Single-Image Laparoscope Calibration Using Deep Neural Networks

Raphaël Piccolin

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

of

University College London.

Department of Physics and Astronomy

University College London

September 6, 2019

I, Raphaël Piccolin, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Laparoscopic (keyhole) surgery is a minimally invasive surgery technique which reduces patient recovery time and patient trauma [1]. However, only 10 % of liver cancer surgeries are performed laparoscopically [2]. This is because an augmented reality (AR) display is necessary to compensate for the loss of direct vision and dexterity compared to an open surgery. Calibration of the laparoscope, a time-consuming and problematic task, is needed to use AR. To help solve this problem, we trained a deep neural network to predict camera calibration parameters, the camera's intrinsic parameters, using a single image from a laparoscope.

Our best-performing network was one which used VGG19 [3] as a base, training only its last two blocks. It obtained mean absolute percentage errors of 2.375 % and 2.776 % for focal parameter prediction, and 9.751 % and 8.517 % for the principal point coordinates. Their standard deviations were 59.44, 64.31, 109.1, and 52.91, respectively. When testing our networks on real calibration images, the best mean absolute percentage error for each parameter was lower than 10.53 %.

This research is part of the SmartLiver project at UCL, which is developing an image-guidance system for laparoscopic liver cancer surgery. The source code for this project is hosted on GitHub:

`https://github.com/raphael-p/weiss-data_generation`

`https://github.com/raphael-p/weiss-networks`

Acknowledgements

I would like to thank Bongjin Koo and Matt Clarkson at WEISS, UCL for their help and advice in producing this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Historical Background

This section discusses the history leading up to the two main fields explored in this thesis: laparoscopic surgery with augmented reality techniques, and pattern recognition with convolutional neural networks.

1.1.1 Laparoscopic Surgery

Laparoscopy was pioneered on dogs by Georg Kelling in 1901, and later performed by Jacobaeus on humans [4]. Laparoscopy was initially used for diagnosis. As technology improved it became more widely used, most notably in gynaecology. It was not until the 1980s when several surgeons performed laparoscopic cholecystectomy that laparoscopy became a means of performing surgery [5].

In the last 20 years, minimally invasive surgery gained in popularity, increasing the interest in laparoscopy. However, using a camera on its own is no match for open surgery, which benefits from direct vision and tactile feedback. This is why laparoscopy has been often paired with Augmented Reality (AR) techniques to help guide surgery by giving a surgeon more information [6].

1.1.2 Deep Neural Networks

The first proposition to build a mathematical model based on neurons came in a 1943 paper by McCulloch and Pitts [7]. The neural network they proposed is the basis for modern Artificial Neural Networks (ANNs). With the advent of modern computers, Frank Rosenblatt was able to create the 'perceptron', which is regarded

as the first deep neural network. Many developments were made over the second half of the 20th century, with the field's growth in popularity. Most relevant to this thesis is the invention of convolutional layers by Fukushima in 1980 [8]. This led to Convolutional Neural Networks (CNNs), which became capable of recognising visual patterns. CNNs are similar to biological visual processing in that individual neurons receive input from only a certain area of the image.

Interest (and funding) for deep learning has varied with time, and it experienced a revival partially thanks to the ImageNet [9] database. Launched in 2009, ImageNet is a massive labelled image dataset which can be used for supervised learning of object classification. A breakthrough for CNNs came 2 years later with AlexNet [10] winning an ImageNet competition, amongst others. Since then, a few other CNNs trained on ImageNet data gained in popularity and started being used for other applications.

1.2 SmartLiver Programme

This work was conducted at the Wellcome / EPSRC Centre for Interventional and Surgical Sciences (WEISS), which collaborates with the UCL Centre for Medical Image Computing (CMIC). They work with researchers from various UCL departments from the Faculties of Engineering and Medical Sciences.

The research for this thesis is part of the SmartLiver project led by Dr. Matthew Clarkson. The aim of the project is to make laparoscopic liver surgery more widely accessible to cancer patients. Benefits of laparoscopy include faster recovery and reduced pain, bleeding, and scarring [2][1]. Despite the benefits of laparoscopy, 90 % of liver cancer patients are currently treated with open surgery instead [11]. This is because it is considered a delicate procedure due to the difficulty associated with visually identifying a tumour during surgery, and a high-risk of bleeding due to the close proximity of major vascular channels.

In this light, the SmartLiver software uses a surface reconstruction of the patient's liver based on computed tomography (CT) scans. The model is overlaid on the liver in real-time in the laparoscopic video feed. The focuses of this software are the calibration of the laparoscope, and the alignment of the liver. This technology can substantially reduce risks associated with laparoscopy and render it viable for a larger portion of patients.

1.3 Aims and Objectives

The aim of this research is to improve on current methods for pre-operative calibration of the laparoscope. State-of-the-art techniques require multiple images of a calibration board in different positions. We attempt to build a deep neural network which is capable of determining the intrinsic parameters of the camera with a single image. This would greatly reduce the time, difficulty, and failure rate of laparoscope calibration, and thus allow for easier implementation of SmartLiver technology in operation theatres.

The objectives of the project are as follows:

1. Generating training samples: images of a calibration pattern in randomised positions and orientations with operating theatre backgrounds;
2. Implementing a deep neural network on TensorFlow, on top of popular ImageNet models, such as VGG;
3. Training a neural network based on image or geometrical features;
4. Evaluating the predictive capability of the calibration algorithm in with real laparoscope images from a laboratory.

The model we are trying to create is somewhat simplified as it only attempts to find the camera matrix's intrinsic parameters, and does not account for lens distortion. This project is a proof-of-concept for using a neural model in this context, and, if successful, would warrant further investigation for it to become a finished product.

Chapter 2

Calibration

2.1 Pinhole camera model

Zhang's method [12] is the current state-of-the-art technique for camera calibration [13] [14], it assumes a pinhole camera model and does not account for lens distortion. A planar calibration board is used, typically with a checker-board pattern as a program can easily identify the corners of each square.

To calibrate a camera, one most determine both its extrinsic and intrinsic matrices. The former maps a point in 3D world coordinates to 3D camera coordinates. The latter maps the camera coordinates to a 2D image plane, as can be seen in figure 2.1.

We have a point in the world coordinates $\boldsymbol{P}(x, y, z, 1)$, the planar image coordinates of the same point $\boldsymbol{p}(x, y, 1)$, a camera matrix M, and a scale factor s. The extra dimension represented by 1 is used to produce an augmented vector/matrix.

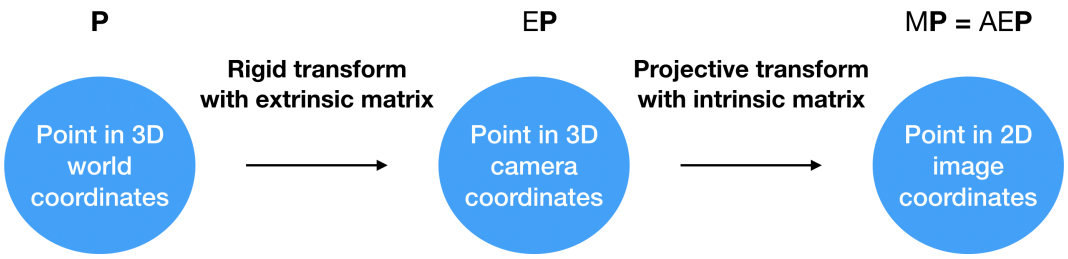

Figure 2.1: Matrix transforms to project a point in 3D world coordinates onto a 2D planar image. $\boldsymbol{P}$ represents the point in 3D world coordinates, E is an extrinsic matrix, A is the intrinsic matrix, and M is the camera matrix.

These are related by:

$$s\boldsymbol{p} = M\boldsymbol{P} \tag{2.1}$$

We denote the intrinsic matrix as A, and the extrinsic matrix as E, and we have that $M = AE$. This transform is illustrated in figure 2.1.

The extrinsic matrix is simply a 4x4 rotation and translation matrix. And the intrinsic matrix is [15]:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

- (u_0, v_0) are the coordinates of the principal point, i.e. the origin of the image plane
- α and β are focal lengths scaled across the image's u and v axes
  Defined as: $\alpha = \frac{F}{\rho_x}$, $\beta = \frac{F}{\rho_y}$
- γ is the skew factor of the image axes
- F is the focal length in world units (mm)
- ρ_x and ρ_y are pixel dimensions in world units (mm)

A 3x3 rotation matrix, R, is embedded in the extrinsic matrix. It describes the 3D rotation of an object in terms of the following three rotation parameters: yaw θ, pitch ϕ, and roll ψ. It is obtained by performing the matrix multiplications: $R = R_\theta \cdot R_\phi \cdot R_\psi$ [16]:

$$R = \begin{bmatrix} cos\psi cos\theta & sin\psi cos\theta & -\sin\theta \\ -sin\psi cos\phi + cos\psi sin\theta cos\phi & cos\psi cos\phi + sin\psi sin\theta sin\phi & cos\theta sin\phi \\ sin\psi sin\phi + cos\psi sin\theta cos\phi & -cos\psi sin\phi + sin\psi sin\theta cos\phi & cos\theta cos\phi \end{bmatrix} \tag{2.3}$$

2.1.1 Geometry & Optics

The geometry of the pinhole camera model is shown in figure 2.2. O is the origin of the 3D world coordinates, with coordinate axes $X1$, $X2$, and $X3$. The point P is described in these coordinates. O corresponds to the location of the camera aperture, also called the *focal point*. $X3$ is the viewing direction of the camera, the *optical axis*, and $X2$ is parallel to the image plane. Thus, $X3$ intersects the image plane perpendicularly at a point R, called the *principal point*. The *focal length* f is the distance between the focal point and the principal point along the optical axis. R is the origin of the 2D image plane coordinates, with axes $Y1$ and $Y2$. Light rays originating from point P travel through O to intersect with the *image plane* at a point Q. This would form an upside-down image on the image plane.

