Face Perception using a Predictive-Coding Neural Network

by

William Pugsley

A thesis
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contribution

The derivations in Section 2.2.1 are a generalization of the derivations in Bogacz's tutorial on predictive coding [3]. The programming implementation of predictive coding neural networks was built off code originally made by Junteng Zheng at the NeuroCognitive Computing Lab at the University of Waterloo. I modified the architecture to match the direction of information flow (discussed at the end of Section 2.2.3) and included algorithms to measure response times and track the activities of each layer. I performed all data preprocessing, hyperparameter tuning, and evaluation of results. There was also a Nengo-based implementation of predictive coding that one of my supervisors, Jeff Orchard, and I built. This approach was abandoned halfway through the project in favour of Junteng Zheng's model.

Abstract

Predictive coding is a theory of neural computation inspired by biological evidence and with mathematical backing. As with any theory, it is only useful if it is explanatory and predictive. This project uses predictive coding neural networks to model the facial perception pathway in human brains. We accurately reproduced the results of an electroencephalography experiment, which was the motivating study for this project [38]. In that study, the authors found a delay in P1, N170, and P2 Event Related Potentials (ERP) components in the EEG recordings of participants when shown faces with fewer parafoveal features. Our predictive coding model exhibits the same time delay. We also propose a mechanistic explanation for this behaviour based on the activities of neurons in V1, the inferior occipital gyrus, and the posterior fusiform gyrus.

Acknowledgements

I would like to thank Jeff Orchard and Roxane Itier for their invaluable support and expertise.

Table of Contents

Author's Declaration ii

Statement of Contribution iii

Abstract iv

Acknowledgements v

List of Figures viii

List of Tables ix

1 Introduction 1

2 Background 2

2.1 Facial Perception and the Brain . . . 2

2.2 Predictive Coding . . . 3

2.2.1 Maximum Likelihood Estimator . . . 4

2.2.2 Gradient Ascent . . . 7

2.2.3 Neural Network Implementation . . . 7

3 Neural Learning and Inference with PC Networks 9

3.1 Neural Learning . . . 9

3.2 Inference . . . 10

4 Training the Network for Facial Perception 11

4.1 Data . . . 11

4.2 Missing Features . . . 14

4.3 Training Algorithm and Hyperparameters . . . 15

4.4 Inference Algorithm and Hyperparameters . . . 15

5 Results **15**

5.1 Accuracy 15

5.2 Activity Phase Space 16

5.3 Feature Combination Response Times 19

5.4 Parafoveal Feature Response Times 22

5.5 Discussion 25

6 Conclusion **26**

References **29**

List of Figures

1 Face perception pathway in the human brain. 2
2 Predictive coding neural network architecture diagram. 8
3 Samples of training images after processing and augmentation. 13
4 L^2 norm of the activities and derivatives for each layer in a PC network. . 17
5 Activities of each of the two neurons in the output layer. 18
6 Histogram of average response times to different feature combinations. . . . 20
7 Boxplot of average response times to different feature combinations. 21
8 Histogram of average response times to different parafoveal feature numbers. 22
9 Boxplot of average response times to different parafoveal feature numbers. 24

List of Tables

1 ImageNet class codes and labels used for training. 12

2 Training dataset composition. 14

3 Grouping of facial feature combinations by parafoveal feature number. . . . 14

4 Mean response times and standard deviation across networks for each feature combination treatment. 19

5 Shapiro-Wilk normality test applied to feature combinations data. 21

6 Results of paired samples t-tests for feature combination. 23

7 Mean response times and standard deviation across networks for each parafoveal number treatment. 25

8 Shapiro-Wilk normality test applied to parafoveal feature number data. . . 25

9 Results of paired samples t-tests for parafoveal feature number. 25

1 Introduction

Facial recognition and face detection are an application of artificial intelligence with a rich history. Beginning in the 1960s with a dataset of only 60 images, machine learning tools for facial recognition have grown to use many millions of inputs [13]. The driving force behind this explosion in capabilities is the advent of neural networks. Neural networks can achieve comparable performance to, or even superior performance than, humans at this task [34].

However, machine learning models trained to recognize faces, even neural networks, are not always suitable models for the facial perception process in human brains. Winward et al. describe an experiment that measures the response times of humans when presented with different kinds of stimuli [38]. Participants are shown images of faces with various combinations of facial features: some have only one eye, others have two eyes and a mouth, etc. As the number of features decreases, the response time of the participants increases. This response time is measured using an electroencephalogram (EEG).

Given a model of the brain where parts of the visual cortex perform some operations on the sensory input and pass the results onto the next layer, these results are counter-intuitive. If there is less data, as is the case for a face with fewer features, the brain should be able to process the information faster. The delayed response times indicate that the brain may operate using a predictive coding (PC) model.

A predictive coding network is one in which the flow of information is bidirectional. Predictions flow from higher-level areas in the brain to lower levels, informing the lower levels what they should expect to observe. Meanwhile, error signals flow back up the network, telling each layer how wrong their predictions are [28]. This has long been held as a theory of brain function [21]. A layer encodes predictions of what the previous layer observes, hence the name.

These kinds of networks have been claimed to model human neural networks and exhibit behaviour mimicking them as well [20]. Particularly, they have seen success as models of the visual system [27, 28]. Predictive coding has even been posited as a model for introspection in the brain [32]. There is also neuro-biological evidence that the wiring within, and between, cortical columns is consistent with predictive coding.[1].

Beyond its presence in neuroscience, this predictive coding framework can be expressed mathematically and implemented as an artificial neural network. PC networks can be used for classification, regression, and generative machine learning tasks [15]. Predictive coding as a model for artificial intelligence has been gaining popularity since around 2017 [3, 24, 25].

Theoretical neuroscience is a field where scientists propose mechanistic neural-network models to explain neuroscientific observations, like EEG. This project seeks to present further evidence supporting the theory that predictive coding networks model the human brain. It will do so by replicating the experimental results in the study by Winward et al. using a predictive coding network, suggesting explanations for these results, and making predictions for future experiments that may be done [38].

2 Background

2.1 Facial Perception and the Brain

The current understanding of facial perception in the brain is informed largely by brain imaging studies and individuals with facial perception disorders [29, 31]. From these experiments and observations, scientists have deduced which regions of the brain are associated with facial perception, what their roles may be, and how they interact. Fig. 1 depicts a simplified version of the facial perception pathway in the brain [11]. We are not concerned with the detailed anatomy of the human brain, just the flow of information between layers. Data from the retina flows into early visual regions (V1). This information passes into higher-level regions of the brain, namely the inferior occipital gyrus (IOG) and the posterior fusiform gyrus (pFUS) [7, 11]. Note that the arrows depicting connections in Fig. 1 are indeed bi-directional. This reflects the predictive coding nature of the model we will implement and use.

Figure 1: A simplified model of the facial perception pathway in the human brain. Stimuli data flows from left to right whereas predictions are sent in the other direction.

In the study by Winward et al. [38], participants fix their gaze on a screen and an image of a face is flashed. The image is positioned on the screen such that one of the face's eyes is in the participant's fovea (centred and in focus). The image sizes are such that the remainder of the face is in parafovea, or just out of focus. The experiment varies the number of facial features in parafovea, henceforth referred to as parafoveal features, and records the brain's activity using an EEG. From these recordings, they extract event-related potentials associated with facial perception; the most important ERPs being P1,

N170, and P2 [39]. As the number of parafoveal features decreases, there is an increased delay in the presence of these ERPs as measured from stimulus onset. There is also a decrease in their magnitude for the N170 and P2 nodes [38].

The response delay of human brains when presented with faces having fewer parafoveal features was measured using EEGs. These are the motivating results for this project, so it is worthwhile to discuss their basis. EEG devices measure the change in electric potential across the scalp resulting from post-synaptic activities of groups of neurons [39]. This provides a natural interpretation of the results from our neural network. EEGs indirectly measure the concentration and distribution of neural activities. Thus, when we analyze the activities of the neurons in an artificial neural network, we are performing an EEG-like experiment on our model. This link will be discussed in more detail in Section 3.2.

2.2 Predictive Coding

The majority of artificial neural networks (ANNs) and their learning algorithms are not biologically plausible representations of the brain. Specifically, we seek to define an ANN that satisfies the conditions of local computation and plasticity. Biologically, the former refers to the principle that neural activity is performed "on the basis of the activity of its input neurons and synaptic weights associated with these inputs" [3]. The latter refers to the principle that changes to these weights are "based on the activity of pre-synaptic and post-synaptic neurons" [3]. For an ANN, this means any computation done in a neuron should only use inputs to that neuron and connection weights. This is the case in standard feed-forward or convolutional neural networks. It also means that the update of synaptic connection weights should be done with only the activities of the neurons that are connected. Classic neural networks fail to satisfy local plasticity because of this. In backpropagation, the same synaptic connection weights are used for forward passes and error propagation back through the network [16]. This sort of weight copying requires synapses to be bi-directional, which is not biologically plausible.

Mathematically, each layer of the network is represented by a vector of activities. The ith layers has activities $\vec{v}_i$. Not all network states are equally likely as a trained network will prefer certain states based on its learned connection weights. Furthermore, the activity of each layer will depend only on the activity of the layer directly above it [20]. This makes the network a Bayesian chain, thus the distribution of neural activations for the layers is given by

$$p(\vec{v}_0, \ldots, \vec{v}_N) = p(\vec{v}_N) \prod_{i=0}^{N-1} p(\vec{v}_i | \vec{v}_{i+1}). \tag{1}$$

Each of these layers may have different dimensionalities. We assume that the conditional probabilities are normally distributed so that

$$p(\vec{v}_i|\vec{v}_{i+1}) = \mathcal{N}(\vec{v}_i; g_i(\vec{v}_{i+1}, W_i), \Sigma_i) \tag{2}$$

where p represents probability, $\mathcal{N}$ is a normal distribution, g_i is a nonlinear function parameterized by the parameter matrix W_i, and Σ_i is the normal distribution's covariance matrix. There are $N+1$ layers, with the 0th layer being the input layer and the Nth being the output layer. Our choice of normal distribution is a general assumption since we can arbitrarily pick g_i to reshape our distribution. We only concern ourselves with a special case of these functions, namely $g_i(\vec{v}_{i+1}, W_i) = W_i h_i(\vec{v}_{i+1})$ where h_i is some other nonlinear scalar function applied element-wise to its vector input. In this case, W_i is the matrix of weights connecting the $(i-1)$th layer to the ith layer and h_i is an activation function, just as with standard neural networks. The biological interpretation of this equation is that W_i is the strength of the synapses connecting neurons [3].

