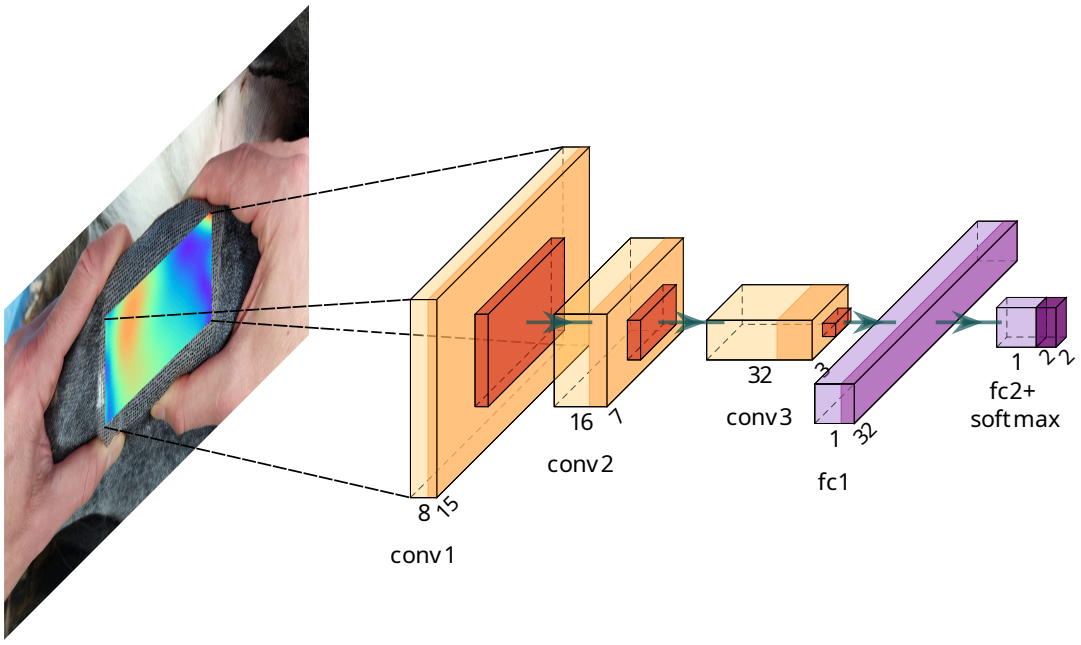
CHALMERS
UNIVERSITY OF TECHNOLOGY



Artificial intelligence for diagnosing knee ligament injuries

CNN for analysing deformation in the knee joint during clinical examination

Master's thesis in Biomedical Engineering

ALICE NILSSON
LINN SÖDERHOLM

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Artificial intelligence for diagnosing knee ligament injuries

CNN for analysing deformation in the knee joint during clinical examination

ALICE NILSSON
LINN SÖDERHOLM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Computer Vision Group
Chalmers University of Technology
Gothenburg, Sweden 2022

Artificial intelligence for diagnosing knee ligament injuries
CNN for analysing deformation in the knee joint during clinical examination
ALICE NILSSON
LINN SÖDERHOLM

Examiner: Fredrik Kahl, Electrical Engineering

Master's Thesis 2022
Department of Electrical Engineering
Computer Vision Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Convolutional neural network for classifying damage maps from the knee joint. Damage map acquired from performing clinical examination of a dog.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Machine learning for diagnosing knee ligament injuries
CNN for analysing deformation in the knee joint during clinical examination
ALICE NILSSON
LINN SÖDERHOLM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis investigates if and how artificial intelligence can be used for diagnosing knee ligament injuries. With current diagnostics for both dogs and humans there is a lack of methods to objectively assess the knee joint. A new technology provides vector fields of movement during clinical examination by recording a patient's knees while care personnel perform diagnostics. This thesis intends to investigate if the vector fields obtained from dogs with cranial cruciate ligament injury can be classified using CNN with the objective of diagnostics. A small dataset consisting of 27 pairs of knees originated from 15 different dogs was provided. An ablation study was designed to evaluate networks, augmentation, transfer learning, learning options and which information from the vector fields were suitable to be used. A machine learning system was implemented mainly using Python and the framework PyTorch. The study demonstrates the importance of using augmentation when provided with a small dataset. One constructed model taking a pair of knees as input achieved an accuracy of 94 % when classifying which of the knees was injured. A second model only using a single knee as input achieved an accuracy of 88 %. This study shows that machine learning has potential to be used for diagnosing knee ligament injuries.

Keywords: Machine learning, CNN, knee joint, dataset, augmentation, transfer learning, image classification, CCL, ligament.

Acknowledgements

Thank you to the examiner Fredrik Kahl for being a great sounding board in the structure of our thesis, guidance on how to best perform our work and excellent insights to a wide and deep knowledge in machine learning and image processing.

Thank you to Martin Fagerström for expert knowledge in material and computational mechanics required to perform parts of this work and for curiosity and insight to new methods and ideas to improve the work.

We would also like to thank Eric Harmin Senorski and Kristian Samuelsson for encouragement and commitment as well as expert knowledge on the medical aspect of this project.

A special thank you to all physiotherapists at SportRehab and veterinarians at Blå Stjärnans animal hospital for believing in our project and for contribution to data collection.

Alice Nilsson & Linn Söderholm, Gothenburg, May 2022

Contents

Glossary

Ablation study A method to investigate how different components affects the performance of a machine learning system.

ACL (Anterior Cruciate Ligament) Ligament in the human knee joint.

Anterior drawer A clinical examination method to diagnose anterior cruciate ligament in humans.

AUC (Area under the ROC curve) This is the area under the ROC curve and give a measurement of a classifiers performance.

CAM (Class activation map) Maps which part of input data is responded by a neural network to when classifying.

CCL (Cranial Cruciate Ligament) Ligament in the dog knee joint.

Cifar10 A dataset containing images of different types of generic objects.

CNN (Convolution Neural Network) Neural network with at least on convolutional layer.

Cranial drawer A clinical examination method to diagnose cranial cruciate ligament rupture in dogs.

Damage map Matrices containing complete in-plain strain field and displacement field for movement occurring on skin of a knee joint during clinical examination for diagnostics of knee ligament injuries.

GAP (Global Average Pooling) A network layer type performing global average pooling.

Laxity Looseness during movement in joints.

Max-pooling A method used for down-sampling in CNN.

PyTorch Python framework for performing machine learning.

ReLU (Rectified Liniear Unit) An activation function.

ROC (Receiver operating characteristics) Plot illustrating the diagnostic ability of a classifier.

ROI Region of interest.

Sensitivity A performance measurement of correctly identifying positive tests.

SGD (Stochastic Gradient Descent) An optimiser used to update weights of network.

Softmax A function to normalise the output from a CNN to a probability distribution.

Specificity A performance measurement of correctly identifying negative tests.

Strain Relative displacement.

Supraphysiological Greater than normally found in the body.

List of Figures

List of Tables

1

Introduction

With current diagnostics of knee ligament injuries there is a lack of methods to objectively assess the laxity of the knee joint. Finding a solution for this problem was researched in a bachelor thesis during the spring of 2020 [1]. The bachelor thesis suggests a method that involves recording of a patients knees during diagnostic tests performed by medical personnel and use images analysis to follow the movement during tests. The movement occurring on the outside of the knee was used for an objective measurement of the laxity as complement to the clinicians subjective assessment. The new technology provides vector fields of movement during clinical examination and can be used to assess all ligaments in the knee joint.

The vector fields are currently illustrated as heat maps for manual interpretation of the laxity, see Figure 1.2. In this thesis it will be researched if artificial intelligence can be used for automating the interpretation of the vector fields from the new technology. Artificial intelligence is currently being investigated in a broad range of applications for clinical diagnostics.

1.1 Clinical background

The knee joint is often described as one of the most advanced joints in the body. For this reason, the knee is also one of the most injured joints [2]. Ligament injuries in knees often present in a distinguishing way allowing higher laxity (looseness during movement) in injured knee joints. Diagnostics by clinical examination is performed by medical personnel to identify increased laxity by firmly provoking excessive movement that would indicate an injury by pulling or pushing the lower leg from the upper leg [3]. The diagnostic tests include a series of tests for evaluating different ligaments and the tests are usually done on both knees for comparison [4]. One test to identify injury to the anterior cruciate ligament (ACL) is the anterior drawer test. The assessment is based on the test executioner's perception which is often dependent on experience [5]. Diagnostics of these injuries can be problematic due to individual differences in natural laxity and subjective assessment.

For dogs one of the most common reasons for limping is an injury to the cranial cruciate ligament (CCL). The clinical examination method for this injury is the cranial drawer test [6] which is similar to the anterior drawer test for humans. The

difficulties with this test are also alike the ones for humans, the test is hard to interpret due to different experience between veterinarians and the various laxity for different breeds but also individual difference for all dogs.

1.2 Current research

A clinical study is performed by Chalmers University of Technology and University of Gothenburg to evaluate the technology. The purpose of the study is to collect recordings from human subjects with knee ligament injuries confirmed using MRI and/or clinical investigation. For each subject included in the study one analysis for each knee and each clinical test performed is collected with related meta data such as age, weight and date of test and injury is collected.

The technology is under development by the start-up company Kneedly AB and is further referred to as the Kneedly method. The company is currently doing a study in collaboration with veterinary clinics to investigate the technology's ability to be used on dogs as a tool for diagnostic aid for veterinarians. Recordings of the cranial drawer test is collected together with data such as diagnose, breed, and weight. The Kneedly method is used on the recordings. The collected data from this study will be available to this thesis.

The vector fields provided from the Kneedly method is further referred to as damage maps. The process of collecting damage maps from clinical tests are presented in Figure 1.1.

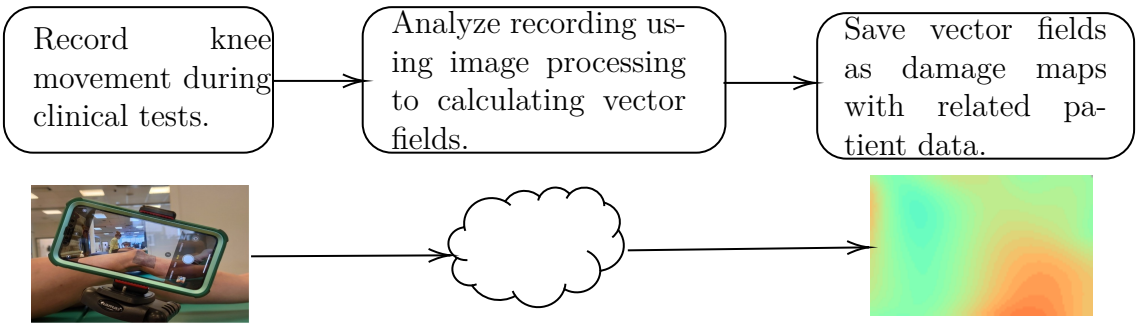


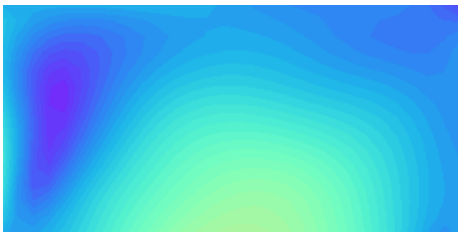
Figure 1.1: Process for collecting damage maps.

The contents of the damage maps are the complete in-plain strain field and displacement field. These fields are represented by matrices of size 15×30. Content of the damage maps is compiled in Table 1.1.

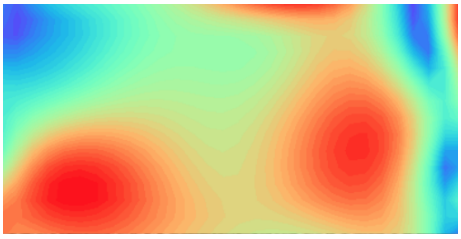
Table 1.1: Contents of damage map

Property:	**Data type:**
Displacement in x-direction	2D-vector matrix [pixels]
Displacement in y-direction	2D-vector matrix [pixels]
First principal strain	2D-vector matrix [unit less]
Horizontal strain	2D-vector matrix [unit less]
Vertical strain	2D-vector matrix [unit less]
Shear strain	2D-vector matrix [unit less]
Maximum shear strain	2D-vector matrix [unit less]

Assessing the damage maps is manually done by plotting the first principal and/or the maximum shear strain for a patient's both knees. The first principal strain was found to be the most illustratively measurement for manual interpretation of human knees [1]. The maximum shear strain is the most illustratively measurement for manual interpretation of dog knees. Figure 1.2 show the first maximum shear strain from damage maps of a dogs both knees. This dog has an injury on the right CCL which results in higher strains indicate by more red colour.



(a) Damage map of left healthy knee.



(b) Damage map of right injured knee.

Figure 1.2: Heat map of maximum shear strain from a dog with a CCL injury on the right knee.

1.3 Aim

The aim of this thesis is to develop a system that can classify damage maps to identify healthy and injured knees. The classification will be based on machine learning using an annotated dataset. The work includes preparing the dataset and modelling of various networks and options for training to achieve high accuracy.

1.4 Specification of issue under investigation

To fulfil the aim of this thesis the following questions will be investigated and answered:

- Accuracy - How well does a convolutional neural network classify damage maps with the aim of providing a diagnosis?
- Model - Which neural network (in terms of architecture, previous training, etc.) is suitable for classification?
- Input fields - Which information in the datasets are suitable as input? (Should a patient's both knees be used as input, which vector fields of the damage maps should be used, etc?)
- Standardisation - How should the damage maps be standardised to be suitable as input for machine learning?
- Augmentation - What are suitable methods to extend the dataset?

1.5 Limitations

- Only human patients with one injured and one healthy knee will be included in the dataset due to the ethical approval for conducting the study from where the data is collected.
- Only dogs with one healthy and one injured knee with tests performed under anaesthesia will be included in the dataset due to the methodology of study from where the data is collected.
- The damage maps are expected to have adequate accuracy and will not be investigated in this thesis.
- The labelled diagnoses corresponding to damage maps in the dataset are assumed to be correct.

2

Theory

With the goal of implementing intelligence to machines artificial neural networks mimic biological neural networks in the central system of the human brain. Artificial neural networks (ANN) were first described in articles over 50 years ago and during the last 20 years the topic has been, and still is, heavily researched. One of the many applications and potentials of ANN's are for it to be used in the medical field with diagnostic purpose [7].

2.1 Neural networks

Artificial neural networks used for classification which are further described in this section consists of multiple layers including one input layer, hidden layers and one output layer [7] (see Figure 2.1). Each layer consists of multiple neurons which in turn has weights, biases, and an activation function [8]. Data handled by a network is often referred to as a tensor which is an array that can be of any size and dimension [9]. For networks classifying colour images the input tensor is typically a three-dimensional array [9].

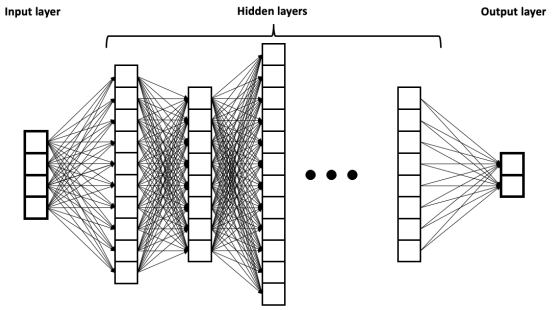


Figure 2.1: Structure of an artificial neural network. From [10]. CC-BY.

2.1.1 The neuron

The human brain consists of 86 billion neurons [11], each of them connected to multiple other neurons. A human neuron is illustrated in Figure 2.2. A neuron receives signals to its dendrites and sends signals through its axon terminal to other neurons [8].

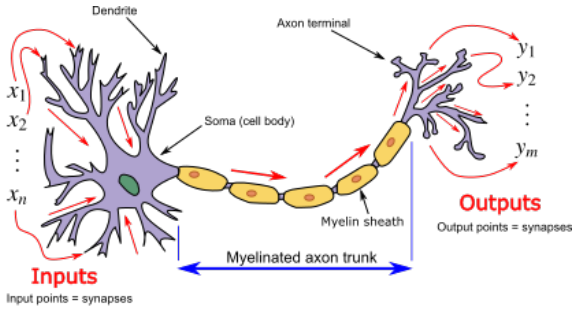


Figure 2.2: Human neuron. From [12] CC BY-SA 3.0.

The human brain is mimicked to perform artificial intelligence by connecting artificial neurons. The artificial neuron, illustrated in Figure 2.3, gets inputs signals, x_n, from input data or from other neurons. Each input signal is weighted by multiplying the input with weights, w_{nj}, and summed together. An activation function, φ, determines which output value, o_j, should be passed on to subsequent neurons [8].

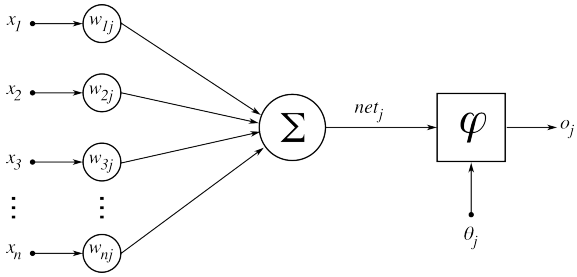


Figure 2.3: Artificial neuron. From [13] CC BY-SA 3.0.

2.1.2 Network layers

Artificial networks consist of neurons structured in layers. The layers consist of neurons structured in different ways to perform the process of training, learning, and classifying the input data [7]. An example of a neural network containing all the different components explained in this section is illustrated in Figure 2.4.