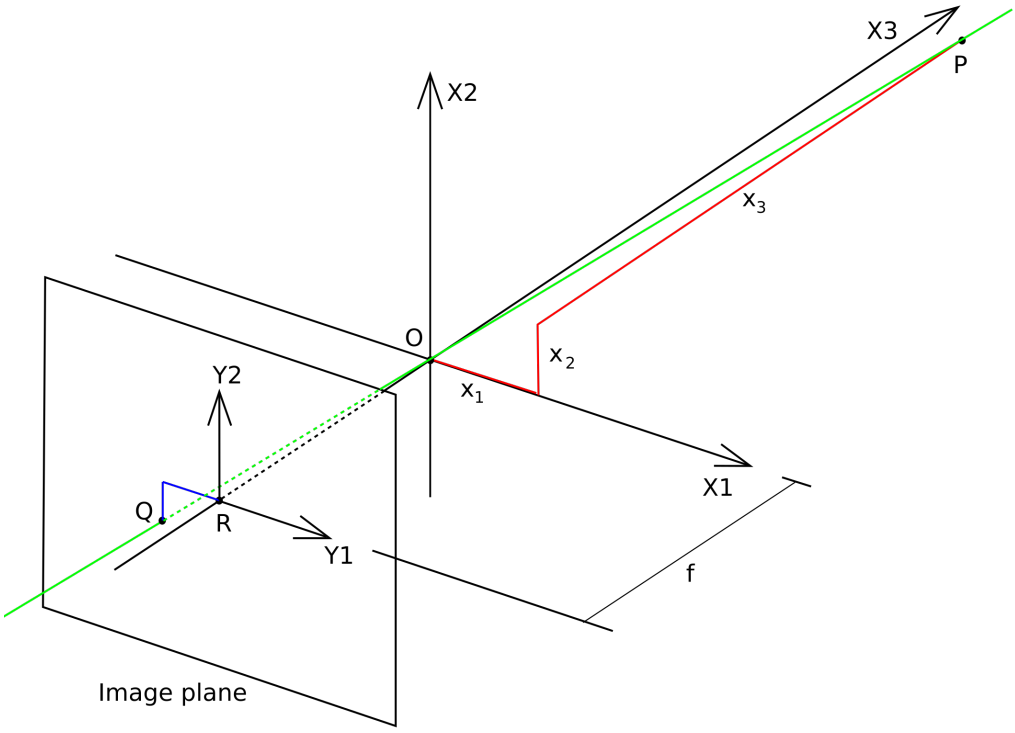

Figure 2.2: The geometry of the pinhole camera. This diagram describes how a point P in 3D world coordinates is projected onto an image plane, creating a point Q in 2D planar coordinates. From [17].

The principal point is also referred to as the centre of projection of the image [18]. That is the property used to determine its location in Zhang's algorithm. The principal point will also be an image's centre of distortion [19].

2.2 Hand-eye calibration

In the previous part, we explored the use of the pinhole camera model to perform camera calibration. However, that is not the only task required for real-time 3D robotics. Tsai describes a 'calibration trio': hand, eye-to-hand (or hand-eye), and eye (or camera) calibrations [20]. See figure 2.3 for an illustration of this.

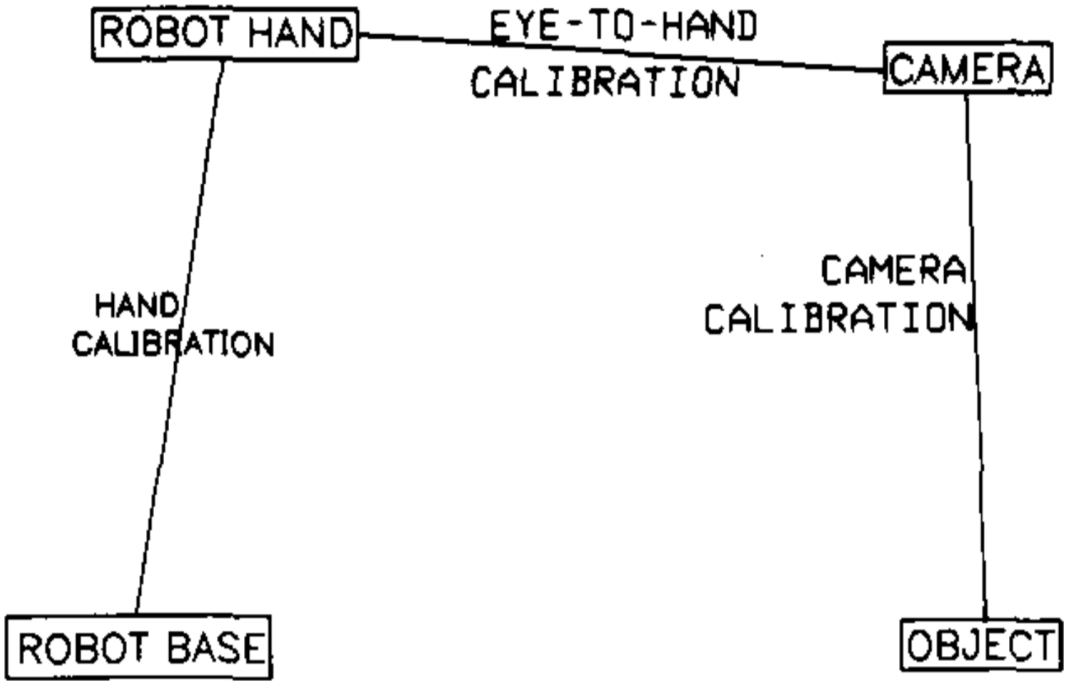

Figure 2.3: The 'calibration trio'. To obtain the 3D position and orientation of an object relative to the robot world base, it is necessary to perform three calibrations; namely, robot hand, eye-to-hand, and eye (camera) calibration. From [20].

Hand-eye calibration is a technique to describe the 3D position and orientation of a robot's 'hand', in the coordinate system of the 'eye'. In our case, the 'hand' is the laparoscope's sheath, equipped with an optical tracking collar . The 'eye' is the laparoscope's camera, which is marked by an electromagnetic (EM) tracking marker. So, the goal is to determine the position and orientation of the camera relative to the tracking markers. The method developed for the SmartLiver project is detailed in this paper [21].

A basic configuration for hand-eye calibration for augmented reality (AR) proposed by Lee [13] is laid out in figure 2.4. It consists of using tracking cameras and markers to be tracked, and then determining the 4x4 rigid transformation ma-

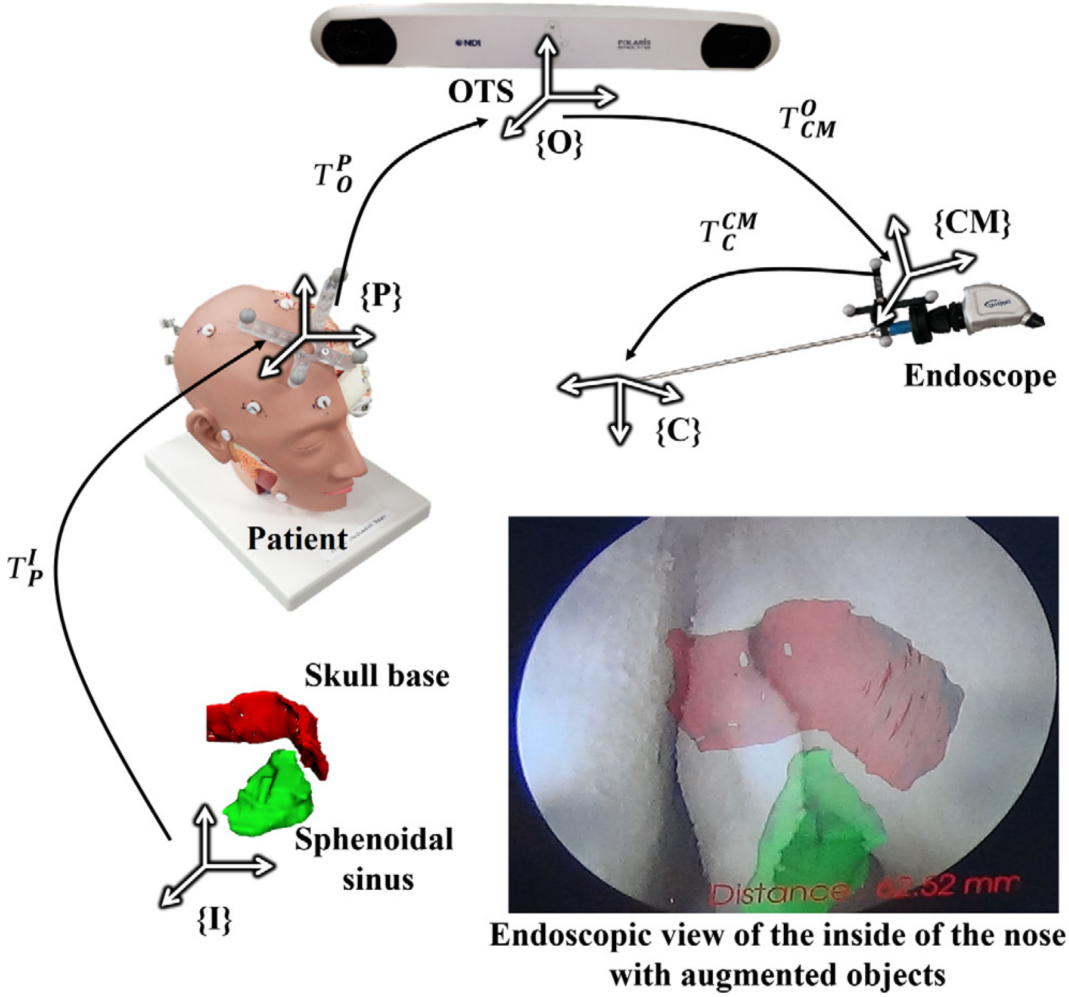

Figure 2.4: Basic configuration for realizing AR. Through the transformation between the coordinate systems of the camera C and image I, both the skull base (red) and sphenoidal sinus (green) are transformed to the coordinate system of the endoscope. Finally, virtual objects are projected into the endoscopic image using the intrinsic parameters of the endoscope. From [13].

trices required to get a given object's pose and position in the camera's frame of reference. Lee also informs us that moving the calibration pattern rather than the camera is more effective, and it also favours a sequential approach to solving the transformations.

2.3 Endoscopic calibration literature

In this section, we will explore several implementations of calibration procedures for endoscopic surgery.

