of), as opposed to the membrane potential as is done in spiking neural networks.

schemes, namely units that calculate the prediction error (i.e. error units) and representation or prediction units.

3

# Methods

Our aim is to investigate whether predictive coding could be an emergent phenomenon, and more specifically whether prediction and error units emerge. We take inspiration from the predictive coding network called "PredNet" by Lotter et al.[8] and simplify this architecture in a number of ways to create more general, less architecturally constrained networks with less to no assumptions in regards to predictive coding elements such as error or prediction units. Similarly to their work we use a rate-based neural model where the unit activity models the firing rate of a neuron. Specifically, we use rectified linear units (ReLUs)[3] meaning that the firing rates are linear and always positive. The network weights can be both positive and negative, where positive weights have an excitatory effect and negative weights an inhibitory effect. The weights are updated using gradient descent to minimise neural activity. Crucially, the networks have recurrent connections to allow for temporal correlations in the data to be learned. Sequences of images are used as input stimuli, where pixel intensities are interpreted as the firing rates of sensory neurons. We create two network architectures, each with a layer that integrates the sensory input with internal connections. The first network architecture is similar to the PredNet model, but without explicitly modelled error units. The second network architecture, which we informally call the Bathtub model, does not explicitly model error units or prediction units.

## 3.1 Simplified PredNet Model

As we said above, our first model architecture is inspired by the PredNet model but simplified in a number of ways. However, the fundamental components have remained mostly the same. The PredNet architecture consists of stackable modules that each have the same components: an input layer (omitted in the bottom module), an error layer, a representation layer, and a prediction layer. The input, representation, and prediction layers are using convolutional operations[40], where each artificial neuron is only connected to a subset of neurons in the previous layer and so integrate more information going up the hierarchy. For simplicity, we chose to use fully connected layers instead. This means that there is no need for spatial integration and thus a hierarchy is less needful. Our simplified model will have only one module, which means that there will be no input layer. Equations 1 through 3 show the network state update formula for an original single module PredNet model. While the representation layer in the original PredNet model uses convolutional LSTM units, we instead created two versions of our model with different types of recurrent units: one with regular LSTM units[41] and one with a simple recurrent units (SRN) based on the Elman network[42]. The LSTM version is closer to the original architecture, but LSTM units are considerably more complex and since we are interested in emergent dynamics with minimal assumptions we also tested a simpler recurrent architecture that is closer to the Bathtub model described in subsection 3.2.

$$\mathbf{p}_t = \mathrm{ReLU}(\mathrm{Conv}(\mathbf{R}_t)) \tag{1}$$

$$\mathbf{E}_t = [\mathrm{ReLU}(\mathbf{x}_t - \mathbf{p}_t); \mathrm{ReLU}(\mathbf{p}_t - \mathbf{x}_t)] \tag{2}$$

$$\mathbf{R}_t = \mathrm{ConvLSTM}(\mathbf{E}_{t-1}, \mathbf{R}_{t-1}) \tag{3}$$

The main difference between our model and the PredNet model is in the error unit layer. One of our objectives for this model is to test whether error units will emerge naturally (see

[3] Rectified linear units transmit only the positive part of their input $f(x) = x^+ = max(0, x)$

subsection 4.2). In the predictive coding framework, error units calculate the difference between the prediction generated by the network and the actual input. In the PredNet architecture this process is explicitly modelled by subtracting the prediction generated by the prediction layer from the input and splitting the positive and negative errors into two distinct populations (see Equation 2). In our model the error units are not explicitly modelled and this layer simply takes the sum of the input, $\mathbf{x}_t$, and the feedback from the prediction layer, $\mathbf{p}_t$, followed by a ReLU activation. See Equations 4 through 6 for the complete set of network state update formula for our simplified PredNet model.

$$\mathbf{p}_t = \mathbf{W}^{\mathbf{p}}\mathbf{r}_t \tag{4}$$

$$\mathbf{e}_t = \text{ReLU}(\mathbf{x}_t + \mathbf{p}_t) \tag{5}$$

$$\mathbf{r}_t = \text{LSTM}(\mathbf{e}_t, \mathbf{r}_{t-1}) \qquad \rightarrow \text{LSTM version} \tag{6}$$

$$\mathbf{r}_t = \text{ReLU}(\mathbf{W}^{\mathbf{e}}\mathbf{e}_t + \mathbf{W}^{\mathbf{r}}\mathbf{r}_{t-1}) \qquad \rightarrow \text{SRN version} \tag{7}$$

The loss function in the original PredNet model is the total firing rates of the error neurons, where negative activation is interpreted as the firing rate of neurons that calculate the negative errors, and vice versa. Here, we do not explicitly model the error neurons and take a more general approach where the loss function is the activity of all neurons. Since we use rectified units there is no negative activity. However, the connection weights can be negative resulting in inhibitory presynaptic activity, while positive weights result in excitatory presynaptic activity. To account for over-inhibition, the loss function is the total of this presynaptic activity of all neurons. From a biological perspective this is perfectly justifiable as it is widely accepted that the interaction between presynaptic and postsynaptic activity is an important factor in neural plasticity and learning. It is similar to the E-I balance method used by Denève[38] and can be interpreted as a Hebbian-like update rule where inhibitory connections are strengthened when neurons fire together and weakened when they don't fire together, and vice versa for excitatory connections. Explicitly, the loss function is the mean of the absolute values (i.e. the L1 loss) of the presynaptic activity of the network units at every timestep. For simplicity, the LSTM units are considered to be single units. The networks are implemented and trained using PyTorch[43] and the source code can be found at: https://github.com/elgar-groot/EmergentPredictiveCoding.

### 3.2 The Bathtub Model

The main objective is to explore emerging properties of predictive coding in a network with minimal architectural assumptions. To this end, we created a simple recurrent network where each unit is wired in the same way and thus without an intrinsic distinction between any of the units. We informally call this the Bathtub model as it resembles a bathtub of freely connected neurons. The network contains a set of $n$ fully connected, recurrent units. This means that each unit is recurrently connected to all the units (including itself). Each units is also connected to the input layer, with a fixed weight of 1. The network activity (or hidden state), $\mathbf{h}_t \in \mathbb{R}^n$, is a function of the previous network activity ($\mathbf{h}_{t-1}$) and the input, $\mathbf{x}_t \in \mathbb{R}^m$. More specifically, the hidden state $\mathbf{h}_t$ is the sum of the input $\mathbf{x}_t$ and a weighted sum of the previous hidden state $\mathbf{W}\mathbf{h}_{t-1}$ followed by a ReLU activation function. The weight matrix, $\mathbf{W} \in \mathbb{R}^{n\times n}$, is initialised to random uniform values between -1 and 1, scaled by $\sqrt{\frac{1}{n}}$. The number of hidden units must be greater than or equal to the input dimensionality, $n \geq m$, to encompass the complete input. If the number of hidden units is greater than the input ($n > m$), then the input vector is padded with zeros prior to the calculations in order to match the hidden state vector dimensionality. This means that technically there is a bijective (one-to-one and onto) mapping between the hidden state and the input vector, but this is practically an injective (only one-to-one) whereas some of the elements in the input vector will always be zero. We like to note here that if the input vector has positions that are zero in all examples of the training data the effective number of ’free’ network units can be greater than $n - m$. Equations 9 and 10 show the update formula. Like in the simplified PredNet model, the Bathtub model is trained to minimise the mean absolute presynaptic activity. This is formalised in Equation 11.

$$\begin{aligned} \mathbf{p}_t &= \mathbf{W}\mathbf{h_{t-1}} \quad (8) \\ \mathbf{a}_t &= \mathbf{p}_t + \mathbf{x}_t \quad (9) \\ \mathbf{h}_t &= \mathrm{ReLU}(\mathbf{a}_t) \quad (10) \end{aligned}$$

$$Loss = \sum_t \frac{1}{n} \sum_i^n |a_t^i| \quad (11)$$

### 3.3 Separating Prediction and Error Units

To find a distinction between prediction and error unit populations in the Bathtub model, we divide the test set into $n$ sequences of length 9 and run each separately through a Bathtub model. After the last sample of a sequence we record the excitatory presynaptic activity from both the input and the recurrent feedback and average them over all sequences, which we will denote as $\langle \mathbf{x}^+ \rangle$ and $\langle \mathbf{p}^+ \rangle$. Algorithm 1 describes the calculations in more detail. By taking the difference between $\langle \mathbf{x}^+ \rangle$ and $\langle \mathbf{p}^+ \rangle$ we can divide the population of units in the Bathtub model between input driven units, $[\langle \mathbf{x}^+ \rangle - \langle \mathbf{p}^+ \rangle] > 0$, and recurrently driven units, $[\langle \mathbf{x}^+ \rangle - \langle \mathbf{p}^+ \rangle] < 0$.

