Deep Learning with Principal Component Wavelet Networks

CS39440 Major Project Report

Author: Samuel Spurling ([sas90@aber.ac.uk](mailto:sas90@aber.ac.uk))

Supervisor: Dr Bernie Tiddeman. My Supervisor (bpt@aber.ac.uk)

29th March 2022

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in  
Computer Science and Artificial Intelligence (With Integrated Year in Industry) (GG47)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

Name …………………………………………

Date ……………………………………………

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name …………………………………………

Date ……………………………………………

Acknowledgements

I am grateful to…

I’d like to thank…

Abstract

In this report we propose a novel application for Principal Component Wavelet Networks (PCWN) [1]. The PCWN decomposes an image using handcrafted invertible 2D wavelet filters-banks and 1x1 filters learnt through principal component analysis to control the size of decompositions. The reconstruction is accomplished using the inverted filter bank and an approximately inverted PCA. Previously the approach has shown competitive results for linear inverse problems in computer vision such as compressive sensing, super-resolution, and in-painting. The proposed novel application is image segmentation. This is a task where there are objects in an image and through segmentation the image is split into distinct components, or segments, where each segment represents an object in the image.

We use 2 variants of the PCWN, a fully convolutional and a fully connected approach. In experiments, there will be a comparison of the PCWN approach to the U-net model, a standard architecture for image segmentation, to benchmark the PCWN models performance. The results of the experiments demonstrate that PCWN has great potential when applied to the task of image segmentation. Possible future work looks at using PCWN as regularisation in the U-net architecture and replacing PCA with a different kind of invertible dimensionality reduction such as Kernel PCA or UMAP.

[Figure 1: Wavelet decomposition of an image of a bird 9](https://d.docs.live.net/455c1cc37aadd471/Desktop/major_project/MajorProject/Writeup/Writeup/MajorProject.docx#_Toc101128378)

[Figure 2: Fully connected PCWN architecture 11](#_Toc101128379)

[Figure 3: Fully convolutional PCWN architecture 12](https://d.docs.live.net/455c1cc37aadd471/Desktop/major_project/MajorProject/Writeup/Writeup/MajorProject.docx#_Toc101128380)

[Figure 4: high-resolution features being used to supplement low-resolution features, providing location information 12](https://d.docs.live.net/455c1cc37aadd471/Desktop/major_project/MajorProject/Writeup/Writeup/MajorProject.docx#_Toc101128381)

[Figure 5: Atrous convolution: Atrous rate = 1 13](https://d.docs.live.net/455c1cc37aadd471/Desktop/major_project/MajorProject/Writeup/Writeup/MajorProject.docx#_Toc101128382)

[Figure 6: Atrous convolution: Atrous rate = 2 13](https://d.docs.live.net/455c1cc37aadd471/Desktop/major_project/MajorProject/Writeup/Writeup/MajorProject.docx#_Toc101128383)

Contents

[1 Background, Analysis & Process 8](#_Toc101128388)

[1.1 Problem Description 8](#_Toc101128389)

[1.2 Background 8](#_Toc101128390)

[1.3 Analysis 8](#_Toc101128391)

[1.3.1 Fundamentals 8](#_Toc101128392)

[1.3.2 Architectures for Segmentation 10](#_Toc101128393)

[1.3.3 Research Questions 14](#_Toc101128394)

[1.4 Process 14](#_Toc101128395)

[2 Experiment Methods 15](#_Toc101128396)

[3 Software Design, Implementation and Testing 16](#_Toc101128397)

[3.1 Design 16](#_Toc101128398)

[3.1.1 Overall Architecture 16](#_Toc101128399)

[3.1.2 Some detailed design 16](#_Toc101128400)

[3.1.3 User Interface 17](#_Toc101128401)

[3.1.4 Other relevant sections 17](#_Toc101128402)

[3.2 Implementation 17](#_Toc101128403)

[3.3 Testing 17](#_Toc101128404)

[3.3.1 Overall Approach to Testing 18](#_Toc101128405)

[3.3.2 Automated Testing 18](#_Toc101128406)

[3.3.3 Integration Testing 18](#_Toc101128407)

[3.3.4 User Testing 18](#_Toc101128408)

[4 Results and Conclusions 19](#_Toc101128409)