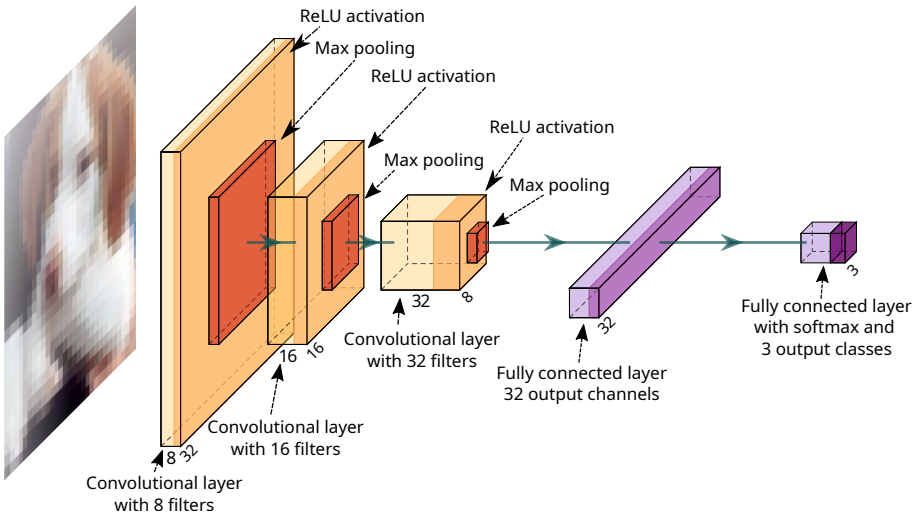


Figure 2.4: Example of a convolutional neural network. The example input image is retrieved from the `Cifar10` dataset [14].

There are different kinds of hidden layers in a neural network including convolutional layers, pooling layers, and fully connected layers [9]. Depending on the layer the tensor is passing through it is shifted in size [9].

Padding can be used to increase the size of the tensor either before the input layer or between layers. A pooling layer can be used to reduce the dimensions of the tensor passing through the network. The most common pooling operation is max pooling. Average pooling is another option that is less widely used. By choosing the maximum value of a region of the tensor and using this value to create a new tensor with smaller height and width a smaller tensor is produced and used to for layers further on in the network [9].

A convolutional layer takes in a tensor, uses a kernel to perform convolution, and produces a new tensor which is passed on to the next layer. The output of a convolutional layer is dependent on the size of the kernel, the stride and optional padding. An activation function is often applied after convolution. One activation function is the rectified linear unit (ReLU) which is used for adding non-linearity [9].

Fully connected layers, also called dense layers, are layers where all neurons are connected to all neurons in the previous layer [9]. The output layer is a layer consisting of neurons which are trained to perform the final classification of the tensor that has been processed from the input layer and through the network via the hidden layers. Fully connected layers are often used provide classification by an

output of the last layer. The outputs from the last fully connected layer are equal to the number of classes. The activation function for the last fully connected layer is often a softmax function [9]. The softmax activation function sums the output to one by dividing each class output by the summation of all the class outputs. The output of a softmax function is hence the probability of an output class [15].

2.1.3 Convolutional neural network

A structure that contains at least one layer performing convolution is called a convolutional neural network (CNN) [16]. The first layers of a CNN tend to learn to distinguish basic features such as identifying edges and colour changes and the later layers learn to identify high-level specifics for the classification problem [17].

A CNN is a type of neural network conventionally used to solve image classification problems [17]. CNNs can achieve very high accuracy for image classification problems when provided large amount of training data and deep network structures [17]. The problem of obtaining these results is often dependent on that there are not enough data to train the network [17]. Input of the same dimension is required for most CNN. Option to obtain this is to use cropping to decrease the dimension of a tensor or to use padding to increase the dimension [18].

2.2 Training of neural networks

Networks are usually initialised with random weights and biases [8]. When training the network the weights, w_{nj} (see Figure 2.3), are updated and some neurons in the network become more important [8]. This is how the network learns. There are several options for training which is outlined in this section.

2.2.1 The backpropagation algorithm

The method of updating weights for improved accuracy is often using the back propagation algorithm. The first step is called feedforward. During this stage a labelled input is passed through the network and the output is computed with the current weights of the network. The output is compared to the correct label of the sample and in this way the prediction error is calculated [8]. A loss function is used to calculate the prediction error. Different loss functions are described in Section 2.2.3.

The second step is using the prediction error and propagate it backwards through the network. Some optimiser is used to update the training parameters to minimise the error of the loss function [8]. Different kinds of optimiser are described in Section 2.2.4.

2.2.2 Learning process

The feedforward and backpropagation steps are redone, and the network is hence trained. A learning rate is determining how much the weights and biases are changed in the direction of the optimiser's result for one round of feedforward and backpropagation [8]. The learning rate can be adjusted when training. A low learning rate is suitable for small datasets [17].

To evaluate the models performance on data unseen during the training process the dataset is split into a training set, a validation set and, a test set [19]. When the model is trained on one dataset is should be evaluated on another dataset. To fairly evaluate a model testing should be done on unseen samples of which the model has not been trained to classify [8].

Overfitting is when the model is well fitted to the training data but performs badly on unseen samples [8]. When training a neural network on small datasets there is an increased risk of the network becoming overfitted and only perform well on the training data [20]. The risk for overfitting can be reduced by using a method called dropout [21]. Dropout is a process where some of the neurons in the network are randomly disabled while training. All parts of the network are hence forced to detect certain features. Hinton et al. investigated this technique and found that using a probability of 50 % that a neuron becomes disabled in the hidden layers and a probability of 20 % that a neuron in the input layer becomes disables while training improved performance [21].

The number of epochs is the number of times the whole dataset is passed through the network during a training process [22]. The number of epochs suitable for a training process is varying. If too few epochs are performed the model cannot be adequately trained while to many epochs can cause overfitting [23]. To identify overfitting a training graph displaying the training loss and validation loss can be used [7]. If the model starts to perform much better on the training data than on the validation data, this is an indication that the model learns to perform well exclusively on the training data. Figure 2.5 illustrates the training process of a model becoming overfitted. By finding the point where the validations loss increases and the test loss continues to be stable, identification of suitable number of epochs is done by looking at the x-axis.

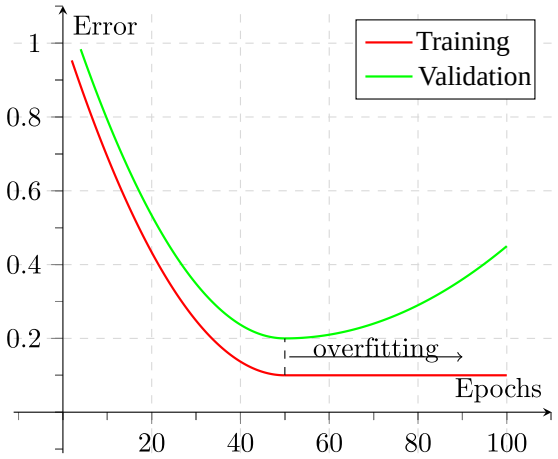


Figure 2.5: Training graph illustrating a neural network trained an excessive number of epochs causing overfitting. Modified from [24]. CC-BY.

The batch size is another parameter that affects the learning process. Batch size refers to the number of samples sent to the network for one iteration during training [22].

2.2.3 Loss functions

Following equations lists a subset of loss functions that can be used. $\boldsymbol{y}$ is the true label and $\boldsymbol{o}$ is the predicted one-hot encoded label from the network. $(\cdot)^{(i)}$ is the i dimension of vector $(\cdot)$.

Least absolute deviations or $\mathcal{L}_1$-loss is defined as,

$$\mathcal{L}_1 = \sum_j |\boldsymbol{y}^{(j)} - \boldsymbol{o}^{(j)}| \text{ [25].} \tag{2.1}$$

Least squared error or $\mathcal{L}_2$-loss is defined as,

$$\mathcal{L}_2 = \sum_j (\boldsymbol{y}^{(j)} - \boldsymbol{o}^{(j)})^2 \text{ [25].} \tag{2.2}$$

Cross entropy loss $\mathcal{L}_{log}$ is defined as,

$$\mathcal{L}_{log} = -\sum_j \boldsymbol{y}^{(j)} \log \sigma(\boldsymbol{o})^{(j)} \tag{2.3}$$

where σ is a probability estimate [25].

2.2.4 Optimisers

To optimise the performance of a neural network the weights are adjusted. A common method is to use gradient descent which updates the model's parameters, θ, by calculating the gradient of an objective function $J(\theta)$ and updating the parameters in the opposite direction of the gradient [26]. Typically, the objective function is a loss function. Various loss functions are described in Section 2.2.3. The learning rate is denoted as η.

The simplest type of gradient descent is batch gradient descent [26] which updates the parameters using the entire dataset,

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta). \tag{2.4}$$

Another option is to update the weights of the model using stochastic gradient descent (SGD) [26]. In contrast to batch gradient descent, SGD updates the parameters for every sample x_i and label y_i,

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta, x_i, y_i). \tag{2.5}$$

Stochastic gradient descent with momentum [26] is another variation of gradient descent. This can reduce oscillation in the stochastic gradient descent function and decrease the time it takes for the function to converge. It is implemented by adding a scaling factor γ of the previous gradient descent vector which is used to update the current gradient descent vector,

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \tag{2.6}$$
$$\theta_{t+1} = \theta_t - v_t. \tag{2.7}$$

A common value for this scaling factor γ is 0.9 [26].

Adam is another algorithm for gradient-based optimisation. This method is generally suitable for problems that are large in terms of training data and network parameters [27].

2.3 Augmentation

One of the challenges of deep learning is the need for large datasets for sufficient accuracy [28]. Data augmentation is used to enlarge the amount of data available for training by producing synthetic variations of the original dataset [29] [30] [31]. This is a common way of increasing robustness [29] and reduce risk of overfitting on a specific dataset for a classifier [30]. It is not possible to conclude what augmentation is suitable for a specific dataset and network structure before testing it in the

intended application [31]. In the following sections various kinds of augmentation methods are presented.

2.3.1 Geometric transformations

Matrix manipulation for the cause of creating augmented data to increase the size of a dataset include various affine transformations [31]. Affine transformations are a type of geometric transformations which is used to perform mirroring, rotation, scaling and shearing of a matrix [32, pp. 101–102].

An affine transformation has the general form,

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} v & w & 1 \end{bmatrix} \boldsymbol{T} = \begin{bmatrix} v & w & 1 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \text{[32, p. 102].} \tag{2.8}$$

By setting the parameters t_{jk} to different values the different transformations can be performed. Table 2.1 lists different configurations of the affine transformation matrix $\boldsymbol{T}$ to perform different transformations.

Table 2.1: Different values of the transformation matrix $\boldsymbol{T}$ to perform different types of transformations [32, p. 102].

Transformation:	**Transformation matrix $\boldsymbol{T}$:**	**Description:**
Rotation	$\begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Rotates with angle θ.
Mirroring	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Flips matrix around x-axis.
Scale	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Changes scales in x and y direction by setting c_x and c_y respectively.
Shear	$\begin{bmatrix} 1 & s_h & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Shear in vertical and horizontal direction by setting s_v and s_h to non-zero.

2.3.2 Contrast and noise manipulation

In addition to the geometrical transformations matrix manipulation can also be performed by using techniques such as adding noise [33]. Augmented images for improving accuracy of classification problems can also be created by increasing the contrast of the images by identifying high and low values and stretching intensity levels [31].

2.4 Transfer learning

Transfer learning is an effective solution for improving performance of a neutral network when training with limited training data [34] [28] [17] [35]. In transfer learning the model is pre-trained on other dataset what is to be classified by the model and then inherit weights, in this way the model does not have to learn all features from the beginning. Instead of initialising the network with e.g. random weights the pre-trained weights are used [34] [28] [35]. ImageNET which consists of $\sim$ 14,000,000 images [36] is an example of dataset often used for transfer learning [37].

In transfer learning fine-tuning of the parameters of an already trained network is performed. Fine-tuning can be done in two main approaches, deep tuning and shallow tuning. In deep-tuning none of the parameters in the layers are frozen between pre-training and fine-tuning. In shallow-tuning some of the layer parameters are frozen and the parameters are only trained to be updated in the remaining layers [35].

Shallow tuning reduces the number of trainable parameters to train on the dataset and improves the performance [34]. The last layers in the model are trained on the dataset for the specific classification problem which optimises the network for the specific dataset [34]. One important aspect to consider for a well balanced transfer learning is the number of layers should be inherited from previous training and which should be fine-tuned for the specific classification problem [34] [28].

2.5 Evaluation of machine learning models

When available dataset is small it can be difficult to divide the dataset into training-, validation- and test sets since it is not affordable to exclude part of the data when training the model. A common method to overcome this problem is to use cross validation. In k-fold cross validation the complete dataset is divided into k subsets. The model is it trained on all but one of the k subsets, the one subset excluded from training is used for validation. The training process is done k number of times and different subsets are held out of the training and used for testing. The mean of the performance from all training processes is used to evaluate the model. Using this approach, all data can be used for training and the model can be evaluated on all the available data [8].

Machine learning models can be evaluated using several measurements. These measures are often derived from the parameters in a confusion matrix illustrated by Table 2.2 [38]. The matrix contains predicted classes compared to the true classes. True positive (T_p) is the number of persons with an injury that are correctly predicted and false negative (F_n) is the number of persons with a disorder which are incorrectly predicted as not having a disorder [39]. True negative (T_n) is the number of persons without a disorder correctly predicted and false negative (F_n) is the number of persons without a disorder which are incorrectly predicted as having a

disorder [39].

Table 2.2: Confusion matrix illustrating the comparison measures between predicted and true classes.

		Predicted classes:		
		Positive:	**Negative:**	
True classes:	**Positive:**	True positive (T_p)	False negative (F_n)	Total count positive (C_p)
	Negative:	False positive (F_p)	True negative (T_n)	Total count negative (C_n)

The accuracy of a machine learning model is defined as,

$$\text{Accuracy} = \frac{\text{Correctly predicted}}{\text{Total number of samples}} = \frac{T_p + T_n}{C_p + C_n} \text{ [38].} \tag{2.9}$$

Sensitivity, $P(T_p)$, is defined as,

$$P(T_p) = \frac{T_p}{T_p + F_n} \text{ [39] [38].} \tag{2.10}$$

Specificity, $P(T_n)$, is defined as,

$$P(T_n) = \frac{T_n}{T_n + F_p} \text{ [39] [38].} \tag{2.11}$$

A common method to evaluate how a model performs for different operating points is to analyse the receiver operating characteristic (ROC) and the area below the ROC curve (AUC) [38]. The ROC curve is created by plotting the probability of false positive $P(F_p)$ on the y-axis and the probability of true positive $P(T_p)$ on the x-axis for different operating points [38].

The AUC value is calculated by integrating over the ROC curve. This can be done by using trapezoidal integration [38]. Figure 2.6 illustrates examples of ROC curves. The perfect ROC curve intersects the point (1,1) and has an AUC=1. The curve equal to guessing has an AUC=0.5.

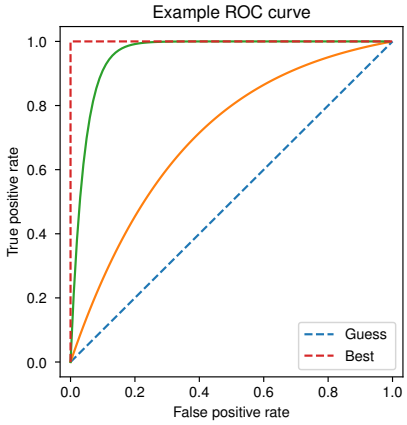


Figure 2.6: Example of a ROC curve.

2.6 Medical machine learning

Machine learning for medical purposes is currently being researched in several fields including skin cancer detection [40], patient–ventilator asynchronous detection [37], cardiovascular images registration [34], electrocardiogram [41] etc. Artificial intelligence has been shown to match the performance of health care professional in several studies [42].

One of the challenges of deep learning in many settings, e.g. the medical field, is the need for large datasets for sufficient accuracy [28] [43]. CNNs requires large amount of data to perform well but large datasets with annotations is within the medical field expensive to obtain due to time-consuming work annotating the samples requiring expertise knowledge [43].

Within the skin cancer field there is a large dataset available online, International Skin Imaging Collaboration (ISIC), with the goal of providing digital images to reduce mortality for skin cancer [44]. Small datasets are problematic for several cases of machine learning for medical purposes. Transfer learning is widely used but is relatively new to medical machine learning but it has shown to contribute to better solutions for classification [34] [37]. Good performance has also been achieved despite small amount of training data. Binary image classification has successfully been done on small datasets of only 200 images by Schouten et al. [45] who used a CNN to classify blood cells into non-malignant leukocytes and other blood cells with good results.

One of the issues with AI within the medical field is that there is communication barrier between AI and health care givers [40]. The diagnostic aid of machine

learning is often seen as a “black box solution” and the lack of interpretability can result in health care personal not trusting the software [46].

2.7 Clinical performance

Sensitivity and specificity presented in Section 2.5 are commonly used to assess the clinical performance of diagnostic methods [47]. The sensitivity and specificity of the anterior drawer test was investigated by Ostowski in a literature study and concluded that the sensitivity is reported as ranging from 18 % to 92 % and specificity from 78 % to 98 % [48]. Other diagnostic methods which use digital assessment methods for diagnostic of human knee ligament injuries are compiled in Table 2.3.

Table 2.3: Sensitivity and specificity of digital diagnostic methods used for diagnosing of knee ligament injuries in human care.

Method/Product:	**Sensitivity:**	**Specificity:**
KiRA (Accelerometer)	59 % [49]	82 % [49]
iPad-application (Pivot)	71 % [49]	71 % [49]
Electromagnetic tracking of pivot shift	79 % [49]	91 % [49]
Magnetic resonance imaging (MRI)	95.5 % [50]	91.7 % [50]

The sensitivity for the cranial drawer test to identify injury to the CCL in dogs is reported to be 60 % for conscious patients and 90 % for sedated patients. The sensitivity is dependent on multiple factors such as experience of veterinarian and the severity of the injury [6]. In one study the cranial drawer test had a specificity of 98 % while performing the examination on sedated dogs [51].

2.8 Class activation maps

Class activation maps (CAM) is used for obtaining understanding of CNNs by illustrating which part of the image the model has responded to for classifying [37]. This is done by mapping the region in an input image that is used, by a trained CNN, to categorise an image in an output class [52] . One method is to perform global average pooling on the feature maps of the convolution and use the features to, with fully connected layer, produce an output and perform classification [52]. Regions of the input images that has been important in the classification process is be mapped by projecting the weights of the output backwards. CAM enables object localisation by highlighting different parts of images. CAM has potential to improve understanding and hence trust of neural networks, an important aspect to consider for image analysis within the medical field [52] . Figure 2.7 illustrates how CAMs are generated from an CNN.