**Algorithm 1** Calculating excitatory presynaptic activity from input and recurrent feedback

```
Require: $S$                              ▷ the set of $n$ sequences from the test set
$x^+ \leftarrow 0$
$p^+ \leftarrow 0$
for $X$ in $S$ do
    $x_t \leftarrow 0$
    $h_t, a_t, x'_t \leftarrow 0$
    for $t = 1$ to 9 do
        $x_t = X[t]$
        $x'_t = x_t$ padded with 0's to match size $h_t$
        $p_t = h_t \cdot W$
        $a_t = p_t + x'_t$
        $h_t = ReLU(a_t)$
    end for
    $x^+ + ReLU(x'_9)$
    $p^+ + ReLU(p_9)$
end for
$\langle x^+ \rangle = x^+ / n$
$\langle p^+ \rangle = p^+ / n$
```

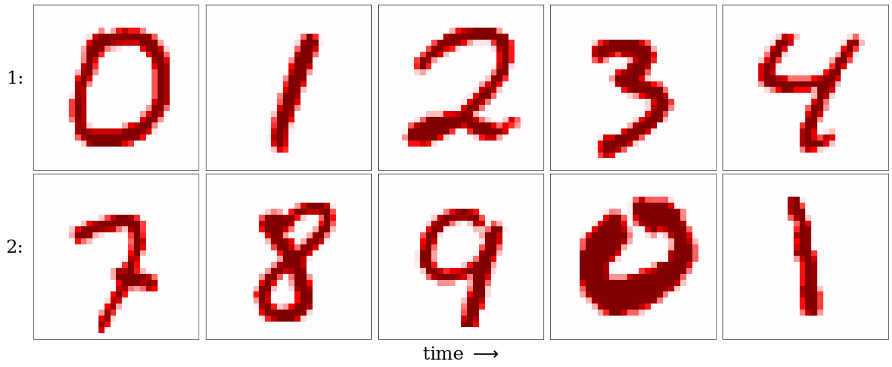


Figure 1: Example MNIST sequences of 5 images long

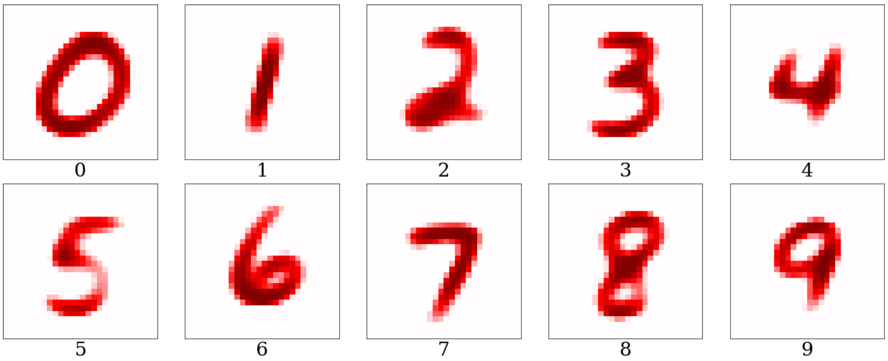
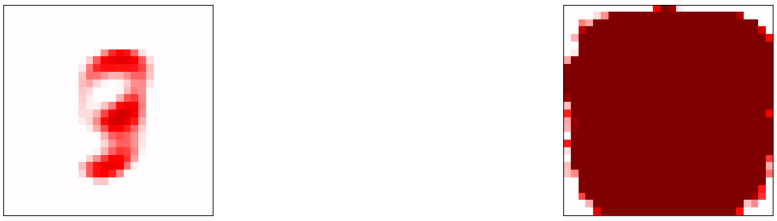


Figure 2: Median pixel intensities per category in the training set.

### 3.4 MNIST Sequences

As input data we use sequences of images drawn from the MNSIT database of handwritten digits[44]. These images are 28-by-28 pixels in size, with pixel intensities ranging from 0 to 1. The database has a training set of 60K images and a test set of 10K images. There are approximately 6K and 1K images per category in the training set and test set respectively. Each set of images can be divided into 10 categories, one for each digit. Sequences are generated by choosing an ample set of random digits as starting points. Digits are appended to each sequence until the desired sequence length is reached. The value of the digits increase by one each step, except after a nine in which case the sequence will continue with a zero. The sequence length can be chosen in advance, but all sequences will have the same length. The sequences are organised into batches. For all digits we then choose a random image from the MNIST database in the category that corresponds to that digit. The images are drawn without replacement and the process is stopped when there is no image available for the next digit. Incomplete batches where not every digit has an accompanying image are discarded. See Figure 1 for example sequences. With this procedure each image is used approximately once, but never more than once each epoch. Due to the stochasticity the amount of batches in each epoch may vary.

Since the images are chosen randomly from each category, the only predictable factor is that the succeeding image will be one of the images in the next category. We will interpret the pixel intensities of input images as (part of the) presynaptic activity for the first network layer and since



(a)

(b)

Figure 3: (a) Median pixel intensities across all 60K images in the training set. (b) Summed pixel values of all images in the training set. White denotes a total intensity of 0 whereas dark red denotes that the total intensity is >1. There are a total of 67 white pixels, with another 40 with a total intensity between 0 and 1 (the lighter shades of red).

we will minimise the L1 loss on the presynaptic activity, we can calculate for each category what value best predicts the intensity at each pixel location. Let $p$ be our estimation of the intensity at a specific pixel location $x$ and let $N$ be the number of images in a particular category. The L1 loss of our estimation is then given by:

$$f(x) = \frac{1}{N} \sum_{n=1}^{N} |x_n - p| \tag{12}$$

The value for $p$ that minimises this equation is given when the derivative is equal to zero:

$$\frac{df}{dx} = \sum_{n=1}^{N} sgn(x_n - p) = 0 \tag{13}$$

where $sgn$ is the sign function. This equation is satisfied when there are equal numbers $x_n > p$ as there are $x_n < p$, which in turn is satisfied when $p$ is the median of $x_n$. This means that the median intensity at each pixel location in each category best predicts the presynaptic activity from the input, given an L1 loss (visualised in Figure 2). In practice, our networks will not directly minimise the L1 loss on all images in each category, as we will be using mini-batch gradient descent. Furthermore, these predictions assume perfect knowledge of the category, which is not available to the networks. Nevertheless, we can use these predictions as a theoretical optimum to benchmark model performance. In the case that there is no category information at all, the optimal prediction is the median pixel intensities across all images in the training set, which are shown in Figure 3a.

One more thing we like to highlight here is that the pixel intensities in the MNIST images are by no means uniformly distributed. The pixels in the centre are by far the most active, as shown by Figure 3a, while at the edges of the visual field there are pixels that always have an intensity of zero. The latter will become relevant later on (See subsection 4.3). Figure 3b shows the sum of the pixel intensities of all the images in the training set. As can be seen there are numerous white pixels at the edges meaning they are never active in any of the training images.

# 4

# Results

## 4.1 Prediction

To test the predictive performances of our models on the MNIST sequences, we trained two versions of the simplified PredNet model, one with LSTM units and one with SRN units in the representation layer, both with 784 error units and 64 representation units. We also trained a Bathtub model with 784+64 hidden units, so that all models have the same total amount of units for fair comparison. The models were trained for 200 epochs of the training set images with a batch-size of 32. Model weights were optimised using the Adam algorithm with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and a learning rate of 1e-4[45]. No extensive hyper-parameter search was performed for the individual models as the objective was not to seek the best possible performance. Figure 4 shows the training progress for the models.

