RESEARCH PROJECT IN MECHATRONICS ENGINEERING

A CONVOLUTIONAL NEURAL NETWORK FOR DETECTING VISUAL TEXTURE

Conrad Scherb

Project Report ME110-2022

Co-worker: Callan Loomes

Supervisor: Dr Luke Hallum

Department of Mechanical and Mechatronics Engineering
The University of Auckland

14 October 2022

ME110-2022

A CONVOLUTIONAL NEURAL NETWORK FOR DETECTING VISUAL TEXTURE

Conrad Scherb

ABSTRACT

Visual systems are fairly poorly understood in contempoary physiology. The components of visual pathways, such as neuron types and their theorised function, are well established, yet the connections between these components are difficult to determine. Only so much can be identified histologically. Neural networks could help further our understanding of visual pathways in biology by creating models consistent with our current knowledge. These can then be re-examined to see what they might tell us about actual biology.

Visual texture, defined as the difference of orientation or spatial frequency between two parts of an image, is a key component of visual processing performed by the primary visual cortex. We developed a wide variety of over one hundred convolutional neural network models trained upon idealised textures modelled as regions of different overall orientation on a Gabor patch array. We found that two-layered models were the most accurate and provided the best match to our biological understanding and non-neural network-based models of texture discrimination. Investigating our convolutional kernels fitted, which are models of receptive fields, allowed us to identify how simple and complex cells work together to discriminate texture. The ideal kernel size, particularly in simpler models, needed to span multiple Gabor patches to detect texture.

Our results show that convolutional neural networks are a valuable tool for investigating visual processing, with our final model correctly classifying the presence or absence of a texture with 95.1% accuracy. However, more research is needed to further model the primary visual cortex, particularly in the activity of complex cells, which are generally poorly understood in the literature.

DECLARATION

Student

I hereby declare that:

1. This report is the result of the final year project work carried out by my project partner (see cover page) and I under the guidance of our supervisor (see cover page) in the 2022 academic year at the Department of Mechanical and Mechatronics Engineering, Faculty of Engineering, University of Auckland.
2. This report is not the outcome of work done previously.
3. This report is not the outcome of work done in collaboration, except that with a potential project sponsor (if any) as stated in the text.
4. This report is not the same as any report, thesis, conference article or journal paper, or any other publication or unpublished work in any format.

In the case of a continuing project, please state clearly what has been developed during the project and what was available from previous year(s):

Signature: Conrad Scherb

Date: 14th of October 2022

Supervisor

I confirm that the project work undertaken by this student in the 2022 academic year is / is not (strikethrough as appropriate) part of a continuing project, components of which have been completed previously. Comments, if any:

Signature:

Date: 10th of October 2022

Table of Contents

List of Figures

Acknowledgements

Throughout the course of this project, I would like to thank several key people who supported me and this research.

- Dr. Luke Hallum, our supervisor for this project for his knowledge in this project field and guiding us in the right direction the entire way.
- Callan Loomes, my project partner for this project for his critical help in developing networks, finding key discoveries and for time and help throughout the whole project.
- The AI team at HeartLab, the company I currently work for, for helping me get past the initial hurdles understanding how CNNs work exactly and how they might be applicable for this project.

Glossary of Terms

TensorFlow	The industry standard open-source library for machine learning and artificial intelligence applications.
MATLAB	A proprietary programming environment with numeric computing capabilities useful for visualising and manipulating data.

Abbreviations

V1	Visual cortex layer 1
2D	Two dimensional
API	Application programming interface
GPU	Graphics processing unit
CNN	Convolutional neural network
FRF	Filter-rectify-filter [model]

1. Scope and Objectives

Visual perception is an extremely complex task, requiring the integration of various parts of an image, such as edges, contours, and textures let alone more complex features, such as faces [1]. Computer vision, the use of computers to analyse and process images, often takes inspiration from actual physiological processes in the brain. One issue with this is that a significant amount of the visual processing pipeline is not well understood - effectively, parts are a "black box" where we know what the input and output are but not the internal workings. As such, developing models that replicate the input-output behavior can help us further understand how physiological processes might work as well as give clues on how to improve computer vision models.

One key technology well suited to modelling how the brain might work is a neural network - after all, they are based on the underlying biology [2]. They are powerful machine learning tools that model artificial "neurons" and there connections. "Learning" is achieved by changing the mathematical parameters that control the connections between neurons, such as weights (whether a connection is inhibitory or excitatory and by how much so) and biases (what threshold is required for activation). Passing in a large set of known outputs for a given input causes slight adjustments to these parameters over time, and eventually, the network algorithm will be trained such that it can achieve a high degree of classification accuracy for a given unknown input. For certain applications, such as audio or visual processing, more advanced neural network types can be used, which have further steps that can improve their accuracy. CNNs in particular are widely used for image classification and feature an additional convolution step that effectively allows them to segment images into smaller components. They are the most useful type for modelling brain activity.

We are particularly interested in how texture detection works in the brain, as detecting texture is done almost instantaneously in day-to-day life but can be difficult for computer vision systems; the physiological basis is not well understood. We are therefore interested in developing a CNN that can be used to detect texture in an image and see what that might imply in terms of the brain itself, linking it back to the known underlying biology. We are also interested in simplifying our model as much as possible to avoid overfitting and to make sure that it models simple biology. We will also compare our generated network against pre-existing models and investigate similarities or differences. We will be defining a visual "texture" as a 2D array of Gabor patches, either containing a signal or not, based on each column sharing an orientation. Noise will be added to this texture to ensure our model is noise-resistant to model the brain's response more closely.

2. Literature Review

This literature review outlines the background context of biological texture detection, current models for discriminating between visual textures, and the current state of research into CNNs and how they could be applied to model physiological neural systems.

2.1 Biological Background of Texture Detection

The visual cortex is a highly organised system comprised of layers of neurons that each have a specific task in visual processing, with more complex tasks such as motion or face recognition performed in extrastriate areas [1]. Different areas on V1 correspond to specific parts of the visual field, effectively projecting the retina onto V1, forming a "retinotopic map" [3]. V1 cells are very dense and have small receptive fields as they perform the first step of integration of visual information into the brain. They are highly selective for orientation and contrast, which are key for the broad detection of texture, and can be classified as either simple or complex cells. Both types are orientation specific, but complex cells respond to large receptive fields instead of an exact mapping and have additional responses to object movement [4]. We expect simple cells will be of primary importance in developing models capable of detecting texture. The receptive field of orientation-specific V1 simple cells is longer in one direction than another, and as such, if an input edge aligns within a certain threshold to the receptive field, a response will be produced. Each grouping of simple cells that forms part of a visual receptive field is only activated by one texture - e.g., a simple cell may respond to a white line on a black background but not vice versa [1].

This sensitivity of the simple cells allows for the development of artificial visual stimuli that will maximally activate the simple cells. Gabor patches, produced via convolution of a Gaussian function and a sinusoid, match the shape of the receptive field and have the contrast between the on and off regions required for activation [5]. Polat found that the intensity of response was increased via "lateral masking", especially under lower contrast conditions, where Gabors sharing the same orientation are placed along a single line [6]. This increased intensity corresponds to the detection of an edge, as an edge is defined by a constant region of uniform orientation.

However, in real-world conditions, the majority of edges that humans perceive are not perfectly straight. For example, a tree trunk has regions of different orientations against a background, yet humans can easily distinguish the overall orientation of the tree. Noise can be added to a model of texture to model for this, such that a signal with a certain orientation has deviating areas of noise. Hussain & Bennet used this approach - they added Gaussian noise to texture patterns that allowed them to investigate improvements in subject's abilities to detect whether a texture or not exists in a given stimulus image [7]. This detection of textures is almost spontaneous and can occur even without the subject's attention [8]. The physiological basis for this ability to integrate output from the simple cells into detecting overall orientation whilst filtering out noise is not well understood however and is of interest for further research.

From this review of the biological components of texture detection, we gain some insight into the key components and steps of discrimination: the use of units with small receptive fields to detect orientation, and that it results in fast segregation of images into groups of similar textures with significant noise filtering. However, the connection on how orientation-detecting cells work to create segmentation is not well understood and could give clues into how an artifical network to detect texture could

be formed [9]. In terms of neural network architecture, this could suggest that we want to use an input layer with a large number of neurons to model simple cells and that we may want to first develop a segmentation model to find edges before considering the overall texture within an image.

2.2 Non-Machine Learning Based Approaches

A significant amount of work has gone into developing models utilising the Fourier transform and filters to distinguish between textures in idealised test samples. Previous work into these models will be outlined here, as well as what clues they provide into the underlying biology and what similarities they might share with a neural network based approach.

