THE UNIVERSITY OF AUCKLAND

RESEARCH PROJECT IN MECHATRONICS ENGINEERING

Mid Year Technical Report

Type of Report

A Convolutional neural network for detecting visual texture

Callan Loomes

Project Report ME110-2022

Department of Mechanical and Mechatronics Engineering

The University of Auckland

22th July 2022

DECLARATION

Student

I ..... Callan Loomes ..... 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 2021 academic year at the Department of Mechanical 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 project sponsor 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 This report is not the same as any report, thesis, conference article or journal paper, or any other publication or unpublished work in any format.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: State clearly what has been developed during the project and what was available from previous year(s):

Signature: _____

Date: 22/07/22

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: _____

Table of Contents

Table of Contents **3**

1 Introduction **4**

2 Methods **4**

2.1 Data Generation and Collection 4

2.1.1 Generation of Gabor Patch Sequences 4

2.2 Creating Neural Networks 5

2.2.1 Input Data and Framework 5

2.2.2 Default Shallow Structure 5

2.2.3 Modifications to Default Structure 5

2.2.4 Testing Neuron Count 6

2.2.5 Testing Deeper Networks 7

3 Results **7**

3.1 User Testing on Generated Data 7

3.2 Network Trained on Ground Truth Data 7

3.2.1 Default Shallow Performance 7

3.2.2 Deeper Network Performance 8

3.3 Network Trained on User Data 8

3.3.3 Default Performance and Implications 8

3.3.2 Performance with Different Neuron Quantity 9

4 Future Objectives **9**

4.1 Robust Data Collection 9

4.2 Testing more Network Structures 9

4.3 Linking to Biological Concepts 9

References **10**

1 Introduction

This mid year technical report outlines the work completed thus far in the project. It includes a methods section which describes the creation of testing data, and how convolutional neural networks (CNNs) were created and trained on that data. The performance of these networks is described in the results section, as well as an analysis of human performance on testing data. Furthermore, a future objectives section highlights the direction of the project for the remaining year.

2 Methods

2.1 Data Generation and Collection

2.1.1 Generation of Gabor Patch Sequences

Early testing with pre-existing response data quite quickly highlighted issues of a lack of data to train the model with; this resulted in highly overtrained models and a lack of validation data to test with. Hence, for the purposes of pilot testing and the generation of comprehensive test sets in the future (for human testing), a pipeline for generating test data was created.

Test data consists of a sequence of 8x8 gabor patches placed on a grey background, each at a different orientation. The generated datasets are generated such that half the images contain a pure noise (a completely random orientation of patches), while the other half contains a signal in addition to this noise. The signal is such that patches on the right hand side of the image are shifted 90 degrees out of phase from those on the left side, resulting in a texture down the middle of the image. Within the same side of the signal, some patches are randomly flipped 180 degrees out of phase.

The sequence of patches were generated in MATLAB; for the individual gabor patches, a gaussian blur was applied to a ramp function applied at a given orientation, similar to the process outlined in (Mathot, 2018). For the datasets with no signal, the list or orientations was simply created using a sequence of random complex numbers. For the datasets with a signal, a signal was generated using an offset angle, to which a signal with random noise was added. A randomly generated noise-gain factor was applied to the signal-noise, such that some signal data sets had very little noise, while others had more. An example of these data patches are shown in *Figure 2.1.*