If our models are successful in predicting, we would expect them to anticipate the input stimuli and inhibit this activity with recurrent connections and thereby lowering the network activity. As we have explored in subsection 3.4 the input stimuli are very stochastic, but nonetheless have distinct statistical structure. Analytically, we have seen that the value that best predicts the pixel

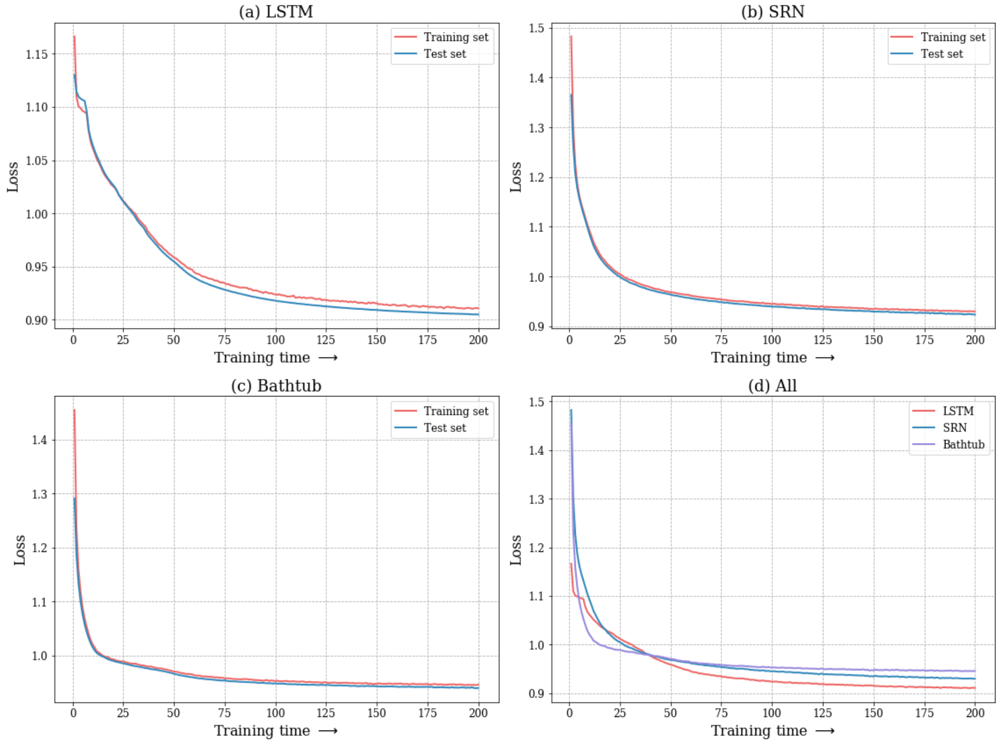


Figure 4: Training progress for our three predictive coding models. (a) Simplified PredNet model with 64 LSTM units. (b) Simplified PredNet model with 64 regular SRN units. (c) Bathtub model with 784 + 64 units. (d) Training progress of all three models compared.

intensity of an input image with an L1 loss is the median of that pixel in all images within that digit category. Inhibiting this value at every input unit will thus be the optimal strategy for minimising the input activity. This assumes knowledge about the category the image is taken from, which is of course not available to our models and must be inferred from past input. There are a number of alternative strategies imaginable that these models might learn to employ instead, considering that the learning objective is to minimise neural activity. It would be hard to say a priori what will happen. Firstly, it could be that all the weights will collapse to zero and there will be no prediction. The remaining network activity will be induced just by the input stimuli, but there will at least be no additional activity. This would also be the best a network can do if the input were completely random. Secondly, it could be that the models do not learn to decode the correct category of the input digit and will learn to inhibit merely the median of all images in the training set (see Figure 3a).

Figure 5 shows the average loss (i.e. presynaptic activity) in the course of time. We plot this against three theoretical benchmark models:

- **Input** shows the mean absolute value of just the input activity. This corresponds to the scenario where all the weights of a model have collapsed to zero and the only activity is provoked by the input.
- **Median** shows the mean absolute value of the input minus the training set median. This corresponds to the scenario where a model has not learned to decode the correct category and always predicts the median of the training set.
- **Category median** shows the mean absolute value of the input minus the median of the input category in the training set. This corresponds to the theoretically optimal prediction with perfect knowledge about the category.

If a model is predicting then we expect it to be better than the first two scenarios. The best it can do is inhibit the median for the correct category every timestep. We see that the activity of the models is high at the first timestep. This is of course not surprising as the activity of the hidden state in the models is initialised to 0. As the sequence progresses and more context information is available the activity of our three models drops and they do significantly better ($p < 0.001$) than the 'median' model after the second image is presented. The network activity drops each step in the sequence until it stabilises from around the fifth or sixth step onwards. The difference between our three models up to that point differ quite a bit, where the LSTM model activity drops significantly faster than the SRN model activity, which in turn drops significantly faster than the Bathtub model activity. After that point the activity of the three is close to each other ($p > 0.05$), and come close to the theoretical optimum (category median). Note that the input activity is not the only activity for these networks since there are recurrent connections as well and whom are necessary to generate the predictions, so there is a discrepancy to be expected. One might have noticed that the initial activity of both the LSTM and SRN models is higher than that of the theoretical models or the Bathtub model. This is because their architecture is different and the representation layer gets activated by the prediction error signal in the same timestep, while there is no separate prediction layer in the Bathtub model. To conclude, our models perform better than the two theoretical scenarios where either nothing about the input statistics is learned, or a global statistic independent of the input category. We see that the network activity drops as more information is processed indicating that past input is indeed used to infer some statistics of future input and the activity comes near to what is the minimal activity with perfect knowledge about the input category. These findings are consistent with the premise that the models use past input to anticipate future input and we can conclude that they are indeed performing prediction.

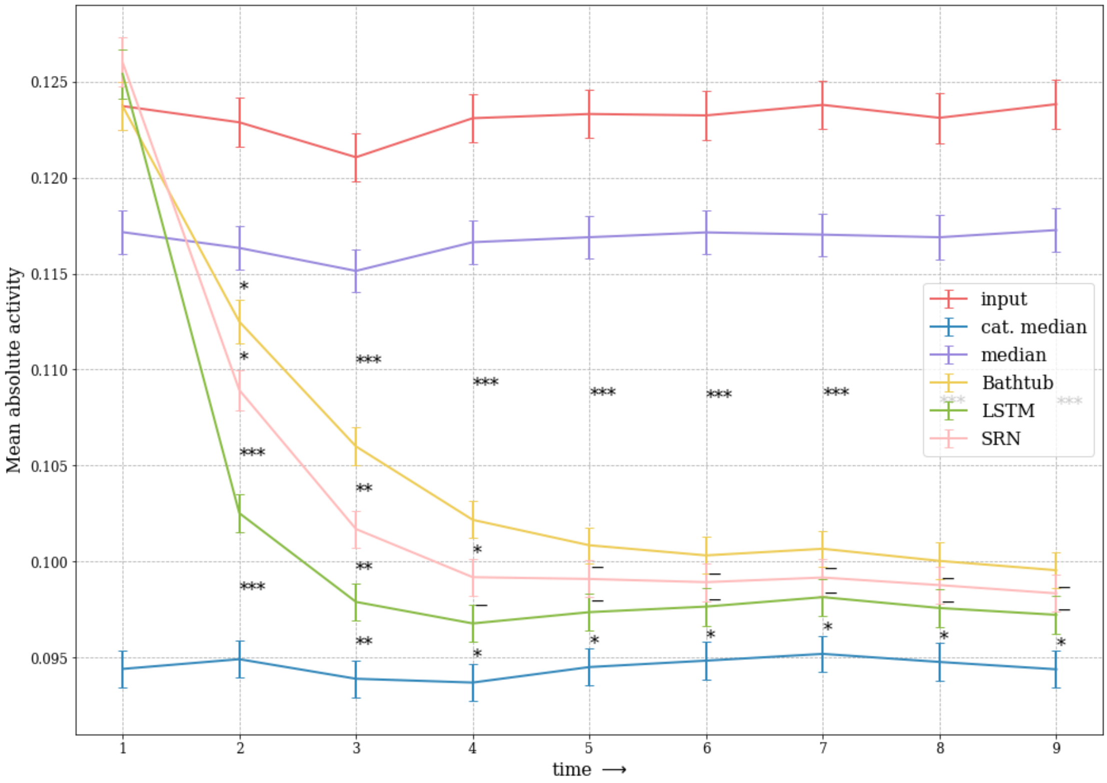


Figure 5: Mean absolute presynaptic activity of our three models compared to three theoretical models as the sequence progresses. (input): the input is the only source of activity, no recurrent feedback or other activity. (cat. median): the recurrent feedback always inhibits the category median of the training set perfectly. (median): the recurrent feedback always inhibits the median of all images in the training set. Lines show the mean over 990 trials (one epoch of the test set). Error bars denote the standard error of the mean. Asterisks denote the p-value of Welch's t-test between the means above and below ($* \rightarrow p < 0.05$; $** \rightarrow p < 0.01$; $*** \rightarrow p < 0.001$; $- \rightarrow p \geq 0.05$).