2.2.1 Fourier Analysis

Spatial frequency is a key component of biological texture detection, as certain populations of visual cortex cells specifically respond to certain spatial frequencies. Mayhew & Frisby suggested that V1 cells could effectively perform a 2D Fourier transform on an image and then use this output for further processing [10]. They found that subjects were able to distinguish between differing spatial frequencies much better than different orientations at the same spatial frequenices which could not be fully explained by a Fourier based model given that all sample textures were composed of pure sinusoids. Nevertheless, Fourier analysis could give some insight into the underlying spatial frequency discrimination observed.

As such, there has been significant research into the use of Fourier analysis to create models capable of discriminating between visual textures. Harvey & Gervais pioneered this work, where they developed a four channel model using 2D Fourier transforms, with each sensitive to a certain spatial frequency or texture [11]. Textures would be discriminated against by combining the output of each channel together. They compared output from the 4-channel Fourier-based system to results from human subjects in a task where textures were to be put into groups of similar textures. They found a strong relationship between the percieved similarity of the textures from humans to that of the model. Other early approaches included the use of exponential or Butterworth filters [12], but these did not respond all as well as the Fourier filter [11].

Hence, we can deduce that segmentation of spatial frequencies could be important in texture detection and that using spatial frequency specific channels and then recombining them later could be a useful approach. This is an analogue to using hidden layers in a CNN, which integrate from the input neurons (i.e. receptive fields) and process specific spatial frequencies in that layer.

2.2.2 Filter-Rectify-Filter Models

Another approach used to simulate visual texture detection is FRF models or the back-pocket model as described by Landy [13]. In this model, the first filter is selective for a certain orientation and spatial frequency, similar to simple cells in the brain. This filter responds maximally positive when the alignment and spatial frequency match completely and minimally negative when they do not match at all. This is then rectified in order to make all filter results positive, which can be achieved by squaring or another nonlinearity, such as full-wave rectification [14]. Finally, a second filter is applied to the rectifier output, which has a much larger receptive field and spatial frequency and

provides the final textural output. In order to discriminate between which texture might be present within an image, multiple channels are used, each with a linear filter tuned differently, then results from each channel are combined similarly to the Fourier-based approach.

This filter-rectify-filter model is thought to have some parallels with actual physiology as cells exhibiting similar input-output behaviour to the second filter in the FRF model have been isolated in the monkey visual cortex [15]. Kingdom et al. sought to investigate the validity of FRF models by using them on texture gratings, either contrast modulated, orientation modulated or spatial frequency modulated as a model for the types of textures that could be found in an image [14]. Using a masking approach where types of modulation were used together, and the response measured, they found that the addition of another type of modulation in the texture did not change the model output, with the exception of contrast modulated masks. This therefore implies that for spatial frequency and orientation modulated textures, independent FRF models are involved in the detection of these elements within a texture. Thus, we learn that our developed CNN might have separate pathways for detecting orientation modulated and spatial-frequency modulated textures.

2.3 Machine Learning Based Approaches

While non-machine learning based models can have similar input-output characteristics to what we expect in physiology, a neural network would still be the best approach to better understand how neurons are connected. CNNs in particular model this behaviour well as the input layer is comprised of receptive fields, with small regions of overlap - effectively, a retinotopic map. The very design of CNNs is based upon our knowledge of underlying biology [16]; hence they are the perfect tool for this application.

2.3.1 CNN Layers

Broadly, CNN hidden layers will be one of three types. Convolutional layers apply convolution to their input, which is limited to a certain receptive field. This reduces the computational complexity as each receptive field region has a smaller number of weights. Convolutional layers generate feature maps, which represent actual features in an image, such as edges or contours [16]. Pooling layers act to reduce the resolution of the output of previous layers to reduce noise while keeping key information from feature maps. Small kernels are used as part of pooling which condense for example 2x2 region of neurons into a single output through taking the average or maximum of values in that kernel [17]. Finally, in fully connected layers each neuron is connected to every other neuron. While computationally expensive, fully connected layers allow the condensation of information from each neuron into a single vector, allowing for overall classfication of an input image. Hence, fully connected layers generally are the last layers in a CNN architecture.

2.3.2 Machine Learning Algorithms

Another key component of CNNs is the machine learning component - that is, how the model is trained. The most common machine learning algorithm is the back-propagation algorithm [18]. This algorithm is based upon the idea that the error in the output of a neuron is the difference between the expected output and the actual output. This error is then propagated back through the network, which is then used to update the weights of the neurons. This process is repeated until the error is sufficiently small [18]. Thanks

to libraries such as TensorFlow and Keras, the learning process can be automated on modern GPU hardware [19]. Given that we will be collecting or otherwise generating a test dataset with input images with texture presence or absence classified by humans, our task falls under the realm of supervised learning [20]. One issue with supervised learning approaches is bias in the training set. However, we do not expect bias to be an issue in our case because the V1 visual processing pathway is the same across all humans and as such should have similar texture detecting ability.

2.3.3 CNN Architectures

Significant work has gone into refining and developing different types of CNNs. Broadly, they can either be classified as shallow or deep neural networks, which depends on the amount of hidden layers in the network. Deep CNNs are more computationally expensive due to the number of layers, suggests that they can fit complex functions better than shallow networks [21]. Deep neural networks are specifically well suited to object recognition tasks, such as object classification, such as the AlexNet architecture which was used to win the ImageNet Large Scale Vision Recognition challenge in 2012 [22]. Zhuang et al. suggested that deep neural networks can be used to model the entire ventral stream of the brain, responsible for object recognition [23]. However, given that we are only interested in texture detection in an idealised image of Gabors, a model that could fit the entire ventral visual processing pathway would not be appropriate. While segmentation of an image is also important in texture detection, and networks exist specialising in biomedical texture segmentation such as U-Net, we believe that a simpler approach using a basic shallow CNN will allow us to more accurately model underlying biology.

2.3.4 Previous Applications of CNNs For Visual System Modelling

By design, CNNs provide a good model for the visual system, with architectures such as AlexNet providing extremely high accuracy at classifying images into one of thousands of categories. The major issue when using CNNs for this approach was the computational load and size of the training set required - over a million images were used in the training of AlexNet in it's debut in the ImageNet classfication challenge [24]. However, texture detection is a much simpler task than classifying an entire image - the response to whether a texture or not exists in a given dataset of Gabors is binary, and the number of images used in the training set will be comparatively small. The majority of the literature on using CNNs to model the visual systems has been focused on the whole cortex level rather than focusing on a specific element such as texture detection. Tripp used a deep system to investigate the overall architecture from a macro level, but this does not give much information as to how texture specifically is detected and modelled [25], wheras Kociolek et al did use CNNs for modelling texture directionality but did not consider what their generated network might suggest about V1 architecture [26]. As such, using a simple CNN architecture to model the texture-discriminating ability alone is a novel research interest.

3. Model and Analytics Development

3.1 Data and CNN Structure

3.1.1 Gabor Patch Datasets

We used three types of datasets: small (106x106), medium (128x128) and large (150x150), which contained an a 6x6, 8x8, or 10x10 array of Gabor patches on a grey background. Different sized datasets were used to investigate any potential differences in hyperparameters was required for a larger image with greater texture complexity. A MATLAB script built up these images by creating a Gabor patch in each of the grid blocks by applying a ramp function at the input orientation angle and then applying a Gaussian blur grating. In the no signal case, a pure noise orientation array was passed into the function for creating Gabor patches, wheras in the signal case noise was added to a pure signal image.

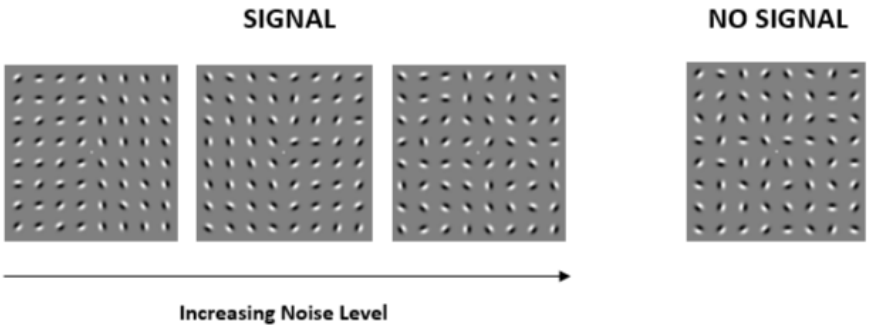

Figure 1 Increasing levels of noise added to a signal image, compared to a purely noise image.