De Buck proposes an AR system to assist surgeons during laparoscopic surgery, with optical tracking of both the patient and the laparoscope [22]. It focuses specifically on the problem of identifying the ureter behind the peritoneal wall. The structures from pre-operative images are used to augment laparoscopic view and reveal hidden structures. It proposes to solve calibration and registration at the same time to reduce the number of steps needed in an operating room (OR). Hand-eye calibration is based on Tsai's procedure [20]. It also manages to correct the radial distortion in laparoscope images in real-time. While distortion is generally undesired, surgeons tend to become accustomed to it. Therefore, there is a choice between distorting the virtual images, or de-distorting the real image (this paper chose the latter). The paper reports errors of ± 4 pixels, although it admits the need for more testing. The proposed application shows a rendering of the ureter in the laparoscope image, which would otherwise be hidden. Note that this a fairly early attempt at AR (2001), and is improved upon by later work.

A year later, Shahidi proposed a new image-guided surgery method: 'image-enhanced endoscopy' [23]. It allows for varying the transparency of tissue in the virtual image. Similarly to the previous paper, pre-operative images are used to enhance the surgeon's view. It states that it improves on previous image-guidance systems which showed the position of the instrument on pre-formatted images, rather than enhance real images. Optical tracking of the endoscope, but not the patient, is in place here. Radial distortion is also accounted for with a 'fast, automatic, and reliable' routine, which involves using a mount for the endoscope (additional equipment). It uses a custom centre-dot calibration board. Several radial distortion correction options are considered: none (fast, inaccurate), non-linear (slow, accurate), and linear (compromise); however, the paper concludes that non-linear correction is necessary, and can be rendered in real-time. With that method, they obtained a mean projection error of 0.5 mm from up to 25 mm from the camera and

1.0 mm from up to 45 mm. Tracking errors are similar; the author notes that image and projection errors could be due to tracking rather than calibration. The calibration routine involves taking several snapshots along the length of the endoscope's calibration mount.

In 2006, Wengert et al. released a research paper on the same problem, but with a focus on OR-specific challenges [24]. The paper proposes a calibration method, with a view of it being applied for enhanced endoscopy (navigation and 3D visualisation). The factors that they paid attention to were: inhomogeneous OR lighting, limited calibration time, resilience of the calibration material from sterilisation, variation of intrinsic parameters in different mediums (such as water and air). Wengert explores established calibration methods but judges that, although easy to implement, they do not address the specific challenges of surgery. Shahidi's method is also deemed too prone to distortions. Thus, Wengert designed a new calibration pattern,' a grid of dots and two special marks for identifying the local coordinate system', that is made to be sterilisation-resistant. They use Heikkilä's camera model to perform calibration [25]. 15 images are recommended, from different view points, to calibrate. The endoscope's light source is sufficient, although additional light sources reduce mean backprojection error from 0.25 px to 0.2 px on a 10 mm laparoscope (which is found to perform much better than the standard 3.8 mm). They estimate the calibration time to be less than a minute with a progressive scan camera. Intrinsic parameters were found to change in media other than air, but the author recommends further investigation in that area. Hand-eye calibration was performed using Tsai 's method [20].

A 2012 paper by Melo, Barreto, and Falcão seeks to implement a GPU and CPU hybrid solution to achieve camera calibration and real-time image distortion correction for high definition images [26]. They achieved root mean square (RMS) reprojection errors between 2 and 5.6 px. In developing this system, they also created a new calibration method. In fact, Barreto had published a paper for this method in 2009 [27]. His method has the particular advantage of being easy to implement, requiring only a single image. Thus, it is well suited to endoscopy and AR

enhancement. It uses an existing Matlab toolbox [28] which is based on Zhang's work [12]. This is the most similar to what our research is attempting to achieve, albeit using different methods.

More recently, Schwalbe proposes a calibration technique especially suited to intraoperative laparoscope calibration, with a focus on simplicity and resistance to sterilisation [29]. The study is particularly interesting as they brought their method to clinical evaluation at the University Hospital of Bern. In their labs, they managed to successfully calibrate in all 10 of their tests, and obtained an average displacement error of 0.52 mm over 12 evaluation images. In the OR, their calibration unit was sterilised without damage, however one of five calibration attempts failed, with an average calibration time of 38 s. These results show that there is still room for improvement and research in camera calibration for augmented endoscopy, as well as highlight the difference between a successful lab implementation and effectiveness in hospitals.

Chapter 3

Neural Networks

This chapter explores some deep neural networks theory, as well as the specific features of neural networks used in this project. Sections 3.1 and 3.2 use information from Goodfellow's book on Deep Learning [30].

3.1 Neural Network Basics

An artificial neural network is a layered system used to predict outputs $\boldsymbol{y}$ for given inputs $\boldsymbol{x}$ through some learning mechanics. Between the input and output layer, there exist hidden layers $\boldsymbol{h}$ which take an input from a previous layer, transform it in some manner and feed it to the next layer. During learning, the network's output is compared to user-defined *label* or *ground truth* values $\tilde{\boldsymbol{y}}$, and then the hidden layer's *weights* are updated accordingly.

3.1.1 The Fully-Connected Layer

An essential building block of neural networks is a type of network layer called *dense* or *fully-connected*. Its purpose can be interpreted as finding linear and non-linear relations between features. It is typically used as the output layers of a network. Figure 3.1 is an example of a network with a single fully-connected layer. It is described mathematically by equation 3.1.

$$\boldsymbol{y} = \sigma\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right) \tag{3.1}$$

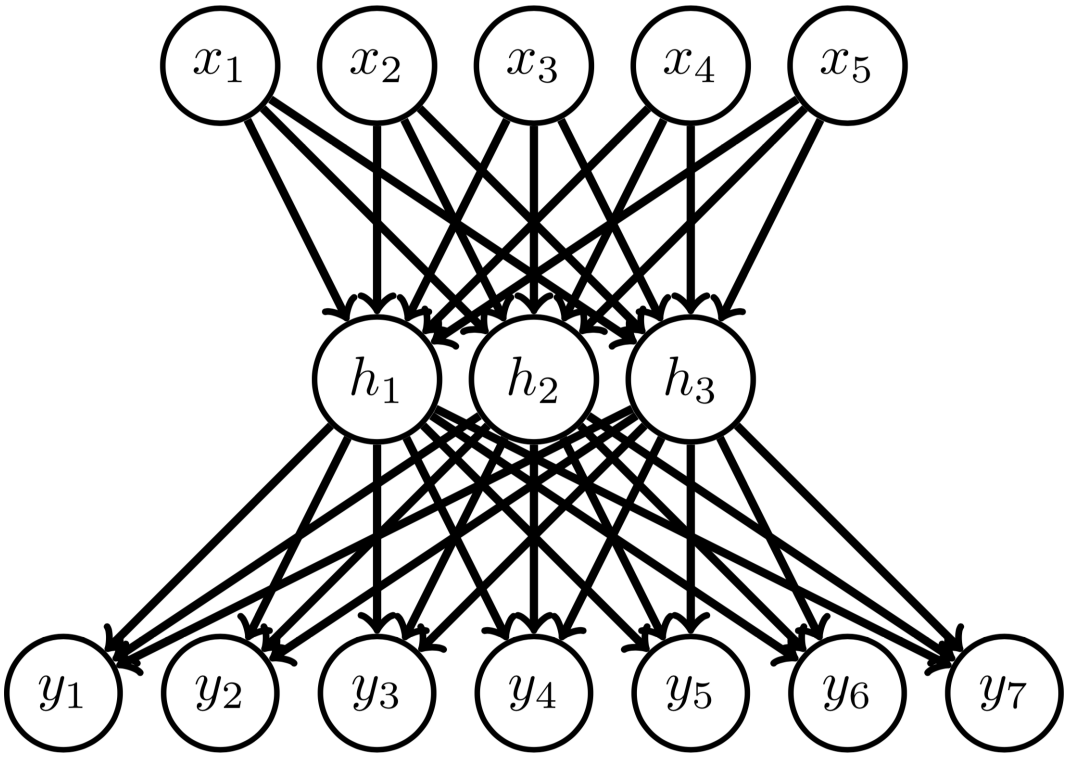

Figure 3.1: A simple one (hidden) layer neural network. Each line corresponds to an input layer $\boldsymbol{x}$, a hidden layer $\boldsymbol{h}_1$, and an output $\boldsymbol{y}$, respectively. From [31]

3.1.2 Feed Forward Neural Net

A standard neural network with L layers is equivalent to a function defined as such:

$$f(\boldsymbol{x}|\mathcal{W}) = \sigma_L(\boldsymbol{W}_L\boldsymbol{h}_{L-1} + \boldsymbol{b}_L) \tag{3.2}$$

Where $\boldsymbol{x}$ is the network's input vector; $\mathcal{W}$ the set of weights in the various network layers. The hidden layer $\boldsymbol{h}$ for a given layer l is defined as:

$$\boldsymbol{h}_l = \begin{cases} \sigma_l(\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{b}_l), & \text{for } l \in [2, L-1] \subset \mathbb{N} \\ \sigma_1(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1), & \text{for } l = 1 \end{cases} \tag{3.3}$$

Where σ_l, $\boldsymbol{W}_l$, $\boldsymbol{b}_l$, and $\boldsymbol{h}_l$ are the *activation* function (see section 3.1.3), *weight* matrix, *bias* vector, and *hidden layer* for a given layer l, respectively. Note that $\boldsymbol{h}$ is defined recursively, so that there are L hidden layers in the network.

Equation 3.1 is a special case of equation 3.2 where $L = 1$. For a hidden layer

with N inputs (or outputs from the previous layer) and M outputs, the matrix $\boldsymbol{W}$ will be of size $M \times N$, and $\boldsymbol{b}$ a vector of length M. The components of $\boldsymbol{W}$ and $\boldsymbol{b}$ are the trainable weights of the network.

A network can be defined concisely as follows, where $\boldsymbol{y}$ represents its output vector:

$$\boldsymbol{x} \longrightarrow \boldsymbol{h}_1 \longrightarrow \ldots \longrightarrow \boldsymbol{h}_L \longrightarrow \boldsymbol{y} \tag{3.4}$$

A basic application of a neural network for function regression can be seen in figure 3.2. It is a 2-layer network using only fully-connected layers. The layers also feature an *activation function*.