## 4.2 Error units

Next we want to test whether error units emerge. In our simplified PredNet model the error units are not explicitly modelled, but simply integrate the input with the feedback from the recurrent representation units (see Equation 5). This feedback is neither forced to be exclusively excitatory nor inhibitory and when untrained is equally distributed between the two (see the left most histograms in Figure 6). Since input activity is always excitatory and the models are optimised to minimise activity, the expectation is that the network will learn to counterbalance this activity with strictly inhibitory feedback signals. This way these units will implicitly calculate the error between the predicted signal (the inhibitory feedback) and the actual input signal. We use the same LSTM and SRN models trained for the previous experiment and compare them to two respective versions with untrained weights and present them a set of input stimuli after which the feedback signal intensities are recorded. The result is shown Figure 6 and it is clear that the trained models solely exhibit inhibitory feedback signals. We can therefore conclude that explicit subtraction is not a necessary component of predictive coding and that inhibition will

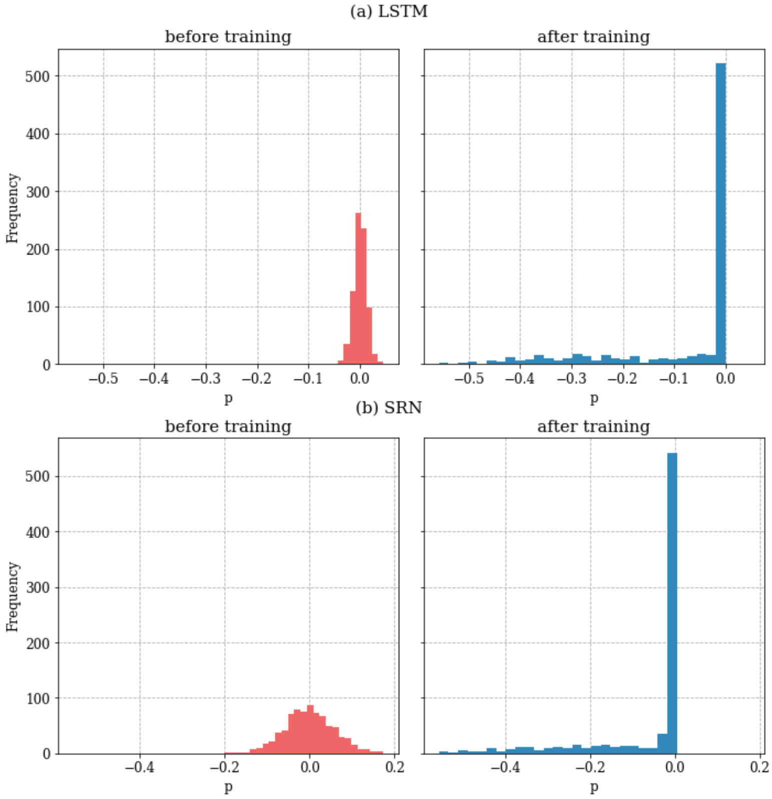


Figure 6: Histogram of feedback signal intensity before and after training for the simplified PredNet models with LSTM or SRN units as representation units.

emerge as a result of minimising neural activity in a recurrent network.

### 4.3 Prediction units

Confident that our models are able to learn to predict and successfully inhibit incoming sensory input, let us now investigate whether prediction units emerge. For this we trained a Bathtub model on MNIST sequences of nine digits per sequence, and with no additional hidden units. Specifically, hidden state vector is the same size as the input vector $|\mathbf{h}| = |\mathbf{x}| = 784$ and the weight matrix, $\mathbf{W}$, is a 784-by-784 matrix. In Figure 3b we showed that there are numerous units around the edges of the visual field that never receive any input, because the pixel intensities are always zero. For the network there is no way of distinguishing between these pixels or the allotted hidden units, so if the network would learn to utilise any of these units as prediction units it would be impartial to either group. For visualisation of the network state however it is favourable to keep the hidden state vector the same size as the input vector. Assuming that the task is sufficiently simple that there will be no shortage in so to say ’free’ (not input-bound) units, we opt to have them the same size.

In Figure 7 the state of the Bathtub model is visualised during the run of an example sequence. showing the input, the feedback from the hidden state and the network activation. Note that the hidden state is initialised to zero so on the first timestep there is no feedback. There are several interesting observations to make. First of all we see that the feedback signal ($\mathbf{p}_t$, second

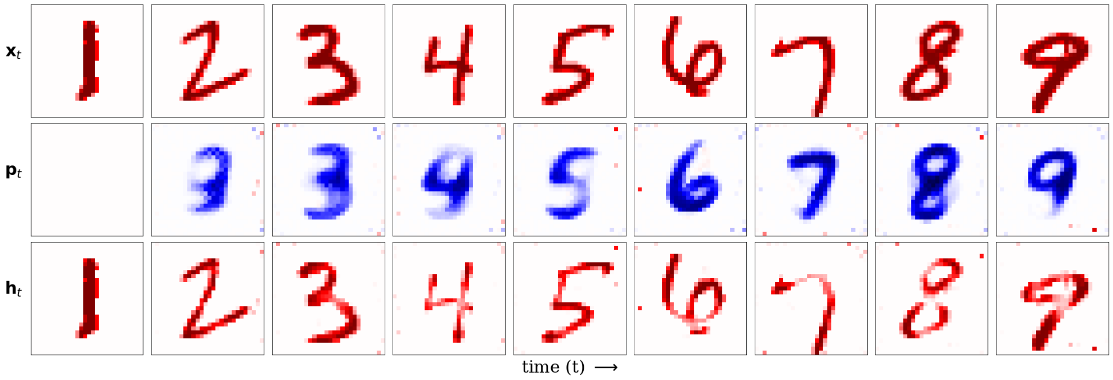


Figure 7: Network state in the course of an example sequence. The top row shows the input activity, $\mathbf{x}_t$, the middle row shows the feedback activity, $\mathbf{p}_t$, and the last row shows the unit activation $\mathbf{h}_t$. Excitatory signals are depicted in red while inhibitory signals are blue and the darker the colour, the stronger the signal. The images were not used for training.

row) is predominantly inhibitory (the blue pixels), consistent with the previous experiments. Secondly we see that the inhibitory feedback patterns resemble the medians of the MNIST categories (see Figure 2) and that the resemblance gets more pronounced as the sequence progresses (see also Figure 14), possibly because with more context information the network can make more accurate predictions. Lastly, we see that aside from the inhibitory activity in the centre there is a sparse pattern of activation near the edges of the images in the region where there is little to no input activity. We will argue that this is no coincidence and that these units are used as prediction/representation units.

In Figure 8 we see the difference in the median[4] unit activity between the same input category shown at timesteps t=2[5] and t=9. The median activity in the centre of the visual field clearly decreases considerably as more sensory information is processed. This is consistent with error calculation units as more contextual information gives more certainty, better predictions, lower prediction error, and thus lower activity. Conversely, we see that for every category there are one or several units near the edges of the visual field whose activity increases with more sensory information. This is similarly consistent with prediction/representation units, since more contextual information increases the certainty of the prediction (or internal representation) and therefore an increase in activity of the corresponding prediction unit(s) is expected.

If the model had developed separate populations of error units and prediction units, they are expected to have different activity profiles. Error units calculate the difference between the predicted input and the actual input, by integrating excitatory signals from the input and inhibitory signals from prediction units. In the Bathtub model this means that the error units receive excitatory signals from $\mathbf{x}_t$ and inhibitory signals from $\mathbf{p}_t$. Prediction units, on the other hand, hold the internal representation of the environment and update this based on the prediction errors. The prediction units are therefore expected to receive signals (excitatory or inhibitory) primarily from the error units. In the Bathtub model this means that the incoming signals for prediction units come from the recurrent connections, $\mathbf{p}_t$, more than from the input, $\mathbf{x}_t$. To test whether this is the case in our model we record the mean excitatory signals from both $\mathbf{x}_t$ and $\mathbf{p}_t$ for each unit at $t = 9$. If there are indeed two populations we would expect a clear distinction between units

[4] The median is used because this is what is approximately optimised for so it better visualises the loss.

[5] At t=1 there is no recurrent feedback yet. t=2 is the first timestep where this has an effect.