The amount of noise chosen to be added to a signal image was chosen by trial and error, with the maximum amount of noise chosen that the subject could just distinguish between the signal and no signal images accurately.

2,500 images were generated this way for each dataset, with an even split between signal and no signal, and labeled with their ground truth response - whether a signal or not was present. This allowed us to train a CNN against the ground truth and compare the structure of that CNN to one trained on user responses of if there was a signal present or not. User generated data was also collected through a MATLAB script, which flashed the test image quickly for 0.5s in between two gray images, then the user was asked to report whether they thought the signal was present or not in the image. Quickly flashing the images was done in order to prevent the user from using eye movements or any higher-level reasoning to detect a texture to improve accuracy, thus ensuring that the subject's response relies on V1 activity alone. These images were labelled with the user response, with the actual presence or not also recorded, allowing for a confusion matrix to be produced and to track false positives and negatives from both the user and our CNN.

3.1.2 CNN Implementation

In order to implement our CNN, we used the TensorFlow library as it has better visualisation features than other deep learning tools like PyTorch. We also found that TensorFlow was generally chosen more often in the literature. Datasets were created using the image files generated in MATLAB, and rescaled to appropriate sizes to reduce

training time.

We initially intended to generate two CNNs - one trained upon the ground truth signal presence data and one trained upon user detected presence of signal or not. This user generated data was obtained by one of our research members labeling the images as described above. However, we found that the same CNN architecture worked well for both the user generated and the ground truth dataset, leading further models to be trained exclusively against subject data. While we will eventually want to get a wider range of user data in order to reduce any biases, we expected that creating training data from only one individual would not affect the way we would need to architect our CNN model. We initially chose a 80%/20% training/validation split as reccomended in the TensorFlow API documentation but further on in our research we switched to using 10-fold cross validation, as it ensured that all our data would be used during training as well as ensuring that our model is consistent by making sure the accuracy is similar across all folds.

3.1.3 Initial CNN Structure

We started investigating potential CNNs with a shallow architecture, which we expected to best model the visual systems because the fitted parameters are easily inspected and the function of the network can be better understood. Additionally, shallow networks train faster, allowing us to more quickly identify the effect of varying hyperparameters in our network. We started off with a network with two convolutional layers, then a fully connected dense layer connected to a single output neuron with sigmoidal activation. Convolutional layers used a 3x3 kernel with the ReLU activation function to introduce a nonlinearity. The Adam optimiser was used as it was the reccomended default optimiser according to TensorFlow API documentation. The binary crossentropy loss function was chosen as we are performing a binary classification task: either signal or no signal. Our script ran our models for up to 350 epochs, however we implemented a callback to stop training if loss values plateaued. This initial CNN resulted in 94.6±1.3% validation accuracy after 61 epochs on the medium dataset - a promising result, but in order to improve accuracy and investigate what might model biology better we made significant changes to our hyperparameters, resulting in the training of over 100 networks as outlined in Appendix A.

3.1.4 Network Modifications

We varied network hyperparameters such as the number of neurons of each layer, layer number, and kernel sizes in order to compare the effects on performance. One issue we ran into was that our model was overfitting, as sometimes around 20 to 50 epochs accuracy was exceeding 95% but validation accuracy had plateaud at 82%. Adding 25% dropout layer right beore the sigmoid classifier helped as in a dropout layer some of the connection weights are randomly set to zero to reduce overfitting, by reducing a "over-reliance" on a few neurons in a certain layer.

We also used data augmentation by flipping our images horizontally and vertically. This helps to reduce overfitting by making sure our model is more generalised to a wider variety of training data, and effectively increases the size of the training set without requiring further user data to be collected. Activation functions were also varied, with us trying Leaky ReLU, eLU, seLU, tanh, expontential and PReLU functions as outlined in Appendix A; however none of these activation functions gave a better result than ReLU.

3.2 Visualisation and Analysis

In order to dig into how our CNN actually was able to detect a texture, several analysis and visualisation scripts were developed. Of particular interest was looking at the convolutional kernels themselves, and seeing how they worked against both signal and no signal images. As we know that simple cells are orientation specific with particular receptive fields, we wanted to investigate if our trained CNN contained orientation sensitive kernels, which would effectively be models of receptive fields as they are sensitive to a certain area and a orientation. We would expect orientation-sensitive convolutional layer outputs to "light up" by producing high outputs if the orientation of the Gabor matches the kernel orientatation, or a low output otherwise.

Additionally, we wanted to stress-test our models by seeing how it would perform on potential unfamiliar texture boundaries. All our training data used a vertical texture boundary horizontally centered in the Gabor patch array. However, a real-world texture boundary can be in any orientation or position. We generated Gabor arrays with horizontal, vertical, and offset boundaries to see whether our models trained on purely vertical boundary data is robust and detects any texture boundary. This could provide insight into how the texture-detecting kernels actually work to determine a classfiication, as well as whether seperate regions of neurons in the brain are responsible for different orientated texture boundaries or if they use the same orientation-specific receptive fields. As such, a script was created to run a previously trained model on an entirely new set of images to obtain an accuracy metric.

4. Model Analysis & Results

4.1 Initial Model Results

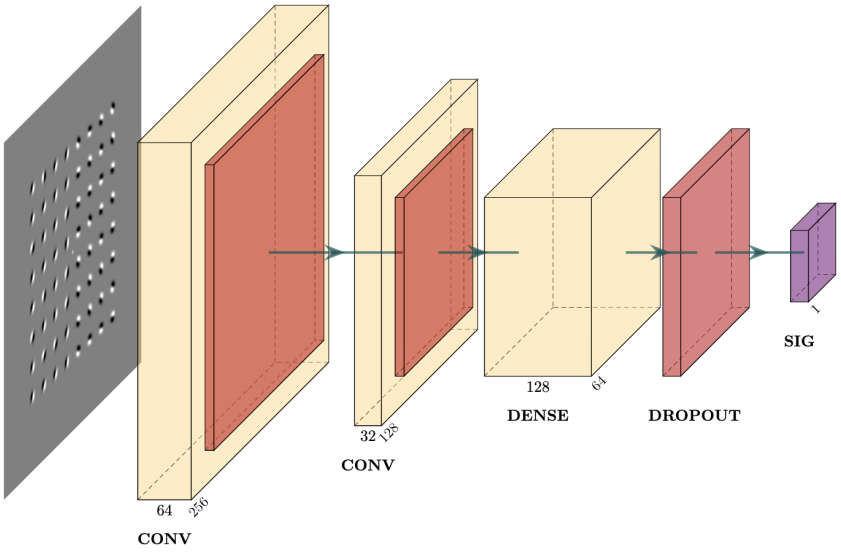

Figure 2 Initial shallow CNN architecture used to classify both user generated and ground truth data. Convolutional layers used a 3x3 kernel with ReLU activation, and the dropout was set to 25%.

Our initial model was based on previous research including that by Kociolek et al. and sample models from the TensorFlow API documentation [19] [26]. Originally, the model did not use a dropout layer or data augmentation, and strong overfitting occured as seen on the left of Figure 3. The model used two convolutional layers to extract features, with a final dense layer before a sigmoid classification. Without data augmentation or

dropout, our model achieved only 89.0% accuracy but after adding these our model gained 95.6% accuracy.

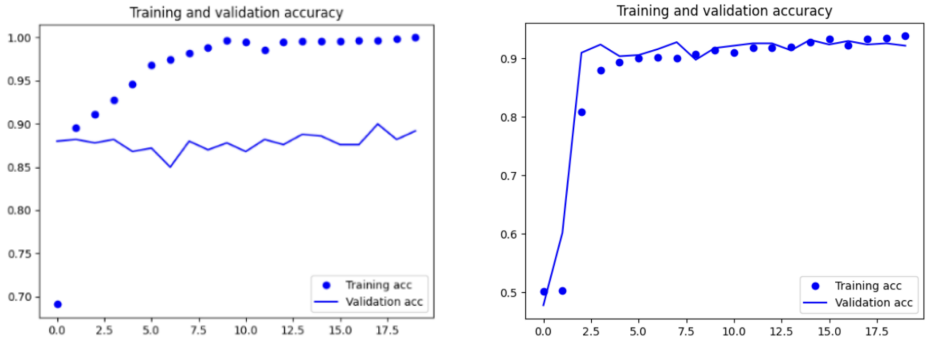

Figure 3 Training and validation accuracy over epochs of our initial model before and after adding dropout layers and using data augmentation.