2.2.1 Maximum Likelihood Estimator

In predictive coding, we seek the maximum likelihood estimator (MLE) of Eq. (2) [3, 20, 40]. That is, we want to maximize the joint probability over its arguments given data (in our case, images of faces). This is biologically justified since at any given moment, it is likely that the brain stores a single value as an estimate rather than a probability distribution of estimates [3, 9]. Our optimization problem is to find the values of each layer's activity, $\vec{\theta}_1, \ldots, \vec{\theta}_{N-1}$, such that

$$\vec{\theta}_1, \ldots, \vec{\theta}_{N-1} = \underset{\vec{v}_1, \ldots, \vec{v}_{N-1}}{\operatorname{argmax}} p(\vec{v}_N) \prod_{i=0}^{N-1} p(\vec{v}_i|\vec{v}_{i+1}). \tag{3}$$

Note that the optimization space is over layers 1 to $N-1$. During training, the values of $\vec{v}_0$ and $\vec{v}_N$ are fixed to be the values of the image inputs and target labels denoted by $\vec{\theta}_0$ and $\vec{\theta}_N$, respectively. This means $p(\vec{v}_N)$ is constant and, after taking the logarithm of the objective function, the right-hand side of Eq. (3) can be rewritten as

$$\underset{\vec{v}_1, \ldots, \vec{v}_{N-1}}{\operatorname{argmax}} \left[\ln p(\vec{v}_N) + \sum_{i=1}^{N-1} \ln p(\vec{v}_i|\vec{v}_{i+1}) \right] \tag{4}$$

since the natural logarithm is monotonically increasing. Plugging in the values of $\vec{\theta}_i$ gives us the log-likelihood of the network state at the MLE:

$$F = \ln p(\vec{\theta}_N) + \sum_{i=1}^{N-1} \ln p(\vec{\theta}_i | \vec{\theta}_{i+1}) \tag{5}$$

where F is called the negative free energy. Plugging in Eq. (2) and dropping constant terms from the normal distribution gives

$$F = \ln p(\vec{\theta}_N) - \sum_{i=1}^{N-1} (\vec{\theta}_i - W_i h_i(\vec{\theta}_{i+1}))^T \Sigma_i \vec{\epsilon}_i, \tag{6}$$

where $\vec{\epsilon}_i = \vec{\theta}_i - W_i h_i(\vec{\theta}_{i+1})$ is the difference in the activities of the ith layer, $\vec{\theta}_i$, and the prediction of the ith layer's activities based on the following layer's activities, the connection weights, and the activation function ($W_i h_i(\vec{v}_{i+1})$). Just as we consider $\vec{\theta}_i$ a vector representing a layer in the neural network, we can think of $\vec{\epsilon}_i$ as a vector representing the error between layers. They act as neurons, just like $\vec{\theta}_i$.

Finding the MLE of Eq. (1) and maximizing the negative free energy are identical processes. To see this we must make use of the fact that finding the MLE is asymptotically equivalent to minimizing the Kullback–Leibler (KL) divergence over a family of possible distributions parametrized by the MLE [8]. As stated previously, we seek the MLE of Eq. (2). We restrict the family of distributions to be Dirac delta distributions, denoted by δ [3, 12]. Thus, the KL divergence between the conditional probability distribution and the δ distribution centred at the MLE, $\delta_{\vec{\theta}_i}$, is

$$\mathrm{KL}(\delta_{\vec{\theta}_i}(\cdot),\ p(\cdot|\vec{\theta}_{i+1})) = \int_{-\infty}^{\infty} \delta(\vec{v}_i - \vec{\theta}_i) \ln \frac{\delta(\vec{v}_i - \vec{\theta}_i)}{p(\vec{v}_i | \vec{\theta}_{i+1})} d\vec{v}_i. \tag{7}$$

We can write the conditioned probability in $p(\cdot|\vec{\theta}_{i+1})$ using the MLE of the next layer instead of $\vec{v}_{i+1}$ because we want to simultaneously maximize Eq. (2) for all i.

Next, consider the sum of Eq. (7) over the index, i. This yields

$$\begin{aligned} \sum_{i=1}^{N-1} \mathrm{KL}(\delta_{\vec{\theta}_i}(\cdot),\ p(\cdot|\vec{\theta}_{i+1})) &= \sum_{i=1}^{N-1} \int_{-\infty}^{\infty} \delta(\vec{v}_i - \vec{\theta}_i) \ln \frac{\delta(\vec{v}_i - \vec{\theta}_i)}{p(\vec{v}_i | \vec{\theta}_{i+1})} d\vec{v}_i \\ = \sum_{i=1}^{N-1} \int_{-\infty}^{\infty} \delta(\vec{v}_i - \vec{\theta}_i) \ln \frac{\delta(\vec{v}_i - \vec{\theta}_i)}{p(\vec{v}_i, \vec{\theta}_{i+1})} d\vec{v}_i &+ \int_{-\infty}^{\infty} \delta(\vec{v}_i - \vec{\theta}_i) \ln p(\vec{\theta}_{i+1}) d\vec{v}_i, \end{aligned} \tag{8}$$

where the second line follows from the properties of the natural logarithm and the law of conditional probabilities [30]. The second integral in the last line of Eq. (8) is trivial, allowing us to rewrite Eq. (8) as

$$\sum_{i=1}^{N-1} \int_{-\infty}^{\infty} -\delta(\vec{v}_i - \vec{\theta}_i) \ln \frac{p(\vec{v}_i, \vec{\theta}_{i+1})}{\delta(\vec{v}_i - \vec{\theta}_i)} d\vec{v}_i + \ln p(\vec{\theta}_{i+1}).$$

We split the remaining integral using logarithm properties to give

$$\sum_{i=1}^{N-1} \int_{-\infty}^{\infty} -\delta(\vec{v}_i - \vec{\theta}_i) \ln p(\vec{v}_i, \vec{\theta}_{i+1}) d\vec{v}_i + \int_{-\infty}^{\infty} \delta(\vec{v}_i - \vec{\theta}_i) \ln \delta(\vec{v}_i - \vec{\theta}_i) d\vec{v}_i + \ln p(\vec{\theta}_{i+1}). \tag{9}$$

The second integral in Eq. (9) is 0 for all terms in the summation. This can be shown with simple integration by parts. The first integral is also straightforward, yielding

$$\sum_{i=1}^{N-1} - \ln p(\vec{\theta}_i, \vec{\theta}_{i+1}) + \ln p(\vec{\theta}_{i+1}).$$

Using the law of conditional probabilities one last time gives our final equation

$$\sum_{i=1}^{N-1} - \ln p(\vec{\theta}_i | \vec{\theta}_{i+1}) - \ln p(\vec{\theta}_{i+1}) + \ln p(\vec{\theta}_{i+1}) = \sum_{i=1}^{N-1} - \ln p(\vec{\theta}_i | \vec{\theta}_{i+1}). \tag{10}$$

This equals the sum of the KL divergences in Eq. (8). Plugging Eq. (5) for the negative free energy into Eq. (10) yields

$$\sum_{i=1}^{N-1} \mathrm{KL}(\delta_{\vec{\theta}_i}(\cdot),\ p(\cdot | \vec{\theta}_{i+1})) = -F + \ln p(\vec{\theta}_N),$$

which, noting that the KL divergence is always non-negative, can be rearranged to the final inequality

$$\ln p(\vec{\theta}_N) \geq F. \tag{11}$$

By maximizing F we are increasing the probability that the system will observe the maximum of $\ln p(\vec{\theta}_N)$.

2.2.2 Gradient Ascent

To find the MLE, we start with a guess for all $\vec{\theta}_i$ and model parameters W_i and Σ_i. We then perform gradient ascent on F with simple Euler time stepping. To find the dynamical system of equations describing this model we need to find the gradients of F. The addition of $\ln p(\vec{\theta}_N)$ is a constant term and does not alter the derivatives. Using matrix calculus identities, we can find the derivatives of F with respect to all relevant parameters [26]. They can be summarized by

$$\nabla_{\vec{\theta}_i} F = -\vec{\epsilon}_i + h'_{i-1}(\vec{\theta}_i) \odot (W_{i-1}^T \vec{\epsilon}_{i-1}) \tag{12a}$$

$$\nabla_{\vec{\epsilon}_i} F = \vec{\theta}_i - W_i h_i(\vec{\theta}_{i+1}) - \Sigma_i \vec{\epsilon}_i \tag{12b}$$

$$\nabla_{W_i} F = \vec{\epsilon}_i h_i(\vec{\theta}_{i+1})^T \tag{12c}$$

$$\nabla_{\Sigma_i} F = \frac{1}{2}(\vec{\epsilon}_i \vec{\epsilon}_i^{\,T} - \Sigma_i^{-1}), \tag{12d}$$

where $\odot$ is element-wise multiplication, also called the Hadamard product, i ranges from 1 to $N-1$, h_i is a scalar function applied element-wise to a vector, and h'_i is its derivative. Special care must be taken when defining the update rules for layers 0 and N; this will be discussed in Section 3. Since the MLE occurs at a maximum by definition, it occurs when the gradient of F is zero, which is an equilibrium solution of Eq. (12). Our variables are unrestricted so there are no boundary conditions to consider.