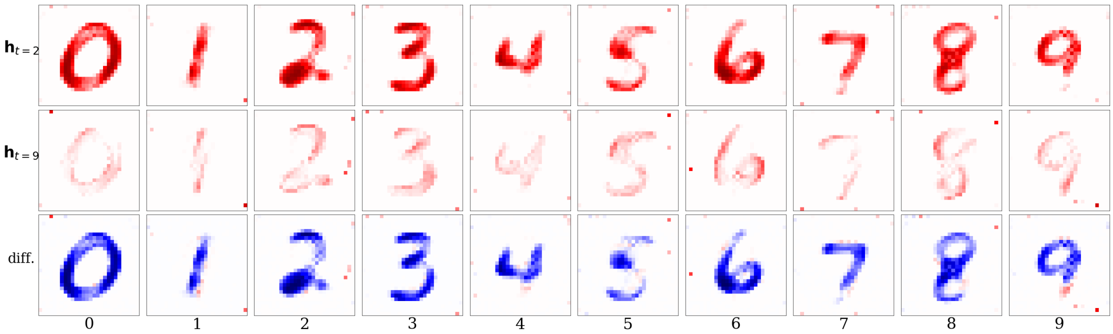


Figure 8: Median unit activity for different input categories at timesteps $\mathbf{h}_{t=2}$ and $\mathbf{h}_{t=9}$ (first two rows). The bottom row shows the difference in activity between the first two rows, $\mathbf{h}_{t=9} - \mathbf{h}_{t=2}$, where blue depicts a decrease of activity, red depicts an increase of activity, and deeper colours depict a greater difference. For each input category approximately 1000 unseen sequences are used (one epoch of the test set).

in their main excitatory source of activation, where error units are input-driven and prediction units are feedback-driven.

Figure 10a and 10b show that the units that receive excitatory signals from the input and those that receive excitatory signals from the recurrent connections visibly form two distinct populations. The input-driven units are concentrated at the centre of the visual field while the feedback-driven units are mostly near the edges of the visual field. In Figure 10c we can see again that input-drive and feedback-drive is well separated between units. It should be noted that there is also excitatory feedback received by units in the centre although the signal is considerably less strong than that from the input. We will discuss this further below.

To confirm that the feedback-driven units are responsible for prediction we divide the network units in two sets: the primarily input-driven units, where $\langle \mathbf{x}^{+} \rangle > \langle \mathbf{p}^{+} \rangle$, which we will call the error units, and the primarily feedback-drive units, where $\langle \mathbf{p}^{+} \rangle > \langle \mathbf{x}^{+} \rangle$, which we will call the prediction units. We then "lesion" the prediction units by masking their activity. We then run the test sequences and record the L1 loss on presynaptic activity. Figure 11 shows the result compared to the original Bathtub model and the theoretical models (see subsection 4.1). We can see that the activity after the first timestep is comparable, but after that the activity of the lesioned Bathtub model increases slightly and remains stable throughout the rest of the sequence, whereas the activity of the un-lesioned model keeps lowering until it stabilises at around $t = 6$. The activity of the lesioned model is still slightly lower than that of the theoretical model that predicts the global median (see Figure 3a), but the difference is not significant ($p > 0.05$).

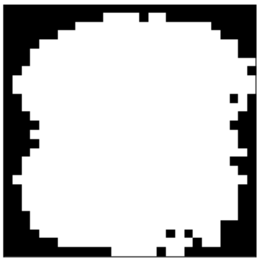


Figure 9: Feedback-driven units (black) vs. input-driven units (white).

The question that remains is why is the activity in the lesioned model still lower than either the *input* or *median* models.[6] Indeed there still seems to be some prediction generated by the error units. One reason for this is that the prediction units have a delayed response. Remember that they are activated by recurrent connections and these is initialised to zero. This means that

[6] See also Figure 15 for a visualisation of the feedback in the lesioned model

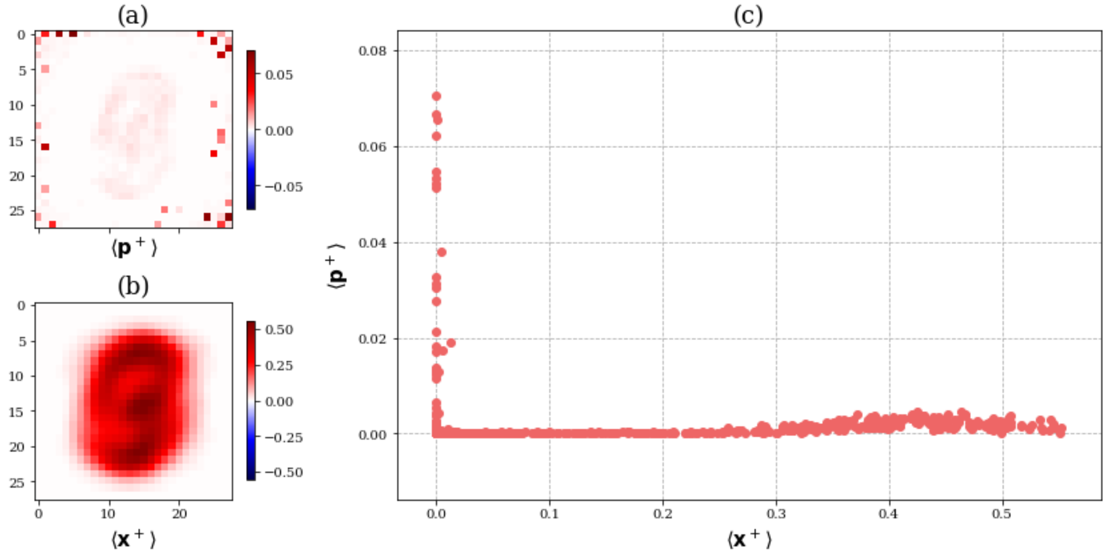


Figure 10: Comparison of mean excitatory signals from input ($\langle \mathbf{x}^+ \rangle$) and recurrent feedback ($\langle \mathbf{p}^+ \rangle$). (a) mean excitatory signals from recurrent connections for each unit in the visual field. (b) mean excitatory signals from input connections for each unit in the visual field. (c) Excitatory signals from input plotted against that from recurrent connections shows a clear distinction between input-driven and feedback-driven units.

the prediction units can be activated at the second timestep and consequently can only exert any effect on the third timestep. This stands in contrast with the PredNet based models, where the separate prediction units are activated right on the first timestep and can already affect the input units on the second timestep. In order for the Bathtub model to lower unit activity on the second timestep, it will have to use the centre error units. The error units thus also perform some prediction as well. The prediction units will have to take into account the activity that is generated by the error units, which could explain the slight excitatory recurrent activity at the error units (see Figure 10) by the prediction units compensating for over-inhibition caused by the less accurate predictions of the error units.[7] Despite that the network was free to use all of its units in any way it seemed fit, there still emerged a clear separation between populations that perform prediction and those that calculate the error.

[7] See also Figure 12 and 13 for visualisations of the recurrent feedback per unit.