Table 1 Subject classification

		Ground truth	
		No Signal	Signal
Prediction	No Signal	84.2%	15.8%
	Signal	13.5%	86.5%

Table 2 CNN classification

		Ground truth	
		No Signal	Signal
Prediction	No Signal	95.2%	4.8%
	Signal	5.6%	94.4%

Table 3 Confusion matrices for classification by the subject and classification from our CNN

We found our network was better at classifying data labelled by a subject than the ground truth data source, with our networks identifying the ground truth correctly about 10% more often than identifying the classification given by a subject as seen in Table 3. One reason this could be is that at the chosen noise added to a signal could still completely scramble the image as a result of random chance. This puts a upper bound on the accuracy of our network as the signal has been erased by the noise. However, when a human subject classifies the same image it is likely that they would class it as having no signal, and as our CNN is meant to model human visual cortex behaviour it will also generally classify it as having no signal. As such, we conclude that there will be no expected difference in the models required against ground truth versus subject classification data. All further models will be trained on classification results from a human subject.

While the results from this initial model are promising, we needed to find the optimal kernel size, convolutional layer number and neuron number in each layer to increase model accuracy. A model with higher accuracy could be indicative of a model that works more similarly to V1.

4.2 Hyperparameter Analysis

4.2.1 Convolutional Layer Number

Varying the number of convolutional layers in our model allows for investigation in what sort of higher-order processing is needed to extract a texture boundary. In a network

with a single convolutional layer, the only information that can be extracted are the shapes, such as edges, selected by each convolutional kernel in the layer. In a network with multiple convolutional layers, the information extracted by each layer can be used as input to the next layer, allowing for more complex features to be extracted. For example, a first layer may extract edges, and a second layer may extract shapes from the edges. As such, we would expect increased accuracy with increased convolutional layer number as a texture boundary is a complex feature that could require high-level integration to distinguish.

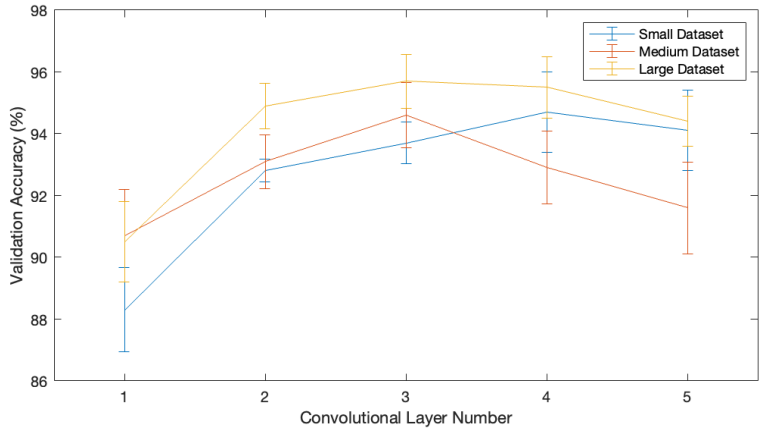

Figure 4 Validation accuracies obtained when varying the convolutional layer number in the network. Neuron numbers at each layer were 32, 16, 8, 8, 8. Kernel sizes were 13, 3, 3, 3, 3.

We found that increasing convolutional layers increased accuracy up to three convolutional layers. Adding further layers past this point reduced accuracy as the original input image becomes too convoluted, with the network trying to extract information that isn't there. Biologically, texture is thought to be a second-order integration of orientation angles, with a texture a defined region of differing orientation angles [27]. As such, we determined that two layers is ideal for this application as it is most consistent with our biological understanding and the accuracy difference between two and three layers can be explained by variation in the data.

4.2.2 Convolutional Filter Number

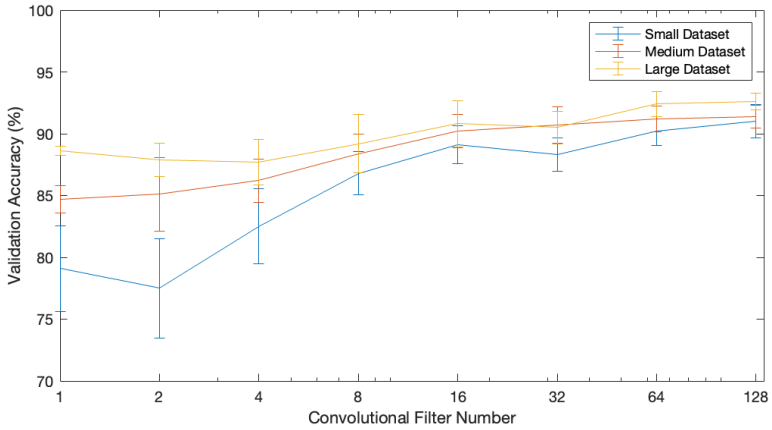

Figure 5 Validation accuracies obtained when varying the number of convolutional filters in the network. A single-layer network was used with kernel size 11.

As expected, increasing the number of convolutional filters in our network gave slight

increases to accuracy. More filters means that more information can potentially be learned by the network at the cost of computation time, however this effect plateaus over time, reaching a maximum validation accuracy with a single-layer 11x11 kernel model of about 91%. However, we found that the convolutional filter number could also be signfiicantly reduced and a single layer network could still discriminate a texture correctly about 85% of the time. We initially did not look at small filter models because we did not expect them to be able to pick up a texture boundary because with only 4 or less filters, no orientation specificity can be fit. However, once we looked at our convolutional filter as outlined in 4.3 we found that many neurons were actually not responding to any stimulus and appeared to have trained as fully noise. This suggests that the network was able to learn a texture boundary with only a few filters, and that not all the filters were actually being used discriminate a texture boundary. This result is further discussed in Section 4.3.

4.2.3 Kernel Size

In the majority of CNN applications, which involve processing normal images or photographs containing objects or other items of interest, smaller kernels such as 3x3 or 5x5 are preferable. However as can be seen below, these kernel sizes on a single convolutional layer with 32 neurons resulted in almost zero ability to discriminate a texture.

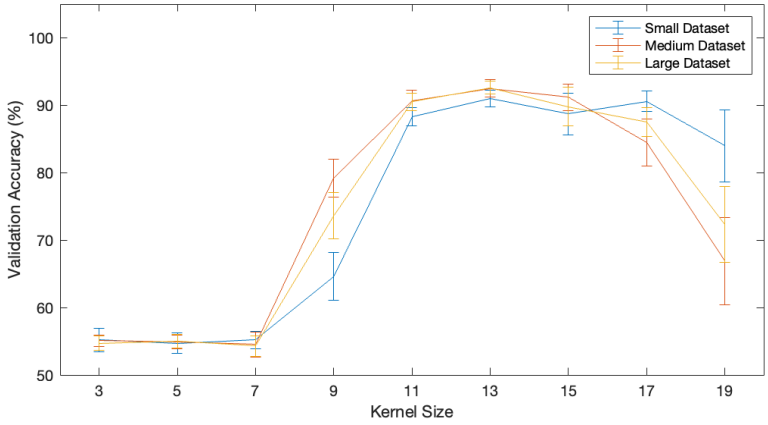

Figure 6 Validation accuracies obtained when varying the kernel size in the network. A single-layer network was used with 32 neurons in the convolutional layer.

When comparing a 13x13 kernel, the size that provided the highest accuracy, to our images as seen in Figure 7, it is clear that this kernel size is able to extract data from multiple Gabors. A 11x11 kernel can also just do this, and had a similar if slightly worse accuracy. However, the 9x9 and 7x7 kernels are unable to span multiple kernels and are unable to produce a texture capable of texture discrimination. This is due to the model used in this exploration using only a single convolutional layer - the inital model fitted had 94.6% accuracy with a 3x3 kernel size, providing more evidence the second convolutional layer is key to integrating information from the first set of convolutional filters. This concept will be further explored in Section 4.3.

Figure 7 Comparison of kernel sizes relative to a Gabor patch array

4.2.4 Links Between Hyperparameters

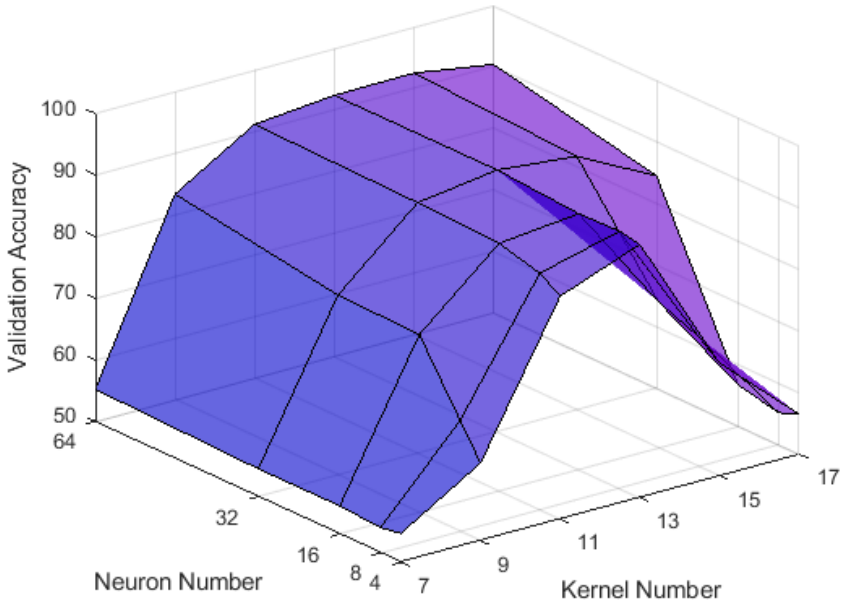

Figure 8 Surface plot showing validation accuracies obtained on a single convolutional layer network with varying neuron number and kernel size.

