Optimization of Bio-Inspired Deep Convolutional Neural Networks

Author: Luis Morales Layja
Supervised by: Benjamin Evans
Candidate number: 282549

24/08/2024

MSc Artificial Intelligence and Adaptive Systems
Department of Engineering and Informatics
University of Sussex

Word count: 11169

Acknowledgments

I would like to express my deepest gratitude, above all and beyond anyone else in this world, to my mother. Through her hard work, dedication, and sacrifices, she has given me the opportunity to pursue this master's degree, which I have diligently completed, to complete this research project, and to write these words today. Without her tireless effort, none of this would have been possible.

I would also like to thank my siblings for being the admirable people they are today. They have my admiration and inspire me daily to incorporate the best parts of them into myself.

To Carolina Salvador, who, through her way of being and her simplicity, has shown me that the world is not always as I perceive it, and thanks to her, I have been able to change and free myself in many different ways. Thanks to her, I have become the person I am today.

I want to personally thank my pets: Mia and Rigoberto. Rigoberto, in his passing, taught me three things: to love unconditionally, to value what you have (in its due time), and to have much more self-confidence in aspects of my life where I lacked it. Mia, in her passing, paved the way for me to pursue this master's degree.

Undoubtedly, I would like to personally acknowledge Saqib, Abysheck, Thenuka, Paras, and many others for making the journey through this master's program much more enjoyable and for sharing many pleasant moments with them. To Saqib, in particular, for becoming a close friend and for sharing with me deep and pleasant conversations.

To my friends in México, especially Christian and Gabriel, who have never forgotten me, and I know that we will share many more moments together.

Also to Dr. Rafael Laredo and Dr. Jesús Porfirio, from México, who have greatly contributed to shaping the person I am today.

Finally, to my supervisor Benjamin Evans, for giving me the opportunity to work on this interesting, exciting and valuable project.

To all the people mentioned in this acknowledgment, thank you very much. Without you, this project would not have been possible in the way it is. This achievement is as much yours as it is mine.

"Whether you think you can or think you can't, you're right."
– Henry Ford

Abstract

In this research project, two properties of the primate visual cortex were incorporated into the architecture presented by Evans, Malhotra, and Bowers [9], which is based on the VGG-16 architecture with fixed Gabor filters in its first convolutional layer. The introduced properties were a bottleneck, mimicking the reduction in ganglion cells as visual information passes from the retina through the optic nerve in primates, and recurrent connections, which are prevalent in the visual cortex. Four new models were developed: two with bottlenecks (one with 2 neurons and another with 4 neurons in the final convolutional layer), a third model incorporating recurrence in the first convolutional layer (using a convolutional Long Short-Term Memory architecture), and a fourth combining recurrent connections with the 2-neuron bottleneck. It was hypothesized that these modifications would enhance the model's generalization capacity and robustness to noise by creating an architecture more similar to the visual cortex, thereby narrowing the performance gap between humans and models. The results, based on accuracy in classifying CIFAR-10G and CIFAR-10 images across varying noise levels, suggest that combining bottleneck and recurrent connections in a single model does not improve performance as expected. However, recurrent connections alone significantly enhance the baseline model's performance in all tests, while the 2-neuron bottleneck provides advantages under certain noise conditions. These findings highlight the need for a more refined approach to successfully integrate these properties, ensuring that the benefits of each are preserved.

Contents

1 Introduction

1.1 Background Research and Motivation

In the last decades, deep learning models have been extensively utilized in a large number of tasks, such as feature extraction, pattern analysis and classification [56]. Among these models, Convolutional Neural Networks (CNNs) [43] stand out as a type of deep learning architecture specifically designed to process data structured as multiple arrays. For instance, a color image is composed of three 2D arrays containing pixel intensities across the three color channels: red, green, and blue [42].

Since their inception in the early 2000s [42], and following the breakthrough success of the AlexNet model in 2012 [37], CNNs have been successfully adopted in a broad spectrum of applications, ranging from intelligent video surveillance [63] to autonomous and robotic systems [75], sentiment analysis [46] and medical purposes [59], achieving remarkable accuracy in the detection, classification, segmentation, and recognition of objects, animals, and even human faces [41] [37] [65] [1] within images and videos. This widespread success has established CNNs as the leading architecture for nearly all recognition and detection tasks, revolutionizing the field of computer vision [42]. This achievement, coupled with the initial use of powerful GPUs [5], has spurred significant emphasis on developing and applying Deep Convolutional Neural Networks (DCNNs), and further research to enhance various characteristics and improve their performance [56].

In addition to their practical success, DCNNs have been employed as models of the human brain. Particularly, the ventral stream [47], which comprises the primary visual cortex (V1), secondary visual cortex (V2), visual cortex (V4), and inferotemporal cortex (IT) [39], as shown in Figure 1.

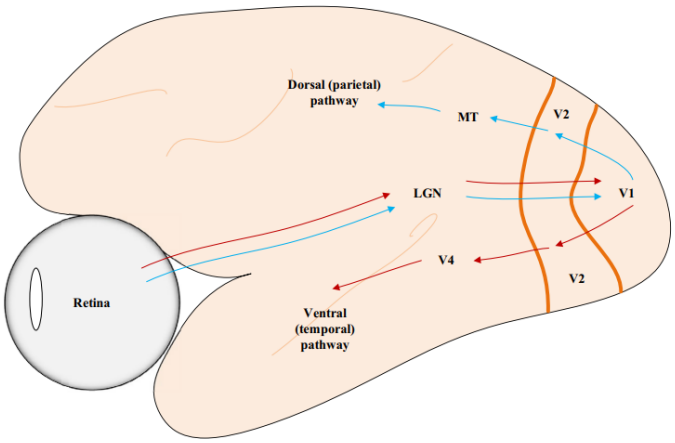

Figure 1: A simplified illustration of the Visual Stream and Dorsal Stream [49]. The red arrows indicate the pathway from the retina and Lateral Geniculate Nucleus (LGN) through the entire ventral stream. It is important to note that after V4, the flow continues to the IT, which is not explicitly depicted.

These areas of the brain are crucial for visual processing, and researchers have found that CNNs can mimic some of their properties, leading to significant insights. For exam-

ple, the firing patterns of neurons in V4 and IT have been shown to resemble the activity in CNN layers [73] [32]. Additionally, Gabor filters tend to emerge in early convolutional layers of CNNs trained end-to-end on various datasets, suggesting that these filters develop as a general mechanism rather than being specific to a particular type of data [74]. This aligns with the empirical findings of the neuroscientists Hubel and Wiesel [24] [25], which indicated that neurons in V1 in primates are specialized in detecting edges and contours across different orientations.

Furthermore, the accuracy of DCNNs in image classification tasks has soared since [37], even surpassing human performance and suggesting that CNNs are ideal candidates for modeling the visual stream in primates due to the similarity in performance [36] [32].

However, notable differences have been found in the ways that DCNNs learn to classify images compared to humans. For example, unlike [38], which finds that DCNNs effectively learn shapes and have high sensitivity to changes in them, followed by [57] that presents similar outcomes, and aligning with the known human shape-bias (humans tend to rely more on shapes rather than textures, size, and fine details in images) [3] [40], studies like those by [15] [50] [2] suggest otherwise, indicating that DCNNs exhibit a texture-bias. In some extreme instances, DNNs are capable of classifying images using just a single pixel, disregarding other features of the image [51]. Specifically, [2] found that, although DCNNs have access to some local shape features, such as orientation relationships, they do not form representations of the global shape of objects, showing a strong reliance on local textures, unlike humans who use global shapes. Moreover, in silhouette experiments, DCNNs demonstrated a limited ability to classify based on shape, and their performance was significantly inferior to that of humans.

These differences in how DCNNs learn from data can result in what is termed "shortcut learning" [13], where models over-rely on very specific features of training benchmarks to classify images, such as fine details like textures. This reliance can lead to poor generalization to out-of-distribution data, intolerance to noise, difficulty in identifying stylized representations [14] [15], and increased vulnerability to adversarial attacks [26].

Likewise, [15] and [21] conclude that DCNNs trained on the ImageNet dataset tend to classify objects primarily based on textures. The studies also highlight that training DCNNs on modified datasets, such as "Stylized-ImageNet" (SIN), which replaces textures with painting styles, reduces this texture bias and increases shape bias. This shift seems to improve the model's generalization and robustness to image distortions [15], indicating that models with a shape bias are more effective than those based on textures for certain computer vision tasks. Similarly, [53] suggests that training DCNNs on more naturalistic and human-aligned datasets is a method to encourage a greater shape preference, reducing over-reliance on fine details and making the models more human-like, and has demonstrated that training deep neural networks on the Ecoset dataset, rather than the commonly used ImageNet data, results in representational spaces within DCNNs that are more aligned with those observed in the human higher-level visual cortex, and these networks better match human perceptual judgments.

Another approach proposed to address the "shortcut learning" in DCNNs is by incorporating more properties of the primate visual system. In [9], several fixed filters were integrated into the early layers of various DCNN architectures. These filters included the Gabor, Difference of Gaussians (DoG), Low-Pass, and a combination of Gabor and DoG filters. The results showed that models incorporating Gabor filters significantly outperformed those without, including end-to-end trained models (without preset filters) when evaluated on out-of-distribution validation data. This demonstrates that incorpo-

rating more properties of the visual cortex, in this case the bio-inspired Gabor filter, can effectively improve the generalization and robustness of models to new data. This improvement in generalization, came with a slight reduction in accuracy on the validation data for which it was trained, achieving a more realistic approximation of human performance in image classification for certain tasks, as shown by [68], where incorporating biological constraints led DCNNs to reduce their overcapacity in learning unstructured data, thereby making their performance more similar to that of humans.

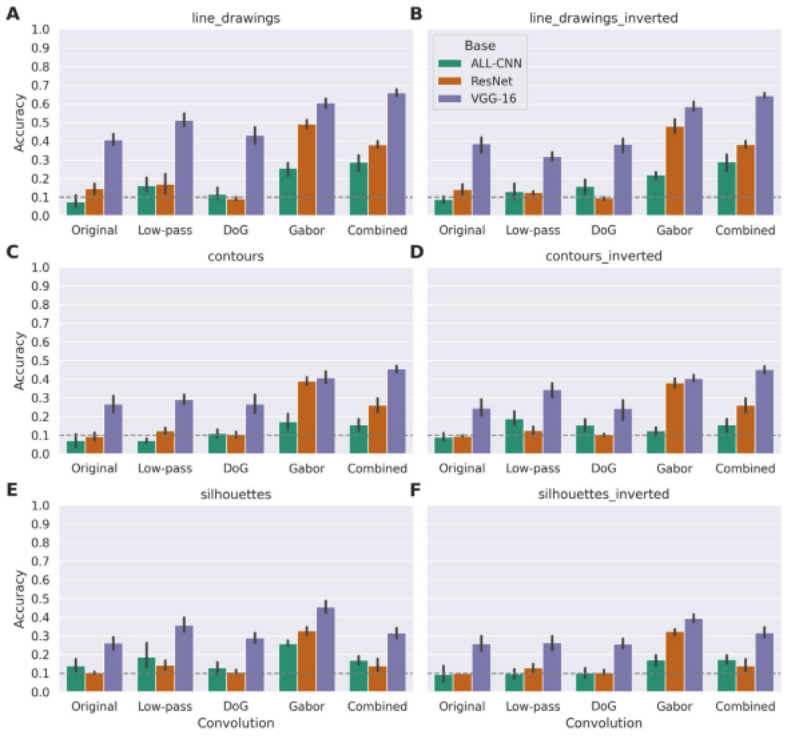

Figure 2: The results obtained by Evans, Malhotra, and Bowers [9] show the accuracy on CIFAR-10G of the ALL-CNN, ResNet, and VGG-16 architectures when incorporating different filters in their first convolutional layer, as well as without any filters. The graphs demonstrate superior performance when adding Gabor and Combined (Gabor with DoG) filters.

Despite the advancements achieved with Gabor filters, there remain opportunities to further optimize these architectures and address current limitations, such as enhancing the robustness to noise and generalization capacity on out-of-distribution data, as well as achieving representations more faithful to the visual cortex and predictive of human behavior judgments. In this research project, the model constructed in [9] is taken to be improved by adding the following visual biological properties: 1) A Retinal Information Bottleneck (RIB) [47], which represents the reduction of information observed in the retinal output through the optic nerve. 2) Recurrent Connections, which, as demonstrated in the literature [45] [30] [60], improve the performance of CNNs and resemble the intrinsic structure of the visual cortex where feed-forward, horizontal and feedback recurrent connections exist [39].

The RIB, as mentioned previously, is inspired by the reduction in the number of

neurons at the retinal output [70]. As [68] mentions, it is the reduction from photoreceptors to ganglion cells in the retina (see Figure 3).

A simplified model of the retina (retina-net) was created by [47], which had two layers. The first layer received the input and the second layer (a convolutional layer, which is the output) had a reduced number of neurons, replicating the bottleneck. This output served as the input to a CNN that, in itself, simulated the visual stream (CNN-VS).

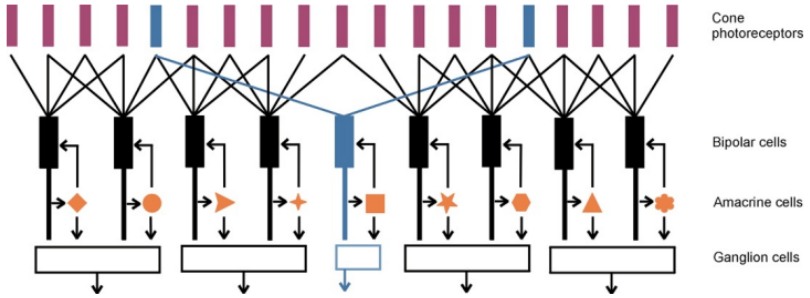

Figure 3: Reduction of neurons in a generic mammalian retina [52].

Training the entire model end-to-end, varying both the number of neurons in the retina-net output and the number of layers in the CNN-VS, resulted in the emergence of filters in the second layer of the CNN-VS that resembled the center-surround receptive fields (RF) of the retina, and oriented receptive fields in the third layer, similar to Gabor filters, which represent the V1 in the CNN-VS. This replicates the organization of biological visual representations.

In parallel with the RIB, studies have incorporated generic recurrent connections into CNNs [27] [4] [76], creating Convolutional Recurrent Neural Networks (CRNNs). Other studies have incorporated Long Short-Term Memory (LSTM) into CNNs [30] [60], showing greater effectiveness than standalone CNNs. Of particular interest to the neuroscience study is research like [35] [34] [33] [45], which are inspired by the ventral stream of the visual cortex. For example, in the research of [45] it is simulated the ventral stream using a multi-state fully recurrent neural network, where the CRNN contains layers representing V1, V2, V4, and IT, fully connected recurrently. This resulted in plausible performance similar to ResNet [20] but with fewer parameters. This study is important because it acknowledges the significant number of recurrent connections within the visual stream [10] [39], which have generally been overlooked in previous visual stream architectures [45].

Furthermore, [62] created CRNNs with bottom-up (B), lateral (L), and top-down (T) connections to understand the role of these recurrent connections for better biological modeling of the visual system. This study employed only two hidden layers, limiting their replication of the visual stream architecture. Nonetheless, they found significant findings such as increased robustness against additive Gaussian noise and occlusion, CRNNs outperformed feed-forward networks when recognizing three digits simultaneously, and when debris and noise were combined.

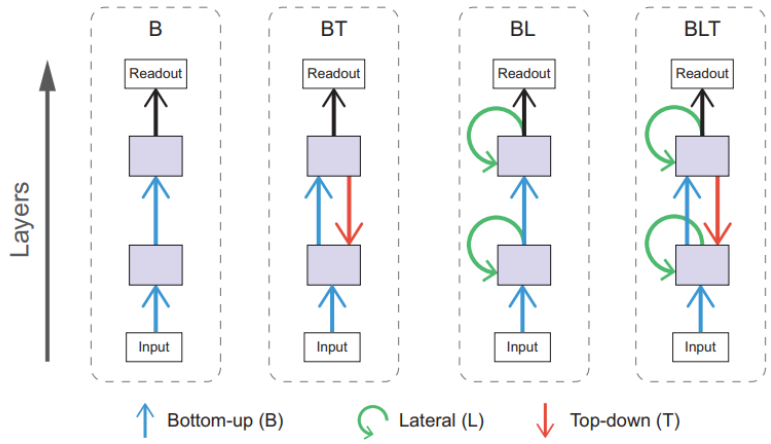

Figure 4: Recurrent neural network architectures used by Spoerer, McClure and Kriegeskorte [62].