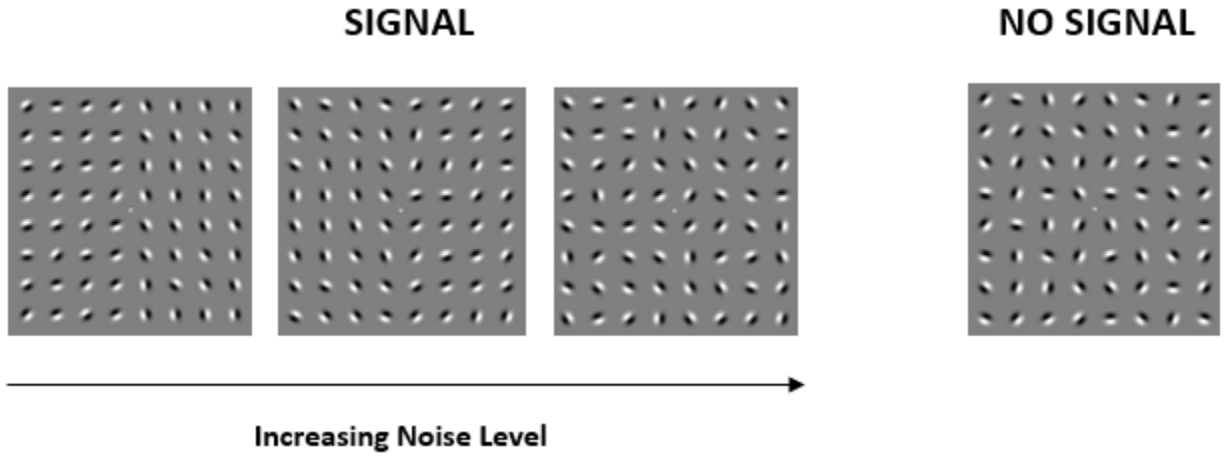

Figure 2.1: *Example gabor patch arrays showing increasing noise in signal and no signal images*

An 8x8 configuration was as it provided a balance between complexity and ease of signal recognition. The noise-gain factor was manually set such that in singal images with the highest amount of noise, the signal was only just detectable by human eyes from the background noise. A test set of 10000 images total was generated using this method; 50% of the images contained a signal while the other

50% contained no signal. The purpose of this dataset was to provide a “ground-truth” trained CNN which compared with user testing data.

2.1.2 Collection of Human Data

In addition to the dataset of 10000 images generated above, a user-tested dataset of 2500 images was also generated. A MATLAB script was used to briefly flash a gabor-patch image for 0.5 seconds to a watching user. The user then inputs whether they thought the image contained a signal or not, and the image is sorted as either a hit, miss, correct rejection (CR) or false affirmation (FA) based on the user response and ground truth. Flashing the image for only a short period of time ensured the human response was mediated only through the low-level V1 processing layer in the brain. Ideally, this limits any higher level decision making on the side of the user, isolating higher level visual layers. Due to time constraints, the testing was performed on only a single user. This means there may be slight inaccuracies and biases in the data, and it's not representative of a larger population.

2.2 Creating Neural Networks

2.2.1 Input Data and Framework

In order to build and test the CNNs for this project, the python package TensorFlow was used, with particular emphasis on the Keras API. In general, TensorFlow and Keras were chosen as they are more flexible and versatile than other packages such as Pytorch (Chirodea et al., 2021). Furthermore, existing literature from which we based our initial structure primarily focused on these packages.

To transfer the images from MATLAB into TensorFlow , the Keras image_dataset_from_directory() package was used. This allowed grayscale images to be loaded from an organised file structure, where all the datasets with a signal were organised into one folder, and all those without a signal organised into another. To speed up training, preprocessing on the images reduced them to 256x256 pixels from their original size of 768x768 pixels; this was performed after testing to ensure the images were still a high quality for user interface. This reduction in size reduced training time by around ~75%, with little to no loss in accuracy of the models. When training the models, an 80%/20% training-validation split was used, as this is recommended in the TensorFlow API documentation .

For this project, two classes of CNN were created; one class of model was trained on the ground truth data, using the fact of whether there was a signal or not as the label. The other class of model was trained using the user responses as the label. The purpose of this was to generate one model which detected the signal as accurately as possible, such that we could compare it with human trials. The other model would be used to mimic a human’s response as closely as possible, allowing further insight into how humans visualise texture.

2.2.2 Default Shallow Structure

When beginning to design CNNs for this project, inspiration was taken from Kociołek et al (2022), in which a similar texture-based problem was resolved using CNNs. Initially, a CNN was generated that would act as a baseline to test network parameters such as kernel size and the number of convolutional layers. This baseline model consisted of two convolutional layers, followed by a dense layer which connected to an single-neuron output layer with a sigmoidal activation function, which would class the images as either having a signal (output of 1) or no signal (output of 0).

The convolutional and dense layers featured ReLU activation layers, and had 64, 32 and 128 neurons respectively. A kernel size of 3x3 pixels was used in both convolutional layers, as this was smaller than an individual gabor patch, allowing the orientation to be determined. Model fitting was generally run for around 20 epochs, as the model would usually start plateauing around this point. This base model was used as a starting point for both the ground-truth data and user-trained data models.