There is a problem with Eq. (12d), namely the matrix inverse of Σ_i. This gradient updates the connection weights between $\vec{\epsilon}_i$ and itself. By the requirement of local plasticity, this can only be done using the value of the connection weights themselves and pre/post-synaptic activities. Finding the matrix inverse requires information about all the terms in that matrix i.e. updating the connection weights between any two neurons in $\vec{\epsilon}_i$ requires information about the connections between all other neurons. For this reason, we set Σ_i to the identity matrix and will not include Eq. (12d) in our network dynamics.

2.2.3 Neural Network Implementation

Eqs. (12a) to (12c) can be implemented by an ANN. This is our predictive coding model of the facial perception pathway in the human brain as outlined in Section 2.1. We will create a five-layered neural network: one input layer corresponding to the retina, an output layer for the higher-level regions of the brain, and three hidden layers for V1, the IOG, and the pFUS. Thus, $N = 4$ since the input layer is indexed by 0. For the rest of this report, when we refer to 'the network architecture,' we mean the specific architecture in Fig. 2.

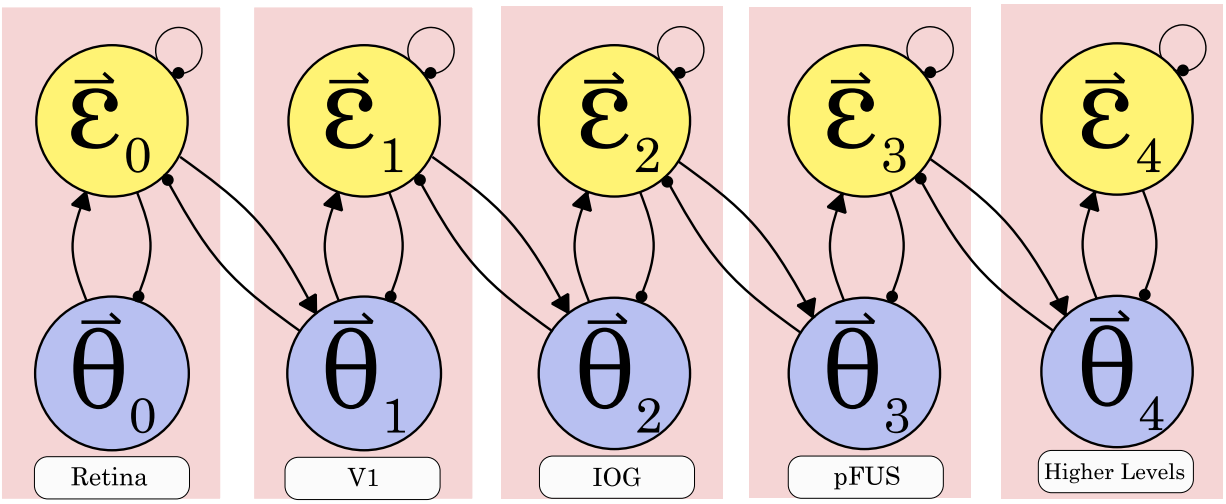

Figure 2: Architecture diagram for the predictive coding network. Paths terminating in circles indicated inhibitory connections and those ending in arrows indicate excitatory ones.

Fig. 2 shows the neural network architecture and connections. The connections between layers implement the second terms in Eqs. (12a) and (12b). The connection from $\vec{\epsilon}_i$ to itself implements the $-\Sigma\vec{\epsilon}_i$ term in Eq. (12b). Each layer is labelled according to a section of the facial perception pathway in the human brain. This reinforces the analogy between the predictive coding model and the model in Fig. 1. The input layer has $6,800$ neurons, the subsequent layers have 300, 200, and 100 neurons respectively, and the output layer has 2 neurons[1]. There are connections going in both directions (up and down the network) but each uses its own unique set of synaptic weights. Thus, predictive coding does not run into the same plausibility issue as backpropagation i.e. bi-directional connections.

The network depicted in Fig. 2 contains closed loops, therefore the order in which we update the system is important. To perform one iteration of Euler time stepping using the ANN, we update the activities of each layer using Eq. (12a), starting with the lowest layer and working our way up. We then perform an update on the errors using Eq. (12b), again starting from the lowest layer. Once the activities and errors of each layer have been updated, we update the connection weight matrices using Eq. (12c). Thus, at iteration $t+1$ of the network, we update the activities using the system values at t, then we update the errors using the activities at $t+1$ but the errors and weights at t, and finally update

[1]In standard diagrams for neural network architectures, each block is a vector containing several circular nodes. These nodes are the scalar elements of that vector. In this diagram, each of the two circles in a block are identically-sized vectors containing the activities of the layers and the errors.

the connection weights using the activities and errors at $t+1$ and the weights at t.

It is worth noting that one may also implement a PC network where information flows in the opposing direction from that described [36]. That is, predictions flow up the network, originating at the stimulus, and the higher-level regions send error signals down the network. Mathematically, the conditional probability terms in Eq. (1) would be $p(\vec{v}_{i+1}|\vec{v}_i)$ and the equation would have a leading $p(\vec{v}_0)$ term. For a model of the brain, it is intuitive that the higher-level layers make predictions as opposed to the lower levels [28].

3 Neural Learning and Inference with PC Networks

3.1 Neural Learning

Training a predictive coding neural network for facial perception is a supervised learning problem. To train a neural network, we would normally tune its connection weight matrices based on the derivatives of a loss function. With PC networks, we use Eq. (12c) to update all W_i. To this end, we set the values of $\vec{\theta}_0$ and $\vec{\theta}_4$ to the inputs and targets, respectively. These values are not updated during training. Note this means $\vec{\epsilon}_4$ is also not updated, but $\vec{\epsilon}_0$ is. To see this, refer to Fig. 2 and note that the $\vec{\epsilon}_0$ node still connects to the first layer, thus the values it stores are important. This also implies that W_0 will be updated using Eq. (12c). For all other $\vec{\theta}_i$, $\vec{\epsilon}_i$, and W_i, we perform gradient ascent using Eqs. (12a) to (12c) for a set number of iterations. Afterwards, we update the inputs and outputs to the next sample and repeat.

This learning method is inherently sequential; we need to run the network to equilibrium for one sample before moving on to the next. However, there is a way to parallelize the network. If we pass samples through the network in batches as in a standard feed-forward neural network, we run into the issue of the connection weight matrices and the fact that all the samples in the batch share the same set of connection weights. For feed-forward networks, we only update the connection weights after running the network and computing the gradients. With Eq. (12), we are supposed to update W_i at every iteration. To remedy this, we can keep the connection weights constant throughout one batch, store the gradients, and then update W_i at the end of one batch. This greatly speeds up training and allows us to use more advanced optimization techniques for updating the connection weights such as momentum-based techniques.

The programming implementation was built off code originally made by Junteng Zheng at the NeuroCognitive Computing Lab at the University of Waterloo.

3.2 Inference

During inference, we are no longer training so we do not want to update W_i. Only Eqs. (12a) and (12b) will play a role in the dynamics. When we want the network to perform inference, we clamp the input layer but not the output layer. This means that $\vec{\theta}_4$ is now a free variable and is included in our optimization problem Eq. (3). Following the pattern in Fig. 2, there is now an excitatory connection from $\vec{\epsilon}_3$ to $\vec{\theta}_4$, and $\vec{\theta}_4$ can be updated using an equation similar to Eq. (12a). However, this leaves the question of what to do with $\vec{\epsilon}_4$; since $\vec{\theta}_4$ is no longer constant, the error nodes will now receive a signal input. The gradients of F with respect to the activities/errors of the last layer are

$$\nabla_{\vec{\theta}_4} F = -\vec{\epsilon}_4 + h_3'(\vec{\theta}_4) \odot (W_3^T \vec{\epsilon}_3) \tag{13a}$$

$$\nabla_{\vec{\epsilon}_4} F = \vec{\theta}_4 - \vec{\epsilon}_4, \tag{13b}$$

recalling that Σ_i is the identity matrix for all i. We may parallelize the network as in Section 3.1 since W_i are all constant.

Since there is no layer after the fourth, there is no value to compare its prediction against hence the missing term in Eq. (13b). Recall that the MLE occurs at an equilibrium point, so by setting Eq. (13b) to 0 we get $\vec{\theta}_4 = \vec{\epsilon}_4$ and Eq. (13a) becomes $\partial F/\partial \vec{\theta}_4 = -\vec{\theta}_4 + \nabla h_3(\vec{\theta}_4) \odot (W_3^T \vec{\epsilon}_3)$. The activity of the fourth layer is constantly driven towards 0 due to the decay term. To avoid this, we completely remove the fourth error node from the system. The activity update equations then become

$$\nabla_{\vec{\theta}_4} F = h_3'(\vec{\theta}_4) \odot (W_3^T \vec{\epsilon}_3) \tag{14}$$

for that layer. This decision is also consistent with the interpretation of $\vec{\epsilon}_i$ as the difference between the activity of the ith layer and the prediction of the following layer; if there is no layer afterwards, there cannot be a prediction error.

Our goal is to replicate the behaviour of human face perception in response to different numbers of parafoveal features, specifically the latent reaction time when presented with a face image with few features. To do this, we need a way to measure the response time of our neural network. The use of the Euler method for the gradient ascent of F provides a natural way to measure this time. With a constant step size, α, and t iterations of gradient ascent, we can record the time passed as $t \cdot \alpha$. We will record t as the number of iterations it takes for the activity of the output neurons to reach some threshold.

The choice to use a threshold merits some justification. A single electrode in an EEG study records a scalar number after every time step. It takes the activities of many millions

of neurons and distills them into one observation i.e. electrical potential. The layers in our neural network are represented by vectors, of which each element corresponds to a single neuron. There are two neurons in our output layer, one that becomes active when the network is shown a face and one that becomes active when it is shown another object. As the network becomes more certain that the image it sees is a face (or not a face), the appropriate output neuron becomes more active. Recording the time when the threshold of activity is met is analogous to the EEG machine recording a voltage potential signal. Moreover, there are results from theoretical decision-making that support the use of thresholds when comparing the activities of neuronal populations [4].