[4.1 Issues with experiments 20](#_Toc101128410)

[4.1.1 Memory errors 20](#_Toc101128411)

[4.2 Experimental Results: Determining best parameters for segmentation 20](#_Toc101128412)

[4.2.1 Convolutional Network approach 20](#_Toc101128413)

[4.3 Experimental Results: Convolutional vs Connected PCWN 21](#_Toc101128414)

[4.4 Experimental Results: Connected PCWN vs U-net 21](#_Toc101128415)

[5 Critical Evaluation 21](#_Toc101128416)

[6 References 23](#_Toc101128417)

[7 Appendices 24](#_Toc101128418)

[A. Third-Party Code and Libraries 25](#_Toc101128419)

[B. Code Samples 26](#_Toc101128420)

# Background, Analysis & Process

## Problem Description

## Background

The motivation for this project stems from an interest in computer vision and dimensionality reduction techniques, and a curiously for wavelets. Image segmentation is an active field of research, with applications in medical imaging, satellite imagery and image-based search. Modern methods for image segmentation use multi-scale feature maps, which means the feature maps encodes information from small scale features, such as a paw in an image of a cat, as well as large scale features, such as the background of the image.

The project extends the PCWN method [1] to the task of image segmentation, so naturally PCWN was a key part of the background research. The motivation for this was to understand the novel approach to linear inverse problems and apply my findings to an image segmentation approach. Convolutional scattering wavelet networks [2] is another method which shares similarities to PCWN, especially on methods for mitigating the issue of large channels after multiple applications of wavelet filter banks. To understand the segmentation task, there was research of U-net [3], DeepLabV3 [4] and a comparison of segmentation methods [5].

## Analysis

### Fundamentals

#### Principal Component Analysis

Principal Component Analysis (PCA) is a method of dimensionality reduction that calculates “principal components” and then orients the data using the principal components as basis vectors. The advantage of using PCA is that high dimensional data can be expressed in lower dimensions whilst retaining almost all the variance between data points. Consequently, the curse of dimensionality [6] can be avoided when using PCA in machine learning, where higher dimensions cause an increase in sparsity, causing data objects to appear dissimilar which impacts the algorithms’ ability to learn.

Given an input matrix of data A, this is how to calculate the principal components. First the covariance matrix and the mean of the data is calculated. The mean is subtracted from the data to standardise the data.

The principal components are found by calculating the eigenvectors and eigenvalues of the covariance matrix, and then the eigenvectors are ordered using the eigenvalues. A n-dimensional dataset has n principal components. The sum of the eigenvalues of n principal components is the total variance, therefore we can calculate explained variance by dividing the eigenvalue of a component by the sum of the eigenvalues. The total explained variance is found by summing the explained variance for each component selected. The following is an example of eigenvectors (v) and eigenvalues ().

The eigenvectors of the covariance matrix are the direction of the axes which have the highest variance, and the eigenvalues are the amount of variance in the respective components. The feature vector is formed using the eigenvectors like so.

We are applying PCA to the feature maps produced by the filter banks. Multiplying the transpose of the feature maps (A) by the transpose of the feature vector (B) produces a dimensionally reduced feature map (C) that retains most of the information of the higher dimension feature map.

#### Convolutional Neural Networks

An artificial neural network (ANN) is a directed and weighted graph where neurons are nodes and neuron connections (weights) are edges. ANNs are composed of layers, where every layer is a collection of neurons. The most common architecture is feed-forward. In this architecture every layer is connected to another layer, except the output layer, which only has a incoming connections. ANNs typically consist of three types of layers, input, output and hidden. The hidden layers of an ANN allow them to function as universal function approximators. That is to say, with the correct architecture and parameters an ANN can model any function to a small degree of error. Finding the correct architecture and parameters is a complex task with no general solution, and there are no guarantees of optimality. Typically, ANN’s are trained using vast amounts of data and an error update algorithm that tweaks the parameters of the network towards a more optimal solution. Normally this algorithm is backpropagation [7].

There are many architectures other than feed-forward networks, such as Recurrent Neural Networks (RNNs), transformers and self-organising maps. The architecture used for this project is Convolutional neural networks (CNNs).