Additionally, top-down connections proved beneficial when the task involved recognizing digits under heavier levels of debris. The study also showed that the best architecture always included B, L, and T connections simultaneously, closely resembling the visual system, which contains many feed-forward, horizontal, and feedback connections [39] [10].

These two features (RIB and CRNNs) are key aspects of the visual system that work together to enable humans to perceive the world visually, but they have been utilized only separately and any architecture has yet been designed that directly integrates them. Therefore, it is hypothesized that the combined incorporation of these biological features to the model of [9] will improve its generalization capacity to out-of-distribution data and robustness to noise, thereby bridging the gap between machine and human generalization capabilities.

The approach to achieve this, involves the training and evaluation of a baseline model based on the VGG-16 architecture [61] with fixed Gabor filters in their first convolutional layer, said architecture is presented in [9]. The baseline model will be then trained with the addition of the Retinal Information Bottleneck (RIB) in one version and with the addition of recurrent connections in another version. Finally, the baseline model is trained with both the RIB and recurrent connections incorporated. Training is conducted using the CIFAR-10 dataset.

After training, the models are evaluated on various test datasets to measure their generalization performance and accuracy against noise on out-of-distribution data. The test sets include the CIFAR-10 perturbed (with different types of noise added), CIFAR-10G, and the CIFAR-10G perturbed, which are detailed in Section 3.1. The performance metrics used for comparison include box plots and line charts that represent the accuracy of all the models across the different test sets and at all levels of added noise.

1.2 Research Objectives

1.2.1 General Objective

Optimize bio-inspired convolutional neural network architecture by integrating biological properties of the human visual cortex such as a Retinal Information Bottleneck and recurrent connections to improve generalization to out-of-distribution data and robustness to noise for image classification tasks.

1.2.2 Specific Objectives

1. Train the baseline model using the CIFAR-10 training dataset and evaluate its performance on the test sets based on CIFAR-10 and CIFAR-10G (see Section 3.1.1 and 3.1.2).
2. Integrate a Retinal Information Bottleneck into the baseline model, repeat the procedure from the first specific objective with the new model, and analyze its impact on improving the model's generalization capacity.
3. Add recurrent connections to the baseline model to replicate the recurrent connections of the human visual system, repeat the procedure from the first specific objective with the new model, and analyze its impact on improving the model's generalization capacity.
4. Implement both the Retinal Information Bottleneck and the recurrent connections in the baseline model, repeat the procedure from the first specific objective with the new model, and analyze its impact on improving the model's generalization capacity.

2 Theoretical Framework

2.1 Gabor Filters

Originally described by [12], this theory was mathematically extended to 2D in the work of [6] and [7], modeling simple cell receptive fields with a set of functions represented as spatial filters composed of sinusoidal plane waves within two-dimensional elliptical Gaussian envelopes [69]. It has been argued that the 2D Gabor function accurately models the receptive field structure of simple cells in the visual cortex of cats [28] and primates such as monkeys [55], playing a crucial role for detecting edges and contours in different orientations. Applying them to models allows neural networks to more effectively learn and capture the shapes within images and in a more human-like manner.

A 2D Gabor filter is defined as a sinusoidal function modulated by a Gaussian function [7]. A general equation is:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)} \cos\left(2\pi\frac{x'}{\lambda} + \psi\right) \tag{1}$$

where:

$$x' = x\cos\theta + y\sin\theta \tag{2}$$

$$y' = -x\sin\theta + y\cos\theta \tag{3}$$

Here:

- λ is the wavelength of the sinusoidal component.
- θ is the orientation of the filter.
- ψ is the phase of the sinusoidal component.
- σ is the standard deviation of the Gaussian.
- γ is the aspect ratio, which controls the "narrowness" of the Gaussian in the direction perpendicular to the sinusoidal wave.
- (x, y) the position of a light impulse in the visual field.

Additionally, the Gabor transform is the dot product of a Gaussian distribution with a complex exponential, the complex plane wave known as the receptive field potential [31].

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = e^{-\frac{x^2+y^2}{\sigma_x^2+\sigma_y^2}} \cdot e^{j2\pi\frac{x}{\lambda}} \tag{4}$$

$$g_{\lambda,\theta,\phi,\sigma,\gamma}(x, y) = e^{\left(-\frac{x_\theta^2+\gamma^2 y_\theta^2}{2\sigma^2}\right)} e^{\left(i\left(2\pi\frac{x_\theta}{\lambda}+\phi\right)\right)} \tag{5}$$

Where the variables are equivalent to those presented in equation (1), and x' and y' are defined in equations (2) and (3) respectively.

As [9] mentioned, instead of defining the wavelength of the sinusoidal component (λ) in pixels, it is more intuitive to specify the bandwidth (b).

$$\lambda = \sigma \cdot \pi \cdot \sqrt{\frac{2}{\ln 2} \cdot \frac{2^b - 1}{2^b + 1}} \tag{6}$$

The Gaussian envelope standard deviation (σ) and another components can be fixed. Consequently, the wavelength (λ) of the sinusoidal factor could be determined indirectly through the bandwidth (b) and the standard deviation (σ) [9].

A great representation of a Gabor filter and its composition is given by the Figure 5.

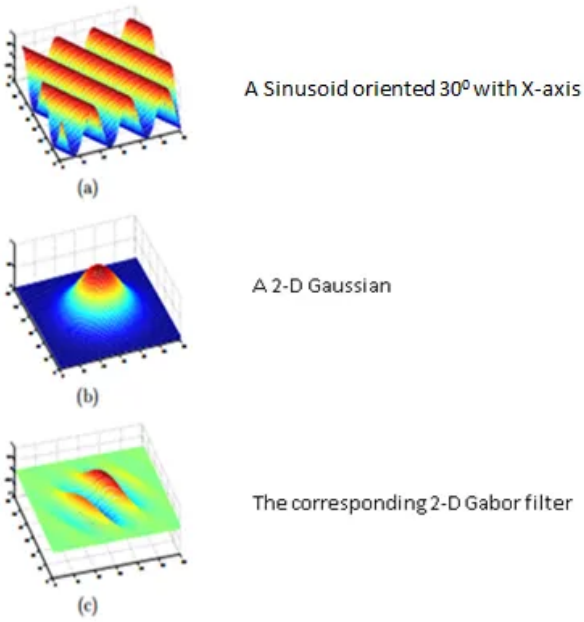

Figure 5: This image illustrates the construction of a 2-D Gabor filter. The top figure (a) shows a sinusoidal wave oriented at 30 degrees with respect to the X-axis. This sinusoid forms the core of the Gabor filter and is responsible for detecting specific frequency and orientation features in an image. The middle figure (b) represents a 2-D Gaussian distribution. This Gaussian envelope localizes the sinusoidal wave, providing spatial confinement and ensuring that the filter responds primarily to features in a specific region of the image. The bottom figure (c) depicts the resulting 2-D Gabor filter, which is a product of the sinusoid (a) and the Gaussian (b). Source of image: Obtained from Medium.com.

As can be observed in Figure 6, the effects of theta (θ), gamma (γ), lambda (λ), and sigma (σ) on Gabor filters are visualized. The Theta parameter (θ) controls the orientation of the Gabor filter, allowing it to detect edges at different angles. As θ changes, the filter rotates, resulting in vertically, diagonally, and horizontally oriented filters. The Gamma parameter (γ) adjusts the spatial aspect ratio, affecting the filter's sensitivity to different spatial frequencies and orientations. This can be thought of as changing the "height" of the filter, making it elongated vertically, balanced, or less elongated. The Lambda parameter (λ) controls the wavelength of the sinusoidal factor, which can

be considered the "width" of the filter. Adjusting λ changes the frequency response of the filter. The Sigma parameter (σ) determines the standard deviation of the Gaussian envelope, controlling the filter's spatial extent and localization. As σ increases, the filter becomes wider, resulting in more stripes within the Gabor function.

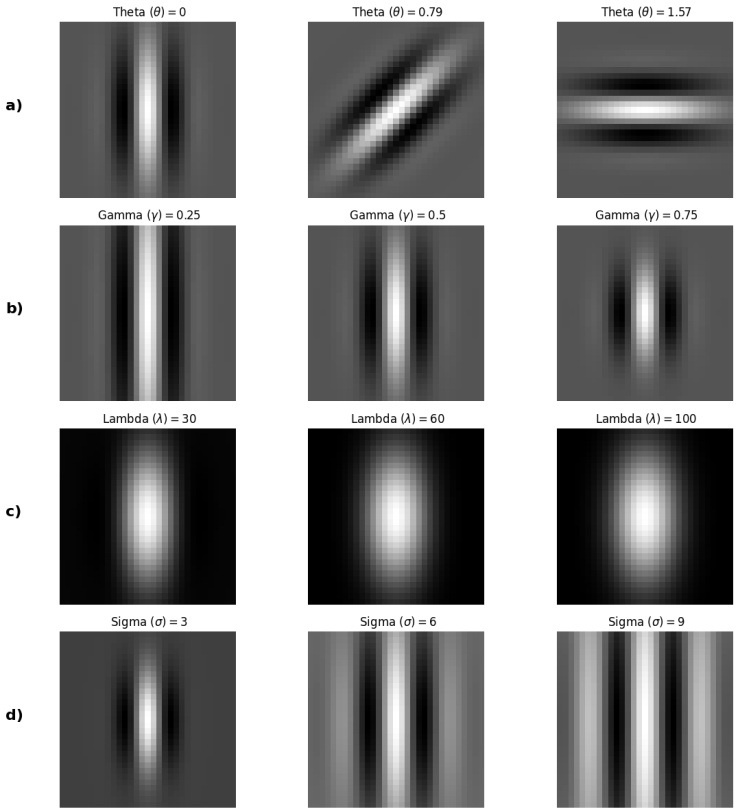

Figure 6: Illustration of the effects of varying parameters on 2D Gabor filters. a) shows Gabor filters with different Theta (θ) values: 0, 0.79, and 1.57 radians, while the other parameters remain unchanged ($\gamma = 0.5$, $\lambda = 10$, $\sigma = 0.4$). b) displays Gabor filters with varying Gamma (γ) values: 0.25, 0.5, and 0.75 ($\theta = 0$, $\lambda = 10$, $\sigma = 0.4$). c) presents Gabor filters with different Lambda (λ) values: 30, 60, and 100 ($\theta = 0$, $\gamma = 0.5$, $\sigma = 0.4$). d) shows Gabor filters with varying Sigma (σ) values: 3, 6, and 9 ($\theta = 0$, $\lambda = 10$, $\gamma = 0.6$).

The Figure 7 depicts the application of a bank of Gabor filters to an input image of a circle. The process can be broken down into three main parts: The top-left part of the image shows the input image, which is a simple circle on a black background. Below the input image is a bank of 16 Gabor filters. Each filter has a different orientation that allows them to be particularly sensitive to specific orientations. The right side of the image displays the outputs after the input circle image is passed through each individual Gabor filter from the bank.

Each output image shows how the circle appears when filtered by one of the Gabor filters. It can be observed that the different orientations of the Gabor filters filter more effectively the parts of the circle that are aligned with the same orientation.

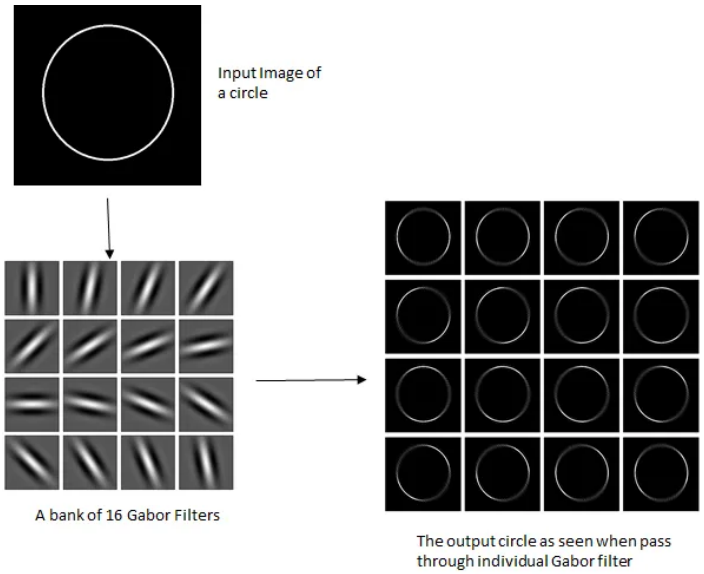

Figure 7: Application of Gabor filters on an input image. Source of image: Obtained from Medium.com.

2.2 Convolutional Neural Networks (CNNs)

The CNNs, as quoted by [18], "are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.". They trace their origins to an earlier architecture known as the Neocognitron [11]. However, the Neocognitron was unable to be trained using algorithms like backpropagation [42]. The breakthrough came when Yann LeCun and his team developed the first successful convolutional neural network for recognizing handwritten zip codes in the U.S. [43]. This was followed by the LeNet-5 architecture, designed for handwritten digit recognition and utilized for character recognition [44], both of which were successfully trainable for these tasks.

The concept of convolution in neural networks, however, has its roots in neuroscience. The inspiration for CNNs comes from the research of Hubel and Wiesel in the 1960s on the visual cortex of cats, where they discovered cells sensitive to specific features such as edges and orientation [25].

The true rise of CNNs began in 2012 when Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton won the ImageNet competition with their model AlexNet, which significantly outperformed previous results in image classification tasks [37]. This milestone marked the beginning of a new era for CNNs and their application in a wide variety of computer vision tasks.

Since then, CNNs have evolved significantly, with more advanced architectures like VGGNet [61], GoogLeNet [66], and ResNet [20], continuously improving performance in multiple image processing tasks.

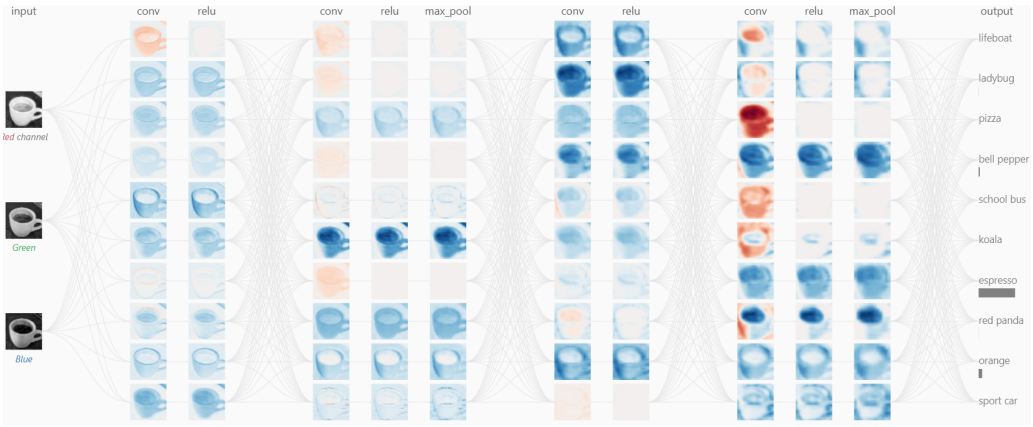

Figure 8: Typical architecture of a CNN. Source of image: Obtained from CNN explainer [71].

Figure 8 illustrates a typical architecture of (CNN). This architecture consists of the input image, in this case labeled as "espresso," which contains three channels: Red, Green, and Blue (RGB). Each of these channels is convolved by kernels (also known as filters) found in the convolutional layers (conv). This convolution operation extracts specific features from the input image and then passes through an activation function, commonly ReLU, represented as a hidden layer (relu).

In the architecture shown in Figure 8, this process repeats until reaching the Pooling layer (max pool). It is common for CNN architectures to follow this pattern: convolutional layer, activation layer, and pooling layer, until reaching the Flatten layer. The Flatten layer converts the 2D channels into a single array. Subsequently, a linear transformation (a weighted sum) and normalization of each value in this array are performed. This is followed by an activation function, such as softmax (for multiclass classification) or sigmoid (for binary classification), which converts each output into a probability. The classification is then based on the output with the highest probability.

CNN Layers

- **Convolutional Layer**:

  In this layer, each channel of the input is convolved by the kernel of each neuron.

  $$Z = X * W + b \tag{7}$$

  where X is the input, W is the convolutional kernel and b is the bias. The operation represented by "*" refers to a convolution.