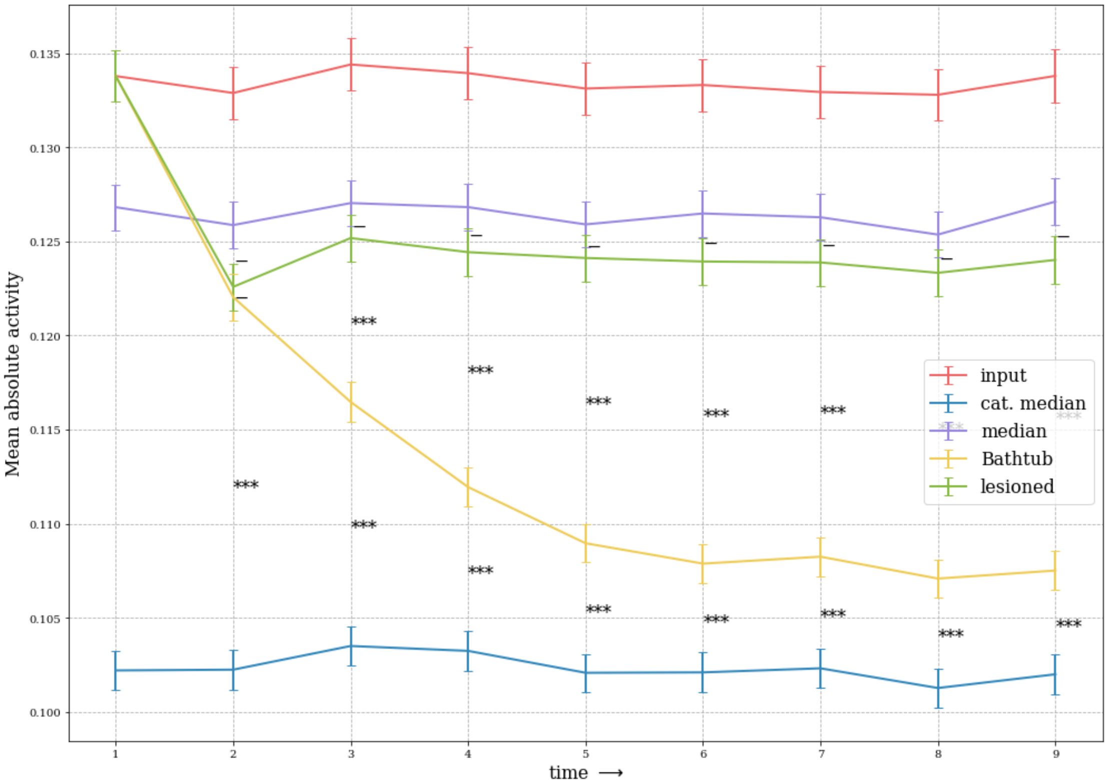


Figure 11: Mean absolute presynaptic activity of the regular and lesioned Bathtub models as the sequence progresses. Three theoretical models are shown for reference. (input): the input is the only source of activity, no recurrent feedback or other activity. (cat. median): the recurrent feedback always inhibits the category median of the training set perfectly. (median): the recurrent feedback always inhibits the median of all images in the training set. Lines show the mean over 990 trials (one epoch of the test set). Error bars denote the standard error of the mean. Asterisks denote the p-value of Welch's t-test between the means above and below ($* \rightarrow p < 0.05$; $** \rightarrow p < 0.01$; $*** \rightarrow p < 0.001$; $- \rightarrow p \geq 0.05$)

# 5 Discussion

We found that recurrent networks that optimise for energy efficiency and with minimal architectural constraints can learn to predict future sensory stimuli and that the population of neurons will naturally divide into distinct groups that perform either error calculations or prediction, two important features of predictive coding. This substantiates the premise that predictive coding, if employed by the brain, is an emergent phenomenon and does not require hard-wired circuitry.

Prediction error calculation and prediction generation are arguably the most essential parts of predictive coding. What we did not address in this work but is also often included in predictive coding theories is hierarchical processing[8,11]. The cortex can be organised into a hierarchy, where areas that are higher up in the hierarchy extract more abstract features than the areas that are closer to the sensory input and thus lower in the hierarchy. In the view of predictive coding, higher areas predict the neural activity of lower areas and project their predictions down, whereas the lower areas send up the error of these predictions. We show that hierarchical processing is not a necessity for predictive coding and that it can work with lateral connections. This is in line with work by Zhu and Rozell[46] who show that a shallow and recurrent sparse coding network can account for similar extra-classical receptive field effects in the V1 visual cortex that the hierarchical predictive coding model by Rao and Ballard[11] was used to explain. However, this is not to say that predictive coding could not also be done in a hierarchical fashion. Future work could adapt our recurrent efficient coding model for hierarchical processing and test whether bottom-up prediction error and top-down predictions emerge in such a setting. Another possibility is to choose a stimuli set with sufficiently complex statistics for which hierarchical processing is beneficial and test whether a hierarchical predictive coding network emerges. It might be necessary to adapt our models for convolutional processing, so that there is a natural increase in receptive field over time and/or through the hierarchy.

Another point to be made is that although the MNIST sequences have desirable properties for this work, such as simple statistics and interpretability, it is not a very realistic dataset for visual processing. To gain a better understanding of the biological plausibility of our model we suggest to test it on natural videos as well. Furthermore, we have seen that the input stimuli are highly stochastic. Our models learn to approximately inhibit the median of each digit category, and while this is the best it can do it will as good as never match the actual input image (just see Figure 7). This means that the error unit activity will never be inhibited completely and there will always be some prediction error signal. In the Bathtub model, this means that the error units could take up part of the role of prediction and that the prediction units will have to account for the activity generated by error units. If the error units would often be completely inhibited, perhaps there would be less prediction done by them. Future research could investigate what the influence is of the predictability of the input signal on the distribution of prediction and error units and their activity profiles.

From a broader perspective what this work shows is that simple ingredients can give rise to complex behaviour, if chosen carefully. It can help direct our search for the fundamental components in cortical circuitry, and how processes in the implementational level (energy efficiency) are linked to computations in the algorithmic level (predictive coding).

# References

1. Mandyam Veerambudi Srinivasan, Simon Barry Laughlin, and Andreas Dubs. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 216(1205):427–459, 1982.

2. Rajesh PN Rao and Terrence J Sejnowski. Predictive sequence learning in recurrent neocortical circuits. In *Advances in neural information processing systems*, pages 164–170, 2000.

3. Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

4. Andre M Bastos, W Martin Usrey, Rick A Adams, George R Mangun, Pascal Fries, and Karl J Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

5. Rakesh Chalasani and Jose C Principe. Deep predictive coding networks. *arXiv preprint arXiv:1301.3541*, 2013.

6. Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

7. Randall C O'Reilly, Dean Wyatte, and John Rohrlich. Learning through time in the thalamocortical loops. *arXiv preprint arXiv:1407.3432*, 2014.

8. William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

9. Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, 2017.

10. William Lotter, Gabriel Kreiman, and David Cox. A neural network trained for prediction mimics diverse features of biological neurons and perception. *Nature Machine Intelligence*, 2 (4):210–219, 2020.

11. Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79, 1999.

12. Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

13. Georg B Keller and Thomas D Mrsic-Flogel. Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435, 2018.

14. Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3): 183, 1954.

15. Bruno A Olshausen and David J Field. Natural image statistics and efficient coding. *Network: computation in neural systems*, 7(2):333–339, 1996.

16. Seymour S Kety. The general metabolism of the brain in vivo. In *Metabolism of the nervous system*, pages 221–237. Elsevier, 1957.

17. Louis Sokoloff. The metabolism of the central nervous system in vivo. *Handbook of physiology, section I, neurophysiology*, 3:1843–1864, 1960.

18. Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1:217–234, 1961.

19. Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

20. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

21. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

22. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529 (7587):484–489, 2016.

23. 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):1–10, 2016.

24. James C.R. Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235 – 250, 2019. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2018.12.005. URL http://www.sciencedirect.com/science/article/pii/S1364661319300129.

25. Guillaume Bellec, Franz Scherr, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*, 2019.

26. Rohan Chopra and Sanjiban Sekhar Roy. End-to-end reinforcement learning for self-driving car. In *Advanced Computing and Intelligent Engineering*, pages 53–61. Springer, 2020.

27. Emmanuel Dupoux. Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner. *Cognition*, 173:43–59, 2018.

28. David Mumford. On the computational architecture of the neocortex. *Biological cybernetics*, 66(3):241–251, 1992.

29. David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *Josa a*, 4(12):2379–2394, 1987.

30. Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3): 287–314, 1994.

31. Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

32. Aapo Hyvärinen, Jarmo Hurri, and Patrick O Hoyer. *Natural image statistics: A probabilistic approach to early computational vision.*, volume 39. Springer Science & Business Media, 2009.

33. Anthony J Bell and Terrence J Sejnowski. The âĂIJindependent componentsâĂİ of natural scenes are edge filters. *Vision research*, 37(23):3327–3338, 1997.

34. Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

35. Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

36. Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.

37. Matthew Chalk, Olivier Marre, and Gašper Tkačik. Toward a unified theory of efficient, predictive, and sparse coding. *Proceedings of the National Academy of Sciences*, 115(1):186–191, 2018.

38. Sophie Denève, Alireza Alemi, and Ralph Bourdoukan. The brain as an efficient and robust adaptive learner. *Neuron*, 94(5):969–977, 2017.

39. Sophie Denève and Christian K Machens. Efficient codes and balanced networks. *Nature neuroscience*, 19(3):375, 2016.

40. Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

42. Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

43. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

44. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

46. Mengchen Zhu and Christopher J Rozell. Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system. *PLoS computational biology*, 9(8):e1003191, 2013.