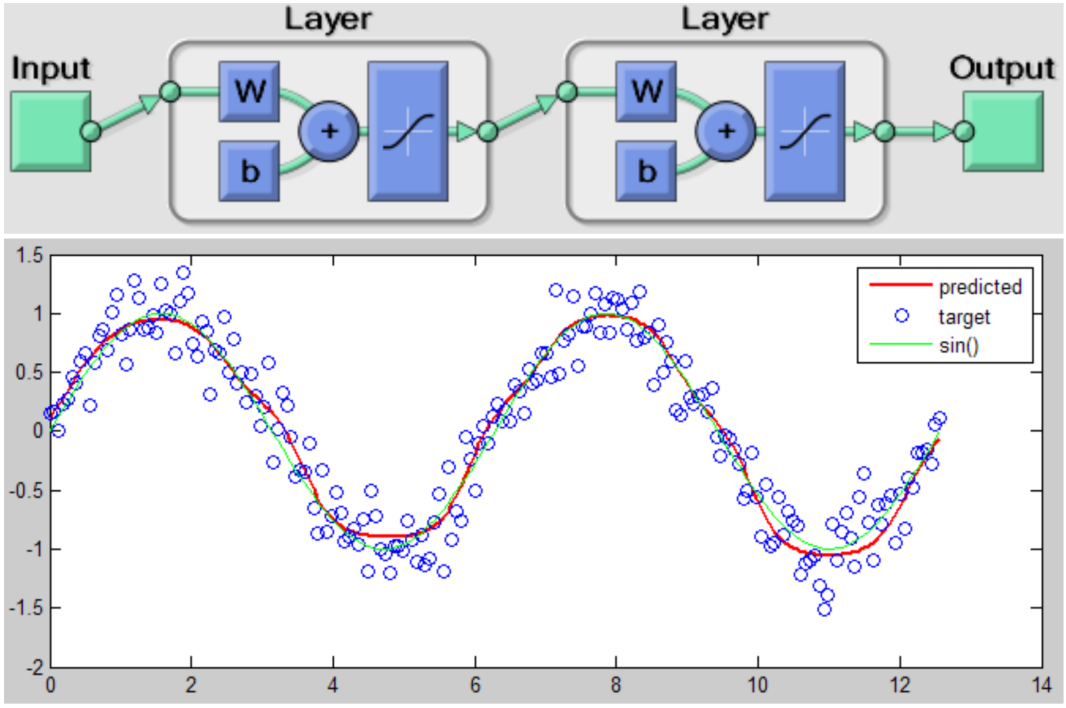

Figure 3.2: A neural network learning the sine function. The architecture of a neural network is shown (top). There is also a graph displaying the target/label training values, the network's predicted function, and the actual sine function (bottom). From [31].

3.1.3 Activation Functions

Activation functions are present at the end of each hidden layer of a network. Their purpose is to give non-linear properties to a network layer, since a layer is otherwise a linear combination of weights and input values. Thus, the activation function is a necessity for a network attempting to predict a non-linear function. There are several types of popular activations.

3.1.3.1 Sigmoid

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.5}$$

The sigmoid activation function mimics the behaviour of neurons; it 'activates' when inputs go above a certain threshold. Indeed, its values are between 0 and 1 and has a sharp gradient about the origin. It has fallen out of favour for its slow convergence, tendency to saturate gradients, and because it is not zero-centred.

3.1.3.2 Tanh

$$f(x) = tanh(x) \tag{3.6}$$

The tanh solves some of the problems that the sigmoid activation has by being zero-centred. However, like sigmoid, it also suffers from the vanishing gradient problem during backpropagation (or any other techniques which need to calculates gradients). This is because it constrains a wide range of input values to a relatively small range (-1 to 1), causing small gradients. If the gradients go to zero, backpropagation fails as no information can 'go through' the zero-gradients.

3.1.3.3 ReLU

$$f(x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \tag{3.7}$$

Rectified Linear Units (or ReLU) are a solution to the vanishing gradient problem. However, they can be prone to 'dead neurons', which is when a neuron is no longer activated regardless of inputs.

3.1.3.4 Leaky ReLU

$$f(x) = \begin{cases} x & \text{for } x \geq 0 \\ \alpha x & \text{for } x < 0 \text{ with } \alpha \in (0,1) \subset \mathbb{R} \end{cases} \tag{3.8}$$

The dead neuron problem is solved with a Leak ReLU activation, since negative values don't disappear, but are simply reduced by a coefficient α. Although simple, this activation has become the standard in many modern networks.

3.1.4 Loss Functions

To train a network, an optimisation method updates its weights in a manner which minimises a loss function, also called an *objective function*. We will focus on loss functions for regression problems (as opposed to classification).

3.1.4.1 Mean Squared Error (MSE)

$$E = \frac{1}{N} \sum_{i=1}^{N} \left(\tilde{\boldsymbol{y}}^i - \boldsymbol{y}^i \right)^2 \tag{3.9}$$

Where N is the total number of output vectors. Equation 3.9[1] represents a standard regression loss. A feature of MSE loss is that it places a very large penalty on high errors. This is an advantage when this behaviour is wanted, but it tends to give a lot of weight to outliers, which could be a problem if the training data has many of them.

3.1.4.2 Mean Absolute Error (MAE)

$$E = \frac{1}{N} \sum_{i=1}^{N} \left| \tilde{\boldsymbol{y}}^i - \boldsymbol{y}^i \right| \tag{3.10}$$

MAE loss is more robust to outliers, which can be desired or not depending on the network; this is a good alternative to MSE.

3.1.4.3 Mean Absolute Percentage Error (MAPE)

$$E = \frac{100}{N} \sum_{i=1}^{N} \left| \frac{\tilde{\boldsymbol{y}}^i - \boldsymbol{y}^i}{\tilde{\boldsymbol{y}}^i} \right| \tag{3.11}$$

MAPE is similar to MAE, but it avoids over-representing larger values in a multi-value regression task.

[1] The exponents on $\tilde{\boldsymbol{y}}$ and $\boldsymbol{y}$ are indices

3.2 Convolution

3.2.1 The Convolution Operation

Convolutional networks are a type of deep neural networks which employ convolution operations. It is applicable on data which have a grid-like features, such as an image which is a 2D grid of pixels. The convolution operation is best explained visually, with figure 3.3 serving as an illustration. The kernel is a matrix, usually square of $k \times k$ dimension, which 'slides' over the $N \times N$ input matrix. The convolution is a series of dot products between the kernel and every possible $k \times k$ sub-matrix in the input matrix. The order of operations is from left to right and top to bottom. The results of the dot products are stored in an output matrix. The values given in the input are either the network's input or the output of the previous layer. The kernel's values, (α, β, γ, and ϵ in figure 3.3) are learned parameters of the network.

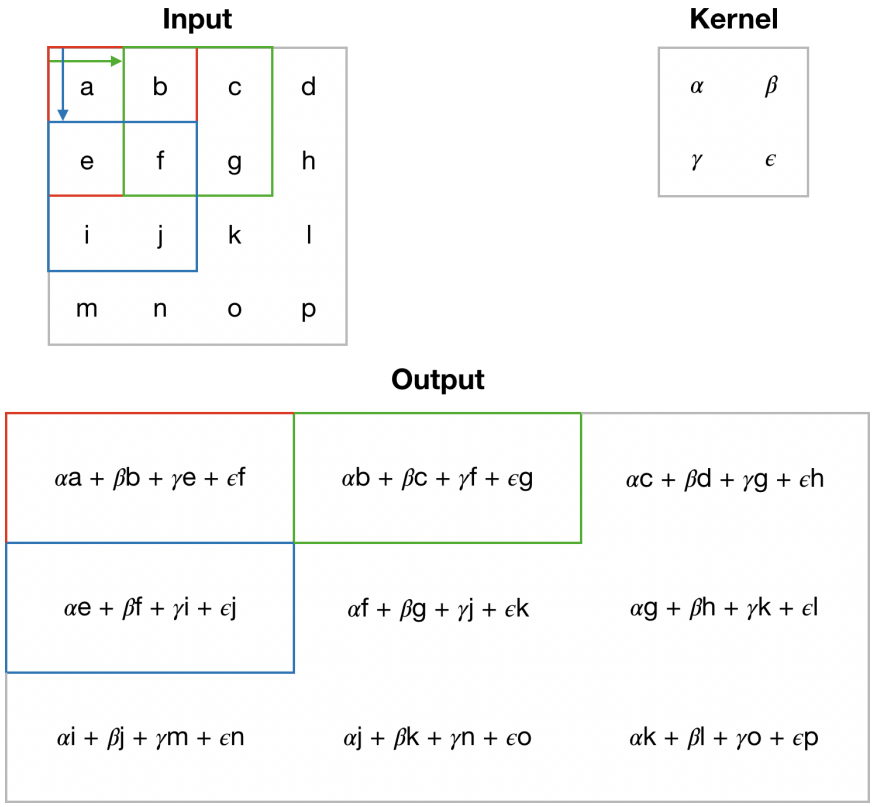

Figure 3.3: A simple convolution operation on a 4-by-4 input by a 2-by-2 kernel. Colours match the sub-matrix that the kernel operates on and the corresponding result in the output matrix.Three examples were colour-coded for readability. Arrows indicate the direction the kernel 'slides' towards in each example.

3.2.2 Convolution Mechanics

3.2.2.1 Padding

In the case of a 2D $k \times k$ kernel sliding over an $N \times N$ 2D input, the output will be an $N \times N$ matrix such that $M = N - k + 1$. With successive convolution layers, the outputs would get increasingly small. This can be avoided with the use of *zero padding*: surrounding the input matrix with null values. The number of columns/rows of null values added is given by the parameter p. The types of padding are:

- *Valid* padding is when $p = 0$, such that $M = N - k + 1$.
- *Same* padding is when $p = \frac{k-1}{2}$, such that $M = N$.

With same padding, the output resolution is conserved. Padding in general allows for control of resolution.

3.2.2.2 Stride

Stride is a parameter, given by s, which determines by how many rows or columns the kernel is displaced every time it 'slides'. A 'strided' convolution effectively decreases the dimensions of the output, as the output size will be $M = \frac{N+2p-k}{s} + 1$

Strided convolution reduces the sampling rate of the input. Formally, this is referred to as *downsampling*. Its purposes are to reduce resolution to allow for faster processing, and to achieve a greater local invariance.

3.2.2.3 Pooling

Pooling is another downsampling technique. It is similar to convolution, except that instead of performing a dot product, it 'pools' the data together according to some operation. Striding can also be applied to pooling. Popular pooling operations include:

- *Average pooling* computes the average value over the kernel's receptive field.
- *Max pooling* computes the largest value over the kernel's receptive field.
- *Global pooling* computes the largest value over the entire input space.

3.2.3 Gradient Instability

Section 3.1.3 already touched on the problem of 'vanishing gradients', and proposed ReLU activations as a way to alleviate the problem. These occur with gradient-based optimisation methods (backpropagation), such as *stochastic gradient descent* with momentum and *Adam*. Vanishing/exploding gradients will occur when weights are too small or too big. This section explores other ways of reducing the chances of this happening.