- **Pooling layers**:

  There are different types of Pooling Layers, but they all have the purpose to reduce the spatial dimensions of the network. This reduction minimizes the number of parameters and the overall computational load of the network [71].

  The Max-Pooling operation, which is the most frequently used, has been employed in this research. The operation slides the kernel over the input based on the defined

stride, and for each position, it selects the maximum value within the kernel's window from the input to produce the output [71].

$$Z = \max(X) \tag{8}$$

Convolution

In the particular case of 2D, the convolution is defined as follows:

$$(f * g)(i, j) = \sum_m \sum_n f(m, n) \cdot g(i - m, j - n) \tag{9}$$

Which in turn, since convolution is commutative, can be equivalently written as:

$$(f * g)(i, j) = \sum_m \sum_n f(i - m, j - n) \cdot g(m, n) \tag{10}$$

In these equations:

- $(f * g)(i, j)$ represents the result of the convolutional operation at position (i, j).
- $f(m, n)$ represents the pixel value at position (m, n) of the input image f.
- $g(i - m, j - n)$ represents the value at position $(i - m, j - n)$ of the convolution kernel g.
- m and n are the indices used to iterate over the dimensions of the kernel.

The convolution operation involves flipping the kernel g and sliding it over the input image f and computing the sum of the element-wise products of the kernel values and the corresponding input image values. This operation is repeated for each position (i, j) to produce the output feature map.

The only reason to flip the kernel is to achieve the commutative property, which is important for theoretical proofs, but this property is not typically a crucial property of a neural network implementation [18]. That is the reason of many neural network libraries to implement a related operation known as cross-correlation. Cross-correlation is essentially identical to convolution, except it does not involve flipping the kernel, and is mathematically represented by:

$$(f * g)(i, j) = \sum_m \sum_n f(i + m, j + n) \cdot g(m, n) \tag{11}$$

It is important to note that the term "convolution" is often used interchangeably with "cross-correlation" in many explanations and documentation. As Bengio, Goodfellow and Courville [18] point out, "many machine learning libraries implement cross-correlation but refer to it as convolution.".

Activation Functions

The activation functions used in CNNs include:

$$\text{ReLU}(x) = \max(0, x) \tag{12}$$

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{13}$$

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{14}$$

As mentioned earlier, the Softmax function is used for multiclass classification whereas the sigmoid function is used for binary classification. On the other hand, the ReLU function is used to introduce non-linearity into the network, which helps in learning complex patterns from the data during training.

Hyper-parameters

- **Padding**: Padding is a technique used to add extra pixels to the input image, usually with zeros, around the border. This is done to control the spatial size of the output feature map. Padding helps preserve the spatial dimensions of the input after applying convolution, allowing the use of more convolutions without reducing the size of the feature maps. There are two types of padding: a) Valid Padding; when no padding is added, and the feature map size reduces after each convolution. b) Same Padding: Padding is added such that the output feature map has the same spatial dimensions as the input.
- **Kernel Size**: The kernel size (also known as filter size) refers to the dimensions of the filter used during convolution. It determines the receptive field of the convolution operation, i.e., the area of the input image that influences a single output value. Common kernel sizes are 3×3, 5×5, and 7×7. Smaller kernels capture fine details, while larger kernels capture more global features.
- **Stride**: Stride refers to the number of pixels by which the filter moves across the input image. A stride of 1 means the filter moves one pixel at a time, while a stride of 2 means it moves two pixels at a time. Increasing the stride reduces the spatial dimensions of the output feature map and results in a more significant reduction in computational complexity. However, too large a stride may lead to a loss of important information.

2.3 Retinal Information Bottleneck (RIB)

The concept of the Retinal Information Bottleneck (RIB) is inspired by the way visual information is processed and reduced within the biological visual system. In mammals, the retina contains a large number of light-sensitive neurons, known as photo-receptors (cones and rods). These photo-receptors convert light into electrical signals, which are then transmitted through several layers of neural cells in the retina before traveling to the brain via the optic nerve [52].

A remarkable feature of this system is a reduction from the photo-receptors to ganglion cells, which leads to a significant compression of visual information passing from the retina to the optic nerve and, ultimately, to the visual cortex. This compression is crucial because the optic nerve has a limited capacity to transmit the vast amount of information captured by the photo-receptors. Therefore, this characteristic can be studied through the "Information Bottleneck Method" [67].

An interesting study by [48] replicates this characteristic and demonstrates that the bottleneck in CNNs results in a diversity of geometrical representations that do not emerge

when the bottleneck is omitted. The study shows that as long as a bottleneck exists, center-surround receptive fields form in the convolutional layers corresponding to the retina, while maintaining oriented receptive fields after the bottleneck, corresponding to the layers analogous to the primary visual cortex (V1).

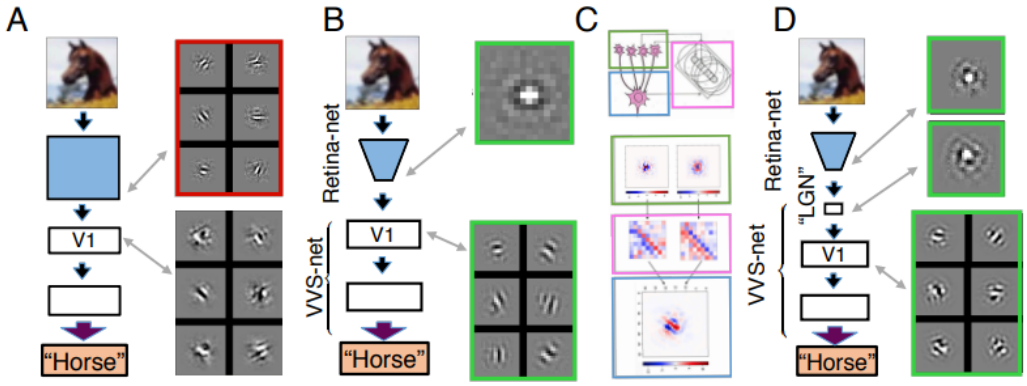

Figure 9: Results obtained by [48].

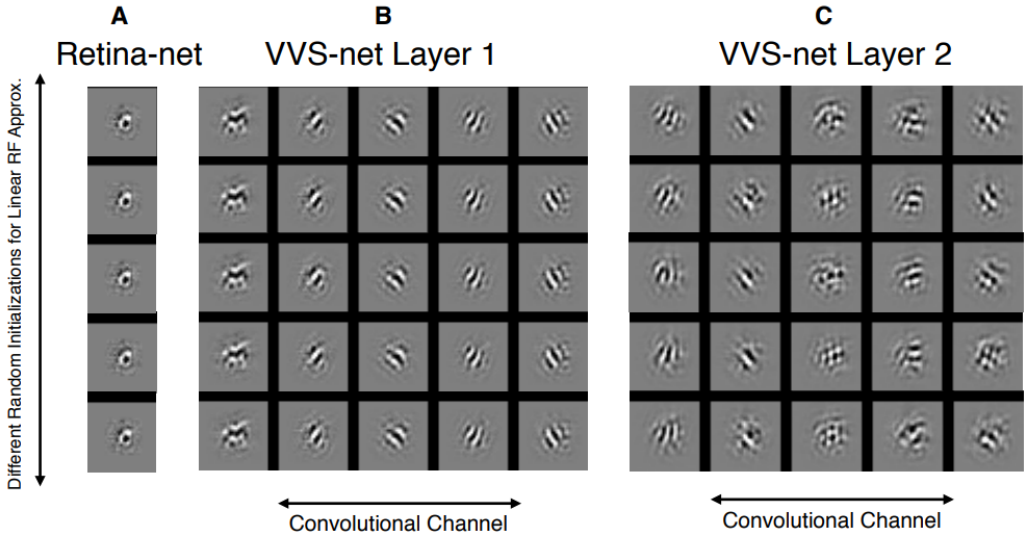

Figure 10: Internal representations obtained in [48]. Figure 6a shows the geometrical representation of the neurons in the bottleneck, while Figure 6b and 6c the geometrical representation of the neurons in subsequent convolutional layers of the architecture.

As illustrated in Figure 9b and 9c, when the bottleneck is introduced (as implemented in the "retina-net" architecture), the geometric representation of the filters displays center-surround characteristics, while still maintaining oriented receptive fields in V1 and subsequent layers. In contrast, when the bottleneck is not included, as seen in Figure 9a, these internal characteristics do not emerge.

Incorporating this bottleneck into the base model of this study, with the aim of endowing it with more characteristics of the visual cortex, is expected to enhance its performance to more closely resemble that of humans. Consequently, its generalization ability is anticipated to improve in conjunction with the recurrent connections detailed in Section 2.4.

2.4 Recurrent Neural Networks (RNNs)

RNNs can be particularly useful in "sequence modeling," which involves the ability of a neural network to utilize information from past events to make informed predictions about future actions. This could mean that recurrent connections allow the visual system to integrate temporal information, potentially enhancing perception and decision-making based on the context of previous events. As [19] quotes, "the key point is that the recurrent connections allow a 'memory' of previous inputs to persist in the network's internal state, and thereby influence the network output.".

A representation of a simple RNN is depicted by Figure 11:

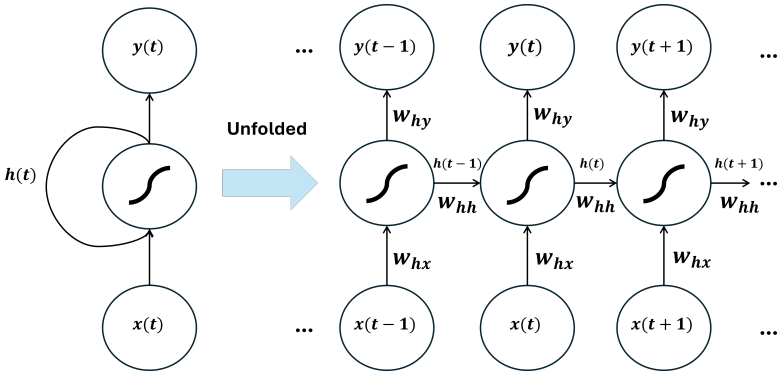

Figure 11: Simple recurrent artificial neuron. A neuron in this context can be viewed as a feed-forward neural network with a temporal dimension. This perspective is referred to as the "unfolded" version of the network.

Based on the diagram presented in Figure 11, the output $y(t)$ is a function that depends not only on the current input $x(t)$ but also on the previous hidden state $h(t-1)$:

$$y(t) = f(x(t), h(t-1)) \tag{15}$$

and the hidden state and output vector at time t are described by:

$$h(t) = \sigma(w_{hh} \cdot h(t-1) + w_{xh} \cdot x(t)) \tag{16}$$

$$y(t) = w_{hy} \cdot h(t) \tag{17}$$

where w_{hy}, w_{hh} and w_{hx} represent the weight matrices.

Long short-term memory (LSTM)

In our case study, we have utilized Recurrent Neural Networks (RNNs) known as Long Short-Term Memory (LSTM) [22]. Traditional RNNs face significant challenges during training, such as the vanishing gradient problem (where the error diminishes to the point of disappearing) and the exploding gradient problem (where the error increases uncontrollably) when using conventional training methods, such as those proposed in [58] and [72]. These issues made it difficult to effectively train traditional RNNs.

In 1997, an innovative Recurrent Neural Network architecture known as LSTM was introduced, specifically designed to overcome the vanishing and exploding gradient problems. LSTMs have proven to be highly efficient in learning temporal data sequences, allowing the network to capture long-term dependencies in incoming data and overcoming the limitations of conventional RNNs [54].

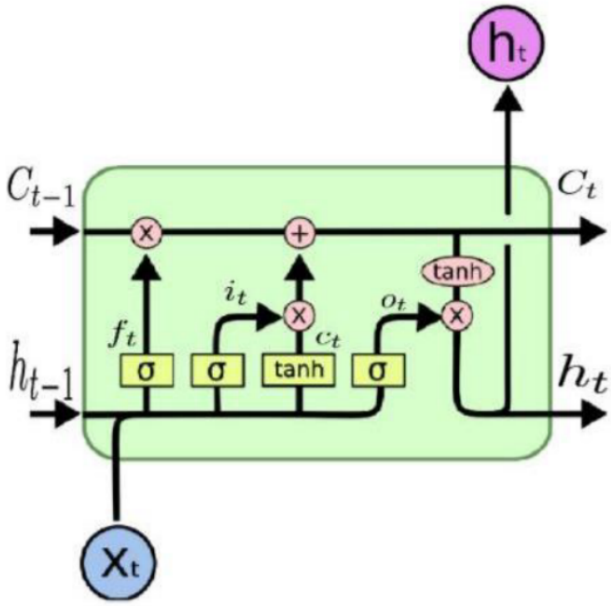

Figure 12: Standard LSTM architecture. This Figure have been obtained from [29].

Where c_{t-1} and c_t are known as memory cells or the cell state, and h_{t-1} and h_t are the hidden outputs or "hidden states" of the network. The memory value is transmitted from one cell to the next over time.

The LSTM has three main gates that control the flow of information: the forget gate (f_t) controls how much of the previous memory c_{t-1} should be forgotten, and it takes h_{t-1} and x_t (the input at time t) as inputs. The second gate is the input gate (i_t), which controls how much of the new information $\tilde{c}_t$ (generated from a tanh function) should be stored in the cell's memory. Finally, the output gate (o_t) controls how much of the memory c_t should be used to generate the cell's output at h_t.

These relationships are described by the following equations:

$$f_t = \sigma(W_{f_t} x_t + U_{f_t} h_{t-1} + b_{f_t}) \tag{18}$$

$$i_t = \sigma(W_{i_t} x_t + U_{i_t} h_{t-1} + b_{i_t}) \tag{19}$$

$$o_t = \sigma(W_{o_t} x_t + U_{o_t} h_{t-1} + b_{o_t}) \tag{20}$$

$$\tilde{c}_t = \tanh(W_{c_t} x_t + U_{c_t} h_{t-1} + b_{c_t}) \tag{21}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{22}$$

$$h_t = o_t * \tanh(c_t) \tag{23}$$

where $W_{f_t}, W_{i_t}, W_{o_t}, W_{c_t}$ are the weight matrices applied to the input x_t for the forget, input, output gates, and cell state update, respectively. $U_{f_t}, U_{i_t}, U_{o_t}, U_{c_t}$ are the weight matrices applied to the hidden state h_{t-1} for the forget, input, output gates, and cell state update, respectively. $b_{f_t}, b_{i_t}, b_{o_t}, b_{c_t}$ are the bias terms for each gate and cell state.

RNNs in the Visual Cortex

In the visual cortex, different layers have distinct patterns of connectivity. For example, V1 sends outputs to V2 and receives feedback from higher-order areas like V4 and IT. It is possible to mimic this by adding feedback connections from higher layers back to lower ones and ensuring certain layers are connected to specific subsequent layers, in the form of the visual stream, according to [10].

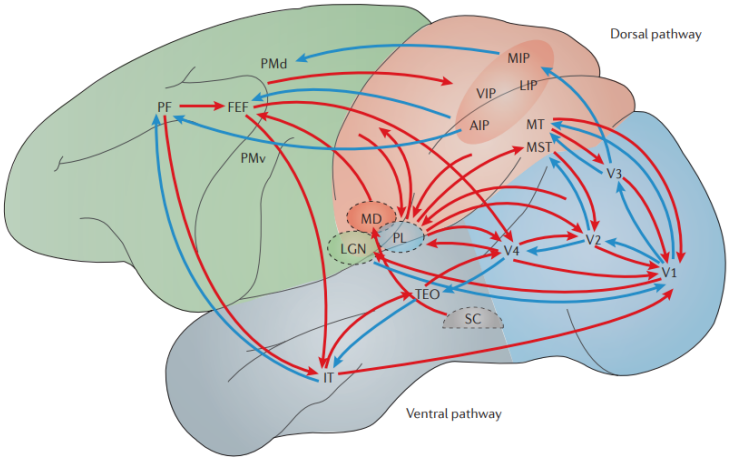

Figure 13: Connectivity in the ventral stream and dorsal stream. This Figure have been obtained from [16], which in turn, was derived from [17].