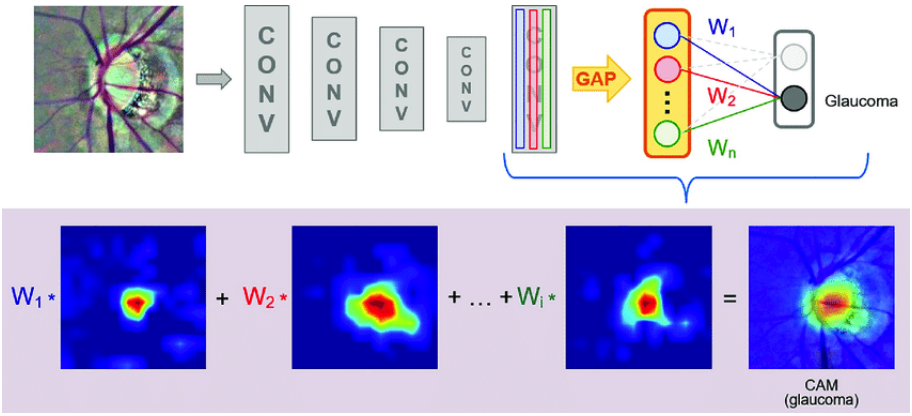


Figure 2.7: CAM to final layer of CNN model. From [53]. CC-BY.

3

Preparation details

This chapter aims to outline the method used for constructing necessary preparations in order to create a foundation for evaluating different settings of network architecture, augmentations, learning options, transfer learning and input fields. The components of the machine learning systems evaluated in this thesis were mainly implemented in Python using the framework PyTorch [54].

A flexible software system was implemented with options to exchange different parts to allow for evaluating and adjusting different options. Figure 3.1 illustrates the architecture of the software. The construction of the solid blocks is described in this chapter and was done as prerequisites for evaluating various machine learning models. The dashed blocks are described in Chapter 4 and was exchanged to test different combinations of augmentations, network architectures and training options.

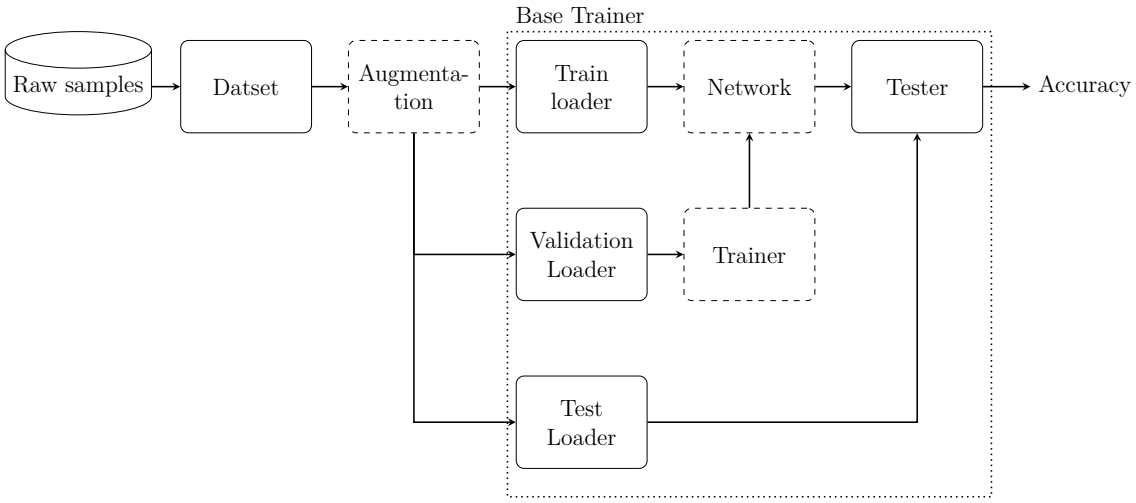


Figure 3.1: Architecture of constructed system. Solid blocks were prerequisites for testing various machine learning models. Dashed blocks were components which were exchanged to obtain various configurations.

3.1 Dataset

Loading the datasets to PyTorch was done by implementing a custom PyTorch dataset-class inheriting from `torch.utils.data.Dataset` named `KneeDataset`.

The provided files which were generated from performing the Kneedly-method were damage maps stored as Numpy uncompressed archives sorted in a directory structure containing a folder for each sample and JSON-formatted files containing meta-data including the label of the injury. The labels were `L` or `R` corresponding to if the left or right knee was injured. The dataset was stored according to the file structure in Figure 3.2.

```
dataset
├── 1
│   ├── L.npz
│   ├── R.npz
│   └── label.json
├── 2
│   ├── L.npz
│   ├── R.npz
│   └── label.json
└── n...
```

Figure 3.2: File structure of dataset.

The region of interest, used to track movement in the video sequence from the diagnostics test, in the Kneedly method did not have a specified size. Producing the dataset for this thesis was done by adding a feature to the Kneedly method to fix the region of interest to 15×30 points as most CNNs are designed with a fixed input size [18].

To allow for the same code to be used to evaluate different input fields, network structures and to reduce bias (see Section 3.1.3) the `KneeDataset` was provided with several options:

- `fields`: Allowed for extracting different lists of matrix fields from the dataset. Created possibility to evaluate which information should be used as input to the system.

- `single_damage_map`: Allowed for setting if only one damage maps from either side should be outputted. This allowed for testing networks which takes a single damage map as input. When only outputting one damage map, left knees were mirrored so that all outputs were alike.

- `equal_injury_probability`: Allows for setting if the dataset should be manipulated so that the probability of injury in right or left knees should be equal. The manipulation was done by transforming knees to mimic the patient's other knee, see Section 3.1.3 for details.

3.1.1 Extending the dataset

Acquiring as much data as possible was crucial due to the limitation of data available. To extend the data a solution applicable of this specific project was to generate multiple samples from one test subject. When performing the clinical test conventionally the test executor performs the test multiple times. When recording the tests for the Kneedly method multiple test sequences was hence often recorded. By editing the recordings to extract several test sequences multiple damage maps was generated.

3.1.2 Available data

The different datasets from the Kneedly method used in the projects are presented in Table 3.1. The `dog_cranial_drawer` was the dataset used for evaluation in this thesis.

Table 3.1: Datasets available acquired from both dog and human subjects.

Name:	Number of samples:	Unique patients:
`dog_cranial_drawer`	27	15
`human_anterior_drawer`	25	14

3.1.3 Reducing bias

The samples collected to majority consisted of injuries on right knees. The `dog_cranial_drawer_test` to 63 % and the `human_anterior_drawer_test` to 72 % consisted of samples with injured right knees. This introduces bias with risk of a neural network, trained on these datasets, classifying right knees as more likely to being injured. To mitigate this possible source of bias an extra step was added in the data loading. When mirroring a sample and changing its label the sample act like an injury on the opposite knee. By randomize the injury and mirror the sample if the actual label of the sample did not match the randomized label the data loader will load 50% samples corresponding to right knee injuries and 50% samples corresponding to left knee injuries. The Listing 3.1 includes the method of which injury probability was equalized.

Listing 3.1: Code for setting the injury probability equal for all classes.

```
# Randomize an injury
new_injury = 'L' if np.random.rand() >= 0.5 else 'R'

# Check if new label is different from the actual
if new_injury != injured_knee:
    temp = right
    # Flip the damage maps
    right = np.fliplr(left)
    left = np.fliplr(temp)
    injured_knee = new_injury
```

3.1.4 Cross validation

Cross validation was implemented since the `dog_cranial_drawer` dataset only consisted of 27 samples and 15 unique patients (see Table 3.1) cross validation was implemented. This was because if the dataset had been split into training set, validation set and test set there would be too few samples in each set. To enable training on all samples as well as ensuring evaluation on all samples for a representable performance the split of the dataset was varying according k-fold cross validation.

To perform cross validation the `dog_cranial_drawer` was split in four partial datasets allowing 4-fold cross validation to be performed. This yielded a total of 12 different combinations when using two of the partial datasets for training, one for validation and one for testing. The mean accuracy was calculated from the test combinations. Table 3.2 lists the partial datasets from splitting the `dog_cranial_drawer`. The split was done ensuring damage maps from the same patients were placed in the same partial dataset to avoid bias when evaluating.

Table 3.2: Split of `dog_cranial_drawer` dataset.

Name of partial set:	Samples:	Unique patients:
dogs_1	7	4
dogs_2	7	4
dogs_3	7	4
dogs_4	6	3

3.2 Augmentation

Augmentation was implemented as PyTorch-transforms. Each transform took one or several parameters to configure the transforms and was chained together to combine different augmentations. The used transforms and parameters are further described in Section 4.2. Listing 3.2 lists the template for constructing transforms.

Listing 3.2: Template for constructing augmentation transforms.

```
class Transform(torch.nn.Module):

    def __init__(self, *args, **kwargs):
        """
        Initialises the transform.
        """
        pass

    def forward(self, matrix):
        """
        Perform transform on input matrix.
        """
        ...
        return transformed_matrix
```

Transforms can be chained together using a `torch.nn.Sequential` layer. This is outlined in Listning 3.3.

Listing 3.3: Chaining of transforms to perform different types of augmentations.

```
transforms = nn.Sequential(
    Transform1(),
    Transform2(),
    ...
    TransformN()
)
```

3.3 Base trainer

The base trainer was the core of the software used for evaluation. The base trainer has support for different augmentation methods, different networks and different network trainers which enabled evaluation of various combinations. Additionally the base trained had support for testing a network after training and provides the accuracy when evaluated on a test set. The base trainer loads samples from the dataset into three groups: train set, validation set and test set. The base trainer was implemented containing abstract methods which controls which optimiser and loss function to use. Listning 3.4 outlines how the base trainer can be used.

Listing 3.4: Base structure for using the base trainer.

```
from dataset import KneeDataset
from network import Network
from trainer import Trainer

transforms = nn.Sequential(
    ...
)

dataset = KneeDataset(<path to dataset>, transforms=transforms)
model = Network().cpu()

trainer = Trainer(model=model, dataset=dataset)
trainer.run()
```

The base trainer was implemented to have the following parameters:

- `model`: Controls which network is trained.
- `dataset`: Controls which dataset is used for training and validation.
- `validation_dataset`: Controls which dataset is used for validation during training.
- `test_dataset`: Controls which dataset is used for testing after training has been done.
- `batch_size`: Sets the batch size.

- `max_epochs`: Sets the maximum number of epochs to train.
- `early_stopping`: Allows for stopping to training process before reaching the amount of `max_epochs`. Sets amount of epochs should be performed since last decrease of validation loss before stopping the training process early.
- `one_hot_encoding`: Sets if target label should be converted from class index to one hot encoding. Required by various loss functions.
- `num_classes`: Sets number of classes which the one hot encoding used for conversion.

The training block illustrated in Figure 3.1 allows for setting different learning options including loss function and optimiser. It was implemented as a single Python class inheriting from the base training class. A template for creating different trainers is outlined in code Listing 3.5.

Listing 3.5: Template for constructing a trainer.

```
class Trainer(BaseTrainer):

    def __optimizer__(self):
        """
        This function should return the optimizer
        which should be used.
        """
        pass

    def __criterion__(self, output, target):
        """
        This function should return the loss
        between the output and the target.
        """
        pass
```

3.4 Network

Networks was constructed and built by inheriting the base class `torch.nn.Module` which is used for all network modules in PyTorch. The different structures in a network was built with functions as `nn.Conv2d` to create a layer performing convolution and `nn.Linear` to create fully connected layers. Network construction also included `nn.Dropout` to use dropout while training, `nn.ReLU` to apply ReLU and `nn.MaxPool2d` to perform max pooling. A template for creating different networks is outlined in Listing 3.6.

Listing 3.6: Template for constructing a network.

```
class Network(nn.Module):

    def __init__(self, channels=2):
        """
        Initialize all layers and adjust them according
        to channels.
        """
        pass

    def forward(self, x):
        ...
        return output
```

3.5 Synthetic data

The small amount of samples in `dog_cranial_drawer` was identified as an important aspect to consider for this project. One of the approaches was to create synthetic data. Construction of synthetic data was done by mimicking a the movement during a clinical test using a simple model of the knee joint and performing finite element method analysis (FEM-analysis). Matlab [55] and the library Calfem [56] was used to create the model.

A grid of 15×30 points was generated and deformed simulating the pulling of a leg which occurs during the cranial drawer test. Synthetic displacement fields was calculated from the deformation to generate samples mimicking the damage maps generated from real real. Only right knees was simulated in these steps. To simulate left knees the grids were vertically mirrored. Which of the knees was simulated to be injured was randomized. Figure 3.3 illustrates the process of generating the synthetic dataset. A total of 1,000 synthetic samples with 50 % right knee injuries and 50% left knee injuries were generated. The samples were collected in a dataset named `synthetic_dog_cranial_drawer` and were mainly used in Chapter 4.3 for evaluating transfer learning.

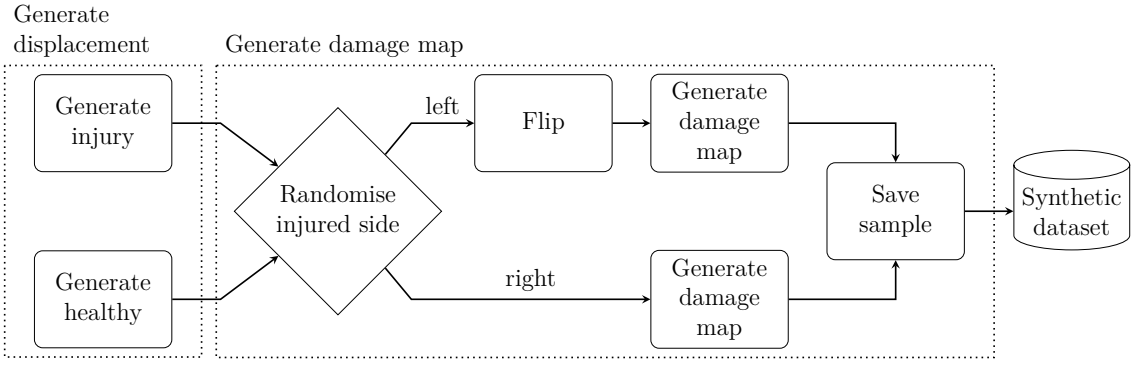
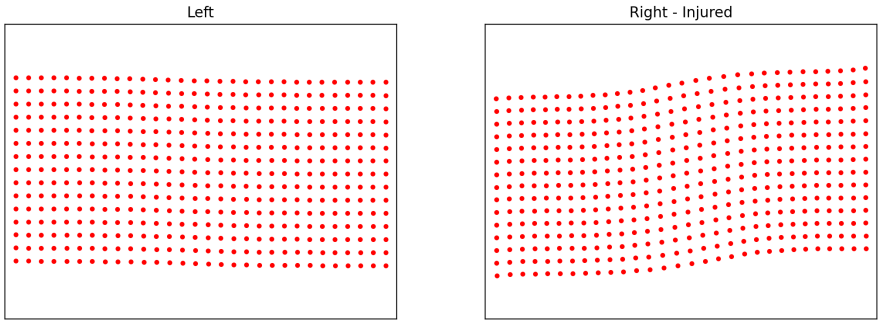
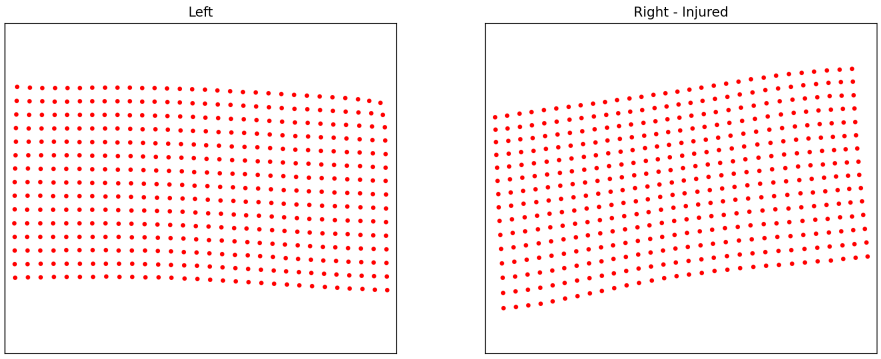


Figure 3.3: Process for generating synthetic data set.

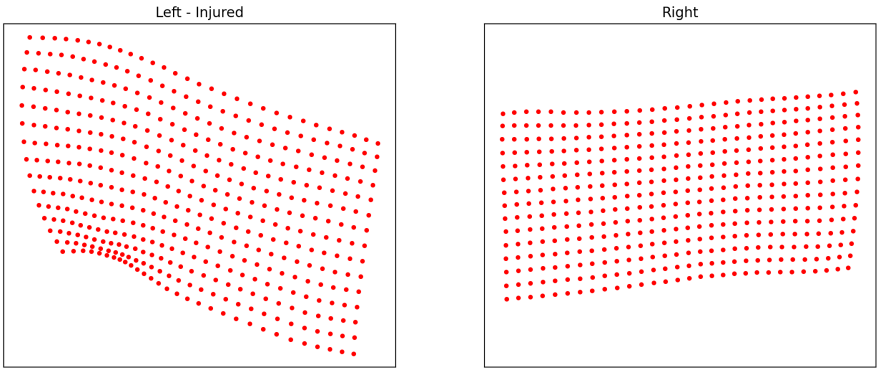
Figure 3.4 illustrates the deformation from a pair of synthetically generated displacements and two examples of samples from the `dog_cranial_drawer` dataset.



(a) Sample from `synthetic_dog_cranial_drawer`.



(b) Sample from `dog_cranial_drawer`.



(c) Sample from `dog_cranial_drawer`.

Figure 3.4: Displacement grid from samples in `synthetic_dog_cranial_drawer` and `dog_cranial_drawer` datasets.

4

Implementation and evaluation of system parts

The approach to solving the problem of classifying the damage maps in the `dog_cranial_drawer` dataset was done using a CNN. An ablation study was performed to evaluate suitable components of the complete system for performing machine learning. Partial evaluation and results were acquired for each component using 4-fold cross validation. The highest yielding option for a component was used for further evaluation of proceeding component evaluations as well as the complete system.

Some of the components are used in partial evaluation before they have been evaluated, for example learning options which are required for evaluation of network and augmentation. A set of default values were used for components which had not been evaluated. The values used are listed for each evaluation.

The methodology enabled part by part evaluation of the components. The order of which the components were evaluated and which partial results were produced from each step are illustrated by Figure 4.1.