4 Training the Network for Facial Perception

4.1 Data

We have 32 unique images of human faces, half male and half female. These images are artificially created using computer software; they are centred and have identical head outlines. They are the same images used by Winward et al. [38]. These images form the basis for what we use to train the network.

In order to classify faces, the network needs other non-face images to compare against. As humans learn, they are bombarded with countless classes of objects and variations. To replicate this learning process, we include several object types in our dataset. The ImageNet dataset contains tens of thousands of classes, making it an ideal source of non-face images to use [6]. We use a small subset of these classes. We pick the classes such that the associated images are unlikely to include a face or a person by accident. For example, the 'cowboy hat' class would not be appropriate since it likely includes many pictures of people wearing cowboy hats. Table 1 compiles all the classes we use during training. In total, we have 664 ImageNet images in our dataset.

We want to ensure that the PC network is learning to distinguish faces by the presence of parafoveal features and facial contours. There is no guarantee that our neural network will use this information when classifying images. It may compare the smoothness of the images, identify the blank background associated with the face images, or recognize the distribution of pixel intensities. To this end, we need to standardize our images.

First, we resize all images to 68×100, convert colour images to grayscale, and normalize the pixel values to the interval $[0,\ 1]$. We then take the two-dimensional Fourier transform of the images. We set the smallest frequencies in a 2×3 rectangle around the center of

Table 1: ImageNet class codes and labels used for training.

n11939491	daisy
n04330267	stove
n04326547	stone wall
n04328186	stopwatch, stop watch
n04330267	stove
n09468604	valley, vale
n09472597	volcano
n04522168	vase
n02974003	car wheel
n03220513	dome
n03223299	doormat, welcome mat
n03240683	drilling platform, offshore rig
n03355925	flagpole, flagstaff
n03930630	pickup, pickup truck
n03837869	obelisk
n03773504	missile
n03530642	honeycomb
n02701002	ambulance
n02747177	ashcan, trash can
n02727426	apiary, bee house
n02980441	castle

the spectrum to 0. This removes low-frequency content, such as overall brightness and brightness gradients. Thus, the network should learn object shapes and contours, which means it should learn facial features.

To further ensure that the network learns to recognize faces using the presence and orientation of features, we will include shuffled faces. This shuffling is done pixel-wise and block-wise. For pixel-wise shuffling, we randomly permute all the pixels of the face images. For block-shuffling, we partition the images into sixteen 25×17 blocks and shuffle these. We label these shuffled faces as non-face images. The resulting images have the same distributions of pixel intensities. This makes sure the network does not simply learn to recognize those distributions.

One technique that often greatly improves the performance of neural networks, and machine learning models in general, is data augmentation [22]. For this project, we restrict our augmentation to rotations, translations, and reflections. This sort of augmentation is

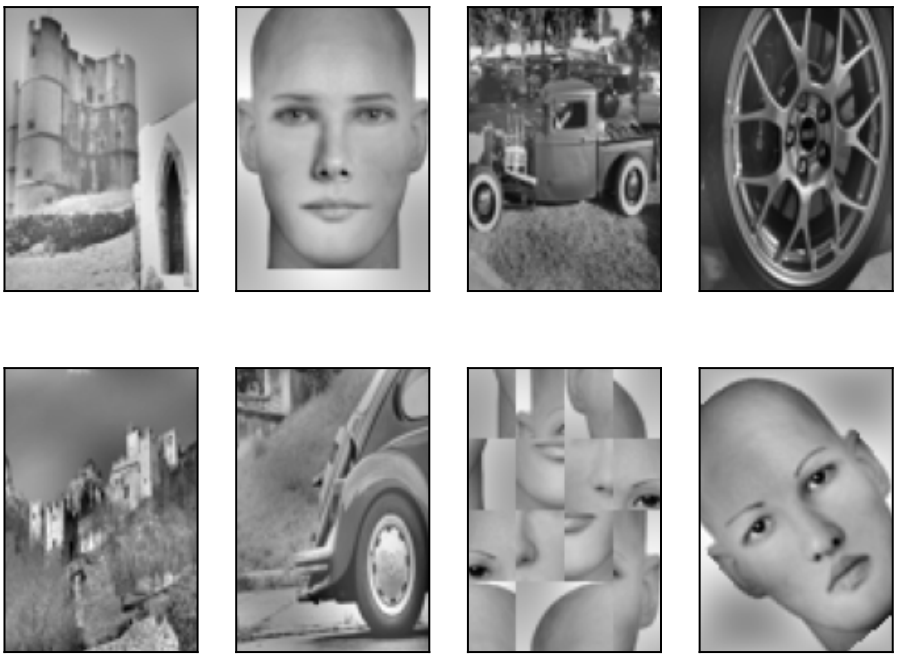

Figure 3: Samples of training images after processing and augmentation.

biologically plausible since human brains learn to recognize faces by seeing them from various positions and orientations. For each face image, we include its horizontal reflection (a reflection across the veritcal axis of symmetry). For each of these now 64 images, we create three rotated versions by randomly choosing a rotation from a uniform distribution over $[-45°, 45°]$ and three random vertical translations in the range of -15 and $+30$ pixels. These ranges are chosen to ensure that all parafoveal features remain visible for training. Fig. 3 shows samples of face and non-face images after processing and augmentation.

The images are labelled using one-hot encoding. This works naturally with our network architecture since we have a two-dimensional output layer. Non-face images are labelled $(1,\ 0)$ and face images are labelled $(0,\ 1)$.

Table 2 shows the breakdown of the training dataset in terms of the kinds of images and their labels. Just over a third of the augmented dataset is composed of face images.

Table 2: Training dataset composition. The face category includes the rotated and translated faces.

Image Category	Frequency	Label
Face	448	(0, 1)
ImageNet	664	(1, 0)
Block-Shuffled Faces	64	(1, 0)
Pixelwise-Shuffled Faces	64	(1, 0)

4.2 Missing Features

In addition to the face images, we also have access to modified versions that are missing parafoveal features. The combinations of features can be found in Table 3. This table also shows the grouping of feature combinations by parafoveal feature count. In the study by Winward et al., gaze fixation is enforced to reduce experimental error; face images are flashed with one of the eyes always kept in the participant's fovea [38]. The number of features in addition to this eye is the number of parafoveal features.

Table 3: Grouping of facial feature combinations by parafoveal feature number.

Number of Parafoveal Features	Feature Combinations
0	One eye
1	Two eyes One eye, nose One eye, mouth
2	Two eyes, nose Two eyes, mouth One eye, nose, mouth
3	Full face

Human eyes perform saccades, rapid eye movements, which introduce experimental noise to EEG studies. To reduce their presence, experimenters enforce fixation in their participants [39]. Fixation is achieved automatically in our neural network, since there is no concept of 'in focus' or 'out of focus'. For easier, clear comparisons to our motivating study, we will continue using the parafoveal feature notation.

4.3 Training Algorithm and Hyperparameters

We train a population of 45 predictive coding neural networks. This is the same as the number of participants in the study by Winward et al. [38]. We train each network on the dataset for 18 epochs, using batch sizes of 64. The activities and errors are updated using a step size of $\alpha = 0.05$. For each sample, we perform 500 steps of gradient ascent using Euler time-stepping.

The connection weight matrices are updated using the process outlined in Section 3.1. We use the Adam optimizer with a learning rate of 10^{-4} and exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [17].

4.4 Inference Algorithm and Hyperparameters

When performing inference, we update the activities and errors using a smaller step size, $\alpha = 0.005$. We do this because we want a finer-grain measurement of response time, as opposed to training when we simply want the network to learn its connection weights. For each sample, we run the network until one of the two output neurons reaches a threshold as described in Section 3.2. The recorded response time is $t \cdot \alpha$, where t is the number of iterations. If after $20,000$ iterations the network does not reach the threshold, we stop the network and do not record the response time. However, this maximum number of iterations was never reached throughout our experiment.

The threshold we choose is 1. This is because our network is trained using one-hot encoded labels, so the output neurons should have activities of one when the network successfully classifies an image.

We always initialize the hidden layers with activities of 0 and the output layer with activities of 0.5. This is done to not bias the output since the expected targets are always $(0,\ 1)$ or $(1,\ 0)$.

5 Results

5.1 Accuracy

After training, the 45 networks achieve an average accuracy of 0.879 on the training set with a standard deviation of 0.009. Most of this accuracy comes from the networks' ability

to recognize face images. The networks correctly categorize all face images, without fail. This extends to faces with different combinations of features. The models have perfect recall, which is what we had hoped for. The models also correctly classify all pixel-shuffled images and classify block-shuffled faces as non-faces with accuracies ranging from 0.95 to 1.0.

Most of the prediction error comes from the ImageNet images, which is to be expected; these images are more varied and contain more details. When testing on ImageNet's validation dataset, again using only the classes in Table 1, we get a mean test accuracy of 0.658 with a standard deviation of 0.046.

Our goal is to train PC neural networks to recognize faces using the presence of facial features. It is more important for our purposes that this is the case as opposed to training models with high accuracies. This is why we include much of the data processing in Section 4.1. Had we not filtered the training images or included shuffled faces, the network could very easily use unintentionally diagnostic, but not face-related, image features to learn to distinguish faces from ImageNet images.

5.2 Activity Phase Space

In Section 4.4, we described the process by which we measure the response times of networks. To summarize, we clamp the activities of the input layer to match the pixel values of an image and run the network until we reach a threshold. We measure the response times as the number of Euler time steps we take, multiplied by the step size.