As depicted in Figure 11, the ventral stream pathway is structured as follows: starting from the LGN, progressing to V1, then to V2, followed by V4, moving to TEO, and finally reaching IT. After traversing the ventral stream pathway and reaching IT, the connections extend to the PF and FEF areas. Where:

1. V1 is the Primary Visual Cortex.
2. V2 is the Secondary Visual Cortex.
3. V4.
4. TEO is the Posterior Inferotemporal Cortex.
5. IT is the Inferotemporal Cortex.
6. PF is the Prefrontal Cortex.
7. FEF is the Frontal Eye Field.

It is also noticeable that the visual system consists not just of many richly feedback interconnected areas but also reflects lateral interactions [39]. Additionally, a significant feature of the visual cortex is that approximately 40% of its connections are recurrent, with a large proportion being reciprocal, indicating extensive bidirectional connectivity between cortical areas [10]. This paves the way for modeling the visual cortex within the framework of computational neuroscience. For example, a recurrent architecture exclusively imitating the visual stream is shown in [45].

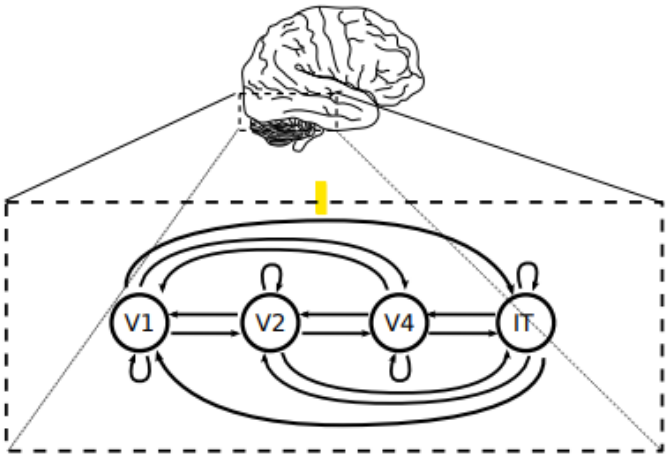

Figure 14: Visual stream recurrent neural network architecture worked by [45].

3 Methodology

A baseline model based on the VGG-16 architecture with biological constraints in the form of fixed Gabor filters in its first convolutional layer, presented in [9] is used in this research project. This first convolutional layer has 24 channels, representing 24 Gabor filters, each one with an unique combination of the parameters depicted in Table 2. Initially, the baseline model is trained and evaluated using the training and test sets described in Section 3.1. This establishes its baseline performances, and serves as a benchmark for comparison with the new models described in Sections 3.3.1, 3.3.2 and 3.3.3.

Then, three new models are developed: in the first model, a Retinal Information Bottleneck in the baseline model. In the second model, recurrent connections are integrated to the baseline model. Lastly, the third model is the implementation of both the Retinal Information Bottleneck and recurrent connections into the baseline model. All these models are trained and evaluated in the same manner and using the same training and test sets as the baseline models. The hyper-parameters of the models are shown in Table 1.

Afterwards, a comparison of the models is carried out, where the relevant metrics are the accuracy and the loss curves of the models in each test set. This is to measure the impact of the new implementations in the capacity of the baseline models to generalise in out of distribution data.

Table 1: Models Hyper-parameters.

Hyper-parameter	**Value**
Loss Function	Categorical Cross Entropy
Optimizer	RMSprop
Batch Size	32
Learning Rate	10^{-4}
Training Epochs	100
Decay	10^{-6}

Table 2: Gabor Parameters.

		Values
σ	=	$\{8\}$
γ	=	$\{0.5\}$
b	=	$\{1, 1.8, 2.6\}$
θ	=	$\{0, \frac{\pi}{4}, \frac{2\pi}{4}, \frac{3\pi}{4}\}$
ψ	=	$\{\frac{\pi}{2}, \frac{3\pi}{2}\}$

The output layer was reduced to classify each image into one of the ten CIFAR-10 categories using a softmax activation function. Each model was trained with a random seed initialization of 42. Additionally, the learning rate was reduced by a factor of 0.2 after five epochs if no improvement was observed.

The code is written in Python 3, utilizing the TensorFlow and Keras frameworks. The training and testing of the models were conducted using an A100 GPU in the Google Colab environment.

3.1 Training Data

In this research project, two datasets are utilized: CIFAR-10 and CIFAR-10G. All models are initially trained on the CIFAR-10 training set. To evaluate the performance of these models, several testing procedures are followed.

Firstly, noise is added to the CIFAR-10 test set to create a series of perturbed datasets. Specifically, ten different types of noise are applied to the test images, each with varying levels of severity. This results in multiple versions of the CIFAR-10 test set, each one representing a different combination of noise type and severity level. The characteristics and types of noise used for these perturbations are detailed in Table 3. The models are evaluated on each of these perturbed test sets, resulting in 10 different evaluations corresponding to each noise type, with further evaluations for each severity level of the noise.

Following this, the models are tested on the CIFAR-10G test set, as explained in Section 3.1.2. CIFAR-10G is a stylized version of CIFAR-10, focusing more on shape information with reduced texture details.

Finally, similar to the procedure with CIFAR-10, the CIFAR-10G test set is also perturbed with the same ten types of noise, each with multiple severity levels. This again results in multiple versions of the CIFAR-10G test set, with models being evaluated on each one. In total, for both CIFAR-10 and CIFAR-10G, the models undergo extensive testing across a range of conditions: initially on the unperturbed data, and then on data perturbed by different noise types and severity levels. This rigorous evaluation process ensures a thorough assessment of the models' robustness and generalization capabilities across varying levels of data corruption.

All the preprocessing detailed and explained in the next subsections is that carried out and explicitly used in [9], aiming to maintain consistency, equality and achieve fair comparisons.

Table 3: Image perturbation descriptions and severity.

Perturbation	**Description**	**Levels**
Uniform	Pixel-wise additive uniform noise drawn from $[-w, +w]$ then clipped at $[0, 1]$.	$w \in \{0, 0.1, \ldots, 0.9, 1.0\}$
Salt and pepper	Pixels are randomly set to either black or white with probability, p.	$p \in \{0, 0.1, \ldots, 0.9, 1.0\}$
High-pass	High-pass filtering with standard deviation of the Gaussian filter, σ.	$\sigma \in \{10^{0.2}, 10^{0.4}, \ldots, 10^{1.8}, 10^{2}\}$
Low-pass	Low-pass filtering with standard deviation of the Gaussian filter, σ.	$\sigma \in \{10^{0.1}, 10^{0.2}, \ldots, 10^{1.8}, 10^{2}\}$
Contrast	Contrast, c adjusted by setting each pixel intensity, i, according to $i' = (1-c)/2 + i \cdot c$.	$c \in \{1, 0.9, \ldots, 0.1, 0\}$
Phase scrambling	Phases are randomly shifted (in the Fourier domain) in the interval $[-w, +w]$ degrees.	$w \in \{0, 18, \ldots, 162, 180\}$
Darken	Each pixel intensity, i, is reduced by l.	$l \in \{0, 0.1, \ldots, 0.9, 1\}$
Brighten	Each pixel intensity, i, is increased by l.	$l \in \{0, 0.1, \ldots, 0.9, 1\}$
Rotation	Each image is rotated by θ degrees.	$\theta \in \{0, 90, 180, 270\}$
Inversion	Pixel intensities are inverted.	$v \in \{0, 1\}$

3.1.1 CIFAR-10

Description

CIFAR-10 dataset consists of 60000 colour images in 10 classes, wirh 6000 images per class of size 32x32 pixels. There are 50000 training images and 10000 test images, created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton at the University of Toronto (see www.cs.toronto.edu/ kriz/cifar.html).

Preprocessing

The preprocessing closely follows the methodology outlined by [9]. As described in that paper, all images were first converted to grayscale using the ITU BT.601 luma transform conversion formula ($Y = 0.299 \cdot R + 0.587 \cdot G \cdot 0.114 \cdot B$). The images were then upscaled from their original dimensions of 32×32 pixels to 224×224 pixels using Lanczos resampling, with luminosity values constrained to the range [0, 255]. Following this, the intensity of each image was rescaled from [0, 255] to [0, 1].

For the testing phase, noise was applied to the images after rescaling, and the values were subsequently adjusted back to the [0, 1] range before being rescaled to [0, 255].

The CIFAR-10 validation images were modified with various types of noise, including Uniform, Salt and Pepper, High-pass, Low-pass, Contrast, Phase Scrambling, Darkening, Brightening, Rotation, and Inversion. These noise types were applied at varying levels of severity, as detailed in Table 3 (taken from [9]).

The mean and standard deviation were computed across the entire modified training dataset to perform feature-wise normalization and centering. Additionally, data augmentation techniques were employed, which included randomly shifting the images both vertically and horizontally by up to 10% (24 pixels) and applying random horizontal flips. This preprocessing approach mirrors the methodology described in [9].

3.1.2 CIFAR-10G

Description

To evaluate the networks' capability to classify images beyond the training set, [8] developed a dataset called CIFAR-10G, which includes stylized images for each of the ten CIFAR-10 categories. These images are out-of-distribution (o.o.d.) and are designed to contain shape information with minimal to no texture details. As the author mentions, this allows for an assessment of the model's ability to classify without depending on high-frequency spatial information. The dataset containts three independent generalization test sets: line drawings, silhouettes, and contours. Each set contained ten images for each of the ten CIFAR-10 categories. Additionally, three more sets were produced by inverting the original three sets.

Preprocessing

To thoroughly evaluate the models, they were not only tested on the unperturbed version of the test set, but also on multiple versions of the test set that were modified with various types of noise. Specifically, each model was subjected to the different noise types and severity levels outlined in Table 3 (the same procedure as with CIFAR-10). This approach allowed for the assessment of the models' performance across a wide range of conditions, with each type of noise and its corresponding severity level creating a distinct version of the test set. Consequently, the models were evaluated on these multiple variations of the test set to ensure a comprehensive analysis of their robustness and generalization capabilities under different levels of data corruption.

3.2 Baseline Model

VGG-16 model architecture

VGG-16 is a deep convolutional neural network developed by the Visual Graphics Group at the University of Oxford, introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" by K. Simonyan and A. Zisserman in 2014 [61]. This architecture was designed to address large-scale image recognition tasks and is noted for its simplicity and effectiveness.

VGG-16 has a total of 16 layers containing trainable weights, of which 13 are convolutional layers and 3 are fully connected layers. Additionally, it has 4 max pooling layers and. The layers of the VGG-16 architecture are arranged in 6 blocks. The first and second blocks contain two convolutional layers with 3x3 kernels and a stride of 1, followed by a max pooling layer with a 2x2 kernel and a stride of 2. The third, fourth, and fifth blocks each have three convolutional layers with 3x3 kernels and a stride of 1, followed by a max pooling layer with a 2x2 kernel and a stride of 2. Finally, the sixth block contains three fully connected to flatten and linearly transform the data, and a fourth layer that uses a softmax function to categorize the outputs. A ReLU activation function is used after each convolutional layer.

Figure 15 and 16 show the original architecture of VGG-16, and Table 4 describes the characteristics of each layer.

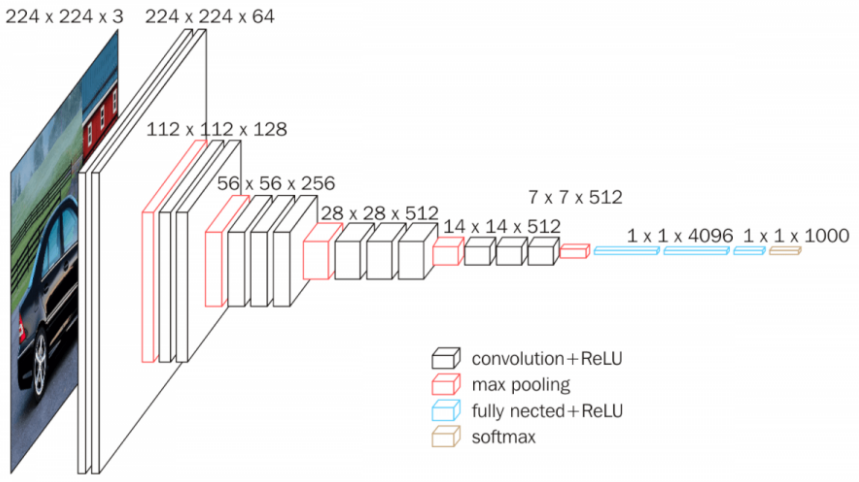

Figure 15: VGG-16 architecture. The numbers represent the dimensions of the output at each layer in the format *height* × *width* × *channels*. Note that, the sequence of dimensions throughout the model assumes a hypothetical (common) input of 224 × 224 × 3, which may not necessarily correspond to the input used in this methodology. Source of image: Obtained from Kaggle.

Figure 16: VGG-16 architecture by blocks. Each block is represented by color. Each layer has a description in the form of *kernel size, layer type, channels*. The last block contains a kernel size of 1. "fc" means fully connected and "conv" means convolution. Source of image: Obtained from Kaggle.

Baseline model architecture

The base architecture is based on the VGG-16 model but has a modified first convolutional layer consisting of 24 channels, each with a kernel size of 63x63 using Gabor filters. The parameters for these Gabor filters are provided in Table 4. This base architecture is illustrated in Figure 17.

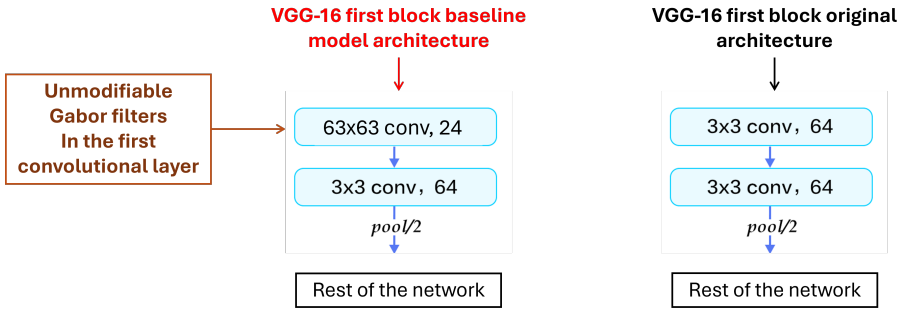

Figure 17: Difference between the baseline model and the VGG-16 architecture. As can be observed, the first convolutional layer is modified. The kernel size is 63x63 and the number of channels 24 in the baseline model, whereas the original VGG-16 architecture consist of a kernel size of 3x3 and 64 channels in its first convolutional layer.

Table 4: Baseline model layers characteristics.

Layer type	Channels	Kernel Size	Stride	Activation
Input (Image)	-	-	-	-
Convolution	24	63x63	1	relu
Convolution	64	3x3	1	relu
Max Pooling	64	3x3	2	relu
Convolution	128	3x3	1	relu
Convolution	128	3x3	1	relu
Max Pooling	128	3x3	2	relu
Convolution	256	3x3	1	relu
Convolution	256	3x3	1	relu
Max Pooling	256	3x3	2	relu
Convolution	512	3x3	1	relu
Convolution	512	3x3	1	relu
Convolution	512	3x3	2	relu
Max Pooling	512	3x3	1	relu
Convolution	512	3x3	1	relu
Convolution	512	3x3	2	relu
Convolution	512	3x3	1	relu
Max Pooling	512	3x3	2	relu
Fully Connected	-	1	-	relu
Fully Connected	-	1	-	relu
Fully Connected	-	1	-	relu
Fully Connected	-	-	-	Softmax

3.3 Enhanced models

3.3.1 Retinal Information Bottleneck Model

As mentioned in [48], which cites to [52], the retinal architecture is strongly conserved across species, and consists of three layers of feed-forward convolutional neurons (photoreceptors, bipolar cells, ganglion cells) and two layers of inhibitory interneurons (horizontal and amacrine cells). They modeled the retina with a two-layer CNN, where the first layer contains 32 channels and receives the input information, and the second layer has a reduced number of channels and serves as the output of the CNN. This architecture was referred to as the retina-net. Additionally, in some experiments, a third layer representing the Lateral Geniculate Nucleus (LGN) was added, containing the same number of channels as the second layer. This study demonstrated that incorporating the retina-net prior to an architecture resembling the visual stream produced center-surround internal representations akin to those found in the receptive fields (RFs) of the LGN [23]. Additionally, it contributed to the formation of oriented RFs in V1.

The approach presented here is based on and inspired by the work of [48]. We recreated a similar architecture, using a three-layer structure, with the third layer representing the LGN. The output of this three-layer architecture serves as the input to the baseline model.