CNNs are a commonly used architecture in computer vision. Popularised by Kunihiko Fukushima [7] and Yann LeCun [8] between 1980-1990, CNNs have a number of desirable qualities for computer vision tasks. CNN’s work using convolutional filters, which are 2D matrices, or kernels, that slide over the image and extract features. At every position a convolution between the kernel and the image is calculated. Multiple kernels can be used to generate deep feature maps, and the kernels can have any size. The size of the steps taken when sliding the kernel over an image is called the stride. A stride of 2 means that a pixel is skipped with every step.

#### Wavelets

Appropriate image representation is key to the success of an algorithm. There exist several different representations. The most familiar representation is the spatial domain. The spatial domain is how we view digital images, with resolution, colour channels and pixel intensity values. Other representations include the Fourier domain and wavelets.

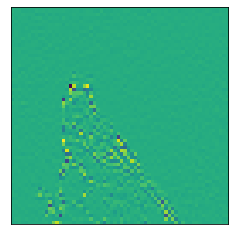
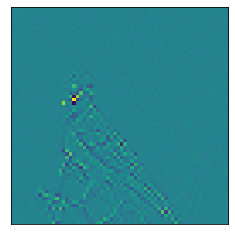
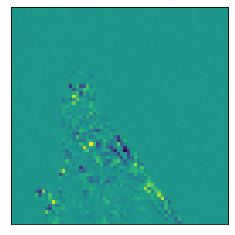
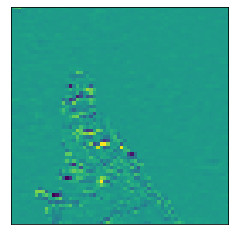
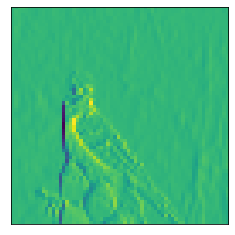
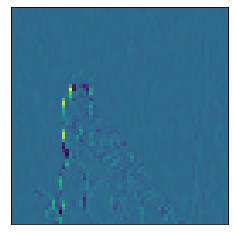
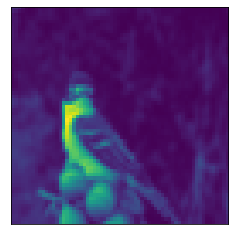
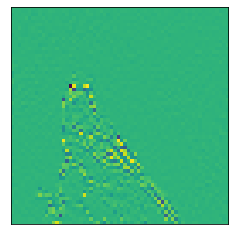
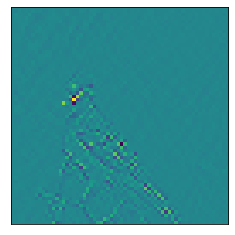
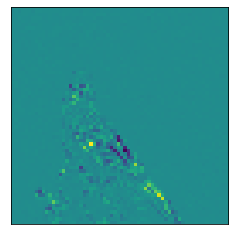
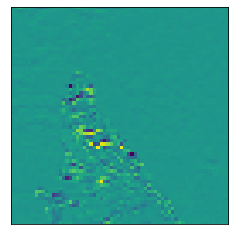
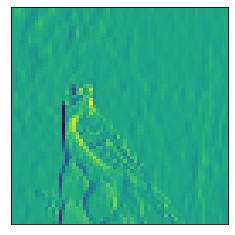
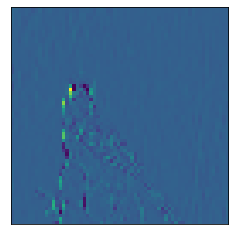
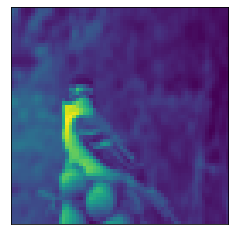
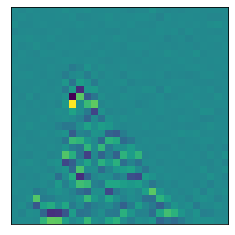
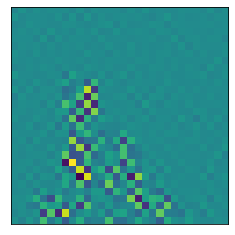
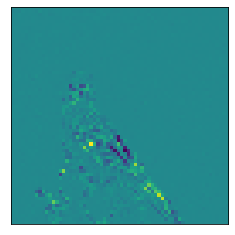
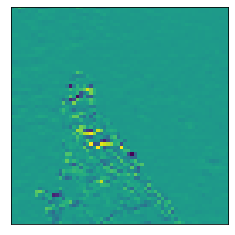
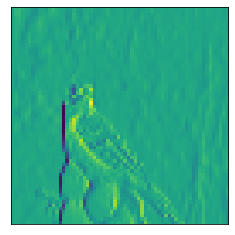
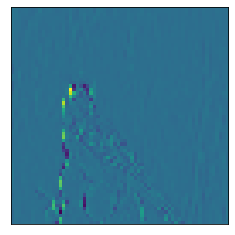
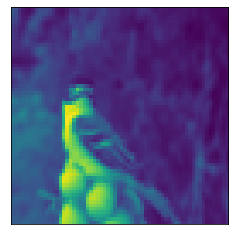
Wavelets transforms are similar to Fourier transforms, except that whereas Fourier can only capture frequency information, wavelets can capture frequency and location information.