3.2.3.1 Dropout

Dropout consists of randomly setting a fraction of input units to 0 at each update during training time, which helps prevent overfitting and unstable gradients.

3.2.3.2 Weight Initialisation

Initialising weights from a zero-mean normal distribution was found to be an effective way of reducing gradient instability. For shallower nets, it is sufficient to initialise all layers with a variance of 0.01. However, deeper nets will tend to get vanishing gradients.

Glorot's method is to adapt variance depending on the layer in order to preserve gradient magnitude [32]. This will be effective for nets up to 20 layers, like VGG (see section 3.3.1), but additional methods are required to construct a deeper network.

3.2.3.3 Batch Normalisation

In 2015, Ioffe et al. identified *internal covariate shift* as a cause of gradient instability [33]. This is when the distribution of each layer's inputs changes during training. To solve this, they propose *batch normalisation*. Networks would have a layer, which normalises its inputs to zero mean and unit variance.The BatchNorm layer is typically placed after each convolution layer, before the activation. A requirement for batch normalisation is to use batched training. Means and variances can be set differently for different batches in order to add randomness and regularise. If a net uses BatchNorm, other regularisation techniques, like 'dropout' and careful initialisation, may not be required.

3.2.4 Pattern Recognition

An obvious advantage of the kernel is that the same few trained parameters can be applied over a whole image (in the 2D case), which is quite efficient. A kernel can be interpreted as a filter which recognises a pattern. In a convolutional network, kernels are trained to recognise increasingly detailed pattern.

Images are typically stored in 3 channels, one for each primary colour (usually Red, Green, Blue). Each channel has a matrix with the dimensions of the image, and its components are 8-bit integers which determine the weight of the channel's colour in the overall image. A feature map is a 3D kernel which has a depth of 3, spanning all channels. A typical choice is a 3x3x3 feature map.

Zeiler's paper on visualising convolutional networks sheds some light on the blackboxes which are deep neural nets [34]. They trained a model on ImageNet [9], a popular data set for object recognition. Figure 3.4 shows the feature visualisation of a trained model. More precisely, it displays the top 9 activations from some of the model's feature maps, which are chosen at random. An observation that can be made is that feature maps are specialised in recognising some particular feature, and that the recognised feature becomes more specific (less fundamental) in later layers. The feature maps evolve towards recognising objects as a whole, rather that just their components. Also, the network gives more weight to an object's distinctive features.

Figure 3.4: Visualization of features in a fully trained model. For layers 2-5 they show the top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using our de-convolutional network approach. Their reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map they also show the corresponding image patches. From [34].

3.3 Convolutional Networks

This section explores three convolutional neural network which became popular following their success in the ImageNet [9] challenge. The ImageNet challenge is a competition on Kaggle where contestants attempt to build the best-performing neural network trained on the ImageNet database. ImageNet is a large labelled object detection data set. These networks are used as a base for the architectures developed in this thesis, and are initialised with their trained ImageNet weights.

3.3.1 VGGNet

A common objective when building a neural network is to make it as deep as possible. A deeper network means more non-linearities, since each layer can be regarded as a linear classifier followed by a non-linear activation. So, the deeper a model, the more complex the patterns it is capable of identifying. However, depth is limited by a number of factors, such as lowering of resolution following pooling layers, unstable gradients, and computational cost of training.

VGGNet [3] is the nickname of the network proposed by Simonyan and Zisserman of Oxford's Visual Geometry Group (VGG) in 2015. They solve this loss of resolution problem by stacking several convolutional layers between pooling layers, using same padding, and a stride of $s = 1$. They use many stack of small 3×3 convolutional layers. Using these small kernels allows for covering a large receptive field with less parameters than a single layer with a larger single kernel would. VGGNet uses max pooling layers between its stacks of convolutional layer, ReLU activations, and applies both dropout and L2 penalties for regularisation. The 19-layer version is shown in figure 3.6.

3.3.2 ResNet

Kaiming He et al.'s ImageNet model, called ResNet [35], implements a 'residual learning framework'. ResNet is much deeper and has better ImageNet results than VGG. Its best performing model has 101 layers. After a certain depth, neural nets would tend to have a higher training error than their shallower counterparts. ResNet solves this with adding layers from a shallower network with *identity* shortcuts skipping several layers at a time. Figure 3.5 shows a basic residual connection. The output of the connection is the residual mapping $\mathcal{H}(\boldsymbol{x}) := \mathcal{F}(\boldsymbol{x}) - \boldsymbol{x}$. Where the stacked layers fit a mapping $\mathcal{F}(\boldsymbol{x})$, and $\boldsymbol{x}$ is called the 'identity mapping'. Kaiming He argues that this network can perform no worse than the shallower networks since if the identity performs better than the layers it skips, then those layers can simply have their weights set to zero. Figure 3.6 shows how residual connections can be added to a standard convolutional network for better results.

ResNet uses BatchNorm for regularisation, and ReLU activations. In its deeper versions (50+ layers) ResNet uses a 'bottleneck' layer, that is a convolutional layer with a 1×1 kernel which operates across channels. This serves to reduce the number of channels before slower 3×3 layers. It is a cheap way to increase depth in a network. They form blocks where these two bottleneck layers surround a regular convolutional layer.

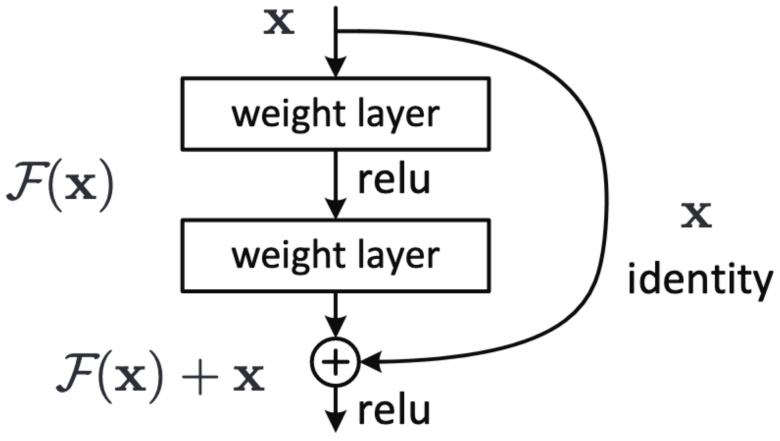

Figure 3.5: Residual learning: a building block. From [35].

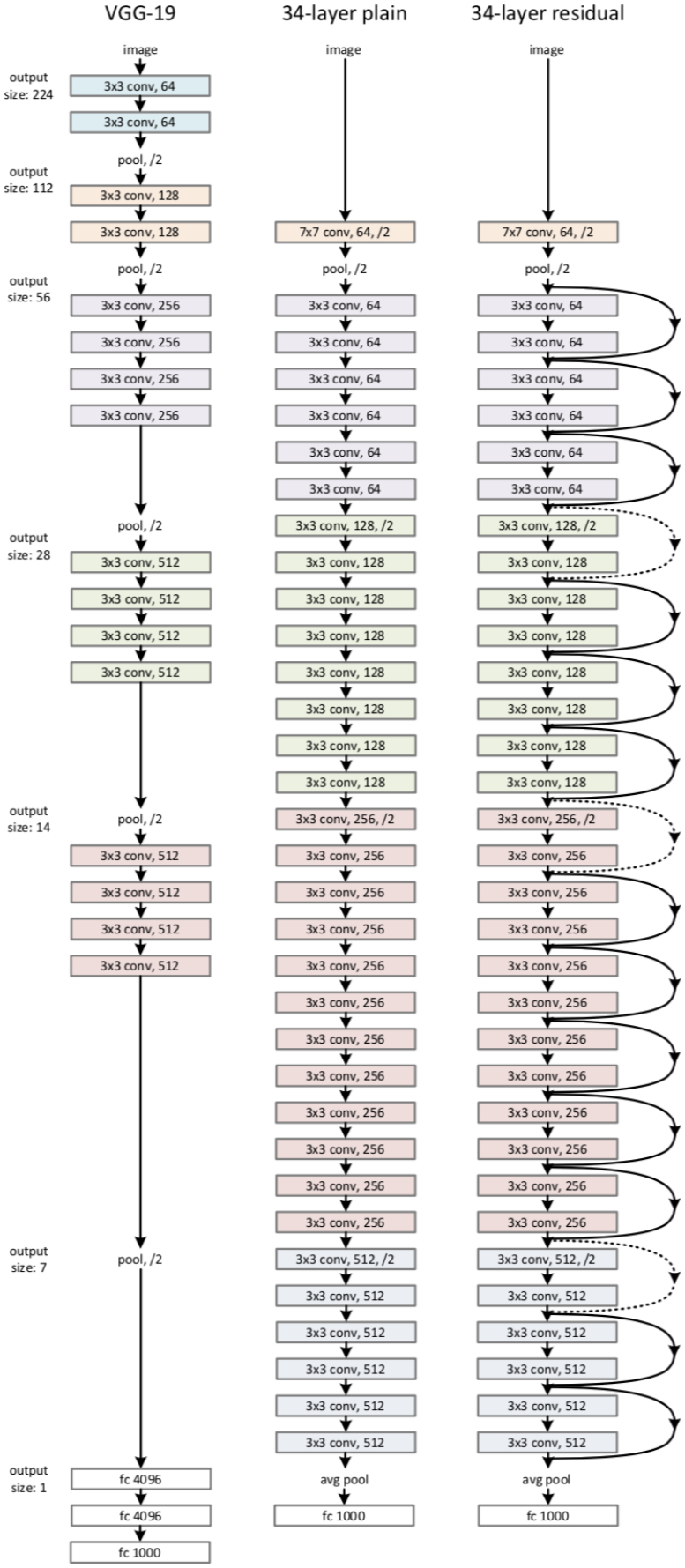

Figure 3.6: Example network architectures from ImageNet. From left to right: the VGG-19 model, a 19-layer implementation of VGG (19.6 billion flops), serving as a reference; a standard 34-layer network (3.6 billion flops); a residual version of the 34-layer network (3.6 billion flops), The dotted shortcuts increase dimensions. From [35].