Figure 18 shows the schematic architecture of the RIB perfomed by [48], where N represents a variable number of channels of the retina-net layers, while in Figure 19 is

depicted the baseline model with the retina-net architecture incorporated.

The bottleneck layers used in this study have a kernel size of 3x3 with stride of 1 and padding as "same" in Tensorflow and Keras frameworks. The first layer contains 32 channels, while the second and third layers contain two channels (N = 2).

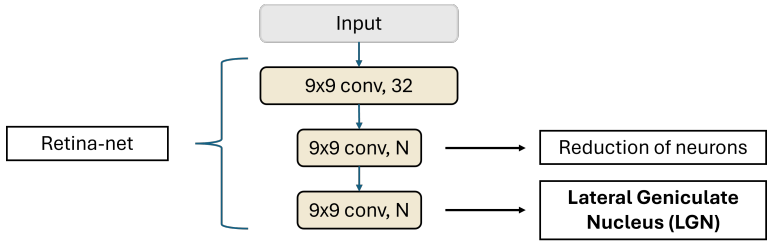

Figure 18: Retina-net, the architecture which represents the Retinal Information Bottleneck worked by [48]. Note that, the three layers version of the retina-net is being used for this work, replicating the LGN as the third layer.

VGG-16 with fixed Gabor in the first convolutional layer and Retinal Informatio Bottleneck

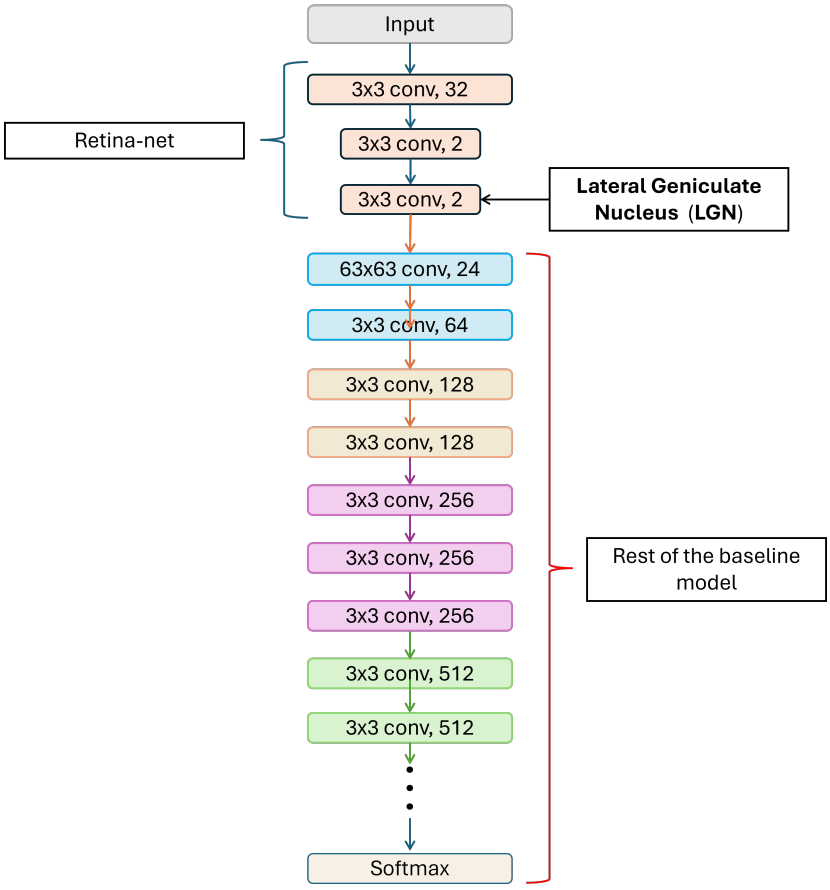

Figure 19: Retinal Information Bottleneck architecture incorporated in the baseline model.

Additionally, an extra version of the bottleneck is carried out with N = 4 . In this sense, it is possible to measure the effects of the size of the bottleneck in the accuracy score.

3.3.2 Recurrent Connections Model

To add recurrent connections to the baseline models, it is important to make them as similar as possible to the visual cortex, acknowledging that the standard architectures used in the baseline models differ from those of the visual stream. As mentioned in Section 2.4, a significant feature of the visual cortex is that approximately 40% of its connections are recurrent. This is a crucial factor to consider for enhancing the model's similarity to the brain, also considering the incorporation of feedback connections (from higher to lower areas), lateral connections, and the bidirectional nature of these connections.

Based on the description and Figure 13 of the ventral pathway in Section 2.4, the ideal recurrent connections to be implemented in the baseline model would be the one represented in Figure 20. In this model, recurrent connections are added to make the

architecture akin to the visual cortex. However, due to limitations in computational costs and time constraints, the proposed architecture followed in this research is that shown in Figure 21 and 22.

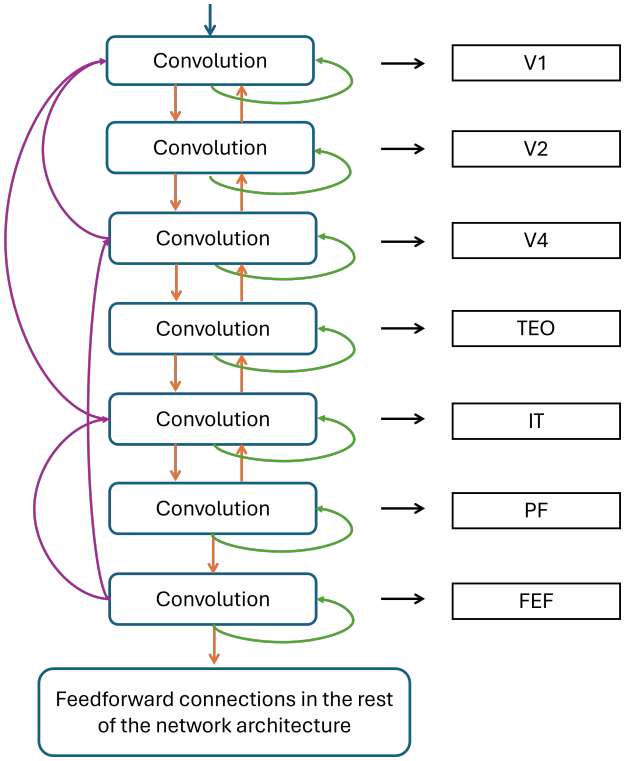

Figure 20: Ideal recurrent connections to achieve an architecture most similar to the visual cortex. Unfortunately, due to the significant increase in model complexity caused by these recurrent connections, it was not feasible to implement this architecture in the current study.

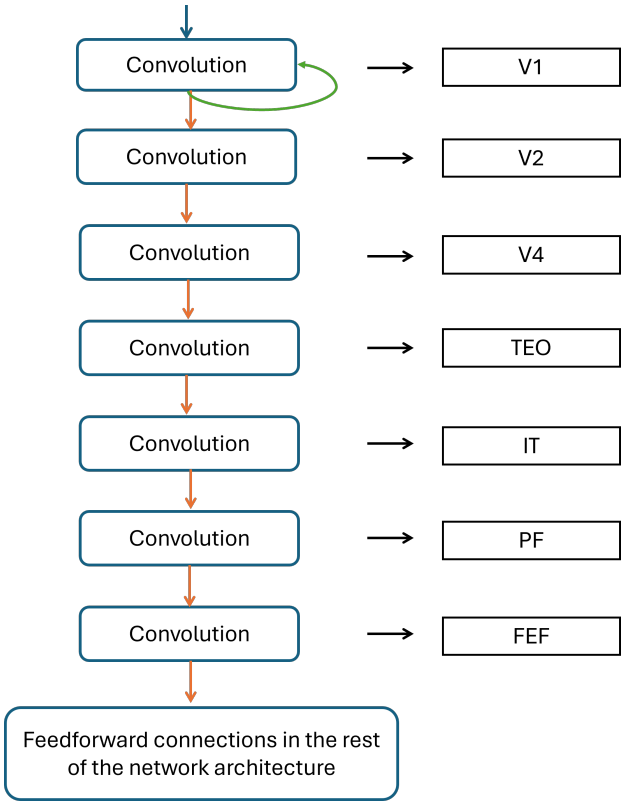

Figure 21: Recurrent connections incorporated in the baseline model.

The diagram of Figure 21 represents how these connections are implemented in the baseline architecture, while Figure 20 represents an ideal (not implemented) case. Here, an analogy of each convolutional layer is made as a region of the visual stream, where the first convolutional layer corresponds to V1, the second convolutional layer represents V2, the third convolutional layer is aligned with V4, the fourth convolutional layer corresponds to TEO, the fifth convolutional layer represents IT, the sixth convolutional layer corresponds to PF, the seventh convolutional layer is aligned with FEF, following the structure seen in Figure 13 of Section 2.4.

The recurrent connections of both diagrams are presented as follows:

- Feedforward and Feedback Connections: Shown in orange arrows, these connections carry information from one layer to the next in a bidirectional manner.
- Top-bottom Connections: Illustrated in purple arrows, these connections provide information back from a higher layer to a lower layer, allowing for feedback processing.
- Lateral Connections: Represented by green arrows, these are connections of a neuron to itself.

VGG-16 with fixed Gabor in the first convolutional layer and recurrent connections

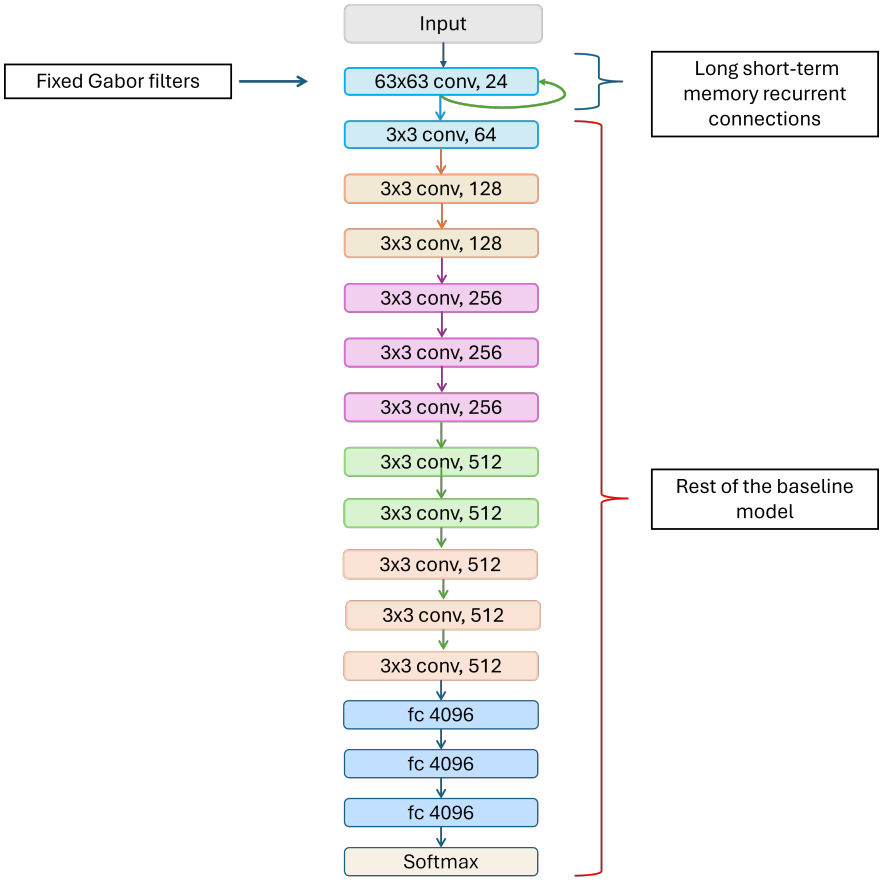

Figure 22: Recurrent connections in the baseline model architecture.

Figure 22 illustrates the complete model architecture. The first convolutional layer, which contains fixed Gabor filters and represents V1, was selected for the integration of recurrent connections. These connections were implemented using the Long Short-Term Memory (LSTM) object in Keras. For the LSTM, a sigmoid function was used for the recurrent activation, while a tanh function was employed for the primary activation function, with an orthogonal initializer. The original properties of the first convolutional layer in the baseline model, including kernel size, padding, and stride, were preserved. Note that, each color represents a distinct block of convolutional layers, with a MaxPooling layer between each different block.

3.3.3 Retinal Information Bottleneck with Recurrent Connections Model

A combination of the architectures shown in Figure 19 and Figure 22 is carried out to integrate the compression of information in the retina output with the recurrent connections existing in the visual stream.

The retina-net with N = 2 outputs information through the third layer, which repre-

sents the LGN, to be received by the recurrent convolutional layer with fixed gabor filters. Then, the rest of convolutional layers remain as the baseline architecture.

VGG-16 with fixed Gabor in the first convolutional layer, recurrent connections and Retinal Information Bottleneck

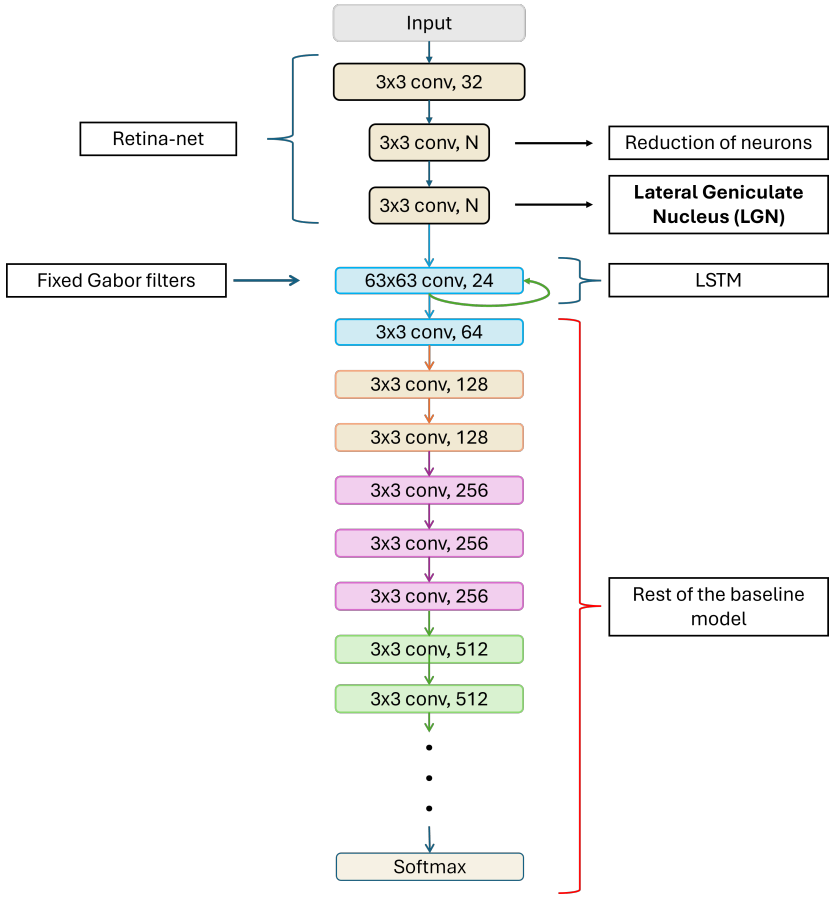

Figure 23: Recurrent connections and information bottleneck incorporated in the baseline model. Note that, the bottleneck in this model has a size of N = 2.

4 Results

4.1 Performance on CIFAR-10G

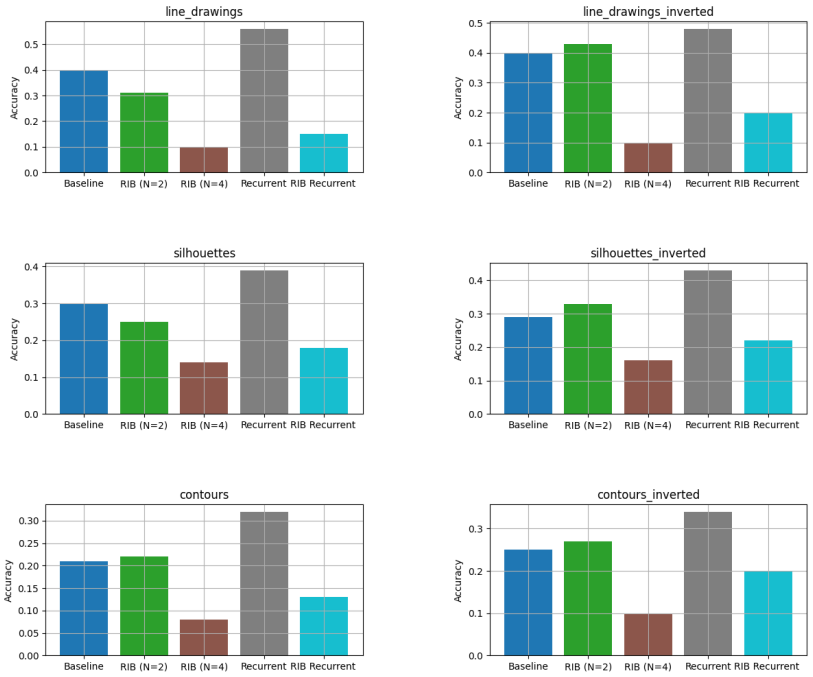

Figure 24: Accuracy of each model on the CIFAR-10G test generalisation sets.