Figure 8 shows that while increasing neuron number seems to always increase accuracy up to a plateau, kernel size appears to have an ideal value of 13x13, as with a low number of neurons futher kernel sizes result in a decrease in accuracy. However, we also observed that this effect can be mitigated by increasing the number of filters. This is a waste of computational time though because both larger kernel size and filter number rapidly increases the amount of kernel weights that need to be set.

4.3 Kernel Analysis

For the purposes of this section, the "multi layer model" refers to a model with first convolutional layer: 32 filters, 13x13 kernel, second convolutional layer: 16 filters, 3x3 kernel - this achieved 93.2±1.1% validation accuracy. While not the highest accuracy model trained, this model was chosen as it was representative of the general trends in these models and the smaller filter number aids visualisation. "Single layer model" refers to the variety of models trained with only a single convolutional layer with a 13x13 kernel, with a wide range of convolutional filter numbers tested. Full makeup and accuracy of these models is outlined in Appendix A. All models mentioned in this section refer to the models trained on the medium dataset, as we did not see any differences in best network across the three datasets.

4.3.1 Kernel & Filter Output Visualisation - Multi Layer Models

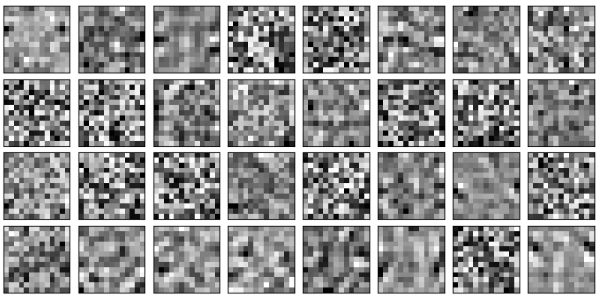

Figure 9 Convolutional filters fitted in the first layer of a multilayered model

Using a script that allowed us to view the convolutional kernels, we looked at the fitted kernels of the first layer of the mulitlayer model discussed above. We expected to see orientation specific kernels with a parallel light and dark line at a specific orientation angle, but this was not the case. Instead, several kernels appeared to have fitted almost fully noise as a result of the training process, and other kernels fitted a defined yet difficult to discern pattern. The output after convolving these filters with signal and no signal images was investigated to figure out what the function of these kernels may be; selected kernels and their convolution result is shown in Figure 10.

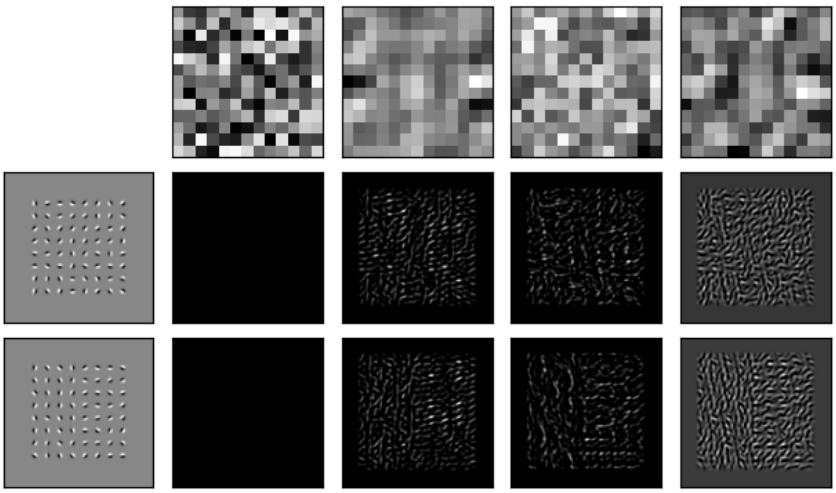

Figure 10 Selected convolutional filters and filter output of the first layer in a multilayered model

As expected, convolutional kernels with no discernable pattern produced zero output on both a signal and no signal image and could essentially be discared as an artifact of the training process. No kernel seemed to respond to only a specfic orientation angle, and instead a large number of kernels effectively "joined" the Gabor patches together, making the texture boundary more obvious. This is most apparent on the rightmost kernel, which highlights the texture boundary on the signal image. Hence, it follows that this highlighted boundary is then used by the second layer in order to determine the presence or abcense of the signal. This is similar to the second filtering in the FRF model, where global texture orientation is determined with a large receptive field, except in the reverse case. However, the kernels fitted in this mulitlayer model do not seem to be consistent with biology due to the lack of orientation specificity.

Given the similarity between this model and a FRF order with reversed filter order, we decided to investigate the validity and kernels of a multilayer CNN with a 32-neuron 5x5 layer into a 16-neuron 13x13 layer. This network achieved a validation

accuracy of 95.1%, and orientation specificity was observed in a signficant number of the convolutional kernels in the first layer as shown below. By testing the kernels on a array of Gabors each of differing orientatation, orientation specificity can easily be visualised; e.g. the leftmost kernel responds most to vertical stimulus as seen in the top row of Figure 11.

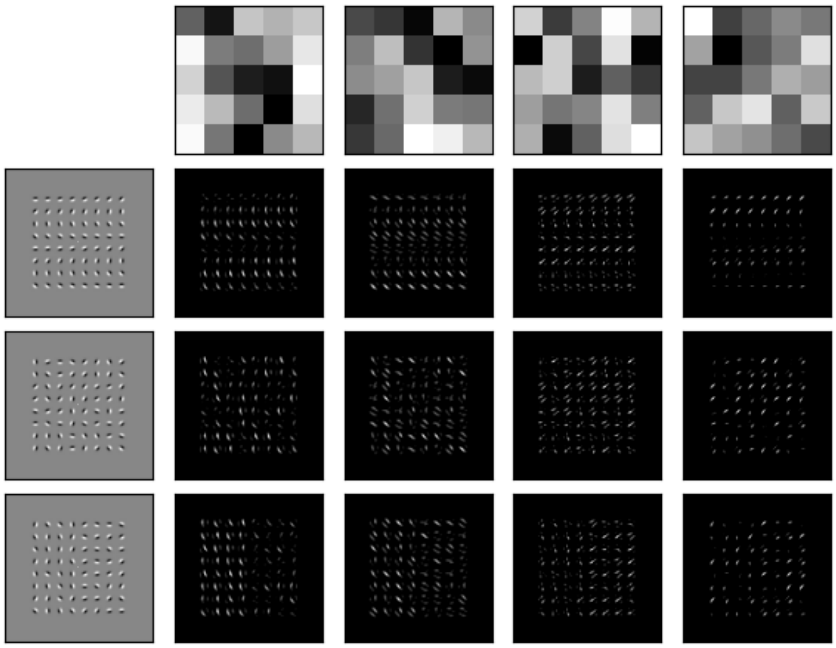

Figure 11 Selected convolutional kernels of the first layer of the inverse multilayer model, and the result when convolved against varying angle Gabor patches, no signal, and signal images.

As orientation specificity with small kernels was trained (i.e. small receptive fields), these provide a good model for simple cells in the brain. The second layer with larger receptive field kernels that performs the overall texture discrimination models complex cells; therefore, this model seems plausible for modelling the biological texture discrimination process in the brain. This is further supported by this model having one of the highest validation accuracies out of any model we investigated.

4.3.2 Kernel & Filter Output Visualisation - Single Layer Models

Given that we could achieve accuracies of up to 90% with one to four convolutional filters in a single-layer model, we were interested in figuring out exactly how such a small number of filters could discriminate a texture. As with multilayer models, we visualised the filters and the convolutional layer output to figure out how our model worked. The 32-neuron model contained a large number of noise kernels that produced blank output, and progressively we reduced the number of neurons in our model all the way down to a single convolutional filter.