2.2.3 Modifications to Default Structure

Initial testing of the base model revealed that the model had significant issues with overfitting; results from training trials would show that accuracy on the training set was significantly higher on the

validation set as training continued. This indicated that the model was simply “remembering” the testing images, rather than learning the general trends that determined whether there was a signal. To remedy this, a 50% dropout layer was added to the model, which randomly sets a proportion of the connection weights to zero during each training iteration. Furthermore, data augmentation in the form of random vertical and horizontal mirrors were applied to the input data; this transformation did not affect the presence of a signal. This effectively meant the training size was increased by a factor of four, reducing overfitting even further. The structure of this updated network is shown in Figure 2.2.

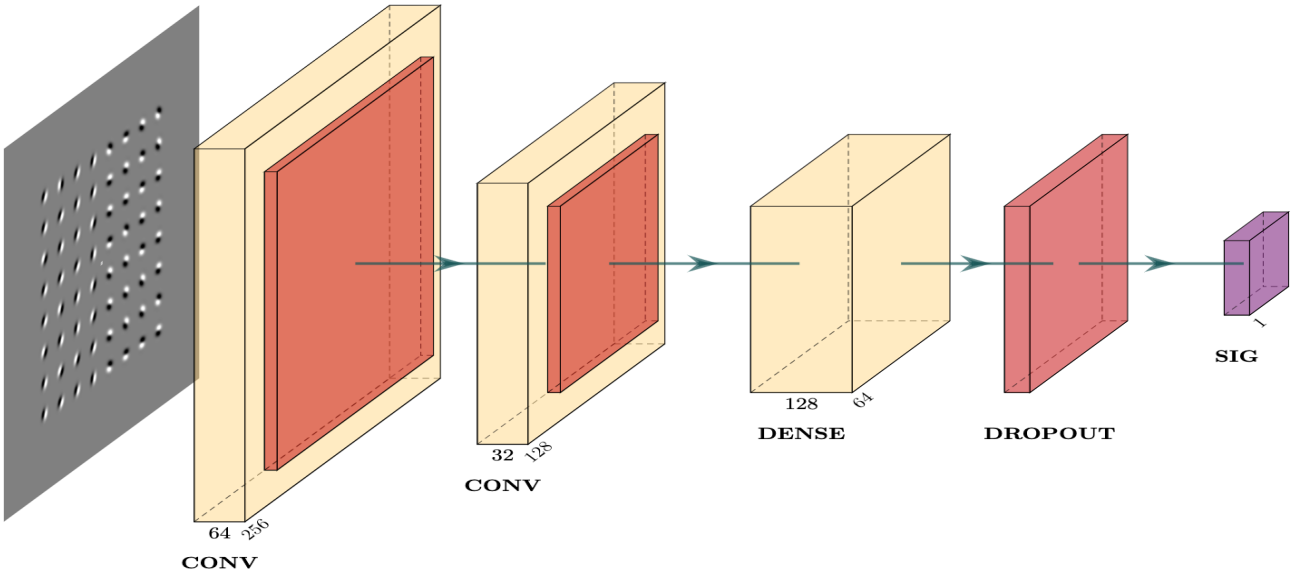

Figure 2.2: *Example gabor patch arrays showing increasing noise in signal and no signal images*

In addition to the anti-overfitting measures, hyperparameters such as learning rate and the optimiser were experimented with. The model trained on user data saw better success using “Adam” as an optimizer with learning rate of 0.001, while the model trained on ground-truth data saw more success using the SDG optimiser with a callback which reduced learning rate as training continued. These adjusted models were used for the structure testing discussed below.

2.2.4 Testing Neuron Count

An indepth exploration into the number of neurons in the first convolutional layer of the user-trained model was performed. The base model has 64 neurons it its first layer, one for each gabor patch. Networks that were exactly the same, but with 32 and 128 neurons in their first convolutional layer were tested in terms of accuracy to determine whether the model could be simplified, or whether it could be improved with more neurons. This could provide insight into the complexity of the V1 layer in the brain. The models were tested using the same random seed for the training/validation split to ensure a fair comparison. A summary of these models is described in *Table 2.1.*