The charts of Figure 24 collectively demonstrate that adding recurrent connections achieves a big enhancement to the model's accuracy across all test sets, scoring the highest punctuation.

Notably, the RIB (N=2) significantly improves performance on all the inverted tests. Additionally, RIB (N=2) also boosts performance in the contours generalization test set when not inverted; however, it appears less effective in the non-inverted line drawings and silhouettes generalization test sets than the baseline model. Interestingly, a marked drop in accuracy is observed when N is increased from 2 to 4, indicating that a smaller number of neurons in the bottleneck contributes to better performance. Moreover, when N = 4, the model shows little difference in performance between the inverted and non-inverted test sets, in contrast to the RIB with N = 2, which shows a notable improvement in inverted test sets in comparison with non-inverted.

The combination of RIB (N=2) and recurrent connections (RIB Recurrent model), remarkably, does not seem to benefit the baseline model. While it performs better on inverted test sets compared to non-inverted ones, its overall performance remains poor compared to the baseline, recurrent-only, and the RIB (N = 2 and N = 4) models.

This indicates that the Baseline model outperforms both the RIB (N = 4) model and the RIB Recurrent model. As shown in Figure 24, the Baseline model achieves a significantly higher score than these two models, whereas the RIB (N=2) model tend to surpass the Baseline model on inverted test sets, but not on the non-inverted, with the

exception of the contours, where the RIB (N=2) model still outperforms the Baseline model.

4.2 Performance on the perturbed CIFAR-10

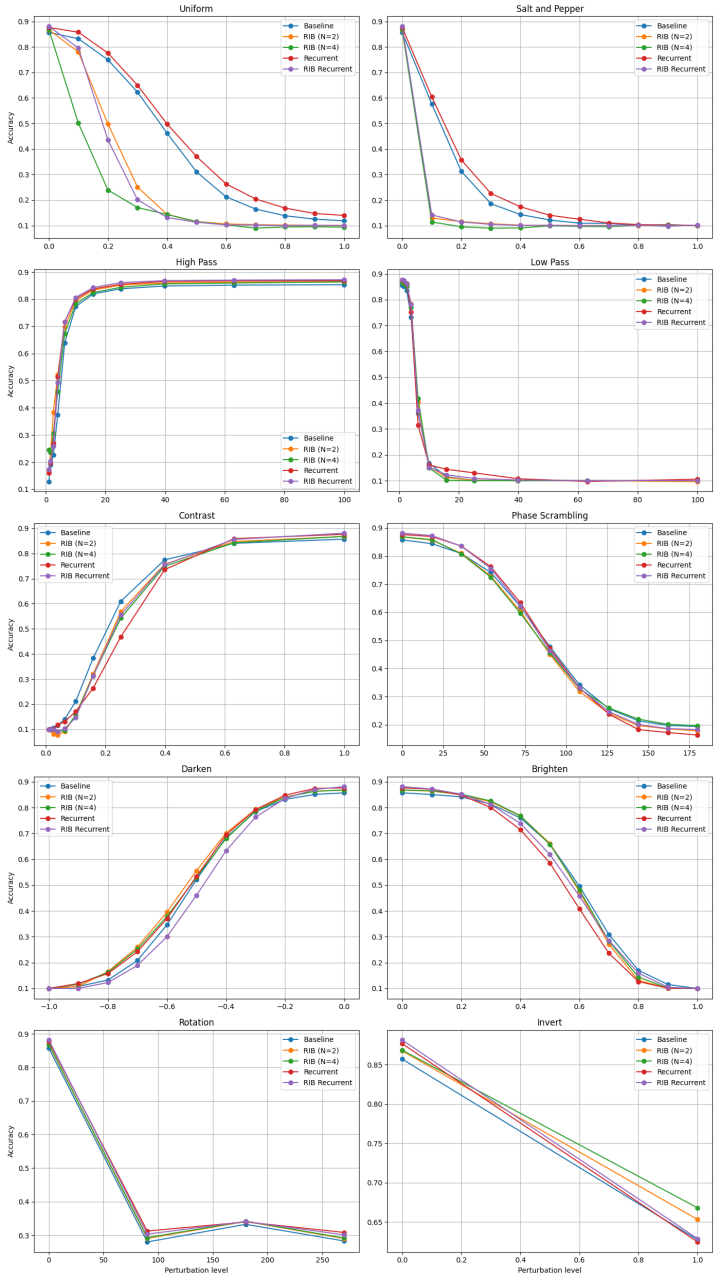

Figure 25: The accuracy of each model on the CIFAR-10 dataset when different types and levels of noise are added. The horizontal axis represents the noise level, while the vertical axis indicates the accuracy of each model. Each graph corresponds to a specific type of noise, showcasing how the models perform as the severity of the noise increases.

The charts present a comprehensive comparison of the accuracy of the Baseline model, RIB (N=2) model, RIB (N=4) model, Recurrent model, and RIB Recurrent model across various types of noise perturbations applied to the CIFAR-10 test set. Each chart corresponds to a specific type of noise or image transformation, including Uniform noise, Salt and Pepper noise, High Pass filter, Low Pass filter, Contrast adjustment, Phase Scrambling, Darkening, Brightening, Rotation, and Inversion.

In general, the Recurrent model consistently performs better under low to moderate noise levels across most perturbation types. The RIB model, particularly with N=4, shows a significant drop in performance regarding RIB model (N = 2), indicating that the reduction in neurons has a substantial impact on accuracy.

The Recurrent model demonstrates improved resilience against noise, particularly in perturbations like Uniform, and Salt and Pepper, where it maintains higher accuracy compared to the other models, followed by the Baseline model. Interestingly, the combination of RIB (N=2) and recurrent connections (RIB Recurrent) does not yield a consistent improvement over the Recurrent model alone, and underperforms compared to the Baseline and Recurrent models. Furthermore, although the RIB (N = 4) model performs poorly in comparison to the other models, it demonstrates the best performance on the Invert test set, followed closely by the standard RIB (N=2) model. This suggests that the RIB (N=2) architecture contributes to improved performance against this specific type of noise.

However, there is no clear improvement evident among the models when it comes to the majority of noise types, as they all exhibit very similar performance across different levels of noise severity, ultimately converging to nearly identical points at corresponding severity levels.

4.3 Performance on the perturbed CIFAR-10G

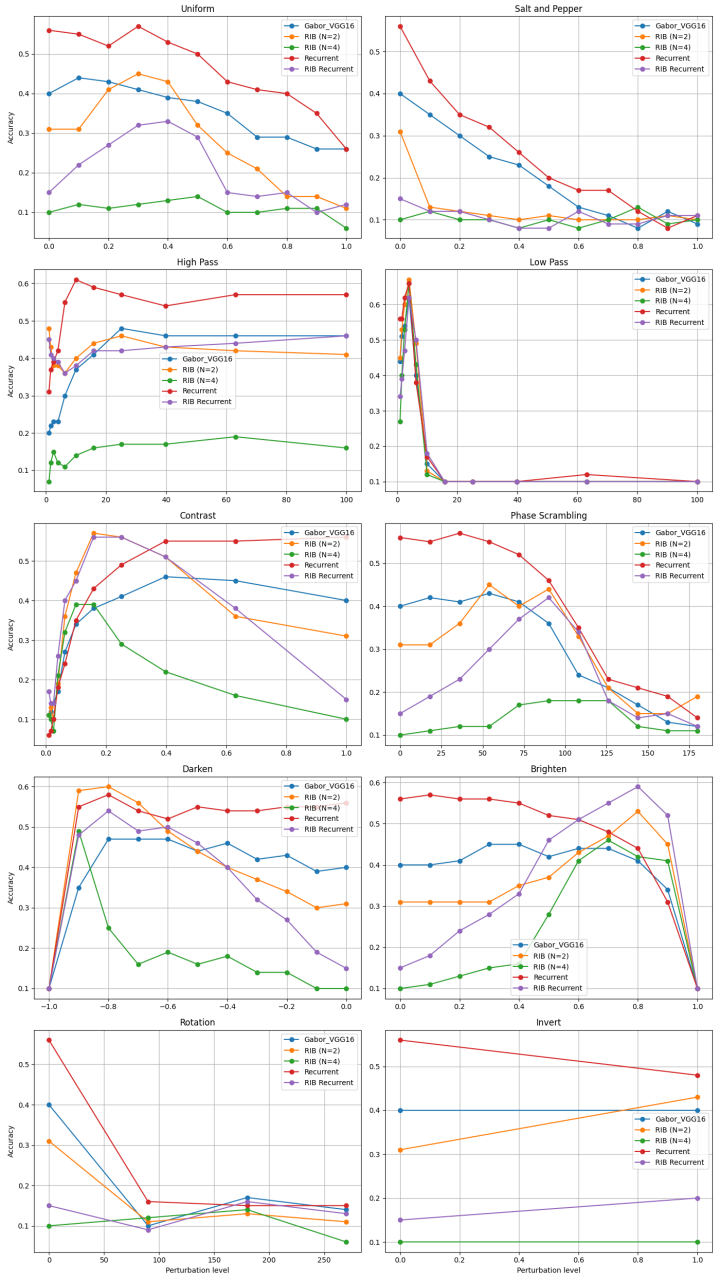

Figure 26: Performance of each model on the Perturbed Line Drawings generalisation test set.

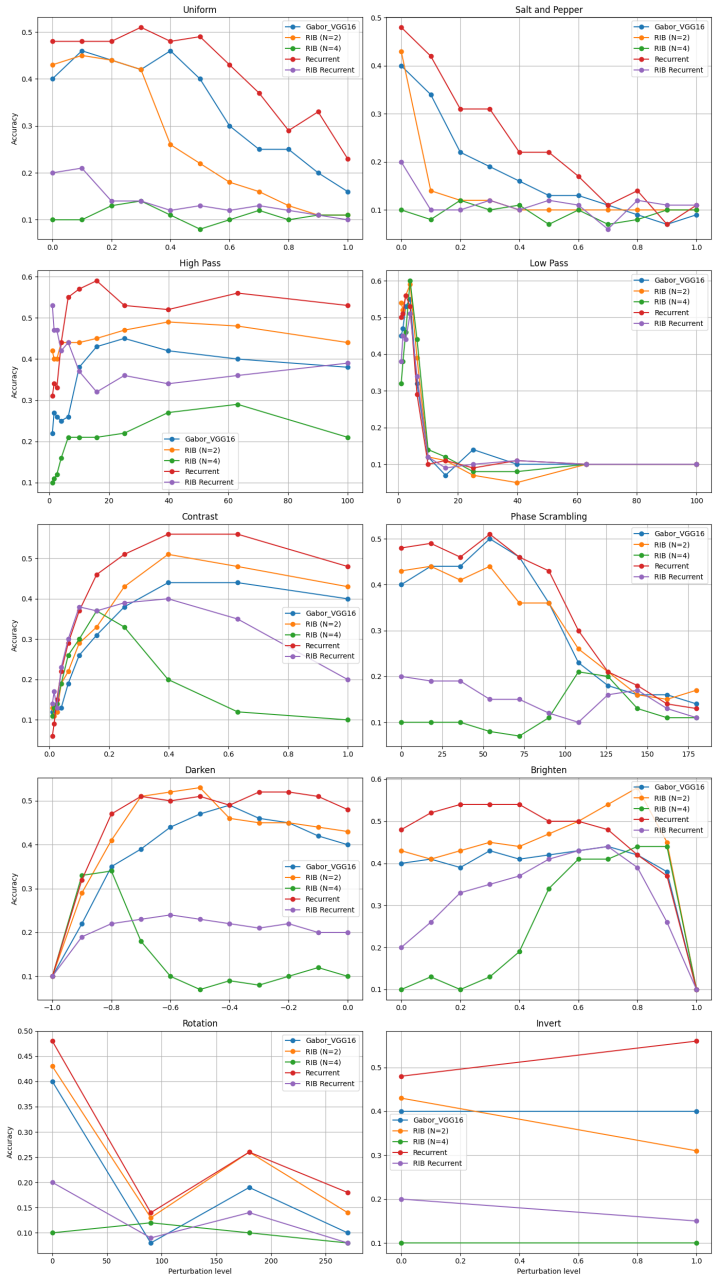

Figure 27: Performance of each model on the Perturbed Line Drawings Inverted generalisation test set.

In Figures 26 and 27, the performance of the models on the perturbed Line Drawings dataset, both inverted and non-inverted, is illustrated. In the non-inverted test set, the Recurrent model consistently outperforms others, maintaining higher accuracy across most noise types. It exhibits significant resilience, although its performance drops rapidly when faced with Low Pass (along with all the other models) and Phase Scrambling noise at higher noise levels.

The Baseline model follows closely to the Recurrent model, showing strong robustness, though not quite matching the Recurrent model's overall performance. It has the same pattern of behaviour as the Recurrent model but in a lower level. Although it seems to have a very similar performance in the rotate noise.

The RIB (N=2) model has great performance only when the noise is low, but drops sharply as the noise rises. The RIB (N=2), RIB (N=4), and RIB Recurrent models generally display lower performance. However, the RIB (N=2) model performs notably well against Phase Scrambling, Brighten, and Invert noise types.

In comparison to these results, the Inverted Line Drawings test set reveals a different trend. Here, the RIB (N=2) model surpasses the Baseline performance, showing strong resilience to inverted test sets. The RIB (N=2) model consistently outperforms the Baseline in all inverted test conditions except for Uniform and Salt and Pepper noise. Remarkably, it even exceeds the Recurrent model's performance at higher noise levels for Phase Scrambling and Brighten noise. Nevertheless, the Recurrent model remains the most robust across the majority of noise types.

Additionally, although the RIB Recurrent model generally underperforms compared to the Baseline model across most noise levels, it does show some peaks in accuracy at low noise levels for the Contrast, Darken, and Phase Scrambling types. However, these peaks are still lower than the accuracies of all other models except for the RIB (N = 4) model, and the performance drops sharply beyond a certain point. Notably, when Brighten noise is introduced in the inverted test set, the RIB Recurrent model surpasses the accuracy of all other models. However, this improvement is not consistent in any other test set.

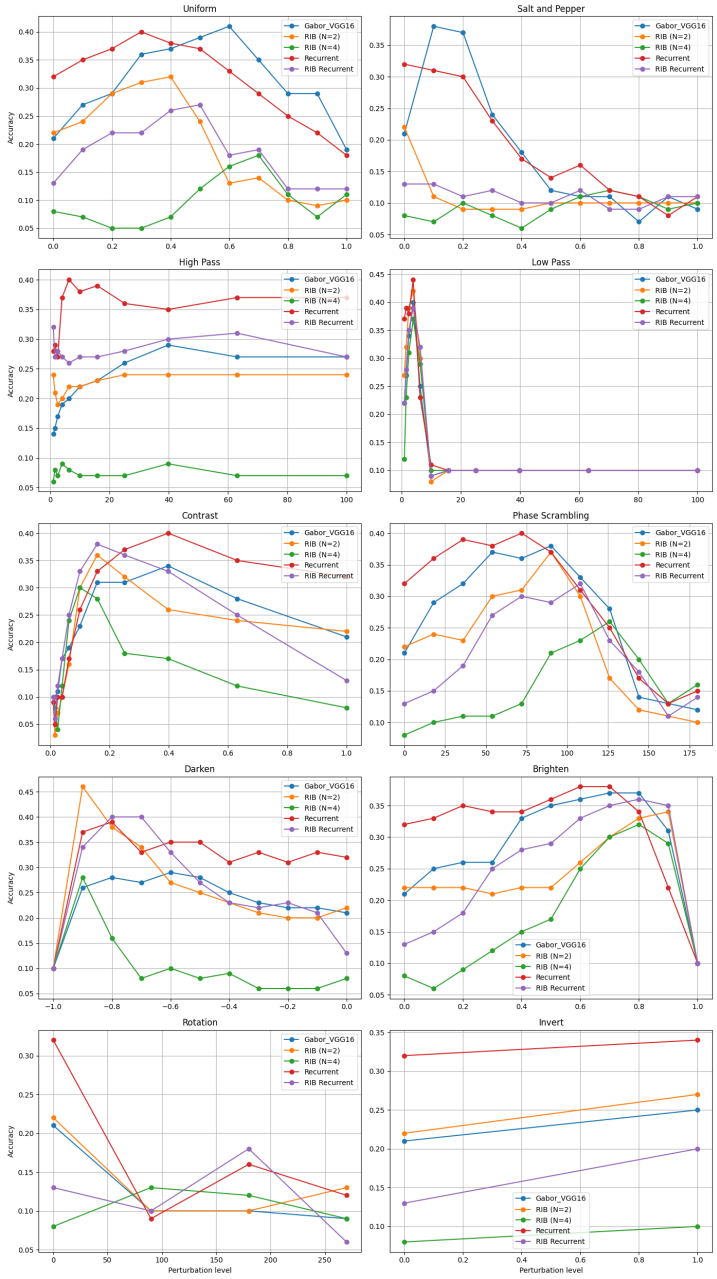

Figure 28: Performance of each model on the Perturbed Contours generalisation test set.

