Designing Convolutional Neural Networks for Scintillation Photography and General Applications

Jeremy Ocampo

University College London
Physics & Astronomy
Centre for Doctoral Training in Data Intensive Science

A dissertation submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

Supervisors: Prof. Adam Gibson and Dr. Simon Jolly
Examiners: Dr. Catarina Veiga and Prof. David Colling

1st February 2023
Declaration

I, Jeremy Ocampo, 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 thesis.
Abstract

Modern radiotherapy treatments provide complex dose distributions which are difficult to measure and verify. This is due to beams having high-dose gradients, time-varying intensity and sizes reducing to millimetre scales. A dosimeter was designed to provide improvements over standard detectors that are unsuitable for measuring complex small fields. The proposed detector system consists of an irradiated scintillator sheet that is photographed, from which the dose is reconstructed. This provides a cheap, fast, and high-resolution solution. But scintillation images come with a variety of visual artefacts that need correcting.

Convolutional neural networks (CNN) have been shown to have excellent accuracy in extracting useful information from noisy images. This requires thousands to millions of training images. In scintillation photography, there is not enough data to achieve an acceptable performance for CNNs. A novel method using domain randomisation was used to solve this issue, where thousands of images were simulated with varying physical parameters. This data can be used to train the CNN to be robust to visual artefacts. These CNNs are designed to assist in our image processing by predicting relative dose distributions and (un)known physical parameters, which gives confidence that the measured images are correct. Results showed that CNNs performed better than classical methods and could provide dose distributions that are suitable for routine QA.

This work was extended by designing a novel CNN layer which can be generalised to non-Euclidean domains while maintaining scalability, e.g., the sphere which has many applications. This is done by developing modern methods in group convolutions and helping them scale to high resolution. This method leverages symmetries in the data, which improves the CNNs ability to generalize to “unseen” data with only a few thousand training examples. The models were tested on spherical data benchmarks for which state-of-the-art performance was achieved.
Impact statement

Quality assurance (QA) of radiotherapy beams is necessary for making sure the correct dose is given. However, the process currently takes up a lot of time which otherwise can be spent treating patients. In this thesis, we developed a faster and cheaper detector system using scintillators. This contributes to the medical physics community by characterizing and simulating the physics of the system, and a contribution as a medical application as this system may be developed for faster QA in hospitals, in particular our experiments were developed for the CyberKnife in University Hospitals Birmingham.

We have also developed a novel method using convolutional neural networks (CNNs) for extracting dose from scintillation images, further improving the accuracy of the dose verification. We have also shown that CNNs can predict known parameters as well as unknown ones. This can offer some confidence in the outputs, potentially helping to overcome barriers to acceptance in risk-critical applications such as radiotherapy. This work is a contribution towards the medical physics community as a starting point on using machine learning methods in scintillation dosimetry.

Our final contribution is for using CNNs for general applications. Most CNNs are designed for planar images, but we have developed a method for using CNNs for a general geometry while maintaining scalability. This is a contribution for the machine learning community for providing a theoretical and empirical framework, and to industry as our new CNNs may be used for a wide variety of applications, in particular for spherical data such as climate data and panoramic images.
Acknowledgements

I would like to thank my supervisors Adam and Simon for their support and wisdom throughout my PhD, they’ve offered excellent advice for work, life and mental health. I would also like to thank a few people for helping me complete the work for my PhD. Tim, for supervising me on the machine learning side in the scintillation dosimetry project. Jason and Matt, for supervising me on the scalable spherical CNNs project. Mayeul, who reviewed a lot of my code. Last but not least, Geoff who ran the CyberKnife beam for our experiments in University Hospital Birmingham.

I thank all my friends and flatmates who helped me get through the lockdown and persevere through my PhD. The filipino community and family from home, and my partner Jasmine for their endless support. A massive thanks to Niel, Arvin, Peter and Paul for their excessive comedy and a reminder to not take life too seriously.
Contents

1 Thesis Outline 11

2 Introduction to neural networks 13
  2.1 Feedforward neural networks 14
    2.1.1 Foundation for feedforward neural nets 14
    2.1.2 Understanding the activation function 16
    2.1.3 Loss functions 17
  2.2 Training feedforward neural networks 18
    2.2.1 Gradient based optimization 18
    2.2.2 Weight initialization 20
    2.2.3 Combating overfitting 20
  2.3 Convolutional neural networks 22
    2.3.1 Convolution operation and motivation 22
    2.3.2 Convolutional layers 24
    2.3.3 Downsampling 26
    2.3.4 CNN architectures 27
  2.4 Geometric deep Learning 30
    2.4.1 Group equivariance 31
    2.4.2 Spherical convolutions 34
  2.5 Summary 36

3 Introduction to radiotherapy dosimetry 37
  3.1 Overview of radiotherapy 37
    3.1.1 Physics 38
    3.1.2 Dose distributions 41
    3.1.3 Quality assurance 43
    3.1.4 Dosimeters 44
  3.2 Scintillation dosimetry 45
    3.2.1 Physics of scintillation 45
    3.2.2 Physics of Cherenkov light 48
    3.2.3 Scintillation detectors 49
  3.3 Monte Carlo simulations 49
    3.3.1 Overview 49
    3.3.2 Geant4 51

4 Scintillation dosimetry for CyberKnife 53
  4.1 Introduction 53
  4.2 Materials and Methods 57
5 CNN assisted scintillation dosimetry

5.1 Introduction

5.2 Method

5.2.1 Measured data

5.2.2 Simulated data

5.2.3 Machine learning pipeline

5.2.4 Predicting PDDs

5.2.5 Predicting Output factors

5.2.6 Predicting Variables

5.3 Results and discussion

5.3.1 Predicting PDD

5.3.2 Predicting Output Factors

5.3.3 Predicting Variables

5.3.4 Robustness

5.3.5 Explainability

5.4 Conclusions

6 CNNs for general applications

6.1 Introduction

6.2 Background

6.2.1 Continuous group convolution

6.2.2 Continuous spherical convolutions

6.2.3 Discrete spherical convolutions

6.3 Discrete-Continuous (DISCO) convolutions

6.3.1 DISCO group convolution

6.3.2 DISCO spherical convolution
6.3.3 Rotational equivariance ........................................ 117
6.3.4 Filter parameterization ....................................... 119
6.3.5 DISCO transposed spherical convolution .................... 119

6.4 Efficient implementation ........................................ 120
6.4.1 Sparse tensor representation ................................. 120
6.4.2 Memory optimizations .......................................... 121
6.4.3 Sparse gradients ................................................ 123

6.5 Experiments ...................................................... 124
6.5.1 Equivariance tests ............................................ 124
6.5.2 Rotated MNIST on the sphere ............................... 125
6.5.3 Semantic segmentation ....................................... 126
6.5.4 Depth estimation ............................................... 129

6.6 Conclusions ........................................................ 130

7 Summary .................................................................. 131

A Discolouration .......................................................... 132

B Simulation details .................................................... 132
B.1 Dose validation ..................................................... 132
B.2 Depth of field ....................................................... 133

C Simulation parameter distributions ............................... 134
C.1 PDD ................................................................. 134
C.2 Output factor ........................................................ 135
C.3 Variables ............................................................ 136

D CNN models for scintillation dosimetry ......................... 137
D.1 PDD models ........................................................ 138
D.2 Output factor models ............................................. 140
D.3 Scintillation ratio and beam width model .................... 143
D.4 Beam energy model .............................................. 143

E Computational Cost and Memory ................................ 144

F Custom sparse gradients ............................................. 144

G Experiments ............................................................ 146
G.1 Rotated MNIST on the Sphere ................................ 146
G.2 Semantic segmentation: 2D3Ds ............................... 147
G.3 Semantic Segmentation: Omni-SYNTHIA ................. 148
G.4 Depth Estimation: Pano3D on Matterport3D .............. 149
References 150

List of figures 164
List of acronyms

ANN Artificial neural network
AbsRel Absolute relative error
CNN Convolutional neural networks
DISCO Discrete-Continuous
DOF Depth of field
DSLR Digital single-lens reflex
DTA Distance to agreement
FLOP Floating point operation
FWHM Full width half maximum
GPU Graphical processing unit
ICRU International commission on radiation units and measurement
MAD Mean absolute difference
MLE Maximum likelihood estimation
MU Monitor unit
OF Output factor
PCA Principal component analysis
PDD Percentage depth dose
QA Quality assurance
RIBE Radiation induced bystander effect
RMSE Root mean square error
RMSLE Root mean square logarithmic error
ReLU Rectified linear unit
SSD Source to surface distance
SVM Support vector machine
SqRel Square relative error
TCP Tumour control probability
1. Thesis Outline

In the past couple of decades, machine learning has caused a paradigm shift in the physical sciences. Scientific research traditionally focuses on making theories to explain natural phenomena and to design experiments for discovering or verifying those phenomena. Nowadays, many research projects have a focus on machine learning and data-intensive methods (Ourmazd, 2020) where models of the phenomena are learnt from the data. The rise in machine learning is in part due to the exponential increase in available data, the increase in computational power and memory of hardware, and the amount of accessible and open-source machine learning libraries. Classical machine learning methods work well for smaller datasets, but as the amount of data increases, neural networks tend to outperform them. In particular convolutional neural networks (CNN) have been shown to perform significantly better than classical methods at making predictions on images (Mahony et al., 2019). In this thesis we will be designing CNNs for a specific scientific problem in radiotherapy, and we will also design a novel CNN for spherical data which may be used for a broad range of applications. We therefore give an introduction to neural networks in Chapter 2 which is written at a level accessible to medical physicists.

Deep neural networks have a common “black box” nature in that it is difficult to interpret what has been learnt. Therefore, it is advantageous to study the classical methods which have simpler interpretations and also gives us a better understanding of what is physically happening. In this thesis we will be comparing a classical and CNN approach to a particular field of radiotherapy: the dose verification of small fields using scintillators. Due to the rise of complex radiotherapy beams, there has been an increase in more sophisticated treatment planning and dose verification methods, in particular for small field radiotherapy (Das et al., 2016). The dosimetry of small field radiotherapy beams are commonly based on point detectors such as ionisation chambers placed in water which depend on the measurements of ionised charge in the chamber. This method is slow, expensive and has a few problems in the small field regime, namely volume averaging and loss of charged particle equilibrium. In this thesis, we will be focusing on dose measurements via photography of large volume scintillators (Hamel, 2021) which is a solution to these problems.

In Chapter 3 we have written an introduction to radiotherapy and scintillators for a broad audience, namely with a (but not limited to) physics background. Scintillators are a luminescent material which emits light when irradiated with X-rays. By capturing the light, we can reconstruct the dose deposited in the scintillator. These are advantageous as we can quickly capture a 2D dose in one image, and the
resolution of the image is high enough that we don’t have to worry about volume averaging.

As we outlined above, this thesis is primarily in the fields of neural networks and dosimetry in radiotherapy for which we give introductions to in Chapters 2-3. The rest of this thesis is applying our research in those fields. We outline our objectives for each chapter here:

**Chapter 4: Scintillation dosimetry for CyberKnife.** We develop a rapid and cheap detector system for the CyberKnife, which is a radiotherapy treatment delivery system for small fields. Ultimately, faster dose verification means there is less workload for clinical scientists, more availability for treatment and more regular quality assurance of the beams. Our detector system is composed of a Camera and a scintillator sheet and we also develop an image processing pipeline to extract the dose from our images.

**Chapter 5: CNN assisted scintillation dosimetry.** In our scintillator detector system of Chapter 4, our images have many optical artefacts and low light output which means our dose extraction is highly obscured. In this chapter, we aim to improve the accuracy of the dose extraction from our scintillation images using CNNs. We also provide additional prediction of (un)known physical variables like beam width which gives us confidence that our measured images are correct.

**Chapter 6: CNNs for general applications.** This work is based on the paper Ocampo et al. (2023). Images from cameras often lie natively on the plane. However, image data also comes in all sorts of different geometries such as the sphere and cylinder, hence we need more effective ML algorithms to extract information from different data geometries. In this chapter we aim to do this using a novel method which we call DISCO convolutions. It uses DIScrete-COrtinuous convolutions on an arbitrary manifold while maintaining scalability.

There are a few possible paths to take when reading this thesis but is entirely up to the reader. As some chapters depend on other preliminary chapters, we recommend reading chapter 3 before 4, chapter (2, 3, 4) before 5 and chapter 2 before 6.
2. Introduction to neural networks

In the past decade, machine learning models have become increasingly popular as they have outperformed humans at certain tasks such as 3D protein structure prediction (Jumper et al., 2021), discovering new physics phenomena (Karagiorgi et al., 2022) and medical diagnosis (Richens et al., 2020). These advances have been primarily driven by the exponential increase in data that humans have produced, allowing machine learning models to increase in performance. The second reason is the increase in the computational power and memory of hardware such as graphical processing units (GPUs) allowing more data to be processed and trained on, in a feasible amount of time. Lastly, the increase in accessible and open-source machine learning libraries such as TensorFlow (Abadi et al., 2015) and Scikit-learn (Pedregosa et al., 2011) has allowed practitioners to easily implement machine learning in their respective fields.

**Machine learning.** While machine learning is a broad field with many applications, it can be simplified to fitting a curve or estimating the probability distribution of the data. This method allows predictions on new data without any manual or explicit programming. This is also why a common problem in machine learning is the model’s inability to perform well on unseen data, as the unseen data may have a different distribution. Most machine learning methods used today are *supervised*, meaning that we train our machine learning models on labeled data and use them to predict the labels of new data (e.g., predicting if an MRI image contains a tumour). *Unsupervised* machine learning is when we don’t have the labels and the machine learning model is used to learn from the structure of the data, for example if we have unlabeled images and we want to group them according to the objects that they contain. There are many classical machine learning algorithms (Murphy, 2013) such as support vector machines (SVM), principal component analysis (PCA) and decision trees which work well for smaller datasets with less features, are computationally cheap and easy to interpret. However, for large datasets with more complex features, neural networks generally work better.

**Deep learning.** Deep neural networks generally outperform classical machine learning methods as the amount of data increases (Sarker, 2021). One reason for this is that in classical machine learning, features from the data are manually engineered and extracted while neural networks are able to learn these features and often features that humans cannot find intuitively (Mahony et al., 2019). With more data, neural networks can learn more features and their combinations. Due to their ability to extract features from images (Zeiler and Fergus, 2013), CNNs have exceptional performance for image datasets (see for example, the image classification
benchmark, ImageNet (Deng et al., 2009)). For this reason we have chosen to focus on CNN architectures as we use them on our scintillation images and for general applications.

We begin the journey by giving the basic formulation of feedforward neural networks followed by explaining why and how they work in the following sections. We also provide some standard tricks and the procedure on how they are trained. Finally, we give an overview of one of the strongest neural network priors to use for image datasets, the planar CNN and the group CNN.

2.1. Feedforward neural networks

2.1.1. Foundation for feedforward neural nets

**Linear regression.** Feedforward neural networks can be built up from the basic concept of linear regression. Given \(n\) data points with \(m\) features each, \(\{x_{1j}, \ldots, x_{mj}\}_{j=1}^{n}\) we can fit their corresponding labels or values \(\{y_{j}\}_{j=1}^{n}\) with a linear map, here \(j\) represents the \(j^{th}\) data point. A simple example of this would be where we try to predict the cost of fuel \(y_{j}\), given the distance travelled and average speed of a car \(x_{1j}, x_{2j}\). This linear map is given by:

\[
\hat{y}_j = \sum_i w_i x_{ij} + b
\]

where the \(\hat{\cdot}\) denotes the predicted values of the map. The parameters \(w_i\) are called the weights and \(b\) the bias*, for which we want to find the optimum values that fit the data. These optimum values are found by minimizing a cost or loss function such as the mean squared error \(E[(y_j - \hat{y}_j)^2]\) which effectively penalizes larger differences in the predicted and known values. If we want to predict multiple values for each data point (e.g., cost of fuel and car mass), then we simply add another dimension to the outputs \(y_j \rightarrow y_{kj}\) and the weights \(w_i \rightarrow w_{ki}\), giving

\[
\hat{y}_{kj} = \sum_i w_{ki} x_{ij} + b_k
\]

This is essentially a simple one-layer, linear, feed forward neural network. But it only makes sense to use linear regression if the data has a linear form. In polynomial regression we can replace the data \(x_{ij}\) in equation 1 by polynomial combinations of the data \(\sum_n x_{ij}^n\). But this only makes sense if the data has a polynomial form. To fit data with an arbitrary form we can use feedforward neural networks which are

*Since we are only predicting one variable (e.g. the cost of fuel), we only need one bias value. Note that there are no \(j\) indices on the weights as that represents the data instance or measurement. This is equivalent to saying that our model is independent of the data.
2.1 Feedforward neural networks

universal approximators (Hornik et al., 1989) meaning they can approximate any continuous function.

Neural network. Due to their large number of parameters, deep feedforward neural networks\(^*\) are known for their “black box” nature as it is difficult to interpret what each weight is doing and why it has taken a particular value. Here we simply state their mathematical form and provide explanations in the next sections. For a deep feedforward neural network, the inputs \(x_{ij}\), are mapped to the predicted outputs \(\hat{y}_{kj}\), by applying many linear maps but with some non-linear function \(h\) after each map. These are called the activation functions and introduce non-linearity for the fitted map (discussed further in section 2.1.2), for example \(h(x) = \tanh(x)\). This can be written in a recursive form, for an \(N\)-layer neural network (see Figure 2.1)

\[
\begin{align*}
\text{first layer:} & \quad \ a_{k}^{[0]} = h^{[0]} \left( \sum_{i} w_{ki}^{[0]} x_{ij} + b_{k}^{[0]} \right) \\
\text{\(l\)th layer:} & \quad \ a_{k}^{[l]} = h^{[l]} \left( \sum_{i} w_{ki}^{[l]} a_{ij}^{[l-1]} + b_{k}^{[l]} \right) \\
\text{final layer:} & \quad \ \hat{y}_{kj} = h^{[N]} \left( \sum_{i} w_{ki}^{[N]} a_{ij}^{[N-1]} + b_{k}^{[N]} \right)
\end{align*}
\]

where the \([l]\) denotes which layer it belongs to\(^1\), i.e. \(w_{ij}^{[l]}\) are the \(l\)th layer weights, \(h^{[l]}\) is the \(l\)th layer activation function, and \(a_{i}^{[l]}\) is called the activation of the \(l\)th layer. The number of features for the activation at layer \(l\) also varies and is often called the number of neurons. Deep learning is often used with a large number of data

---

\(\ast\)Sometimes referred to as dense neural networks

\(\dagger\)It does not denote a power
points and so optimization algorithms often cannot store the whole dataset. Thus, mini batch gradient descent (Goodfellow et al., 2016) is almost always used to find the optimal parameters, we discuss this further in section 2.2.1.

2.1.2. Understanding the activation function

Here we will explain the utility of the activation function through an example. Consider the problem of fitting a quadratic dataset, $y = x^2$. If this is fitted with no activation functions, then no matter how many layers are used, the neural network will be entirely linear, and the data will be fitted with a line as shown in Figure 2.2a. By adding the non-linear activation functions, we’re allowing the neural network to have a non-linear form. But this isn’t enough to justify that we can fit a quadratic or any arbitrary dataset.

The most commonly used activation function is the rectified linear unit (ReLU), which is just $h(x) = \max(0, x)$. Before the data is passed to $h$, it is scaled and translated by the parameters, i.e. $h(x) = \max(0, wx + b)$, by adding many of these activations together we are essentially adding lines with different gradients and offsets except the 0 tail allows for a change in gradient*. By adding the right combinations of $\max(0, wx + b)$ with different $w$ and $b$, you can essentially fit any continuous curve if you add enough combinations. This is what the feedforward neural network does once the optimal parameters are found as shown in Figure 2.2b which has fitted the function $\hat{y} = \max(0, wx) + \max(0, -wx)$, a 1 layer neural network with 2 neurons. By creating more combinations of $\max(0, wx + b)$ the neural network creates a more refined curve as shown 2.2c, fitted with 2 layers and 10 neurons each.

![Figure 2.2: Fitting a quadratic with a neural network.](image)

*For example there is no $a, b$ such that $ax + bx$ will have a change in gradient, but $\max(0, ax + c) + \max(0, -bx + d)$ does, and the change in gradient is placed along $x$ such that it fits the data
It is easy to see that this method can fit a general curve by placing the scaled activation functions in the right position between the datapoints. We can also note that neural networks work by interpolating between datapoints and the more datapoints the better the interpolation will be. If we want to make a prediction for some datapoint which is far from our dataset then it is unlikely that our neural network will make an accurate prediction, for example in our quadratic example the data increases with $x^2$ but our neural network increases with $x$. In machine learning, our assumptions for our model are called an inductive bias and will often determine if our model will generalize to unseen datapoints.

2.1.3. Loss functions

In machine learning, loss functions are used as a measure of the difference between the predicted and true output for a given input. They are used to provide guidance for the optimization process to minimize the difference and improve the model’s predictions. The chosen loss function depends on the problem at hand and what we want to optimize. Here we list some common problems and loss functions in deep learning.

**Binary classification.** In classification tasks, the output $\hat{y}$ of a neural network is generally mapped to a probability between 0 and 1, and the true outputs $y$ is 0 or 1. An example could be where the input to the neural network is an image and the predicted output is the probability that the image contains a cat or not. The loss function that is commonly used for this is problem is the binary cross-entropy (Goodfellow et al., 2016), which has the form

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

and is minimized when $y = \hat{y} = 1$ (cat), or $y = \hat{y} = 0$ (no cat). From a probabilistic perspective, this can be interpreted as a maximum-likelihood-estimation (MLE) problem (Murphy, 2013). If $\hat{y}$ is the probability that there is a cat, and $1 - \hat{y}$ is the probability that there is not cat, then the probability distribution of the output data is given by the Bernoulli distribution, $\hat{y}^y(1 - \hat{y})^{1-y}$. In MLE, to get the most likely probability $\hat{y}$, we need to maximize our probability distribution given the data. This is equivalent to minimizing the negative log-likelihood of our distribution, $-\log(\hat{y}^y(1 - \hat{y})^{1-y})$, which is equivalent to minimizing the loss in equation 4. To get an output $\hat{y}$ between 0 and 1, a common final activation function that is used in the final layer is $\hat{y} = h(x) = (1 + e^{-x})^{-1}$ called the sigmoid.

**Multiclass classification.** For classification with multiple classes, such as predicting the probability that an image contains a dog, cat or duck, the output is a vector
2.2 Training feedforward neural networks

\(\hat{y}_i\), which gives the probability for each class. The loss function commonly used here is the categorical cross-entropy

\[
L(\hat{y}, y) = - \sum_i y_i \log(\hat{y}_i)
\]  

(5)

which is the generalization of binary cross-entropy. The final activation commonly used is \(\hat{y}_i = h(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}}\), which allows the probabilities to add to 1.

**Regression.** In regression tasks, the output data can be any real number and so the loss functions used must be chosen to fit the problem. For example, for predicting the temperature tomorrow, you could use mean squared error, mean absolute error etc. depending on how much you want to penalize larger errors. In practice, the best loss function is chosen through trial and error.

### 2.2. Training feedforward neural networks

In order to predict on unseen data, we need to find the optimal parameters for the neural network. It is worth noting that there are two types of parameters we can optimize, the parameters which are the weights of the neural network, and the hyper-parameters which are used to configure the model and learning process e.g. number of neurons and learning rate (see next section 2.2.1). Due to the rise of open-source python libraries such as TensorFlow and PyTorch (Abadi et al., 2015; Paszke et al., 2019) and the rise of powerful GPUs, many scientists have been able to quickly implement and train their own neural networks. Here we provide a basic understanding of neural network training that is needed for this thesis.

#### 2.2.1. Gradient based optimization

Gradient descent is almost always used to minimize the loss function \(L(y, \hat{y}(x, w))\) of a neural network and the aforementioned python libraries have been developed and optimized for this purpose. Gradient descent updates the parameters \(w\), iteratively based on the data \(x\). Each parameter is updated by moving it in the direction towards a minimum of the loss function, i.e. the opposite direction of the gradient. The iterative step is then,

\[
w \rightarrow w - \alpha \nabla_w(L)
\]  

(6)

where \(\alpha\) is called the **learning rate.** In **batch** gradient descent, we use the whole dataset to compute the loss, and if the loss is a convex function this would eventually lead to the global minimum of \(L(y, \hat{y}(x, w))\).

**Mini-batch gradient descent.** Batch gradient descent is not practical for large datasets as you would have to store the whole dataset in the RAM to compute
2.2 Training feedforward neural networks

the gradient. It is common to use mini-batch gradient descent, where equation 6 is iteratively computed using mini-batches* of the whole dataset. There is debate among which mini-batch size is the best but a typical choice is between 8 and 256 data points (e.g. see Kandel and Castelli (2020)). Nevertheless, the optimum mini-batch size can be found for example via grid search and depends on the dataset. This does not guarantee reaching the global minimum loss of the whole dataset as the gradient steps now correspond to minimizing the loss for each mini-batch.

Gradient descent methods. There are many methods which has improved the convergence speed of mini-batch gradient descent (Ruder, 2016). Mini-batch gradient descent has random oscillations due to the randomness of the mini-batches but the average directions generally point towards the minimum, see Figure 2.3b. The momentum method (Qian, 1999) leverages this fact by updating the weights using the moving average of the gradients, see Figure 2.3c. In equation 6, the learning rate $\alpha$ is constant which means that it is not robust to a lot of gradient changes of the loss curve, for example the loss function can have very steep gradients which could cause it to get stuck in a well. Adagrad (Duchi et al., 2011) is a method that solves this issue by scaling the learning rate by the inverse of the size of the previous gradients, meaning big gradients will get scaled down and small gradients scaled up. This is called an adaptive learning rate method. RMSProp (Hinton, 2018) is like Adagrad but uses the moving average of the gradients to scale the learning rates. Adam (Kingma and Ba, 2015) is the most used gradient descent method for deep learning, it combines the momentum method and RMSProp. We will be using the Adam method in many of the models in this thesis.

Backpropagation. The gradient term in equation 6 can be computed using backpropagation (Goodfellow et al., 2016; Rumelhart et al., 1986), which is essentially using the chain rule. Using the recursive form in the neural net in equation 3, the

Figure 2.3: Depiction of different types of gradient descent in deep learning. The contours show the loss curve as a function of the neural network weights $w$ for the whole dataset. The arrows show the gradient steps taken.

*This is often just called the batch size.
2.2 Training feedforward neural networks

The gradient with respect to the \( l \)th layer weights is

\[
\frac{\partial L(\hat{y}, y)}{\partial w_{ij}^l} = \sum_{k,l,...,y,z} \frac{\partial L(\hat{y}(a^{[N]}), y)}{\partial a_k^{[N]}} \frac{\partial h(a^{[N-1]})}{\partial a_i^{[N-1]}} \cdots \frac{\partial h(a^{[1]}), y}{\partial a_z^{[1]}} \frac{\partial a^{[l]}_z}{\partial a^{[l]}_i} \frac{\partial w_{ij}^l}{\partial w_{ij}^l},
\]

where the sum comes from the chain rule for partial differentiation*. For computational convenience, this is usually written using matrix multiplications and dot products (e.g. see Goodfellow et al. (2016)). Each differentiation term is usually computed using an automatic differentiator (Neidinger, 2010) which is usually built in to the deep learning library you are using, e.g. TensorFlow and PyTorch (Abadi et al., 2015; Paszke et al., 2019).

2.2.2. Weight initialization

Before training a neural network, we must decide what initial weights they should start off with. If we know nothing about the data and their loss curve, then a first guess would be to randomize the weights initially. In supervised learning, we usually know the range of outputs that we want to learn, for example in classification the outputs should be between 0 and 1, the weights could then be randomly initialized so that the outputs are close to this range which means there are fewer iterations for convergence.

**Exploding and vanishing gradients.** A problem with initializing the weights randomly is the possibility that the activations will exponentially get larger or smaller throughout the neural network because we are multiplying many numbers together, this is called the exploding/vanishing gradient problem (Goodfellow et al., 2016). This problem is also true for the gradients in equation 7 as we are multiplying many gradients together. A common solution is to initialize the weights such that the gradients and activations don’t explode or vanish. Glorot initialization (Glorot and Bengio, 2010) initializes the weights such that the standard deviation and mean of the activations and gradients stay constant in every layer. He (He et al., 2015a) initialization does the same but considers the ReLU activations.

2.2.3. Combating overfitting

**Bias-Variance tradeoff.** By increasing the number of parameters in our model, we would be able to fit the complexities in the data but then we run the risk of overfitting to the training data. This problem is generally known as **Bias-Variance tradeoff** (Murphy, 2013). Say we have a randomly sampled dataset, and we train a model. With enough parameters, our model is will fit the noise of the training data, thereby

---

*For example the first sum over \( k \) is due to the dependance of \( \hat{y}(a^{[N]}) \) on \( a_k^{[N]} \) where \( k = 1, ..., n \) where \( n \) is the number of neurons at layer \( N \).
increasing the error on unseen data. If we repeat this process with another randomly sampled dataset, then our model is likely to fit the noise of the new dataset. Thus, our model varies depending on the dataset sampled. In general, a highly complex model has a high variance (see Figure 2.4). The bias of the model refers to how close the true model is from our hypothesized model. Obviously, for a complex dataset, we would need a more complex model, but if we keep increasing the complexity then at some point we will stray away from the true model. Regularization is a method which reduces the complexity of our model, usually by promoting sparsity of the weights.

Weights regularization. Common regularization techniques in machine learning are L1 and L2 regularization (Murphy, 2013) where an extra term \( \lambda |w| \) or \( \lambda |w|^2 \) is added in the loss function. This promotes sparsity in the weights which reduces the complexity of the model and therefore reduces the tendency to overfit. Dropout (Srivastava et al., 2014) is also a method which promotes learning of a sparser model but without promoting sparsity of weights. It does this by randomly choosing weights with probability \( p \) and letting them be zero during the gradient descent step. During inference the weights are set to their learned values.

Data augmentation. As the dataset gets larger, the data noise is “averaged” out during training, and we are less likely to overfit. For smaller datasets, we can create more variation by modifying the data without making it look too unrealistic. For example, in image datasets, we can flip, rotate and translate images. This is generally called data augmentation (Shorten and Khoshgoftaar, 2019).