3.3.3 DenseNet

The architecture of DenseNet [36] is such that each layer is connected to each layer that is in front of it. While a traditional L layer ConvNet will have L connections, one between each adjacent layers, DenseNet will have $\frac{L(L+1)}{2}$ connections. Unlike ResNet, where layers were combined with summation, in DenseNet combined layers are concatenated. Thus, the number of input channels will grow with network depth. Figure 3.7 shows how a single dense block is organised, featuring batch normalisation and ReLU activations. Figure 3.8 features the integration of the dense blocks in the wider DenseNet with an image classification task as an example.

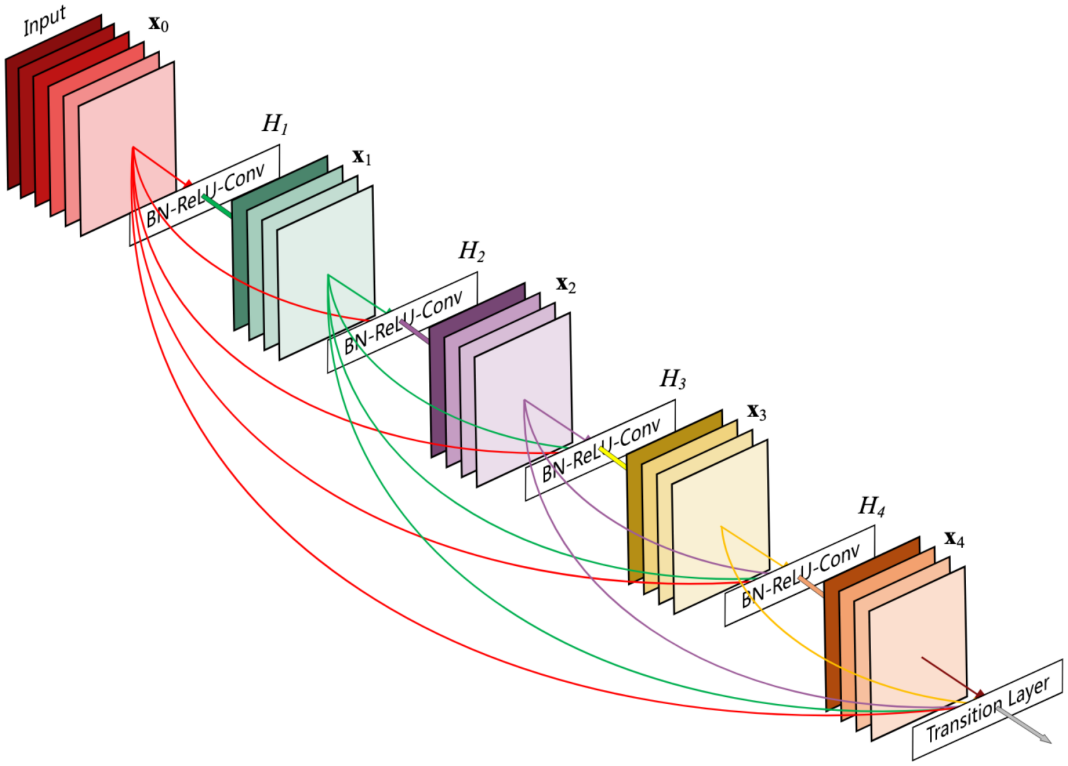

Figure 3.7: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input. From [36].

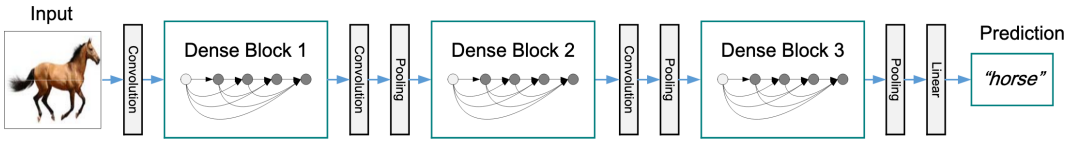

Figure 3.8: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling. From [36].

Chapter 4

Method

This chapter describes the methods used for both generating data, and designing and training networks using the generated data set. The code for each of these parts is available on github:

`https://github.com/raphael-p/weiss-data_generation`

`https://github.com/raphael-p/weiss-networks`

4.1 Data Generation

4.1.1 Simulating Real Calibration Images

For the input of our neural network, we generated calibration images. The purpose of the image is to simulate the existing calibration setup of the laboratory at the WEISS. As mentioned previously, the hand-eye calibration method used at the SmartLiver project is detailed in a 2016 paper by Thompson et al. [21].

A surgical camera is calibrated by placing a calibration board in front of the camera at varying angles and positions. Figure 4.1 is an example of a calibration image. We simulated this by creating a plane of the same dimension as a calibration board, 1920 by 1080 pixels (the resolution of our laparoscopes), and importing it into a vtk [37] application, where the checker pattern was overlaid onto it. This was done based on methods from a python package developed for SmartLiver under the SNAPPY software project [38]. Modifying the extrinsic matrix allows us to randomly rotate the plane in three dimensions using normally distributed small rotation angles, and randomly translate the board about the centre of the image. See

figure 4.2 for an example of a generated image, and figure 4.1 for a comparison to a real calibration image.

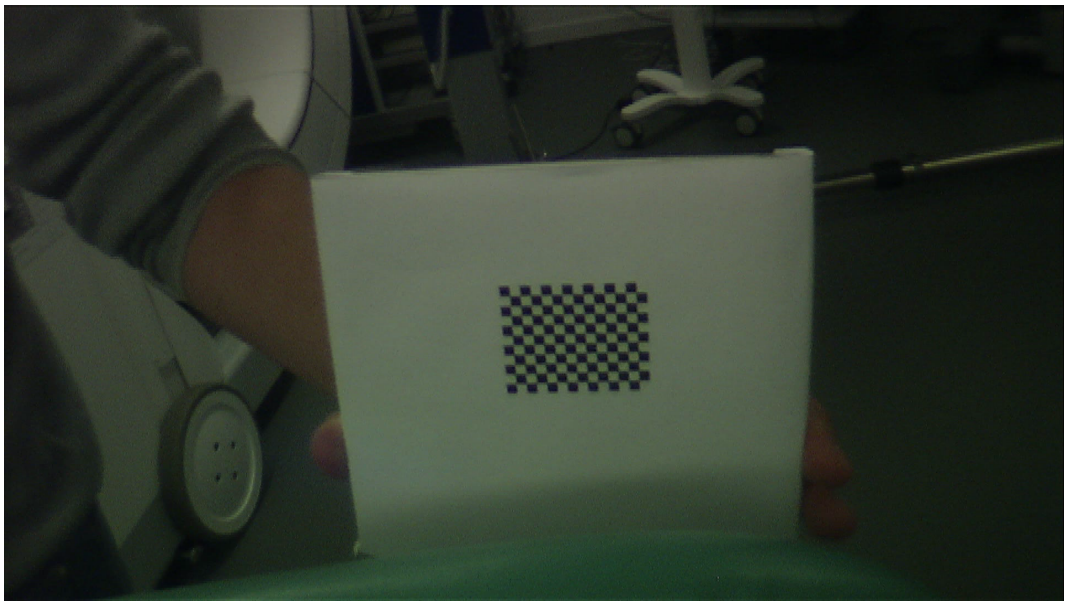

Figure 4.1: A calibration image taken at WEISS's lab. The image is taken with a laparoscope, the pattern is a 42×33 mm rectangle with 3 mm squares.

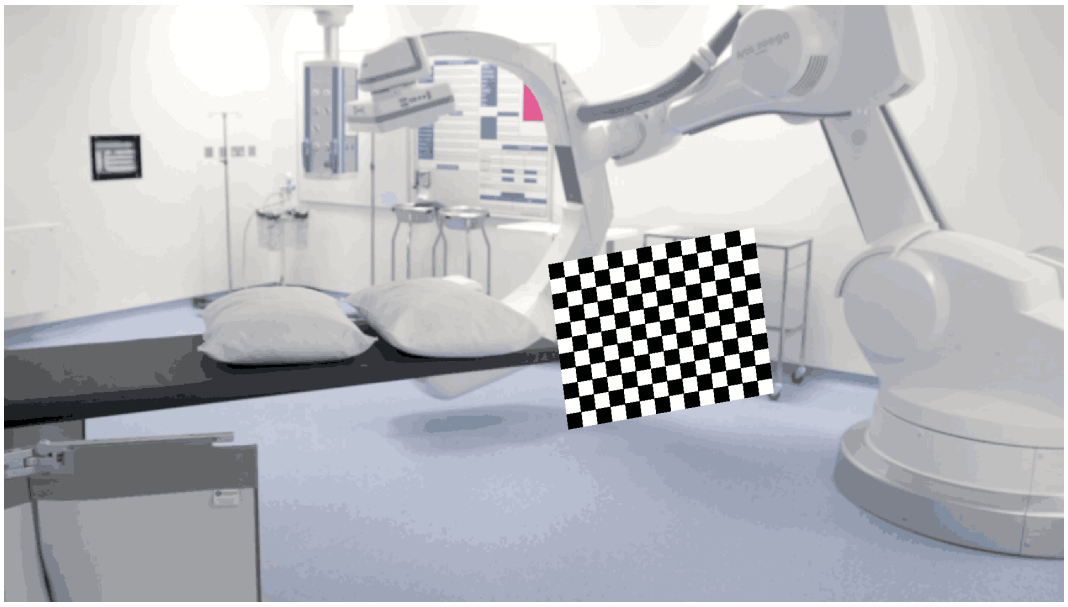

Figure 4.2: A generated image for training. The backgrounds for image generation are all of operating theatres, with or without personnel. The foregrounds are a randomly rotated and translated calibration board.

4.1.2 The Datasets

The intrinsic matrix parameters consist of the scaled focal lengths α and β, and (u_0, v_0), the coordinates of the principal point. We used an actual intrinsic calibration matrix for the lab's laparoscope, shown in equation 4.1. The skew factor of the image axes, γ, is set to zero. The generated images used normally distributed sample around these values, with a standard deviation of 20 %.

$$A = \begin{bmatrix} 1740.660258 & 0.000000 & 913.206542 \\ 0.000000 & 1744.276691 & 449.961440 \\ 0.000000 & 0.000000 & 1 \end{bmatrix} \tag{4.1}$$

Two different data sets are generated, one where the principal point was fixed, and one where it was not. The reasoning for this is doubts over whether the network was capable of predicting the principal point. The challenges with determining the principal point are two-fold:

- The principal point is the image's centre of perspective [18]. However, since the background image was taken from a different camera it would not have the same principal point as the virtual camera of the generated image. On the other hand, the network could be able to determine the image's centre of perspective from the board alone.
- Another helpful property of the principal point is that it is an image's centre of distortion [19]. However, the images we generated are not distorted, so this property cannot be taken advantage of by the network.