Wavelets can be classified as discrete or continuous. Wavelets have been in use since the early 20th century, originating with Harr wavelets [7] which have enjoyed a wide range of applications, notably in Viola-Jones facial detection [8]. Harr wavelets are discrete wavelet transforms. There are many applications of wavelets, for example scattering wavelets are used in convolutional scattering wavelet networks [2]. This method in the paper used sets of different scale Morlet wavelets, an approach which has desirable properties such as deformation, rotation, and scale invariance. Morlet wavelets do not have a simple inversion, requiring direct optimisation [9] or generative networks [10] to solve them.

The bank of wavelet filters used in PCWN are discrete and invertible. At every level of decomposition, each filter in the filter bank is applied to each channel in the partial decomposition. The filter bank used is based on approximations of the zeroth, first and second derivatives of the gaussian, applied along the x and y direction. Every input channel outputs 9 more channels in the next decomposition.

Below is an example of one level of decomposition. The input is an image, which naturally has 3 channels, red, green, and blue. The number of output channels is 27, however many of these channels are redundant. This can be seen in the decomposition, where the difference between the channels rapidly decreases. This is where PCA is used, to reduce the number of channels, whilst retaining the variance.

Figure 1: Wavelet decomposition of an image of a bird



### Architectures for Segmentation

#### PCWN

PCWNs combined PCA and separable derivative of Gaussian filters. The purpose of applying PCA is that the filter bank is applied to every channel at every level of the decomposition. Applying the wavelet filter to every channel after every partial decomposition causes a rapid growth in the number of channels, and many of the resultant channels will contain redundant information. The use of 1x1 PCA filters controls this growth.

PCWNs have worked excellently for linear inverse tasks in computer vision. For segmentation the PCWN architecture requires some edits. The method used for adapting PCWN segmentation is incredibly versatile and can easily be adapted to a number of tasks, such as crowd counting and point detection.

The PCWN architecture is a hybrid of CNNs and wavelet transforms. [ADD MORE ON PCWN architecture]

The method involves training 2 separate PCWN networks. One network trains on the input, in this case an image. The other network trains on the images with the segmented masks applied. Both networks are only able to reconstruct the inputs, or corrupted versions of them. To extend PCWN to the task of segmentation, there needs to be a method for mapping the decomposition of the image network to the decomposition of the segmentation network.

The mapping can be thought of as a linear equation, where y is the segmentation decomposition, x is the image decomposition, and matrix A and vector b are variables that once solved will map x to y.

Using linear least squares error it is possible this equation quite efficiently and simply. The aim is to minimise the least squares expression below:

To do this requires finding the derivatives of each element with respect to A and b, the equations to do this end up looking like the two below:

The vector b is eliminated through substitution:

The substitution results in the following equation:

For the readers sake, the equation can be simplified by introducing X and Y as substitutions for and respectively, giving the following equation.

To find A, both sides of the equation are multiplied by the inverse of X:

X =

Which simplifies to:

=

It is then trivial to find vector b by substituting A back into the original equation and rearranging:

The method for calculating A and b can be accomplished in 2 ways, a fully connected method, or a fully convolutional method. The method of using linear least squares is the same, but the shape of the tensors is different.