Early stopping. During training, it is important to check if we are overfitting. This is done by having a separate “test” dataset which our model has not been trained on. If the model does well on the training set and bad on the test set, that is an indicator that it has overfitted to the training set. In many cases there is a point at which the model performance stops improving on the validation set, here the weights of the model should be the ones used for inference.
2.3 Convolutional neural networks

CNNs have been the main driving force in the past decade for automated image tasks such as object detection (Redmon et al., 2016) and semantic segmentation (Ronneberger et al., 2015). Here we cover how and why they work so well which is mainly due to equivariance and intuitively engineered CNN architectures.

2.3.1 Convolution operation and motivation

Image convolution. Images of the real world are often formed of combinations of edges, colours, and gradients. Convolutions are a way to emphasize particular features by mapping an image $x_{ij}$ to its feature map $f_{ij}$, where $i, j$ are the rows and columns of the image. The 2D discrete convolution is given by

$$f_{ij} = (\psi \ast x)_{ij} = \sum_{m,n} \psi_{mn} x_{i+m, j+n} \quad (8)$$

where $\psi$ is the convolution kernel. In words, the output at position $i, j$ is the dot product of the image and the kernel placed at position $i, j$, see Figure 2.5. For lower computational cost, the size of the kernel is often chosen to be a lot lower than the image (e.g. $3 \times 3$) and equation 8 can be visualized by a localized kernel that is a moving “window”. We note that this operation is in fact just a sparse matrix multiplication and therefore a sparse feedforward layer with shared weights*. In fact, this is a regularized feedforward neural network since the weights are sparse.

![Image of convolution with localized kernel](image.png)

Figure 2.5: Depiction of a convolution with a localized kernel.

Feature maps. Figure 2.6 shows examples of what happens when you convolve an

*We can rewrite equation 8 by a change of variables and then flattening $ij \to a, mn \to b$,

$$f_{ij} = \sum_{m,n} \psi_{(m-i),(n-j)} x_{mn} \quad (9)$$

$$f_a = \sum_b W_{ab} x_b \quad (10)$$

which is just a sparse-dense matrix multiplication since $w$ is localized. Note that many of the elements $W_{ab} = \psi_{(m-i),(n-j)}$ is repeated since $m - i$ is the same for some combinations of $m, i$. 

22
2.3 Convolutional neural networks

image with different 3×3 kernels. There are many kinds of kernels which emphasize different features. By combining these features, we can emphasize more complex features. But manually finding the correct kernels for a specific task is not always obvious. For example, if we were to make a dog classifier, we could take a linear combination of features maps that might emphasize dog ears, and then the sum of the output image should be high if it is a dog. We then must think about what combination of kernel weights to use which can be a tedious task. For CNNs, the kernel weights and combination of feature maps are automatically learned through gradient descent (see section 2.3.2).

**Equivariance** One important property of convolutions is that they have translational equivariance which means a translation of the input image gives the same feature map but translated the same amount, as illustrated in Figure 2.7. This is especially powerful for example in image classification, since we get similar features for each translated image variant, we should get a similar prediction. This provides an efficient way for neural networks to learn features since we learn all the feature maps of the translated variants of an image using only one image. This is in contrast to learning the same features for each translated image, e.g. from data augmentation, which you would have to do for non-equivariant neural networks. For some translation $T$, translational equivariance is written as (see equation 15 for a proof)

$$Tf = T(\psi * x) = (\psi * Tx)$$

(11)
2.3 Convolutional neural networks

2.3.2. Convolutional layers

There are many ways to combine feature maps, here we present the standard and depth-wise separable versions (Chollet, 2017). In a CNN the kernel weights and combination of feature maps are learnable parameters (Goodfellow et al., 2016), and many layers are applied sequentially.

**Standard convolution layer.** The standard way is to use a different kernel for each input channel, and then sum all of the output images from that, this gives one output channel (see Figure 2.8). In a convolutional layer we can choose however many output channels (or filters) we like, the more channels the more feature maps we can learn and combine, but as we learn more complex features the more likely we are to overfit (Bias-Variance tradeoff, see section 2.2.3). Mathematically this is given by,

\[
f_{ijc} = (\psi \ast x) = \sum_{m,n,d} \psi_{mn,d}x_{i+m,j+n,d}
\]

where \( i, j, c, d \) represent the rows, columns, output channels and input channels. The total number of kernels used for spatial convolutions is (no. output channels) \( \times \) (no. input channels).
2.3 Convolutional neural networks

**Figure 2.8:** Standard convolutional layer.

**Figure 2.9:** Depth-wise separable convolutional layer.

**Depth-wise separable layer.** A more memory and parameter efficient way to combine the feature maps is using depth-wise separable convolutions (Chollet, 2017). The idea is first to perform a channel-wise convolution and then a standard pointwise convolution (1 × 1 kernel), see Figure 2.9. A channel-wise convolution performs a convolution with a different kernel for each input channel. A standard pointwise convolution has the same effect as replacing the convolution with a multiplication (which is just a scaling factor) in the standard convolution. With $w_{mnd}$ representing the channel-wise convolution weights and $p_{cd}$ representing the point-wise convolution weights, this is given by,

$$f_{ijc} = (\psi \ast x) = \sum_d p_{cd} \sum_{m,n} \psi_{mnd} x_{i+m,j+n,d}$$

(13)

This essentially takes different linear combinations of the feature maps after the channel-wise convolution. Here the number of spatial kernels is equal to the number of input channels. This method gives a small loss in performance for a big save in memory which is useful for model scaling (Chollet, 2017).
2.3 Convolutional neural networks

2.3.3. Downsampling

For many problems we want to map the image into a vector of values that we want to predict. Images usually have more pixels than the length of the output vector which means we need to reduce the number of pixels throughout the CNN. An example CNN architecture which does a number of downsampling steps is shown in Figure 2.10, where the final output is a length 1000 vector of probabilities for each class that the input image could be in. There are several ways to downsample the inputs which we list here.

- **Strided convolution.** For a convolution where the output dimensions are the same, the convolution in equation 8 is evaluated at every pixel \( i, j \). If we want to halve the output length and width, we can evaluate the convolution at every other pixel e.g. only when \( i, j \) is both even.

- **Max pooling.** In the convolution, instead of taking the dot product of the kernel and image, we can take the maximum of the image pixel values inside the kernel. This gives a feature map where only the strongest neighbouring response is kept for each pixel co-ordinate \( i, j \). To downsample we can also evaluate the max pool when \( i, j \) is even. This idea can also be extended to mean pooling, median pooling etc.

- **Global max pooling.** Here we can take a global max of each image channel, leading to a vector with length equal to the number of channels. This idea can also be extended to mean pooling, min pooling etc. Global pooling is useful for fully equivariant (section 2.4) neural networks which have a better representational capacity.

- **Dense.** The most obvious method is to flatten the input image into a vector and then apply a dense feedforward layer with less output neurons than input pixels.

Figure 2.10: VGG16 architecture (Simonyan and Zisserman, 2014) with various downsampling layers and a mapping to 1000 probabilities of each class.
2.3.4. CNN architectures

There are many CNN architectures, each with a different way of combining convolutional layers. In general, the type of architecture is chosen to suit the problem you are solving. For simple problems such as classifying images of digits (MNIST dataset (LeCun and Cortes, 2010)), it only takes 3 layers of a CNN to get 97% accuracy, this is shown in many tutorials for beginners e.g. Brownlee (2018). For more complicated problems such as the ImageNet dataset (Deng et al., 2009) which classifies real life images into 20000+ categories, more layers and parameters are needed to extract all the feature combinations required for each category. ImageNet is a dataset on which new CNN architectures (or any other algorithm) are benchmarked, and almost every year there are new methods which achieves state of the art performance*. Here we list some common techniques to improve the CNN architecture, depicted in Figure 2.11.

1. **Residual blocks** (He et al., 2015b): In a residual block, an activation at some layer $l$ is added to an activation at some layer $l + n$ where $n$ is commonly 2 or 3. This is then repeated every $n$ layers. An architecture with many residual blocks is commonly called a ResNet. There are two main advantages, one is that information from earlier layers are passed onto the later layers, the second is that it helps with the vanishing gradient problem (Veit et al., 2016) allowing deeper networks to be trained.

2. **Inception module** (Szegedy et al., 2017): Here different kernel sizes are combined to form the output channels. The advantage here is that we can get different feature maps corresponding to different local scales.

3. **UNet** (Ronneberger et al., 2015): This type of architecture is commonly used for image to image problems. The architecture is composed of a downsampling block and an upsampling block with skip connections between each resolution. The advantage here is that the convolutional layers are applied with different receptive fields† meaning we’ll get feature maps at different scales. The skip connections allow information from earlier higher resolution layers to get passed onto later higher resolution layers with the same alignment. Note that the skip connections here are concatenated rather than added which is the case for the ResBlocks.

*For an updated history of CNN architectures applied to the ImageNet benchmark see: https://paperswithcode.com/sota/image-classification-on-imagenet

†Receptive field refers to how much of the neuron is connected to the original input image (via 1 or multiple consecutive layers). Here the kernels at different layers cover different proportions of the image. Since the image is downsampled by 2 every down block, the kernel size to input image size ratio doubles.
2.3 Convolutional neural networks

(a) ResBlock
(He et al., 2015b)

(b) Inception module
(Szegedy et al., 2017)

(c) UNet (Ronneberger et al., 2015)

Figure 2.11: CNN architecture techniques.
Wide vs Deep CNN. Increasing the width of the CNN architecture corresponds to increasing the number of channels or kernel size and increasing the depth corresponds to applying more consecutive convolutional layers. The general advice is that deepening CNNs improve the performance better than widening (see for example (Eldan and Shamir, 2015; Basha et al., 2020)), but this of course also depends on the dataset and the problem at hand. There is generally no consensus for explaining this, but two different arguments are as follows. The first layer only has a small receptive field and hence provides a local feature representation (e.g. whiskers of a cat) while a deeper layer with the same kernel size would have a much larger receptive field and hence can provide features with a more global representation of the image (e.g. a cat face) which is more informative. Another argument is that the number of features that you can emphasize with 1 convolution layer is limited (e.g. edges and blurring) while adding more layers effectively adds more processing steps to the features that has been emphasized, thus you can get “features of (combinations of features of (combinations ...))” instead of just combinations of features.
2.4. Geometric deep Learning

While it is true that neural networks are universal approximators (Hornik et al., 1989), there is no guarantee that the neural network will turn out to be a good model for prediction. One of the reasons why neural networks work so well is the set of priors used in designing their architectures. For image classification we know that the combination of edges will play an important role in the classification of the object in the image, therefore we would expect a convolutional neural network to be a suitable prior. In general, the most suitable priors used would depend on the underlying geometry of the data. There has been an increasing amount of research in designing neural network architectures for different types of geometries, which lead to the coining of the term *Geometric deep learning* by Bronstein (Bronstein et al., 2021). Here we will look at some key properties which will allow us to design neural networks for a general dataset with some underlying geometry.

**Geometric priors.** When designing a neural network for a given geometry of a problem, it is useful to consider how information is encoded in the feature space of a neural network. There are a few important geometric priors to consider when trying to improve the efficiency with which features are learned (Bronstein et al., 2021),

- **Symmetries:** A layer is *invariant* to transformations of the data if the outputs are the same when the data is transformed. A layer is *equivariant* when the features transform in the same way as the data is transformed (see Figure 2.7). These two symmetries in general improves the efficiency of learning representations of the data as you would expect that an object in an image is the same object when translated, we will discuss this more in the next sections.

- **Multiscale representation:** Information is encoded on many scales of an image for example the body shape of a cat is more of a global representation of a cat while its whiskers is a local feature. Being able to represent these features on many scales is a problem for high resolution images due to computational cost. A popular CNN designed for multiscale representation is UNet (Ronneberger et al., 2015) where feature maps are learned on different scales/receptive fields (see section 2.3.4).

- **Stability:** For data points within the same class, their feature maps should remain close in the feature space throughout all the layers, while different class data points should have feature maps that are far from each other (see Figure 2.12). CNNs are an example of a stable neural network which is mainly due to their equivariance.
Figure 2.12: Illustration of stability mapped onto the feature space (diagram made by the author of McEwen et al. (2022))

There are many ways to represent data and there has been an increasing number of papers for constructing neural networks to find efficient representations of the data. Bronstein (Bronstein et al., 2021) has categorized this study of geometric deep learning into 5 categories: Grids, Groups, Graphs, Geodesics and Gauges (Cohen and Welling, 2016; Zhou et al., 2020; Cohen et al., 2019). In this thesis we will mainly be dealing with Grids and Groups.

**Grids:** The data points are regularly sampled, for example in planar images each pixel is surrounded by 4 pixels. Here CNNs are the common choice as they are equivariant with respect to translation on the grid.

**Groups:** The data are represented or projected onto elements of a group (Cohen and Welling, 2016). It is useful to do this as we will see later that this leads to symmetries with respect to the group operations. A common example for this is data on the sphere, for which a spherical CNNs with rotational equivariance is a suitable choice.

### 2.4.1. Group equivariance

The planar convolutions of section 2.3 only have translational equivariance, but in most cases we would want our CNNs to also have mirror/rotational equivariance (e.g. a cat should have the similar features to a rotated cat). This can be achieved by representing our input/output co-ordinates as a group (Cohen and Welling, 2016), \( g \in G \) where \( G \) is the group that we want our convolution to be equivariant to. These are called *group convolutions*. For example, we may write the co-ordinates \( x \in \mathbb{Z}^2 \), as a translation, \( t_x \), and the input image may be represented as \( f(t_x) \) instead.
of \( f(x)^* \). Here we will give the formulation for a group convolution and then we’ll give the justification and proof.

**Discrete group convolution.** With \( f \) being the input to a layer and \( \psi \) the kernel of the layer, the group convolution of the first and subsequent layers are given by,

\[
\text{first layer: } (f * \psi)[g] = \sum_{y \in \mathbb{Z}^2} f[y] \psi[g^{-1}y]
\]

\[
\text{subsequent layers: } (f * \psi)[g] = \sum_{h \in G} f[h] \psi[g^{-1}h] \tag{14}
\]

where the square brackets denote a discrete input and \( g \in G \). For the usual planar convolutions, \( G \) is the group of translations, for mirror and rotational equivariance, we may add the mirroring and rotation groups into \( G \). The purpose of the first layer is to map the outputs onto our group domain \( g \in G \), i.e. \( (f * \psi) : G \to \mathbb{R} \). This is needed since our image only lies on the group of translations, and we need the outputs to be on \( G \) for group equivariance, this will become clearer in the equivariance proof below.

This type of convolution can be shown to be equivariant with respect to the group operations \( q \in G \). As an example, we illustrate equivariance with respect to translations and multiples of 90 degree rotations called the \( p4 \) group in Figure 2.13 which represents the first layer of equation 14. Here we show a standard equivariance proof following Cohen and Welling (2016), denoting the corresponding group transformation on functions as \( Qf[h] = f[q^{-1}h] \),

\[
((Qf) * \psi)[g] = \sum_{h \in G} f[q^{-1}h] \psi[g^{-1}h] \tag{15}
\]

\[
= \sum_{h \in G} f[h] \psi[(q^{-1}g)^{-1}h] \tag{16}
\]

\[
= (f * \psi)[g^{-1}g] \tag{17}
\]

\[
= (Q(f * \psi))[g] \tag{18}
\]

note that in the last step, it was necessary that \( q^{-1}g \in G \) because \( (f * \psi) \) only takes inputs from the group \( G \). But because of the closure property of groups, we do have \( q^{-1}g \in G \). Note that the equivariance here applies only for discrete groups, e.g. rotations by 90°, which has limited equivariance.

*In group language, this is phrased as “\( \{x\} \) is isomorphic to \( \{t_x\} \)”*
2.4 Geometric deep Learning

Figure 2.13: Illustration of group equivariance for a group convolution with $q \in p4$. Each of the four output images corresponds to a convolution with the same kernel but with the kernel rotated by 0, 90, 180 and 270 degrees.

**Continuous group convolution.** A more powerful equivariance is when the group becomes continuous which means the convolution becomes equivariant to a continuous set of operations instead of a limited discrete set. This is especially useful since objects in images have a continuous set of possible rotations. The discrete group convolution may be generalized to a continuous variant (Esteves, 2020) which is given by

$$ (f \star \psi)(g) = \int_{h \in G} f(h)\psi(h^{-1}g)d\mu(h) \quad (19) $$

where $d\mu$ is the Haar measure*. Note that the signals and co-ordinates are now continuous, and this convolution may be evaluated by transforming into Fourier space and using group convolution theorems (Esteves, 2020) which is computationally expensive. The equivariance proof here follows the same reasoning as in equation 15. The standard example where this would be useful is for spherical data, which we cover in the following section.

---

*This is the corresponding area element when talking about integrals with functions on groups. For the work presented in this thesis, we won't need to go into detail on Haar measures.
2.4 Geometric deep Learning

2.4.2. Spherical convolutions

Spherical data. There are many examples in the world where the data lie on the sphere including brain activity, cosmic microwave background radiation and global climate as illustrated in Figure 2.14. Data on the sphere or $S^2$ can be thought of as having a data point(s) for each spherical co-ordinate $(\theta, \phi)$ where this is the usual azimuth and polar angle. There are many methods for sampling these two angles for example equirectangular$^*$ sampling which does not have equal spacing on the sphere. HealPix and Icosahedral both have equal areas per grid point (Gorski et al., 2005; Jiang et al., 2019) but does not have a regular point distribution. It is well known that a regular point distribution on the sphere does not exist (Tegmark, 1996). Consequently, there is no sampling scheme that is invariant to rotations, this leads to there being no spherical convolution which has a co-ordinate mapping $S^2 \rightarrow S^2$ that is equivariant to rotations. However if we map the convolution to the rotation group SO(3) (described below) then we do have rotational equivariance (Cohen et al., 2018). Here we will first describe the rotation group before covering the spherical convolution.

Rotations of the sphere: In order to define a convolution for spheres with a spherical kernel, we have to define how to move the kernel around the sphere. This is conventionally done with rotations about the cartesian axes $x, y, z$. To cover all possible orientations of the sphere, we need at least 3 rotations about these axes. The SO(3) group may be represented by the combination of these operators. Using the ZYZ Euler convention, these operators may be written as $R = Z(\alpha)Y(\beta)Z(\gamma) \in SO(3)$, depicted in Figure 2.15 and 6.2. The rotation angles have the ranges $\alpha, \gamma \in [0, 2\pi)$ and $\beta \in [0, \pi]$. Here the axes are kept fixed after every rotation. We will be using the ZYZ Euler convention for the rest of the thesis.

*Equirectangular sampling is when $\theta, \phi \in \left\{ \frac{\pi}{2}t, \frac{\pi}{2}p \mid t, p \in \mathbb{Z} \right\}$ and $L$ is called the bandlimit. Since $0 \leq \theta \leq \pi, 0 \leq \phi \leq 2\pi$, there are $L \times 2L$ data points.
Spherical convolution. There are a few ways to perform convolutions on the sphere, here we’ll follow the rotationally equivariant way (Cohen et al., 2018). Since $S^2$ does not form a group, we cannot have rotational equivariance, but we can map our first layer onto the SO(3) group in order to gain this property. For subsequent layers, our signals are always on SO(3),

\[
\begin{align*}
\text{first layer:} & \quad (f \ast \psi)(R) = \int_{\omega \in S^2} f(\omega)\psi(R^{-1}\omega)d\omega \\
\text{subsequent layers:} & \quad (f \ast \psi)(Q) = \int_{R \in SO(3)} f(R)\psi(Q^{-1}R)dR
\end{align*}
\]

where $R, Q \in SO(3)$, $\omega \in S^2$ and $d\omega = \sin(\theta)d\theta d\phi$, $dR = d\alpha \sin(\beta)d\beta d\gamma$. These layers have rotational equivariance, i.e. $(Q(f \ast \psi))(R) = ((Qf) \ast \psi)(R)$ where $Q$ is the rotation operator for functions on SO(3) and the sphere. This may be proven in a similar manner to equation 15. We note that this a continuous convolution and is often performed in Fourier space with convolution theorems which is computationally demanding. Spherical CNNs are particularly more useful than planar CNNs when it comes to performance on unseen spherical data as the features that are extracted for one spherical image may also be extracted for any other rotated variant of the same spherical image. This is evident in the rotated MNIST experiment of (Cohen et al., 2018) where it is shown that training on images with only 1 orientation can give roughly the same accuracy on a validation set of randomly oriented images.
2.5. Summary

In this section we covered the essentials of neural networks, their formulation, how their activation functions work and how to train them. We also gave an overview of planar and group CNNs which are powerful priors when designing neural networks for image data. Due to their translational and group equivariance properties, these models are robust to translations and group operations in the data and also means they are more efficient at learning representations compared to dense neural networks. Due to this, fewer parameters are needed to gain the required performance of the model (Gerken et al., 2022).

CNNs in this thesis. Due to their predictive power on image datasets, we have decided to use and design planar CNNs to extract dose and predict physical properties in scintillation photography for radiotherapy. As mentioned above, continuous group CNNs are often computed in Fourier space which is computationally demanding. In this thesis, we work on this problem by designing a scalable, discrete-continuous group CNNs for general applications and specialize to the canonical example for group CNNs which is for spherical data.
3. Introduction to radiotherapy dosimetry

Cancer is a disease in which mutated cells in the body grow and multiply abnormally with the potential to spread to other parts of the body, thereby destroying healthy cells and disrupting the functioning of organs. In the UK, 1 in 2 people are diagnosed with cancer in their lifetime and around 167,000 die from it every year (CRUK, 2019). The main modalities for cancer treatment include surgery, chemotherapy, and radiotherapy. The type of treatment given will depend on factors such as the type of tumour and stage, but radiotherapy is given to 27% of patients who are diagnosed with cancer, for all cancers combined (CRUK, 2019).

In this chapter, we explain the physics of radiotherapy and then give a brief overview of quality assurance (QA) of radiotherapy beams with a focus on measuring beam characteristics for dose verification using scintillators.

3.1. Overview of radiotherapy

**Radiotherapy.** The idea behind radiotherapy is to deliver sufficient dose to the tumour such that it is controlled, i.e. completely eradicated or no longer cancerous. To minimize long term damage to the surrounding radiosensitive organs, the total radiation applied is split up into multiple treatments (fractions) which lets the surrounding tissues heal in the meantime. In radiotherapy, this balance between complications and tumor control is modelled by the tumor control probability (TCP) and normal tumor control probability (NTCP) (Zaider and Minerbo, 1999). The main ways to prevent cancer cells multiplying is by breaking the DNA bonds directly, either from the X-rays or from free radicals created by the X-rays. We also note that a phenomenon called radiation induced bystander effect (RIBE) also kills cancer cells, this is where non-radiated cells near the radiated cells get a similar response (Baskar et al., 2014). Radiation in the form of charged particles such as electrons and protons may also be used but in this thesis we will mainly be looking at X-rays.

**Radiotherapy Pathway.** For the best radiotherapy outcomes, it takes meticulous planning of the steps that will be taken for a specific patient. A brief summary of the radiotherapy process from diagnosis to outcome is depicted in Figure 3.1. After diagnosis, the tumour is located via 3D imaging, for example with CT or MRI scans (Villanueva-Meyer et al., 2017). Using this, a treatment plan is then made by optimizing the best intensity/directions of the X-ray beams and calculating the resulting dose distributions (Mayles et al., 2007). The dose is then measured before and during the treatment for verification of the correct dose (Mijnheer et al., 2013).
Radiotherapy is usually a part of a combination of treatments, but it has been clearly shown to improve survival depending on the type of cancer and stage (Miller et al., 2019).

### 3.1.1. Physics

When an X-ray photon is fired into water\(^*\), there is some probability that it will interact with an electron in the water at some given depth. Given a unit distance of depth, only a fraction of the X-ray photons passes through uninterrupted, this leads to an exponential decay in intensity as a function of depth. This can be formulated by the Beer-Lambert law which is given by \( I(x) = I_0 e^{-\mu x} \), where \( I \) is the intensity of the primary (unscattered) photons at depth \( x \) and \( \mu \) is the attenuation coefficient.

---

\(^*\)Water is often used for QA of radiotherapy beams as it has close radiological properties to human tissue.
3.1 Overview of radiotherapy

Interactions. There are 3 main interactions between the X-ray photons and electrons. The one being most dominant depends on the energy of the X-ray and the material. Below we list a summary of the interaction types and the photon energies at which they are most dominant in water (Khan, 2014; Mayles et al., 2007), see Figure 3.2 for a plot of their cross-sections.

1. **Photoelectric effect** 1-30keV: The incoming photon energy is completely absorbed by a bound electron. The electron is then liberated with extra kinetic energy.

2. **Compton scattering** 30keV-20MeV: The incoming photon energy is much higher than the binding energy of the electron and interacts with it as if the electron was “free”. The photon transfers some of its energy to the electron and both particles scatter in some direction.

3. **Pair production** > 20MeV: The incoming photon interacts strongly with the electromagnetic field of the nucleus and converts its energy into an electron-positron pair.

![Figure 3.2: Cross sections according to interaction type and photon energies from Mayles et al. (2007). Incoherent scattering here refers to Compton scattering from “free” electrons while Coherent scattering is from atomic electrons. Triplet production is a special case of Pair production where the interaction happens in the electric field of an atomic electron instead of the nucleus, consequently the atomic electron is ejected while an electron-positron pair is created.](image-url)
3.1 Overview of radiotherapy

**Energy transfer.** For high energy X-rays (>1MeV), the energy transferred to the electrons is large enough such that it moves far away from the interaction site\(^*\). This means that it will deposit most of its energy at a distance from where the photon-electron interaction took place, see Figure 3.3. There are two quantities which describe where the energy of the X-rays are transferred, and where energies of X-rays and charged particles are deposited (Mayles et al., 2007):

- **Kerma** (Kinetic energy released per unit mass): The kinetic energy transferred to the charged particles when liberated by ionising radiation, per unit mass, measured in gray (Gy, Jkg\(^{-1}\)). In a small volume with mass \(dm\) the kerma is \(dE_{\text{transferred}}/dm\).

- **Dose**: Energy absorbed in matter per unit mass, measured in gray (Gy, Jkg\(^{-1}\)). In a small volume with mass \(dm\) this dose \(dE_{\text{absorbed}}/dm\).

Note that when talking about kerma we’re usually talking about the energy transferred from X-rays to electrons while Dose is the energy absorbed either from electrons or X-rays. We note that the distributions of dose and kerma is different as shown in Figure 3.3b, we expect the kerma to decrease exponentially with depth since the number of X-rays decreases exponentially, but for the dose there is a build-up region which is explained in the next section.

![Illustrations of the difference between kerma and dose.](image_url)

(a) Energy transfer.  
(b) Dose vs. depth in water.

Figure 3.3: Illustrations of the difference between kerma and dose.

\(^*\)For example, the average range of a 1MeV electron in water is about 0.5mm, see reference data in Mayles et al. (2007).
3.1 Overview of radiotherapy

3.1.2. Dose distributions

Measuring dose distributions is a necessary part of commissioning and quality assurance of a radiotherapy beam as we want to make sure the dose that it is giving is correct. These distributions are usually measured in a water phantom as water closely approximates human tissue. Here we list 3 different dose distributions that are commonly measured and explain their shape and why they are important. We illustrate the detector setup and what is being varied in Figure 3.4 and the corresponding relative dose distributions in Figure 3.5. All measurements are usually measured with a constant source to surface distance (SSD).

**Percentage Depth Dose.** The PDD is obtained by moving the detector along the beam axis. There are three main characteristics to note when looking at the PDD curve. The dose build-up region to the peak (at around 15mm), the peak and the dose falloff tail. These characteristics are energy dependent. While the intensity of
3.1 Overview of radiotherapy

X-ray photons (and therefore kerma) reduces exponentially with depth, this does not mean that the dose deposited by the electrons does the same. First the X-rays transfer energy to the electrons and then the electrons are scattered in different directions for some distance and then deposits the dose. When the detector is at the surface, almost all electrons are coming toward the front of the detector (from the water) and almost no electrons from the air. As the detector moves deeper, the amount of water behind it increases, therefore the number of electrons scattering into the back of the detector also increases. This causes the dose to increase up to a certain point depending on how far the electrons can penetrate*, and how fast the X-ray intensity falls off. For higher energies the peak is further from the surface and hence less dose may be given to the skin, this is known as skin-sparing (Khan, 2014).

As the detector goes deeper, the dose falls off exponentially due to the decrease in X-ray intensity. The PDD is needed as a reference dose for treatment planning and is also a good check of the beam energy. It is also a check that the peak dose is the right distance from the skin and with tumours lying deeper in the body, we want to check the dose falls off correctly.

**Profile.** The profile is obtained by laterally (perpendicular to the beam axis) moving the detector at a constant depth. The dose falloff from 80% to 20% is called the penumbra and is generally preferable to have it be steeper as there will be less dose to surrounding tissue.

**Output factor.** As the collimator size is increased, the volume irradiated increases and the total amount of scattered radiation increases. Some of this additional scattered dose scatters back onto the central axis, increasing the dose on the axis. The output factor (OF) essentially measures this extra electron scattering by placing the detector at a fixed point while varying the collimator size. The X-ray output from the target (before the collimator) is kept constant, this is measured in monitor units (see below). The OF is important to measure as we need to know how much dose each collimator applies relative to each other; this data is then imported into the treatment planning system which calculates the dose applied to a patient. It is also a measure of how much radiation is reaching the patient from the beam head at each collimator size and how much dose is scattering in the patient.

**Monitor units.** The standard unit for the output of a beam is called monitor units (MU) and is a measure of X-ray output from the target. The MU is measured via an ionisation chamber placed between the collimator and the target. A typical definition is as follows (Mayles et al., 2007): “The monitor chamber reads 1 MU when 1 cGy is delivered to a point at the depth of dose maximum in a water

*Their range is related to their stopping power (Khan, 2014) and depends on the material and energy of the electrons.
equivalent phantom whose surface is at the isocentre of the machine (i.e. usually at 100 cm from the source) with a field size at the surface of 10cm × 10 cm. The difference between dose and MU is that dose depends on factors such as the machine, material of the phantom and distance from the source while MU only depends on the machine.

3.1.3. Quality assurance

Modern radiotherapy treatment requires time-varying 3D dose distributions in which the X-ray beam is fired at different angles and at different times. Due to the development of more complex beams in the last few decades (Ahmad et al., 2012), QA of the beams has become increasingly more important to check that the actual planned dose will be delivered precisely to the tumours and to minimise the dose to the surrounding healthy tissue.