#	Layer Type	Network 32	Network 64	Network 128
1	Input (min size x, min size y, channels)	256, 256, 1	256, 256, 1	256, 256, 1
2	Convolution (size x, size y, count)	3, 3, 32	3, 3, 64	3, 3, 128
3	Max pool	2,2	2,2	2,2
4	Convolution (size x, size y, count)	3, 3, 16	3, 3, 32	3, 3, 32
5	Max pool	2,2	2,2	2,2
6	Flatten			
7	Dense (count)	128	128	128
8	Dropout	0.5	0.5	0.5
9	Output (count)	1	1	1

Table 2.1: *Summary table of models for testing neuron count in first layer*

2.2.5 Testing Deeper Networks

Testing was also performed using models of different complexity/depth. In this case, depth refers to the number of convolution layers in the model. Typically, deeper networks perform better as they are able to solve more complex problems, and are less prone to overfitting (Ososkov & Goncharov, 2017). As a downside however, they are more difficult to train. Furthermore, for some problems, such as in Kociołek et al (2022), deep networks can become too complex, and shallow networks perform better. Models trained on the ground-truth data with one, two and three convolution layers were tested in an attempt to judge the complexity of texture recognition.

3 Results

3.1 User Testing on Generated Data

The performance of the human response to the ground truth data is summarised in *Table 3.1*.

		Ground Truth	
		Signal	**No Signal**
Prediction	**Signal**	1026 (41.4%)	193 (7.8%)
	No Signal	171 (6.9%)	1091 (44.0%)

Table 3.1: *Confusion matrix for user performance on test data*

$$Accuracy = \frac{1026 + 1091}{1026 + 1091 + 193 + 171} \times 100\% = 85.3\% \quad \textbf{(1)}$$

$$Precision = \frac{1026}{1026 + 193} \times 100\% = 84.2\% \quad \textbf{(2)}$$

As calculated in ***(1)*** and ***(2)***, the user was able to achieve an accuracy of 85.3% and a precision of 84.2%. These values appear valid, however the relatively low precision may suggest that the signal-noise gain is slightly too high.

3.2 Network Trained on Ground Truth Data

3.2.1 Default Shallow Performance

The performance of the CNN before and after improvement (hyperparameter tuning and anti-overfitting measures) are shown in *Figure 3.1A* and *Figure 3.1B* respectively.

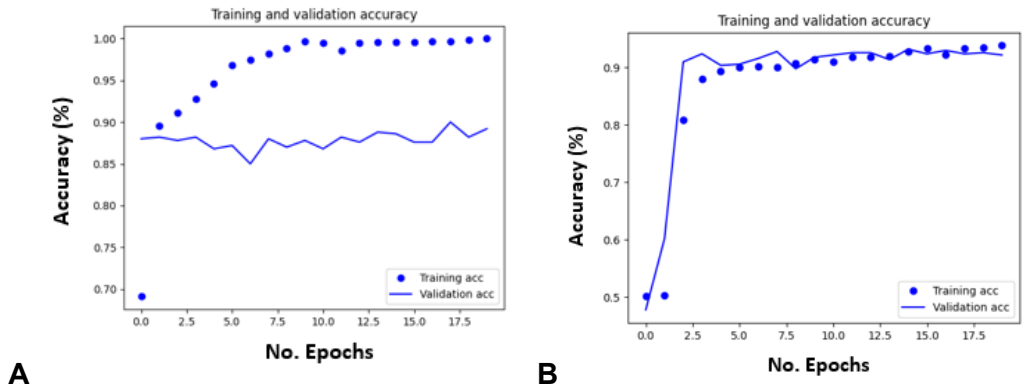

Figure 3.1: *Accuracy graphs for base model (**A**) and base model with improvements (**B**)*

The introduction of dropout layers and data augmentation can be seen as having clearly mitigated the problem of overfitting. In *Figure 3.1A,* the training accuracy is consistently higher than the validation accuracy, while in *Figure 3.1B* there is much less of a disparity. The improved model plateaus at an accuracy of around 92.5%, which is higher than the user performance of 85.3%.

3.2.2 Deeper Network Performance

A comparison between the accuracy of networks with one, two and three convolutional layers respectively is shown in *Figures 3.2A, 3.2B* and *3.2C* respectively.