Both the 2 and 4 neuron models fit some noise filters; hence the non-noise filter must be doing all the classification work. This is further confirmed by that only one or two of the filters have any output at all from the convolutional layer. We also observe that all three of these models fit a defined shape that is difficult to describe but similar across each model. With kernel size of 13, this filter is able to "view" multiple Gabor patches at once and hence we suspect it identifies a texture by filtering for an orientation difference between the Gabor patches the kernel rolls over. In the signal case, the total amount of high outputs is greater, which is then directly used to make the classification. One

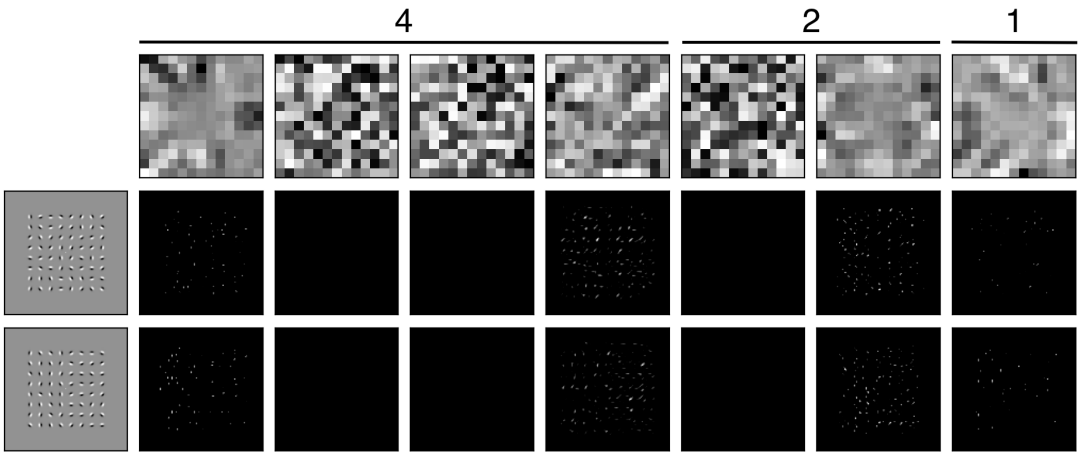

Figure 12 Convolutional filter kernels & layer output of 1, 2 and 4 neuron single-layer models.

biological pathway this network type could model is complex cells that are connected directly to the thalamus. Complex cells usually take input from simple cells, but can also be connected directly to output from the lateral genticulate nucleus in the thalamus [4]. The large 13x13 receptive field used that integrates across multiple Gabors is consistent with complex cell receptive field properties. Henceforth this model, while unlikely to be representative of the entire texture-discriminating ability of V1 could be part of a parallel processing pathway in the visual systems which bypasses simple cells directly [28], in conjunction with a multilayer model previously discussed.

4.4 Model Robustness On Unfamiliar Data

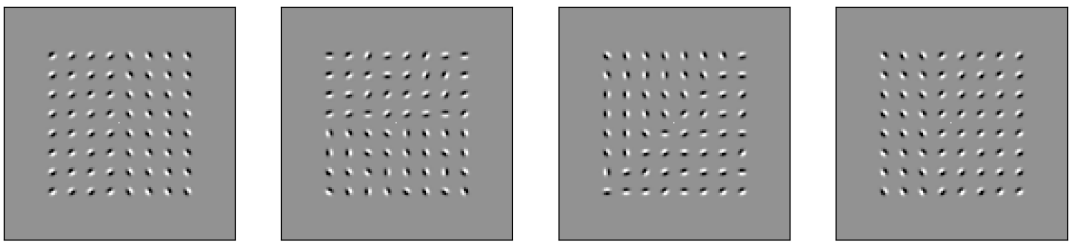

Figure 13 Vertical (training), horizontal, diagonal, and offset (left-to-right) boundaries used as unfamiliar texture boundaries for validating model robustness.

Every model that we created was trained on the same type of texture boundary: a vertical boundary horizontally centered in the Gabor array. A valid model of the visual system would be able to detect a texture boundary at any point in the image. As such, images were generated with signals in horizontal as well as offset from the centereline. Using Tensorflow `model.evaluate`, we were able to generate accuracy metrics for these unfamiliar datatypes to see what model types might or might not be able to detect unfamiliar textures.

Interestingly, all of these models achieved 85% accuracy on the majority of the unfamiliar datasets. While accuracy was reduced, the fact that all of these models had some degree of accuracy reinforces the validity and robustness of these models. In particular, the inverse multilayer model had the highest overall validation accuracy on familiar data and the least accuracy loss on the unfamiliar datasets, further supporting the potential validity of this model.

Network	Validation	Horizontal	Diagonal	Offset
Multilayer 32 (13x13) → 16 (3x3)	93.2%	87.4%	83.8%	85.0%
Multilayer 32 (5x5) → 16 (13x13)	95.1%	90.2%	93.2%	88.4%
Single layer 32 (13x13)	92.5%	89.0%	87.2%	89.0%
Single layer 4 (13x12)	91.1%	87.4%	87.8%	88.9%
Single layer 2 (13x13)	89.7%	85.6%	80.2%	84.0%
Single layer 1 (13x13)	90.0%	84.2%	87.0%	85.3%

Table 4 Validation accuracy, horizontal accuracy, diagonal accuracy and offset accuracy of selected CNN models

4.5 Biological Plausibility

Both single layer and multilayer models have links to biological processes in the brain. Kernel size was the most important hyperparameter, with kernels both too small and large able to provide almost any texture discrimination ability. This is consistent with the model of receptive fields, in that for a input of a certain spatial frequency a certain receptive field sensitivity is needed. While our textures were only comprised of a single boundary, the spatial frequency involved in the texture required a receptive field capable of capturing Gabors on both sides of the boundary; the 13x13 kernel size was ideal for our specific data input. If the grey space between Gabors was reduced in size, we would expect a smaller kernel would work better as long as it was able to capture multiple Gabors within it. This finding regarding kernel size is agrees with Kociolek, as their best-performing networks used 13x13 or 17x17 kernel sizes for a similarly sized input image [26].

Based on the overall structure and accuracy of our networks, we deduce the "inverse multilayer" model is the most plausible representation of biological texture discrimination processes. This network, which shares parallels with the FRF model described by Landy [13] and the highest accuracy CNN found by Zhang et al [29]. In particular, orientation sensitive kernels are well established to be excellent models of simple cell behaviour in V1, even down to the choice of activation function used. The ReLU function provides a half-wave rectification, which was used by Rust et al. in a LNP (linear-nonlinear-Poisson) model of V1 responses, providing futher evidence for the validity of smaller orientation sensitive convolutional kernels as simple cell models [30]. Other orientation-specific models, including high neuron number single layer models are unlikely to be biological relevant due to the larger receptive field precluding modelling of simple cells, and orientation specificity is not characteristic of complex cells. That being said, the fact that even a singular large kernel can achieve 89% validation accuracy suggests there may be some merit to this appraoch as a complex cell model. In this model, kernels alone perform top-level texture discrimination integrating across to multiple Gabors. However, this theory is currently in debate amongst the scientific community, with evidence both for the direct input to complex cells versus them only recieving input from simple cells [31]. Complex cells are located throughout all V1 layers, with differing receptive field sizes depending on their location. A new network type that closely models the layers found histologically in the primary visual cortex would be required to gain a more complete understanding of the function of complex cells in a V1 model.

5. Conclusion

Through an exploration into primary visual cortex neuroscience as well as pre-existing models of its function, over one hundred convolutional neural networks were trained and evaluated for their accuracies, critical features, and how they might link back to known biology. We found that convolutional filter kernels are good models of receptive fields and can be used to model both simple and complex cell behaviour in V1. By investigating hyperparameters that provided the best accuracy and analysing the output of our convolutional filters, we identified the most plausible network design to mimic V1 structure and function. This network used two convolutional layers, with smaller orientation-specific convolutional kernels modeling simple cell receptive fields and larger kernels integrating this information together modelling complex cells. 95.1% accuracy was achieved by this network and it'was consistent with non-machine learning based V1 models such as the filter-rectify-filter model.

6. Further Research

Further research interests related to this project include:

- *Gather a more diverse dataset*: All data we used was labelled by the researchers. While we would not expect any difference between individuals or groups that would change model architecture, it would be better scientific practice to have a dataset that minimises bias. Additionally, further research would need to contact the Maori-Pasifika community regarding any potential implications of this research.
- *Investigate the effect of spatial frequency*: Our research focused on the effect of orientation and how it might be modeled by kernels. However, spatial frequency was kept constant for all tests. Varying spatial frequency could provide more insight into the parallel processing involved with K, M and P cells in the visual cortex.
- *Further investigate "inverse mulitlayer" model*: The best model of visual texture that we decided was developed late on into the project after realising the similarity of a multilayer model to FRF. As such, we did not investigate the effects of hyperparameters extensively on this type of model. Further research in this area could identify optimal hyperparameters or investigate the action of the second complex-cell modelling convolutional layer in more detail.