An additional smaller set of about 4000 images was generated to make some predictions using trained models and compute some evaluation statistics to compare them.

4.2 Neural Network

The neural networks developed for this project have a straightforward structure:

1. An input layer
2. A 'base' network that was pre-trained on ImageNet
3. A flattening layer
4. 2 sequential dense layers with LeakyReLU activations
5. A split into different 'heads', each with a dense layer, a LeakyReLU activation, and an output dense layer

The output of the last 'body' layer (dense with 512 outputs and LeakyReLU) is copied to each of the heads' first layer. The network's architecture can be seen in figure 4.3.

The output 'heads' are:

1. *focal*: the focal lengths along each image axis; 2 outputs
2. *principal*: the coordinates of the principal point; 2 outputs
3. *rotation*: the flattened 3×3 rotation matrix of the calibration board; 9 outputs
4. *translation*: the 3 dimensional translation of the calibration board relative to the center of the image; 3 outputs

The different 'base' networks that we use are:

1. *VGG19*: the 19-layer implementation of VGGNet
2. *ResNet50*: the 50-layer implementation of ResNet
3. *DenseNet201*: the 201-layer implementation of DenseNet

Each head in the network has a separate MSE loss, with no weights. The total loss of the network is simply a sum of these losses. Although we are only interested in predicting the *focal* and *principal* elements, the networks were found to perform

better when having to train for rotation and translation as well. We suspect that this is because the network is more 'well-rounded' in its analysis of images: it is easier to determine focal lengths and principal points if it also understands the board's position and orientation.

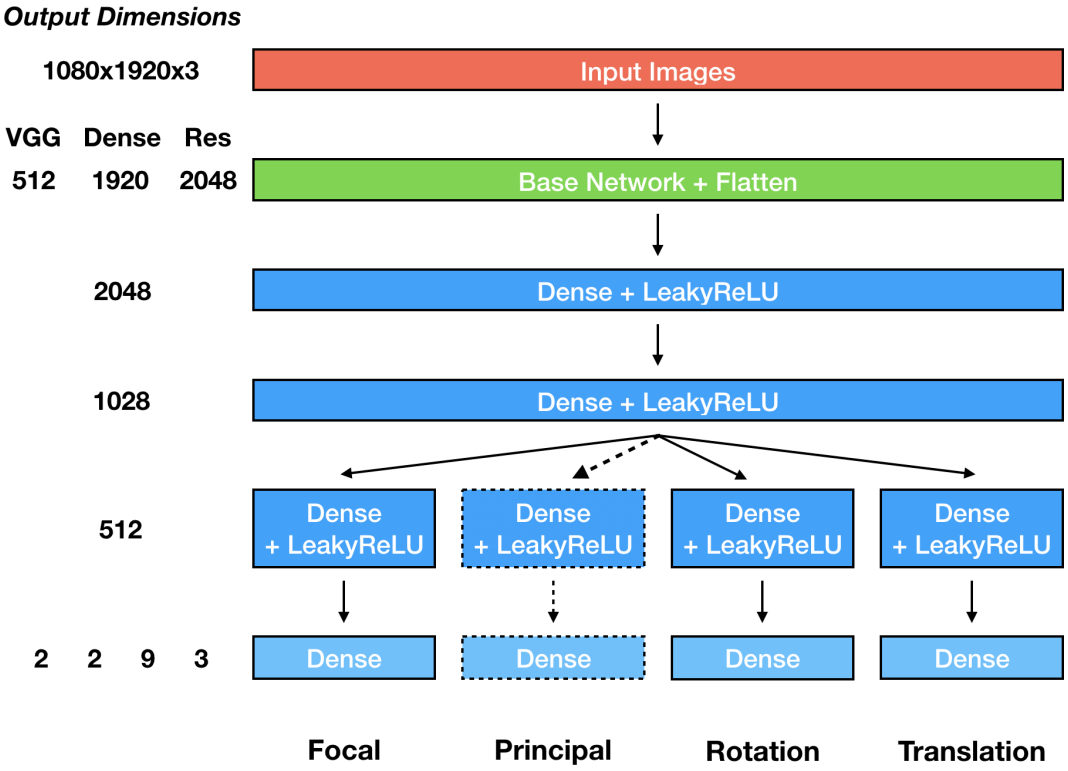

Figure 4.3: A blueprint of the neural networks trained in this thesis. The 'Base Network' is either be VGGNet, ResNet, or DenseNet. These are all pre-trained on ImageNet data. The dotted line represents an optional path, as we have trained networks with and without principal point prediction.

Although no weights are attached to the losses, the outputs of the *focal* and *principal* heads have larger absolute values than that of the two others. Therefore, their mean squared error loss will be much larger for the same relative error. The use of MSE loss does in practice give the *focal* and *principal* outputs more weight. This is a desired effect since it is what we are trying to optimise for. Although it is helpful that MSE eliminates the need for setting weights to each loss, this is not a sufficient reason. MAE was considered as well, but MSE performed better in initial tests of the model, so it was retained later on. MAE and MAPE are instead used as evaluation metrics for the model as they provide complimentary information.

As mentioned in section 4.1, we had doubts over the prediction of principal point coordinates, so the *principal* head was omitted in certain networks. When that was the case, we used input images with a fixed principal point.

Another alternative configuration we attempt, is to set several of the layers within the 'base' net as trainable. This results in a significantly more parameters to train. By default, the networks we trained had the base net set as untrainable.

For a summary of the settings used to train our models, see table 4.1.

Loss	MSE
Epochs	Automatic stopping after 3 epochs without improvement
Minibatch size	8
Optimiser	Adam [39]
Optimiser parameter: Learning Rate	0.0001
Optimiser parameter: β_1, β_2	0.9, 0.999
Optimiser parameter: ε	1×10^{-7}
Training data size	16 600
Validation data size	3500
Evaluation data size	3500
Prediction data size	3900
Initialisation	Glorot [32] uniform
Base layer initialisation	pre-trained ImageNet weights
Activation	LeakyReLU
Activation parameter: α	0.01

Table 4.1: Parameters common of the networks we trained.

Chapter 5

Results

In this section, we present some metrics regarding the performance of our networks on the prediction dataset. We also tested one of our model on a very small number of real images.

5.1 Comparing the base networks

We trained three networks which were identical except for the base network that was selected. These networks are already pre-trained on ImageNet data, and are set as untrainable. The dataset with fixed principal points (no principal point prediction) is used. These choices were made so these networks could be trained faster. We are interested in how well these networks performed at predicting the intrinsic matrix parameters from an image. As a reminder, here is the intrinsic matrix from equation 2.2:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where (u_0, v_0) are the coordinates of the principal point, α and β are focal lengths scaled across the image's u and v axes. Since we are using the fixed principal point dataset, u_0 and v_0 are not predicted here. We set $\gamma = 0$ as this corresponds to the settings of WEISS's laparoscope.

The results are shown below, in table 5.1.

Base nets	**VGG19**	**ResNet50**	**Dense201**
Epochs	36	137	142
SD on α	147.9	581.6	2903
SD on β	129.4	418.3	1021×10^1
MAPE on α	7.273 %	29.63 %	157.1 %
MAPE on β	6.324 %	19.70 %	593.1 %
MAPE on rotation	579.7 %	352.7 %	3432×10^2 %
MAPE on translation	402.8 %	216.1 %	3792×10^1 %

Table 5.1: Prediction results comparing different pre-trained networks. All networks were trained with the same settings (see table 4.1). 'SD' is the standard deviation of samples.

5.2 Variations of the network

With VGG19, a clear winner out of the previous networks, it is interesting to try out different variations of neural networks with VGG19 as a base. Four networks are tested here, with the results in table 5.2:

1. *Baseline*: the same as in section 5.1, with no principal point prediction and an untrainable base layer.

2. *Trainable*: the last two blocks (out of 5) of VGG19 are trained.

3. *Principal*: trained on a dataset with varying principal point, which the network attempts to predict.

4. *Combined*: the last two blocks of VGG19 are trained, and the principal point is predicted.

Name	**Baseline**	**Trainable**	**Principal**	**Combined**
Predicts principal point	No	No	Yes	Yes
Trainable base net layers	No	Yes	No	Yes
Base net	VGG19	VGG19	VGG19	VGG19
Epochs	36	9	99	30
SD on α	147.9	46.90	107.9	59.44
SD on β	129.4	54.09	100.0	64.31
SD on u_0	N/A	N/A	131.4	109.1
SD on v_0	N/A	N/A	67.94	52.91
MAPE on α	7.273 %	2.101 %	4.689 %	2.375 %
MAPE on β	6.324 %	2.380 %	4.258 %	2.776 %
MAPE on u_0	N/A	N/A	11.63 %	9.751 %
MAPE on v_0	N/A	N/A	10.89 %	8.517 %
MAPE on rotation	579.7 %	2815 %	1877 %	1493 %
MAPE on translation	402.8 %	988 %	4035 %	135.7 %

Table 5.2: Prediction results comparing different pre-trained networks. All networks were trained with the same settings (see table 4.1). 'SD' is the standard deviation of a sample. All networks are trained with VGG19 as their base.

5.3 Tests on real data

We recorded a video of a calibration pattern in the lab, and extracted several still-frame (like figure 4.2) to form a small dataset of 33 images. The labels were the same for each picture, using the values in equation 4.1, which are the known intrinsic parameters of our camera. Extrinsic parameters were ignored. The brightness of the pictures are collectively modified to counteract darkness.

The results are displayed in table 5.3.

Name	**Principal**	**Combined**
SD on α	145.9	527.2
SD on β	222.9	608.1
SD on u_0	173.0	76.21
SD on v_0	37.68	173.9
MAPE on α	6.687 %	29.10 %
MAPE on β	10.53 %	33.59 %
MAPE on u_0	16.23 %	6.968 %
MAPE on v_0	7.192 %	37.15 %

Table 5.3: Prediction results on a real dataset. We are using networks described in section 5.2.

Chapter 6

Discussion

6.1 Discussion

Concerning the result statistics in tables 5.1, 5.2, and 5.3, standard deviation gives an idea of the scale of the error, while MAPE informs us on the magnitude of the error relative to its size. When comparing how good a network is at predicting different elements, it is only useful to look at MAPE. When comparing the same predicted variables in different networks either statistic is relevant.