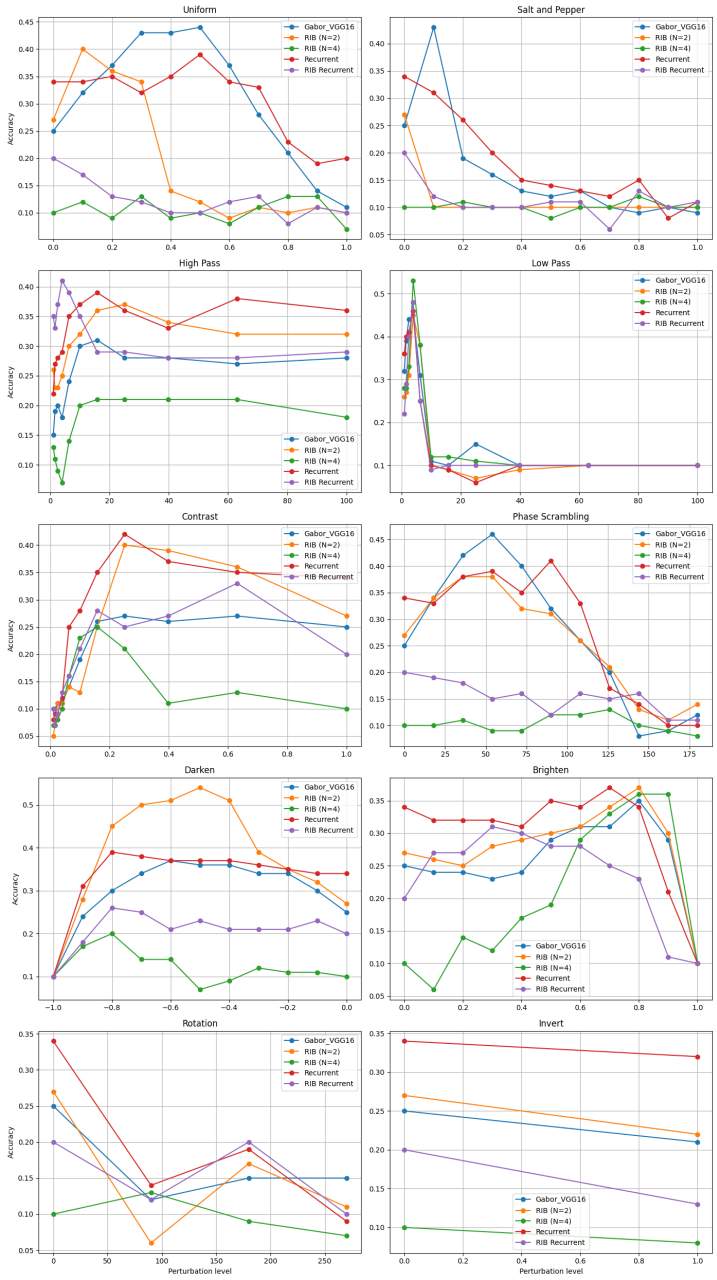

Figure 29: Performance of each model on the Perturbed Contours Inverted generalisation test set.

The Contours dataset (Figures 28 and 29) depicts that the Baseline model surpasses to the Recurrent model in the Uniform noise, and it has better accuracy at some particular points in other noise types such as Salt and Pepper, and Phase Scrambling. This happens also in the inverted test sets with the same type of noises. Nonetheless, the Recurrent model undoubtedly outperforms the baseline model in general. As the line drawing test set, the Recurrent model outperforms clearly to the Baseline model.

In the Inverted Contours dataset, the RIB (N=2) model demonstrates a marked improvement, not only matches but also surpasses the performance of the Baseline model in the majority of noises when the test set is inverted, having similar performance to the Recurrent model in noise types such as High Pass, Contrast, Phase Scrambling, Invert and even outstanding in the Darken noise. Showing a higher capability to face contours in comparison with silhouettes and line drawings. However, the Recurrent model remains remarkably as the best due to the similarity on the inverted test and the advantage in the non-inverted. It is also interesting that the RIB (N=2) model, while worst at lower levels of noise in the non-inverted tests, it surpasses to the baseline model in the hardest severity levels in noises as Darken, Contrast, Rotation. The RIB (N=2) model along with the Recurrent dominate the Perturbed Contours Inverted test sets, while the Recurrent model and the Baseline model dominate the non-inverted one.

Furthermore, there are specific instances where the RIB Recurrent's accuracy significantly improves. For example, it exhibits peaks in accuracy under Contrast (inverted), High Pass (inverted), and Rotation noise conditions. Notably, the RIB Recurrent model performs better with High Pass noise, although this appears to be a rather isolated case. However, it is clear that it generally does not outperform the Recurrent, RIB (N=2), and Baseline models at any circunstance.

On the other hand, the RIB (N = 4) remains performing as the poorest model in general.

Regarding the Perturbed Silhouettes generalization tests, the Recurrent model, again, outweighs the rest of the models across nearly all noise types. It is surpassed in the maximum level of Brighten noise and minimally in the Uniform noise by the Baseline model, but Recurrent model is strongly superior in the other cases.

Overall, in this test set, the Baseline model outperforms all other models, with the exception of the Recurrent model. In the inverted test set, the RIB (N=2) and RIB Recurrent models only challenge the Baseline model in the High Pass, Contrast, and Darken noise conditions, and they surpass it in the Rotation noise scenario.

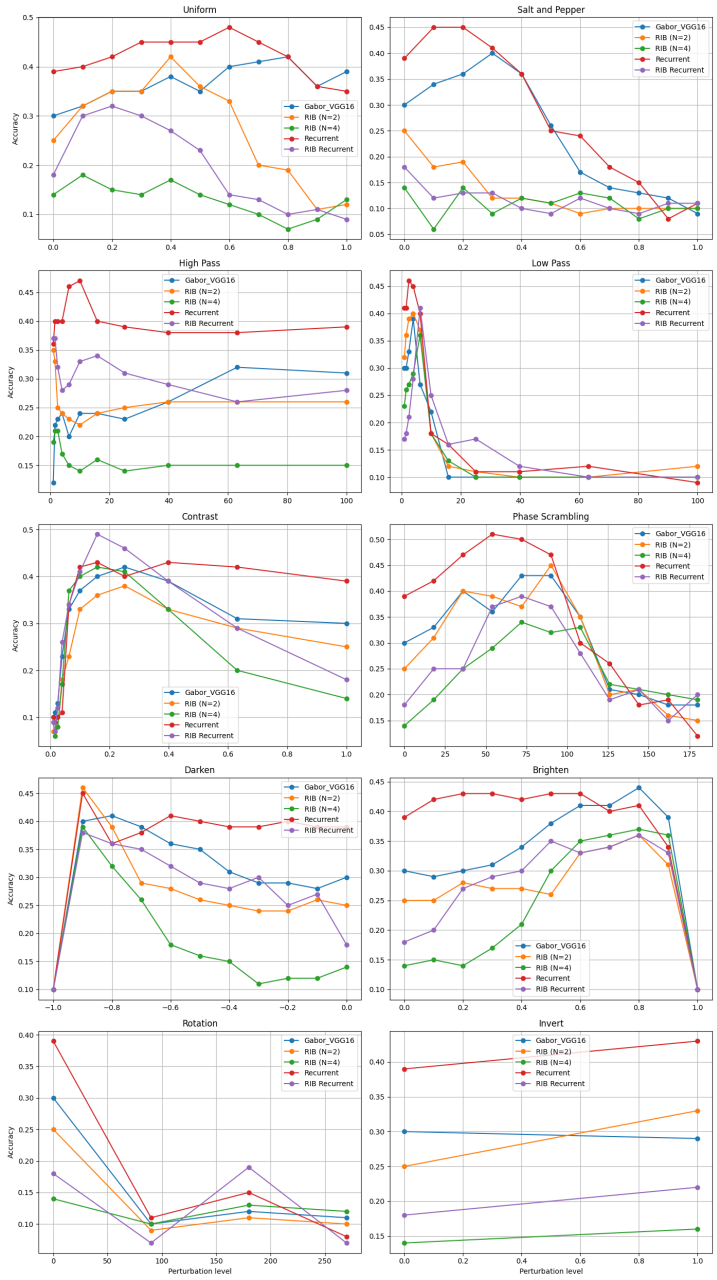

Figure 30: Performance of each model on the Perturbed Silhouettes generalisation test set.

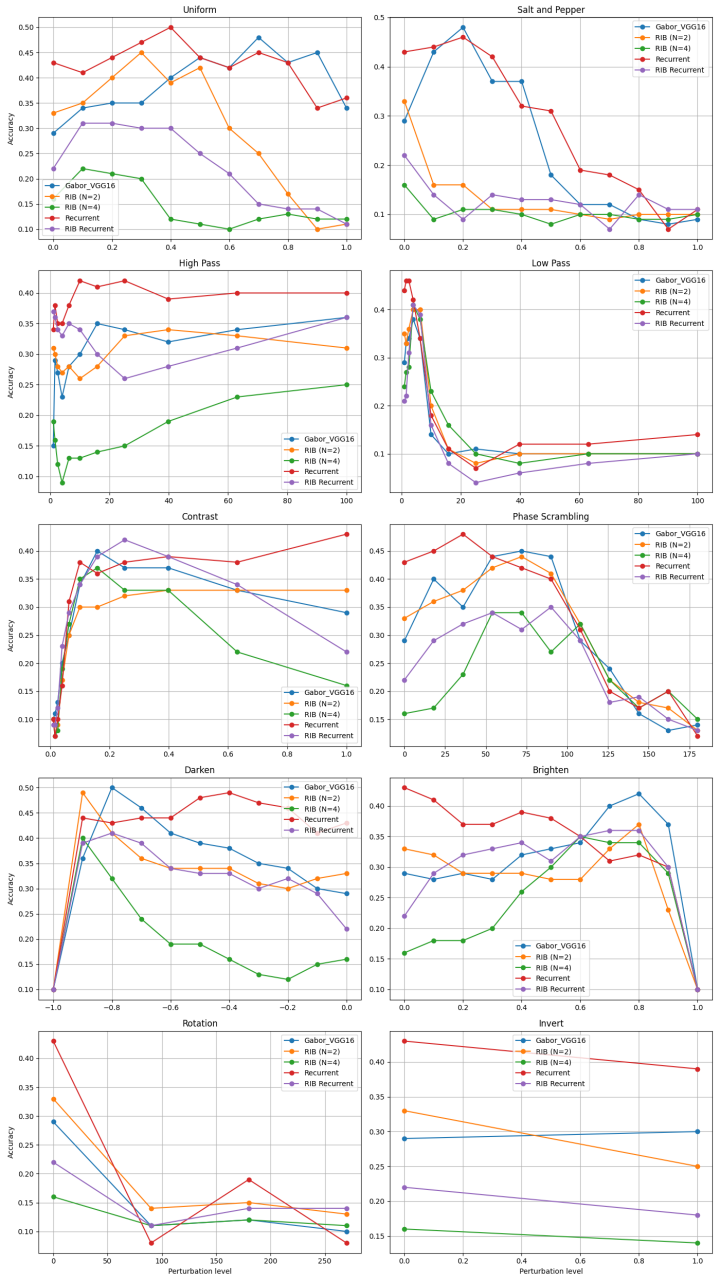

Figure 31: Performance of each model on the Perturbed Silhouettes Inverted generalisation test set.

5 Discussion and Future Work

5.1 Summary of Findings and Conclusion

In this research project, the aim was to enhance the architecture of the Baseline model presented in [9] by incorporating properties inspired by the visual cortex, with the purpose of increasing its robustness to noise and its ability to generalize to unseen data. The original Baseline model, based on the VGG-16 architecture, utilizes fixed Gabor filters that emulate the early stages of visual processing in the human brain. To explore potential improvements, several models were developed that added recurrent connections and an information reduction mechanism (RIB) in different configurations. The first modified model introduced recurrent connections in the layer where the Gabor filters are located, allowing for feedback of information in a structure inspired by the recurrent connections observed in the visual cortex. This model, referred to as Recurrent, demonstrated a notable superiority compared to all other models, including the Baseline model, as shown in Figures 24 to 31. However, it is important to emphasize that further research is necessary to determine whether this improvement is solely due to the increase in the model's parameters or if it truly represents an optimization based on the properties of the visual system. The addition of these recurrent connections resulted in a considerable increase in the model's ability to handle out-of-distribution data and improve its robustness to noise, findings that are also suggested by [62] [34]. On the other hand, a model incorporating an information reduction mechanism (RIB) inspired by [48] was developed. Two configurations of the RIB were tested: one with two neurons and another with four neurons. The RIB model with N = 2 exhibited lower performance compared to the Baseline in the "non-inverted line drawings" and "silhouettes" test sets of the CIFAR-10G dataset. However, in the inverted test sets, the RIB model with N = 2 showed significant improvement, surpassing the Baseline in all these cases, as well as in the "non-inverted contours" test set. Likewise, it notably increases generalization performance in response to Darken, Contrast, and High Pass filters, and the contours generalisation tests. This could be aligned with the "center-surround receptive fields" that emerge in the bottleneck, as described in [48], which help to form oriented receptive fields in posterior layers that allow a better edges and contours detection in images, and which might enable the visual system to efficiently detect contrasts and succeed in difficult low-level light scenarios. This behavior suggests that the RIB functions as a mechanism that enhances the model's capacity in scenarios where images are inverted, where there is greater contrast, lower light levels, and where classification relies more heavily on the contours of the images.

In contrast, the RIB model with N = 4 demonstrated significantly lower performance across almost all tests. This poor performance could be attributed to the fact that, as the number of neurons in the bottleneck increases, the model's ability to produce "center-surround receptive fields" might be diminished, thereby limiting its capacity to detect contours and contrasts. This, in turn, affects its ability to generalize in scenarios where these elements are crucial. This finding highlights the importance of appropriately reducing the amount of information in the model. The RIB Recurrent model, which combined the RIB with two neurons and recurrent connections, did not meet the expectations of outperforming the models when used separately. In fact, it was outperformed by both the RIB (N = 4) model and the Recurrent model when evaluated independently, as well as by the Baseline model. Similar to the RIB (N=2) model, the RIB Recurrent model demonstrated superior performance in scenarios where the dataset was inverted. However, the combination of these two modifications, rather than enhancing performance, have led to a decrease in generalization capacity and robustness to noise. Despite some performance

peaks observed in specific situations, such as under high levels of noise from the Contrast, Darken, High Pass, and Rotation noise types, these were isolated cases and generally did not indicate a consistent improvement. This suggests two possibilities: either the reduction in neurons within the bottleneck may have limited the amount of rich information reaching the layer with recurrent connections, which is positioned after the bottleneck, thereby reducing the effectiveness of the recurrent connections and, consequently, the overall performance of the model; or the recurrent connections may have inhibited the emergence of the "center-surround receptive fields" that naturally form with the RIB. In the first case, it will be necessary to find a way to address this issue, as this reduction is based on the literature [52]. One possible solution could be to incorporate additional recurrent connections that mimic the visual stream, like feedback from higher layers to the first layer (for example, from V4 to V1 in the visual stream). In the second case, a more appropriate method for integrating recurrence, consistent with the visual cortex, should be sought. For example, this could involve lateral connections between neurons within the same layer, since that is the form in which recurrence is presented in the visual cortex [10][39], rather than recurrence of a neuron onto itself. In conclusion, the results obtained in this study indicate that incorporating recurrent connections into the Baseline model significantly enhances its ability to handle out-of-distribution data and increases its robustness against various types of noise. This finding suggests that the recurrent properties observed in the visual system can be leveraged to optimize deep learning models for computer vision tasks. Additionally, although the use of the RIB showed limitations against most types of noise, it proved to be effective in improving the model's generalization in scenarios where images are inverted, the crucial information primarily comes from contours, and there is a high level of Darken, Contrast, Rotation, or High Pass noise. This reinforces the idea that certain architectures may be better adapted to handle specific image scenarios, although not necessarily to improve overall performance across all types of noise. However, the combination of the RIB with recurrent connections did not result in an additional improvement; in fact, it reduced the model's generalization capacity and robustness to noise. This suggests that further investigation is needed to determine the most effective way to implement these two modifications together, in order to maintain their properties within a single model and enhance its overall performance.