References

[1] D. Purves, "Central visual pathways," in *Neuroscience*, 2019, ch. 12.

[2] R. Eberhart and R. Dobbins, "Early neural network development history: the age of camelot," *IEEE Engineering in Medicine and Biology Magazine*, vol. 9, no. 3, pp. 15–18, 1990.

[3] G. Perry, P. Adjamian, N. J. Thai, I. E. Holliday, A. Hillebrand, and G. R. Barnes, "Retinotopic mapping of the primary visual cortex - a challenge for MEG imaging of the human cortex," *Eur J Neurosci*, vol. 34, no. 4, pp. 652–661, Aug 2011.

[4] Y. Lian, A. Almasi, D. B. Grayden, T. Kameneva, A. N. Burkitt, and H. Meffin, "Learning receptive field properties of complex cells in V1," *PLOS Computational Biology*, vol. 17, no. 3, pp. 1–27, 03 2021.

[5] D. Durrie and P. S. McMinn, "Computer-based primary visual cortex training for treatment of low myopia and early presbyopia," *Trans Am Ophthalmol Soc*, vol. 105, pp. 132–138, 2007.

[6] U. Polat, "Functional architecture of long-range perceptual interactions," *Spat Vis*, vol. 12, no. 2, pp. 143–162, 1999.

[7] Z. Hussain and P. J. Bennett, "Perceptual learning of detection of textures in noise," *Journal of Vision*, vol. 20, p. 22, 2020.

[8] S. P. Heinrich, M. Andrï¿œs, and M. Bach, "Attention and visual texture segregation," *Jouranl of Vision*, vol. 7, p. 6, 2007.

[9] J. Bergen and M. Landy, "Computational modeling of visual texture segregation," in *Computational Models of Visual Processing*, 1991.

[10] J. E. W. Mayhew and J. P. Frisby, "Texture discrimination and fourier analysis in human vision," *Nature*, vol. 275, no. 5679, pp. 438–439, Oct. 1978.

[11] L. O. Harvey and M. J. Gervais, "Visual texture perception and fourier analysis," *Perception and Psychophysics*, vol. 24, pp. 534–542, 1978.

[12] H. Mostafavi and D. Sakrison, "Structure and properties of a single channel in the human visual system," *Vision Research*, vol. 16, no. 9, pp. 957–IN4, 1976.

[13] M. S. Landy, "Texture analysis and perception," in *The New Visual Sciences*, 2013, pp. 639–652.

[14] K. F. A. A., P. Nicolaas, and H. Anthony, "Mechanism independence for texture-modulation detection is consistent with a filter-rectify-filter mechanism," *Visual Neuroscience*, 2003.

[15] E. Peterhans and R. von der Heydt, "Subjective contours–bridging the gap between psychophysics and physiology," *Trends Neurosci*, vol. 14, no. 3, pp. 112–119, Mar 1991.

[16] G. Franchini, , V. Ruggiero, F. Porta, and L. Z. and, "Neural architecture search via standard machine learning methodologies," *Mathematics in Engineering*, vol. 5, no. 1, pp. 1–21, 2022.

[17] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang, "Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study," *Neural Computing and Applications*, vol. 34, no. 7, pp. 5321–5347, 2022.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[19] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[20] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 1310–1315.

[21] S. Chauhan, L. Vig, M. De Filippo De Grazia, M. Corbetta, S. Ahmad, and M. Zorzi, "A comparison of shallow and deep learning methods for predicting cognitive performance of stroke patients from mri lesion images," *Frontiers in Neuroinformatics*, vol. 13, 07 2019.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[23] C. Zhuang, S. Yan, A. Nayebi, M. Schrimpf, M. C. Frank, J. J. DiCarlo, and D. L. K. Yamins, "Unsupervised neural network models of the ventral visual stream," *Proceedings of the National Academy of Sciences*, vol. 118, no. 3, p. e2014196118, 2021. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2014196118

[24] G. W. Lindsay, "Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future," *J Cogn Neurosci*, vol. 33, no. 10, pp. 2017–2031, 09 2021.

[25] B. Tripp, "Approximating the architecture of visual cortex in a convolutional network," *Neural Comput*, vol. 31, no. 8, pp. 1551–1591, Jul. 2019.

[26] M. Kociolek, M. Kozlowski, and A. Cardone, "A convolutional neural networks-based approach for texture directionality detection," *Sensors*, vol. 22, no. 562, p. 562, 01 2022.

[27] H. X. Wang, D. J. Heeger, and M. S. Landy, "Responses to second-order texture modulations undergo surround suppression," *Vision research*, vol. 62, pp. 192–200, 2012.

[28] J. Nassi and E. Callaway, "Parallel processing strategies of the primate visual system," *Nature reviews: Neuroscience*, vol. 10, pp. 360–72, 05 2009.

[29] Y. Zhang, T. Lee, M. Li, F. Liu, and S. Tang, "Convolutional neural network models of v1 responses to complex patterns," *Journal of Computational Neuroscience*, 04 2018.

[30] N. C. Rust, O. Schwartz, J. A. Movshon, and E. P. Simoncelli, "Spatiotemporal elements of macaque v1 receptive fields," *Neuron*, vol. 46, no. 6, pp. 945–956, 2005.

[31] I.-h. Chou *et al.*, "Disentangling simple from complex cells," *Nature neuroscience*, vol. 8, no. 3, pp. 266–266, 2005.

Appendix A Models Trained and Accuracy Metrics

In this appendix, the following abbrevations are used on table headers to conserve space:

- CLN: Convolutional layer number and number of neurons in each layer
- DLN: Dense layer number and number of neurons in each layer
- KS: Kernel size for each convolutional layer
- O: Optimiser
- D: Dropout percentage
- A: Activation function
- VA: Validation accuracy ± standard deviation across folds
- EN: Epoch number

A.1 Large Dataset

All models trained on the large dataset used a 25% dropout layer, the SDG optimiser, with the binary crossentropy loss function and ReLu activation function. All cross-validation was performed with ten folds. No dense layers were tested on large datasets, excluding the final sigmoid classification layer.

CLN	KS	VA	EN
1: 32	3x3	54.7±1.7%	59.3
1: 32	5x5	55.0±1.6%	55
1: 32	7x7	54.3±2.4%	55.5
1: 32	9x9	73.6±5.5%	404.9
1: 32	11x11	90.5±2.1%	336.2
1: 32	13x13	92.6±1.5%	348.5
1: 32	15x15	89.8±4.6%	366.2
1: 32	17x17	87.5±3.4%	356.2
1: 32	19x19	72.3±9.1%	306.7
1: 128	11x11	92.6±1.1%	315.7
1: 64	11x11	92.4±1.6%	319
1:16	11x11	90.8±3.0%	354.9
1:8	11x11	89.2±3.8%	330.44
1:4	11x11	87.7±3.0%	386.7
1:2	11x11	87.9±2.2%	379
1:1	11x11	88.6±0.6%	356.7
2: 32, 16	13, 3	94.8±1.3%	375.8
2: 32, 16	11, 3	94.9±1.2%	364.1
2: 32, 16	9, 3	94.8±0.5%	369.2

2: 32, 16	7, 3	93.9±1.2%	388.5
2: 32, 16	5, 3	90.9±1.3%	381.5
2: 32, 16	3, 3	57.5±2.5%	357
3: 32, 16, 8	11, 3, 3	95.7±1.4%	384.7
4: 32, 16, 8, 8	11, 3, 3, 3	95.5±1.6%	391.3
5: 32, 16, 8, 8, 8	11, 3, 3, 3, 3	94.4±1.3%	356.1

A.2 Medium Dataset

All models trained on the medium dataset used the binary crossentropy loss function, the ReLu activation function and were cross-validated with ten folds.