##### Fully connected method

Diagram

Description automatically generated

Figure 2: Fully connected PCWN architecture

In the fully connected method, the decompositions are flattened. This approach is location variant, and so is suited to images where the object of interest is in a consistent position, such as the centre of the image. When calculated using the fully connected method, the X and Y matrix can become very large. This can cause memory issues, especially when calculating the Moore-Penrose pseudoinverse. The actual size of A is the product of the product of the dimensions of each decomposition.

The method for solving the linear inverse for the fully connected PCWN is exactly as stated in the previous section. However, when using A and b to transform an image decomposition into a segmentation decomposition, the image decomposition must be flattened, and the output is then reshaped, where the shape is that of a valid segmentation decomposition.

A screenshot of a computer

Description automatically generated with medium confidenceFully convolutional method

Figure 3: Fully convolutional PCWN architecture

The fully convolutional method is slightly more complex. This method is appropriate for images where the object of interest does not have a fixed position

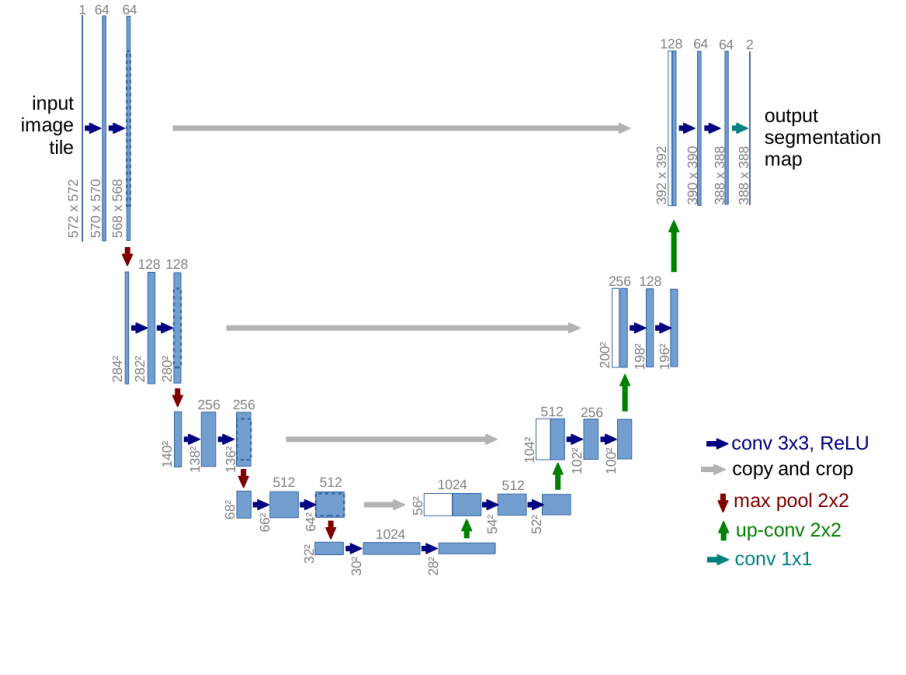
#### U-Net Architecture

The U-net architecture was conceived in 2015 and is so called because of its U-shaped architecture. The success of U-map comes from the use of skip connections. These are connections in the network that allow layer outputs to bypass successor layers of the network. The result is then concatenated or reintroduced later in the network. Skip connections, or residual connections, first appeared in highway networks [11]. Highway networks implemented skip connections with gates, similar to a long-short term memory (LSTM) [12] mechanism. Perhaps the most famous architecture to use skip-connections is ResNet [13], an image recognition network that is still used as a benchmark today.

Graphical user interface, text, application, Word

Description automatically generatedThe advantage of using skip connections is feature reusability. In a standard fully convolutional network there is a contracting path. This contracting path reduces the resolution of the feature map with every successive convolution and pooling operation. This means that feature information is lost, especially information regarding location of the object being segmented. For image segmentation, the location of an image is important in constructing the mask. By using skip connections, it is possible to reuse high-resolution features to supplement up-sampled outputs from deeper layers, resulting in a more precise output. In the below example the blue box shows the location information provided by the high resolution feature to the high-level low resolution feature in the yellow box

Figure 4: high-resolution features being used to supplement low-resolution features, providing location information



#### DeeplabV3 Architecture

The DeepLabV3 is state of the art for semantic segmentation. The approach proposes the use of atrous convolutions, used in a method called Atrous Spatial Pyramid Pooling (ASPP) [11]. The word “trous” in French means holes, which is the principle of atrous convolutions. The method uses convolutional filters with a dilation rate which determines the padding between values in the filter. Atrous convolutions are excellent for multi-scale features, capturing dense features whilst maintaining the resolution of resultant feature maps. As mentioned, when discussing U-net, deep convolutional networks that use pooling and striding do not maintain resolution of feature maps, meaning information is lost.