5.2 Limitations and Future Research Directions

In future research, incorporating internal noise into the architecture could be a key area for further exploration. Internal noise is an inherent characteristic of biological neurons, contributing to the stochastic nature of neuronal responses [64]. This variability, while seemingly random, can actually improve the robustness and generalization capabilities of neural networks by preventing over-fitting. Recent studies have demonstrated that adding controlled amounts of internal noise to CNNs can lead to improved performance [68], especially in the presence of noise and out of distribution data.

The architectural design of the recurrent networks has been a significant limitation in this study, particularly in efforts to make the network more closely resemble the visual cortex. This resemblance is essential for testing whether adding more properties based on the visual stream can enhance generalization and robustness to noise. Although this research only incorporated a single convolutional recurrent layer, it would be far more effective to integrate recurrent connections into approximately 40% of the convolutional layers to better mimic the visual cortex, where such recurrent connections are abundant and constitute around that percentage of the connectivity [10][39]. Unfortunately, due to computational costs and time constraints, it was not feasible to design a more biolog-

ically accurate architecture, as adding recurrent neural networks significantly increases the model's complexity and the number of parameters.

Furthermore, one of the most crucial aspects when incorporating properties of the visual cortex is not just the presence and quantity of recurrent connections, but rather the specific manner in which these connections are found in the visual stream. For example, higher visual areas such as the IT provide feedback to areas like V1, while V4 also connects back to V1, and the FEF sends feedback to the PF, among others. This type of structured connectivity is illustrated in Figure 20 and is important to replicate to yield a better representation of the visual cortex. Additionally, the RIB model with recurrent connections should incorporate these connections in the final neural layer of the bottleneck, mirroring the existing recurrent connections found in the LGN, where the reduction in the number of neurons is.

Moreover, current implementations of frameworks such as TensorFlow and Keras in Python do not support lateral recurrence among neurons within each convolutional layer and therefore, there was not enough time to technically developed these properties. However, this type of recurrence is fundamental in the visual cortex and has been shown to play a significant role in enhancing robustness to noise [62]. Designing these connections to replicate those present in the visual stream could better mimic its functionality and yield improved results in terms of generalization and noise robustness. An interesting approach would be reducing the number of convolutional recurrent neural layers to the amount presented in the visual stream, as done by [45], and applying these specific recurrent designs to see if it helps to achieve similar or better results while also reducing the number of layers, thereby making the architecture more comparable to the fewer layers present in the visual stream.

Also, exploring how the internal representations of each layer change while adding these properties should be pursued to better understand why certain combinations of properties may fail.

Lastly, it is important to include a greater number of severity levels in future research, as some models in the current study seem to benefit more than others as noise levels increase. This potential advantage may not be fully captured due to the interruption in the severity levels currently used.

References

[1] S. Agarwal, J. O. D. Terrail, and F. Jurie, "Recent advances in object detection in the age of deep convolutional neural networks," *arXiv preprint arXiv:1809.03193*, 2018.

[2] N. Baker, H. Lu, G. Erlikhman, and P. J. Kellman, "Deep convolutional networks do not classify based on global object shape," *PLOS Computational Biology*, vol. 14, no. 12, 2018. DOI: 10.1371/journal.pcbi.1006613.

[3] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 2, pp. 115–147, 1987. DOI: 10.1037/0033-295X.94.2.115.

[4] C. Cao, Y. Liu, Z. Yang, Y. Wang, and Y. Wang, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2015, pp. 2956–2964. DOI: 10.1109/ICCV.2015.338.

[5] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, Université de Rennes 1, La Baule, France, 2006.

[6] J. G. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision Research*, vol. 20, no. 10, pp. 847–856, 1980. DOI: 10.1016/0042-6989(80)90065-6. [Online]. Available: https://doi.org/10.1016/0042-6989(80)90065-6.

[7] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of the Optical Society of America A*, vol. 2, no. 7, pp. 1160–1169, Jul. 1985. DOI: 10.1364/josaa.2.001160.

[8] B. Evans, *Cifar-10g (version v1.0.0)*, 2021. DOI: 10.5281/zenodo.5648918. [Online]. Available: http://dx.doi.org/10.5281/zenodo.5648918.

[9] B. D. Evans, G. Malhotra, and J. S. Bowers, "Biological convolutions improve dnn robustness to noise and generalisation," *Neural Networks*, vol. 148, pp. 96–110, 2022. DOI: 10.1016/j.neunet.2021.12.005.

[10] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol. 1, no. 1, pp. 1–47, 1991. DOI: 10.1093/cercor/1.1.1-a.

[11] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[12] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[13] R. Geirhos, J. H. Jacobsen, C. Michaelis, *et al.*, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020. DOI: 10.1038/s42256-020-00257-z.

[14] R. Geirhos, D. H. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann, "Comparing deep neural networks against humans: Object recognition when the signal gets weaker," *arXiv preprint arXiv:1706.06969*, 2017.

[15] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.

[16] C. Gilbert and W. Li, "Top-down influences on visual processing," *Nature Reviews Neuroscience*, vol. 14, pp. 350–363, 2013. DOI: 10.1038/nrn3476. [Online]. Available: https://doi.org/10.1038/nrn3476.

[17] C. D. Gilbert, "Principles of neural science," in *Principles of Neural Science*, E. R. Kandel, J. Schwartz, T. Jessel, S. A. Siegelbaum, and A. J. Hudspeth, Eds., 5th, McGraw-Hill Companies, 2012, ch. 25.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[19] A. Graves, *Supervised Sequence Labelling*. Springer Berlin Heidelberg, 2012, pp. 5–13.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[21] K. L. Hermann, T. Chen, and S. Kornblith, "The origins and prevalence of texture bias in convolutional neural networks," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., vol. 33, 2020, pp. 19 000–19 015.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] D. H. Hubel, *Eye, brain, and vision*. Scientific American Library/Scientific American Books, 1995.

[24] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *Journal of Physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[25] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[26] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *arXiv preprint arXiv:1905.02175*, 2019.

[27] C. Jarvers and H. Neumann, "Incorporating feedback in convolutional neural networks," in *Proceedings of the Cognitive Computational Neuroscience Conference*, 2019, pp. 395–398.

[28] J. Jones and L. Palmer, "Simple receptive fields in cat striate cortex: A comparison with gabor functions in two dimensions of space and two dimensions of spatial frequency," in *Abstracts of the Society for Neuroscience*, vol. 10, 1984, p. 800.

[29] R. Kafieh, R. Arian, N. Saeedizadeh, *et al.*, "Covid-19 in iran: A deeper look into the future," *MedRxiv*, Apr. 2020, doi:10.1101/2020.04.

[30] G. Keren and B. Schuller, "Convolutional rnn: An enhanced model for extracting features from sequential data," in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016, pp. 3412–3419.

[31] H. Kesserwani, "The biophysics of visual edge detection: A review of basic principles," *Cureus*, vol. 12, no. 10, e11218, Oct. 2020. DOI: 10.7759/cureus.11218.

[32] S.-M. Khaligh-Razavi and N. Kriegeskorte, "Deep supervised, but not unsupervised, models may explain it cortical representation," *PLOS Computational Biology*, vol. 10, no. 11, e1003915, 2014. DOI: 10.1371/journal.pcbi.1003915.

[33] T. C. Kietzmann, P. McClure, and N. Kriegeskorte, "Deep neural networks in computational neuroscience," *BioRxiv*, p. 133 504, 2017. DOI: 10.1101/133504.

[34] T. C. Kietzmann, C. J. Spoerer, L. K. A. Sörensen, R. M. Cichy, O. Hauk, and N. Kriegeskorte, "Recurrence is required to capture the representational dynamics of the human visual system," *Proceedings of the National Academy of Sciences*, vol. 116, no. 43, pp. 21 854–21 863, 2019. DOI: 10.1073/pnas.1905544116.

[35] G. Kreiman and T. Serre, "Beyond the feedforward sweep: Feedback computations in the visual cortex," *Annals of the New York Academy of Sciences*, vol. 1464, no. 1, pp. 222–241, 2020. DOI: 10.1111/nyas.14320.

[36] N. Kriegeskorte, "Deep neural networks: A new framework for modeling biological vision and brain information processing," *Annual Review of Vision Science*, vol. 1, no. 1, pp. 417–446, 2015. DOI: 10.1146/annurev-vision-082114-035447.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, vol. 25, 2012.

[38] J. Kubilius, S. Bracci, and H. P. O. de Beeck, "Deep neural networks as a computational model for human shape sensitivity," *PLOS Computational Biology*, vol. 12, no. 4, e1004896, 2016. DOI: 10.1371/journal.pcbi.1004896.

[39] V. A. Lamme, H. Supèr, and H. Spekreijse, "Feedforward, horizontal, and feedback processing in the visual cortex," *Current Opinion in Neurobiology*, vol. 8, no. 4, pp. 529–535, 1998. DOI: 10.1016/s0959-4388(98)80042-1.

[40] B. Landau, L. B. Smith, and S. S. Jones, "The importance of shape in early lexical learning," *Cognitive Development*, vol. 3, no. 3, pp. 299–321, 1988. DOI: 10.1016/0885-2014(88)90014-7.

[41] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997. DOI: 10.1109/72.554195.

[42] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.

[43] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] Q. Liao and T. Poggio, "Bridging the gaps between residual learning, recurrent neural networks and visual cortex," *arXiv preprint arXiv:1604.03640*, 2016.

[46] S. Liao, J. Wang, R. Yu, K. Sato, and Z. Cheng, "Cnn for situations understanding based on sentiment analysis of twitter data," *Procedia Computer Science*, vol. 111, pp. 376–381, 2017, The 8th International Conference on Advances in Information Technology, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.06.037.

[47] G. W. Lindsay, "Convolutional neural networks as a model of the visual system: Past, present, and future," *Journal of Cognitive Neuroscience*, vol. 33, no. 10, pp. 2017–2031, 2021. DOI: 10.1162/jocn_a_01544.

[48] J. Lindsey, S. A. Ocko, S. Ganguli, and S. Deny, "A unified theory of early visual representations from retina to cortex through anatomically constrained deep cnns," *arXiv preprint arXiv:1901.00945*, 2019.

[49] K. Liu and X. Li, "Bio-inspired multi-sensory pathway network for change detection," *Cognitive Computation*, vol. 14, no. 6, pp. 1421–1434, 2022. DOI: 10.1007/s12559-021-09968-w.

[50] G. Malhotra, B. Evans, and J. Bowers, "Adding biological constraints to cnns makes image classification more human-like and robust," in *2019 Conference on Cognitive Computational Neuroscience*, Cognitive Computational Neuroscience, Berlin, Germany, 2019. DOI: 10.32470/CCN.2019.

[51] G. Malhotra, B. D. Evans, and J. S. Bowers, "Hiding a plane with a pixel: Examining shape-bias in cnns and the benefit of building in biological constraints," *Vision Research*, vol. 174, pp. 57–68, 2020. DOI: 10.1016/j.visres.2020.06.005.

[52] R. Masland, "The fundamental plan of the retina," *Nature Neuroscience*, vol. 4, pp. 877–886, 2001. DOI: 10.1038/nn0901-877. [Online]. Available: https://doi.org/10.1038/nn0901-877.

[53] J. Mehrer, C. J. Spoerer, E. C. Jones, N. Kriegeskorte, and T. C. Kietzmann, "An ecologically motivated image dataset for deep learning yields better models of human vision," *Proceedings of the National Academy of Sciences*, vol. 118, no. 8, e2011417118, 2021. DOI: 10.1073/pnas.2011417118.

[54] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021.

[55] N. Petkov and P. Kruizinga, "Computational models of visual neurons specialised in the detection of periodic and aperiodic oriented visual stimuli: Bar and grating cells," *Biological Cybernetics*, vol. 76, no. 2, pp. 83–96, 1997.

[56] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017. DOI: 10.1162/NECO_a_00990.

[57] S. Ritter, D. G. Barrett, A. Santoro, and M. M. Botvinick, "Cognitive psychology for deep neural networks: A shape bias case study," *arXiv preprint arXiv:1706.08606*, 2017. DOI: https://doi.org/10.48550/arXiv.1706.08606.

[58] A. J. Robinson and F. Fallside, "The utility driven dynamic error propagation network," Cambridge University Engineering Department, Technical Report CUED/F-INFENG/TR.1, 1987.

[59] D. Sarvamangala and R. Kulkarni, "Convolutional neural networks in medical image understanding: A survey," *Evolutionary Intelligence*, vol. 15, no. 1, pp. 1–22, 2022. DOI: 10.1007/s12065-020-00540-3.

[60] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[62] C. J. Spoerer, P. McClure, and N. Kriegeskorte, "Recurrent convolutional neural networks: A better model of biological object recognition," *Frontiers in Psychology*, vol. 8, p. 1551, 2017. DOI: 10.3389/fpsyg.2017.01551.

[63] G. Sreenu and M. Saleem Durai, "Intelligent video surveillance: A review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 1, p. 48, 2019. DOI: 10.1186/s40537-019-0212-5.

[64] R. Stein, E. Gossen, and K. Jones, "Neuronal variability: Noise or part of the signal?" *Nature Reviews Neuroscience*, vol. 6, pp. 389–397, 2005. DOI: 10.1038/nrn1668. [Online]. Available: https://doi.org/10.1038/nrn1668.

[65] F. Sultana, A. Sufian, and P. Dutta, "Evolution of image segmentation using deep convolutional neural network: A survey," *Knowledge-Based Systems*, vol. 201, p. 106 062, 2020.

[66] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[67] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000. [Online]. Available: https://arxiv.org/abs/physics/0004057.

[68] C. Tsvetkov, G. Malhotra, B. D. Evans, and J. S. Bowers, "The role of capacity constraints in convolutional neural networks for learning random versus natural data," *Neural Networks*, vol. 161, pp. 515–524, 2023. DOI: 10.1016/j.neunet.2023.03.018.

[69] M. R. Turner, "Texture discrimination by gabor functions," *Biological Cybernetics*, vol. 55, no. 2-3, pp. 71–82, 1986. DOI: 10.1007/BF00341922.

[70] J. Wagemans, *The Oxford Handbook of Perceptual Organization*. Oxford University Press, 2015. DOI: 10.1093/oxfordhb.

[71] Z. J. Wang, R. Turko, O. Shaikh, *et al.*, "Cnn explainer: Learning convolutional neural networks with interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1396–1406, 2020.

[72] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in *Back-propagation: Theory, Architectures and Applications*, Hillsdale, NJ: Erlbaum, 1992.

[73] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo, "Performance-optimized hierarchical models predict neural responses in higher visual cortex," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8619–8624, 2014. DOI: 10.1073/pnas.1403112111.

[74] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in Neural Information Processing Systems*, Curran Associates, Inc., vol. 27, 2014.

[75] É. Zablocki, H. Ben-Younes, P. Pérez, *et al.*, "Explainability of deep vision-based autonomous driving systems: Review and challenges," *International Journal of Computer Vision*, vol. 130, pp. 2425–2452, 2022. DOI: 10.1007/s11263-022-01657-x.

[76] A. R. Zamir, T.-L. Wu, L. Sun, *et al.*, "Feedback networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1308–1317.