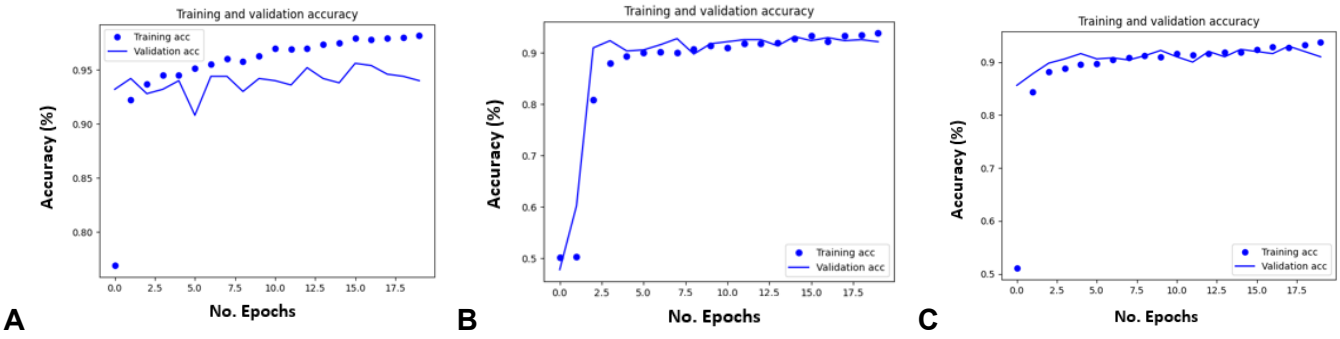

Figure 3.2: *Accuracy graphs for models with one (**A**), two (**B**) and three (**C**) convolutional layers.*

From the accuracy graphs in *Figure 3.2*, we can see that in general, as the CNN became deeper with moer convolutional layers, their overall accuracy decreased. The most shallow CNN with only one convolutional layer was able to reach accuracies around 95%, while those with two and three convolutional layers reached only 92.5% and 91% respectively. This is consistent with the results in Kociołek et al (2022), where the shallow networks outperformed the deeper ones. It can be noted that there may be a degree of overfitting in the shallow network, as training accuracies are higher than those on the validation set. This could be reduced with a wider selection of data and higher dropout.

3.3 Network Trained on User Data

3.3.3 Default Performance and Implications

The performance of the CNN trained on the user-generated data is shown in *Figure 3.3*.

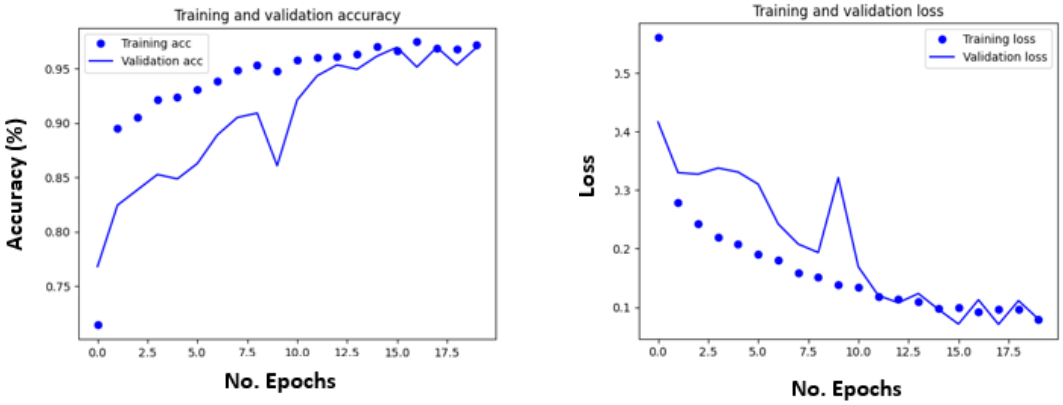

Figure 3.3: *Accuracy and loss graphs for base model trained on user-data*

This network was able to achieve very high accuracies of around 97% with very little overfitting. The accuracies achieved by this network are significantly higher than those reached by models trained with the ground truth data. This may be indicative of a few things; it is possible that this architecture shares similarities with the V1 later in the brain, and hence performs better on user data. It is also possible that there is a problem with the base data, in that the noise in the signal is too high such that

it is undetectable from datasets with no signal. This may mean the base dataset is improperly labelled, hence why models trained with it cannot achieve high accuracies.