Reference dosimetry is a common procedure for QA of the beam where the beam characteristics such as percentage depth dose (PDD) is measured and is compared with a reference dose. To make sure the dose measurements are reproducible and repeatable, the whole procedure including dose calculation, detectors used and measuring conditions are done under national standards set by the dosimetry community which include the International Atomic Energy Agency (IAEA, 2003) and the International Commission on Radiation Units and Measurements (ICRU).

Metrics. There are various metrics which compares a measured dose distribution with a reference dose distribution, for example root mean square error and mean absolute difference. For dose distributions with high gradients, these aren’t good indicators since dose curves may be close to each other that have a high difference. A better metric is a combination of distance to agreement (DTA) and the difference, denoted as r and δ respectively, see Figure 3.6. DTA is the nearest distance between a point on the measured dose distribution and a point on the reference dose distribution with the same dose value. The standard way to combine these is called the gamma (γ) metric (Low and Dempsey, 2003), given by

\[ \gamma = \sqrt{\frac{r^2}{\Delta d^2} + \frac{\delta^2}{\Delta D^2}} \]  

(21)

where \( \Delta d, \Delta D \) set the thresholds for acceptable agreement. Typical values are \( \Delta d = 2\text{mm}, \Delta D = 2\% \) (when comparing relative doses, the dose distributions are normalized by their peaks and multiplied by 100 which gives a percentage). If both r and δ are below these values, then \( \gamma < 1 \) and the measured dose is accurate with this criterion.
3.1 Overview of radiotherapy

Figure 3.6: Illustration of DTA and difference between two dose distributions, e.g. a reference and measured dose. This may also be extended to 2D and 3D dose.

**Single beam QA.** Ultimately, we want to check that the treatment plan given to a patient is correct. An indirect way is to measure the dose of a static beam fired in water with an ionisation chamber (Bouchard and Seuntjens, 2004). This checks that a single static beam is correct and therefore so should the superposition of many static beams. These checks are usually carried out before the treatment and are also checked on a regular basis.

**Multiple beam QA.** A more direct way is to measure the 3D dose of the whole treatment plan which would take significantly longer for ionisation chambers as they are point detectors. A sufficiently faster method is by measuring all the exit beams for example with an electronic portal imaging device (van Elmpt et al., 2008) and then reconstructing the 3D dose. This can be done during the treatment (in-vivo dosimetry). There are a number of detectors used for 2D or 3D dose verification for a few examples, see https://detector-group.com/radiation-therapy/.

### 3.1.4. Dosimeters

There are many types of detectors used to measure dose, each with its own pros and cons. The type of detector chosen would depend on the types of beams and what we want to characterize. Here we list some of these detectors.

- **Ionisation chambers** are standard detectors due to their precision, accuracy and rapid readout time. Inside the chamber is gas, which is ionised when irradiated with X-rays, the ionised particles are then measured by attracting them to an anode or cathode which then creates a current that you can measure. They typically go down to around 0.5cm³ in volume. Some disadvantages are that they are point detectors and are therefore slow at reading 2D/3D
3.2 Scintillation dosimetry

Scintillators are materials which emit visible light when exposed to radiation (see Figure 3.8). In scintillation dosimetry, the idea is to irradiate the scintillator, measure the light output and then reconstruct the dose from the data (Beddar et al., 1992b). Cherenkov light is also emitted during irradiation in scintillators which may sometimes obscure the scintillation signal, there has been extensive research done to correct for this, either by measuring or estimating the Cherenkov light (Frelin et al., 2008). Here we review the mechanisms of scintillation and Cherenkov light and how it can be used in a dosimetry setting.

3.2.1. Physics of scintillation

Many scintillators used in dosimetry are organic and contain benzene molecular structures (e.g. anthracene). The valence electrons of these molecules occupy singlet,
3.2 Scintillation dosimetry

Figure 3.7: Energy transitions in an organic scintillator (Knoll, 2010)

Figure 3.8: Blue plastic scintillator irradiated by a 6MV X-ray beam.

triplet and vibrational states with energy gaps on the order of 1eV which corresponds to visible light. When radiation interacts with the molecule, the valence electrons are excited and then decay to a ground state emitting scintillation light (Knoll, 2010). The decay times have a fast-component typically on the order of nanoseconds for light emitted via luminescence, caused by a singlet to singlet state decay (see Figure 3.7). The slow-component happens on the order of microseconds to hours (Knoll, 2010) emitted via phosphorescence, this is when the electrons are excited to a triplet state which slowly and indirectly decays to a singlet state. In general, the fast-component of scintillation light is emitted fast enough so that it is captured by the detector within the data acquisition time.

For comparison with standard dose measurements in water, we want to use water equivalent phantoms*. In practice this is usually done by suspending the scintillation material (fluorophores) in a medium such as plastic polystyrene (Beddar et al., 1992b). To demonstrate water equivalence, some relevant characteristics for water and a typical plastic scintillator are shown in Table 1. For X-ray irradiated plastic scintillators, the number of scintillation photons emitted is proportional to the dose deposited and is emitted isotropically (see Figure 3.9a) which simplifies the conversion from measured light output to dose. The light output of scintillators is usually computed as a percentage of light output from anthracene, typically 65%

*Water equivalence here means that they have the same effective atomic number, number of electrons per gram, and mass density as water. This leads to similar dose distributions when X-rays are used in the phantoms, and we can therefore perform similar standard QA measurements.
3.2 Scintillation dosimetry

Table 1: Physical characteristics of a plastic scintillator (with a vinlyltoluene base) compared to water (Beddar et al., 1992b).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Plastic scintillator</th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density (g cm(^{-1}))</td>
<td>1.032</td>
<td>1.000</td>
</tr>
<tr>
<td>Ratio of electron to molecular weight Z/A</td>
<td>0.5414</td>
<td>0.5551</td>
</tr>
<tr>
<td>Electron density ((10^{23} \text{ g}^{-1}))</td>
<td>3.272</td>
<td>3.343</td>
</tr>
</tbody>
</table>

For heavier particles like protons, the number of visible light photons emitted has a dependence on energy following *Birk’s law* (Birks, 1951). Because of this, the conversion from measured scintillation to dose requires correction.
3.2.2. Physics of Cherenkov light

When charged particles are moving at relativistic speeds in a polarizable medium (e.g. electrons in water), visible light photons called Cherenkov light are emitted. This is the result of the rapidly moving charged particle which momentarily polarizes nearby molecules/atoms. When those molecules return to their original state, a brief electromagnetic pulse is emitted (Jackson, 1998). In general the superposition of the electromagnetic pulses which are emitted along the particle track, interfere destructively. But, when the charged particle’s speed is greater than phase velocity of light in the medium, the electromagnetic pulses interfere constructively in a direction which lies on the surface of a cone with the central axis being the particle direction, see Figure 3.9b. In contrast, scintillation light is emitted isotropically from which it is easier to reconstruct the dose compared to the anisotropic emission of Cherenkov light. A common analogy of the Cherenkov effect is the shockwave (or sonic boom) produced by supersonic jets moving faster than sound.

The number of Cherenkov photons emitted may be computed using the Frank-Tamm formula (Jackson, 1998), given by

\[
\frac{dN}{dx} = \frac{\alpha Z^2}{\hbar c} \left(1 - \frac{1}{n^2 \beta^2}\right) d\epsilon dx
\]

(22)

\[
\frac{dN}{dx} \approx \frac{Z^2}{137 \hbar c} \left(\epsilon_{\text{max}} - \epsilon_{\text{min}} - \frac{1}{\beta^2} \int_{\epsilon_{\text{min}}}^{\epsilon_{\text{max}}} \frac{d\epsilon}{n(\epsilon)^2}\right)
\]

(23)

where \(N\) is the number of Cherenkov photons emitted, \(Z\) atomic number of medium, \(\epsilon\) is the energy of Cherenkov photons, \(\beta = \frac{v}{c}\) and \(n\) the refractive index. We can roughly estimate \(N\) in 1mm of water following Helo (2015), with \(Z \sim 1\), and for a 1.5MeV electron (\(\beta = 0.94\)) and the \(\epsilon\) range in water or wavelength range is about 350nm to 850nm which gives \(N = 28\). This is a lot less than the number of scintillation photons emitted from a scintillator which is roughly 10,000 per MeV, for comparison the range of a 1MeV electron in water is about 0.5mm (Mayles et al., 2007) which means most of the scintillation photons are emitted in that range. We therefore expect the scintillation light output to be \(\sim 10^3\) higher than Cherenkov in a scintillator. The angle of emission \(\theta\) depends on the speed and dispersion relation of the material according to \(\cos(\theta) = \frac{1}{\beta n(\epsilon)}\); for a 1.5MeV electron in water, the Cherenkov photons are emitted at angle \(\theta \approx 37^o\).
3.3 Monte Carlo simulations

3.2.3. Scintillation detectors

Scintillators have been successfully used as relative dose detectors due to their advantages of spatial resolution, low energy dependence and water equivalence. (Beddar et al., 1992b;c). However there is currently no standard to convert light output to absolute dose. This is a harder problem since we cannot measure ionised charges directly like an ion chamber. There are many experimental setups that have been successfully used for relative dosimetry, we list a few here.

- **Point scintillators** coupled to an optical fibre provides a cheap and small ~1mm detector for dose measurements but is a slow method as it is scanned through the beam with a resolution ~1mm (Fontbonne et al., 2002). This could then be extended to an array of point detectors (Guillot et al., 2011), but the disadvantage of this is the excess Cherenkov signal generated in the optical fibre and a complicated setup can be time consuming.

- **Scintillator sheet** imaging (Yogo et al., 2017; Frelin et al., 2008) is advantageous as it has a high spatial resolution due to the high amount of pixels per mm in the image which reduces dose averaging errors and also provides a fast measurement as 2D dose is captured in one image and there are less apparatus to align/setup. The disadvantage is that the sheet is placed in between plastic or water which creates an excess of Cherenkov light when irradiated with X-rays.

- **Scintillator block** imaging gives the projected image of the scintillation light. In order to reconstruct the dose, we need multiple projections. Previous attempts include using a rotating camera, which would be slow (Pogue et al., 2015) or a plenoptic camera from which they can extract depth information (Goulet et al., 2014) but has limited resolution.

3.3. Monte Carlo simulations

3.3.1. Overview

In terms of accuracy, Monte-Carlo techniques have been the standard for dose calculation in radiotherapy for many years (Verhaegen and Seuntjens, 2003). Monte-Carlo provides a statistical solution to the linear Boltzmann transport equations (Bedford, 2019) which describes the fluence of particles and their conservation of energies in some region of interest, thereby, the absorbed dose may be computed.

*For charge measuring detectors e.g. ionisation chamber, the absolute dose can be computed. For light detectors there is no direct conversion to dose, therefore light detectors are often used to measure relative dose which is the dose as a percentage of the maximum dose measured.
Monte-Carlo method. A Monte-Carlo simulation involves running an experiment with randomly sampled variables many times, from which we can gather summary statistics. A simple example would be finding the average dose deposited $\bar{d}$ in a small volume inside a water tank. We fire $N$ X-ray photons at the water tank and see how much dose is deposited in the small volume. We know the probabilities of scattering and other interactions via cross-sections and hence we can model the particle transport. If we measure $\bar{d}$ many times, the central limit theorem says the histogram of $\bar{d}$ will approach a gaussian, and as we take $N \to \infty$, the variance goes to zero. This means that we can compute the dose to arbitrary precision given we fire enough X-ray photons.

Example algorithm. We give a brief example of how a Monte-Carlo algorithm would work when transporting particles in water and calculating dose (a simplified version of Mayles et al. (2007)). Assuming we know all conditions of our experiments such as cross-sections, materials and dimensions, our algorithm is as follows (see Figure 3.10):

1. Sample primary photon energy, momentum, from some predefined sampling distribution (e.g. Bremsstrahlung energy distribution), and transport photon to first boundary.
2. Sample the distance travelled until the first interaction
3. Sample the type of interaction (Compton, photo-electric, pair-production)
4. Sample the energy, momentum of new particles (e.g. electrons from Compton scattering). Put them on the list of secondary particles to iterate through (the stack).
5. Transport scattered photon until it leaves the geometry or gets absorbed.
6. Transport secondary electrons and keep track of any other interactions and particles produced.
7. Score deposited energy in the region of interest.
8. Repeat steps 1-7 for many primary photons until the desired uncertainty is reached.
3.3 Monte Carlo simulations

Figure 3.10: Illustration of Monte-Carlo simulation of dose (edited version from Mayles et al. (2007))

3.3.2. Geant4

In this thesis we will mainly be using the Geant4 (GEometry ANd Tracking) C++ toolkit for Monte-Carlo simulations (Agostinelli et al., 2003). Geant4 is a toolkit for the simulation of particles moving through matter and has been used in numerous research domains around the world such as high energy physics, space science and medical physics. We use Geant4 as it can simultaneously model electromagnetic, Cherenkov and scintillation interactions. We also have a lot of freedom when tracking particles as it allows us to decide what to do with particles at the lowest level, e.g. kill a scintillation photon given that it has been internally reflected.

Geant4 simulation. The Geant4 algorithm is depicted in Figure 3.11. It requires the predefinition of the detector geometries and materials which is usually implemented by the user. There are many physics lists that have already been implemented depending on the particles/energies/materials that you would like to use. For example in our case, we would like to use the $G4EmLivermorePhysics$ and $G4OpticalPhysics$ physics lists which model low-energy electron and gamma transport (below 1 GeV) and Cherenkov/scintillation light which is sampled from equations 22. When all is initialized, we may run the beam and multiple events are created from which multiple tracks and steps are created. What happens in each of these scenarios may be controlled in a user defined EventAction/TrackAction/StepAction class. We may then count values such as the dose in a volume for every step which deposits dose via histograms or any C++ object.
Figure 3.11: Schematic of a Geant4 simulation process.
4. Scintillation dosimetry for CyberKnife

4.1. Introduction

In this chapter, we focus on the development of a scintillator dosimeter specifically designed for the CyberKnife radiotherapy system. The CyberKnife consists of a linear accelerator mounted on a robotic arm, enabling the radiation beam to be rotated freely on a hemisphere with a non-fixed centre. This feature sets it apart from conventional beams, which are isocentric and fixed on a plane. Additionally, the CyberKnife has the capability to generate small field sizes, reaching as low as 5mm. This characteristic proves valuable in treating small tumours, particularly in stereotactic radiosurgery (O’Beirn et al., 2018). The mobility and precision offered by the CyberKnife system are utilised in the treatment of tumours such as brain metastases.

Dosimetry and quality assurance (QA) of small-field radiotherapy beams are commonly based on point detectors such as ionisation chambers (also known as ion chambers) which is scanned inside a water phantom. Dose measurements such as the percentage depth dose (PDD) and output factors (OF) take a whole day due to the setup of the water phantom (e.g. detector and water surface alignment) and long acquisition times for sub-millimetre resolution. This is time consuming for clinical scientists and reduces availability for patients. These dose measurements are therefore taken every 6-months for the CyberKnife radiotherapy treatment delivery system, which is infrequent. For instance, one important scenario is verifying if the magnetron of the linear accelerator is generating the correct energy. Since the magnetron typically requires replacement every 2 years, there is a possibility of failure before the 6-month full QA check*. Therefore, a faster QA process is necessary to enable more regular checks, which instills greater confidence that the beam is functioning as intended. Furthermore, the apparatus is expensive and has a range of problems in the small-field regime. In fact, the definition of small field originates from these problems (IAEA, 2017):

- **Loss of charged particle equilibrium**: The range of electrons is typically much smaller than the detector size, but for small detectors the range can be bigger than the detector size. This means the number of charged particles entering the chamber is not equal to that leaving. The charge to dose calculation of the ion chamber will then have to be corrected.

- **Occlusion of primary photon source by the collimator**: The source is

*Note that a more regular check for the beam energy is done using a two point measurement approach, but a full PDD curve would give more confidence that the energy is correct.
typically much smaller than the collimator size. As the collimator size approaches the size of the source, far fewer X-ray photons get through which leads to a big drop in output dose (see IAEA (b)). This is in comparison to when the collimator size is much larger than the source, then changing the collimator size does not produce a drastic change in output dose.

- **Volume averaging**: For detectors that are much smaller than the beam width, the dose is approximately constant in the detector and hence the dose being calculated is the dose at the position of the detector. But if the detector size is similar or larger than the beam width, then the dose will vary across the detector leading to a measurement of average dose rather than point dose.

With the use of a scintillating phantom and DSLR camera, we aim to design a faster, cheaper and higher resolution detector system which can be used to complement current QA methods. The dose is extracted from the light output of the scintillator which solves the loss of CPE problem because we are not measuring charge. We can also solve the volume averaging problem as the images are high enough resolution such that the pixel size is much smaller than the beam width (typically 0.1mm per pixel). We cannot solve the second problem as that is to do with the collimator and source rather than the detectors used.

**QA for the CyberKnife.** Our experiments are based on the QA of the CyberKnife system which produces fixed collimator sizes\(^*\) from 5-60mm and operates at an energy of 6MV. One of the QA checks of the CyberKnife is the Output Factor of the beam\(^†\). This is the dose measured per monitor unit (MU) at a single point as a function of field size. For the CyberKnife this is measured at a point 15mm deep using a 785mm source to surface distance (SSD). The OF is important to measure as we need to know how much dose each collimator applies relative to each other, this data is then imported into the treatment planning system which calculates the dose applied to a patient. Due to the small field sizes of the CyberKnife, high resolution detectors of sub-millimetre sizes are needed to account for dose volume averaging. The most accurate point detectors currently used to measure output factor, PDD and profiles, are diodes and ion chambers (Chalkley and Heyes, 2014), in Figure 4.1 we show measurements made by the microDiamond diode. They are typically scanned through the beam with a spatial resolution of 1mm, and can take a whole day to obtain the dose curves. They are checked every 6-months, but with the use of high-resolution scintillator detectors there is potential for faster and more regular dose verification.

\(^*\)The collimator size corresponds the full width half maximum at a depth of 15mm and source to surface distance 785mm.

\(^†\)See section 3.1.2 for definitions of output factor, PDD, profile, SSD and monitor units.
4.1 Introduction

(a) PDD

(b) Profile

(c) Output factor

Figure 4.1: Relative dose distributions of the CyberKnife, measured using a micro-Diamond diode (Chalkley and Heyes, 2014).

Scintillation and Cherenkov dosimetry. Scintillators have been successfully used as dosimeters due to their advantages of spatial resolution, low energy dependence and water equivalence (Beddar et al., 1992b;c). They emit light isotropically and the amount emitted is proportional to the dose deposited up to an energy limit. The use of scintillators as point detectors coupled to an optical fibre provides a cheap and small $\sim 1\text{mm}$ detector for dose measurements (Fontbonne et al., 2002; Gonod et al., 2021; Thrower et al., 2022) and has been shown to work well for measuring output factors of the CyberKnife (Morin et al., 2013). But for spatial dose, it is a slow method as it is scanned through the beam with a resolution $\sim 1\text{mm}$. This could then be extended to an array of point detectors (Guillot et al., 2011; Hupman et al., 2022), but the disadvantage of this is the excess Cherenkov signal generated in the optical fibre and a complicated setup can be time consuming. The photography of large blocks of scintillating phantoms ($\sim 20^3\text{cm}^3$) have also been tested but here light is captured close and far from the camera and so we get the total projected light (Almurayshid et al., 2017). This is undesirable as we would want a point dose distribution ideally from the axis of the beam. The shelf-life of a large volume scintillator is typically a few years, but they are known to discolour and reduce in light output over time, e.g. light output loss of 25% after 2 years (Hamel, 2021). Liquid scintillating phantoms have to be containerized e.g. with plastic and so the beam has to pass through some non-scintillating material which needs correction (Almurayshid, 2016), for plastic scintillating phantoms, the whole phantom is homogeneous. Cherenkov light may also be used to reconstruct dose. By irradiating quinine water, the Cherenkov light is re-emitted isotropically (Pogue et al., 2017), but this needs photographs from different angles to reconstruct the dose which is time-consuming.

Scintillator sheets. Scintillating sheets (Yogo et al., 2017; Frelin et al., 2008) allows the photography of light coming from a 2D slice of the beam which in turn
4.1 Introduction

gives a 2D dose image. It is advantageous as it has a high spatial resolution due to the high amount of pixels per mm in the image which reduces dose averaging errors. It also provides a fast measurement as 2D dose is captured in one image and there is less apparatus to align/setup. The disadvantage is that the sheet is placed in between plastic or water which creates an excess of Cherenkov light when irradiated with X-rays (discussed in section 4.2.2). Previous methods to discriminate the scintillation from Cherenkov light includes spectral analysis (Frelin et al., 2005), subtracting off the Cherenkov with an image without the scintillating sheet (Yogo et al., 2017; Beddar et al., 1992a) and the use of a checkerboard black paint pattern on the sheet (Frelin et al., 2008) which limits the spatial resolution to 2mm.

Our method. We aim to design a detector system for the CyberKnife which follows a list of desirable criteria:

1. Water equivalence.
2. A spatial resolution better than 0.5 mm to allow high dose gradients to be plotted.
3. Linear response with dose and dose rate.
4. Able to detect from multiple points simultaneously or rapidly so that a PDD or profile can be obtained efficiently.
5. Able to be automated so that multiple collimators can be assessed efficiently.
6. Ease of use.
7. Capable of measuring PDD with a gamma pass rate 90% for a criterion 2mm/2%.
8. Capable of measuring OF to within 2% of the reference dose.

We propose a detector system composed of a scintillator sheet sandwiched in between plastic blocks of the same plastic base as the scintillator. This is imaged using a commercial SLR camera. This provides a cheap and accessible method. However, we use thin scintillator sheets to take slices of small beams so as to ensure that all scintillation light comes from the same imaging plane. This leads to a lower scintillation light output which means they are noisier and background signals become more significant. We use a number of image processing techniques to account for this and we use a subtractive Cherenkov discrimination method similar to Yogo et al. (2017) but we use polystyrene blocks instead of water. With this system the relative OF, PDD and profile curves can be obtained in a couple of hours. In the following sections we show that we can achieve criteria 1-6 using our method. We cannot achieve 7-8 using a manual image processing method as our curves are highly affected by noise and optical artefacts. However, in chapter 5 we show that we can
achieve the final two criteria with machine learning methods.

4.2. Materials and Methods

Due to the low light output of our scintillator sheets and the use of a commercial camera, our method heavily relies on image processing which we outline in this section along with the experimental setup.

4.2.1. Experimental setup

**Setup.** The experimental setup is shown in Figures 4.2-4.3. The main part of the setup is in a plastic Peli protector case (Peli UK, 2022) with a similar material to perspex. We have two clear polystyrene blocks of 200x200x50mm sandwiching a scintillator sheet of 200x200x2mm which is held tight together by an adjustable holder. The case is shut so that no light can leak in. It has a 1mm thick window (made of the same plastic as the case) for the beam to pass with minimal scatter. We use water-equivalent, green and blue scintillator sheets manufactured by Nuvia (Hamel et al., 2020) (see specifications in Figure 4.4). We chose to experiment with green scintillators as the background Cherenkov light has a lower output at green wavelengths (Frelin et al., 2008). To minimize light leakage into the case, the camera is connected via a USB cable going into a patch panel and then into a data acquisition laptop which is in the treatment room. This is controlled in the control room via an ethernet into a network hub. We connect another laptop via the network hub to acquire and analyze data in real time.

**Beam.** Here we list the standard specifications used for the QA of our CyberKnife beam. The CyberKnife is a linear accelerator which generates a 6MV X-ray treatment beam without a flattening filter. The dose rate is 10 Gy min\(^{-1}\) at 15mm depth with a 60mm collimator beam at 785mm SSD (see Accuray, (2018) for full specifications). It has a constant dose rate of 1000 ± 15 MU/min and is controlled on a PC in the control room. The photographs are taken while the beam is continually on so that the beam is essentially always on during the acquisition time of the camera. We take photographs with different collimators sizes [5, 7.5, 10, 12.5, 15, 20, 25, 35, 40, 50, 60]mm which are changed by hand. All measurements are taken at 785mm SSD which is the standard for taking output factor measurements with a water tank and ion chamber.
4.2 Materials and Methods

CyberKnife beam 6MV
Plastic case
Camera scintillator sheet
$2 \times 200 \times 200 \text{mm}^3$
Polystyrene blocks
$50 \times 200 \times 200 \text{mm}^3$ each
Jack

Figure 4.2: Experimental setup.
4.2 Materials and Methods

(a) Inside case

(b) Treatment room

Figure 4.3: Photos of experimental setup

<table>
<thead>
<tr>
<th></th>
<th>Standard Blue-Emitting Scintillators (SP32)</th>
<th>Green-Emitting Scintillators (SP33)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polymer base</td>
<td>polystyrene</td>
<td>polystyrene</td>
</tr>
<tr>
<td>Density</td>
<td>1.03 g/cm$^3$</td>
<td>1.03 g/cm$^3$</td>
</tr>
<tr>
<td>Refractive index</td>
<td>1.57</td>
<td>1.57</td>
</tr>
<tr>
<td>Softening point</td>
<td>70 - 75$^\circ$C</td>
<td>70 - 75$^\circ$C</td>
</tr>
<tr>
<td>Light output (relative to anthracene)</td>
<td>56%</td>
<td>55%</td>
</tr>
<tr>
<td>Decay time</td>
<td>2.5 ns</td>
<td>4.4 ns</td>
</tr>
<tr>
<td>Wavelength of maximum emission</td>
<td>425 nm</td>
<td>503 nm</td>
</tr>
</tbody>
</table>

Figure 4.4: Scintillator specifications (Nuvia, 2019).
4.2 Materials and Methods

**Camera configuration.** While cameras with CCD sensors are less susceptible to noise (Theuwissen, 2001), they are more expensive in general. Hence, we use a SLR Nikon D7500 camera with a CMOS censor. To get a wide field-of-view and capture the whole scintillator we use a lens with 20mm focal length and f-number 5.6. To ensure that the plastic blocks and scintillator sheet are in focus, we place a ruler at the front and back of the blocks and check that the numbers on the ruler are in focus by eye. The distance from the sheet and the sensor is 53cm which is close enough to capture enough light and far enough to capture the whole scintillator. To keep the camera in place, it is screwed on a jack which is height-adjustable. The optical axis is aligned with the top surface of the blocks to avoid capturing internally reflected light from inside the top surface. The alignment is done via plotting a central crosshair on a captured image and checking it is aligned with the top surface. To get as much light as possible and hence lower noise, we choose ISOs and exposure times such that the images are well exposed without saturating. We choose a constant 400 ISO and exposure times listed in Table 2. The output files are 14-bit images. Note that for image processing in python, we scale all images to 16-bit images, i.e. they have the same relative value to the maximum pixel value.

Table 2: Shutter speed of the camera used for each collimator size.

<table>
<thead>
<tr>
<th>Collimator size (mm)</th>
<th>Shutter speed (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>7.5</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>12.5</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>13</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>60</td>
<td>10</td>
</tr>
</tbody>
</table>
4.2 Materials and Methods

4.2.2. Background signals

The case is “light tight” meaning there is no measurable light when the beam is off. But there are still many sources for background signals and artefacts. We list them here with a given correction that we make in the image processing.

**Cherenkov light.** The largest source of error we know is due to Cherenkov light produced in the polystyrene blocks, illustrated in Figure 4.5. We want to measure the scintillation light in the 2mm sheet in order to give us a 2D slice of the beam. When scintillating material is irradiated with high energy X-rays, it is well known that the light emitted is composed of $> 99\%$ scintillation and $< 1\%$ Cherenkov (see for example Almurayshid (2016) or section 3.2.2). However we are using a thin slice and therefore as the collimator size grows larger the amount of Cherenkov light emitted in the polystyrene blocks grows with $(\text{collimator size})^2$ and the total amount of scintillation light emitted only grows with the collimator size. The images that we measured is therefore a projection of the Cherenkov light from the volume of the beam and the scintillation light from a plane through the beam, see Figure 4.5. Rather than being $< 1\%$ of the total light output, the Cherenkov light is now of similar magnitude to the scintillation light. To correct this, we take an image with and without the scintillator sheet, and subtract the images to get a final image which represents the scintillation light, see section 4.2.3.

![Figure 4.5: Projected light is composed of Cherenkov from all 3D plastic components, and scintillation from the 2D scintillator sheet.](image-url)
4.2 Materials and Methods

**Background light.** We take images with the case shut and the beam off, we measure a mean background level 21.8±163.7 at the highest exposure 30s, which is $3 \times 10^{-4}$ of the max pixel value $2^{16} - 1$. Furthermore, 99.97% of all pixels are less than 1% of the max pixel value and 94.9% of all pixels are less than 0.1% of the max pixel value. From this we can conclude that there is negligible background light when the beam is off. A comparison of a beam on/off image is shown in Figure 4.6.

![Beam on vs beam off images (green channel).](a) 30s exposure, beam on  ![Beam on vs beam off images (green channel).](b) 30s exposure, beam off

Figure 4.6: Beam on vs beam off images (green channel).

**X-ray scattering.** As we increase the exposure time, we collect more light but we also collect more X-ray photons scattered from the block which appears as salt and pepper noise. This noise covers the whole image as shown by the blue/purple background in Figure 4.6a. We take the mean pixel values of the top half of the images and plot the mean noise level with exposure time in Figure 4.7. To correct for this, we simply subtract the mean X-ray scattering noise from the images given their exposure time.

![Background X-ray scatter as a function of exposure time.](error bars shown (but too small to see) as standard deviation over 5 images.)

Figure 4.7: Background X-ray scatter as a function of exposure time. Error bars shown (but too small to see) as standard deviation over 5 images.
Photon counting noise. As the number of photons reaching the sensor cavities (or “photosites”) are random and increases with the exposure time, the pixel values will vary according to Poisson noise. We therefore take 5 images and correct this with a pixel-wise median over the 5 images (see section 4.2.3).

4.2.3. Image processing

We use the rawpy python package (Riechert, 2018) for processing raw images. It allows us to load raw sensor data without the application of post-processing on the camera such as gamma correction and white balance. For further processing of the images, we use the OpenCV package in python (Bradski, 2000). We list a number of image processing steps here.