The formula for calculating an atrous convolution is:

Where y is the output feature map, x is the image, i is a location in x and y, r is the atrous rate and w is the kernel. By increasing the atrous rate, we increase the number of zeros between consecutive values in the filter. For example, an atrous value of 1 is a conventional filter. An atrous rate of 2 pads one zero between every value in the filter.





Figure 5: Atrous convolution: Atrous rate = 1

Figure 6: Atrous convolution: Atrous rate = 2

The reason for including details on DeeplabV3 and atrous convolutions is that the ASPP method could be applied to future PCWN work. Atrous convolutions have been used with undecimated wavelet networks before successfully [12] and would be an interesting edit to the PCWN architecture, however ASPP was beyond the scope of this project.

### Research Questions

There were several initial questions posed:

* Can you reimplement the network?
* Can you apply the network to a novel task?
  + Image segmentation
  + Crowd counting
* Can another dimensionality reduction be used instead of PCA?
* What are the effects of changing network parameters?

Of these questions, I have completed 3, and would suggest the remainder for a future project.

#### Can you reimplement the network?

This task is necessary to complete for the success of the rest of the project. Simply, the task is to reimplement the network for a new dataset.

#### Can you apply the network to a novel task? (Image Segmentation)

This task was the main focus of the project. There are a number of technical aspects to answering this question which were expected to take a long time, which is why the remaining questions have been delegated to future research.

#### What are the effects of changing network parameters?

This task is a natural extension of the segmentation task. Some experimentation would be required to determine the best parameters for segmentation. There are several parameters to vary through experiments. For the model, the parameters to edit include layer count, percentage of channels kept, different architectures and different activation functions. For the dataset, the editable parameters are image resolution and training data size.

## Process

You need to briefly describe the life cycle model or research method that you used. You do not need to write about all of the different process models that you are aware of. Focus on the process model or research method that you have used. It is possible that you needed to adapt an existing method to suit your project; clearly identify what you used and how you adapted it for your needs.

For the research-oriented projects, there needs to be a suitable process for the construction of the software elements that support your work.

[TO DO]

# Experiment Methods

This section should discuss the overall hypothesis being tested and justify the approach selected in the context of the research area. Describe the experiment design that has been selected and how measurements and comparisons of results are to be made.

You should concentrate on the more important aspects of the method. Present an overview before going into detail. As well as describing the methods adopted, discuss other approaches that were considered. You might also discuss areas that you had to revise after some investigation.

You should also identify any support tools that you used. You should discuss your choice of implementation tools or simulation tools. For any code that you have written, you can talk about languages and related tools. For any simulation and analysis tools, identify the tools and how they are used on the project.

For the parts of your project that need some engineering (hardware, software, firmware, or a mixture) to support the experiments, include details in your report about your design and implementation. You should discuss with your supervisor whether it is better to include a different top-level section to describe any engineering work. In this template, Chapter 3 is suggested as a place for that discussion.

# Software Design, Implementation and Testing

This could be one chapter or a few chapters. It should define and discuss the software that is developed to support the research that is being conducted. For example, if your research involves running experiments, what software are you creating to support that work? What functionality is required? What design will be used? What implementation issues are there and what testing is used?

Even though a research project is investigating specific research questions, it is still necessary for you to discuss the software that you develop. Research has a habit of generating bits of software that can exist for several years and need future modification. Therefore, you need to be able to discuss the technical issues as well as the research approach.

## Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

### Overall Architecture

### Some detailed design

#### Even more detail

### User Interface

### Other relevant sections

## Implementation

This section should discuss issues you encountered as you tried to implement your experiments. What were the results of running the experiments? What conclusions can you draw from these results?

During the work, you might have found that elements of your experiments were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

If you had multiple experiments to run, it may be sensible to discuss each experiment in separate sections.