Fig. 4 plots the L^2 norms of each layer's activity and the derivative of this norm[2]. The 0th layer is not shown since it is held constant, so plots of the norm of the activity and its derivative would not be informative. To create these plots, we randomly select one of the 45 networks and present it with a random full face as input. We repeat using the same network and face with two eyes, only one eye and a mouth, etc. through all the face conditions. This is the same face identity, the only difference is the presence or absence of parafoveal features.

The activities and derivatives of each layer follow a similar fluctuating pattern before tapering off to some equilibrium. The final layer is an exception because it has no error feedback to taper its growth. As the third layer reaches some non-zero equilibrium, the output layer will continue to receive constant input. This is evidenced by the fourth layer's

[2]The L^2 norm is defined for any vector $\vec{x}$ of size d as $\|\vec{x}\|_2 = \left(\sum_{i=1}^{d} x_i^2\right)^{1/2}$.

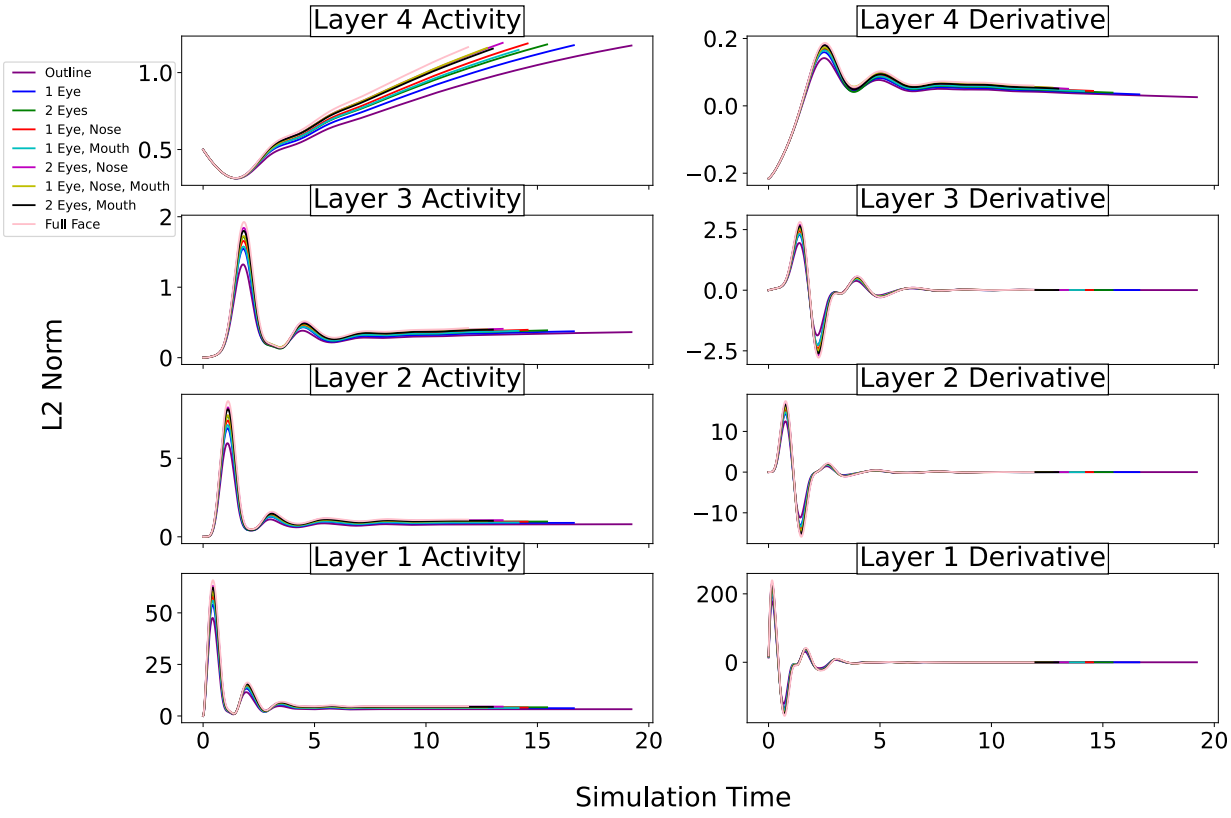

Figure 4: L^2 norm of the activities and derivatives for each layer in a PC network. The 0th layer is not shown since it is held constant.

roughly constant derivative. It takes longer for the fluctuations in the hidden layers' activities to settle the further up the network they are from the input. This is because the input layer is clamped, so the first hidden layer quickly adapts to this information before passing it on to the next layer, which must adapt before passing it on again. While the L^2 norm of the activities is greater for the first layer than the others, it is because it has more neurons than them (300 as opposed to 200 and 100). The only differences between the full-face condition and the partial feature conditions are the magnitudes of the norms; all curves have the same shape with no horizontal translation between the two. As the number of features decreases, so does the magnitude of the norms. We also include the network's response when shown just the outline of a face (one with no features). It has the smallest amplitude and greatest response time, which is consistent with the pattern we observed.

Fig. 5 shows the activities of the neurons in the output layer under similar circumstances as in Fig. 4; the network is shown both a full face and a face with one eye and one mouth. A different network and different images from Fig. 4 were randomly selected. The solid lines plot the activity of the neuron in the output layer that activates when the network is shown a face. The dashed line is for the neuron that activates when the network is shown anything else.

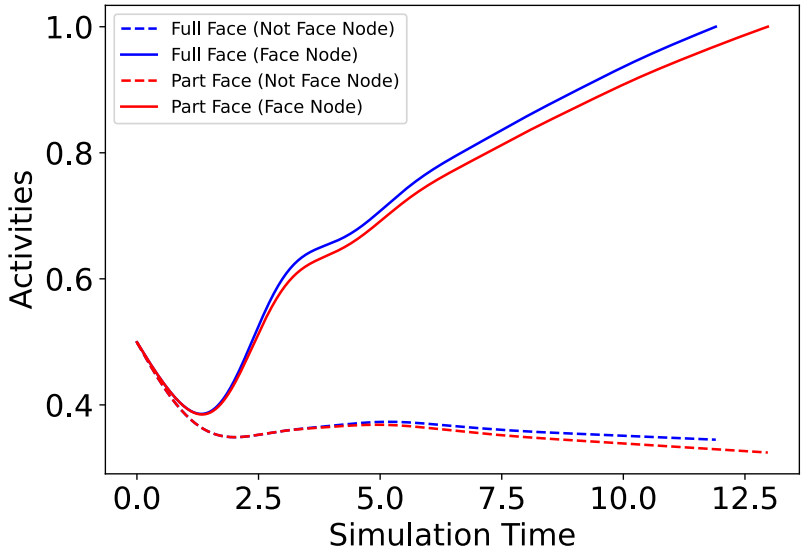

Figure 5: Activities of each of the two neurons in the output layer. The blue curves are the output neurons for a network when shown a full face. The red curves are the output neurons when shown a face with one eye and a mouth, labelled as a part face in this figure. The solid lines are the activities of the neurons associated with the presence of a face, the dashed lines are for the other neurons.

The face neuron's activities grow while those of the other neuron remain roughly constant, even slightly decreasing. The initial dip in activities is likely an artifact of initializing both neurons at 0.5. Information from the rest of the network harshly corrects this initial guess, then the activities settle into regular behaviour as the network stabilizes. The general shape of this curve is what we expect to see. The network processes information from the input image and gathers evidence that the image is a face. This evidence is reflected by the increasing solid curves in Fig. 5, all the while there is no evidence convincing the network that the input image may not be a face. Again, the only difference between the

Table 4: Mean response times and standard deviation across networks for each feature combination treatment. Recall, that there are 45 networks.

Treatment	Mean	Standard Deviation
1 Eye	15.26	1.02
2 Eyes	13.54	0.86
1 Eye, Nose	14.31	0.94
1 Eye, Mouth	13.54	0.77
2 Eyes, Nose	12.78	0.80
1 Eye, Nose, Mouth	12.79	0.70
2 Eyes, Mouth	12.17	0.64
Full Face	11.57	0.60

two conditions is the amplitude of the curves, not their shapes or their positions on the graph.

5.3 Feature Combination Response Times

For each network, we measure the response times when presented with the 32 face images. We take the mean of these response times as our data point. Each of the 45 networks is treated as an individual participant in the experiment. This is repeated for all eight feature combinations in Table 3. This is a repeated measures experiment where the trained neural networks are the experimental units [19]. The treatments are the eight different feature combinations. The measurements are the mean response times when presented with images of faces whose feature combinations correspond to the treatment. We do not include the face outline condition in our statistical analysis, even though we include it in Fig. 4. This is because all face identities share the same outline, so we only have one data point for each of the 45 neural networks.

The data analysis in this section and in Section 5.4 is done using the Pingouin software package for Python [35]. Specifically, we use the `rm_anova`, `normality`, `sphericity`, and `pairwise_tests` functions.