3.3.2 Performance with Different Neuron Quantity

The performance of user-data trained models with 32, 64 and 128 neurons in their first convolutional layer are shown in *Figures 3.4A, 3.4B* and *3.4C* respectively .

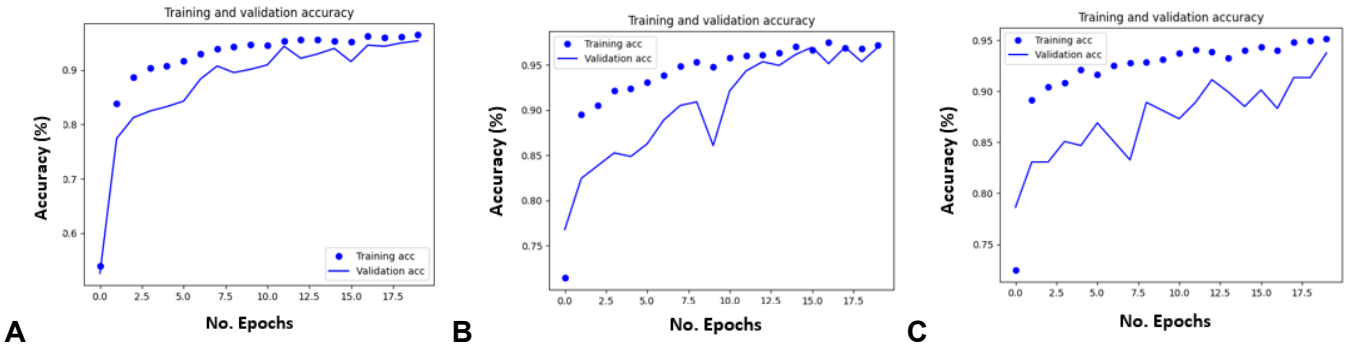

Figure 3.4: *Accuracy graphs for models with 32 (**A**), 64 (**B**) and 128 (**C**) neuron-layers.*

From these performances, we can see that the model with 64 neurons performed the best with a peak accuracy of around 97%. The 32 neuron model appeared to plateau near a 95% accuracy, while the 128 neuron model peaked at around 94%. Note that there appears to be some overfitting in the 128 neuron model, and an upward trend in accuracy, meaning more epochs may result in a better trained model.

4 Future Objectives

4.1 Robust Data Collection

As previously mentioned, the current user testing data was collected in a rapid manner from a single experimental subject. Moving forward, it would be ideal to test the user-training dataset on a larger population, and take an average of the results. This would remove any bias and accidental misinputs. Furthermore, more variability could be added to the data. For example, rather than being vertical, the texture boundary could be horizontal. Furthermore, rather than having opposing signals on both sides of the image, texture could be generated through signal on one side and noise on the other. This would allow for a greater representation of texture.

4.2 Testing more Network Structures

Although some of the fundamental structural differences and their effects on the model have been tested, there are still gaps that need to be tested to ensure the generated model is as efficient as possible. For example, the effects of kernel size and activation layers requires further testing. Likewise, the depth of the model should be tested on the user-trained data to determine what depth of network may correspond to the human brain.

4.3 Linking to Biological Concepts

Lastly, there is a need to link the networks tested to a biological foundation, such that we can learn more about how the V1 layer of neurons in the brain responds to texture. This may be done by evaluating the trained models to determine whether they mimic a human response properly, and applying further structural testing on a larger scale.

References

Chirodea, M. C., Novac, O. C., Novac, C. M., Bizon, N., Oproescu, M., & Gordan, C. E. (2021). Comparison of Tensorflow and PyTorch in Convolutional Neural Network - based Applications. *2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. https://doi.org/10.1109/ecai52376.2021.9515098

Kociołek, M., Kozłowski, M., & Cardone, A. (2022). A Convolutional Neural Networks-Based Approach for Texture Directionality Detection. *Sensors*, *22*(2), 562. https://doi.org/10.3390/s22020562

Mathot, S. (2018). *Cogsci*. Online Gabor-Patch Generator. https://www.cogsci.nl/gabor-generator

Ososkov, G., & Goncharov, P. (2017). Shallow and deep learning for image classification. *Optical Memory and Neural Networks*, *26*(4), 221–248. https://doi.org/10.3103/s1060992x1704004x