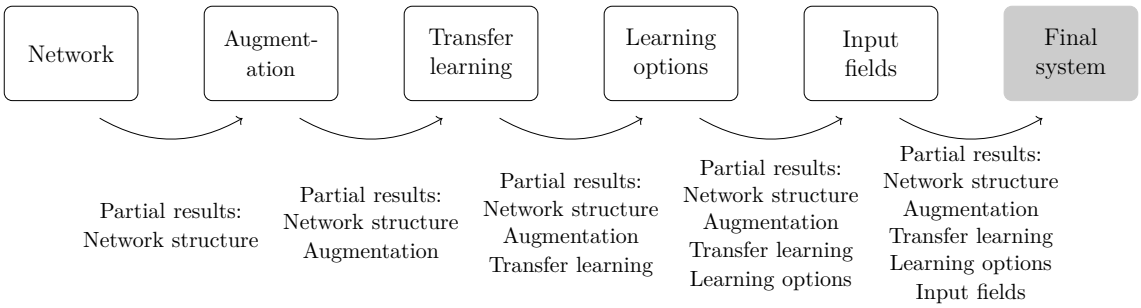


Figure 4.1: Overview of ablation study methodology for this thesis.

4.1 Networks

This section aims to implement and evaluate different neural networks both in terms of architecture and model complexity. Evaluation of which network was suitable was done using the training properties listed in Table 4.1.

Table 4.1: Training properties for evaluating network structure.

Component:	**Description:**	**Details:**
Augmentation	Contrast, $\beta_c = 0.1$ Gaussian noise, $\beta_g = 0.01$ Speckle noise, $b_s = 0.1$ Mirror, $p_m = 0.5$ Rotation, $\sigma_\Theta = 15°$ Scale, $\sigma_{c_x}, \sigma_{c_y} = 0.15$ Shear, $\sigma_{s_v}, \sigma_{s_h} = 0.15$	Default
Learning options	Learning rate: 0.001 Loss function: Cross Entropy Loss Optimiser: SGD with momentum $\gamma = 0.9$ Batch size: 4 Max epochs: 1000 Early stopping: 200	Default
Input field	Maximum shear strain field	Default

4.1.1 Network structures

Three different types of networks in terms of input variations were initially constructed. This was to provide possibility of evaluating if a patient's both knees or a single knee was suitable for classification. Table 4.2 compiles the initial networks constructed and evaluated. The networks were designed to have a relatively low number of trainable parameters due to the small dataset available for updating parameters. The networks were implemented using the code template presented in Listing 3.6. The code for each implemented network is included in Appendix A.

Table 4.2: Properties of constructed neural networks when calculated with one input channel. n-corresponds to number of input channels.

Name:	**Input size:**	**Output size:**	**Trainable parameters:**
`ConcatenatedInputNetwork`	$n \times 30 \times 30$	2 (Left or right injured)	15,235
`SiameseNetwork`	$2 \times n \times 15 \times 30$	2 (Left or right injured)	12,163
`SingleInputNetwork`	$n \times 15 \times 30$	2 (Injured or healthy)	9,058

Figure 4.2 illustrates `ConcatenatedInputNetwork`. This network takes concatenated damage maps from a patient's both knees. The network was configured to have an input size of n-channels × 30 × 30 which corresponds to two damage maps put side by side. The network is configured to have two output classes to classify which of the patient's knee is injured.

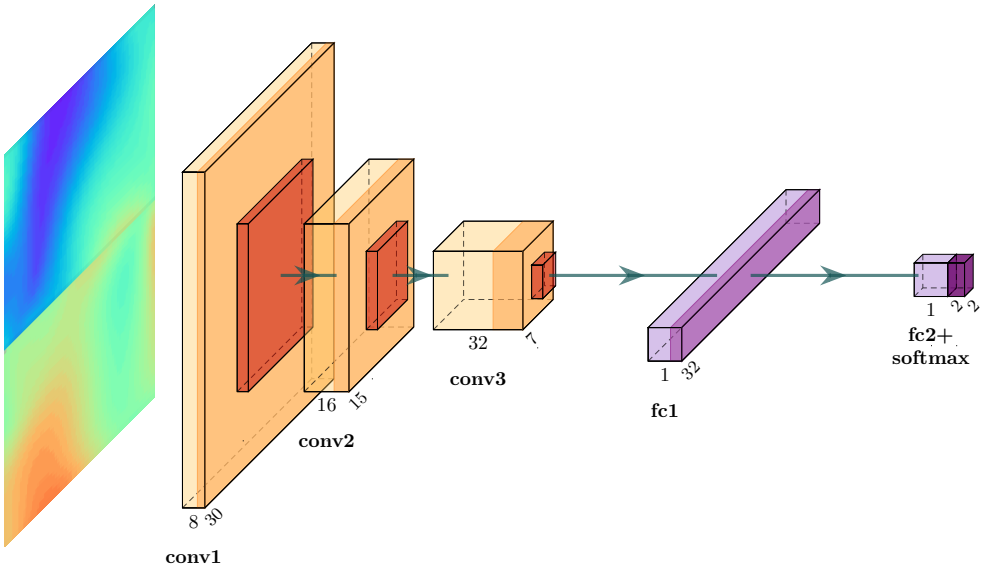


Figure 4.2: Illustration of `ConcatenatedInputNetwork`. Two concatenated damage maps as input.

Figure 4.3 illustrates the neural network `SiameseNetwork`. This network also takes two damage maps from a patient's both knees as input. The network was configured to take two inputs of size n-channels × 15 × 30 which corresponds to two separated damage maps. The three first layers forms branch networks where each branch shares the same weights and are then connected to the same fully connected layer. This network is based on the Siamese network architecture presented in [57]. The output size of the network is two classes corresponding to classification of which of the patient's knees are injured.

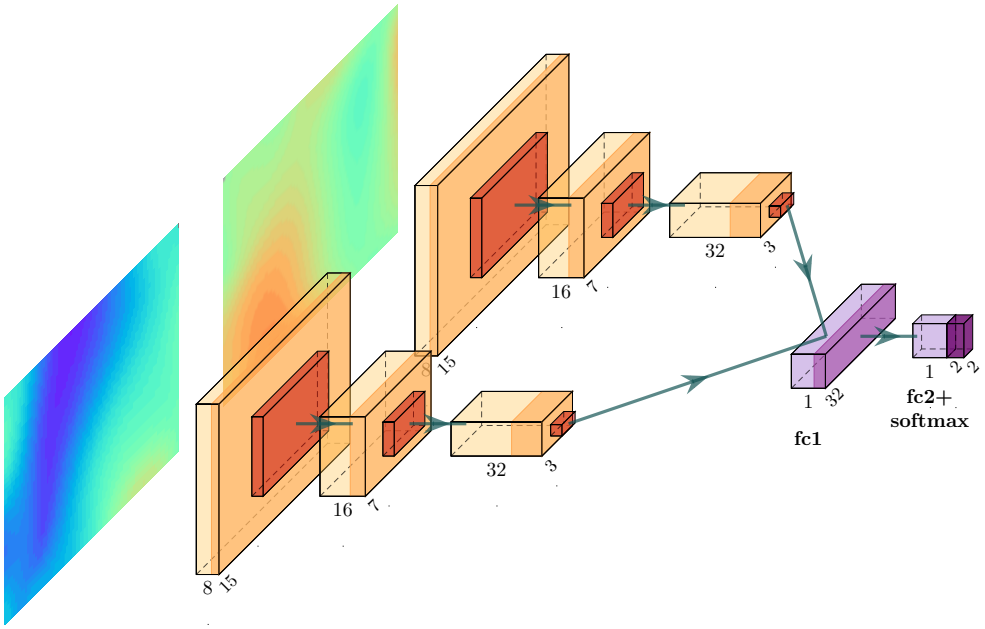


Figure 4.3: Illustration of `SiameseNetwork`. Two separated damage maps as inputs. The first three layers shares the same weights.

Figure 4.4 illustrates the neural network `SingleInputNetwork`. This network takes a single damage map of size n-channels $\times$ 15 $\times$ 30 as input. The output size of the network is two classes in order to determine classify if the input knee is injured or healthy.

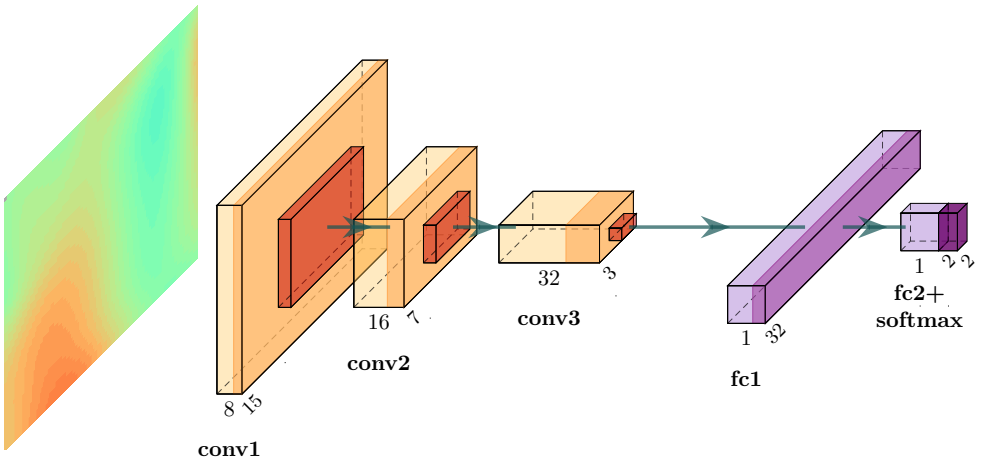


Figure 4.4: Illustration of `SingleInputNetwork`. Single damage map as input.

Table 4.3 lists the test accuracy when the networks were trained according to the training options listed in Table 4.1.

Table 4.3: Accuracy for networks structures using the training properties outlined in Table 4.1.

Network name:	**Test accuracy:**	**AUC:**
`ConcatenatedInputNetwork`	**82 %**	-
`SiameseNetwork`	81 %	-
`SingleInputNetwork`	78 %	0.86

4.1.2 Increased complexity for transfer learning

The constructed networks in Section 4.1.1 have a relative low number of trainable parameters to manage the low number of damage maps available for training. A common method to mitigate lack of training data is to pre-train on other type of data such as generic images. Example of dataset consisting of generic images is the `Cifar10` dataset [14]. The first layers of the networks tend to learn to distinguish the basic features such as identifying edges, therefore `Cifar10` was evaluated to use as pre-training despite the difference from the `dog_cranial_drawer`. The complete methodology of evaluating transfer learning for the system is described in Section 4.3.

The `Cifar10` dataset contains 60,000 RGB images of size $\times 32 \times 32$ [14] and allows for using a more complex network structure. The `ConcatenatedInputNetwork` yielded the highest accuracy from the evaluation performed in Section 4.1.1 and has an input shape of (30×30). Due to good performance and input size similar to the size of images in `Cifar10` the network was chosen to form the basis of a network adjusted for transfer learning.

A new network `ConcatenatedInputNetworkV2` based on `ConcatenatedInputNetwork` was created with more layers by adding an extra fully connected layer. The extra layer entails more parameters and hence a more complex structure. The network is illustrated by Figure 4.5. The network was constructed to have an adjustable amount of input channels as input. The network contains approximately 16,000 trainable parameters depending on the number of input channels.

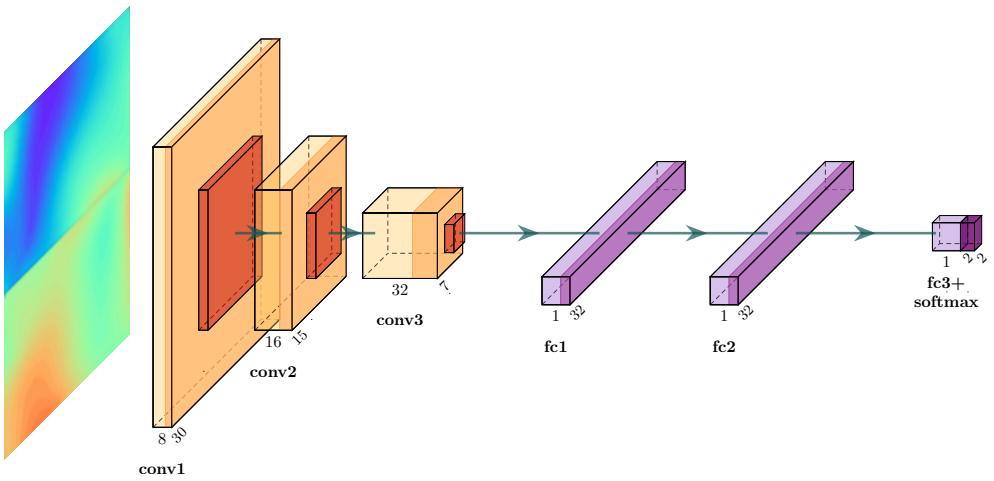


Figure 4.5: Illustration of `ConcatenatedInputNetworkV2`.

4.2 Augmentation

This section describes implemented matrix manipulations for augmentation of the dataset. The augmentations were implemented as transforms in PyTorch according to the method presented in Section 3.2. Evaluation of augmentation was done using the training options listed in Table 4.4.

Table 4.4: Training options for evaluating augmentation methods.

Component:	**Description:**	**Details:**
Network:	`ConcatenatedInputNetwork`	Evaluated in Section 4.1.1
Learning options:	Learning rate: 0.001 Loss function: Cross Entropy Loss Optimiser: SGD with momentum $\gamma = 0.9$ Batch size: 4 Max epochs: 1000 Early stopping: 200	Default
Input field:	Maximum shear strain	Default

4.2.1 Augmentation methods

To generate synthetic variation, different kinds of matrix manipulation was performed. The methodology of performing the different geometric image manipulations was done by following the theoretical approaches presented in Section 2.3.1. Geometric transformations includes manipulation to the damage maps by mirroring, scaling, shearing and rotation. The mirroring transformation should not be confused with the flipping step done to reduce bias in Section 3.1.3. The mirroring transformation used for augmentation only mirrors a damage map without exchanging the label. Increased contrast of the damage maps was obtained by increasing

high values, decreasing low values, and stretching intensity levels. Noise manipulation included applying randomised noise using two approaches. Contrast and noise manipulations were performed using the approaches listed in Table 4.5.

Table 4.5: Methods for matrix manipulation transforms.

Method:	**Algorithm:**	**Description:**
Contrast	See algorithm 1	$I_{i,j}$ is the original value and R controls the magnitude of contrast increase.
Gaussian noise	$J_{i,j} = I_{i,j} + R_{i,j}$	$I_{i,j}$ is the original value and $R_{i,j}$ controls the magnitude of added noise.
Speckle noise	$J_{i,j} = I_{i,j} + R_{i,j} * I_{i,j}$	$I_{i,j}$ is the original value and $R_{i,j}$ controls the magnitude of added noise.

Algorithm 1 Contrast manipulation

$a \leftarrow I_{max} - R \cdot (I_{max} - I_{min})$
$b \leftarrow I_{min} + R \cdot (I_{max} - I_{min})$
if $I_{i,j} > a$ **then**
    $J_{i,j} = I_{max}$
else if $I_{i,j} < b$ **then**
    $J_{i,j} = I_{min}$
else
    $J_{i,j} = (I_{i,j} - b) \cdot \frac{I_{max} - I_{min}}{a - b} + I_{min}$
end if

The transformations were combined by chaining them together in the order illustrated in Figure 4.6.

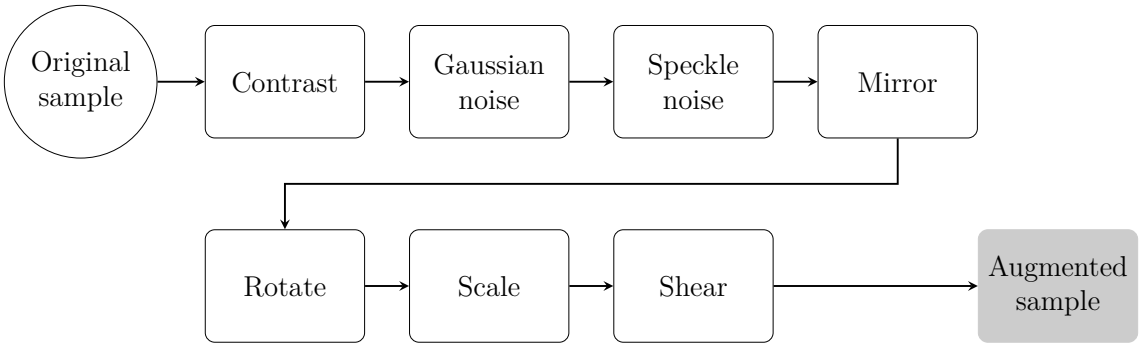
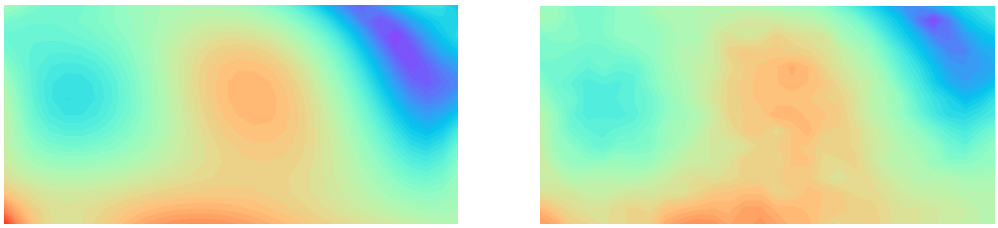


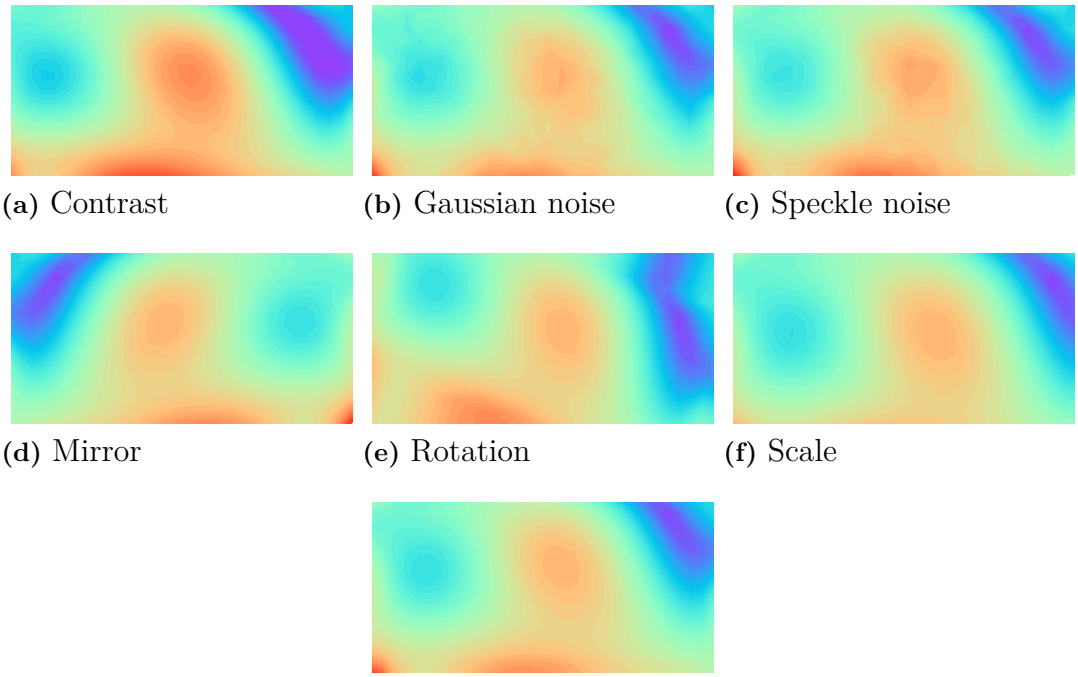
Figure 4.6: Application of transforms when combining augmentation methods.