According to table 5.1, the VGG19 implementation performs much better than the others, in both time required to train and accuracy of results. MAPE values of the rotation and translation are poor across the board, which is a reflection of how an MSE loss function gives much more weight to the intrinsic matrix parameters, which are larger. Overall, VGG19 is much better suited to the task, at least with its pre-trained weights. Setting some of its layers as trainable could make a difference for the other base networks.

We notice as well that the deeper the base networks are, the worse their implementations perform. This is counter-intuitive as deeper networks are generally able to identify more complex patterns. Our analysis is that it is not the depth of the base model that is counting against them, but rather their depth relative to the implementation they are a part of. Since the more parameters are in the base layer (which is the case with a deeper base network), the smaller the the proportion of trained parameters in the system. A solution to this would be to set the base networks as

trainable, or at least some its layers. Even disregarding poor accuracies, the training time required by the other two networks make further investigation fairly unattractive.

Results from table 5.2 tell us that our earlier concerns, over a neural network's ability to find the principal point on a generated image, were unfounded. At least for VGG19, training more layers improves accuracy. In fact, both the 'trainable' and'principal' versions performed above baseline for prediction of focal parameters. So, not only is it possible to get good predictions for a generated image's principal point, but also by learning this prediction the network becomes better at predicting the two focal parameters as well. Unfortunately, the 'combined' version of VGG19 could not be trained for any longer due to time constraints. It performs only slightly worse than 'trainable' on focal parameters, but has better accuracy than 'principal' on principal points.

On real data, see table 5.3, 'combined' has worse accuracies than'principal' for all but one variable. However, between them, each intrinsic parameter can be predicted with mean accuracies of 89.47 % or better. This is very promising as it shows that real-life applications for these models is viable. A caveat is that these numbers are calculated on a sample of 33 images, from a single laparoscope, with the same intrinsic parameters for each image. Also, the brightness of the images has to be modified, although perhaps this would not have been necessary had there been better lighting.

6.2 Further Work

From our results, there are several areas that are worth looking into:

- Training the 'combined' version of VGG19 for longer.
- The focal variables, which have a larger value than the principal point coordinates, tend to be predicted better by our models. Giving more weight to the principal point coordinates loss could change that.
- Seeing as predicting principal points improves the prediction of other intrinsic parameters, it could be worth giving greater weight to extrinsic parameter losses, hoping that this would also improve intrinsic parameter prediction accuracy.
- Setting the whole VGG19 base layer as trainable in our implementations.
- Having more 'heads' in our network, one for every intrinsic parameter instead of one for the two focal parameters and another for the principal point coordinates.
- Trying other base networks (and setting their layers as trainable).
- Constructing a real dataset for training. However, given the cost and the surprisingly good results we had testing our network on real images, it might be worth investigating the other points first.
- Using a larger dataset with more varied background images for training.
- Further investigation into different network settings: loss function, learning rate, activations, etc.
- More rigorous testing in lab and operating theatre environments would give us a better idea of how much progress is to be made before this calibration technique becomes usable.

Chapter 7

Conclusion

The work conducted in this thesis was done under time constraints, it is however a promising first step towards using deep learning techniques for laparoscope calibration.

Our aim is to provide a proof-of-concept for single-image calibration using Deep Learning models. This is an improvement on state-of-the-art methods which require multiple calibration images, adding to the time and complexity of the pre-operation setup for augmented laparoscopy. Our work, by reducing difficulty of such a setup and as part of the SmartLiver project, goes towards making laparoscopy more widely available for liver cancer patients.

We have completed the stated objectives for this thesis:

- generate a large number of training data,
- build a neural network on top of popular ImageNet models,
- train these networks and compare their performances,
- evaluate one of our neural networks on real data.

In chapter 2, we described current calibration theory and procedures, documenting the changes in the state-of-the-art in the past 20 years. Chapter 3 covers the basics of artificial neural networks and convolution, and introduces the convolutional neural network architectures that we used as a base to build our own system.

Our most successful system is a neural network which uses VGG19 as a base, and has 4 separate outputs for focal parameters, principal point coordinates, the calibration board's rotation matrix, and the calibration board's translation vector.

Our VGG19 ‘base’ is untrainable except for its last two blocks. It obtained MAPEs of 2.375 % and 2.776 % for the focal parameters, and 9.751 % and 8.517 % for the principal point coordinates. In terms of standard deviation, the results were 59.44, 64.31, 109.1, and 52.91, respectively. On real images, the best MAPE for each variable is below 10.53 %.

We suggested a few ideas for further work in section 6.2.

Bibliography

[1] S. K. Reddy, A. Tsung, and D. A. Geller. Laparoscopic liver resection. *World Journal of Surgery*, 35(7):1478–1486, Jul 2011.

[2] UK National Health Service (NHS). Overview: Laparoscopy (keyhole surgery). `https://www.nhs.uk/conditions/Laparoscopy/`, Aug 2018. Accessed: 2019-07-17.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, Sep 2014.

[4] M. Hatzinger, S.T. Kwon, S. Langbein, S. Kamp, A. Häcker, and P. Alken. Hans christian jacobaeus: Inventor of human laparoscopy and thoracoscopy. *Journal of Endourology*, 20(11):848–850, 2006. PMID: 17144849.

[5] R. Vecchio, B. V. MacFayden, and F. Palazzo. History of laparoscopic surgery. *Panminerva medica*, 42(1):87–90, Mar 2000.

[6] S. Bernhardt, S. A. Nicolau, L. Soler, and C. Doignon. The status of augmented reality in laparoscopic surgery as of 2016. *Medical Image Analysis*, 37:66 – 90, 2017.

[7] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[8] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[10] A. Krizhevsky, I. Sutskever, and G. E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[11] WEISS. New nihr award to improve outcomes for liver cancer patients through minimally-invasive surgery. `https://www.ucl.ac.uk/interventional-surgical-sciences/news/2018/feb/new-nihr-award-improve-outcomes-liver-cancer-patients-through-m` Feb 2018. Accessed: 2019-07-01.

[12] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.

[13] S. Lee, H. Lee, H. Choi, S. Jeon, H. Ha, and J. Hong. Comparative study of hand–eye calibration methods for augmented reality using an endoscope. *Journal of Electronic Imaging*, 27(4):1, Jul 2018.

[14] L. Ou and X. Gu. Application research of Zhengyou Zhang calibration method in visual recognition of hull welds. *International Journal of Engineering and Applied Sciences (IJEAS)*, 6, Mar 2019.

[15] Mathworks. What is camera calibration? `https://uk.mathworks.com/help/vision/ug/camera-calibration.html`. Accessed: 2019-07-25.

[16] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, Aug 1987.

[17] Wikimedia Commons. The geometry of a pinhole camera. `https://upload.wikimedia.org/wikipedia/en/thumb/7/7c/`

Pinhole.svg/1280px-Pinhole.svg.png, Jan 2019. Accessed: 2019-08-16.

[18] W. Burger. Zhang's camera calibration algorithm: In-depth tutorial and implementation. Technical report HGB16-05, University of Applied Sciences Upper Austria, School of Informatics, Communications and Media, Hagenberg, May 2016.

[19] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, Dec 2000. MSR-TR-98-71, Updated March 25, 1999.

[20] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, Jun 1989.

[21] S. Thompson, D. Stoyanov, C. Schneider, K. Gurusamy, S. Ourselin, B. Davidson, D. Hawkes, and M. J. Clarkson. Hand–eye calibration for rigid laparoscopes using an invariant point. *International Journal of Computer Assisted Radiology and Surgery*, 11(6):1071–1080, Jun 2016.

[22] S. De Buck, J. Van Cleynenbreugel, I. Geys, T. Koninckx, P. R. Koninck, and P. Suetens. A system to support laparoscopic surgery by augmented reality visualization. In W. J. Niessen and M. A. Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001*, pages 691–698, Berlin, Heidelberg, Oct 2001. Springer Berlin Heidelberg.

[23] R. Shahidi, M. R. Bax, C. R. Maurer, J. A. Johnson, E. P. Wilkinson, Bai Wang, J. B. West, M. J. Citardi, K. H. Manwaring, and R. Khadem. Implementation, calibration and accuracy testing of an image-enhanced endoscopy system. *IEEE Transactions on Medical Imaging*, 21(12):1524–1535, Dec 2002.

[24] C. Wengert, M. Reeff, P. C. Cattin, and G. Székely. Fully automatic endoscope calibration for intraoperative use. In H. Handels, J. Ehrhardt, A. Horsch, H.-

P. Meinzer, and T. Tolxdorff, editors, *Bildverarbeitung für die Medizin 2006*, pages 419–423, Berlin, Heidelberg, Jan 2006. Springer Berlin Heidelberg.

[25] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, Jun 1997.

[26] R. Melo, J. P. Barreto, and G. Falcao. A new solution for camera calibration and real-time image distortion correction in medical endoscopy-initial technical evaluation. *IEEE transactions on bio-medical engineering*, 59:634–44, Nov 2011.

[27] J. P. Barreto, J. Roquette, P. Sturm, and F. Fonseca. Automatic camera calibration applied to medical endoscopy. In *Proceedings of the British Machine Vision Conference*, pages 52.1–52.10. BMVA Press, 2009. doi:10.5244/C.23.52.

[28] J.-Y. Bouguet. Camera calibration toolbox for matlab. Available: `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`, Oct 2015. Accessed: 2019-08-13.

[29] M. Schwalbe, M. Fusaglia, P. Tinguely, H. Lu, and S. Weber. Design and implementation of a laparoscope calibration method for augmented reality navigation. In *CURAC 2015 Tagungsband der 14. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V.*, pages 71–76, Sep 2015.

[30] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[31] D. Barber. From lecture notes in COMP0090 Introduction to Deep Learning: Introduction to FeedForward Neural Nets. University College London, Department of Computer Science, 2018.

[32] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, Jan 2010.

[33] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[34] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[36] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[37] W. Schroeder, K. M. Martin, and W. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*. Kitware, 4th edition, Dec 2006.

[38] T. Dowrick et al. scikit-surgeryvtk: VTK for image guided surgery applications. Available: `https://weisslab.cs.ucl.ac.uk/WEISS/SoftwareRepositories/SNAPPY/scikit-surgeryvtk`, 2019–. Accessed: 2019-08-11.

[39] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, Dec 2014.