Fig. 6 shows the distribution of response times of the networks for different treatments. The distributions are unimodal without a large spread. The response times across treatments are on the same order of magnitude, but with some clear differences. These differences are clearer in Fig. 7. The white circles on the boxplot indicate the treatment's mean. The boxplot and the histograms tell us the distribution of response times tends

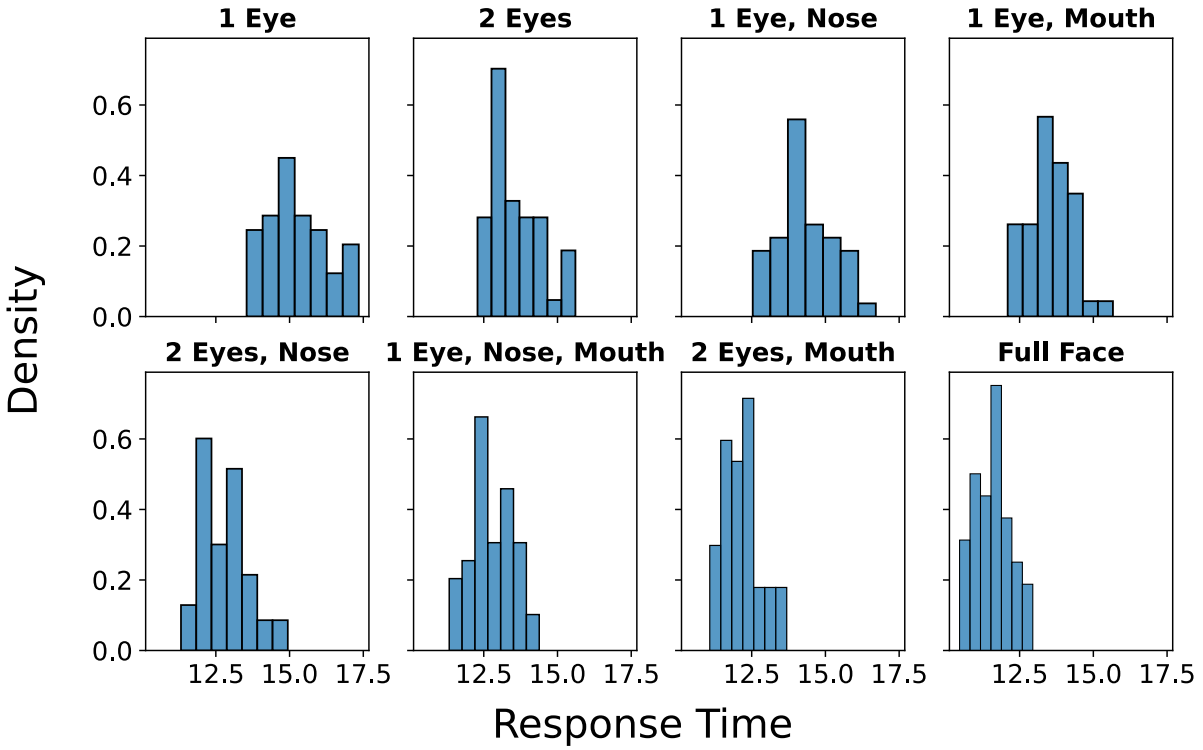

Figure 6: Histogram of average response times to different feature combinations.

to be right-skewed towards longer response times. The skewness tends to be small, as evidenced by the presence of only two outliers out of 360 observations across all treatments and participants.

Table 4 tabulates the means of the response times for each treatment, across the 45 networks. This is a mean of means.

To formalize this difference in means, we apply a repeated measures analysis of variance (ANOVA). We use the Greenhouse-Geisser correction for the p-value to correct for the lack of sphericity in the data [10]. Another sufficient requirement is normality across each treatment. Using the Shapiro-Wilk normality test, we do not find enough evidence to reject normality at the 0.05 significance level for any treatments, except for the two eyes condition, see Table 5. Given the relatively small skewness of the data for that treatment and the normality of all other treatments, this violation is not likely to significantly increase the probability of a Type I error [14]. The repeated measurement ANOVA finds significant evidence that there is a difference in means ($F(7,\ 308) = 655.88,\ p\text{-value} = 1.85 \cdot 10^{-69}$).

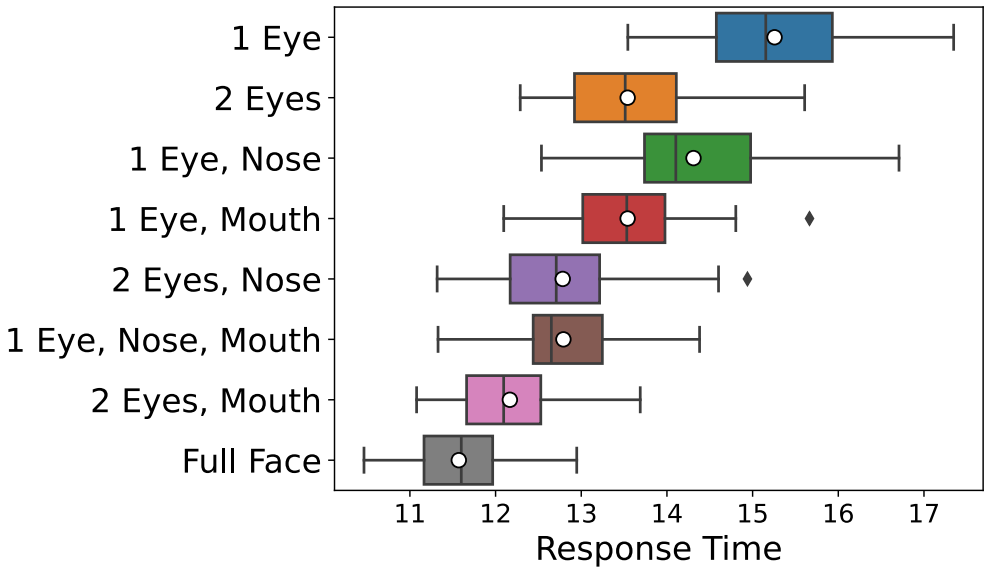

Figure 7: Boxplot of average response times to different feature combinations. The boxplot follows standard conventions [19]. The white circle indicate the treatment's mean response time.

Table 5: p-values for the Shapiro-Wilk normality test applied to feature combinations data.

Treatment	p-value
1 Eye	0.200260
2 Eyes	0.007801
1 Eye, Nose	0.556800
1 Eye, Mouth	0.493095
2 Eyes, Nose	0.224850
1 Eye, Nose, Mouth	0.540330
Full Face	0.566282
2 Eyes, Mouth	0.214591

To further explore this difference, we apply pairwise paired samples t-tests to each pairing of treatments. Since we are performing multiple comparisons, we must adjust the p-value to correct for Type I errors. To do so, we use the Bonferroni correction, which amounts to multiplying the p-value by the number of tests we perform, 28 in our case [5]. Table 6 summarizes the results from these tests. All comparisons, except for two, provide significant statistical evidence against the null hypothesis. These exceptions are the entries in the table with p-values of 1.0. In Fig. 7, these comparisons correspond to the orange vs. red boxplots and the purple vs. brown boxplots. It is visually clear that these means are similar.

5.4 Parafoveal Feature Response Times

The next question we seek to answer is, how does the number of facial features affect the response times of the networks, regardless of the type of feature? EEG studies indicate a positive linear relationship between the parafoveal feature number and the response time [38]. This design is also a repeated measures experiment, albeit with different treatments. The four parafoveal feature number treatments $(0-3)$ are created by combining the inference results of varying feature combinations, as outlined in Table 3.

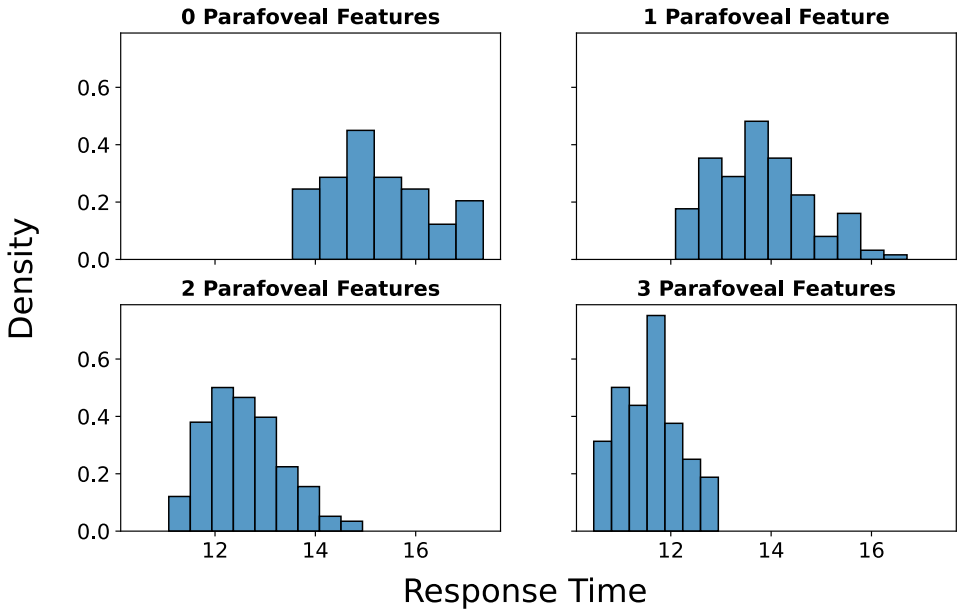

Figure 8: Histogram of average response times to different parafoveal feature numbers.

Table 6: Results of paired samples t-tests for feature combination. T is the test statistic. The p-values are corrected using a Bonferroni correction.