Figure 4.7 illustrates the heat map of maximum shear strain from a sample in the `dog_cranial_drawer` dataset of both the original damage map and the chained augmentations. Figure 4.8 illustrates all augmentation method applied one by one to the same original sample.



(a) Original **(b)** All augmentation

Figure 4.7: Example of a sample from `dog_cranial_drawer` dataset without augmentation and with all augmentation methods applied.



(a) Contrast **(b)** Gaussian noise **(c)** Speckle noise

(d) Mirror **(e)** Rotation **(f)** Scale

(g) Shear

Figure 4.8: Sample with all augmentation methods applied one by one.

4.2.2 Magnitude of augmentations

All transforms are performed with a randomised parameter set according to a distribution which controls the degree of augmentation. Each time a sample is requested augmentation is performed which includes generating a parameter. Table 4.6 lists the different transformations and the distribution from the parameters were generated. The probability of a sample being mirrored was set to be 50% and not further evaluated.

Table 4.6: Limiting parameter and distribution.

Transform:	**Adjusted parameter:**	**Distribution:**
Contrast	β_c	$R \sim \mathcal{N}(0, (I_{max} \cdot \beta_c)^2)$
Gaussian noise	β_g	$R \sim \mathcal{N}(0, (I_{max} \cdot \beta_g)^2)$
Speckle noise	b_s	$R_{i,j} \sim \mathcal{U}(0, b_s)$
Rotation	σ_θ	$\theta \sim \mathcal{N}(0, \sigma_\theta^2)$
Scale	σ_{c_x}, σ_{c_y}	$c_x, c_y \sim \mathcal{N}(1, \sigma_{c_x,c_y}^2)$
Shear	σ_{s_v}, σ_{s_h}	$s_v, s_h \sim \mathcal{N}(0, \sigma_{s_v,s_h}^2)$

To determine which distribution parameters were suitable to use the network was trained on the dataset using one transformation at the time and varying the limiting parameter. Suitable parameters were established by comparing accuracies achieved. The results are listed in Table 4.7.

Table 4.7: Variation of parameters for distributions when randomly applying transformations.

	Contrast			**Gaussian noise**			**Speckle noise**		
Distribution parameter:	β_c			β_g			b_s		
Value:	0.5	**0.25**	0.01	0.1	0.01	**0.001**	**0.5**	0.25	0.1
Accuracy:	85%	**87%**	85%	84%	85%	**87%**	**86%**	83%	83%

	Rotation			**Scale**			**Shear**		
Distribution parameter:	σ_θ			σ_{c_x} and σ_{c_y}			σ_{s_v} and σ_{s_h}		
Value:	45°	30°	**15°**	**0.45**	0.15	0.05	0.45	**0.15**	0.05
Accuracy:	83%	82%	**86%**	**91%**	88%	82%	81%	**86%**	85%

4.2.3 Combining augmentation methods

In total seven approaches to augment the damage maps were evaluated. To establish which methods were suitable for this dataset and classification problem evaluation of several combinations were tested. Table 4.8 lists the combinations tested and the resulting accuracy for each combination. Training was done twice since the obtained accuracies were varying for each training processes. The combination yielding the highest mean accuracy of the two training processes were further used as augmentation methods. The distribution parameters yielding the highest accuracy from the evaluation in Table 4.7 was used for this evaluation. The matrices were chained together according to the illustration in Figure 4.6.

Table 4.8: Variation of combinations of transforms used for augmentation.

Transform(s):	Contrast	Gaussian Speckle	Mirror	Rotation Scale Shear	Contrast Gaussian Speckle
Accuracy:	0.82	0.86	0.85	0.93	0.7
	0.87	0.82	0.83	0.9	0.82
Average:	0.85	0.84	0.84	0.92	0.76

Transforms:	Contrast Mirror	Contrast Rotation Scale Shear	Gaussian Speckle Mirror	Gaussian Speckle Rotation Scale Shear	Mirror Rotation Scale Shear
Accuracy:	0.86	0.92	0.82	0.92	0.89
	0.85	0.93	0.8	0.89	0.93
Average:	0.86	0.93	0.81	0.92	0.91

Transforms:	Contrast Gaussian Speckle Mirror	Contrast Gaussian Speckle Rotation Scale Shear	Contrast Mirror Rotation Scale Shear	Gaussian Speckle Mirror Rotation Scale Shear	Contrast Gaussian Speckle Mirror Rotation Scale Shear
Accuracy:	0.9	0.91	0.93	0.95	0.94
	0.83	0.93	0.92	0.92	0.9
Average:	0.87	0.92	0.93	**0.94**	0.92

Table 4.9 lists the augmentations methods in terms of distribution parameter for magnitude of transform and the combination yielding highest accuracy. The augmentation methods listed were used for further evaluation of proceeding components.

Table 4.9: Combination of augmentation methods yielding highest accuracy.

Transforms combined:	**Distribution parameter:**
Gaussian noise	$\beta_g = 0.001$
Speckle noise	$b_s = 0.5$
Mirror	$p_m = 0.5$
Rotation	$\sigma_\Theta = 15°$
Scale	$\sigma_{c_x}, \sigma_{c_y} = 0.45$
Shear	$\sigma_{s_v}, \sigma_{s_h} = 0.15$

4.3 Transfer learning

This section aims to evaluate suitable methods of transfer learning. Transfer learning was evaluated by training on data from the `human_anterior_drawer` dataset and `synthetic_dog_cranial_drawer` as well as training on generic images from the `Cifar10` dataset. Evaluation of transfer learning was done with the components listed in Table 4.10.

Table 4.10: Training properties for evaluating transfer learning.

Component:	**Description:**	**Details:**
Network	`ConcatenatedInputNetwork`	Evaluated in Section 4.1.1
Augmentation	Gaussian noise, $\beta_g = 0.001$ Speckle noise, $b_s = 0.5$ Mirror, $p_m = 0.5$ Rotation, $\sigma_\Theta = 15°$ Scale, $\sigma_{c_x}, \sigma_{c_y} = 0.45$ Shear, $\sigma_{s_v}, \sigma_{s_h} = 0.15$	Evaluated in Section 4.2
Learning options	Learning rate: 0.01 Loss function: Cross Entropy Loss Optimiser: Stochastic gradient descent with momentum $\gamma = 0.9$	Default
Input field	Maximum shear strain field	Default

Samples in the `Cifar10` dataset contains RGB images with size of 32×32 pixels. To adapt these images to the dimension of concatenated damage maps these were centre cropped to 30×30 pixels. The images were also converted to grayscale to correspond to the number of input channels of the maximum shear strain field. Figure 4.9 illustrates the transforms done to adapt the dataset.

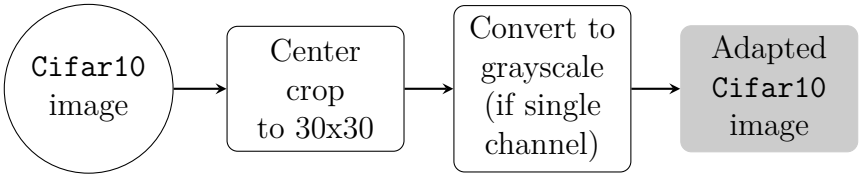


Figure 4.9: Adapting `Cifar10` images to networks.

`ConcatenatedInputNetworkV2` was trained on grayscale converted images from the `Cifar10` dataset using a batch size of 32. No augmentation was used. The network achieved an accuracy of 65% when evaluated on the `Cifar10` test set. Because the `Cifar10` dataset contains 10 different output classes the last fully connected layer (`fc3`) was exchanged to have 10 outputs. After training on the `Cifar10` dataset `fc3` was restored to two output classes for possibility of training the field from the damage maps again.

Transfer learning was evaluated using different networks and pre-training on different datasets. Table 4.11 compiles the different combinations which was tested as well as the accuracy of each combination. ID Ref. is from the evaluation performed in Section 4.1.

Table 4.11: Evaluated transfer learning using different combinations of datasets and locked layers.

ID:	Network:	Training layers:	Dataset:
Ref.	`ConcatenatedInputNetwork`	1-5	`dog_cranial_drawer`
		Accuracy:	95 %
			92 %
		Average:	**94 %**
1.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		1-5	`dog_cranial_drawer`
		Accuracy:	87 %
			92 %
		Average:	90 %
2.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`
		Accuracy:	94 %
			88 %
		Average:	91 %
3.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`human_anterior_drawer`
		4-5	`dog_cranial_drawer`
		Accuracy:	94 %
			92%
		Average:	93 %
4.	`ConcatenatedInputNetworkV2`	1-6	`cifar10`
		1-6	`synthetic_dog_cranial_drawer`
		4-6	`dog_cranial_drawer`
		Accuracy:	68 %
			81 %
		Average:	75 %
5.	`ConcatenatedInputNetworkV2`	1-6	`cifar10`
		1-6	`dog_cranial_drawer`
		Accuracy:	51 %
			67 %
		Average:	59 %
6.	`ConcatenatedInputNetworkV2`	1-6	`cifar10`
		4-6	`dog_cranial_drawer`
		Accuracy:	65 %

When testing the combination with ID 6 the model never improved on the `dog_cranial_drawer` dataset and reported a validation accuracy of 50% constantly without any changes in validation or training loss. This combination was hence only evaluated once.

Since none of the tested transfer learning combinations yielded a higher accuracy

compared to not using transfer learning, it was decided to not use any transfer learning in the following partial evaluations.

4.4 Learning options

This section aimed to evaluate which learning options should be used. This includes evaluating optimiser, loss function and learning rate. The learning options were evaluated by testing different combinations for training. Table 4.12 list the components used for the evaluation. All combinations of two different optimisers, three loss functions and three learning rates were evaluated.

The optimisers evaluated are SGD and Adam. For each optimiser the three loss functions least absolute deviation loss ($\mathcal{L}_1$-loss) least squared error loss ($\mathcal{L}_2$-loss) and cross entropy loss were evaluated. For learning rates, the default values of the PyTorch documentation [58] for each optimiser respectively were used as well as two other similar values. For the highest yielding accuracy using SGD it was also tested to use the optimiser without momentum with the same loss function and learning rate. The options tested and the accuracy obtained are listed in Table 4.13.

Table 4.12: Training options for evaluating learning options.

Component:	**Description:**	**Details:**
Network	`ConcatenatedInputNetwork`	Evaluated in Section 4.1.1
Pre-training	None	Evaluated in Section 4.3
Augmentation	Gaussian noise, $\beta_g = 0.001$ Speckle noise, $b_s = 0.5$ Mirror, $p_m = 0.5$ Rotation, $\sigma_\Theta = 15°$ Scale, $\sigma_{c_x}, \sigma_{c_y} = 0.45$ Shear, $\sigma_{s_v}, \sigma_{s_h} = 0.15$	Evaluated in Section 4.2
Input field	Maximum shear strain	Default

Table 4.13: Accuracy of evaluated learning options including optimiser, loss function and learning rate.

Optimiser:	**Loss function:**	**Learning rate:**	**Accuracy:**
SGD with momentum, $\gamma = 0.9$	$\mathcal{L}_1$-loss	0.1	54 %
		0.01	79 %
		0.001	93 %
	$\mathcal{L}_2$-loss	0.1	61 %
		0.01	**96 %**
		0.001	93 %
	Cross entropy loss	0.1	52 %
		0.01	82 %
		0.001	93 %
SGD without momentum	$\mathcal{L}_2$-loss	0.01	92 %
Adam	$\mathcal{L}_1$-loss	0.01	46 %
		0.001	94 %
		0.0001	93 %
	$\mathcal{L}_2$-loss	0.01	48 %
		0.001	94 %
		0.001	92 %
	Cross entropy loss	0.01	60 %
		0.001	92 %
		0.0001	92 %

The best options are evaluated to be $\mathcal{L}_2$-loss, SGD with momentum $\gamma = 0.9$ and learning rate of 0.01.

4.5 Input fields

This section aimed to describe the method used to investigate which input field(s) was most suitable to use. The Kneedly method generates several fields including the displacement in x- and y- direction, the first principal strain and the maximum shear strain. All fields provided are listed in Table 1.1.

The maximum shear strain field was set as default since it was the most illustrative for interpretation directly by humans for the `dog_cranial_drawer_test`. The most illustrative value for the damage maps generated from humans in the `anterior_drawer_test` is the first principal strain [1]. The displacement fields are the raw data generated from the Kneedly method and directly used to compute the strain fields. The displacement fields, the maximum strain field, the first principal strain as well as the combinations of these fields were evaluated.

Evaluation was done using the components listed in Table 4.14. The different input fields evaluated and the accuracy for each evaluation is listed in Table 4.15.

Table 4.14: Training options for evaluating input fields.

Component:	**Description:**	**Details:**
Network:	`ConcatenatedInputNetwork`	Evaluated in Section 4.1.1
Pre-training	None	Evaluated in Section 4.3
Augmentation	Gaussian noise, $\beta_g = 0.001$ Speckle noise, $b_s = 0.5$ Mirror, $p_m = 0.5$ Rotation, $\sigma_\Theta = 15°$ Scale, $\sigma_{c_x}, \sigma_{c_y} = 0.45$ Shear, $\sigma_{s_v}, \sigma_{s_h} = 0.15$	Evaluated in Section 4.2
Learning options	Learning rate: 0.01 Loss function: $\mathcal{L}_2$-loss Optimizer: Stochastic gradient descent with momentum $\gamma = 0.9$	Evaluated in Section 4.4

Table 4.15: Test accuracy and test loss for alternated input fields.

Input fields:	**Number of channels:**	**Test accuracy:**
Displacements	2	52 %
First principal strain	1	83 %
Max shear strain	1	96 %
Max shear strain First principal strain	2	83 %
Displacements Max shear strain	3	52%
Displacements First principal strain	3	48 %
Displacements Max shear strain First principal strain	4	52 %

5

Evaluation of complete system

This chapter outlines the method of evaluating the final system which combines the results from the partial evaluations in Section 4. The final configuration is listed in Table 5.1. Additional evaluations of variants of systems built from combining other parts were performed. These were evaluated using the same configuration unless otherwise stated. All evaluations in this chapter were evaluated with 4-fold cross validation according to the methodology described in Chapter 3.1.4.

Table 5.1: Properties of system determined from evaluation of components in Section 4.

Component:	**Description:**	**Evaluation:**
Network	`ConcatenatedInputNetwork`	In Section 4.1.1
Augmentation	Gaussian noise, $\beta_g = 0.001$ Speckle noise, $b_s = 0.5$ Mirror, $p_m = 0.5$ Rotation, $\sigma_\Theta = 15°$ Scale, $\sigma_{c_x}, \sigma_{c_y} = 0.45$ Shear, $\sigma_{s_v}, \sigma_{s_h} = 0.15$	In Section 4.2
Pre-training	None	In Section 4.3
Learning options	Learning rate: 0.01 Loss function: $\mathcal{L}_2$-loss Optimizer: Stochastic gradient descent with momentum $\gamma = 0.9$	In Section 4.4
Input field	Maximum shear strain	In Section 4.5

Evaluations have resulted in different accuracies even when the same components of the model have been used. The final system was evaluated one more time to ensure the model did yield similar results as in the evaluation in Chapter 4. The evaluation was done on the `dog_cranial_drawer`. The result of the evaluation is reported in Section 6.1.

5.1 Testing of additional variants

The combinations tested in Section 4 were evaluated part by part and therefore only a subset of all possible combinations have been tested. Because of this there could exist combinations yielding higher results. For example, the network architecture was determined early in the process when suitable components had not been established.

The network architecture was evaluated early in the methodology described in Chapter 4 and hence networks were evaluated once more with the final properties listed in Table 5.1. In the evaluation of which transfer learning that should be used in Section 4.3 the highest accuracy of 94 % was achieved using no transfer learning. The result from using the `synthetic_dog_cranial_drawer` and locking the first four layers of the network achieved an accuracy of 93% and also this proved to be suitable. Therefore both alternatives were re-evaluated.

The additional combinations tested are listed in Table 5.2.

Table 5.2: Additional variants of the final system tested.

ID:	Network:	Training layers:	Dataset:
7.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`
8.	`SiameseInputNetwork`	1-5	`dog_cranial_drawer`
9.	`SiameseInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`
10.	`SingleInputNetwork`	1-5	`dog_cranial_drawer`
11.	`SingleInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`

5.2 Single input with normalisation

`ConcatenatedInputNetwork` and `SiameseNetwork` classifies which of the patient's knee is injured which assumes that one of the knees is injured. `SingleInputNetwork` fulfils the purpose of only classifying if a knee is injured or healthy. Previous testing of this network has not achieved high results which could be because a patient's normal laxity is individual. For this reason, clinicians in practice always test the patient's both knees when diagnosing to get a reference of the patient's normal laxity. When only using one damage map as input this reference is lost which could be the reason the `SingleInputNetwork` for lower performance.