Denoising. We experiment with different methods of denoising summarized in Table 3, which shows the mean noise to signal ratio. We observe a better noise to signal ratio when using either a pixel-wise median or minimum over the 5 images. This is due to the largest source of noise being additive. We further use median filtering with a $3 \times 3$ kernel for a smoother image. To clarify, the first median is between images, and the second median is within an image. The denoised images and profile is illustrated in Figure 4.8-4.9. We also experiment with a minimum filter which takes the minimum over the $3 \times 3$ kernel instead of a median. Since most of the noise is additive, we also expect this to work well (see Table 3). For a completely smooth image, the minimum filter would actually make the beam narrower since it chooses a lower value on the beam edges. Therefore, we chose the processing steps of Median over 5 images + 5 median filters which produces the best noise to signal.

Alignment. To extract a more accurate dose, we need to take slices of the image in the correct positions which means we need to align the image with the beam. The beam axis is aligned with the image central axis via a laser pointer which goes through the axis of the beam. The image is rotated such that it is aligned with the laser line. We convert from pixels to mm by imaging a ruler placed on the sheet covering 200mm resulting in 0.09mm per pixel. The PDD is extracted via slicing through the centre of the beam (centre obtained by the peak profile position). To reduce noise, we take the average pixel value over a 1mm thick slice. Similarly, the profile is extracted at 15mm depth which is also averaged over a 1mm thick slice. For the Output factor we take the average pixel value over a $1 \times 1 \text{mm}^2$ box at 15mm depth.
4.2 Materials and Methods

Table 3: Noise to signal with different methods of denoising. The true signal is obtained via fitting with a 3-degree polynomial on a sliding window of size 20 pixels (Savitzky-Golay filter). The noise to signal for a given pixel value is then (True signal-Measured signal)/(True signal) i.e. the fractional difference between the true and measured signal. The mean is taken over all pixel values within over 0.05 of the max pixel value.

<table>
<thead>
<tr>
<th>Processing steps</th>
<th>Mean noise to signal ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>0.60 ± 0.80</td>
</tr>
<tr>
<td>Mean over 5 images</td>
<td>0.34 ± 0.40</td>
</tr>
<tr>
<td>Mean over 5 images + median filter</td>
<td>0.20 ± 0.22</td>
</tr>
<tr>
<td>Mean over 5 images + 5 median filters</td>
<td>0.13 ± 0.13</td>
</tr>
<tr>
<td>Median over 5 images</td>
<td>0.29 ± 0.46</td>
</tr>
<tr>
<td>Median over 5 images + median filter</td>
<td>0.071 ± 0.13</td>
</tr>
<tr>
<td>Median over 5 images + 5 median filters</td>
<td>0.038 ± 0.056</td>
</tr>
<tr>
<td>Min over 5 images</td>
<td>0.035 ± 0.043</td>
</tr>
<tr>
<td>Min over 5 images + median filter</td>
<td>0.018 ± 0.020</td>
</tr>
<tr>
<td>Min over 5 images + 5 median filters</td>
<td>0.013 ± 0.014</td>
</tr>
</tbody>
</table>

Figure 4.8: Denoising an image via pixel-wise median over 5 images and median filtered 5 times. Note that the distances shown on the axes is at the scintillator sheet.

Figure 4.9: Profile of raw and denoised signal
4.2 Materials and Methods

Cherenkov subtraction. After processing and aligning all images similarly, we subtract images with and without the scintillator sheet. An example processed image is shown in Figure 4.10 with and without a green scintillator sheet and with the Cherenkov subtracted.

(a) Scintillation + Cherenkov, green scintillator sheet

(b) Cherenkov, no sheet

(c) Cherenkov subtracted, showing output from green scintillator.

Figure 4.10: Example images for field size 30mm showing the red, green and blue channels. Note we have normalized the images here according to the exposure time calibration curve (see section 4.2.4). Figure (c) is the result of subtracting (b) from (a). Since some pixel values become negative after subtraction, we have clipped values to be between 0 and infinity so that the dose is always positive.
4.2.4. Corrections

**Refraction.** Due to refraction the depths of the pixel values are only apparent depths, hence when we take a slice of our images, we will get a dose vs. apparent depth curve. With basic geometry and Snell’s law, we convert this to a dose vs. real depth curve.

**Light intensity.** The number of photons emitted from a point is proportional to the dose deposited at that point. But the light intensity drops as you move further from the point by $1/\text{distance}^2$. This is corrected by multiplying the depth dose curves by the distance$^2$ from the lens to the depth at which the light was emitted.

**Exposure times.** The Output factor requires images taken with different collimator sizes and these require different exposure times. However, the exposure time with pixel intensity is not completely linear$^\ast$. In Figure 4.11 we measure the mean pixel intensities (over the whole image) from a constant light source over different exposures and average over 5 images (error bars shown as standard deviation). We use this calibration curve to normalize the scintillation images by dividing by the average pixel intensity on this curve given the exposure time.

![Exposure calibration curve](image)

Figure 4.11: Exposure calibration curve.

$^\ast$This could be for a number of reasons but it is not completely clear, e.g. the amount of thermal noise may be different for each exposure time or leakage of electrons between cells in the sensor.
4.2 Materials and Methods

**Dose light linearity.** When we convert from light output to dose, we assume that they are proportional. To test this, we show that the pixel values are proportional to the dose applied (measured in amount of MU applied) in Figure 4.12. Note that these images are measured with the same exposure time, and so the exposure to pixel non-linearity above has no effect here. This curve is obtained by taking a separate image for each different amount of MU applied, with a constant 20s exposure for each image and 20mm collimator size. The plot is fitted with a correlation coefficient of 1.

![Figure 4.12: Dose vs pixel intensities.](image)

4.2.5. Monte-Carlo Simulations

Simulating our experimental setup gives us insights into the data as it allows us to compare with measured data and see what differences arise. We will compare our simulated Cherenkov images with the measured images and compare the separate simulated scintillation image with the measured dose. We simulate our experimental setup using the Geant4 (Agostinelli et al., 2003) simulation toolkit for particle physics which is a standard choice for simulating dose. In addition, we also use Geant4 to simulate our scintillation and Cherenkov images via photon counting. For low-energy electron and gamma transport (below 1 GeV), the *G4EmLivermorePhysics* physics class was used. This method was validated by comparing our simulated dose to measured dose for which we find agreement to better than 2% (see Appendix B.1 for details). To simulate emission of scintillation and Cherenkov photons we use the *G4OpticalPhysics* class.

**Image simulation algorithm.** We use a “thin lens approximation” in the sense that the photon rays emitted from a point on the focus plane are refocused on to the
sensor of the camera after refraction from the lens. If an object is far from the focus plane, then that object would appear blurry. For a perfect lens with no aberrations, we don’t need to simulate the lens refraction and sensor. This is because the position of “photon hits” on the sensor can be retraced back to a (virtual) focus plane (see Figure 4.13), which is an inverted mirror image of the photons hitting the sensor. Counting the photons on a 2-D histogram on the focus plane then corresponds to counting photons on the sensor. The 2-D histogram is the final image with pixel values being the number of photon counts on each pixel of the 2-D histogram. The algorithm is presented here for clarity:

1. Optical photon created via scintillation or Cherenkov and is refracted/reflected
2. If the photon hits the lens, get the direction of the photon
3. Trace back in the opposite direction up to the focus plane
4. Count photon on a 2-D histogram on the focus plane

![Image simulation algorithm.](image)

Simulation details. For simplicity and computational speed, the 2-D histogram bins are 1mm × 1mm boxes on the focus plane, which means each pixel length corresponds to 1mm length on the focus plane. Similar to our experimental setup, we have used a polystyrene plastic base. There is a tradeoff between increasing the aperture which gives us more light signal, and decreasing the aperture which increases the depth of field and we would ideally like the whole plastic block to be in focus. We have chosen an aperture diameter of 4.1cm which is small enough to contain the blocks within the depth of field (DOF) given the pixel sizes (see Appendix B.2 for validation of the DOF).
4.2 Materials and Methods

**Simulation speedups.** The simulation was heavily slowed down due to some photons being internally reflected thousands of times in the scintillator before being absorbed. These photons are trapped in the scintillator and never captured by the lens which means they make no difference to the image. By killing these photons at their moment of creation the simulation speed was $\sim 10\times$ faster.

Most optical photons that are created in the simulation never reach the camera lens and are therefore unnecessary to simulate. By simulating only optical photons which hit the lens we can reduce the number of computations significantly*. There are two ways to reduce the computation time which we present in Figure 4.14. Scintillation photons are emitted at a random direction $\theta \in [0, \pi], \phi \in [0, 2\pi]$ where $(\theta, \phi)$ are spherical co-ordinates see Figure 4.14a. We can reduce this range such that it always contains the lens, i.e. the photons are now randomly emitted at a random direction $\theta \in [\theta_{\text{min}}, \theta_{\text{max}}], \phi \in [\phi_{\text{min}}, \phi_{\text{max}}]$, see Figure 4.14b. We then further reduce the photons created by computing their trajectory via Snell’s law and only creating the photon if it reaches the lens, see Figure 4.14c. This removes the background computations in Geant4 when a particle track is created, e.g. computing scattering probabilities and volume tracking. The final implementation improved simulation times from 1 week to about 10 hours per image with $\sim 10^5$ Cherenkov and scintillation photons collected. Here we use an Intel Xeon E5-2430 2.7 GHz CPU. This is a significant improvement, for example compared to Helo et al. (2014) with $\sim 10^5$ Cherenkov photons collected in 2 weeks simulation time.

---

*Approximately reduced by $\text{(solid angle of the lens)}/4\pi \approx \frac{\pi(2.1\text{cm})^2}{(50\text{cm})^2} \approx 0.0004$
4.2 Materials and Methods

(a) Photons sampled isotropically

(b) Photons sampled in a restricted range

(c) Photons sampled in a restricted range and only created if it hits the lens

Figure 4.14: Photon trajectory cuts
4.3. Simulations

4.3.1. Scintillation image vs. Dose

To see if our scintillator sheet method would work with no optical artefacts in theory, we compare our simulated scintillation image with the measured dose. Figure 4.15 shows our simulated scintillation images with no Cherenkov. The profiles are clearly flatter because the light is coming from a sheet rather than the whole plastic block. From these images we may extract the dose distributions in a similar fashion to the measured images and compare to the microDiamond diode measurements. For the output factors (Figure 4.16), we have a mean absolute difference (MAD) of 0.67 ± 0.54 % and maximum difference 2.03%. For the PDD, we have a MAD of 2.03 ± 2.08 % (Figure 4.17). For the profiles, we have a MAD of 0.94 ± 1.76 % (Figure 4.17). The dose and errors are obtained by averaging over 8 images. Overall, the simulated scintillation images are in good agreement with the measured microDiamond dose distributions.

Figure 4.15: Scintillation images with no Cherenkov.

Figure 4.16: Output factor extracted from simulated scintillation images compared with the microDiamond measurements.
4.3 Simulations

Figure 4.17: Comparison of PDD and profile extracted from simulated scintillation images and measured with the microDiamond. The dose and errors are obtained by averaging over 8 images.
4.3 Simulations

4.3.2. Cherenkov image comparison

Figure 4.18 shows a comparison of measured and simulated Cherenkov images (no scintillator sheet), where we have also plotted the image difference. For fair comparison, the images are normalized by the maximum pixel value so that the maximum is 1. In the difference plot, white means the images are equal, red means the measured image is higher, and blue means the simulated image is higher. Here we see that the measured light is higher for deeper depths compared to the simulated light and we suspect this is due to optical scattering. In the blue channel of Figure 4.10b (i.e. where most of the Cherenkov light is), there is clearly scattered light coming from the volume of the block outside the beam, here we simply subtract this with a constant for fair comparison with the simulation. But there should also be scattered light coming from the centre which we suspect is the reason for the difference in the image for deeper depths.

To quantify the image difference, we could use the MAD but because most of the pixel values are zero outside the beam, this brings the MAD close to zero. As we only care about non zero pixel values i.e. where the beam is, we choose an alternative image difference metric via a total absolute difference percentage,

$$\epsilon = 100 \times \frac{\sum_{ij} |M_{ij} - S_{ij}|}{\sum_{ij} |M_{ij}|}$$

where $M_{ij}, S_{ij}$ are the measured and simulated image pixels with $i,j$ as the row and column index. This gets the total pixel differences as fraction of the total light output of the measured image. We get a $\epsilon = 23 \pm 10\%$ averaged over all field sizes, indicating a noticeable difference between the simulated and measured image. As mentioned above we suspect this is due to optical scattering.
4.4 Results

In this section we present our results for extracting PDD, profile and output factors from our measured scintillation and Cherenkov images. We have applied the necessary image processing and corrections steps outlined above, but our results show that the main factor which needs improvement is the noise. This is due to the low light output of our thin scintillator sheets.

4.4.1. PDD

We compare relative PDDs measured by our scintillation images (after Cherenkov subtraction) and a reference dose measured by a microDiamond diode scanned in a water tank (Chalkley and Heyes, 2014) which is currently the standard method for the CyberKnife at Birmingham hospital. Qualitative results are shown in Figure 4.19. The figure qualitatively shows a better signal for the green scintillator compared to the blue. The green has a better noise to signal ratio and follows the reference dose, this is due to Cherenkov mostly being emitted in the blue channel which means the blue signal is affected more by the Cherenkov signal. We summarize the differences for each collimator size in Table 4. We compute the MAD and max difference, which clearly shows the green scintillator perform better. While the green follows the reference dose, it is far too noisy to achieve a high accuracy for a Gamma criterion of 3%/3mm as shown in Table 4. Typically, a good passing rate is over 90% for a Gamma criterion 2%/2mm (e.g. Li et al. (2021); Dechambre et al. (2017)) which means 90% of the measured dose points are within the radius of 2mm
and 2% difference from the reference dose. The differences in reference and measured dose here is inadequate for QA and we therefore investigate machine learning methods in section 5.

Table 4: PDD differences between scintillation and microDiamond measurements. The reference PDD and measured PDD are normalised by their maximums. Since the signals are noisy, we have used a curve fit to estimate the maximum using a 3-degree polynomial with a sliding window of size 20 pixels (Savitzky-Golay filter). The difference is then between the normalised PDDs. Here we give the MAD over all points on the PDD curve and the maximum absolute difference. We also include Gamma passing rates for the green sheet as it roughly follows the reference curves in Figure 4.19 while the blue sheet does not.

<table>
<thead>
<tr>
<th>Collimator size</th>
<th>Blue sheet</th>
<th>Green sheet</th>
<th>Gamma (3mm/3%)</th>
<th>Gamma (2mm/2%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAD</td>
<td>Max diff.</td>
<td>MAD</td>
<td>Max diff.</td>
</tr>
<tr>
<td>5mm</td>
<td>0.12 ± 0.08</td>
<td>0.12</td>
<td>0.04 ± 0.03</td>
<td>0.04</td>
</tr>
<tr>
<td>7.5mm</td>
<td>0.15 ± 0.08</td>
<td>0.15</td>
<td>0.03 ± 0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>10mm</td>
<td>0.14 ± 0.08</td>
<td>0.14</td>
<td>0.03 ± 0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>12.5mm</td>
<td>0.14 ± 0.08</td>
<td>0.14</td>
<td>0.03 ± 0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>15mm</td>
<td>0.13 ± 0.08</td>
<td>0.13</td>
<td>0.02 ± 0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>20mm</td>
<td>0.15 ± 0.08</td>
<td>0.15</td>
<td>0.02 ± 0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>25mm</td>
<td>0.15 ± 0.09</td>
<td>0.15</td>
<td>0.03 ± 0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>30mm</td>
<td>0.15 ± 0.09</td>
<td>0.15</td>
<td>0.03 ± 0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>35mm</td>
<td>0.17 ± 0.10</td>
<td>0.17</td>
<td>0.03 ± 0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>40mm</td>
<td>0.16 ± 0.09</td>
<td>0.16</td>
<td>0.03 ± 0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>50mm</td>
<td>0.14 ± 0.09</td>
<td>0.14</td>
<td>0.03 ± 0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>60mm</td>
<td>0.16 ± 0.10</td>
<td>0.16</td>
<td>0.06 ± 0.04</td>
<td>0.06</td>
</tr>
</tbody>
</table>
4.4 Results

(a) 5mm

(b) 30mm

(c) 60mm

Figure 4.19: PDDs after Cherenkov subtraction for different collimator sizes. We have normalised by dividing by the maximum. Since the signals are noisy, we have used the Savitzky-Golay filter to estimate the maximum.
4.4 Results

4.4.2. Profiles

We compare relative profiles measured by our scintillation images (after Cherenkov subtraction) and a reference dose measured by a microDiamond diode (Chalkley and Heyes, 2014). Qualitative results are shown in Figure 4.20. The figure qualitatively shows a better noise to signal for the green scintillator compared to the blue, but neither sufficiently follow the reference dose curve due to the high amount of noise and optical artefacts. We also observe a higher tail as the field size grows larger. The cause of this has been attributed in the past as optical scattering in the plastic (Petric et al., 2006; Beddar et al., 2009; Ponisch et al., 2009), but it is not clear or proven what the cause is. It has been corrected for in the past with a deconvolution (Petric et al., 2006). We also summarize the differences in Table 5 which shows that the green sheet measurements are closer to the reference dose compared to the blue sheet.

Table 5: Profile differences between scintillation and microDiamond measurements. Similar to the PDD metrics, the reference profile and measured profile are normalised by their fitted maximums using the Savitzky-Golay filter. The difference is then between the normalised profiles. Here we give the MAD over all points on the profile curves and the maximum absolute difference.

<table>
<thead>
<tr>
<th>Collimator size</th>
<th>Blue sheet</th>
<th>Green sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAD</td>
<td>Max diff.</td>
</tr>
<tr>
<td>5mm</td>
<td>0.05 ± 0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>7.5mm</td>
<td>0.05 ± 0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>10mm</td>
<td>0.05 ± 0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>12.5mm</td>
<td>0.07 ± 0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>15mm</td>
<td>0.06 ± 0.04</td>
<td>0.06</td>
</tr>
<tr>
<td>20mm</td>
<td>0.07 ± 0.05</td>
<td>0.07</td>
</tr>
<tr>
<td>25mm</td>
<td>0.09 ± 0.05</td>
<td>0.09</td>
</tr>
<tr>
<td>30mm</td>
<td>0.08 ± 0.06</td>
<td>0.08</td>
</tr>
<tr>
<td>35mm</td>
<td>0.10 ± 0.06</td>
<td>0.10</td>
</tr>
<tr>
<td>40mm</td>
<td>0.11 ± 0.06</td>
<td>0.11</td>
</tr>
<tr>
<td>50mm</td>
<td>0.13 ± 0.07</td>
<td>0.13</td>
</tr>
<tr>
<td>60mm</td>
<td>0.16 ± 0.08</td>
<td>0.16</td>
</tr>
</tbody>
</table>
Figure 4.20: Profiles after Cherenkov subtraction for different collimator sizes. We have normalised by dividing by the maximum. Since the signals are noisy, we have used the Savitzky-Golay filter to estimate the maximum.
4.4 Results

4.4.3. Output Factors

We compare relative output factors at 15mm depth measured by our scintillation images (after Cherenkov subtraction) and a reference dose measured by a microDiamond diode (Chalkley and Heyes, 2014). Qualitative results are shown in Figure 4.21. Typical acceptable values for measuring output factors for the CyberKnife are within differences of 0.01 or 0.02 (see e.g. Al Kafi et al. (2023); Manavalan et al. (2021)). For the green scintillator the MAD is $0.09 \pm 0.05$ and max difference 0.17. For the blue scintillator the MAD is $0.07 \pm 0.05$ and max difference -0.15. They clearly disagree unacceptably from the reference dose, and we attribute this to noise and optical artefacts. This can clearly be seen in Figure 4.22 where relative profiles are plotted at 15mm depth. The output factors are extracted from the central dose in Figure 4.22c where there is a high amount of noise for large field sizes.

![Output factors after Cherenkov subtraction at 15mm depth](image)

(a) Green scintillator

(b) Blue scintillator

Figure 4.21: Output factors after Cherenkov subtraction at 15mm depth
4.4 Results

(a) Scintillation+Cherenkov
(b) Cherenkov
(c) Cherenkov subtracted

Figure 4.22: Relative profiles for different field sizes in the green channel. Figure (c) are the profiles when the Cherenkov profiles (b) is subtracted from the scintillation+Cherenkov profiles (a).
4.5 Conclusions

We have proposed a detector system consisting of a scintillator sheet sandwiched between plastic blocks which is capable of rapid quality assurance of the CyberKnife. We were able to setup our detectors and take all measurements in a couple of hours, but to extract the dose we required multiple image processing steps. Our simulations of the setup show that we get a good agreement with the reference doses, but they also reveal that there are unwanted background signals such as optical scattering. If we can remove all the optical artefacts, then it would be a viable method for QA. However, due to smaller beam widths, we must use thinner scintillator slices to obtain approximately point dose distributions and to ensure that light comes from the central axis of the beam. This leads to a lower light output which means a higher noise to signal ratio. On top of this, the background Cherenkov light also becomes more significant. While we perform many image processing and correction steps, it is not enough to get clinically acceptable PDDs, profiles and output factors which is mainly due to the low light output and noise. However, in the next section, we investigate the use of machine learning for dose extraction from the scintillation images which prove to be better method than manually processing the images.

**Future work.** There are a few things we can do to improve our experimental design and image processing. The first thing to consider is using a higher light output scintillator which would give a lower noise to signal ratio. The machining of the plastic blocks isn’t perfect and the surfaces are not completely flat and smooth (there are scratches), to combat this we can use optical grease inbetween the blocks which fills empty air gaps between the scintillator and plastic blocks. Perhaps a metal shielding around the camera will help reduce the X-ray scattering noise. We also suspect that capturing more images with a lower exposure time would be better than capturing less images with higher exposure time. This is because the higher exposure images get more X-ray noise, and with more images, using pixelwise median (section 4.2.3) would be able to filter out more noise. We also acknowledge that there are further corrections to consider such as chromatic aberration, lens distortion and vignetting, but we suspect they will not improve our results as our dose differences are mainly obstructed by noise. Our observations have also revealed radiation damage in both scintillating and non-scintillating plastic materials. This damage manifests as a temporary green discoloration that persists for several hours before fading away. Notably, the discoloration is only visible when illuminated with light, effectively functioning as a green filter and assuming the shape of the deposited radiation dose (see Appendix A). Given the potential impact on our images, it is essential to thoroughly characterize and quantify this effect to ensure accurate future measurements.
Summary. The aim of this chapter was to improve the current dose verification methods for the CyberKnife which is currently done via point detectors such as the microDiamond. In particular, we were interested in the OF curves. We aimed to design a scintillator-based detector system that would improve data acquisition times, cost of equipment and reduce the amount of corrections needed. Our objective was to do this via scintillator sheet imaging and Cherenkov subtraction. Scintillator sheets have been used numerous times for dose verification, but the novelty lies in using it for the OFs curves which requires comparison and processing of images with different collimator sizes. My contributions to this chapter were the development of the image processing to dose extraction pipeline (alignment, denoising etc.), the Geant4 simulation of the experiment and assisting with the experimental setup and data acquisition.
5. CNN assisted scintillation dosimetry

5.1. Introduction

The photography of scintillating sheets provides a fast and cheap detector system for QA of radiotherapy beams. But in the small field regime, thinner scintillating sheets are required which leads to a lower light output and noisier data. Furthermore, there are image processing steps required to correct for visual artefacts such as background Cherenkov light (Yogo et al., 2017; Frelin et al., 2008). While the images contain many artefacts, they still contain a lot of information which can be used to improve the dose reconstruction. In this section, we investigate the use of convolutional neural networks (CNNs) to extract the features from the scintillation images and predict the relative dose.

Artificial neural networks (ANNs) in radiotherapy. In many parts of radiotherapy, ANNs are being used to reduce workload and in some cases improve the safety and accuracy of radiotherapy. ANNs are used in tumour detection, treatment planning, beam QA, treatment delivery QA, predictions of biological effects and predictions of outcomes (Shan et al., 2020). This ultimately means more patients can be treated safely in a shorter amount of time. There are a few areas in QA where machine learning is being researched to assist in the QA procedure, such as prediction of monitor units and prediction for gamma evaluation (Sun et al., 2018; Tomori et al., 2018; Nyflot et al., 2019; Grewal et al., 2020). But these methods do not have a large amount of data samples which is needed for deep neural networks to work well.

Manual approach to scintillation photography. Scintillation photography is useful for acquiring rapid, 2D or 3D dose using a camera and a large volume of water-equivalent scintillator (Hamel, 2021). The advantages over point detectors is that multi-dimensional dose may be captured in one or multiple images which is much faster than scanning a point detector (e.g. ionisation chamber) and cameras also have a high resolution (typically $\sim$0.1mm per pixel) which means volume averaging effects are minimal. In addition, they are water-equivalent which is advantageous over e.g. gas-filled ionisation chambers which need correction. However, due to a number of limitations, large volume scintillation dosimetry is still in the research stage and to the best of our knowledge, little to no detector setups are commercially in use (though point scintillator detectors are, e.g. Carrasco et al. (2015)). The scintillator and camera setup requires many image processing steps to correct for visual artefacts including Cherenkov correction (Frelin et al., 2008; Yogo et al., 2017), de-blurring (Petric et al., 2006; Ponisch et al., 2009), 3D reconstructions (Goulet et al.,
5.1 Introduction

2014), denoising and alignment. To obtain dose distributions from the processed images, it is then a simple matter of choosing a particular slice of the image. These methods have been proven to work well for large field sizes, but as we move into the small field regime, we need to take thinner slices of the beam. This leads to a lower scintillation light output, making noise and background signals more significant and introducing larger errors (see chapter 4). In our CNN approach, we combat these problems by automation of the image processing and making use of the whole image rather than a slice.

**Our CNN approach to scintillation photography.** In the classical approach, the dose reconstruction is done by measuring the dose at certain points of the image; this only uses information from those certain points and the rest of the image is often ignored. In fact, the rest of the image contains information or features which can also be utilized for dose prediction, but this would require a heavy amount of manual feature engineering. CNNs have been proven useful to automatically extract useful features from noisy images, mostly in object classification/detection but they are also useful for general quality assurance applications such as manufacturing (Ferguson et al., 2018), tumour contouring (Chen et al., 2020) and in radiotherapy (Nyflot et al., 2019; Tomori et al., 2018). We therefore utilize CNNs to predict dose from our scintillation images. In addition, we also predict physical known variables from the images which gives us confidence that our models are consistent with the real world. Our approach is outlined as follows:

- The main purpose of our CNN models is to extract the features from the scintillation images and predict the PDD and output factor. This is part of the QA procedure for the CyberKnife which is currently checked every 6 months.

- For CNNs to work well we need a large amount of data but scintillation photography provides a limited amount of data. We therefore simulate a dataset using a similar method to *domain randomisation* (Tobin et al., 2017) which creates an effective dataset which contains realistic and randomly parameterised visual artefacts allowing our models to become robust to these artefacts.

- As a first step, we train our models to predict the beam energy accurate within an order of magnitude. This is relevant since the CyberKnife operates at a constant energy of 6MV and a change in energy could imply that the magnetron needs changing which typically occurs every 2 years. Our images are currently not of high enough quality such that the models can be trained to predict slight variations in energy. Realistically, the typical changes in energy are less than an order of magnitude which are currently not distinguishable in our simulated images.
5.2 Method

• We predict the collimator size/beam width (defined as the full width half maximum (FWHM) at 15mm depth) which is a quick check that we have put on the correct collimator. We note that this method only outputs a predicted collimator size given an image which does not provide information about the shape of the beam. The full QA of the CyberKnife requires more information from the profile such as beam flatness, symmetry and penumbra.

• We predict the ratio of scintillation to total light (scintillation+Cherenkov) captured by the camera. This is useful since it would let us know if the amount of scintillation light is acceptable (e.g. we would ideally like the proportion of scintillation light to be much higher than the noise to signal ratio).

• Admittedly, neural networks have the disadvantage of being inexplicable due to their large number of parameters. To account for this, we provide a discussion on their feature maps in section 5.3.5.

5.2. Method

To summarize, we develop an end-to-end method which uses CNNs to predict output factors, PDDs, scintillation ratio, beam width and beam energy. The CNNs are trained and validated with a randomised simulated dataset. The trained CNN model is then used to make predictions on the measured images (from chapter 4). For reproducibility, we provide our training code at https://gitlab.com/jeremyocampo/scintinets. In this section we provide details of the randomized simulated data/augmentations, details of the models used and how they are trained.

5.2.1. Measured data

We use the same experimental results previously in section 4 for comparison with our CNN model predictions. For reference, we give a summary of the setup in section 4 here. The setup uses a similar approach to (Yogo et al., 2017) except we use plastic blocks instead of water. The setup is enclosed in a plastic case with the camera aligned with the top surface of the plastic blocks, see Figure 5.1. A 2mm green or blue scintillator sheet is sandwiched between 2 plastic blocks (200mmx200mmx50mm). We use a 6MV CyberKnife beam with collimator sizes 5-60mm which are photographed with exposures such that the images are not saturated (see section 4.2). For a higher scintillation to Cherenkov ratio, we only use one channel of the image to extract the dose, e.g. for the green scintillator we use the green channel. For dose distributions we use data measured by a microDiamond diode (Chalkley and Heyes, 2014), the PDDs and output factors are measured at 785mm SSD. We also note that the collimator sizes or beam widths that are used
5.2 Method

Figure 5.1: Experimental setup for extraction of measured data (from section 4), used for comparison with CNN prediction.

refers to the full-width-half-maximum of the profile at 15mm depth measured with 785mm SSD.

5.2.2. Simulated data

The measured data effectively contains 5 images for each field size, which is not enough to train a neural network. We solve this problem by simulating a larger dataset using a method known as domain randomisation (Tobin et al., 2017) where simulation parameters are randomized, for example the position of camera, beam energy etc. We do this with the intention of having the real world parameters being contained in the space of the simulated parameters. This allows the models to become robust to slight changes in these parameters. We are also able to randomize some parameters during training time such as the fraction of scintillation to scintillation+Cherenkov light captured, noise levels and photo orientation. Since these variables are continuous, we can effectively create an infinite number of images.