Treatment A	Treatment B	p-value	DOF	T
1 Eye	1 Eye, Mouth	$1.31 \cdot 10^{-27}$	44	27.05
1 Eye	1 Eye, Nose	$6.35 \cdot 10^{-23}$	44	20.77
1 Eye	1 Eye, Nose, Mouth	$1.06 \cdot 10^{-31}$	44	33.86
1 Eye	2 Eyes	$4.73 \cdot 10^{-26}$	44	24.81
1 Eye	2 Eyes, Mouth	$1.73 \cdot 10^{-32}$	44	35.33
1 Eye	2 Eyes, Nose	$8.22 \cdot 10^{-30}$	44	30.54
1 Eye	Full Face	$2.09 \cdot 10^{-34}$	44	39.18
1 Eye, Mouth	1 Eye, Nose	$1.27 \cdot 10^{-12}$	44	-10.89
1 Eye, Mouth	1 Eye, Nose, Mouth	$2.67 \cdot 10^{-23}$	44	21.23
1 Eye, Mouth	2 Eyes	1.0	44	-0.00
1 Eye, Mouth	2 Eyes, Mouth	$9.77 \cdot 10^{-27}$	44	25.77
1 Eye, Mouth	2 Eyes, Nose	$4.41 \cdot 10^{-10}$	44	9.00
1 Eye, Mouth	Full Face	$4.81 \cdot 10^{-30}$	44	30.93
1 Eye, Nose	1 Eye, Nose, Mouth	$1.08 \cdot 10^{-27}$	44	27.18
1 Eye, Nose	2 Eyes	$1.14 \cdot 10^{-12}$	44	10.92
1 Eye, Nose	2 Eyes, Mouth	$1.93 \cdot 10^{-26}$	44	25.35
1 Eye, Nose	2 Eyes, Nose	$1.54 \cdot 10^{-26}$	44	25.49
1 Eye, Nose	Full Face	$1.21 \cdot 10^{-32}$	44	35.63
1 Eye, Nose, Mouth	2 Eyes	$5.07 \cdot 10^{-11}$	44	-9.68
1 Eye, Nose, Mouth	2 Eyes, Mouth	$1.19 \cdot 10^{-13}$	44	11.70
1 Eye, Nose, Mouth	2 Eyes, Nose	1.0	44	0.12
1 Eye, Nose, Mouth	Full Face	$3.91 \cdot 10^{-27}$	44	26.35
2 Eyes	2 Eyes, Mouth	$9.38 \cdot 10^{-28}$	44	27.27
2 Eyes	2 Eyes, Nose	$1.74 \cdot 10^{-23}$	44	21.46
2 Eyes	Full Face	$6.06 \cdot 10^{-32}$	44	34.30
2 Eyes, Mouth	2 Eyes, Nose	$7.29 \cdot 10^{-13}$	44	-11.08
2 Eyes, Mouth	Full Face	$1.48 \cdot 10^{-23}$	44	21.54
2 Eyes, Nose	Full Face	$6.93 \cdot 10^{-28}$	44	27.47

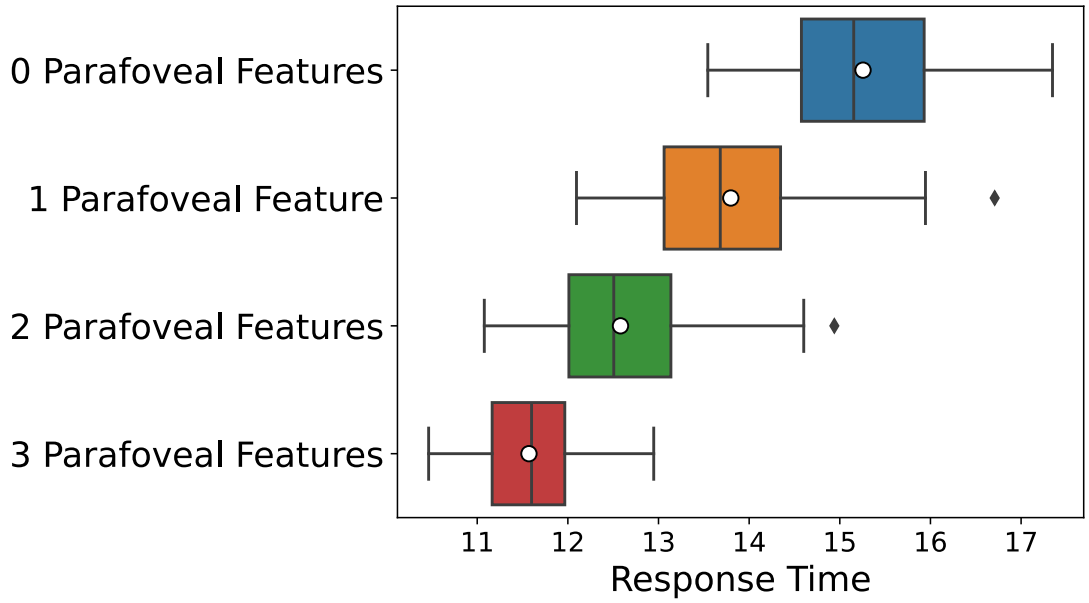

Figure 9: Boxplot of average response times to different parafoveal feature numbers. Same conventions as in Fig. 7.

Figs. 8 and 9 plot the distribution of response times for the different parafoveal number treatments. They follow the same conventions as Figs. 6 and 7. Here we can more clearly see the linear relation between feature number and response time, in addition to the difference in means.

Table 7 tabulates the means of the response times for each parafoveal feature treatment, across the 45 networks. This is a mean of means. The one and two parafoveal feature treatments are comprised of three feature combinations, so there are 135 data points.

Performing a repeated measures ANOVA with Greenhouse-Geisser correction on the parafoveal features treatments indicates significant statistical difference between the means ($F(3,\ 132) = 1070.81,\ p\text{-value} = 1.44 \cdot 10^{-53}$). A Shapiro-Wilk normality test finds that only the one parafoveal feature treatment is not normal at the 0.05 significance level. This is unlikely to affect our results for the same reasons outlined in Section 5.3. The normality test results are summarized in Table 8.

Pairwise paired samples t-tests with the Bonferroni correction indicate that the mean response times across all parafoveal feature numbers are statistically different from one another. The results of the t-tests are not surprising given Section 5.3. The only pairs of means which were not significantly different were found within the same parafoveal feature

Table 7: Mean response times and standard deviation across networks for each parafoveal number treatment.

Treatment	Mean	Standard Deviation	Data Points
0 Parafoveal Features	15.26	1.02	45
1 Parafoveal Feature	13.80	0.93	135
2 Parafoveal Features	12.58	0.77	135
3 Parafoveal Features	11.57	0.60	45

Table 8: p-values for the Shapiro-Wilk normality test applied to parafoveal feature number data.

Treatment	p-value
0 Parafoveal Features	0.20
1 Parafoveal Feature	0.0075
2 Parafoveal Features	0.069
3 Parafoveal Features	0.57

number treatment. These results are summarized in Table 9.

5.5 Discussion

The results from Sections 5.3 and 5.4 show that predictive coding neural networks replicate the behaviour of human brains when trained to recognize faces. The difference in response times across parafoveal feature treatments, and the smaller differences within each treatment, are consistent with experimental results [38].

Table 9: Results of paired samples t-tests for parafoveal feature number. T is the test statistic. The p-values are corrected using a Bonferroni correction.

Treatment A	Treatment B	p-value	DOF	T
0 Parafoveal Features	1 Parafoveal Feature	$1.01 \cdot 10^{-26}$	44	34.02
0 Parafoveal Features	2 Parafoveal Features	$1.76 \cdot 10^{-30}$	44	36.65
0 Parafoveal Features	3 Parafoveal Features	$4.48 \cdot 10^{-35}$	44	39.18
1 Parafoveal Feature	2 Parafoveal Features	$3.72 \cdot 10^{-24}$	44	40.29
1 Parafoveal Feature	3 Parafoveal Features	$1.30 \cdot 10^{-32}$	44	43.36
2 Parafoveal Features	3 Parafoveal Features	$1.49 \cdot 10^{-28}$	44	47.67

EEGs measure voltage potentials across the scalp, which are used to extract ERPs and make inferences about activity inside the brain. It is important to note that these measurements do not directly measure the activation levels of individual neurons; an EEG measures the electrical fields of neuronal populations after they are propagated and distorted through different mediums before reaching the scalp. This distinction is crucial because, in our PC network experiment, we *are* measuring individual neuronal activity directly. The decreased ERP results found by Winward et al. [38] and the results in Figs. 4 and 5 are similar but present different information. Figs. 4 and 5 provide a hypothetical explanation for the ERP latency effects but not their amplitudes.

The neurons in our PC neural networks are less active when shown faces with fewer parafoveal features, as compared to being shown a full face. Because the neurons are not as active, it takes longer for them to reach a threshold where a signal can be registered. We can interpret this behaviour as the network being 'less sure' of what it sees. Biologically, this means the neurons fire less often, slowing down the rate of charge buildup. This is not what is observed in experiment [38]. Our results do not explain this discrepancy but potential explanations and future avenues of research will be discussed in Section 6. In contrast, the delay in response times can be explained. Since the neurons collect charges more slowly, it will take longer for the population to reach the state where it creates the face signal charge. Whatever that state is, the results from this project cannot suggest; it may be a certain amount of information exchange between layers or interactions of neurons within a layer, something this predictive coding model does not account for.

In the study by Winward et al., they also recorded EEG response times when participants were shown images of isolated eyes (just one eye on a white background, no outline or other features) [38]. These types of images had the greatest delay in response times. We cannot replicate these results because our networks classify these isolated eyes as non-face images. This is an understandable classification since it is reasonable to say that an isolated eye is not actually a face.

6 Conclusion

In this project, we implemented a predictive coding neural network as a mechanistic model for facial perception in the human brain. We trained the network using a mixture of face images, ImageNet images, and data augmentation. The accuracy of the model was not high in comparison to how other models would perform on the dataset, but accuracy is not a good measure of model aptitude for our purposes. Using analogies to biology and EEGs, we devised a way to measure the network's response time when presented with

stimuli. We find statistically significant increases in response times when exposing the network to face images that are missing features versus full-face images. These findings are consistent with experimental evidence of how the human brain works [38]. They also provide a possible explanation for the delayed response time's origin, and make predictions for future experiments. As exciting as these results are, they do not come without caveats.

During training, we used very regular images of human faces. We attempted to address this limitation by including data augmentation. While this certainly helped the network recognize faces from the presence of facial features, it is a far cry from how human brains are trained. From birth, people are exposed to countless faces of different shapes and sizes, and at various angles in three-dimensional space. In addition, we see many more kinds of objects than listed in Table 1. Training our neural network is a limited approximation of this human learning that takes many years. It would be interesting to train a predictive coding neural network using a much larger, richer dataset and compare the results to those in this paper.

Our predictive coding neural network architecture is mathematically derived from our assumptions about the layers' probability distributions and their conditioned variables. Setting all Σ_i to the identity matrix is a further restriction. Besides these restrictions, our overall model is a simple interpretation of the human brain. The number of neurons in each layer and the number of layers are simplifying assumptions since there are no hard delimitations between regions of the human brain. In a human brain, there are many connections between neurons within each layer, which would greatly alter the behaviour of the network.