# Appendix

## A

## Additional Figures

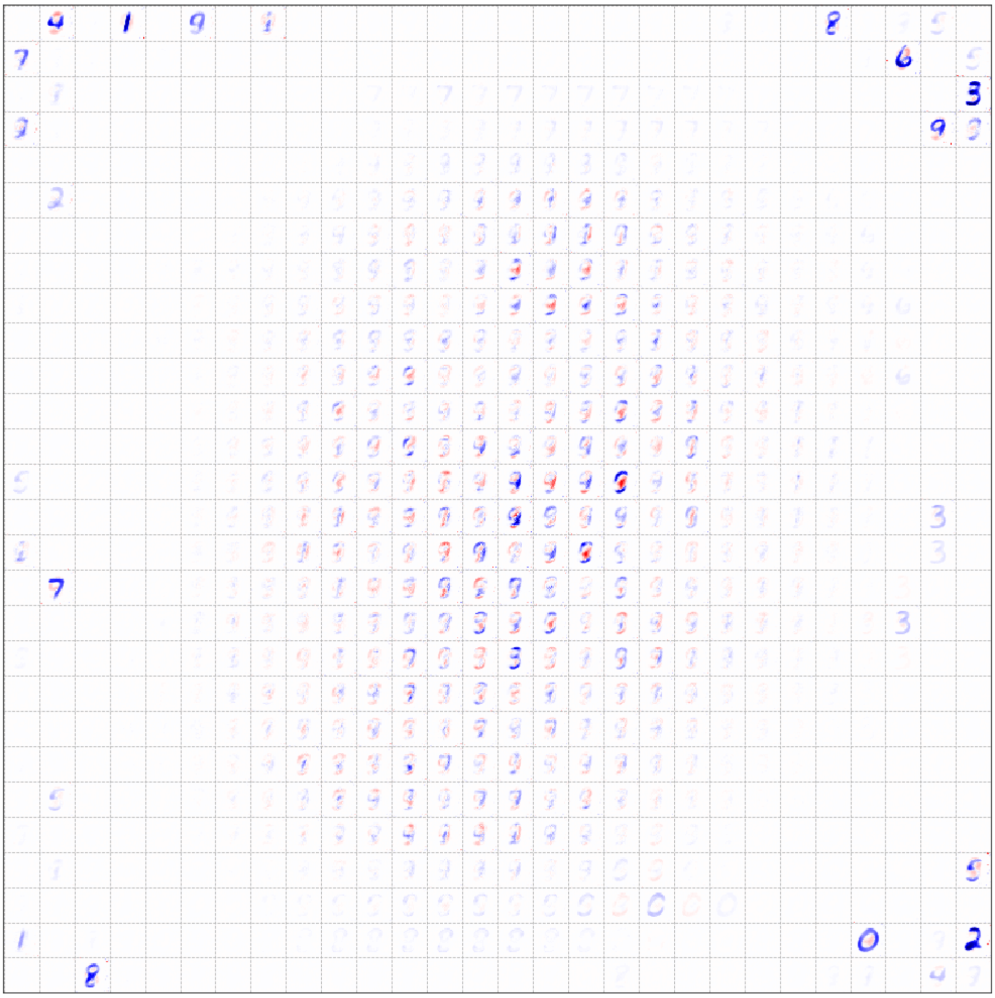


Figure 12: Mean recurrent feedback per unit at $t = 2$ visualised as 28x28 squares representing each unit at their position in the visual field. Each square contains 28x28 pixels showing the mean recurrent feedback of that unit on the next timestep. The sparse set of active units around the edges (i.e. prediction units) show recognisable patterns of inhibition resembling different digits, while this effect is less pronounced in the centre units.

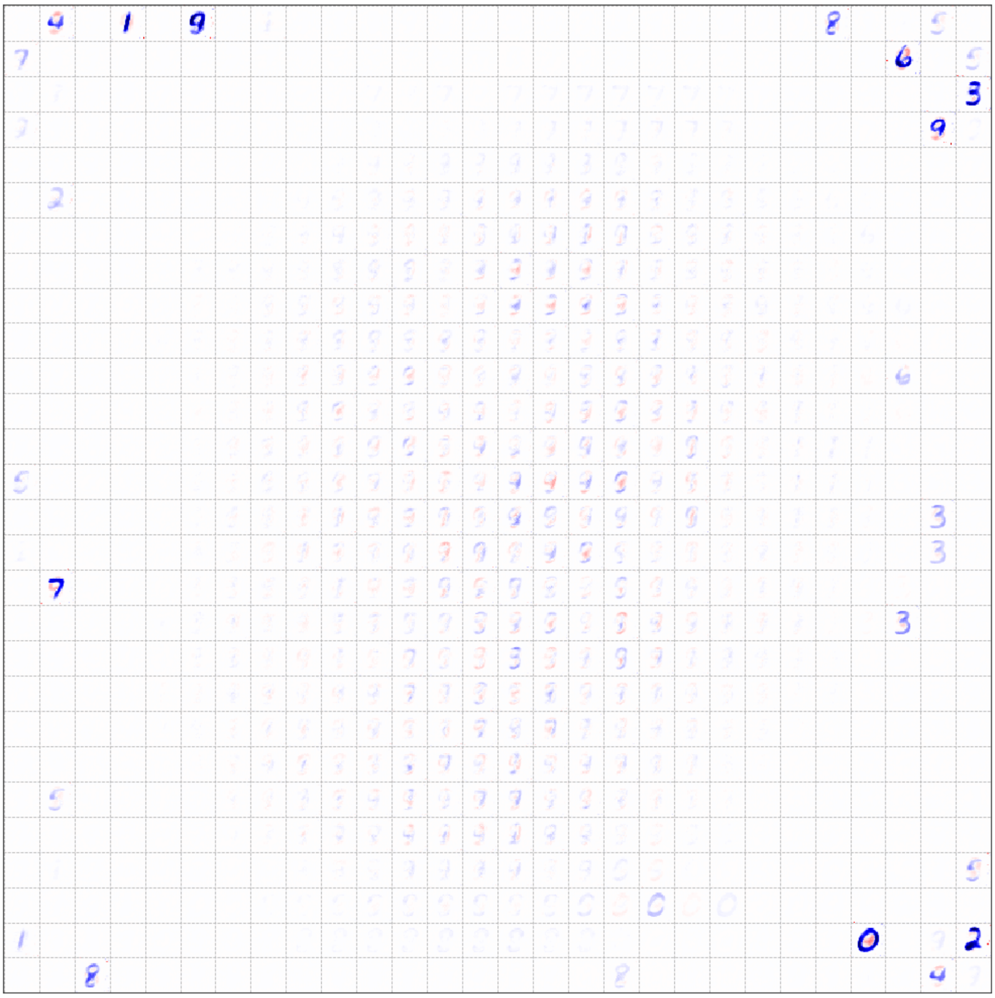


Figure 13: Mean recurrent feedback per unit at $t = 9$ visualised as 28x28 squares representing each unit at their position in the visual field. Each square contains 28x28 pixels showing the mean recurrent feedback of that unit on the next timestep. The difference between centre and edge (i.e. error and prediction) is even more pronounced after longer sequences.