Simulation Details. We simulate our PDDs, output factors and images in Geant4 with the same setup as in section 4.2.5 and appendix B.1, where we simulate a separate scintillation and Cherenkov image and the corresponding dose distribution. In the simulation we randomly parameterize variables that would realistically vary, such as beam energy, camera position (not orientation), scintillator thickness, source to surface distance, beam direction, beam width, number of fired X-rays and beam direction (for more details see Appendix C). If our CNN models are trained on these images, then they can become robust to these random changes. These images are then randomly augmented using random image rotations, blurring, random salt and pepper noise, random black boxes and random total scintillation light as a fraction of the total light, this is denoted as \( r \) where

\[
I = rI_S + (1-r)I_C,
\]

where \( I_S, I_C \) denotes
5.2 Method

the scintillation and Cherenkov images. Example images are shown in Figure 5.2. For simplicity and simulation speed, we have chosen to simulate 240×240 grayscale images which have 1mm per pixel on the focus plane. The randomised parameters we choose depend on the problem we are trying to solve which we outline in the next sections. The sampling distributions of each dataset is given in Appendix C.

![Figure 5.2: Randomly simulated scintillation+Cherenkov images.](image)

5.2.3. Machine learning pipeline

As we have image/dose pairs, we take a supervised learning approach where we train our models to predict the dose or physical variable given an image. Since these values are continuous numbers, we are solving a regression problem with each model. The simulated dataset is split into a training and validation set. The training set is used to train the models. The validation set is unseen by the models and is used to check the performance of the trained model. Finally, we use the measured images and dose as a test dataset to see if the model works on real measurements, this process is illustrated in Figure 5.3.
5.2 Method

Figure 5.3: Flow chart of supervised learning approach.

One of the advantages of using a CNN is that there are a plethora of accessible architectures that are generalizable to different data-domains, such as ResNet (He et al., 2015b). This means that we don’t have to heavily tune the CNN architecture for it to work well on our dataset. We experiment with different models and data depending on what we’re trying to predict and provide an overview of the data, training and models in the next sections.

Designing a machine learning system is generally an iterative process (Huyen, 2022). The selection of our model, loss function and other hyperparameters are chosen through trial and error by training newly designed models and evaluating its performance iteratively until we are satisfied with the performance. It is generally better to start off with a simpler model with fewer parameters and then improve our model from there, e.g. by adding more parameters or using an architecture inspired by previous works. A simpler model means there are fewer assumptions in our hypothesis function which maps the data to the predictions. This ultimately leads to better generalizability to unseen data (see section 2.2.3).

5.2.4. Predicting PDDs

Data: The input to the model is one grayscale image and the output is the relative PDD for depths 0 to 199mm with resolution 1mm, this is represented as a length 200 vector. We have simulated a training and validation set of 1546 and 386 examples respectively. But deep neural networks commonly need tens of thousands to millions of images to perform well, here our dataset could be increased to an infinite amount of images through augmentation, e.g. by choosing an infinite number of scintillation ratios, $r$ (see section 5.2.2). Each image has random simulation parameters and
random augmentation, for details see Appendix C. Finally, our models are tested on measured images.

**Simple Model:** Our input is a $240 \times 240$ image and the output is a 200 length vector, therefore we have to reduce the dimensions throughout the model. As previously mentioned, it is best to start with a simple model as a lower number of assumptions leads to better generalizability. If we want to improve the performance, we could increase the complexity. To start off, we use a model composed of two downsampling convolutional layers with output channels (32, 64) followed by global average pooling and then a dense layer of output size 200, (for details see Appendix D.1). To choose the initial number of channels or features maps, we simply guess the number of features we expect the model to learn (it should learn beam width, SSD, background signals and maybe some other combination of features). If the model does not perform well, we can change the number of parameters and the architecture until we are satisfied with the performance. As a starting point, this model has 65k parameters.

**Improved Model:** To improve our model, we use an encoder-decoder style CNN (illustrated in Figure 5.4). Denoising autoencoders (Vincent et al., 2010) were developed to reconstruct noisy signals by compressing the information from the signal into a smaller representation (or a smaller vector), thereby noise information is lost. The output features from the encoder are then “decoded” via more layers that up-samples the features into a smoother signal. The encoder part of our model acts as a “denoiser”. It uses 8 convolutional layers and downsamples the image height/width by 2 every other layer and upsamples the number of channels by 2 every other layer. We have increased the number of channels for the first layer to 64. The decoder part reconstructs the PDD and is composed of 3 dense layers which have output dimensions 50, 100, 200 (see Appendix D.1 for more details). The output from the encoder is obtained through a global pooling layer which is crucial for having a fully translational equivariant model. This means that the outputs are invariant to translations in the input image, making the model more robust. For example, if we were to re-measure our images but with the camera slightly budged by 1cm, we should expect to get the same PDD because we have enforced that our neural network be translation-invariant. This is not always the case for non-translationally-invariant neural networks (e.g. fully dense) where a translation would result in a different PDD. The final improved model has 4 million parameters.
5.2 Method

Training: In reference dosimetry, the PDD is often characterized by the gamma ($\gamma$) metric (Low and Dempsey, 2003) which combines the difference and distance to agreement of dose (see section 3.1.3). Typically, a good gamma passing rate is over 90\% for a Gamma criterion 2\%/2\ mm (for example see Li et al. (2021); Dechambre et al. (2017)) which means 90\% of the predicted dose points are within the radius of 2\ mm and 2\% difference from the reference dose. We therefore define our custom loss function as mean absolute difference (MAD) + 10$\overline{\gamma}$ where the bar denotes a mean. Here we have empirically found the weighting of 10 to have better convergence. For gradient descent, we use the Adam (Kingma and Ba, 2015) optimizer with learning rate 0.001 for 50 epochs.

5.2.5. Predicting Output factors

Data: The output factor (OF) curve refers to the point dose at a given depth (for the CyberKnife it is 15mm) which increases as you increase the collimator size (see section 3.1.2). The relative OF of some collimator size is the point dose made by that collimator as a percentage of the point dose made by the 60mm collimator. Therefore, if we were to make a machine learning model to predict the relative OF, the inputs should be an image of some collimator size (the target image) and the image of the 60mm collimator (see Figure 5.5). This is because we want the model to predict how much dose is produced in the target image relative to the reference image (60mm).

We simulated a training and validation data set of 768 and 192 examples. Through the augmentation steps outlined section 5.2.2, this could essentially be increased to an infinite amount of images, e.g. by choosing an infinite number of scintillation ratios, $r$. We simulate and batch the images in groups of 12 collimator sizes (5mm-60mm) with the same simulation parameters in each group. The intention here is that it will help the model learn the differences in dose between those collimator sizes rather than differences in the physical simulation parameters. Each group has different simulation parameters, and each image has different augmentations. The images are normalized by the mean of the reference image. As most of the

![Figure 5.4: Model for predicting PDDs.](image)
5.2 Method

Target (e.g. 20mm) → Reference 60mm → Stack → ResNet → Output factor

![Flow chart for predicting Relative output factors.](image)

Values of the relative output factors are close to 1 (see section 3.1.2), we use an additional augmentation which gives it more variance. This is done by randomly scaling the output factor and target image by a random number sampled from $\sim \mathcal{N}(1, 0.01)$.

**Simple model:** The input to the model is two $240 \times 240$ grayscale images which is the target and reference (60mm) image, and these two images are stacked to give two channels. The output is the relative OF which is just a scalar. As previously mentioned, it is best to start with a simple model as a lower number of assumptions leads to better generalizability. Throughout the model we need to downsample our signal since the output is zero dimensional. Our initial model is composed of two downsampling convolutional layers followed by global average pooling followed by a dense layer with a scalar output (or a vector with length one), further details are given in Appendix D.2. Our initial model has 54k parameters.

**Improved model, 12 Layers:** To improve our model we may increase the number of channels or add more layers. Deepening neural networks generally improves the performance more than widening them (see section 2.3.4). For this reason, we decided to add more layers. Our improved model is composed of 10 convolutional layers and two dense layers (see Appendix D.2 for details). This model has 2 million parameters.

**Improved model, 24 Layer ResNet:** To further improve the model, we use a 24 Layer ResNet-style (He et al., 2015b) CNN which allows deeper CNNs to be trained and hence improving the performance (see section 2.3.4). Here the number of parameters is increased to 4 million. After applying all the convolutional layers, we use global average pooling followed by 2 fully connected layers. The global average pooling step is crucial for having a fully translational equivariant model. This means that the outputs are invariant to translations in the input image, making the model more robust.
5.2 Method

Training: We train the model using a MAD loss. For gradient descent, we use the Adam (Kingma and Ba, 2015) optimizer with a learning rate 0.001 which is reduced by a factor of 10 every 40 epochs. We train for 120 epochs with batch size $12 \times 6$, i.e. 6 groups of images with the 12 unique collimator sizes in each group.

5.2.6. Predicting Variables

Data: The input to the model is 1 grayscale image and the output is the variable we’re trying to predict which is represented as a scalar. Here we use different models to separately predict energy, scintillation ratio and beam widths.

The dataset for predicting energies have a training/validation split of 1546:387. Images where the beam has energies with the same magnitude are harder to distinguish. We therefore use a dataset with a larger range of energies which gives us a proof of concept that our model can distinguish between different beam energies. Half of the data has 6MV energy and the other half has energies randomly sampled from $6 \times 10^{s}$ MV where $s$ is uniformly sampled from [-1.5, 1.5]. This allows for an “energy scale” prediction which is more informative as small variations in energy are harder to distinguish. Example images are shown in Figure 5.6.

We use a different dataset for predicting scintillation ratios and beam widths which have more realistic energies, sampled from $s \times 6$ MV where $s$ is a scaling factor sampled from $\mathcal{N}(1,0.25)^{∗}$, see appendix C for more details. The training/validation split is 3076:770. Scintillation ratio is randomly selected from [0,1] during training and validation. We want the model to distinguish between different scintillation ratios, regardless of parameters/augmentation used. To enforce this, we use a trick where for each training image, we use the same identical image 3 times in the same batch, but each image has a randomized scintillation ratio. Example images are shown in Figure 5.7.

∗Note that this decision was made in our first trial of our machine learning pipeline. We have chosen this since the measured beams are all 6MV so it would be best for our model to learn what a 6MV beam looks like. We chose to vary this slightly since we want our model to be robust to slight changes in the measured data. We also could try for example a uniform sampling instead of a normal sampling, which means we would have to simulate a new dataset and train the whole model again. But there are many decisions that can be varied, and it would take an unpractical amount of time to try them all. Since the simulations generally take a lot longer (e.g. 100 images per day) then the model to train, it is more practical to perform trial and error on the models and training rather than the simulated data.
5.2 Method

Model: The energy, scintillation ratio and beam width are independent. We therefore don’t expect an increase in performance if we use a model that predicts all variables at the same time. For predicting energies, we use a simple model with 5 convolution layers, global average pooling and 2 dense layers (a fully translational equivariant model). For predicting scintillation ratio or beam width, we use a 24 layer ResNet-style (He et al., 2015b) CNN followed by global average pooling and 2 dense layers (see Appendix D.3 for details).

Training: For predicting energies, we train the model using a MAD loss, an Adam (Kingma and Ba, 2015) optimizer with a learning rate 0.001 using batch size 64 and train for 50 epochs. For predicting scintillation ratio or beam width, we train the model using a MAD loss, an Adam (Kingma and Ba, 2015) optimizer with a learning rate 0.001 using batch size 60 and train for 100 epochs.

Figure 5.6: Example simulated training images for predicting energies.

Figure 5.7: Example simulated training images for predicting scintillation ratios. Note that we have randomly varied the beam intensities which means images will have different noise levels as shown in these images.
5.3. Results and discussion

Here we present our results for predicting PDD, output factors and variables using our CNNs described above. In each subsection, we present the results for evaluating the models on the simulated validation datasets after they have been trained. Finally, we present the results after choosing the best model and applying them on our measured images from chapter 4.

5.3.1. Predicting PDD

![Histograms of PDD differences](image)

(a) Simple model  
(b) Improved model

Figure 5.8: Difference histogram of PDDs on validation set. Here the whole validation set is used and we have counted the difference for each depth on the histogram. $\mu = \text{fitted mean}$, $\sigma = \text{fitted standard deviation}$.

**Training results:** We evaluate our models on the simulated validation data by computing the difference between predicted and true dose at each depth. All the computed differences are counted on a histogram in Figure 5.8. Our simple model achieves a fitted standard error $\sigma$ of 1.7% and mean difference $\mu$ of -1.1% in Figure 5.8a. As we would like at least 90% of our dose points to be less than 2% difference, we improved our model by adding more layers which lead to an improved $\sigma$ of 0.7% and $\mu$ of 0.2%.

**Measured results:** We test our improved model on measured images and compare with the manual method using Cherenkov Subtraction (the same results from section 4). We use microDiamond measurements (Chalkley and Heyes, 2014) as the “True” dose for reference. The results are outlined in Table 6 which shows that the CNN method outperforms the Manual method for all collimator sizes, we also show qualitative PDDs in Figure 5.9. Our CNN models achieve a mean Gamma(3mm/3%) accuracy of $94.8 \pm 13.0\%$ and mean Gamma(2mm/2%) accuracy $92.0 \pm 17.5\%$ over all collimator sizes. This is compared with the manual method achieving a mean Gamma(3mm/3%) accuracy of $40.9 \pm 9.57\%$ and mean Gamma(2mm/2%)
accuracy $34.1 \pm 8.24\%$ over all collimator sizes. We also notice that the performance at 5mm size is not as good. We expect that this is due to a lower light output which means the noise and background is more significant leading to a more distorted signal and a worse prediction. Another reason for the drop in accuracy at 5mm and 60mm is that the training set only contains field sizes between 5-60mm meaning it would be harder for the model to extrapolate near and outside the edges of this range.

Table 6: Performance comparison of CNN vs. manual method for reconstructing the PDD. The reference PDD and predicted PDD are normalised by their maximums (so that the maximum is 1). The difference is then between the normalised PDDs. Here we give the MAD over all points on the PDD curve. We also give the Gamma passing rate as a percentage for criterions 3mm/3% and 2mm/2%.

<table>
<thead>
<tr>
<th>Collimator size</th>
<th>MAD</th>
<th>Gamma (3mm/3%)</th>
<th>Gamma (2mm/2%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5mm</td>
<td>0.04 ± 0.03</td>
<td>0.02 ± 0.02</td>
<td>28.9 %</td>
</tr>
<tr>
<td>7.5mm</td>
<td>0.03 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>42.2 %</td>
</tr>
<tr>
<td>10mm</td>
<td>0.03 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>40.4 %</td>
</tr>
<tr>
<td>12.5mm</td>
<td>0.03 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>50.6 %</td>
</tr>
<tr>
<td>15mm</td>
<td>0.02 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>50.6 %</td>
</tr>
<tr>
<td>20mm</td>
<td>0.02 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>53.0 %</td>
</tr>
<tr>
<td>25mm</td>
<td>0.03 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>41.0 %</td>
</tr>
<tr>
<td>30mm</td>
<td>0.03 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>42.2 %</td>
</tr>
<tr>
<td>35mm</td>
<td>0.03 ± 0.02</td>
<td>0.01 ± 0.01</td>
<td>44.6 %</td>
</tr>
<tr>
<td>40mm</td>
<td>0.03 ± 0.03</td>
<td>0.0 ± 0.01</td>
<td>44.0 %</td>
</tr>
<tr>
<td>50mm</td>
<td>0.03 ± 0.03</td>
<td>0.0 ± 0.01</td>
<td>36.8 %</td>
</tr>
<tr>
<td>60mm</td>
<td>0.06 ± 0.04</td>
<td>0.01 ± 0.01</td>
<td>16.9 %</td>
</tr>
</tbody>
</table>
5.3 Results and discussion

Figure 5.9: Comparison of PDD predictions for a CNN and Manual method using a Gamma criterion of 3mm/3%.
5.3 Results and discussion

5.3.2. Predicting Output Factors

Training results: After training, we evaluate the performance of the models on the simulated validation dataset. We compute the difference between predicted and true output factors for the validation dataset and show histograms for the best model (ResNet) and various field sizes in Figure 5.10. We also show the fitted mean ($\mu$) and standard deviation ($\sigma$) for all the models that we tried in Table 7 for the different models and collimator size.

We started off with a simple 3 Layer model and to improve the performance we increased the layers to 12 and then to a 24 layer ResNet. Table 7 shows that the ResNet outperforms the other models which is due to its higher number of parameters. For the ResNet, the standard error $\sigma$ decreases with collimator size from 12.7% to 0.5% with 5mm and 60mm collimator sizes. This is expected as our simulated output factors contains less variation as you increase the field size.

![Histograms showing the difference between true and predicted OFs using the 24 layer ResNet model for different collimator sizes on the simulated validation data.](image)

Figure 5.10: Histograms showing the difference between true and predicted OFs using the 24 layer ResNet model for different collimator sizes on the simulated validation data.
5.3 Results and discussion

Table 7: Fitted mean ($\mu$) and standard deviation ($\sigma$) to histogram of differences when comparing true and predicted output factors. The best $\mu$ for each collimator size is highlighted as bold. Note that the predicted and true relative OFs are normalised to be a percentage of the highest field size at 60mm. The difference is then taken between the normalised predicted/true OF curves.

<table>
<thead>
<tr>
<th>Model</th>
<th>Fitted $\mu, \sigma$</th>
<th>Collimator size (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu$ (%)</td>
<td>5 7.5 10 12.5 15 20 25 30 35 40 50 60</td>
</tr>
<tr>
<td>3 Layers</td>
<td></td>
<td>-26.7 -13.5 -5.5 -1.5 2.2 3.3 2.8 2.8 2.7 1.6 -0.4 -2.3</td>
</tr>
<tr>
<td></td>
<td>$\sigma$ (%)</td>
<td>8.9 6.8 5.5 4.0 2.6 1.6 1.0 1.2 1.1 1.7 1.6 0.4</td>
</tr>
<tr>
<td>12 Layers</td>
<td>$\mu$ (%)</td>
<td>-16.0 -3.0 5.5 9.0 10.4 11.5 10.8 10.8 10.3 8.6 4.1 1.1</td>
</tr>
<tr>
<td></td>
<td>$\sigma$ (%)</td>
<td>10.3 8.6 5.6 5.1 4.8 4.1 4.3 4.1 4.6 4.7 6.0 5.1</td>
</tr>
<tr>
<td>24 Layer</td>
<td>$\mu$ (%)</td>
<td>-3.6 -3.4 0.2 2.0 0.3 0.6 0.1 0.2 0.1 -0.2 -0.5 -0.2</td>
</tr>
<tr>
<td>ResNet</td>
<td>$\sigma$ (%)</td>
<td>12.7 10.2 6.2 4.0 2.7 1.8 1.0 0.8 0.7 0.6 0.5 0.5</td>
</tr>
</tbody>
</table>

**Measured results:** To evaluate the measured dataset, we choose the best model which is the ResNet. Our results in Figure 5.11 show that our CNNs outperform manual image processing techniques. Our model is compared with a Manual method using Cherenkov Subtraction (see section 4). We use microDiamond measurements (Chalkley and Heyes, 2014) as the “True” dose for reference. Typical acceptable values for measuring output factors for the CyberKnife are within differences of 1% or 2% (see e.g. Al Kafi et al. (2023); Manavalan et al. (2021)) which we are close to achieving. For the green scintillator our CNN model achieves a MAD 1.1 ± 0.7% and maximum difference 2.5% (at 5mm), which is a significant improvement over the manual method with MAD 9.2 ± 4.5% and maximum difference 16.6%. For the blue scintillator our CNN model achieves a MAD 3.4 ± 3.6% and maximum difference 12.4%, which is a significant improvement over the manual method with MAD 7.2 ± 4.6% and maximum difference 15.2%. We suspect that the decrease in predicted dose in the blue scintillator is due to it having a lower light output than the green. A lower dose usually translates to a lower and noisier signal, and since neural networks are “interpolators” (see section 2.1.2) our model might have interpreted the noisier signal as a lower dose.
5.3 Results and discussion

(a) Green scintillator

(b) Blue scintillator

Figure 5.11: Predicted output factors using CNNs, error bars denote standard deviation over 5 different images
5.3 Results and discussion

5.3.3. Predicting Variables

**Training results:** After training, the regression models are evaluated on the validation dataset by their difference histograms and fitting with a Gaussian to quantify their performance. The mean ($\mu$) and standard deviations ($\sigma$) are fitted and shown in Figure 5.12. The percentage differences* between true and predicted energies are plotted in Figure 5.12a. The differences between true and predicted scintillation ratios are shown in Figure 5.12b. Because the scintillation ratio $\in [0, 1]$, the maximum possible difference is 1. The differences between true and predicted beam widths are shown in Figure 5.12c. Here it is clear that the histogram for beam widths does not represent a true Gaussian, so instead we compute and give a MAD of $1.65 \pm 1.53$mm. There is a positive skew which means that the predicted beam widths are on average lower than the true beam width, we suspect this is due to a data imbalance as there are more beam widths $<15$mm (see appendix C). This means that there will be more gradient steps that change the weights such that they predict a lower beam width.

![Difference histograms for energy, scintillation ratio, and beam width](image)

Figure 5.12: Difference histograms with a Gaussian fit for mean and standard deviation ($\mu, \sigma$).

*We use percentage difference here as the energies of the dataset has a uniformly sampled energy scale, i.e. the energies are $E = 6e^s$MV where $s$ is sampled uniformly from $[-1.5, 1.5]$. 

100
5.3 Results and discussion

Measured results: Results for predicting energies for each collimator size are shown in Figure 5.13a. The mean predicted energies over all collimator sizes are $5.85 \pm 1.43$ MV compared to the true energy 6MV. For field size 5mm, the CNN predicted an energy of 1.3MV. We suspect that this is a bias of the model to predict lower energies for images with higher noise as there is lower light output for lower energies in the training data. Results for predictions of scintillation ratio over all collimator sizes are shown in Figure 5.13b. The MAD between the measured and predicted scintillation ratio is $0.20 \pm 0.14$. We suspect the bigger difference for lower field sizes is due to the image looking more like a “fully Cherenkov image” as the beam size approaches the scintillation sheet width*. This makes images at lower collimator size harder to distinguish scintillation ratios. Results for predicting beam widths are shown in Figure 5.13c which is averaged over all 5 images. The MAD is $0.92 \pm 0.50$mm and mean absolute percentage difference $5.3 \pm 3.6\%$.

![Variable predictions](Image)

*This point is subtle. To see this consider two extreme images, a beam fired into a scintillator block and the same beam fired into a plastic block, the images should look similar.
5.3 Results and discussion

5.3.4. Robustness

We have extracted the PDDs and OFs from photographed scintillation images and they are either within or close to clinically accepted differences compared to the reference dose. We have done this using end-to-end CNNs meaning the input to the CNN is the scintillation image and the output is the required dose (PDD or OF), thereby automating the image processing. However, the test dataset has all been measured under similar conditions, e.g. all materials have the same scintillation light yield and all measurement were done with the with the same camera. To give us better confidence that our models are robust, we should test if they work under different conditions.

Here we give a couple of robustness tests to see how the model performs as a function of noise and scintillation ratio. We also acknowledge that the inner workings of neural networks is usually not well explained, here we give a few methods to visualize what has been extracted from the images using convolutional and dense feature maps.

Noise. To test robustness with respect to noise, we make predictions on the measured images with added noise and evaluate our CNNs performance for predicting PDDs. We add salt and pepper noise on a random selection of 50% of pixels with random values in \([0, sM]\) where \(M\) is the maximum of the image and \(s\) is the noise level which we vary. For each measured image (each of the 12 collimator sizes) we add the noise image and predict a PDD for 10 random realizations, making \(12 \times 10\) total images per noise level. The predicted PDD is compared to the measured microDiamond PDD using Gamma (3mm/3%) accuracy and MAD shown in Figure 5.14a-5.14b as a function of noise level, where the standard deviation here refers to all the differences for each image. The model’s performance is robust up to around 0.2 noise levels as Figure 5.14a shows that the models accuracy goes down from 94.8% to 87% at 0.2 noise level. Figure 5.14b shows a roughly constant MAD of 1% up to 0.2 noise level.

Scintillation ratio. We test the performance of our CNN models with varying scintillation ratios (scintillation/(scintillation+Cherenkov)). We cannot physically vary the amount of scintillation or Cherenkov light emitted physically and so we use the simulated dataset where we add separate scintillation and Cherenkov images together. We fit the mean \(\mu\) and standard deviation \(\sigma\) of our difference histograms as before (see Figure 5.8) but we vary the scintillation ratio as shown in Figure 5.15. This shows that the performance is robust over all scintillation ratios when predicting PDDs and output factors. Note that when the ratio is zero, we still get good performance which means the CNNs can predict dose using only Cherenkov light.
5.3 Results and discussion

(a) Gamma accuracy

(b) MAD

Figure 5.14: Performance as a function of noise levels.

(a) PDD

(b) Output factor

Figure 5.15: Performance as a function of the scintillation ratio.

5.3.5. Explainability

Due to the large number of parameters of neural networks, they are well known for their lack of interpretability which is a disadvantage when applying them in any field of science. A common way to understand a CNN is by extracting the outputs of each layer and back-projecting them onto the original input image (Zeiler and Fergus, 2013). Here we use a simpler method and plot the outputs of each convolutional layer which is sufficient to see what features are emphasized after each convolution. We also visualize the features of the high-dimensional dense layers by using a commonly used dimensional reduction technique called t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten and Hinton, 2008). Note that these methods only provide qualitative explanations which is a disadvantage in using CNNs.
Convolution feature maps. Convolutional layers work by applying different kernels onto the input and then combining the outputs into channels (feature maps). We can look at the feature maps that the CNN extracts at different layers by plotting the outputs of each convolutional layer shown in Figure 5.16 where we have chosen the feature maps at random and normalized by the max pixel value. For the input image we use the measured image at 25mm collimator size. We see that the CNN extracts and separates features such as noise, background, the beam and the top internal reflection. Note that when using the manual method, these features are what is used in the image processing, however it is evident from the feature maps that the CNN uses the whole beam instead of taking a slice to predict the PDD. Note that the only explanation this provides is that these feature maps are what is used to predict the outputs which is only a qualitative explanation.

Figure 5.16: CNN feature maps at different layers given a measured input image. Due to the downsampling layers in the CNN, the resolution decreases with the layer number.
**Dense feature maps.** In our model for predicting PDDs (Figure 5.4) there is a CNN encoder which extracts the features and transforms them into a vector. This vector of features is then used to predict the PDD by passing them through a dense neural network. We would like to visualize the outputs of the dense layers as this can give us insights into what is being extracted and used to predict the PDD. The output dimensions of the dense layers are 50, 100 and 200 which means we need to reduce the dimensions in order to visualize them. Here we use the t-SNE method (van der Maaten and Hinton, 2008) which maps our dense features onto a lower dimension, while optimally preserving the distances between data points.

We choose the layer with output size 50 and map this feature vector for each image to 2-D in Figure 5.17, where each point represents the feature vector of an image. We note that this is a qualitative explanation of what the neural network has learned as we can see how separated each image is in the latent feature space. We expect the PDD to be mostly dependent on the collimator size and source-to-surface distance (Figure 5.17a, 5.17b), therefore the feature maps should be separated according to these variables because these features are used to predict the PDD. On the other hand, the PDD is physically independent from the scintillation ratio $r$, hence we don’t expect the dense features to have any correlation with $r$ as shown in Figure 5.17c. Finally, we also plot the features of the measured and simulated images in Figure 5.17d which shows that the features of the measured images lie in the feature space of the simulated images.
5.3 Results and discussion

(a) Beam widths

(b) SSD

(c) Scintillation ratio $r$

(d) Simulated vs Measured images

Figure 5.17: Each point in these plots represents a vector of features of an image (extracted from the neural network) in our simulated and measured datasets. But this vector is high dimensional, therefore we project them onto 2-D using t-SNE so that we are able to visualise them. We colour code the data points according to different variables to see if the features are separated according to these variables. The axes here have arbitrary units since they are just features learned by a neural network.
5.4 Conclusions

In this chapter, we proposed novel methods for extracting PDDs and output factors from scintillation and Cherenkov images using end-to-end CNNs and randomised data. The CNN method is advantageous over manual image processing as it can provide better dose differences and accuracy while being robust to noise and visual artefacts, for example the average PDD accuracy using CNNs is 94.8% compared to 40.9% for the manual method. When predicting output factors, our CNNs achieves a MAD and maximum difference of 1.1% and 2.5% which is almost clinically acceptable. This is in comparison to 9.2% and 16.6% using the manual method. We have also predicted the beam widths and energies well using CNNs. In practice this allows us to automatically check that we have put on the correct collimator and that the energies are on the correct energy scale. While we aren’t able to predict small changes in energy, this work provides a proof of concept that we can predict large changes in energy (e.g. 10× smaller or larger) and with improvements to the data (e.g. higher resolution and less noise) we expect the performance to improve. However, CNNs are disadvantaged when it comes to explainability, i.e. what is the neural network actually learning and why. We have given feature maps but this only gives a vague explanation. For the classical method we know exactly what is being done. Another disadvantage of this method is that it is difficult to reproduce for other radiotherapy beams and setups as we have to simulate a new dataset each time.

Future work. There are many ways in which this work could be extended, for example applying CNNs to different experimental setups, e.g. for proton beam. Higher resolution data with less noise would be beneficial for sub-millimetre dosimetry and prediction of smaller energy changes in the beam since this is harder to distinguish for low quality images. Neural networks improve in performance as we increase the number of data examples and so we expect the performance to increase if we simulate more data. Our approach relied on creating a different CNN model for each task, but in future the training and inference could be more efficient if we trained a single model that can output PDD, OF, beam width, energy and scintillation ratio. Another application of CNNs could be for denoising our measured images, from which we can directly extract PDD, profiles and OF. This would be advantageous as we can directly extract the dose distribution of the beam from the denoised image rather than going through the process of feature extraction through to dose prediction which contains more complex operations. While we effectively have an infinite number of images from random augmentation, we do not cover the whole space of possible realistic images. As we only have thousands of images, we should perform K-Fold cross-validation to assure that our models work well with
any randomly sampled training/test datasets.