To take the individual laxity into account it was tested to normalise a patient's both knees by dividing the matrix fields with the mean of the opposite matrix. Table 5.3 lists the evaluations performed. The accuracy was evaluated on the `dog_cranial_drawer` dataset. Sensitivity and specificity were calculated according to Equations 2.10 and 2.11. A receiver operating curve was generated.

Table 5.3: Additional variants using `SingleInputNetwork` with normalisation.

ID:	Network:	Training layers:	Dataset:
12.	`SingleInputNetwork`	1-5	`dog_cranial_drawer` with opposite knee as normalisation
13.	`SingleInputNetwork`	1-5	`synthetic_dog_cranial_drawer` with opposite knee as normalisation
		4-5	`human_anterior_drawer` with opposite knee as normalisation

5.3 Class activation maps

The class activation maps (CAM) are generated by extracting the weights of the global average pooling filter and calculating the dot-product with the output of the last convolutional layer. The theory of CAM is presented in Section 2.8.

5.3.1 Concatenated damage maps

Class activation maps are generated from the final system. This was done to increase understanding of what parts of the input the network bases classification on. In order to generate CAMs the network needs to be modified. Figure 5.1 illustrates the network constructed for extracting CAMs. This network is based on `ConcatenatedInputNetwork` but with the max-pooling layer after `conv3` removed. The last fully connected layer was also removed and global average pooling (`GAP`) was used between the last convolutional layer and the fully connected layer `fc1`. Removal of the max-pooling layer was done to increase the spatial resolution of the CAMs.

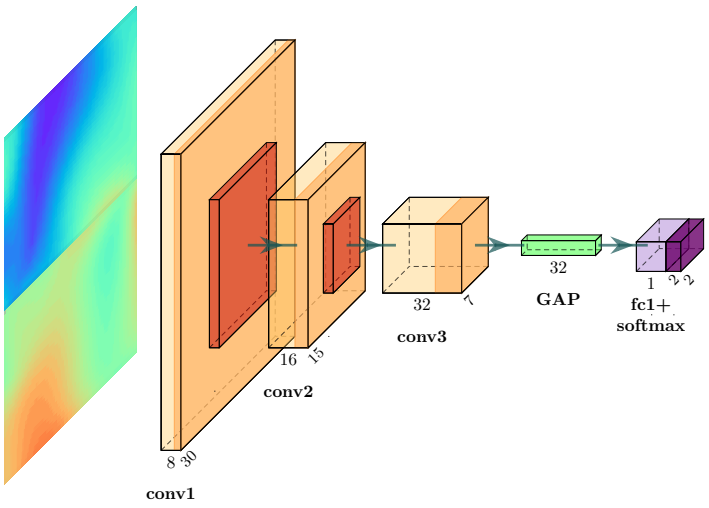
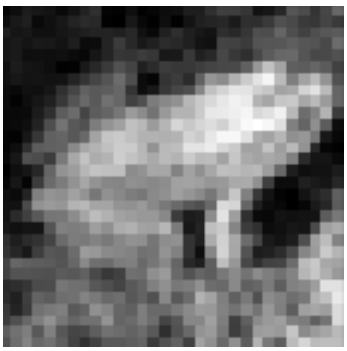
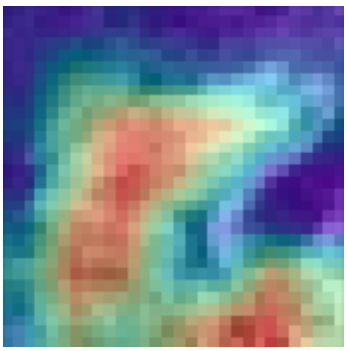


Figure 5.1: Illustration of `ConcatenatedInputNetworkCam` for generating CAMs from concatenated damage maps.

To initially test generation of CAMs `ConcatenatedInputNetworkCam` was trained on the `Cifar10` dataset. The network achieved 60% accuracy when tested on the `Cifar10` test dataset. Examples of CAMs acquired from `Cifar10` are included in Figure 5.2.

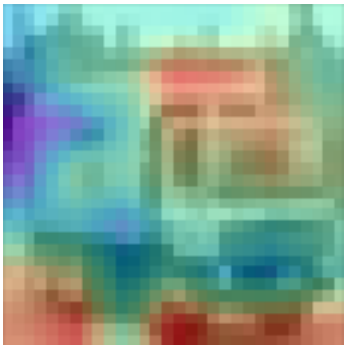


(a) Input image of a frog.



(b) Input image with overlaying CAM.

(c) Input image of a truck.



(d) Input image with overlaying CAM.

Figure 5.2: Example of CAMs for two samples from the `Cifar10 dataset`. In both cases the network classified the images correctly as frog and truck.

Generation of CAMs from the `dog_anterior_drawer` dataset was done using the training options in Table 5.1 and using the partial dataset `dogs_3` for test and `dogs_2` for validation. The remaining partial datasets were used for training.

5.3.2 Single input

The CAMs acquired from the concatenated network shows that the network usually considers one side of the input corresponding to one damage map as important for classification (see result Chapter 6.4.1). To be able to investigate if there are areas in the damage maps corresponding to the same regions of the knee joint which the network considers important for classification it was also decided to investigate CAMs acquired from a single input network. To generate CAMs from single damage maps `SingleInputNetwork` was modified similar to `ConcatenatedInputNetworkCam`. Figure 5.3 illustrates the constructed network `SingleInputNetworkCam`.

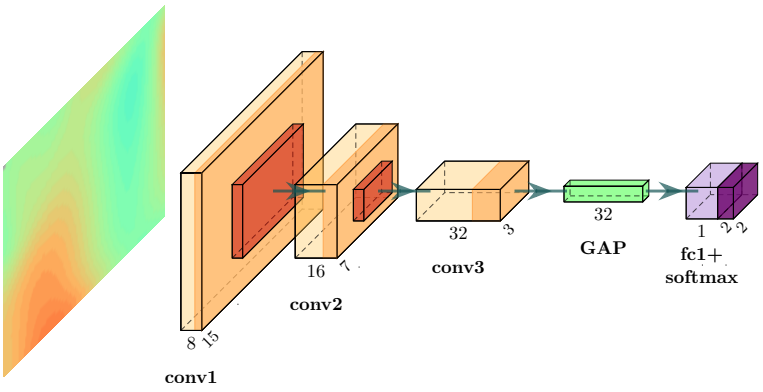


Figure 5.3: Illustration of `SingleInputNetworkCam` for generating CAMs from single damage maps.

Generation of CAMs for the `SingleInputNetwork` from the `dog_anterior_drawer` dataset was done using the training options in Table 5.1 and using the partial dataset `dogs_3` for test and `dogs_2` for validation. The remaining partial datasets were used for training.

5.4 Testing of dataset collected from humans

Additional evaluation was done to test data collected from performing clinical tests on humans. Configurations in Table 5.1 were used and the `human_anterior_drawer` dataset was used for training, evaluation and testing using 4-fold cross validation performed with the same methodology as described in Section 3.1.4. For transfer learning testing without pre-training and using the synthetic dataset with locked layers was evaluated. Because the most illustrative field for manual interpretation for the `dog_cranial_drawer` dataset also yielded the highest accuracy, the first principal strain was used as input when evaluating the `human_anterior_drawer` dataset. Table 5.4 lists the evaluations done on the `human_anterior_drawer`.

Table 5.4: Additional variants tested for evaluation of using the damage maps acquired from humans.

ID:	Network:	Training layers:	Dataset:
14.	`ConcatenatedInputNetwork`	1-5	`human_anterior_drawer`
15.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`human_anterior_drawer`

6

Results

This chapter lists the results from the evaluation of the systems outlined in Chapter 5.

6.1 Final system

The partial results with highest accuracy achieved for components in Section 4 are put together for the final system. The model achieved a mean accuracy of 94% for the final evaluation when using 4-fold cross validation on the `dog_cranial_drawer` dataset. Figure 6.1 lists the accuracy for each combination of test and validation set done using cross validation.

Figure 6.1: Accuracies achieved for the final system.

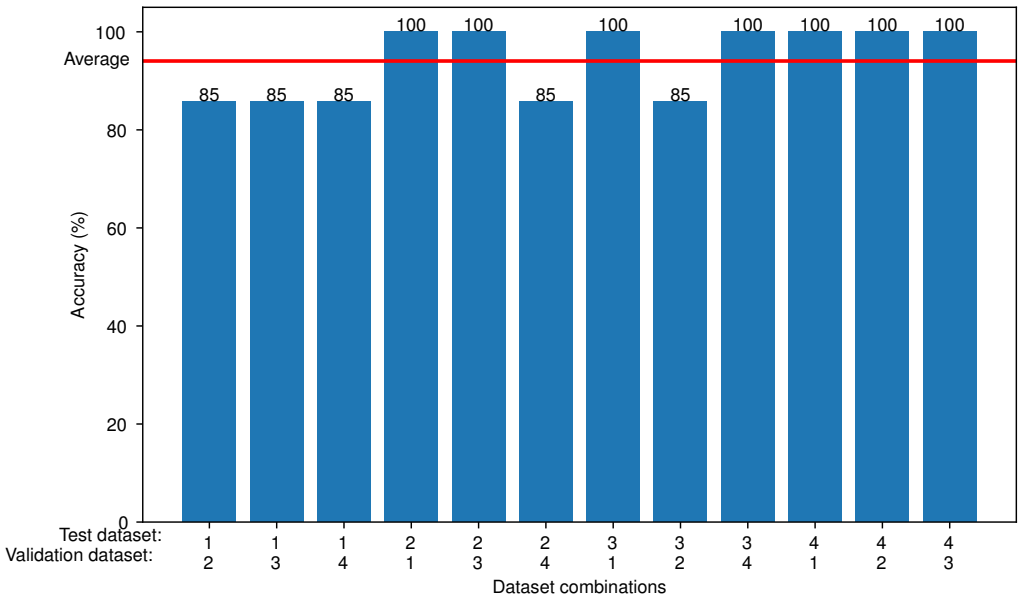


Figure 6.2 illustrates the learning process for one of 4-fold combinations from evaluation of the final system. Twelve different diagrams are created when using 4-fold

cross validation. Only one training process is illustrated in the figure, however every training process yields similar graphs.

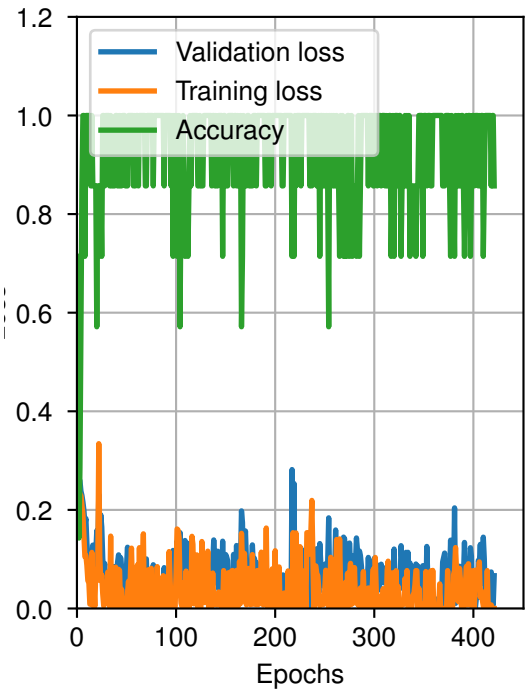


Figure 6.2: Training process when evaluating one of the combinations when performing 4-fold cross validation on the final system.

The clinical performance, in terms of sensitivity and specificity, cannot be evaluated for networks using a patient's both knees as input and classifies which of the knees are injured since a false positive will always result in a false negative and vice versa.

6.2 Additional variants

Table 6.1 lists the configurations and accuracies from performing the method in Section 5.1.

Table 6.1: Accuracies when evaluating additional variants.

ID:	Network:	Training layers:	Dataset:
7.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`
		Accuracy:	78%
8.	`SiameseInputNetwork`	1-5	`dog_cranial_drawer`
		Accuracy:	88%
9.	`SiameseInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`
		Accuracy:	89 %
10.	`SingleInputNetwork`	1-5	`dog_cranial_drawer`
		Accuracy:	82%
		AUC:	0.92
		Sensitivity:	89%
		Specificity:	75%
11.	`SingleInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`dog_cranial_drawer`
		Accuracy:	82%
		AUC:	0.94
		Sensitivity:	84%
		Specificity:	80%

6.3 Single input with normalisation

Table 6.2 lists the accuracies when evaluating the models configured according to the methodology in Section 5.2.

Table 6.2: Accuracies when evaluating additional variants using `SingleInputNetwork` and normalisation.

ID:	Network:	Training layers:	Dataset:
12.	`SingleInputNetwork`	1-5	`dog_cranial_drawer` with opposite knee normalisation.
		Accuracy:	87%
		AUC:	0.95
		Sensitivity:	89%
		Specificity:	85%
13.	`SingleInputNetwork`	1-5	`synthetic_dog_cranial_drawer` with opposite knee normalisation.
		4-5	`dog_cranial_drawer` with opposite knee normalisation.
		Accuracy:	88%
		AUC:	0.95
		Sensitivity:	85%
		Specificity:	91%

Figure 6.3 illustrates the receiver operating characteristics and AUC for one of the cross validations when training `SingleInputNetwork` with pre-training and normalisation labelled as ID 13 in Table 6.2. The receiver operating characteristics for the remaining tests done in the cross validation was similar and was because of this not included.

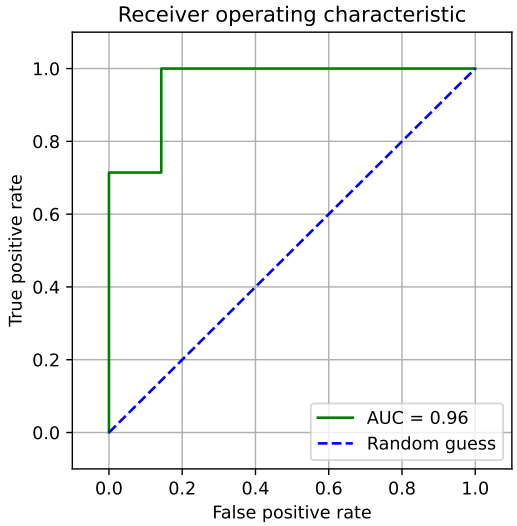


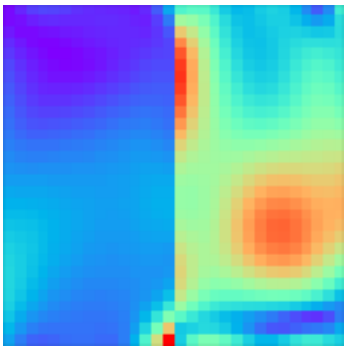
Figure 6.3: Receiver operating characteristics for one validation when performing cross validation on `SingleInputNetwork` with pre-training and normalisation.

6.4 Class activation maps

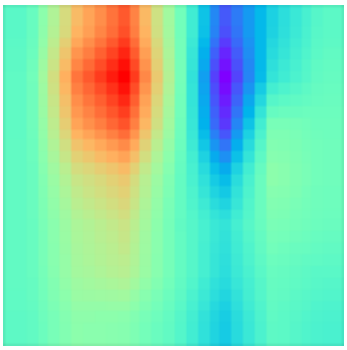
This section lists the results from generation of class activation maps in Section 5.3.

6.4.1 Concatenated

Figures 6.4, 6.5 and 6.6 illustrates the input and associated CAMs from three different concatenated samples from the `dog_cranial_drawer` dataset. In the Figures 6.4 and 6.5 two cases the network correctly classified the damage maps as right injured respectively left injured. In Figure 6.5 the network incorrectly classified the damage maps as a left knee injury. The left sides of the heat maps correspond to the left knee. The network achieved an accuracy of 66%.

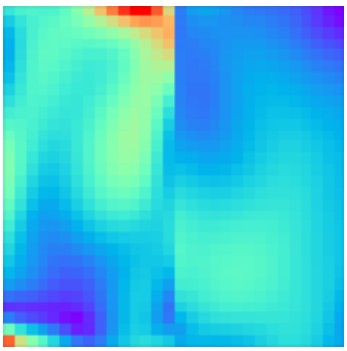


(a) Concatenated damage maps.

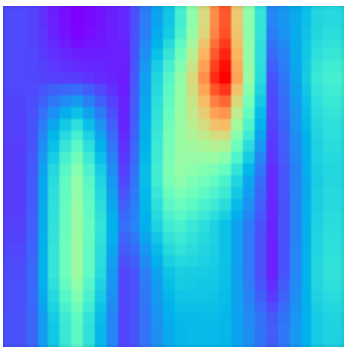


(b) CAM.

Figure 6.4: Example of CAM from the `dog_cranial_drawer` dataset. Right knee correctly classified as injured.

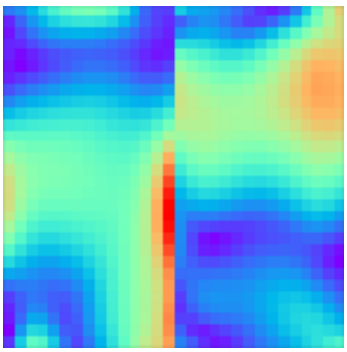


(a) Concatenated damage maps.

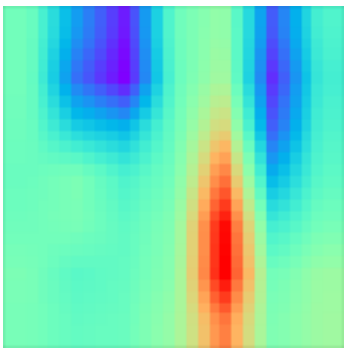


(b) CAM.

Figure 6.5: Example of CAM from the `dog_cranial_drawer` dataset. Left knee correctly classified as injured.



(a) Concatenated damage maps.



(b) CAM.

Figure 6.6: Example of CAM from the `dog_cranial_drawer` dataset. Left knee incorrectly classified as injured.

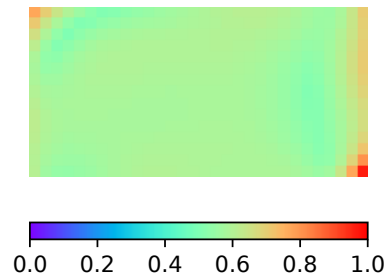
6.4.2 Single input

Table 6.3 lists the performance of the `SingleInputNetworkCAM`.