In our neural network architecture, we label the fourth and final layer as 'higher levels'. This is a useful abstraction allowing us to interpret the behaviour of the network. In reality, there are more than just two neurons after the pFUS; these higher-level layers are used for more than just facial perception in the actual human brain. With this simplification, we disregard all higher-level activity that may contribute to observed phenomena.

Another aspect of the human brain not reflected in our architecture is skip connections. Our hierarchical layering of the network, enforced by Eq. (2), prevents us from including this. There is evidence for a connection between V1 and the pFUS [11]. This would change the behaviour plotted in Fig. 4; the signals in each layer would no longer follow this delayed pattern. Our PC network can be modified to include this skip-connection architecture.

Mathematically, the implementation of skip connections poses a non-trivial question: how does the probability distribution in Eq. (2) change? For brevity, we will consider a skip between the first and third layers of Fig. 2 only, but this discussion can be extended to connections between any other layers. Thus, the distribution of the first layer's activity

may be

$$p(\vec{v}_1|\vec{v}_2,\vec{v}_3) \propto \mathcal{N}(\vec{v}_1; g_{12}(\vec{v}_2, W_{12}), \Sigma_{12}) \cdot \mathcal{N}(\vec{v}_1; g_{13}(\vec{v}_3, W_{13}), \Sigma_{13}), \tag{15}$$

where variables with subscript $1i$ correspond to the interaction between the first and the ith layer. Alternatively, the probability could be

$$p(\vec{v}_1|\vec{v}_2,\vec{v}_3) = \mathcal{N}(\vec{v}_1; g_{12}(\vec{v}_2, W_{12}) + g_{13}(\vec{v}_3, W_{13}), \Sigma_1). \tag{16}$$

In deriving F and the parameters' update equations from Eq. (15), we would find that we need two error nodes for the first layer, one to compute the prediction error for layer two and one to compute the prediction error for layer three. From Eq. (16), we would find that the activities from the second and third layer both feed into the first layer's error node. The former can be interpreted as the first layer needing to simultaneously predict the activities of both the second and third layers, the latter means that the first layer needs to predict the sum of their activities.

It is not immediately obvious which of these interpretations is most biologically plausible. Should the distinction become clearer, it would be natural to enhance our predictive coding model of the visual system with this skip connection. This is only one of the numerous avenues of exploration we may take in applying predictive coding models to the brain. Any combination of these improvements would be exciting and worth pursuing in future work.

Besides changes to the architecture, there are numerous experiments left to run using trained networks. How does permuting the facial features affect the network's perception of faces? What about inverting the faces? The answers to these questions and others provide predictions that can be verified or refuted using human experiments to test our predictive coding model.

References

[1] Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. Canonical microcircuits for predictive coding. *Neuron 76*, 4 (2012), 695–711.

[2] Bennett, S. R. Linear Algebra for Data Science, 2021. Available at https://shainarace.github.io/LinearAlgebra/.

[3] Bogacz, R. A Tutorial on the Free-Energy Framework for Modelling Perception and Learning. *Journal of Mathematical Psychology 76* (2017), 198–211. Model-based Cognitive Neuroscience.

[4] Bogacz, R., Brown, E., Moehlis, J., Holmes, P., and Cohen, J. The Physics of Optimal Decision Making: A Formal Analysis of Models of Performance in Two-Alternative Forced-Choice Tasks. *Psychological review 113* (10 2006), 700–65.

[5] Bonferroni, C. *Teoria statistica delle classi e calcolo delle probabilità*. Pubblicazioni del R. Istituto superiore di scienze economiche e commerciali di Firenze. Seeber, 1936.

[6] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), Ieee, pp. 248–255.

[7] Duchaine, B., and Yovel, G. A Revised Neural Framework for Face Processing. *Annual Review of Vision Science 1*, Volume 1, 2015 (2015), 393–416.

[8] Fan, Z. *Lecture Notes in Statistics 200: Introduction to Statistical Inference*. Stanford University, 2016. Available at https://web.stanford.edu/class/stats200/lectures.html.

[9] Friston, K. Learning and Inference in the Brain. *Neural Networks 16*, 9 (2003), 1325–1352.

[10] Greenhouse, S. W., and Geisser, S. On Methods in the Analysis of Profile Data. *Psychometrika 24* (1959), 95–112.

[11] Grill-Spector, K., Weiner, K. S., Kay, K., and Gomez, J. The Functional Neuroanatomy of Human Face Perception. *Annu Rev Vis Sci. 3* (07 2017), 167–196.

[12] Guth, A. *Lecture Notes: Relativistic Quantum Field Theory I.* Massachusetts Institute of Technology Physics Department, March 2008, ch. Dirac Delta Function as a Distribution.

[13] Hao, K. *This is how we lost control of our faces.* MIT Technology Review, 02 2021. Available at www.technologyreview.com/2021/02/05/1017388/ai-deep-learning-facial-recognition-data-history/.

[14] Harwell, M. R., Rubinstein, E. N., Hayes, W. S., and Olds, C. C. Summarizing Monte Carlo Results in Methodological Research: The One- and Two-Factor Fixed Effects ANOVA Cases. *Journal of Educational Statistics 17*, 4 (1992), 315–339.

[15] Huang, Y., and Rao, R. P. Predictive Coding. *Wiley Interdisciplinary Reviews: Cognitive Science 2*, 5 (2011), 580–593.

[16] Kelley, H. J. Gradient Theory of Optimal Flight Paths. *Ars Journal 30*, 10 (1960), 947–954.

[17] Kingma, D. P., and Ba, J. Adam: A Method for Stochastic Optimization. *CoRR abs/1412.6980* (2014).

[18] Mauchly, J. W. Significance Test for Sphericity of a Normal n-Variate Distribution. *Annals of Mathematical Statistics 11* (1940), 204–209.

[19] Mendenhall, W., Beaver, R. J., and Beaver, B. M. *Introduction to Probability and Statistics*, 10 ed. Duxbury Press, 1999.

[20] Millidge, B., Salvatori, T., Song, Y., Bogacz, R., and Lukasiewicz, T. Predictive Coding: Towards a Future of Deep Learning beyond Backpropagation? In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22* (7 2022), L. D. Raedt, Ed., International Joint Conferences on Artificial Intelligence Organization, pp. 5538–5545. Survey Track.

[21] Mumford, D. On the Computational Architecture of the Neocortex. *Biological Cybernetics 65*, 2 (Jun 1991), 135–145.

[22] Mumuni, A., and Mumuni, F. Data Augmentation: A Comprehensive Survey of Modern Approaches. *Array 16* (2022), 100258.

[23] Munakata, Y., and O'Reilley, R. *Computational Explorations in Cognitive Neuroscience.* Cambridge: MIT Press, 2000.

[24] Ororbia, A., and Kifer, D. The Neural Coding Framework for Learning Generative Models. *Nature Communications 13*, 1 (Apr 2022), 2064.

[25] Ororbia, A., Mali, A., Giles, C. L., and Kifer, D. Lifelong Neural Predictive Coding: Learning Cumulatively Online without Forgetting. In *Advances in Neural Information Processing Systems* (2022), S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., pp. 5867–5881.

[26] Petersen, K. B., and Pedersen, M. S. *The Matrix Cookbook*. Technical University of Denmark, Nov 2012.

[27] Price, B. H., and Gavornik, J. P. Efficient Temporal Coding in the Early Visual System. *Frontiers in Computational Neuroscience* (07 2022).

[28] Rao, R. P. N., and Ballard, D. H. Predictive Coding in the Visual Cortex: a Functional Interpretation of some Extra-Classical Receptive-Field Effects. *Nature Neuroscience 2*, 1 (Jan 1999), 79–87.

[29] Rapcsak, S. Z. Face Recognition. *Current Neurology and Neuroscience Reports 19*, 7 (May 2019), 41.

[30] Rozanov, Y. A. *Proability Theory: A Concise Course*. Dover Publications, 1969.

[31] Sellal, F. Anatomical and Neurophysiological Basis of Face Recognition. *Revue Neurologique 178*, 7 (2022), 649–653.

[32] Seth, A. K. Interoceptive Inference, Emotion, and the Embodied Self. *Trends in Cognitive Science 17* (2013), 565–573.

[33] Shapiro, S. S., and Wilk, M. B. An Analysis of Variance Test for Normality (Complete Samples). *Biometrika 52*, 3-4 (12 1965), 591–611.

[34] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708.

[35] Vallat, R. Pingouin: Statistics in Python. *Journal of Open Source Software 3*, 31 (2018), 1026.

[36] Wen, H., Han, K., Shi, J., Zhang, Y., Culurciello, E., and Liu, Z. Deep Predictive Coding Network for Object Recognition. In *Proceedings of the 35th International Conference on Machine Learning* (10–15 Jul 2018), J. Dy and A. Krause, Eds., vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 5266–5275.

[37] Whittington, J. C. R., and Bogacz, R. An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation 29*, 5 (05 2017), 1229–1262.

[38] Winward, S. B., Siklos-Whillans, J., and Itier, R. J. Impact of Face Outline, Parafoveal Feature Number and Feature Type on Early Face Perception in a Gaze-Contingent Paradigm: A Mass-Univariate Re-Analysis of ERP Data. *Neuroimage: Reports 2*, 4 (2022), 100148.

[39] Woodman, G. F. A Brief Introduction to the use of Event-Related Potentials in Studies of Perception and Attention. *Attention, Perception, & Psychophysics 72*, 8 (Nov 2010), 2031–2046.

[40] Zahid, U., Guo, Q., and Fountas, Z. Predictive Coding as a Neuromorphic Alternative to Backpropagation: A Critical Evaluation. *Neural Computation 35*, 12 (11 2023), 1881–1909.