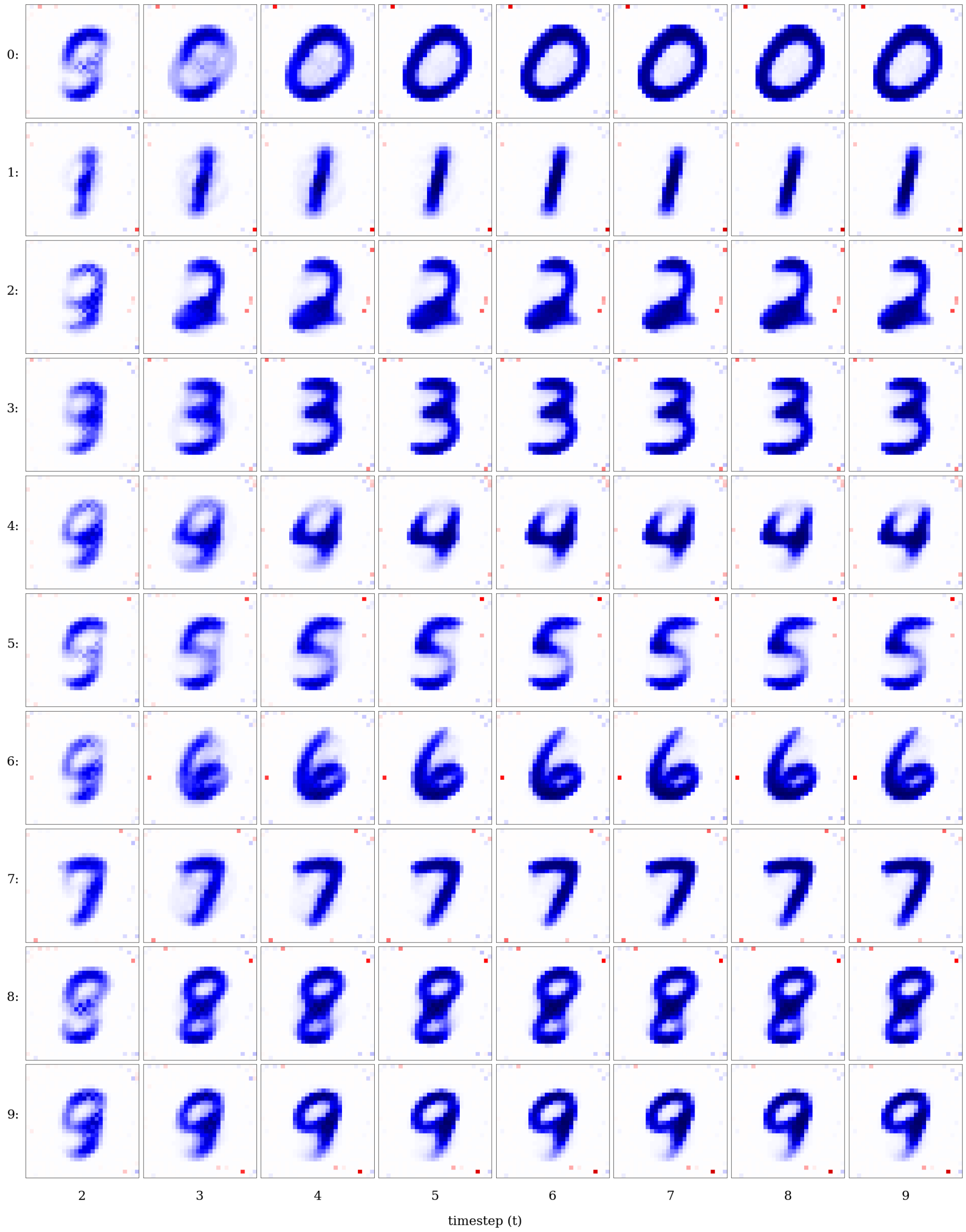


Figure 14: Recurrent feedback ($\mathbf{p}_t$) in the bathtub model for different input categories at different points in time. Each row shows the recurrent feedback at the step with a specific input category will be expected. Each column shows the recurrent feedback after a different amount of preceding images in the sequence. The inhibitory effect (i.e. predictions) get more pronounced as sequences progress.

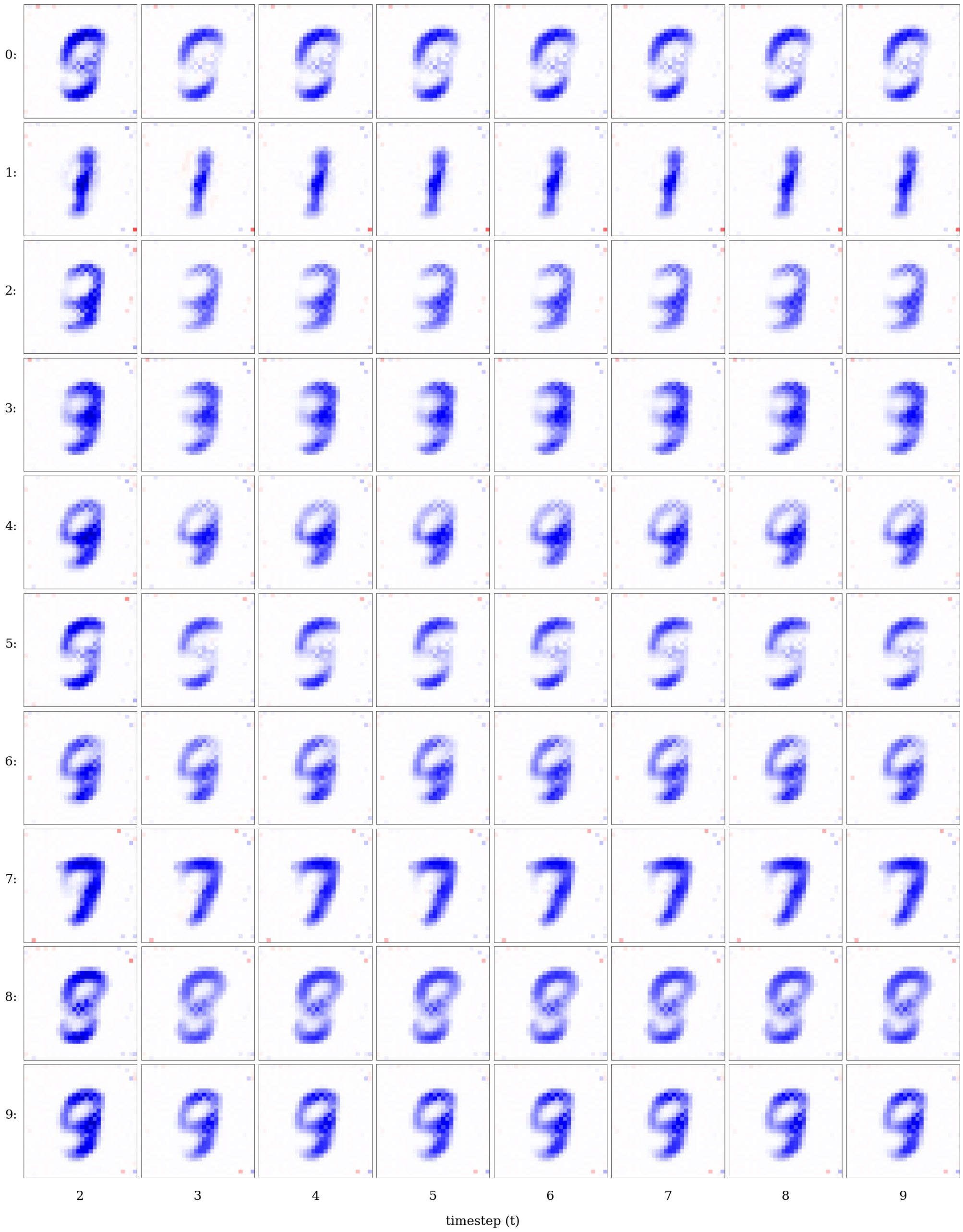


Figure 15: Recurrent feedback ($\mathbf{p}_t$) in the lesioned Bathtub model for different input categories at different points in time. Each row shows the recurrent feedback at the step with a specific input category will be expected. Each column shows the recurrent feedback after a different amount of preceding images in the sequence. The inhibitory effect with lesioned prediction units is weak and remains constant throughout the sequence.

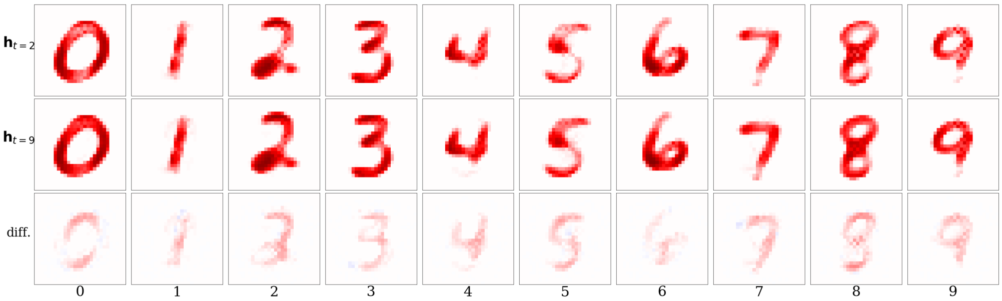


Figure 16: Median unit activity in the lesioned Bathtub model for different input categories at timesteps $\mathbf{h}_{t=2}$ and $\mathbf{h}_{t=9}$ (first two rows). The bottom row shows the difference in activity between the first two rows, $\mathbf{h}_{t=9} - \mathbf{h}_{t=2}$, where blue depicts a decrease of activity, red depicts an increase of activity, and deeper colours depict a greater difference. For each input category approximately 1000 unseen sequences are used (one epoch of the test set). Lesioning the prediction units shows considerably less inhibition than in the regular Bathtub and the effect even diminishes over time.