CLN	DLN	KS	O	D	VA	EN
2: 64, 16	1:128	3x3	Adam	0	95.6±1.3%	61.2
2: 32, 16	1:128	3x3	Adam	0	96.1±1.2%	66.9
2: 16, 16	1:128	3x3	Adam	0	95.8±1.6%	86
2: 8, 16	1:128	3x3	Adam	0	95.4±1.1%	75.4
2. 4, 16	1:128	3x3	Adam	0	94.9±2.3%	62.2
2: 32, 16	1:64	3x3	Adam	0	95.0±1.5%	72.6
2: 32, 16	1:32	3x3	Adam	0	95.2±0.7%	89.6
2: 32, 16	1:16	3x3	Adam	0	95.0±1.4%	84.6
2: 32, 16	1:8	3x3	Adam	0	92.5±2.1%	89.3
1: 64	0	3x3	Adam	0	53.5±1.7%	25.7
1: 64	0	5x5	Adam	0	54.5±2.6%	46
1: 64	0	7x7	Adam	0	53.4±2.6%	23.8
1: 64	0	9x9	Adam	0	51.4±1.6%	21
2: 64, 16	0	3x3	Adam	0	54.0±3%	35.8

CLN	KS	O	D	VA	EN
1: 64	3x3	SDG	0.25	54.5±2.5%	55.7
1: 64	5x5	SDG	0.25	55.2±1.3%	52.3
1: 64	7x7	SDG	0.25	55.2±2.4%	55.8
1: 64	9x9	SDG	0.25	83.4±3.9%	327.6
1: 64	11x11	SDG	0.25	91.2±1.64%	189.5
1: 64	13x13	SDG	0.25	92.4±1.51%	184.5
1: 64	15x15	SDG	0.25	92.5±1.57%	195.2
1: 64	17x17	SDG	0.25	90.4±4.00%	251.8
1: 64	19x19	SDG	0.25	75.0±10.42%	227.4

1: 32	3x3	SDG	0.25	55.1±1.35%	57.8
1: 32	5x5	SDG	0.25	54.9±1.62%	60.7
1: 32	7x7	SDG	0.25	54.5±3.0%	58.8
1: 32	9x9	SDG	0.25	79.2±4.6%	313.1
1: 32	11x11	SDG	0.25	90.7±2.4%	178.5
1: 32	13x13	SDG	0.25	92.5±2.1%	193.4
1: 32	15x15	SDG	0.25	91.2±3.2%	227.1
1: 32	17x17	SDG	0.25	84.5±5.6%	245.3
1: 32	19x19	SDG	0.25	66.9±10.4%	261.3
1: 16	7x7	SDG	0.25	54.6±1.0%	203.6
1: 16	9x9	SDG	0.25	78.9±6.6%	340.5
1: 16	11x11	SDG	0.25	90.2±2.2%	205.5
1: 16	13x13	SDG	0.25	91.5±2.6%	250.3
1: 16	15x15	SDG	0.25	73.7±6.4%	365.3
1: 16	17x17	SDG	0.25	56.7±1.2%	350
1: 8	7x7	SDG	0.25	54.1±1.6%	210.4
1: 8	9x9	SDG	0.25	68.0±8.3%	252.2
1: 8	11x11	SDG	0.25	88.4±2.5%	219
1: 8	13x13	SDG	0.25	91.7±1.6%	263.6
1: 8	15x15	SDG	0.25	71.0±7.8%	200
1: 8	17x17	SDG	0.25	55.2±3.2%	200
1: 4	7x7	SDG	0.25	54.7±1.5%	200
1: 4	9x9	SDG	0.25	62.6±2.8%	218.1
1: 4	11x11	SDG	0.25	86.2±2.8%	218.1
1: 4	13x13	SDG	0.25	91.1±2.0%	256.7
1: 4	15x15	SDG	0.25	68.3±7.2%	242.3
1: 4	17x17	SDG	0.25	56.5±1.0%	211
1:16	11x11	SDG	0.25	90.2±2.2%	205.5
1:8	11x11	SDG	0.25	88.4±2.5%	219
1:4	11x11	SDG	0.25	86.2±2.8%	218.1
1:2	11x11	SDG	0.25	85.1±4.8%	245.9
1:1	11x11	SDG	0.25	84.7±1.8%	242.2
1:16	13x13	SDG	0.25	91.5±2.6%	250.3
1:8	13x13	SDG	0.25	91.7±1.6%	263.6
1:4	13x13	SDG	0.25	91.1±2.0%	256.7
1:2	13x13	SDG	0.25	89.7±1.7%	255
1:1	13x13	SDG	0.25	90.0±1.9%	276.5
2: 32, 16	13, 3	SDG	0.25	93.2±1.1%	182.3
2: 32, 16	11, 3	SDG	0.25	93.1±1.4%	171.4
2: 32, 16	9, 3	SDG	0.25	93.1±1.4%	215.1
2: 32, 16	7, 3	SDG	0.25	92.0±1.7%	230.8

2: 32, 16	5, 3	SDG	0.25	90.1±3.0%	307.8
2: 32, 16	3, 3	SDG	0.25	56.6±2.2%	206.4
2: 64, 16	13, 3	SDG	0.25	94.0±1.6%	214.5
2: 64, 16	11, 3	SDG	0.25	93.5±1.2%	185.8
2: 128, 16	13, 3	SDG	0.25	94.0±1.6%	177.4
3: 32, 16, 8	11, 3, 3	SDG	0.25	94.6±1.7%	181.2
4: 32, 16, 8, 8	11, 3, 3, 3	SDG	0.25	92.9±1.9%	166.9
5: 32, 16, 8, 8, 8	11, 3, 3, 3, 3	SDG	0.25	91.6±2.4%	249.4
1:16	9x9	SDG	0.25	78.9±6.6%	340.5
1:8	9x9	SDG	0.25	68.0±8.3%	252.2
1:4	9x9	SDG	0.25	62.6±4.9%	237.7
1:2	9x9	SDG	0.25	60.4±5.1%	231.4
1:1	9x9	SDG	0.25	57.2±4.2%	202.6

A.3 Small Dataset

In the small dataset, a 25% dropout, the SDG optimiser, the binary crossentropy loss function, and no dense layers were used for all models. Ten-fold cross-validation was used for each model as well. Apart from the final group, the ReLU activation function was used for each model.

CLN	KS	VA	EN
1: 32	3x3	55.2±2.8%	102
1: 32	5x5	54.7±2.4%	100
1: 32	7x7	55.2±2.1%	105
1: 32	9x9	64.6±5.7%	309.1
1: 32	11x11	88.3±2.2%	291.2
1: 32	13x13	91.0±2.0%	305.3
1: 32	15x15	88.7±5.0%	336
1: 32	17x17	90.6±2.5%	373.3
1: 32	19x19	84.0±8.6%	384.3
1: 128	11x11	91.0±2.2%	326.7
1: 64	11x11	90.2±1.9%	345.4
1:16	11x11	89.1±2.5%	326.7
1:8	11x11	86.8±2.8%	346.8
1:4	11x11	82.5±4.9%	330.5
1:2	11x11	77.5±6.5%	380.2
1:1	11x11	79.1±5.6%	371.1
1: 128	13x13	91.5±1.8%	327.1
1: 64	13x13	92.1±1.6%	387.3

1:16	13x13	91.2±2.3%	373.7
1:8	13x13	89.4±3.1%	371.4
1:4	13x13	89.5±2.4%	400.1
1:2	13x13	86.4±4.0%	392.9
1:1	13x13	85.6±3.7%	476.2
2: 32, 16	13, 3	93.8±1.1%	378.2
2: 32, 16	11, 3	92.8±0.6%	381.9
2: 32, 16	9, 3	92.2±1.2%	377.4
2: 32, 16	7, 3	91.3±1.9%	404.8
2: 32, 16	5, 3	88.5±2.4%	501.5
2: 32, 16	3, 3	56.5±2.7%	350.7
3: 32, 16, 8	11, 3, 3	93.7±1.1%	373.2
4: 32, 16, 8, 8	11, 3, 3, 3	94.7±2.1%	381.8
5: 32, 16, 8, 8, 8	11, 3, 3, 3, 3	94.1±2.1%	424.6
1: 1	3x3	55.4±2.8%	350
1: 1	5x5	55.3±2.0%	350
1: 1	7x7	55.5±2.1%	350
1: 1	9x9	55.4±1.2%	361.2
1: 1	11x11	80.7±3.6%	458.6
1: 1	13x13	84.0±2.0%	427
1: 1	15x15	80.4±9.5%	380
1: 1	17x17	86.2±2.4%	428.7
1: 1	19x19	80.3±6.6%	388.3

CLN	KS	A	VA	EN
1: 32	13x13	Leaky ReLU	90.1±2.2%	391.4
1: 32	13x13	eLU	90.8±2.3%	388.2
1: 32	13x13	SeLU	55.7±1.1%	350
1: 32	13x13	tanh	57.1±1.2 %	413
1: 32	13x13	exponential	55.1±1.6%	350.4
1: 32	13x13	PReLU	92.3±1.6%	380.6