**Summary.** The aim of this chapter was to improve the accuracy of dose extraction from scintillation images which extended from chapter 4 where manual image processing methods produced inaccurate results due to optical artefacts. Our objective was to design a machine learning pipeline to extract the dose with improved accuracy and reduced error relative to the reference dose. The method comprised of simulating the training dataset, training CNN models and predicting on the measured scintillation photographs. While CNNs have been used in various areas in radiotherapy, the novelty of our method is using CNNs for dose extraction in scintillation photography. My contributions to this chapter comprised of the simulation of the training dataset and the development of the machine learning pipeline from scratch.
6. CNNs for general applications

This chapter is based on the paper: Ocampo et al. (2023).

6.1. Introduction

Most CNN architectures are defined natively on the plane, achieving high predictive performance when applied to planar images. This is in part due to their translational equivariance property which is a powerful inductive bias. When considering a different manifold such as images projected on the sphere, planar CNN aren’t as useful as they are not rotationally equivariant. Existing deep learning techniques for spherical data may solve this problem. However, they cannot scale to high-resolution while also exhibiting equivariance. In this chapter we present a novel framework to design CNNs for a general geometry while simultaneously having group equivariance and scalability. While our method can be used for a general manifold, we focus our application on the sphere which is the prototypical example of group convolutions. Spherical data is prevalent across many fields, from the relic radiation of the Big Bang in cosmology, to 360° imagery in virtual reality and computer vision. High-resolution data on the sphere is increasingly common in these and many other fields. Furthermore, many tasks involve dense predictions (e.g. semantic segmentation, depth estimation), necessitating pixel-wise outputs from deep learning models, exacerbating computational challenges.

Continuous spherical CNNs approaches. Bronstein et al. (2021) present the categorization of geometric deep learning approaches. Deep learning on the sphere falls into the group category, since the sphere is a homogeneous space with global symmetries on which the group of 3D rotations SO(3) acts. A number of group-based spherical CNNs have been developed (Cohen et al., 2018; Kondor et al., 2018; Esteves et al., 2018; 2020; Cobb et al., 2021; McEwen et al., 2022; Mitchel et al., 2022), where Fourier representations of spherical signals (i.e. spherical harmonic representations) are used which provides a continuous representation. While such approaches live natively on the sphere and capture rotational equivariance, they are highly computationally costly due to the need to compute spherical harmonic transforms (while fast spherical harmonic transforms exist they remain computationally demanding). McEwen et al. (2022) develop scattering networks on the sphere to alleviate computational considerations. While such an approach helps to scale to high-resolution input data, some high-resolution information is inevitably lost and architectures providing dense predictions are not supported.
Discrete spherical CNNs approaches. Other approaches to spherical CNNs generally fall under the grid, graph or geodesic geometric deep learning categories yielding discrete approaches (Boomsma and Frellsen, 2017; Jiang et al., 2019; Zhang et al., 2019; Perraudin et al., 2019; Cohen et al., 2019). These approaches offer more favorable computational performance than the group-based frameworks but at the cost of equivariance. Since a completely regular point distribution on the sphere does in general not exist, these discrete approaches lose the connection to the underlying continuous symmetries of the sphere and thus cannot fully capture rotational equivariance (although in some cases limited rotational equivariance may be achieved; e.g. Cohen et al. 2019).

Our approach. Previous spherical CNN constructions can be categorised into two broad approaches: continuous approaches that capture rotational equivariance but are computationally demanding; and discrete approaches that do not fully capture rotationally equivariance but offer improved computational scaling. In this article we develop a novel hybrid discrete-continuous (DISCO) approach that is computationally scalable to high-resolution, while also exhibiting excellent equivariance properties (see Figure 6.1). We define a DISCO group convolution, which we then specialize to the sphere. A transposed DISCO convolution can be considered to support dense-prediction tasks; hence, both high-resolution inputs and outputs are supported by our framework. DISCO convolutions afford a computationally scalable implementation through sparse tensor representations. We build DISCO spherical CNNs that achieve state-of-the-art performance on a number of high-resolution dense prediction tasks, including semantic segmentation and depth estimation.

![Figure 6.1: Spherical CNN categorization. The blue spot represents a localised kernel when rotated to two different positions.](image-url)
6.2. Background

6.2.1. Continuous group convolution

**Definition.** To generalize CNNs to group geometric deep learning settings the group convolution can be considered (see, e.g., Cohen and Welling, 2016; Esteves, 2020) where the convolution (sometimes called correlation) of two functions $f, \psi : G \to \mathbb{R}$ on the group $G$ is given by

$$(f \star \psi)(g) = \int_{u \in G} f(u)\psi(g^{-1}u)d\mu(u),$$  \hspace{1cm} (25)$$

where $g, u \in G$ and $d\mu(u)$ is the invariant Haar measure. For example, in the planar setting, $G$ represents the translation operators and $d\mu(u)$ represents a constant square area element. The group convolution exhibits equivariance to group actions, that is $(Qf \star \psi)(g) = (Q(f \star \psi))(g)$, where $Q$ denotes the group action on functions corresponding to $q \in G$, i.e. $(Qf)(g) = f(q^{-1}g)$.

**Fourier representation and sampling theory.** There is a generalized convolution theorem for the group convolution. This is written in Fourier space by the product $\hat{(f \star \psi)}(k) = \hat{f}(k)\hat{\psi}^*(k)$ (see, e.g., Esteves, 2020), where $k$ is the Fourier conjugate variable of $u$, $\hat{\cdot}$ denotes the Fourier transform, and $\cdot^*$ denotes complex conjugation. For compact manifolds (groups), the Fourier space is discrete (the standard example being the Fourier series of functions defined on the circle $S^1$, i.e. periodic functions). Furthermore, for bandlimited signals on compact homogeneous manifolds, exact quadrature formulae exist (Pesenson and Geller, 2011), from which the existence of a sampling theorem follows where all information content of a continuous bandlimited signal can be captured in a finite number of spatial samples.

**Computation.** These results provide a strategy to compute the continuous group convolution exactly for bandlimited functions defined on compact homogeneous manifolds: (i) compute the finite set of Fourier coefficients of the signal and filter leveraging the sampling theorem; (ii) compute the group convolution as a product in Fourier space; (iii) compute the convolved signal from its Fourier coefficients. Although the computation is performed in a discrete manner through the finite Fourier representation, the underlying continuous group convolution is computed exactly, without any discretization error. Consequently, perfect group equivariance is achieved. Note, however, that this approach does require computation of the Fourier transform on the manifold, which may be costly.
6.2 Background

6.2.2. Continuous spherical convolutions

The strategy outlined above to compute the continuous group convolution exactly for bandlimited signals on compact homogeneous manifolds is the generalization of precisely the approach taken for group-based spherical CNNs (Cohen et al., 2018; Kondor et al., 2018; Esteves et al., 2018; 2020; Cobb et al., 2021; McEwen et al., 2022), where Fourier (i.e. spherical harmonic) representations of signals are computed to provide access to the underlying continuous structure and symmetries of the sphere. The corresponding spherical convolutions exhibit perfect rotational equivariance, with any numerical errors arising simply from finite precision arithmetic (see, e.g., Cobb et al., 2021). We concisely review the details of this approach to constructing spherical CNNs.

Definition. The spherical convolution of two functions \( f, \psi : S^2 \rightarrow \mathbb{R} \) is given by

\[
(f \star \psi)(R) = \int_{S^2} f(\omega)\psi(R^{-1}\omega)d\mu(\omega),
\]

where \( \omega = (\theta, \phi) \) denote spherical coordinates with colatitude \( \theta \in [0, \pi] \) and longitude \( \phi \in [0, 2\pi) \), and \( d\mu(\omega) = \sin \theta d\theta d\phi \) is the Haar measure. We consider rotations \( R \in \text{SO}(3) \), where \( \text{SO}(n) \) is the special orthogonal group of rotations in \( n \)-dimensions. The sphere \( S^2 \) is isomorphic to the quotient space \( \text{SO}(3)/\text{SO}(2) \); it does not exhibit a group structure but is a homogeneous manifold.

Fourier representation and sampling theory. Since both \( S^2 \) and \( \text{SO}(3) \) are compact, their Fourier spaces are discrete and sampling theorems can be leveraged to compute Fourier representations exactly for sampled bandlimited signals (Driscoll and Healy, 1994; Kostelec and Rockmore, 2008; McEwen and Wiaux, 2011; McEwen et al., 2015). The spherical convolution can then be expressed as a product in Fourier space (see, e.g., McEwen et al., 2013; Cohen et al., 2018; Cobb et al., 2021).

Computation. For signals on the sphere bandlimited at \( L \), spherical convolutions can be computed exactly as a product in Fourier space. Although the computation is performed in a discrete manner through a Fourier representation, by appealing to underlying sampling theorems the continuous convolution is computed exactly. Consequently, perfect rotational equivariance is achieved. However, the harmonic transforms required are computationally demanding (even with fast algorithms; e.g. McEwen and Wiaux 2011; McEwen et al. 2015) and present the major computational

\[
f(\omega) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} f_{m}^{\ell} Y_{m}^{\ell}(\omega)
\]

where \( Y_{m}^{\ell} \) are the spherical harmonics and \( f_{m}^{\ell} \) are the coefficients which are zero for \( \ell > L \).
bottleneck in scaling equivariant spherical CNNs to high-resolution.

6.2.3. Discrete spherical convolutions

Alternative approaches to constructing spherical CNNs typically consider a discrete approximation of the convolution of Equation 26 (Boomsma and Frellsen, 2017; Jiang et al., 2019; Zhang et al., 2019; Cohen et al., 2019). It is well-known that a completely regular point distribution on the sphere does in general not exist. Consequently, while a variety of spherical discretization schemes exists, it is not possible to discretize the sphere in a manner that is invariant to rotations (Cobb et al., 2021). On the other hand, since discrete approaches avoid the need for Fourier transforms on $S^2$ and $SO(3)$, they are generally more computationally efficient than the continuous approaches discussed above that do exhibit rotational equivariance.

6.3. Discrete-Continuous (DISCO) convolutions

To simultaneously achieve rotational equivariance and computational scalability, we require a convolution that avoids a complete discretization and avoids Fourier transforms on manifolds. We present a novel method using a DISCO group convolution that achieves precisely this aim. We then specialize to the spherical setting, before discussing equivariance, filter parameterization and the corresponding transposed convolution for dense predictions.

6.3.1. DISCO group convolution

The DISCO group convolution follows by a careful hybrid representation of the group convolution of Equation 25. Some components of the representation are left continuous, to facilitate accurate rotational equivariance, while other components are discretized, to yield scalable computation.

**Definition.** We carefully approximate the group convolution by the DISCO representation

\[
(f \star \psi)(g) = \int_{u \in G} f(u) \psi(g^{-1}u) \hat{d}\mu(u)
\]

\[\approx \sum_i f[u_i] \psi(g^{-1}u_i) q(u_i),\]

where square brackets and index subscripts denote discretized quantities and round brackets denote continuous quantities. Clearly the signal of interest $f$, must be discretized at sample positions $u_i$, where $i$ denotes the sample index. Critically, however, the filter $\psi$ and the group action remain continuous. This allows the filter $\psi$ to be transformed continuously by any $g$, keeping a coherent representation
6.3 Discrete-Continuous (DISCO) convolutions

that avoids any discretization errors and, consequently, affords group equivariance, unlike a fully discrete method. The integral with respect to \( u \) is also discretized. For bandlimited signals on compact homogeneous manifolds, the existence of a sampling theorem ensures that the integral can be approximated accurately using quadrature weights \( g(u_i) \) (Pesenson and Geller, 2011).

**Approximation accuracy and equivariance.** The DISCO approximation of Equation 28 is highly accurate for bandlimited signals (note that real-world signals can be well approximated by bandlimited signals for a sufficient bandlimit). By appealing to a sampling theorem, all information content of the signal can be captured in the finite set of samples \( \{f[u_i]\} \). The filter is represented continuously so does not introduce any error. The only source of approximation error is thus the quadrature used to evaluate the integral. For a sufficiently dense sampling, one can appeal to the sampling theorem and corresponding quadrature to evaluate this exactly. Therefore, it is possible in principle to compute the DISCO group convolution exactly, without any approximation error. Since the approximation is highly accurate, which can be made exact for a sufficiently dense sampling, and group actions are treated continuously, the DISCO group convolution exhibits excellent equivariance properties.

**Computational considerations.** The DISCO convolution also affords a scalable implementation. Firstly, it avoids any need for costly Fourier transforms on manifolds and groups. Secondly, as in the Euclidean setting, filters with local spatial support are considered to reduce both memory and computational requirements. The naive computational scaling of the DISCO group convolution for a general kernel is \( O(N^2) \) *, where for simplicity \( N \) represents both the number of input and output samples. However, for localized filters with \( K \) non-zero samples, where \( K \ll N \), the scaling reduces to \( O(KN) = O(N) \), just as for Euclidean CNNs.

---

* e.g. For every output \( g_j \), one has to compute each term in the sum over \( i \) where \( i, j \in [1, N] \)
6.3.2. DISCO spherical convolution

**Definition.** We now specialize to the spherical setting, where the spherical convolution of Equation 26 can be accurately approximated by the DISCO representation

\[
(f \star \psi)(R) = \int_{S^2} f(\omega)\psi(R^{-1}\omega)d\mu(\omega)
\]

\[
\approx \sum_i f[\omega_i]\psi(R^{-1}\omega_i)q(\omega_i),
\]

where, for now, we consider 3D rotations \( R \in \text{SO}(3) \), i.e. \( f \star \psi : \text{SO}(3) \rightarrow \mathbb{R} \). The spherical signal \( f \) and quadrature weights \( q \) must clearly be discretized at sample positions \( \omega_i \). Again, critically, the filter \( \psi \) and group action \( R \) remain continuous, facilitating rotational equivariance.

**Approximation accuracy and equivariance.** We appeal to the sampling theorem on the sphere of McEwen and Wiaux (2011) since it provides the most efficient sampled signal representation (halving the Nyquist rate compared to Driscoll and Healy 1994). For spherical signals bandlimited at \( L \), all information content of the underlying continuous signal is captured in the \( \sim 2L^2 \) samples \( \{ f[\omega_i] \} \). The only source of approximation error is then the discretization of the integral. By adopting the exact quadrature of McEwen and Wiaux (2011) this integral could in principle be computed exactly (see Appendix A of Ocampo et al. (2023)). However, since the integrand is the product of two signals each bandlimited at \( L \), it is itself bandlimited at \( 2L \) (Driscoll and Healy, 1994). Exact quadrature would thus require the signals to be sampled at \( \sim 2(2L)^2 = 8L^2 \) positions. For our implementation we avoid upsampling signals and tradeoff a small approximation error to avoid a four-fold increase in computation. Nevertheless, we do adopt the quadrature weights of McEwen and Wiaux (2011) to keep approximation errors to a minimum. Since the approximation remains highly accurate, and the action of rotations on filters is treated continuously, the DISCO spherical convolution exhibits excellent SO(3) rotational equivariance properties (see Section 6.3.3).

**Computational considerations.** The naive computational scaling of the DISCO spherical convolution is \( O(N^2) \). When considering localized filters with on average \( K \) non-zero terms, with \( K \ll N \), the scaling of the DISCO spherical convolution is reduced to \( O(N) \), again just as in the Euclidean setting. In comparison to the continuous spherical convolution approach computed via Fourier space, the computation cost is dominated by the Fourier transform on the sphere, which naively is \( O(N^2) \) but for fast algorithms scales as \( O(N^{3/2}) \) (e.g. McEwen and Wiaux, 2011).
6.3 Discrete-Continuous (DISCO) convolutions

\begin{align*}
(a) & \quad R = Z(\alpha)Y(\beta)Z(\gamma) \in \text{SO}(3) \text{ rotation} \\
(b) & \quad R = Z(\alpha)Y(\beta) \in \text{SO}(3)/\text{SO}(2) \text{ rotation}
\end{align*}

Figure 6.2: SO(3) and SO(3)/SO(2) rotations of a localised kernel.

Restricting rotations to SO(3)/SO(2). While the DISCO spherical convolution is already efficient, we seek further computational savings by reducing the space of rotations considered from SO(3) to SO(3)/SO(2) (we provide a visual illustration of SO(3) and SO(3)/SO(2) rotations in Fig. 6.2). These two rotations may be represented using the zyz Euler convention:

\begin{align*}
Z(\alpha)Y(\beta)Z(\gamma) & \in \text{SO}(3) \quad (32) \\
Z(\alpha)Y(\beta) & \in \text{SO}(3)/\text{SO}(2) \quad (33)
\end{align*}

where \( \alpha, \gamma \in [0, 2\pi), \beta \in [0, \pi] \) and \( Z, Y \) are rotation operators around the \( z, y \) axes. It is easy to see that SO(3)/SO(2) is isomorphic to \( S^2 \) since the sphere may be written with co-ordinates \( (\alpha, \beta) \). Restricting rotations to SO(3)/SO(2) is analogous to typical Euclidean planar CNNs, where filters are translated across the image but are not rotated in the plane. However, as the space SO(3)/SO(2) is not a group, we no longer have group equivariance (as elaborated in Section 6.3.3). To avoid confusion, we denote the spherical convolution when restricting \( R \in \text{SO}(3)/\text{SO}(2) \) by \( f \circ \psi : \text{SO}(3)/\text{SO}(2) \to \mathbb{R} \), using a different symbol to the SO(3) spherical convolution. Since SO(3) is a three dimensional space, whereas SO(3)/SO(2) is two dimensional, restricting rotations to SO(3)/SO(2) reduces the dimension of the output space from \( L^3 \) to \( L^2 \), yielding an overall computational scaling of \( O(N) = O(L^2) \), where we recall \( L \) denotes the spherical signal bandlimit.
6.3.3. Rotational equivariance

**SO(3) rotational equivariance.** Consider the DISCO spherical convolution \( f \star \psi \) for rotation \( R \in \text{SO}(3) \), with \( Q \) denoting the action of the rotation \( Q \in \text{SO}(3) \) on spherical signals. We show \( f \star \psi \) satisfies SO(3) rotational equivariance:

\[
((Qf) \star \psi)(R) \approx \sum_i (Qf)[\omega_i]\psi(R^{-1}\omega_i)q(\omega_i) \\
= \sum_i f[\omega_i]\psi((Q^{-1}R)^{-1}\omega_i)q(\omega_i) \\
\approx (f \star \psi)(Q^{-1}R) \\
= (Q(f \star \psi))(R).
\]

The only approximations involved here (highlighted by the approximate equality symbols) follow from the quadrature of the integral. Critically, no approximation is involved in applying the rotation \( Q \) due to the continuous representation of rotations and the filter \( \psi \), i.e. equation 35 is exact. Consequently, the DISCO spherical convolution \( f \star \psi \) exhibits excellent SO(3) rotational equivariance. An important point to note in the SO(3) rotational equivariance proof, which is often ignored, is that it is only possible to relate the expression back to a spherical convolution (equation 36), if the input to \( f \star \psi \) is always \( \in \text{SO}(3) \). The closure property of a group implies that \( Q^{-1}R \in \text{SO}(3) \), i.e. two SO(3) rotations give another SO(3) rotation. This is not the case if the input space of \( f \star \psi \) does not exhibit a group structure, as discussed below.

**Asymptotic SO(3) rotational equivariance.** Consider the DISCO spherical convolution \( f \Theta \psi \) and rotations \( R \in \text{SO}(3)/\text{SO}(2) \) and \( Q \in \text{SO}(3) \) applied in equation 34-37. Since \( \text{SO}(3)/\text{SO}(2) \) is not a group, \( Q^{-1}R \notin \text{SO}(3)/\text{SO}(2) \) and equation 36 does not hold for \( f \Theta \psi \). Consequently, \( f \Theta \psi \) for \( R \in \text{SO}(3)/\text{SO}(2) \) does in general not satisfy SO(3) or \( \text{SO}(3)/\text{SO}(2) \) equivariance (contrast with the Euclidean planar setting that is translationally equivariant for the analogous setting). However, the DISCO spherical convolution \( f \Theta \psi \) does satisfy a form of asymptotic SO(3) equivariance. Defining the rotations as,

\[
Q = Z(\alpha)Y(\beta)Z(\gamma) \\
R = Z(\alpha')Y(\beta') \\
\implies Q^{-1}R = Z(-\gamma)Y(-\beta)Z(\alpha' - \alpha)Y(\beta').
\]

As \( \beta \to 0 \), we have \( Q^{-1}R \to Z(\alpha' - \alpha - \gamma)Y(\beta') \in \text{SO}(3)/\text{SO}(2) \) and therefore we may use the trick of equation 36 which leads to \( ((Qf) \Theta \psi)(R) \to (Q(f \Theta \psi))(R) \), i.e. we recover asymptotic SO(3) rotational equivariance as \( \beta \to 0 \). Note that there
is no restriction on $\alpha$ or $\gamma$, i.e. we achieve full rotational equivariance for rotations about the $z$ axis. This asymptotic SO(3) equivariance is validated numerically in Section 6.5.1 and shown to be reasonably accurate even for moderately large $\beta$. Asymptotic SO(3) equivariance is of significant practical use since content in spherical signals is often orientated and similar content often appears at similar latitudes, particularly for 360° panoramic photos and video, e.g. a table is almost always on the floor and its legs vertical. Hence, asymptotic SO(3) rotational equivariance may be more desirable than SO(3) equivariance for many applications.

**Axisymmetric filters and SO(3) rotational equivariance.** The rotational equivariance considerations presented above hold for general filters that take any value or form, we call these directional filter. Axisymmetric filters, i.e. filters that are invariant to rotations about their own axis, may also be considered. While axisymmetric filters are less expressive than directional filters they have nevertheless been shown to be effective for spherical CNNs (Esteves et al., 2018). For axisymmetric filters the SO(3) spherical convolution $f \star \psi$ and the SO(3)/SO(2) spherical convolution $f \odot \psi$ are the same since $f \star \psi$ for $R = Z(\alpha)Y(\beta)Z(\gamma) \in SO(3)$ is independent of $\gamma$. This is due to $\psi$ being unchanged after the first rotation $Z(\gamma)$ in equation 31. Consequently, the DISCO spherical convolution $f \odot \psi$ for $R \in SO(3)/SO(2)$ with an axisymmetric filter $\psi$ does exhibit SO(3) rotational equivariance (as validated numerically in Section 6.5.1). For an illustration, see Figure 6.3.

Figure 6.3: Illustration of SO(3) equivariance of a spherical convolution with an axi-symmetric kernel. Here the rotation $Q \in SO(3)$ is done before or after the convolution. The signals here are illustrated with a sphere ($S^2$) which is isomorphic to SO(3)/SO(2).
6.3 Discrete-Continuous (DISCO) convolutions

6.3.4. Filter parameterization

We consider a continuous representation of the filter $\psi$ in the DISCO spherical convolution, which allows us to evaluate $\psi(\omega)$ for any coordinate $\omega \in S^2$, thereby allowing us to compute $\psi(R^{-1}\omega)$ for any continuous rotation $R$ exactly. Nevertheless, the filter must be parameterized by a finite number of learnable parameters $p$. We consider filters that are either axisymmetric $\psi(\theta; p)$, directional and separable $\psi(\theta, \phi; p) = \vartheta(\theta; p)\varphi(\phi; p)$, or directional $\psi(\theta, \phi; p)$ without any further constraint. In all cases the filters are parameterized with equally spaced nodes along $\theta$ and $\phi$, with each node value a learnable parameter. To evaluate the filter for any continuous argument we simply use linear interpolation between the nodes. We localize the filter spatially by constraining it to be zero for angles above some cutoff angle $\theta_{\text{cutoff}}$.

6.3.5. DISCO transposed spherical convolution

Dense pixel-wise predictions are required for numerous deep learning tasks such as semantic segmentation. Existing equivariant spherical CNNs cannot support dense predictions since they are too computationally demanding. A DISCO transposed spherical convolution can be defined to increase the resolution of output signals in an analogous manner to the transposed convolutions considered in Euclidean U-Net architectures (Ronneberger et al., 2015), thereby supporting dense predictions. We define the DISCO transposed spherical convolution $f \otimes^\dagger \psi : S^2 \to \mathbb{R}$, restricting rotations to $\text{SO}(3)/\text{SO}(2)$,

$$
(f \otimes^\dagger \psi)(\omega) = \int_{S^2} f(\omega')\psi(R_{\omega'}^{-1}\omega)d\mu(\omega')
$$

$$
\approx \sum_i f[\omega_i]\psi(R_{\omega_i}^{-1}\omega)q(\omega_i),
$$

where we introduce the notation $R_{\omega} = Z(\phi)Y(\theta) \in \text{SO}(3)/\text{SO}(2)$ for $\omega = (\theta, \phi)$ in order to make the dependence of the rotation on the integration variable explicit. The output of the transposed convolution is the result of summing all filters rotated to each $R_{\omega_i}$, with a weight of $f[\omega_i]$, analogous to the Euclidean transposed convolution. Identical approximation and equivariance properties as the (non-transposed) DISCO spherical convolution hold.
6.4. Efficient implementation

The DISCO convolution affords a computationally scalable implementation through sparse tensor representations*. While we focus on the spherical setting of immediate practical interest, the same principles can be applied to any compact group. We leverage sparse-dense tensor multiplication operators to compute the DISCO spherical convolution efficiently on hardware accelerators (e.g. GPUs, TPUs). By exploiting symmetries of the sampling scheme we also describe how memory usage can be optimized. Automatic differentiation of sparse tensor operations that include learnable parameters is not supported in either TensorFlow or PyTorch, hence we derive custom sparse gradient representations to be used when training models.

6.4.1. Sparse tensor representation

Representation and sampling. The DISCO spherical convolution for rotations $R \in \text{SO}(3)/\text{SO}(2)$, here denoted by the variable $h = f \odot \psi$ for brevity, may be written by the tensor multiplication

$$h_j = \sum_i \Psi_{ji} f_i,$$

where $\Psi$ is a tensor with elements $\Psi_{ji} = \psi(R_j^{-1} \omega_i)$, $f_i$ is the sampled spherical input signal (collapsed to a vector) with quadrature weights absorbed, and $h_j$ is the convolved output at $R_j = Z(\phi_j)Y(\theta_j)$, or more simply, output coordinate $(\theta_j, \phi_j)$. We adopt the sampling theorem of McEwen and Wiaux (2011) so that all of the information content of bandlimited spherical signals can be captured in a finite set of $\sim 2L^2$ samples (see section 6.4.2). We also adopt the associated quadrature weights but, as discussed in Section 6.3.2, do not upsample signals, trading off a small approximation error in numerical integration to realize a significant reduction in computational cost. While the numerical integration is then no longer exact, it remains highly accurate.

Sparse representation. To compute the elements of $\Psi$ we first compute the continuously rotated coordinates, which we denote as tensors $(\Theta_{ji}, \Phi_{ji}) = R_j^{-1} \omega_i$, and then pass them to the continuous filter function to evaluate $\Psi_{ji} = \psi(\Theta_{ji}, \Phi_{ji})$ without any discretization error. For spatially localized filters, $\Psi$ will be sparse and we need only compute the rotated coordinates if $\Psi_{ji}$ will be non-zero. We exploit knowledge of the support of the filters, given by $\theta_{\text{cutoff}}$, to limit the number of rotated coordinates that need to be computed. Consequently, $\Theta$ and $\Phi$ are also sparse tensors since we

---

*We use the word tensor here to mean an array of numbers given their indices. For example $A_{ijkl}$ is a 4-D tensor and each combination of indices gives a number, e.g. $A_{0412} = 1.23, A_{0000} = 4$. A tensor here can be thought of as one of python’s numpy arrays, tensorflow’s tensors or Matlab’s arrays.
compute only coordinates where $\Psi$ will be non-zero. The sparse representation of $\Psi$ can thus be computed efficiently and Equation 43 can then be computed by a sparse-dense tensor multiplication (or sparse-dense matrix multiplication). For transposed spherical convolutions, $\Psi$ is simply replaced by $\Psi^\dagger$. DISCO spherical convolutions and transpose convolutions can thus be computed efficiently by sparse-dense tensor operations that map well onto hardware accelerators, which for practical purposes can result in substantial enhancements in computational performance.

**Scalable computation.** Empirically we find that the computational cost of the DISCO spherical convolution scales linearly in the number of spherical pixels, i.e. as $O(N) = O(L^2)$, as predicted; furthermore, not only does it scale more favorably than the most efficient alternative equivariant spherical convolution layer but its absolute cost is also considerably lower (see section 6.4.2). For example, for 4k spherical images we achieve a saving in computational cost of $10^9$.

**Architectures.** Efficient spherical implementations of common CNN architectures can then be constructed by combining the DISCO forward and transpose spherical convolutions with pointwise non-linear activations and other common architectural features, such as skip connections, batch-normalization*, multiple channels, etc. Note that in the architectures we consider we typically adopt depthwise separable convolutions (Chollet, 2017) for computational efficiency.

### 6.4.2. Memory optimizations

**Explicit sampling.** For structured spherical sampling schemes many of the elements of $\Psi$ are repeated, which opens the possibility of memory compression. While memory may be compressed for various structured samplings, we focus on the sampling scheme that we adopt. Specifically, we sample spherical coordinates by $(\theta_t, \phi_p) = (\pi t/L, \pi p/L)$ for $t, p \in \mathbb{Z}$, $0 \leq t \leq L$, $0 \leq p \leq 2L - 1$ (McEwen and Wiaux, 2011; Daducci et al., 2011), where recall $L$ is the spherical harmonic bandlimit, which plays the role of resolution, yielding $2L(L+1)$ pixels on the sphere. Incidentally, this sampling scheme maps onto the typical sampling of equirectangular $360^\circ$ panoramic images well.

**Compressed tensor.** For clarity we write the input and output coordinates in terms of their individual spherical coordinates $\omega_i = (\theta_t, \phi_p)$ and $\omega_j = (\theta_{t'}, \phi_{p'})$, respectively, i.e. $i = (t, p)$ and $j = (t', p')$. It is then clear that the filter tensor $\Psi$ is a 4-dimensional tensor that may be written explicitly as $\Psi_{ji} = \Psi_{t'p'tp}$. By exploiting symmetries of our sampling scheme $\Psi$ can be reduced to 3 dimensions by noticing

---

*In order to ensure batch-normalization for sampled spherical signals is rotationally equivariant we adapt the standard batch-normalization to compute averages over the sphere, with appropriate sample weighting.
that the two tensor elements $\Psi_{t'p tp}$ and $\Psi_{t0t(p−p')}$ are equal. This follows since a rotation in $Z(\phi_{p'})$ is equivalent to a translation in the input coordinates by $\phi_{p'}$:

$$\Psi_{t'p tp} = \psi((Z(\phi_{p'})Y(\theta_{t}))(\theta_{t}, \phi_{p}))$$

$$= \psi(Y^{-1}(\theta_{t}))(\theta_{t}, \phi_{p} - \phi_{p}')$$

$$= \Psi_{t0t(p−p')}.$$  \hfill (44)

For our sampling scheme the set of all translated coordinates falls on the set of original coordinates, i.e. $\{p−p'\} = \{p\}$, and hence we can store $\Psi_{t0t(p−p')}$ as $\Psi_{t0tp}$. For a dense representation $\Psi_{t0tp}$ has $O(L^3)$ elements; however, for localized filters this is reduced to $O(L^2)$ non-zero elements; hence, memory usage also scales linearly with number of spherical pixels.