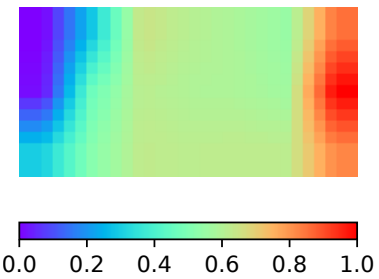
Table 6.3: Results from evaluation of `SingleInputNetworkCAM` using cross validation.

Using normalisation:	Accuracy:	Sensitivity:	Specificity:	AUC:
No	87%	82%	93%	0.91
Yes	93%	90%	96%	0.95

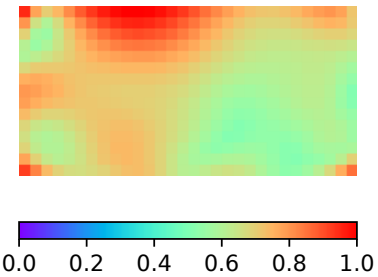
Figure 6.7 illustrates a pair of damage maps and their associated CAMs when not using normalisation. Figure 6.8 illustrates the same pair of damage maps associated CAMs when using normalisation.



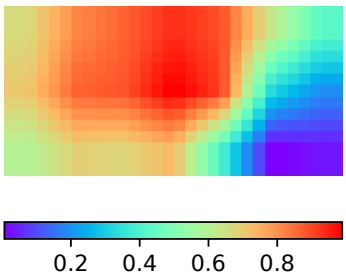
(a) Right damage map. Correctly classified as healthy.



(b) Right CAM.

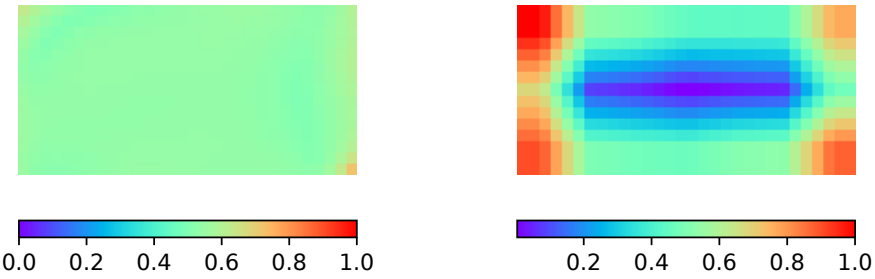


(c) Left damage map. Correctly classified as injured.



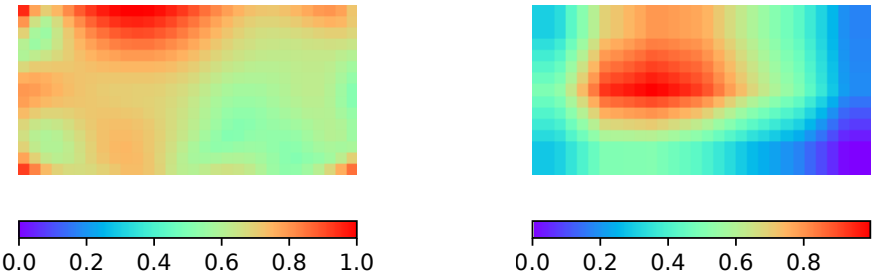
(d) Left CAM.

Figure 6.7: A pair of damage maps and their associated CAMs. Left knee is injured.



(a) Right damage map. Correctly classified as healthy.

(b) Right CAM.

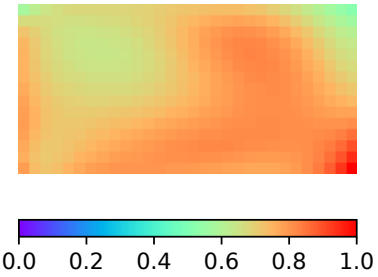


(c) Left damage map. Correctly classified as injured.

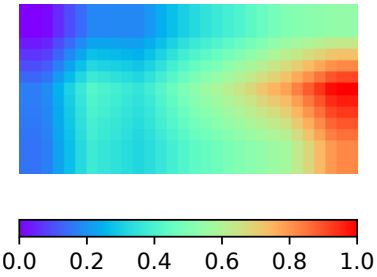
(d) Left CAM.

Figure 6.8: A pair of damage maps and their associated CAMs when using normalised damage maps. Left knee is injured.

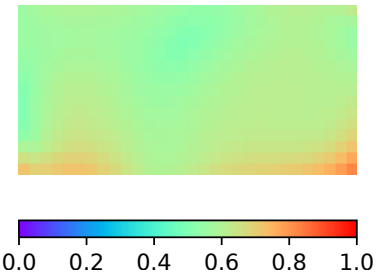
Figure 6.9 illustrates a pair of damage maps and their associated CAMs when not using normalisation. Figure 6.10 illustrates the same pair of damage maps associated CAMs when using normalisation.



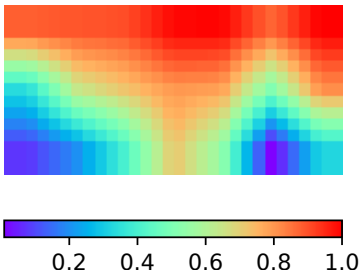
(a) Right damage map. Correctly classified as injured.



(b) Right CAM.

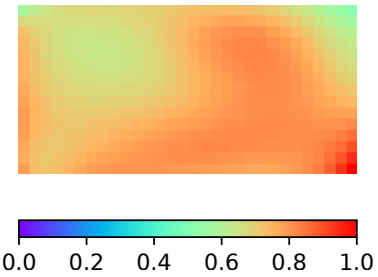


(c) Left damage map. Correctly classified as healthy.

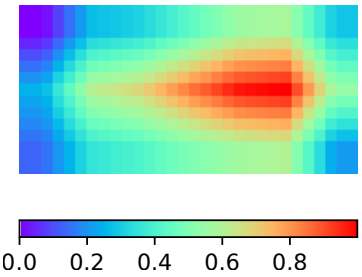


(d) Left CAM.

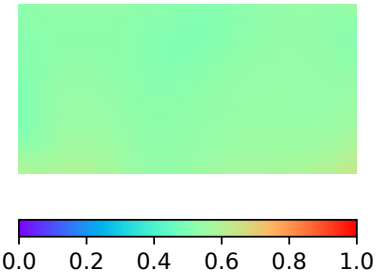
Figure 6.9: A pair of damage maps and their associated CAMs. Right knee is injured.



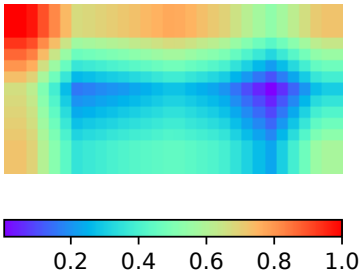
(a) Right damage map. Correctly classified as injured.



(b) Right CAM.



(c) Left damage map. Correctly classified as healthy.



(d) Left CAM.

Figure 6.10: A pair of damage maps and their associated CAMs using normalised damage maps. Right knee is injured.

6.5 Testing of dataset collected from humans

This section lists the results for evaluation using the dataset collected from humans. The methodology of the evaluations are described in Section 5.4. Table 6.4 lists the performance of the `human_anterior_drawer` dataset with and without transfer learning.

Table 6.4: Results from the evaluations on the dataset collected from humans.

ID:	Network:	Training layers:	Dataset:
14.	`ConcatenatedInputNetwork`	1-5	`human_anterior_drawer`
		Accuracy:	43%
15.	`ConcatenatedInputNetwork`	1-5	`synthetic_dog_cranial_drawer`
		4-5	`human_anterior_drawer`
		Accuracy:	65%

7

Discussion

In this chapter the approach to meet the aims of this thesis, cause of partial results and the utility of the results from the final system are discussed. Several limitations are considered connected to evaluation of different components and preconditions. Future improvements are suggested to mitigate the limitations of the study. Ethical considerations are also raised.

7.1 Difference in accuracy

For every time the networks were trained with the same options and dataset different results were obtained. One possible explanation is the randomised parameters which the networks are initialised using. Each time the network are initialised they are given randomised weights and with the small amount of training data it is likely that all parameters cannot be sufficiently trained, hence some difference in accuracy will occur from each testing. The augmentation and the order of which samples are loaded also have some randomised properties. Another possible reason contributing to the difference in results from each evaluation is that with a small dataset the test dataset also becomes even smaller and with few samples in the test data the resolution of the accuracy becomes low. If only one sample is classified different from one evaluation compared to another this will have considerable effect on the accuracy. This was the reason for performing some evaluations multiple times and using the mean of the accuracy for comparison of suitable options.

The highest accuracy achieved during the ablation study was 96%. Because of the difference in accuracy when training and evaluating using the same options some accuracies obtained will probably be higher or lower than what usually is achieved for the same validation. Due to the methodology of the ablation study which chooses the highest accuracy from several options it is possible that an accuracy higher than representative have been achieved. For the last evaluation in Chapter 4 an accuracy of 96 % was achieved but several evaluations yielded an accuracy of 92-96 %. Due to the risk of getting an accuracy not representative of the performance for the model an additional evaluation was done in Section 5 which resulted in an accuracy of 94 % listed in 6.1. The accuracy of 94 % is considered to be the accuracy achieved for the final system.

7.2 Partial evaluations

In the partial evaluation some variations of the components clearly provided increased accuracy. An example is the evaluation of input fields in Section 4.5 where the highest accuracy when using the maximum shear strain (96 %) was 13 percentage points higher than the accuracy of the second-best option using the maximum shear strain combined with the first principal strain (83 %). For these evaluations it was simple to identify the most suitable configuration without calculating the mean of several evaluations.

For other evaluations the results were similar despite varying configurations. The highest result was for these evaluation further used although it cannot be assured that the best option was chosen. On the other hand it can be argued that if multiple components resulted in high accuracy they were all suitable. An example of this kind is the evaluation of augmentation methods in Section 4.2.2. It was expected that with the high variation of parameters one configuration would stand out clearly but this was not the case for all options.

Since it cannot be guaranteed that the best components were chosen throughout the study evaluation of additional variants were performed in Section 5.1. An example of the additional variants was to evaluate the networks structures again but using the options yielding highest accuracy from the partial evaluations. During the first evaluation in Section 4.1.1 of suitable networks structure the `ConcatenatedInputNetwork` achieved one percentage point higher accuracy than the `SiameseInputNetwork` (82 % accuracy compared to 81 % accuracy). The difference is too small to draw any conclusion about which network structure is optimal. In Section 5.1 when the `SiameseInputNetwork` was re-evaluated using the determined suitable options from the partial evaluations the `SiameseInputNetwork` did still not perform better than the `ConcatenatedInputNetwork`. Because of this it can with higher certainty be established that the `ConcatenatedInputNetwork` was more suitable.

7.3 Datasets

The initial plan was to use a dataset collected from performing clinical examinations on humans but only 65 % of the samples in the dataset could be classified correctly. One possible reason for poor performance was the limited amount of data. The data collected from the study of dogs neither reached the number of samples expected but the network could still be trained to classify the dataset with high accuracy. One possible reason for higher accuracy on the `dog_cranial_drawer` dataset is that the samples within the same class had less variations than the samples in the `human_anterior_drawer` dataset.

A possible reason for samples in the `dog_cranial_drawer` being more similar than the data in the `human_anterior_drawer` is that dogs are sedated while performing the clinical tests. On the human dataset the samples tend to vary more depending if

the patient for example is afraid that the test will hurt and then tenses their muscles. This is also a common reason for the diagnostic tests being hard to interpret for clinicians. A second probable reason is that dogs have less soft tissue, such as muscle or fat tissue, around the knee joint and hence movement from this tissue does not affect the movement in the recordings as much as for the examinations of humans. A third possible reason is that tests collected on dogs are mainly performed by highly experienced veterinary orthopaedists, while the human dataset was collected from medical personnel with varying experience and hence the test execution might also differ somewhat more for the human dataset.

The classification problem on the dataset collected from performing knee test on humans can be expected to be more difficult to solve because the samples are more varying. However, the promising result from the `dog_cranial_drawer` indicates that a similar approach as evaluated in this thesis could also be used for human knee tests if more data is collected. With more training data the model would have possibility to identify which values in the damage maps arise from higher laxity indicating injury and should be noticed and which high values arise from other factors and should be disregarded.

7.4 Transfer learning

Using transfer learning with the `Cifar10` dataset did not increase the performance of the trained models. This result was not expected since one hypothesis was that given the small dataset pre-training would be crucial and since the first layers tend to learn to identify basic features such as edges it was assumed that pre-training could only increase performance. One possible explanation for the low accuracy when using pre-training on the `Cifar10` dataset it that the dataset was too much unlike the damage maps in the `dog_cranial_drawer` and hence was not suitable. Instead of using networks initialised with weights for solving a too different classification problem, it could be better to use randomised weights given that all weights could not be adequately trained using a small dataset.

The synthetic dataset did not significantly increase nor decrease performance, which also was surprising. One possible reason is that with the similarities in the `dog_cranial_drawer` the classification problem is simple enough to be solved with the provided samples.

7.5 Single knees or reference knees

The models taking two damage maps as inputs, representing a patient's both knees, yields higher accuracy than the models taking one damage map as an input. This is assumed to be because the normal laxity differs between patients and by using both knees as input the difference can be accounted for. The individual laxity is the reason for care personnel to examine both knees as standard procedure and one reason why knee ligament injuries are hard to diagnose correctly.

The networks taking two damage maps as input classifies the patient to have an injury on the right or the left knee. One considerable limitation is that the system assumes that one knee is injured. The model will fit each patient into the two classes and always output one injured knee. For the design of this thesis this is applicable since all patients included in the dataset have one injured and one healthy knee but the model does not provide future value for diagnostics in clinical setting. For further work samples from patients with two healthy knees or two injured knees should be collected for possibility training models to have output classes representing all possible cases.

The reason for limiting the collection of data to only include patients with one healthy and one injured knee was because of how the data collections were structured. The damage maps collected from dogs are from patients with an already set diagnosis, visiting a veterinarian clinic to undergo surgery to repair the injury. For humans an ethical approval for collecting damage maps only from patients with one injured knee and one healthy knee had been approved.

The network taking only one damage map as input has the benefit of being able to classify a damage map as injured or healthy. Although the performance does not reach the accuracy of the networks taking two damage maps as input the results are higher than expected. The `SingleInputNetwork` with normalisation was tested in Section 5.2 to take the individual laxity into account. This evaluation is of interest since is provides possible use for diagnosing knee ligament injury in clinical settings. When using normalisation, the accuracy of `SingleInputNetwork` increase to 88 % compared to 82 % when not using normalisation. This result is not only promising but also confirms the importance of taking individual laxity when diagnosing knee ligament injuries.

7.6 Class Activation Maps

To increase understanding of how the networks classifies the damage maps CAMs was generated. CAMs generated for the concatenated damage maps are provided in Figures 6.4 and 6.5. In these figures it is evident that areas with low strains are important for classifications. For Figure 6.6, which was classified incorrectly, the model weighted areas of low strains in the injured knee as important for classification. This indicates that the `Concatenated_input_network` tends to look at the healthy knee and at areas of low values representing low movement in the knee. This further suggests that the model learns to identify the healthy knee rather than the injured, but also that the model takes the healthy knee into consideration when classifying.

CAMs was also generated for single damage maps using both normalisation and without normalisation. For these CAMs no clear connection to the damage maps can be distinguished. However, by comparing the damage map from a pair of knees the strains in the damage maps representing injured knees are typically larger than for the healthy knees. It is possible that the network simply uses the magnitude of the strains for classification without considering patterns or locations representing

special areas over the knee joint. When using normalisation, to account for the magnitude of the strains from the damage map from the opposite knee, higher accuracy was achieved. This is additionally an indication of that magnitudes of the strains are important.

7.7 Augmentation

Augmentation was evaluated early in the ablation study. When using the augmentation method evaluated to be most suitable accuracy increased with 12 percentage points compared to not using augmentation, from 82% to 94 %. It can be established that the augmentation was important for this problem and one possible reason for the large impact can be the small dataset.

7.8 Learning options

Learning options have high impact on performance. The choice of optimiser and loss function did not impact the accuracy of the model substantially but choice of learning rate had high impact on accuracy. Compared to the learning rate chosen for best performance a much lower learning clearly reduced performance while increasing learning rates did not impact the accuracy considerably. The computational cost is higher with a higher learning rate although it was not measured in this thesis. Learning options is generally an important aspect for using neural networks and the impacts from these components are not supposed to be especially important for this thesis.

7.9 Ethical considerations

For the data collection of dogs clinical tests were recorded on dogs before they were about to undergo surgery conventionally diagnosed with an injury on one knee. The dogs are sedated which is standard procedure before surgery but also of advantage for recordings. Before surgery the fur of the injury knee is conventionally shaved. To capture the movement of the skin origin from the knee joint shaving of the fur was advantageous. To collect the damage map of the healthy knee this knee was also shaved for the purpose of collecting equal recordings which can be questioned from an ethical perspective since it would not have been done as standard procedure. All dog owners were informed about the study and approved that both knees were shaved for the purpose of collecting recordings to evaluate the performance of the Kneedly method.

The data collected from both humans and dogs did not include patients with two healthy knees which is considered a limitation of this thesis. The reason for this was partly ethical approval. To collect data from patients with two healthy knees for humans it would require an amendment of the ethical approval of the study. No ethical approval was required for the study on dogs since the dog are considered

being in ordinary environment during veterinary visits since the dog is brought by its owner. The tests are additionally performed by experienced veterinarians that would have performed these test regardless of if the data collection was in question or not and the methodology is non-invasive. However, collection of data from dogs with two healthy knees is ethically questionable because this would require doing unnecessary test, sedate and shave fur only for the purpose of collecting damage maps.

No information from the Kneedly method or from this study was provided to care personnel before setting diagnosing and determining treatment plan. This was an important aspect to consider to not bias the care providers and ensure that the study did not affect patient care.

If the methodology was to be implemented in clinical settings for diagnosing knee ligament injuries an important aspect would be to ensure certainty. One aspect would be to evaluate on a larger dataset to ensure a training and evaluation on a broad population to avoid possible bias of a limited dataset. The sensitivity should especially be evaluated to be adequate to ensure not to miss a positive diagnose since untreated injuries or delayed diagnose can lead to secondary injuries and increased suffering for the patient [59].