## Testing

Detailed descriptions of every test case are definitely not what is required in this section; the place for detailed lists of tests cases is in an appendix. In this section, it is more important to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

Whilst testing with “real users” can be useful, don't see it as a way to shortcut detailed testing of your own. Think about issues discussed in the lectures about until testing, integration testing, etc. User testing without sensible testing of your own is not a useful activity.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

### Overall Approach to Testing

### Automated Testing

#### Unit Tests

#### User Interface Testing

#### Stress Testing

#### Other Types of Testing

### Integration Testing

### User Testing

# Results and Conclusions

This section should discuss issues you encountered as you tried to implement your experiments. What were the results of running the experiments? What conclusions can you draw from these results? What graphs or other information have you assessed regarding your experiments? Discuss those.

During the work, you might have found that elements of your experiments were unnecessary or overly complex; perhaps third-party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

If you had multiple experiments to run, it may be sensible to discuss each experiment in separate sections.

## Issues with experiments

### Memory errors

By far the most prevalent error was out of memory errors. The cause of this ranged from inefficient programming, insufficient RAM, and poor memory management. To solve the inefficient programming and memory management issues we removed unnecessary objects. The insufficient RAM issue was solved initially by using cloud-based platforms, and then by purchasing a new laptop, upgrading the RAM from 8GB to 16GB.

Memory errors meant initial experiments for the fully connected network were throttled at a 128x128 image resolution. This in turn limited the “count” parameter of the PCWN. This parameter controlled the number of layers in the head, and the inverse head. Every layer reduced the resolution of the composition by half. A 128x128 image would have a feature map with a 4x4 resolution after passing through the head component of a network with 5 layers. This caused an error with the convolutional filters used in reconstruction, as the resolution is too small.

The fully convolutional network had notably fewer memory problems compared to the fully connected version. The reason for this was the calculation of the A matrix, which uses computationally intense methods, such as calculating the pseudo-inverse. The A matrix for the fully connected method was larger than the fully convolutional matrix, meaning the memory usage was much higher for the fully connected variant.

## Experimental Results: Determining best parameters for segmentation

### Convolutional Network approach

Table 1 Results of convolutional PCWN with different parameters

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Exp** | **training data size** | **count** | **keep %** | **resolution** | **dice mean test** | **dice mean train** | **dice std test** | **dice std train** | **seg train time** | **linear inverse train time** |
| **1** | None | 4 | 0.8 | 128 | 0.339 | 0.326 | 0.245 | 0.253 | 1072.086 | 1702.609 |
| **2** | 4000 | 3 | 0.3 | 128 | 0.358 | 0.348 | 0.218 | 0.222 | 168.371 | 242.668 |
| **3** | 2000 | 3 | 0.3 | 128 | 0.360 | 0.353 | 0.218 | 0.220 | 83.499 | 124.600 |
| **4** | 6000 | 3 | 0.2 | 64 | 0.564 | 0.557 | 0.203 | 0.205 | 251.632 | 297.255 |
| **5** | 6000 | 3 | 0.3 | 256 | 0.564 | 0.557 | 0.203 | 0.205 | 256.319 | 355.888 |
| **5** | 6000 | 3 | 0.1 | 128 | 0.565 | 0.558 | 0.204 | 0.206 | 226.381 | 327.403 |
| **6** | 6000 | 3 | 0.1 | 128 | 0.565 | 0.558 | 0.204 | 0.206 | 260.870 | 327.596 |
| **7** | None | 3 | 0.3 | 128 | 0.591 | 0.584 | 0.196 | 0.194 | 328.039 | 433.187 |
| **9** | None | 4 | 0.3 | 128 | 0.602 | 0.595 | 0.182 | 0.183 | 441.355 | 783.941 |
| **10** | None | 4 | 0.1 | 128 | 0.602 | 0.595 | 0.182 | 0.183 | 391.624 | 726.183 |

This experiment tests the strength of the convolutional network approach to PCWN. The results are sorted by the mean dice score. We can make a number of deductions from the results about what factors improve the performance of the model.

Training data size gives a significant improvement in the score (“None” means the full dataset was used, with 100 samples reserved for testing). In experiment number 2 and 3 there appears to be a small decrease in performance after increasing the training data size from 2000 to 4000. However, the result is not statistically significant.

Keep percent has a surprisingly negative effect on the dice score. Comparing result 1 to result 10 there is a very clear increase in dice score, 0.339 to 0.602, corresponding to decreasing the keep percent from 0.8 to 0.1. Furthermore, the training time decreases.

Count also has an effect, however it is not statistically significant unless

A cat sitting on a window sill

Description automatically generated with medium confidence

## Experimental Results: Convolutional vs Connected PCWN

## Experimental Results: Connected PCWN vs U-net

# Critical Evaluation

Examiners expect to find a section addressing questions such as:

* Were the requirements correctly identified?
* Were the design decisions correct?
* Could a more suitable set of tools have been chosen?
* How well did the software meet the needs of those who were expecting to use it?
* How well were any other project aims achieved?
* If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

The questions are an indication of issues you should consider. They are not intended as a specification of a list of sections.

The evaluation is regarded as an important part of the project report; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things in the work and aspects of the work that could be improved. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

# References

|  |  |
| --- | --- |
| [1] | M. G. Bernard Tiddeman, "Principal Component Wavelet Networks for Solving Linear Inverse Problems," *Symmetry,* vol. 13, no. 6, p. 1083, 2021. |
| [2] | S. M. Joan Bruna, "Invariant Scattering Convolution Networks," *IEEE transactions on pattern analysis and machine intelligence,* vol. 35, no. 8, pp. 1872-1886, 2013. |
| [3] | O. P. F. T. B. Ronneberger, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Cham, 2015. |
| [4] | Y. H. Ş. a. G. U. S. C. Yurtkulu, "Semantic Segmentation with Extended DeepLabv3 Architecture," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, 2019. |
| [5] | S. e. a. Ghosh, ""Understanding deep learning techniques for image segmentation."," *ACM Computing Surveys (CSUR) ,* vol. 52, no. 4 , pp. 1-35, 2019. |
| [6] | R. E. Bellman, Dynamic Programming, Princeton University Press, 1957. |
| [7] | A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Mathematische Annalen,* vol. 69, no. 3, pp. 331-371, 1910. |
| [8] | M. J. Paul Viola, "Rapid Object Detection using a Boosted Cascade of Simple," in *Computer Vision and Pattern Recognition*, 2001. |
| [9] | E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko and E. Belilovsky, "Scattering networks for hybrid representation learning," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 41, pp. 2208-2221, 2018. |
| [10] | T. Angles and S. Mallat, "Generative networks as inverse problems with scattering transforms," in *ICLR*, 2018. |
| [11] | G. P. I. K. K. M. a. A. L. L.-C. Chen, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," 2016. |
| [12] | R. K.-M. J. M. a. M. Holschneider, "A real-time algorithm for signal analysis with the help of the wavelet transform," *Wavelets: Time-Frequency Methods and Phase Space,* pp. 289-297, 1989. |

# Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

If you have used any 3rd party code, i.e. code that you have not written yourself such as libraries, then you must include Appendix A. In that appendix, you will provide details of the 3rd party code that you have used.

For most other items, it would be better to include them in your technical submission instead of including them as an appendix. For example:

* If you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document in the technical work. In this report, you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.
* If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification at the start of the project. Perhaps you used stories to keep track of the functionality and the ‘future conversations.’ If it isn’t relevant to include all those stories in the body of your report, you could detail those stores in a document in the technical work.
* If you have used manual testing, then include a document in the technical work that records the tests that have been done. In this report, you would talk about the use of those tests.

Documents included in the technical work or in the appendices are supporting evidence of the work done. Where you include documents, this report should refer to the documents. You should not be relying on detailed study of those documents in order to understand what is written in this report.

Speak to your supervisor or the module coordinator if you have questions about this.

* 1. Third-Party Code and Libraries

If you have made use of any third-party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third-party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

**Apache POI library** – The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client’s existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation **Error! Reference source not found.**. The library is released using the Apache License **Error! Reference source not found.**. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

* 1. Code Samples

Least Squares: Fully Connected

Calculating matrix A and vector b for least squares. This method is used for the fully connected version, where the decompositions are flattened.

Graphical user interface, application, Word

Description automatically generated

Least Squares: Fully Convolutional

Calculating matrix A and vector b for least squares. This method is used for the convolutional variant of PCWN, and reshapes the decompositions into 2D matrices.

Graphical user interface, text, application

Description automatically generated