**Efficient compressed sparse-dense tensor multiplication.** The compressed version of $\Psi$ can be used efficiently in the sparse tensor multiplication by re-writing Equation 43 as

$$h_{t'p'} = \sum_{tp} \Psi_{t'p tp} f_{tp}$$

$$= \sum_{tp} \Psi_{t0t(p−p')} f_{tp}$$

$$= \sum_{tp} \Psi_{t0tp} f_{t(p+p')}.$$  \hfill (48)

Notice that the last line is a multiplication between $\Psi_{t0tp}$ and $f_{tp}$ but shifted by $p'$ in the $p$ coordinate. Hence, we can implement the compressed tensor multiplication by looping over $p'$ and simply shifting the input in $p$ every iteration. This is considerably more efficient than repeatedly shifting the elements of a sparse tensor. To support multiple channels we adopt depthwise separable convolutions (Chollet, 2017) for computational and memory efficiency.

**Scalable memory usage.** Empirically we find that when incorporating the optimizations discussed above, memory requirements and computational cost of the DISCO spherical convolution indeed scale linearly with number of spherical pixels, i.e. as $O(L^3)$, as predicted, providing a considerable saving over the most efficient alternative equivariant spherical convolution layer as shown in Figure 6.4. For example, for 4k spherical images we achieve a saving in memory of $10^4$. We provide further details of how Figure 6.4 is computed in Appendix E.
6.4 Efficient implementation

(a) Computational cost

(b) Memory usage

Figure 6.4: Comparison of floating point operations (FLOPs) and memory usage of the DISCO spherical convolution and the most efficient alternative equivariant spherical convolution (an axisymmetric harmonic convolution) (Esteves et al., 2018; Cobb et al., 2021).

Inspecting the theoretical computational cost, as considered above, has the advantage that it abstracts away details of a particular implementation, its level of optimization, and the hardware on which the model runs. Wall-clock compute time is nevertheless of practical interest and so is also evaluated. On an NVIDIA RTX 3090 GPU we observe a wall-clock compute time of 0.0302 ± 0.0018, 0.0898 ± 0.0025, and 0.3255 ± 0.0043 seconds for resolutions of $L = 1024$, $L = 2048$, and $L = 4096$, respectively, when averaged over 10 experiments.

6.4.3. Sparse gradients

Deep learning models are typically trained by optimization algorithms that use gradients, which are computed efficiently by automatic differentiation. However, automatic differentiation of sparse tensor operations that include learnable parameters, as in the DISCO convolution, is not supported either by TensorFlow or PyTorch. Thus, we implement custom sparse gradients which is outlined in Appendix F.
6.5 Experiments

We first demonstrate the equivariance properties of our scalable DISCO spherical convolutional layers, before tackling a number of dense-prediction benchmark problems with spherical CNNs built using our DISCO framework (implemented in the CopernicAI* code). We achieve state-of-the-art performance on all benchmarks. See Appendix G for further details on benchmark experiments.

6.5.1. Equivariance tests

Method. To demonstrate the equivariance of our DISCO layers at high resolution we perform equivariance tests for \( L \in \{128, 256, 512, 1024\} \), using a method similar to Cobb et al. (2021, Appendix D). We measure the extent to which our DISCO spherical convolution operator \( D \) is equivariant by evaluating the mean relative equivariance error with respect to random rotation operators \( Q \) and random signals \( f \),

\[
\epsilon(D(Qf), Q(D(f))) = \frac{1}{N_f N_Q} \sum_{i=1}^{N_f} \sum_{j=1}^{N_Q} \frac{\|D(Q_j f_i) - Q_j (D(f_i))\|}{\|D(Q_j f_i)\|} \tag{50}
\]

resulting from pre-rotation of the signal, followed by application of \( D \), as opposed to post-rotation after application of \( D \). The operator norm \( \|\cdot\| \) is defined by the inner product on the sphere \( \langle \cdot, \cdot \rangle_{L^2(S^2)} \). We consider \( N_f = 50 \) random spherical signals \( \{f_i\}_{i=1}^{N_f} \) with harmonic coefficients sampled from the standard normal distribution and \( N_Q = 50 \) random rotations \( \{Q_j\}_{j=1}^{N_Q} \) sampled uniformly on SO(3) for axisymmetric kernels. To demonstrate asymptotic SO(3) equivariance for directional filters (see Section 6.3.2), we consider rotations of the form \( Q_j = Z(\alpha_j)Y(\beta)Z(\gamma_j) \) for random \( \alpha_j, \gamma_j \) and constant \( \beta \in \{0^\circ, 5^\circ, 10^\circ\} \). We do not impose filters be bandlimited. As the smoothness of the filter reduces, spherical aliasing is increased, and equivariance errors are increased. We therefore consider best and worst cases to bracket equivariance errors. For the worst case scenario we allow filter parameters to be random and consider \( \theta_{\text{cutoff}} = 5\pi/L \), with 4 nodes along \( \theta \) and \( \phi \). For the best case scenario we use the same nodes and cutoff but consider smoother filters, with the axisymmetric filter given by the smooth function \( \psi_S(\theta) = \exp(-\theta^2_{\text{cutoff}}/((\theta^2_{\text{cutoff}} - \theta^2))) \) at node positions and the directional filter by \( \psi_S(\theta) \cos(\phi) \).

*Can be found at: https://www.kagenova.com/products/copernicAI and will be made publicly available this year.
Table 8: Mean relative equivariance error expressed as a percentage (i.e. $100 \times \epsilon$) for axisymmetric and directional filters (best and worst cases are shown, where worst is shown in gray inside brackets).

<table>
<thead>
<tr>
<th>Resolution $L$</th>
<th>Axisymmetric filters $\beta = 0^\circ$</th>
<th>$\beta = 5^\circ$</th>
<th>$\beta = 10^\circ$</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>$0.04 \pm 0.01 (0.27 \pm 0.02)$</td>
<td>$0.02 \pm 0.00 (0.04 \pm 0.03)$</td>
<td>$0.86 \pm 0.05 (1.11 \pm 0.60)$</td>
</tr>
<tr>
<td>256</td>
<td>$0.04 \pm 0.01 (0.24 \pm 0.03)$</td>
<td>$0.01 \pm 0.00 (0.02 \pm 0.01)$</td>
<td>$0.83 \pm 0.02 (1.18 \pm 0.72)$</td>
</tr>
<tr>
<td>512</td>
<td>$0.04 \pm 0.01 (0.48 \pm 0.03)$</td>
<td>$0.00 \pm 0.00 (0.01 \pm 0.01)$</td>
<td>$0.83 \pm 0.01 (1.31 \pm 0.85)$</td>
</tr>
<tr>
<td>1024</td>
<td>$0.04 \pm 0.01 (0.55 \pm 0.00)$</td>
<td>$0.00 \pm 0.00 (0.01 \pm 0.00)$</td>
<td>$0.82 \pm 0.00 (1.04 \pm 0.69)$</td>
</tr>
</tbody>
</table>

Results. Equivariance errors computed in this manner for the DISCO spherical convolution are presented in Table 8. For axisymmetric filters we achieve excellent rotational equivariance. For directional filters, for which equivariance is asymptotic since we consider $\text{SO}(3)/\text{SO}(2)$ rotations, we achieve excellent equivariance in the limit of latitudinal rotation $\beta = 0^\circ$. As $\beta$ increases, equivariance errors increase as expected but, nevertheless, we observe only moderate equivariance error of just a few percent at $\beta = 10^\circ$ (this compares favorably to the equivariance error of $\sim 35\%$ for a spherical ReLU at $L = 32$; Cohen et al. 2018; Cobb et al. 2021).

6.5.2. Rotated MNIST on the sphere

We consider the benchmark of classifying randomly rotated MNIST digits projected onto the sphere (Cohen et al., 2018) for three experimental modes NR/NR, R/R and NR/R, indicating whether the training/test sets, respectively, have been rotated (R) or not (NR). Results are presented in Table 9 for bandlimit $L = 1024$. The DISCO framework performs well at high resolution and, moreover, similar performance is achieved in all rotational settings, demonstrating excellent rotational invariance.

Table 9: Results for spherical MNIST showing the accuracy of predictions for each model as a percentage. The columns represent if the training/test set has been rotated (R) or not (NR).

<table>
<thead>
<tr>
<th></th>
<th>NR/NR</th>
<th>R/R</th>
<th>NR/R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planar</td>
<td>98.3</td>
<td>53.9</td>
<td>14.3</td>
</tr>
<tr>
<td>DISCO- Axisymmetric</td>
<td>98.6</td>
<td>98.7</td>
<td>98.6</td>
</tr>
</tbody>
</table>
6.5.3. Semantic segmentation

We consider semantic segmentation of 360° photos where the input is a RGB spherical image and the output is the corresponding image where each pixel has a predicted label (or category), see Figure 6.5 for an example. We use a common backbone of a DISCO spherical convolutional residual U-Net architecture, and compare to existing benchmark results (note that previous experiments have been performed at different resolutions; hence we quote the effective resolution $\tilde{L}$ determined by number of samples on the sphere).

Evaluation metrics. Intersection-over-union (IoU) is a standard metric used in semantic segmentation tasks. It helps determine how well the predicted segmentation aligns with the true segmentation at the pixel level. Given a set of true pixels $T_c$ with a given category $c$ (e.g. the pixels of a dog in an image), and the predicted pixels with the same category, $P_c$, the IoU is given by*:

$$\text{IoU} = \frac{|T_c \cap P_c|}{|T_c \cup P_c|} = \frac{\text{Area of overlap between } T_c, P_c}{\text{Combined area of } T_c, P_c}. \quad (51)$$

A perfectly predicted segmentation will have $P_c = T_c$ and IoU= 1, but typically a good IoU score is $> 0.4$. Another metric we use here is the accuracy of correctly predicted pixels. This is simply the fraction of correctly predicted pixel categories over all the true pixel categories,

$$\text{Accuracy} = \frac{|T_c \cap P_c|}{|T_c|}. \quad (52)$$

Omni-SYNTHIA dataset of outdoor 360° photos. Results for the semantic segmentation of outdoor city scenes of the Omni-SYNTHIA dataset (Ros et al., 2016) are presented in Table 10 and Figure 6.5. We consider a DISCO-Separable architecture, with and without augmentation for ablation purposes. In both settings we achieve state-of-the-art performance. For details of architecture, training and data see appendix G.3.

*The $| \cdot |$ operator counts the number of elements (in this case pixels) in the set, e.g. $|T|$ is the number of elements in the set $T$. 
Table 10: Results for Omni-SYNTHIA showing the mean intersection over union (mIoU) and mean accuracy (mAcc) both as a percentage. The mean is taken over all categories, e.g. the IoU is computed for each category over the whole dataset, the mean is then computed using all those IoUs. For comparison we also include previous results from different types of spherical CNNs as well as the effective resolution used, $\tilde{L}$.

<table>
<thead>
<tr>
<th>Model</th>
<th>mIoU</th>
<th>mAcc</th>
<th>$\tilde{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planar UNet</td>
<td>38.8</td>
<td>45.1</td>
<td>143</td>
</tr>
<tr>
<td>UGSCNN (Jiang et al., 2019)</td>
<td>36.9</td>
<td>50.7</td>
<td>143</td>
</tr>
<tr>
<td>HexUNet (Zhang et al., 2019)</td>
<td>43.6</td>
<td>52.2</td>
<td>143</td>
</tr>
<tr>
<td>TangentImg (Eder et al., 2020)</td>
<td>41.3</td>
<td>52.8</td>
<td>143</td>
</tr>
<tr>
<td>Planar UNet</td>
<td>44.6</td>
<td>52.6</td>
<td>286</td>
</tr>
<tr>
<td>UGSCNN (Jiang et al., 2019)</td>
<td>37.6</td>
<td>48.9</td>
<td>286</td>
</tr>
<tr>
<td>HexUNet (Zhang et al., 2019)</td>
<td>48.3</td>
<td>57.1</td>
<td>286</td>
</tr>
<tr>
<td>TangentImg (Eder et al., 2020)</td>
<td>35.8</td>
<td>55.3</td>
<td>286</td>
</tr>
<tr>
<td>DISCO-Separable (Ours)</td>
<td>48.3</td>
<td>59.3</td>
<td>256</td>
</tr>
<tr>
<td><strong>DISCO-Separable-Aug (Ours)</strong></td>
<td><strong>49.2</strong></td>
<td><strong>63.7</strong></td>
<td><strong>256</strong></td>
</tr>
</tbody>
</table>

Figure 6.5: Example predictions for semantic segmentation of Omni-SYNTHIA data. Here we have used a Mollweide projection in order to see the whole surface of the sphere.
2D3DS dataset of indoor 360° photos. Results for the semantic segmentation of indoor scenes of the 2D3DS dataset (Armeni et al., 2017) are presented in Table 11 and Figure 6.6. Our DISCO-Directional architecture outperforms the DISCO-Axisymmetric architecture, which we attribute to being due to its asymptotic SO(3) equivariance (that can be considered as “orientation aware”) and having more expressive directional filters. We also include augmentation, which helps to improve performance. We achieve state-of-the-art performance compared to all previous methods. For details of architecture, training and data see appendix G.2.

Table 11: Results for 2D3DS showing the mean intersection over union (mIoU) and mean accuracy (mAcc) both as a percentage. For comparison we also include previous results from different types of spherical CNNs as well as the effective resolution used, \( \tilde{L} \).

<table>
<thead>
<tr>
<th>Model</th>
<th>mIoU</th>
<th>mAcc</th>
<th>( \tilde{L} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planar UNet</td>
<td>35.9</td>
<td>50.8</td>
<td>72</td>
</tr>
<tr>
<td>UGSCNN (Jiang et al., 2019)</td>
<td>38.3</td>
<td>54.7</td>
<td>72</td>
</tr>
<tr>
<td>GaugeNet (Cohen et al., 2019)</td>
<td>39.4</td>
<td>55.9</td>
<td>72</td>
</tr>
<tr>
<td>HexRUNet (Zhang et al., 2019)</td>
<td>43.3</td>
<td>58.6</td>
<td>72</td>
</tr>
<tr>
<td>SWSCNNs (Esteves et al., 2020)</td>
<td>43.4</td>
<td>58.7</td>
<td>91</td>
</tr>
<tr>
<td>CubeNet (Shakerinava and Ravanbakhsh, 2021)</td>
<td>45.0</td>
<td>62.5</td>
<td>83</td>
</tr>
<tr>
<td>MöbiusConv (Mitchel et al., 2022)</td>
<td>43.3</td>
<td>60.9</td>
<td>91</td>
</tr>
<tr>
<td>TangentImg (Eder et al., 2020)</td>
<td>41.8</td>
<td>54.9</td>
<td>286</td>
</tr>
<tr>
<td>HoHoNet (Sun et al., 2021)</td>
<td>43.3</td>
<td>53.9</td>
<td>256</td>
</tr>
<tr>
<td>DISCO-Axisymmetric (Ours)</td>
<td>39.7</td>
<td>54.1</td>
<td>256</td>
</tr>
<tr>
<td>DISCO-Separable (Ours)</td>
<td>43.9</td>
<td>60.9</td>
<td>256</td>
</tr>
<tr>
<td>DISCO-Directional (Ours)</td>
<td>45.2</td>
<td>61.5</td>
<td>256</td>
</tr>
<tr>
<td><strong>DISCO-Directional-Aug (Ours)</strong></td>
<td><strong>45.7</strong></td>
<td><strong>62.7</strong></td>
<td><strong>256</strong></td>
</tr>
</tbody>
</table>

Figure 6.6: Example predictions for semantic segmentation of 2D3DS data. Here we have used a Mollweide projection in order to see the whole surface of the sphere.
6.5.4. Depth estimation

We consider the task of monocular depth estimation from 360° photos where the input is a RGB spherical image and the output is the corresponding image where each pixel has a predicted depth. We tackle the Pano3D benchmark (Albanis et al., 2021) for the Matterport3D dataset (Chang et al., 2017). We adopt the same backbone architecture as for semantic segmentation. Results are presented in Table 12 and Figure 6.7. Pano3D is a relatively new benchmark; the only previous results are those of Albanis et al. (2021). We achieve excellent performance with significantly fewer parameters (40× less). While we do not achieve the best performance across all metrics (which we suspect is due to the alternative model being trained with a virtual normal loss that leverages surface normal information), we perform better on accuracy metrics than error metrics and nevertheless achieve the state-of-the-art in six of nine metrics. For details of architecture, training and data see appendix G.4.

Table 12: Results for Pano3D. To measure the difference in predicted vs true depths we have chosen similar metrics to Albanis et al. (2021). The range of depths are measured between 0 to 8 metres and so the error is a comparison between true and predicted depth in metres. For relative errors, it is measured as a fractional difference. To avoid skewing of mean errors, we have chosen to weight the error of each pixel by the area of each pixel. This is denoted by the \( w \). The error metrics chosen are the root mean square error (RMSE), root mean square logarithmic error (RMSLE), absolute relative error (AbsRel) and square relative error (SqRel). We also include depth accuracy metrics similar to Albanis et al. (2021), given by \( \delta_{1.05}^{ico} \) which translates to the percentage of predicted pixels that are within 1.05 times the value of the true pixels, and similarly for the rest of the \( \delta_{i}^{ico} \).

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Depth Error Metrics</th>
<th>Depth Accuracy Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \text{wRMSE} )</td>
<td>( \text{wRMSLE} )</td>
</tr>
<tr>
<td>Planar UNet</td>
<td>27M</td>
<td>0.4520</td>
<td>0.1300</td>
</tr>
<tr>
<td>(Albanis et al., 2021)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISCO-Directional</td>
<td>658k</td>
<td>0.5063</td>
<td>0.1695</td>
</tr>
<tr>
<td>(Ours)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.6 Conclusions

We have developed the DISCO (discrete-continuous) group convolution that is simultaneously computationally scalable and equivariant. While our framework can be applied to any compact group, we specialize to the sphere where our DISCO spherical convolutions can exhibit full or asymptotic SO(3) rotational equivariance. We present a sparse tensor implementation with custom gradients that exhibits excellent rotational equivariance properties, is well-suited to hardware accelerators (e.g. GPUs, TPUs), and achieves linear scaling in both computational cost and memory usage. For 4k spherical images we realize a saving of $10^9$ in computational cost and $10^4$ in memory usage when compared to the most efficient alternative equivariant spherical convolution. We apply our DISCO spherical CNN framework to numerous dense-prediction tasks, such as semantic segmentation and depth estimation, achieving state-of-the-art performance on all benchmarks.

Summary. The aim of this chapter was to improve the scalability of rotationally equivariant CNNs i.e. their memory and computation time complexity. Previous equivariant CNNs used harmonic transforms which are costly, but here the novelty of our work lies in the use of sparse matrices to make a direct computation of the spherical convolution. We implemented this efficiently using memory compression of sparse matrices and their sparse gradients. The theory for the DISCO layers was a group contribution. My contribution to this chapter were the implementation and theory of the sparse gradients, the efficient implementation of the DISCO layer itself and the benchmark experiments.
7. Summary

Throughout this thesis, we explored a wide range of scientific methods, including theoretical, experimental, computational and data-intensive methods in the fields of scintillation dosimetry and convolutional neural networks. To summarize, we initially set off trying to solve the problem of rapid and cheap quality assurance in small field radiotherapy using our scintillator sheet and camera detector setup. While we managed to get fast measurements, our images had low light output and a number of optical artefacts such as Cherenkov light. This reduced the accuracy of the extracted dose. To assist in solving this problem, we investigated the use of CNNs to extract the dose and other physical variables such as beam energy, for which we found much better results compared to manual dose extraction. Our results showed that CNNs performed better than classical methods and could provide dose distributions that are suitable for routine QA. We then set out to develop a novel method using CNNs for general applications, in particular for high resolution images on different manifolds such as the sphere. Our method was tested on a number of high resolution benchmarks for which we achieved state-of-the-art performance.

While our scintillator detector setup is not accurate enough to be used clinically, we have shown that it can be a time-saving method. We suspect it might be a viable tool in the near future given the system is further improved, for example by using a higher light output scintillator or using a CCD camera for a lower noise to signal ratio.

To complement the manual method of dose extraction from the scintillation images, we have also developed an end-to-end method using CNNs. It is common in radiotherapy to have small number of data examples, hence we used domain randomization where we simulated a larger and random dataset such that our CNN models can be trained to become robust to random variations in the images. We hope that this provides insight to the scintillation dosimetry community on how to utilize AI in this field.

Our final chapter developed rotationally equivariant CNNs for high resolution spherical images. We hope that this method may be used in future for any application with a spherical data domain. For example in 360 image segmentation and climate weather prediction, for which a PhD project is currently being offered as a continuation of this project.
A. Discolouration

Figure A.1: Radiation-induced damage in both scintillating and non-scintillating plastic materials. This damage manifests as a temporary green discoloration that persists for several hours before fading away. Notably, the discoloration is only visible when illuminated with light, effectively functioning as a green filter and assuming the shape of the deposited radiation dose.

B. Simulation details

B.1. Dose validation

Geant4 settings. For low-energy electron and gamma transport (below 1 GeV), the physics class \textit{G4EmLivermorePhysics} was used. The dose was measured in an array of 1mm$^3$ detector cubes which were positioned according to the type of measurement (PDD, profile or OF), the material of the detector was also water. The counting of the dose in each detector cube is implemented in the \textit{G4SteppingAction} class, this is done by adding up all the energy deposited from each step that is in that detector cube.

Beam source. The implementation to read the phase files was done following
(Cortés-Giraldo et al., 2009). These files contained the position and momenta of $10^7$ primary photons for each beam size. Cyberknife circular beams of field sizes 5mm, 7.5mm, 10mm, 15mm and 60mm were available (Biasi et al., 2018), but to get an arbitrary field size below 60mm, we can start with a 60mm beam and then kill any photon above a threshold radius to reduce the field to any size. This also allows us to simulate a constant monitor unit by having the same number of incoming photons from the beam head before collimation. The results are shown in Figure B.1.

![Figure B.1](image)

Figure B.1: Geant4 simulations of ion chamber in a water phantom compared with measured values from microDiamond™ (Chalkley and Heyes, 2014). (a) PDD - 15mm field size. (b) Profile - 15mm field size at 50mm depth. (c) OF - at 15mm depth.

**Output factor.** The OF is measured with the detector at a constant depth while varying the field size. This is done with a constant monitor unit across all field sizes. This effectively means that as you’re changing the field size, the number of photons that’s coming from the beam head before the beam is collimated, is kept constant. This is simulated by having the same number of primary gamma photons fired towards the phantom for each field size but killing photons before they hit the phantom if they are above a certain radius that’s defined by the field size.

### B.2. Depth of field

The effect of depth of field is intrinsically included in the image simulations. This is due to the fact that photons which are coming behind/infront of the focus plane, are not refocused onto the sensor. The depth of field can be approximated by (Levoy, 2010)

$$\text{DOF} \approx \frac{2u^2 Nc}{f^2}$$  \hspace{1cm} (53)
where $u = 35\text{cm}$ is the distance from the lens to the focus plane, $N$ is the f-number, $f$ is the focal length, $c$ is the circle of confusion. By placing the virtual sensor on the focus plane and using the lens equation, $f = \frac{u}{2}$. With aperture $D = 4.1\text{cm}$, $N = \frac{f}{D}=4.27$. When light rays emitted from a point are no longer focused onto the same pixel sensor, that point is out of focus and the circle of confusion is equal to the size of the pixel sensor, here we use the size of the virtual pixel sensor $c = 1\text{mm}$. This leads to $\text{DOF} = 3.41\text{cm}$, we verify this by imaging a black rectangle at different distances from the lens in Figure B.2.

![Figure B.2: Depth of field in Geant4 image simulation. $x$ is the distance of the rectangle from the focus plane.](image)

### C. Simulation parameter distributions

#### C.1. PDD

There are 1932 images each for scintillation and Cherenkov. The parameters for the simulations are sampled from the distributions in Figure C.1. For the SSDs, there is an additional 937 counts at 80cm. We note that there are more images with small field sizes as they are faster to simulate with less noise and hence uncertainty. The energies are randomly chosen on a log scale as dose differences are harder to distinguish when the energy is the same order of magnitude. Here all X-ray beam directions are perpendicular to the scintillator phantom because small changes in the direction cause large changes in the PDD while the image has an unnoticeable change.
C.2 Output factor

There are 960 images each for scintillation and Cherenkov. In experiments the OF is obtained with the same experimental conditions for each collimator size. For this reason, we simulate the images and OF in groups of 12 where each group has the collimator sizes [5, 7.5, 10, 12.5, 15, 20, 25, 30, 35, 40, 50, 60]mm, and each group has the same simulation parameters. The simulation parameters are sampled from the distributions shown in Figure C.2.

Figure C.1: Simulation parameter distributions for PDD data.

C.2. Output factor

There are 960 images each for scintillation and Cherenkov. In experiments the OF is obtained with the same experimental conditions for each collimator size. For this reason, we simulate the images and OF in groups of 12 where each group has the collimator sizes [5, 7.5, 10, 12.5, 15, 20, 25, 30, 35, 40, 50, 60]mm, and each group has the same simulation parameters. The simulation parameters are sampled from the distributions shown in Figure C.2.
C.3. Variables

**Energies.** For predicting energies, we use the same dataset for predicting PDDs (see Figure C.1). This is because that dataset has a logarithmic dependence on energy which is easier to distinguish between images.

**Scintillation ratio and Beam widths.** We use the same dataset for predicting the scintillation ratio $r$ and beam width which comprises of 3846 examples. The simulation parameter distributions are shown in Figure C.3. Smaller beam sizes are faster to simulate and hence we have more examples with less than 15mm beam width. We also vary the beam direction and its rotation centre (isocentre), the rotation axis is randomly chosen perpendicularly to the original beam axis and the rotation angle randomly chosen between -90 to 90 degrees. We also note that we have a lot of examples with rotation angle close to 0 as that is more realistic. We also note there is an additional 943 examples with energy at 6MV and SSD at 80cm.

Figure C.2: Simulation parameter distributions for OF data.
as they are more realistic.

![Simulation parameter distributions for OF data.](image)

**Figure C.3:** Simulation parameter distributions for OF data.

## D. CNN models for scintillation dosimetry

Here we provide details of how the models are made using the keras package from tensorflow (Abadi et al., 2015). First we list the classes needed below and then we build each of the models used for predicting PDD, output factor and variables.

**Import classes needed from keras:**

```python
from keras.layers import (Conv2D, Conv2DTranspose, Model, Dense, Input, GlobalAveragePooling2D)
```
D.1. PDD models

For the PDD models we start with a simple 3 layer CNN and improved it to an 11 layer encoder-decoder CNN.

Simple model:

```python
input_shape = (240, 240, 1)
inputs = Input(input_shape)

x = Conv2D(filters=32, kernel_size=7, strides=(1, 1), padding="same", activation="relu")(inputs)

x = Conv2D(filters=64, kernel_size=5, strides=(2, 2), padding="same", activation="relu")(x)

x = GlobalAveragePooling2D()(x)

output = Dense(units=200, activation="sigmoid", use_bias=False)(x)

model = Model(inputs=inputs, outputs=output)
```
### Improved model:

```python
input_shape = (240, 240, 1)
inputs = Input(input_shape)

x = Conv2D(filters=64, kernel_size=7, strides=(1, 1), padding="same",
          activation="relu")(inputs)

x = Conv2D(filters=64, kernel_size=5, strides=(2, 2), padding="same",
          activation="relu")(x)

x = Conv2D(filters=128, kernel_size=3, strides=(1, 1), padding="same",
           activation="relu")(x)

x = Conv2D(filters=128, kernel_size=3, strides=(2, 2), padding="same",
           activation="relu")(x)

x = Conv2D(filters=256, kernel_size=3, strides=(1, 1), padding="same",
           activation="relu")(x)

x = Conv2D(filters=256, kernel_size=3, strides=(2, 2), padding="same",
           activation="relu")(x)

x = Conv2D(filters=512, kernel_size=3, strides=(1, 1), padding="same",
           activation="relu")(x)

x = Conv2D(filters=512, kernel_size=3, strides=(2, 2), padding="same",
           activation="relu")(x)

x = GlobalAveragePooling2D()(x)

x = Dense(units=50, activation="relu")(x)

x = Dense(units=100, activation="relu", use_bias=False)(x)

output = Dense(units=200, activation="sigmoid", use_bias=False)(x)

model = Model(inputs=inputs, outputs=output)
```
D.2 Output factor models

For the output factor models, we started with a simple 3 Layer CNN and improved it to a 12 Layer CNN and further improved it to a 24 Layer ResNet.

Simple model, 3 Layers:

```python
input_shape = (240, 240, 2)
inputs = Input(input_shape)

x = Conv2D(filters=32, kernel_size=7, strides=(1, 1), padding="same", activation="relu")(inputs)

x = Conv2D(filters=64, kernel_size=5, strides=(2, 2), padding="same", activation="relu")(x)

x = GlobalAveragePooling2D()(x)

output = Dense(units=1, use_bias=False)(x)

model = Model(inputs=inputs, outputs=output)
```
Improved model, 12 Layers:

```python
input_shape = (240, 240, 2)
inputs = Input(input_shape)

x = Conv2D(filters=64, kernel_size=7, strides=(1, 1), padding="same", activation="relu")(inputs)

x = Conv2D(filters=64, kernel_size=5, strides=(2, 2), padding="same", activation="relu")(x)

x = Conv2D(filters=64, kernel_size=3, strides=(1, 1), padding="same", activation="relu")(x)

x = Conv2D(filters=64, kernel_size=3, strides=(2, 2), padding="same", activation="relu")(x)

x = Conv2D(filters=128, kernel_size=3, strides=(1, 1), padding="same", activation="relu")(x)

x = Conv2D(filters=128, kernel_size=3, strides=(2, 2), padding="same", activation="relu")(x)

x = Conv2D(filters=128, kernel_size=3, strides=(1, 1), padding="same", activation="relu")(x)

x = GlobalAveragePooling2D()(x)

x = Dense(units=128, use_bias=True, activation="relu")(x)

output = Dense(units=1, use_bias=False)(x)
model = tf.keras.models.Model(inputs=inputs, outputs=output)
```
Improved model, 24 Layer ResNet:

```python
width = 240
input_shape = (width, width, 2)
inputs = Input(input_shape)

x = Conv2D(64, kernel_size=7, strides=(1, 1), padding="same")(inputs)
x = Conv2D(64, kernel_size=5, strides=(2, 2), padding="same")(x)
x = ResBlock(channels=64)(x)
x = ResBlock(channels=64)(x)
x = ResBlock(channels=64)(x)
x = ResBlock(channels=128, downsample=True)(x)
x = ResBlock(channels=128)(x)
x = ResBlock(channels=128)(x)
x = ResBlock(channels=256, downsample=True)(x)
x = ResBlock(channels=256)(x)
x = ResBlock(channels=256)(x)
x = GlobalAveragePooling2D()(x)

x = Dense(units=128, activation="relu")(x)
output = Dense(units=1, use_bias=False)(x)
model = Model(inputs=inputs, outputs=output)
```

ResBlock:

```python
class ResBlock(Model):
    def __init__(self, channels, downsample=False):
        super(ResBlock, self).__init__()

        self.downsample = downsample
        if downsample:
            strides = (2, 2)
            self.convdown = Conv2D(channels, 1, strides=strides,
                                    padding="same", use_bias=True)
        else:
            strides = (1, 1)

            self.conv1 = Conv2D(channels, 3, strides=strides,
                                 padding="same", use_bias=True)
            self.conv2 = Conv2D(channels, 3, strides=(1, 1),
                                 padding="same", use_bias=True)

    def call(self, input_tensor):

        skip_tensor = input_tensor
        if self.downsample:
            skip_tensor = self.convdown(skip_tensor)

        x = self.conv1(input_tensor)
        x = tf.nn.relu(x)
        x = self.conv2(x)
        x = x + skip_tensor
        x = tf.nn.relu(x)

        return x
```
D.3. Scintillation ratio and beam width model

24 Layer ResNet:

```python
channels = 32
input_shape = (240, 240, 1)
inputs = Input(input_shape)

x = Conv2D(channels, 7, strides=(1, 1), padding="same",
            activation="leaky_relu")(inputs)

x = MaxPooling2D(pool_size=(2, 2), padding="same")(x)

x = ResBlock(channels)(x)
x = ResBlock(channels)(x)
x = ResBlock(channels)(x)
x = ResBlock(channels * 2, downsample=True)(x)
x = ResBlock(channels * 2)(x)
x = ResBlock(channels * 2)(x)
x = ResBlock(channels * 4, downsample=True)(x)
x = ResBlock(channels * 4)(x)
x = ResBlock(channels * 4)(x)
x = GlobalAveragePooling2D()(x)

x = Dense(channels, activation="leaky_relu")(x)
output = Dense(1, activation=None, use_bias=False)(x)
model = Model(inputs=inputs, outputs=output)
```

D.4. Beam energy model

5 convolutional, 2 dense layers:

```python
channels = 32
input_shape = (240, 240, 1)
inputs = Input(input_shape)

x = Conv2D(channels, 11, strides=(1, 1), padding="same")(inputs)

x = Conv2D(channels * 2, 7, strides=(2, 2), padding="same")(x)

x = Conv2D(channels * 4, 5, strides=(2, 2), padding="same")(x)

x = Conv2D(channels * 8, 3, strides=(2, 2), padding="same")(x)

x = Conv2D(channels * 16, 3, strides=(2, 2), padding="same")(x)

x = GlobalAveragePooling2D()(x)

x = Dense(256, use_bias=False, activation="relu")(x)
x = Dense(1, use_bias=False)(x)
model = Model(inputs=inputs, outputs=x)
```
E. Computational Cost and Memory

We compute the theoretical computational cost and empirical memory usage of the DISCO spherical convolution, contrasting it to the most efficient alternative equivariant spherical convolutional layers of an axisymmetric convolution computed via Fourier space (Esteves et al., 2018; Cobb et al., 2021). Note that alternative equivariant spherical convolutional layers that are more expressive are considerably more costly, both in terms of compute and memory (Cohen et al., 2018; Kondor et al., 2018; Cobb et al., 2021).

Without loss of generality we consider 1 layer, 1 channel, and a batch size of 1. For the DISCO spherical convolutional layer we consider axisymmetric filters with 4 nodes and $\theta_{\text{cutoff}} = 3\pi/L$. For the axisymmetric harmonic layer we parameterize the filters with 10 nodes in harmonic space and use linear interpolation, similar to Esteves et al. (2018).

For the DISCO layer we compute computational floating point (FLOP) cost via analytic calculations of the sparse tensor products and linear interpolation. For memory usage we explicitly construct the sparse $\Psi_{ji}$ tensor and then count the number of non-zero values; hence, the evaluation of memory usage is empirical rather than purely theoretical. For the axisymmetric harmonic layer we compute computational cost and memory usage via analytical calculations of the tensor products used in the precompute-based spherical harmonic transforms, convolutions and linear interpolation.

Theoretical computational cost and empirical memory usage results are presented in Fig. 6.4. For the DISCO spherical convolution both computational costs and memory usage scales linearly in the number of spherical pixels, i.e. as $O(N) = O(L^2)$. Furthermore, not only does the computational cost of the DISCO spherical convolution scale more favorably than the most efficient alternative equivariant spherical convolution layer but its absolute cost is also considerably lower. For 4k spherical images we achieve a saving in computational cost of $10^9$ and a saving in memory usage of $10^4$.

F. Custom sparse gradients

Since automatic differentiation of sparse tensor operations that include learnable parameters, as in the DISCO convolution, is not supported in either TensorFlow or PyTorch we implement custom sparse gradients.
Consider the sparse tensor representation of the DISCO convolution

\[ h_{dj} = \sum_i \Psi_{ji} f_{di}, \quad (54) \]

where, for completeness, we also indicate the data instance by index \( d \). Since CNN architectures typically include many convolutional layers, with outputs of one layer providing the inputs for another, to compute gradients of the loss function with respect to the parameters of the model it is necessary to compute the gradient of the convolution with respect to both input and dense filter tensor:

\[ \frac{\partial h_{dj}}{\partial f_{d'i'}} = \Psi_{j'i'} \delta_{dd'}; \quad \frac{\partial h_{dj}}{\partial \Psi_{j'i'}} = f_{d'i'} \delta_{jj'}. \quad (55) \]

Pointwise activation functions \( \sigma(\cdot) \) are typically included in CNN architectures to introduce non-linearity in an equivariant manner. Gradients can be traced through activation functions by

\[
\begin{align*}
    \left( \frac{\partial \sigma}{\partial h} \frac{\partial h}{\partial f} \right)_{d'i'} &= \sum_{dj} \frac{\partial \sigma}{\partial h_{dj}} \frac{\partial f_{d'i'}}{\partial f_{d'i'}} = \sum_j \frac{\partial \sigma}{\partial h_{d'j}} \Psi_{j'i'}; \\
    \left( \frac{\partial \sigma}{\partial h} \frac{\partial h}{\partial \Psi} \right)_{j'i'} &= \sum_{dj} \frac{\partial \sigma}{\partial h_{dj}} \frac{\partial h_{d'i'}}{\partial \Psi_{j'i'}} = \sum_d \frac{\partial \sigma}{\partial h_{d'j}} f_{d'i'}. 
\end{align*}

(56) (57)
\]

Equation 56 is a sparse-dense tensor multiplication but with the \( \Psi \) matrix transposed in input and output coordinates. Thus, we can use a similar method to Section 6.4.2 to convert to a compressed sparse tensor multiplication. Note also that we need only update the non-zero values of \( \Psi_{ji} \), for which we know the corresponding indices. Hence, we need only compute Equation 57 for the sparse set of indices of \( j'i' \). Pseudo code of our custom sparse gradient implementation is shown in Algorithm 1.

**Algorithm 1** Function to compute custom sparse gradients in TensorFlow.

**Input:** The upstream gradient \( \frac{\partial \sigma}{\partial h_d} \), input signal \( f_d \) and sparse filter tensors \( \Psi \), \( \Psi^T \)

**Output:** Activation gradient \((\frac{\partial \sigma}{\partial h} \frac{\partial h}{\partial f})_d \) and sparse kernel tensor gradient \((\frac{\partial \sigma}{\partial h} \frac{\partial h}{\partial \Psi})_d \)

1. \((\frac{\partial \sigma}{\partial h} \frac{\partial h}{\partial f})_d = \text{tf.sparse.sparse_dense_matmul}(\Psi, \frac{\partial \sigma}{\partial h_d})\)
2. \(i', j' = \Psi.\text{indices}\)
3. \(f'_d = \text{tf.gather}(f_d, i')\)
4. \(\frac{\partial \sigma'}{\partial h_d} = \text{tf.gather}(\frac{\partial \sigma}{\partial h_d}, j')\)
5. \(\frac{\partial \sigma}{\partial h} \frac{\partial h}{\partial \Psi} = \sum_d \text{tf.math.multiply}(\frac{\partial \sigma'}{\partial h_d}, f'_d)\)
We validate our implementation of custom sparse gradients presented here against gradients for a dense tensor computed by automatic differentiation (a highly inefficient approach where the dense tensor has many entries set to zero), finding agreement to machine precision.

G. Experiments

G.1. Rotated MNIST on the Sphere

Data. We project the MNIST digits onto the sphere at resolution $L = 1024$, using the same projection as in Cohen et al. (2018). We also standardize the data by subtracting the mean and dividing by the standard deviation (both computed from the entire dataset).

Architecture. We consider two models consisting of 7 convolutional layers, one with DISCO spherical convolutions and one with standard planar convolutions for comparison. Convolutional layers are followed by a global integration over the sphere (Cobb et al., 2021) for the spherical model and a global pooling for the planar model, which are followed by 2 dense layers, where all layers include ReLU activations. As typical for CNNs, we continually reduce the resolution of feature maps and increase the number of channels through the model. The bandlimit and number of channels ($L$, channels) considered for each convolutional layer are as follows when progressing through the model: $(1024, 4)$, $(512, 8)$, $(256, 16)$, $(128, 32)$, $(64, 64)$, $(32, 128)$, $(16, 256)$. For the DISCO model we consider a hybrid model containing DISCO spherical convolutions for early layers that operate at high-resolution and harmonic convolutions for subsequent layers operating at resolutions $L \leq 64$. For both the spherical and planar model we adopt depthwise separable convolutions for channel mixing for layers at resolutions $L \leq 64$ and the standard channel treatment otherwise. For the DISCO spherical convolutions we adopt axisymmetric filters with $\theta_{\text{cutoff}} = 3\pi/L$ and 4 nodes along $\theta$, while for the harmonic convolutions we adopt axisymmetric filters parameterized with 10 nodes in harmonic space, similar to Esteves et al. (2018). For the planar model we adopt $3 \times 3$ convolutional filters. For both models the two proceeding dense layers contain 256 and 10 neurons, respectively. The resulting DISCO model has 449k learnable parameters, while the planar model has 411k. The architectures of the spherical and planar models are closely matched; however, since there are differences in the structure of the spherical and planar layers this results in a slight difference in number of learnable parameters. In any case, this will not induce any difference in the main results of Table 9.

Training. We train for 10 epochs using the ADAM optimizer (Kingma and Ba,
2015), with a learning rate of 0.001 and a batch size of 8.

**G.2. Semantic segmentation: 2D3Ds**

**Data.** The 2D3DS dataset (Armeni et al., 2017) consists of 1413 equirectangular RGB-Depth indoor 360° images, with each pixel belonging to one of 14 classes. We downsample the images to $L = 256$ with bilinear interpolation for the RGB-D images and nearest-neighbor interpolation for the classes. We standardize the RGB-D data by subtracting the mean and dividing by the standard deviation channel-wise. We also experiment with image augmentation via random mirroring, i.e. switching the co-ordinate $\phi \rightarrow -\phi$, which improves performance a little (see Table 11).

**Architecture.** We consider a U-Net style architecture with residual blocks as illustrated in Figure G.1. Convolutional layers are typically composed of depthwise-separable DISCO spherical convolutions. Downsampling layers are DISCO spherical convolutions but with outputs at half the resolution, i.e. $L/2$. For upsampling layers we use transposed DISCO spherical convolutions sampled at double the resolution, i.e. $2L$. We use spherical batch-normalizations and consider similar residual blocks to Jiang et al. (2019). Note in the final layer we consider 14 channels to represent the different classes and also consider 1 invalid channel. We consider models with axisymmetric (DISCO-Axisymmetric), directional-separable (DISCO-Separable) or unconstrained directional (DISCO-Directional) filters for all convolution layers. For axisymmetric filters we set $\theta_{\text{cutoff}} = 3\pi/L$ and 4 nodes along $\theta$ for $L > 64$, else we use standard harmonic convolutions with 10 nodes in harmonic space, similar to Esteves et al. (2018). For separable filters we set $\theta_{\text{cutoff}} = 3\pi/L$ and 4 nodes along $\theta$ and $\phi$. Finally, we also use unconstrained filters where the nodes are a square $3 \times 3$ pixel grid projected onto the sphere. The projection is such that the central node is on the pole at $\theta = 0$ and the edge nodes are at $\theta = \frac{\pi}{L}, \phi \in \{0, \pm\frac{\pi}{2}, \pi\}$ and $\theta = \frac{\sqrt{2}\pi}{L}, \phi \in \{\pm\frac{\pi}{4}, \pm\frac{3\pi}{4}\}$. We use bilinear interpolation on this regular grid to obtain values outside the nodes. For the DISCO-Directional model we also consider the case with the mirror augmentation discussed above (DISCO-Directional-Aug). The models DISCO-Axisymmetric, DISCO-Separable and DISCO-Directional have 1.78M, 1.47M and 1.68M learnable parameters respectively.

**Training.** We train for 120 epochs using the ADAM optimizer (Kingma and Ba, 2015) with a batch size of 8 and learning rate of 0.01 which is reduced on plateau by 0.1 with patience 8. We use a class-wise weighted cross-entropy loss to balance the class examples. We also weight the loss and metrics by the area of each pixel on the sphere. We use the same 3-fold split for cross-validation as in Jiang et al. (2019). Example segmentation results are shown in Figure 6.6.
G.3 Semantic Segmentation: Omni-SYNTHIA

Data. The Omni-SYNTHIA dataset (Ros et al., 2016) consists of 2269 panoramic RGB images of outdoor city sceneries with each pixel belonging to one of 13 classes. We downsample the images to $L = 256$ with bilinear interpolation for the RGB images and nearest-neighbour for the classes. We also experiment with copy-and-paste augmentation (Ghiasi et al., 2021), which is straightforward to implement for our sampling scheme and improves performance a little (see Table 10). This is implemented by choosing a random training image, then choosing a random small object out of pole, traffic sign, pedestrian, or bike and pasting this object on a different training image for both RGB and segmented values. The pasted objects are also translated randomly by $\phi \rightarrow n\pi/L$ which can be performed straightforwardly for our sampling scheme; we also do a random mirroring of the pasted object.

Architecture. We consider the same backbone of a DISCO spherical convolutional residual U-Net architecture as for the 2D3DS data (as in Figure G.1) but with 13 output channels (+1 channel for invalid) and 3 channels for the input. Here we consider only separable directional filters parameterized identically as for 2D3DS (DISCO-Separable). We also consider the case with the copy-and-paste augmentation discussed above (DISCO-Separable-Aug). The resulting model has 1.47M learnable parameters.

Training. We train for 70 epochs using the ADAM optimizer Kingma and Ba (2015) with a batch size of 8 and learning rate of 0.01, which is reduced on plateau by 0.1 with patience 8. We use a class-wise weighted cross-entropy loss to balance the class examples. We also weight the loss and metrics by the area of each pixel on
the sphere. We use the same training/test split as in Zhang et al. (2019). Example segmentation results are shown in Figure 6.5.

G.4. Depth Estimation: Pano3D on Matterport3D

Data. The Matterport3D dataset (Chang et al., 2017) contains 7907 spherical RGB images from which we predict spherical depth maps. We downsample the RGB-Depth images to $L = 512$ with bilinear interpolation.

Architecture. We consider the same backbone of a DISCO spherical convolutional residual U-Net architecture as for the semantic segmentation problem but adapted for depth estimation. The architecture is similar to Figure G.1 but we replace each ResBlocks simply by a DISCO spherical convolution. All resolutions are doubled to start at resolution $L = 512$ for the input. We consider 3 input channels and 1 output channel (depth). The number of channels for the down leg of the model are $(64, 64, 128, 256, 512)$, which is reversed for the up leg. For the upsampling layers we increase the bandlimit by 2 using nearest neighbor interpolation. We adopt this upsampling rather than transposed convolutions to simplify transfer learning (discussed below) from planar to spherical convolutions. Due to the higher resolution we use smaller batch sizes, which means batch-normalization is unsuitable and so we therefore use group-normalization (Wu and He, 2018) with 16 groups. Here we apply a group-normalization and ReLU after every DISCO spherical convolution. We consider only unconstrained directional filters (DISCO-Directional), adopting the same parameterization as for the segmentation problems, as described above. The resulting model has 658k learnable parameters.

Training. We use an area weighted $\ell_1$ loss and the same area weighted metrics as in Albanis et al. (2021). We use the same train/test/validation split as in Albanis et al. (2021). For faster training we use transfer learning by first training the above architecture with planar convolution layers with $3 \times 3$ filters, for 60 epochs using the ADAM optimizer (Kingma and Ba, 2015), with a batch size of 2 and learning rate of 0.0001. The $3 \times 3$ filter weights are then easily transferred to the filters projected on the sphere for the unconstrained directional DISCO filters. After the weight transfer, the DISCO model is then trained for a further 60 epochs with the same hyperparameters. Example depth estimation results are shown in Figure 6.7.


**List of Figures**

2.1 Illustration of a simple two-layer neural network with 3 and 2 neurons at the first and second layer. .................................................. 15
2.2 Fitting a quadratic with a neural network. ................................. 16
2.3 Depiction of different types of gradient descent in deep learning. The contours show the loss curve as a function of the neural network weights $w$ for the whole dataset. The arrows show the gradient steps taken. ................................................................. 19
2.4 Illustration of the Bias-Variance tradeoff. ..................................... 21
2.5 Depiction of a convolution with a localized kernel. ......................... 22
2.6 Convolving an image with different types of filters. Horizontal edge filter (Left), blurring filter (middle), sharpening filter (Right). ........ 23
2.7 Illustration of translational equivariance. ..................................... 24
2.8 Standard convolutional layer. .................................................. 25
2.9 Depth-wise separable convolutional layer. ................................. 25
2.10 VGG16 architecture (Simonyan and Zisserman, 2014) with various downsampling layers and a mapping to 1000 probabilities of each class. 26
2.11 CNN architecture techniques. .................................................. 28
2.12 Illustration of stability mapped onto the feature space (diagram made by the author of McEwen et al. (2022)) ................................. 31
2.13 Illustration of group equivariance for a group convolution with $q \in p4$. Each of the four output images corresponds to a convolution with the same kernel but with the kernel rotated by 0, 90, 180 and 270 degrees. 33
2.14 Examples of spherical data (Defferrard et al., 2020). ..................... 34
2.15 ZYZ Euler rotation convention. .............................................. 35
3.1 Illustration of the radiotherapy pathway. CT images taken from Barrett et al. (2009). ................................................................. 38
3.2 Cross sections according to interaction type and photon energies from Mayles et al. (2007). Incoherent scattering here refers to Compton scattering from “free” electrons while Coherent scattering is from atomic electrons. Triplet production is a special case of Pair production where the interaction happens in the electric field of an atomic electron instead of the nucleus, consequently the atomic electron is ejected while an electron-positron pair is created.

3.3 Illustrations of the difference between kerma and dose.

3.4 Schematic of what is varied in dose measurements.

3.5 Relative dose distributions.

3.6 Illustration of DTA and difference between two dose distributions, e.g., a reference and measured dose. This may also be extended to 2D and 3D dose.

3.7 Energy transitions in an organic scintillator (Knoll, 2010).

3.8 Blue plastic scintillator irradiated by a 6MV X-ray beam.

3.9 Illustration of visible light photons emitted when X-rays are fired into a phantom.

3.10 Illustration of Monte-Carlo simulation of dose (edited version from Mayles et al. (2007)).

3.11 Schematic of a Geant4 simulation process.

4.1 Relative dose distributions of the CyberKnife, measured using a microDiamond diode (Chalkley and Heyes, 2014).

4.2 Experimental setup.

4.3 Photos of experimental setup.

4.4 Scintillator specifications (Nuvia, 2019).

4.5 Projected light is composed of Cherenkov from all 3D plastic components, and scintillation from the 2D scintillator sheet.

4.6 Beam on vs beam off images (green channel).

4.7 Background X-ray scatter as a function of exposure time. Error bars shown (but too small to see) as standard deviation over 5 images.

4.8 Denoising an image via pixel-wise median over 5 images and median filtered 5 times. Note that the distances shown on the axes is at the scintillator sheet.

4.9 Profile of raw and denoised signal.
4.10 Example images for field size 30mm showing the red, green and blue channels. Note we have normalized the images here according to the exposure time calibration curve (see section 4.2.4). Figure (c) is the result of subtracting (b) from (a). Since some pixel values become negative after subtraction, we have clipped values to be between 0 and infinity so that the dose is always positive.

4.11 Exposure calibration curve.

4.12 Dose vs pixel intensities.

4.13 Image simulation algorithm.

4.14 Photon trajectory cuts.

4.15 Scintillation images with no Cherenkov.

4.16 Output factor extracted from simulated scintillation images compared with the microDiamond measurements.

4.17 Comparison of PDD and profile extracted from simulated scintillation images and measured with the microDiamond. The dose and errors are obtained by averaging over 8 images.

4.18 Comparison of simulated and measured Cherenkov images.

4.19 PDDs after Cherenkov subtraction for different collimator sizes. We have normalised by dividing by the maximum. Since the signals are noisy, we have used the Savitzky-Golay filter to estimate the maximum.

4.20 Profiles after Cherenkov subtraction for different collimator sizes. We have normalised by dividing by the maximum. Since the signals are noisy, we have used the Savitzky-Golay filter to estimate the maximum.

4.21 Output factors after Cherenkov subtraction at 15mm depth.

4.22 Relative profiles for different field sizes in the green channel. Figure (c) are the profiles when the Cherenkov profiles (b) is subtracted from the scintillation+Cherenkov profiles (a).

5.1 Experimental setup for extraction of measured data (from section 4), used for comparison with CNN prediction.

5.2 Randomly simulated scintillation+Cherenkov images.

5.3 Flow chart of supervised learning approach.

5.4 Model for predicting PDDs.

5.5 Flow chart for predicting Relative output factors.

5.6 Example simulated training images for predicting energies.

5.7 Example simulated training images for predicting scintillation ratios. Note that we have randomly varied the beam intensities which means images will have different noise levels as shown in these images.
5.8 Difference histogram of PDDs on validation set. Here the whole validation set is used and we have counted the difference for each depth on the histogram. $\mu = \text{fitted mean, } \sigma = \text{fitted standard deviation}$. 94

5.9 Comparison of PDD predictions for a CNN and Manual method using a Gamma criterion of 3mm/3%. 96

5.10 Histograms showing the difference between true and predicted OFs using the 24 layer ResNet model for different collimator sizes on the simulated validation data. 97

5.11 Predicted output factors using CNNs, error bars denote standard deviation over 5 different images. 99

5.12 Difference histograms with a Gaussian fit for mean and standard deviation ($\mu, \sigma$). 100

5.13 Variable predictions. 101

5.14 Performance as a function of noise levels. 103

5.15 Performance as a function of the scintillation ratio. 103

5.16 CNN feature maps at different layers given a measured input image. Due to the downsampling layers in the CNN, the resolution decreases with the layer number. 104

5.17 Each point in these plots represents a vector of features of an image (extracted from the neural network) in our simulated and measured datasets. But this vector is high dimensional, therefore we project them onto 2-D using t-SNE so that we are able to visualise them. We colour code the data points according to different variables to see if the features are separated according to these variables. The axes here have arbitrary units since they are just features learned by a neural network. 106

6.1 Spherical CNN categorization. The blue spot represents a localised kernel when rotated to two different positions. 110

6.2 SO(3) and SO(3)/SO(2) rotations of a localised kernel. 116

6.3 Illustration of SO(3) equivariance of a spherical convolution with an axi-symmetric kernel. Here the rotation $Q \in \text{SO}(3)$ is done before or after the convolution. The signals here are illustrated with a sphere ($S^2$) which is isomorphic to SO(3)/SO(2). 118

6.4 Comparison of floating point operations (FLOPs) and memory usage of the DISCO spherical convolution and the most efficient alternative equivariant spherical convolution (an axisymmetric harmonic convolution)(Esteves et al., 2018; Cobb et al., 2021). 123
6.5 Example predictions for semantic segmentation of Omni-SYNTHIA data. Here we have used a Mollweide projection in order to see the whole surface of the sphere. ................................. 127
6.6 Example predictions for semantic segmentation of 2D3DS data. Here we have used a Mollweide projection in order to see the whole surface of the sphere. ........................................ 128
6.7 Example predictions for depth estimation of Pano3D data (depth plotted in meters). Here we have used a Mollweide projection in order to see the whole surface of the sphere ..................... 130
A.1 Radiation-induced damage in both scintillating and non-scintillating plastic materials. This damage manifests as a temporary green discoloration that persists for several hours before fading away. Notably, the discoloration is only visible when illuminated with light, effectively functioning as a green filter and assuming the shape of the deposited radiation dose ........................................ 132
B.1 Geant4 simulations of ion chamber in a water phantom compared with measured values from microDiamond™ (Chalkley and Heyes, 2014). (a) PDD - 15mm field size. (b) Profile - 15mm field size at 50mm depth. (c) OF - at 15mm depth. ........................................ 133
B.2 Depth of field in Geant4 image simulation ................................... 134
C.1 Simulation parameter distributions for PDD data. ......................... 135
C.2 Simulation parameter distributions for OF data. ......................... 136
C.3 Simulation parameter distributions for OF data. ......................... 137
G.1 DISCO spherical convolutional residual U-Net architecture. ........... 148

List of Tables

1 Physical characteristics of a plastic scintillator (with a vinyltoluene base) compared to water (Beddar et al., 1992b). ............................. 47
2 Shutter speed of the camera used for each collimator size. ............... 60
3 Noise to signal with different methods of denoising. The true signal is obtained via fitting with a 3-degree polynomial on a sliding window of size 20 pixels (Savitzky-Golay filter). The noise to signal for a given pixel value is then (True signal-Measured signal)/(True signal) i.e. the fractional difference between the true and measured signal. The mean is taken over all pixel values within over 0.05 of the max pixel value. .................................................... 64
PDD differences between scintillation and microDiamond measurements. The reference PDD and measured PDD are normalised by their maximums. Since the signals are noisy, we have used a curve fit to estimate the maximum using a 3-degree polynomial with a sliding window of size 20 pixels (Savitzky-Golay filter). The difference is then between the normalised PDDs. Here we give the MAD over all points on the PDD curve and the maximum absolute difference. We also include Gamma passing rates for the green sheet as it roughly follows the reference curves in Figure 4.19 while the blue sheet does not.

Profile differences between scintillation and microDiamond measurements. Similar to the PDD metrics, the reference profile and measured profile are normalised by their fitted maximums using the Savitzky-Golay filter. The difference is then between the normalised profiles. Here we give the MAD over all points on the profile curves and the maximum absolute difference.

Performance comparison of CNN vs. manual method for reconstructing the PDD. The reference PDD and predicted PDD are normalised by their maximums (so that the maximum is 1). The difference is then between the normalised PDDs. Here we give the MAD over all points on the PDD curve. We also give the Gamma passing rate as a percentage for criterions 3mm/3% and 2mm/2%.

Fitted mean ($\mu$) and standard deviation ($\sigma$) to histogram of differences when comparing true and predicted output factors. The best $\mu$ for each collimator size is highlighted as bold. Note that the predicted and true relative OFs are normalised to be a percentage of the highest field size at 60mm. The difference is then taken between the normalised predicted/true OF curves.

Mean relative equivariance error expressed as a percentage (i.e. $100 \times \epsilon$) for axisymmetric and directional filters (best and worst cases are shown, where worst is shown in gray inside brackets).

Results for spherical MNIST showing the accuracy of predictions for each model as a percentage. The columns represent if the training/test set has been rotated (R) or not (NR).
Results for Omni-SYNTHIA showing the mean intersection over union (mIoU) and mean accuracy (mAcc) both as a percentage. The mean is taken over all categories, e.g. the IoU is computed for each category over the whole dataset, the mean is then computed using all those IoUs. For comparison we also include previous results from different types of spherical CNNs as well as the effective resolution used, $\tilde{L}$. 

Results for 2D3DS showing the mean intersection over union (mIoU) and mean accuracy (mAcc) both as a percentage. For comparison we also include previous results from different types of spherical CNNs as well as the effective resolution used, $\tilde{L}$. 

Results for Pano3D. To measure the difference in predicted vs true depths we have chosen similar metrics to Albanis et al. (2021). The range of depths are measured between 0 to 8 metres and so the error is a comparison between true and predicted depth in metres. For relative errors, it is measured as a fractional difference. To avoid skewing of mean errors, we have chosen to weight the error of each pixel by the area of each pixel. This is denoted by the $w$. The error metrics chosen are the root mean square error (RMSE), root mean square logarithmic error (RMSLE), absolute relative error (AbsRel) and square relative error (SqRel). We also include depth accuracy metrics similar to Albanis et al. (2021), given by $\delta_{1.05}^{\text{ico}}$ which translates to the percentage of predicted pixels that are within 1.05 times the value of the true pixels, and similarly for the rest of the $\delta_{x}^{\text{ico}}$. 