7.10 Future improvements

A major limitation of this thesis is the lack of data. Both in number of samples but also lack of patients with two healthy knees or two injured knees. Including an extended study population would allow the trained models to achieve higher clinical applicability.

Using displacement as input to the models did not work in this thesis. With more training data it is also possible that other input fields could be used as input and achieve equal or higher accuracy. Since the strains can be derived from the displacements more training data might allow the model to directly use displacements as input.

Future ideas for increased automation of the method for collecting damage maps are not included in the scope of this thesis but expected to increase standardisation of the damage maps. Example of tasks which could be automated is to automate the area of where damage maps are generated from by using segmentation of the knee joint and to use automated extractions of video sequences.

Another option that could increase accuracy and should be investigated for future projects are optical flow for determining displacements using machine learning trained on recordings of knees.

Further work additional includes to implement the methodology described in this thesis on humans given more data is provided. The models classify samples obtained from dogs with high accuracy and the clinical tests for dogs and humans are very

similar. This indicates that with sufficient training data there is a potential for also classifying damage maps from humans with high accuracy.

8

Conclusion

This thesis concludes that it is possible to identify injury to the CCL in dogs from pairs of healthy and injured knees using machine learning to classify damage maps acquired from performing the cranial drawer test. The CNN developed in this thesis achieved an accuracy of 94% when using cross-validation. A maximum accuracy of 96% was from the same model achieved but 94% is considered a representative value of performance. These accuracies might not be generalised to a larger population since the dataset used had few samples and few unique patients.

Using augmentation when training the model increased accuracy from 82% to 94% which indicates it is of high importance. However, it is not possible to determine which augmentation methods are most suitable. Transfer learning did not increase performance. Using pre-training of data unlike the dataset decreased performance while data similar to the dataset did not decrease nor increase performance. Using maximum shear strain as input to the model archived highest accuracy. The displacement fields could not be used for classification which indicates the need of processing of the raw data.

Highest accuracy was yielded when using two damage maps as input to the model. This indicates the importance of taking the patient's individual laxity into account, which is also considered when conventionally diagnosing CCL injuries. A model only using a single damage map as input achieved an accuracy of 88% when using normalisation with a scalar value obtained from the opposite knee and pre-training. Using a single damage map as input without normalisation achieved an accuracy of 82%. This indicates potential to determine if a knee is injured without taking individual laxity into account.

References

[1] A. Nilsson, J. Nilsson, A. Rydevald, and L. Söderholm, "Objektiv diagnostik av ligamentskador i knän," 2020.

[2] W. E. Prentice, "The knee and related structures," in *Essentials of Athletic Injury Management*. New York: McGraw-Hill Education, 2016, ch. 16, ISBN: 978-0-07-802275-3.

[3] G. A. Malanga, S. Andrus, S. F. Nadler, and J. McLean, "Physical examination of the knee: A review of the original test description and scientific validity of common orthopedic tests," *Archives of Physical Medicine and Rehabilitation*, vol. 84, no. 4, pp. 592–603, Apr. 2003. DOI: 10.1053/apmr.2003.50026.

[4] M. F. Sobrado, M. B. Bonadio, G. F. Ribeiro, P. N. Giglio, C. P. Helito, and M. K. Demange, "Lever sign test for chronic acl injury: A comparison with lachman and anterior drawer tests," *Acta Ortopédica Brasileira*, vol. 29, no. 3, pp. 132–136, Aug. 2021. DOI: 10.1590/1413-785220212903238345.

[5] Y. Hoshino, P. Araujo, M. Ahlden, C. G. Moore, R. Kuroda, S. Zaffagnini, J. Karlsson, F. H. Fu, and V. Musahl, "Standardized pivot shift test improves measurement accuracy," *Knee Surgery, Sports Traumatology, Arthroscopy*, vol. 20, no. 4, pp. 732–736, Dec. 2011. DOI: 10.1007/s00167-011-1850-0.

[6] K. Engdahl, "The epidemiology of stifle joint disease in dogs with a focus on cranial cruciate ligament disease," *Acta Universitatis Agriculturae Sueciae*, no. 2021:59, Nov. 2021.

[7] I. Silva, *Artificial neural networks : a practical course*. Switzerland: Springer, 2016, ISBN: 978-3-319-43162-8.

[8] M. Mohammed, *Machine learning : algorithms and applications*. Boca Raton: CRC Press, 2017, ISBN: 9781315371658.

[9] U. Michelucci, *Advanced Applied Deep Learning*. Apress, 2019. DOI: 10.1007/978-1-4842-4976-5.

[10] BrunelloN, *Example of a deep neural network*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Example_of_a_deep_neural_network.png (visited on 02/22/2022).

[11] S. Herculano-Houzel, "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost," *Proceedings of the National Academy of Sciences*, vol. 109, no. supplement_1, pp. 10 661–10 668, Jun. 2012. DOI: 10.1073/pnas.1201895109.

[12] Egm4313.s12, *Neuron and myelinated axon*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Neuron3.png (visited on 02/24/2022).

[13] Chrislb, *Diagram of an artificial neuron*. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=224550 (visited on 02/22/2022).

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[15] R. Shanmugamani, *Deep Learning for Computer Vision*. Packt Publishing, 2018, ISBN: 978-1-78829-562-8.

[16] I. Goodfellow, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016, ISBN: 9780262337373.

[17] G. Chandrarathne, K. Thanikasalam, and A. Pinidiyaarachchi, "A comprehensive study on deep image classification with small datasets," in *Lecture Notes in Electrical Engineering*, Springer Singapore, Dec. 2019, pp. 93–106. DOI: 10.1007/978-981-15-1289-6_9.

[18] S. Ghosh, N. Das, and M. Nasipuri, "Reshaping inputs for convolutional neural network: Some common and uncommon methods," *Pattern Recognition*, vol. 93, pp. 79–94, 2019, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2019.04.009.

[19] S. Russell, *Artificial intelligence : a modern approach*. Harlow, England: Pearson Education, 2016, ISBN: 9781292153971.

[20] A. H. Masquelin, N. Cheney, C. M. Kinsey, and J. H. T. Bates, "Wavelet decomposition facilitates training on small datasets for medical image classification by deep learning," *Histochemistry and Cell Biology*, vol. 155, no. 2, pp. 309–317, Jan. 2021. DOI: 10.1007/s00418-020-01961-y.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, 2012. eprint: arXiv:1207.0580.

[22] S. Ranjan, *Applied deep learning and computer vision for self-driving cars : build autonomous vehicles using deep neural networks and behavior-cloning techniques*. Birmingham: Packt Publishing, 2020, ISBN: 978-1-83864-630-1.

[23] Y. Yang, L. Zhang, M. Du, J. Bo, H. Liu, L. Ren, X. Li, and M. J. Deen, "A comparative analysis of eleven neural networks architectures for small datasets of lung images of covid-19 patients toward improved clinical decisions," *Computers in Biology and Medicine*, vol. 139, p. 104 887, 2021, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2021.104887.

[24] Martin Thoma. (Feb. 15, 2016). "Overfitting," [Online]. Available: https://commons.wikimedia.org/wiki/File:2d-epochs-overfitting.svg (visited on 02/22/2022).

[25] K. Janocha and W. M. Czarnecki, *On loss functions for deep neural networks in classification*, 2017. DOI: 10.48550/ARXIV.1702.05659.

[26] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. eprint: arXiv:1609.04747.

[27] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[28] T. Gültekin and A. Uğur, "A hybrid approach based on transfer and ensemble learning for improving performances of deep learning models on small datasets.," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 29, no. 7, pp. 3197–3211, 2021, ISSN: 13000632.

[29] D. Mungra, A. Agrawal, P. Sharma, S. Tanwar, and M. S. Obaidat, "PRATIT: A CNN-based emotion recognition system using histogram equalization and data augmentation," vol. 79, no. 3-4, pp. 2285–2307, Nov. 2019. DOI: 10.1007/s11042-019-08397-0.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," vol. 60, no. 6, pp. 84–90, May 2017. DOI: 10.1145/3065386.

[31] T. A. Korzhebin and A. D. Egorov, "Comparison of combinations of data augmentation methods and transfer learning strategies in image classification used in convolution deep neural networks," in *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, IEEE, Jan. 2021. DOI: 10.1109/elconrus51938.2021.9396724.

[32] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 4th ed. Parson, 2018.

[33] P. Rivas, *Deep Learning for Beginners*. Birmingham, England: Packt Publishing, Sep. 2020, p. 69.

[34] S. Guan, T. Wang, K. Sun, and C. Meng, "Transfer learning for nonrigid 2d/3d cardiovascular images registration," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 9, pp. 3300–3309, Sep. 2021. DOI: 10.1109/jbhi.2020.3045977.

[35] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, May 2016. DOI: 10.1109/tmi.2016.2535302.

[36] *Imagenet*. [Online]. Available: https://www.image-net.org/ (visited on 02/20/2022).

[37] Q. Pan, M. Jia, Q. Liu, L. Zhang, J. Pan, F. Lu, Z. Zhang, L. Fang, and H. Ge, "Identifying patient–ventilator asynchrony on a small dataset using image-based transfer learning," *Sensors*, vol. 21, no. 12, p. 4149, Jun. 2021. DOI: 10.3390/s21124149.

[38] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997. DOI: 10.1016/s0031-3203(96)00142-2.

[39] D. N. Carvajal and P. C. Rowe, "Research and statistics: Sensitivity, specificity, predictive values, and likelihood ratios," *Pediatrics in Review*, vol. 31, no. 12, pp. 511–513, Dec. 2010. DOI: 10.1542/pir.31-12-511.

[40] M. Goyal, T. Knackstedt, S. Yan, and S. Hassanpour, "Artificial intelligence-based image classification methods for diagnosis of skin cancer: Challenges and opportunities," *Computers in Biology and Medicine*, vol. 127, p. 104 065, Dec. 2020. DOI: 10.1016/j.compbiomed.2020.104065.

[41] K. C. Siontis, P. A. Noseworthy, Z. I. Attia, and P. A. Friedman, "Artificial intelligence-enhanced electrocardiography in cardiovascular disease management," *Nature Reviews Cardiology*, vol. 18, no. 7, pp. 465–478, Feb. 2021. DOI: 10.1038/s41569-020-00503-2.

[42] X. Liu, L. Faes, A. U. Kale, S. K. Wagner, D. J. Fu, A. Bruynseels, T. Mahendiran, G. Moraes, M. Shamdas, C. Kern, J. R. Ledsam, M. K. Schmid, K.

Balaskas, E. J. Topol, L. M. Bachmann, P. A. Keane, and A. K. Denniston, "A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: A systematic review and meta-analysis," *The Lancet Digital Health*, vol. 1, no. 6, e271–e297, Oct. 2019. DOI: 10.1016/s2589-7500(19)30123-2.

[43] T. Zhang, L. Yu, N. Hu, S. Lv, and S. Gu, "Robust medical image segmentation from non-expert annotations with tri-network," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, Springer International Publishing, 2020, pp. 249–258. DOI: 10.1007/978-3-030-59719-1_25.

[44] *ISIC*. [Online]. Available: https://www.isic-archive.com/.

[45] J. P. E. Schouten, C. Matek, L. F. P. Jacobs, M. C. Buck, D. Bošnački, and C. Marr, "Tens of images can suffice to train neural networks for malignant leukocyte detection," vol. 11, no. 1, Apr. 2021. DOI: 10.1038/s41598-021-86995-5.

[46] C. M. Felmingham, N. R. Adler, Z. Ge, R. L. Morton, M. Janda, and V. J. Mar, "The importance of incorporating human factors in the design and implementation of artificial intelligence for skin cancer diagnosis in the real world," *American Journal of Clinical Dermatology*, vol. 22, no. 2, pp. 233–242, Dec. 2020. DOI: 10.1007/s40257-020-00574-4.

[47] J. F. Ludvigsson and A. Ekbom, *Medicinsk statistik – diagnostiska tester*, 2021. [Online]. Available: https://www.internetmedicin.se/behandlingsoversikter/ovrigt/medicinsk-statistik-diagnostiska-tester/ (visited on 02/22/2022).

[48] J. A. Ostrowski, "Accuracy of 3 diagnostic tests for anterior cruciate ligament tears," *Journal of athletic training*, vol. 41, no. 1, p. 120, 2006.

[49] T. Tanaka, Y. Hoshino, N. Miyaji, K. Ibaragi, K. Nishida, Y. Nishizawa, D. Araki, N. Kanzaki, T. Matsushita, and R. Kuroda, "The diagnostic reliability of the quantitative pivot-shift evaluation using an electromagnetic measurement system for anterior cruciate ligament deficiency was superior to those of the accelerometer and iPad image analysis," vol. 26, no. 9, pp. 2835–2840, Oct. 2017. DOI: 10.1007/s00167-017-4734-0.

[50] M. Zhao, Y. Zhou, J. Chang, J. Hu, H. Liu, S. Wang, D. Si, Y. Yuan, and H. Li, "The accuracy of MRI in the diagnosis of anterior cruciate ligament injury," *Annals of Translational Medicine*, vol. 8, no. 24, pp. 1657–1657, Dec. 2020. DOI: 10.21037/atm-20-7391.

[51] H. de Rooster, B. V. Ryssen, and H. van Bree, "Diagnosis of cranial cruciate ligament injury in dogs by tibial compression radiography," *Veterinary Record*, vol. 142, no. 14, pp. 366–368, Apr. 1998. DOI: 10.1136/vr.142.14.366.

[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[53] *Deep learning assisted detection of glaucomatous optic neuropathy and potential designs for a generalizable model - scientific figure on researchgate*. DOI: 10.1371/journal.pone.0233079.g004.

[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai,

and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[55] *Matlab r2020b*. [Online]. Available: https://matlab.mathworks.com.

[56] Austrell, *CALFEM : a finite element toolbox : version 3.4*. Lund: Structural Mechanics, LTH, 2004, ISBN: 91-8855823-1.

[57] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7299064.

[58] *Pytorch documentation*, 2019. [Online]. Available: https://pytorch.org/docs/stable/index.html.

[59] H. Parwaiz, A. Q. Teo, and C. Servant, "Anterior cruciate ligament injury: A persistently difficult diagnosis," *The Knee*, vol. 23, no. 1, pp. 116–120, Jan. 2016. DOI: 10.1016/j.knee.2015.09.016.

A

Appendix 1

Listing A.1: Constructed CNN for concatenated damage maps.

```
class ConcatenatedInputNetwork(nn.Module):

    def __init__(self, channels=2):
        super(ConcatenatedInputNetwork, self).__init__()

        self.conv1 = torch.nn.Sequential(
            nn.Conv2d(channels, 8, kernel_size=3, padding=1),
            nn.Dropout(0.2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.conv2 = torch.nn.Sequential(
            nn.Conv2d(8, 16, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.conv3 = torch.nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.fc1 = nn.Linear(32 * 3 * 3, 32, bias=True)
        self.fc2 = nn.Linear(32, 2, bias=True)

    def forward(self, x):
        out = self.conv1(x)
        out = self.conv2(out)
        out = self.conv3(out)
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.softmax(self.fc2(out), dim=1)
        return out
```

Listing A.2: Constructed CNN for corresponding damage maps using a Siamese network structure.

```
class SiameseNetwork(nn.Module):

    def __init__(self, channels=2):
        super(SiameseNetwork, self).__init__()

        self.branch = nn.Sequential(
            nn.Conv2d(channels, 8, kernel_size=3, padding=1),
            nn.Dropout(0.2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Conv2d(8, 16, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.fc1 = nn.Linear(2 * 32 * 3 * 1, 32, bias=True)
        self.fc2 = nn.Linear(32, 2, bias=True)

    def forward(self, x):
        x1, x2 = torch.tensor_split(x, 2, dim=2)

        # Branch networks
        out1 = self.branch(x1)
        out1 = out1.view(out1.size(0), -1)

        out2 = self.branch(x2)
        out2 = out1.view(out2.size(0), -1)

        out = torch.cat((out1, out2), 1)
        out = F.relu(self.fc1(out))
        out = F.softmax(self.fc2(out), dim=1)
        return out
```

Listing A.3: Constructed CNN for single damage maps.

```
class SingleInputNetwork(nn.Module):


    def __init__(self, channels=2):
        super(SingleInputNetwork, self).__init__()

        self.conv = nn.Sequential(
            nn.Conv2d(channels, 8, kernel_size=3, padding=1),
            nn.Dropout(0.2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Conv2d(8, 16, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
```

```
            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.fc1 = nn.Linear(32 * 3 * 1, 32, bias=True)
        self.fc2 = nn.Linear(32, 2, bias=True)

    def forward(self, x):
        out = self.conv(x)
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.softmax(self.fc2(out), dim=1)
        return out
```

Listing A.4: Constructed CNN for generating CAMs from concatenated damage maps.

```
class ConcatenatedInputNetworkCam(nn.Module):

    def __init__(self, channels=2):
        super(ConcatenatedInputNetworkCam, self).__init__()

        self.conv1 = torch.nn.Sequential(
            nn.Conv2d(channels, 8, kernel_size=3, padding=1),
            nn.Dropout(0.2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.conv2 = torch.nn.Sequential(
            nn.Conv2d(8, 16, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )

        self.conv3 = torch.nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
        )

        self.fc1 = nn.Linear(32, 2, bias=True)

    def forward(self, x):
        out = self.conv1(x)
        out = self.conv2(out)
        out = self.conv3(out)
        out = F.avg_pool2d(out, 7).squeeze()
        out = F.softmax(self.fc1(out), dim=0)
        return out
```

Listing A.5: Constructed CNN for generating CAMs from single damage maps.

```
class SingleInputNetworkCam(nn.Module):

    def __init__(self, channels=2):
        super(SingleInputNetworkCam, self).__init__()

        self.conv = nn.Sequential(
            nn.Conv2d(channels, 8, kernel_size=3, padding=1),
            nn.Dropout(0.2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Conv2d(8, 16, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),

            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.Dropout(0.5),
            nn.ReLU(),
        )

        self.fc1 = nn.Linear(32, 2, bias=True)

    def forward(self, x):
        out = self.conv(x)
        out = F.avg_pool2d(out, (7, 3)).squeeze()
        out = F.softmax(self.fc1(out), dim=0)
        return out
```

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY