CHALMERS
UNIVERSITY OF TECHNOLOGY

Analyzing right ventricular attributes with machine learning

A deep learning solution trained on model-annotated data

Master's thesis in Master Programme Computer Science - Algorithms, language and logic

David Hagerman Olzon

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Analyzing right ventricular attributes with machine learning

A deep learning solution trained on model-annotated data

David Hagerman Olzon

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden 2020

Analyzing right ventricular attributes with machine learning
A deep learning solution trained on model-annotated data
David Hagerman Olzon

Supervisor: Richard Johansson, Department of Computer Science and Engineering
Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2020:NN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Quantifying right ventricular ultrasound images using a classifier trained on model-annotated data
A two-step method for increasing model performance using large volumes of un-annotated data
David Hagerman Olzon
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

An accurate analysis of echocardiograms is key to diagnosing heart conditions in modern medicine. This analysis is today performed by physicians and requires expertise that might only come after years of training in the area. In this master thesis we explore the possibility of using machine learning models, trained on large amounts of echocardigrams, that when trained can predict the mobility and size of the right ventricle. A successful model could be useful in assisting both inexperienced physicians during training or as a tool to speed up analysis for the already trained experts.

To train a supervised model, large amounts of annotated data is required but annotating the echocardiograms requires high expertise and is time consuming. As a possible solution to this problem, a separate text classifier trained on transcript data were used to annotate the echocardiograms that were linked to the respective examination. The transcript data consisted of a physicians analysis on all examination data that belonged to a patient, including the echocardiograms.

Several different architectures for the text classifier were trained and evaluated and the best performing model achived 92% accuracy on classifying the mobility and 95% accuracy at classifying the size of the right ventricle using transcript data. The trained text models were then used to annotate the echocardiograms in the dataset and the resulting data were then was used to train a set of image classifiers. The echocardiograms are 3-dimensional data and our results showed that models using a 3-dimensional representation also performed the best on the two classification tasks. The best models used a combination of human-annotated and model-annotated data and achieved 82% and 83% accuracy on mobility and size respectively. This result can be compared to the interobserver agreements between our transcript analysis and two experts annotating echocardiograms from the test set which were 82% for mobility and 72% for size.

The study showed that the use of a machine learning model as tool for physicians is a feasible and an interesting prospect. As a continuation of the work done here, the first step would be to increase the size of the models and training data. With the help of the automatic annotation, additional data is relatively easy to process. Other ways forward would be to include additional parameters from the examinations such as heart rate and to analyze the right ventricle attributes in relation to each other.

Keywords: machine learning, ai, bert, cnn, classification, echocardiography, resnext, ultrasound, deep learning.

Acknowledgements

First of all, I would like to thank my supervisor and examiner Richard Johansson. During the study, he has always been encouraging and continued to give prompt and thorough feedback on any questions or concerns I had during these six months. Eva Hagberg, thank you not only for annotating the data but for your expertise input as a medical professional and our interesting conversations. Ola Hjelmgren, thank you for your enthusiasm regarding the project, your kind hospitality, your input and help during these months. For the both of you and Sahlgrenska University hospital, thank you for making me feel welcome and as a part of the team there, I look forward to seeing you again this autumn. Finally, I would like to thank the open source machine learning community as the sharing of model architectures, datasets and techniques is in my belief one of the key factors to the successes in the area.

David Hagerman Olzon, Gothenburg, June 2020

Contents

List of Figures

List of Tables

Acronyms

3DConv 3D Convolutional Layer. 27, 28

BERT Bidirectional Encoder Representations from Transformers. 11–13, 19, 21–23, 32
BN Batch Normalization. 9

CNN Convolutional Neural Network. 7–9, 15, 16, 20, 23, 38
CPU Central Processing Units. 1

DL Deep Learning. 1

GIS Greatly Increased Size. 18, 34
GPU Graphical Processing Units. 1
GRM Greatly Reduced Mobility. 18, 32–34

IS Increased Size. 18, 19, 26, 33–35, 39

MIS Moderately Increased Size. 18, 34
ML Machine Learning. 1, 7, 11, 27
MRM Moderately Reduced Mobility. 18, 32–34

NI No Information. 18, 19, 33–35, 42
NLP Natural Language Processing. 1, 11
NM Normal Mobility. 18, 19, 26, 33–35, 38, 39, 41
NS Normal Size. 18, 19, 26, 33, 35, 39, 42

ReLU Rectified Linear Unit. 8, 28
RM Reduced Mobility. 18, 19, 26, 33–35, 38, 39

SGD Stochastic Gradient Descent. 7
SIS Slightly Increased Size. 18, 34
SRM Slightly Reduced Mobility. 18, 33, 34

1
Introduction

Cardiovascular disease is the leading cause of death in the world today [11] and improvements to the cardiological diagnostic process is therefore of great interest not only to the medical community but to the general public. Cardiovascular imaging (echocardiography) is one of the most common tools used in the diagnostic process and it has been shown that there is a link between the quality and usage of echocardiographic analysis and mortality rate [1]. As the interpretation of echocardiographic images requires high expertise in cardiology and has been shown to be hard even for trained professonals [3] [9], an automated interpretation process could both lead to more accurate results and a wider adoption. The use of machine learning in cardiology is far from widely spread but research in other medical areas and in general image processing has shown that the potential is there.

Machine Learning (ML) as an area has existed for several decades but it is only in recent years that its popularity has exploded. The popularity mainly comes from the many impressive results it has shown in several different areas. Image recognition models such as ResNet [54] is famous for being one of the first models with higher than human performance on a common image recognition benchmark. AlphaGo [17], a Deep Learning (DL) model, beat the world's best player in Go in 2016, a game considered by many as too hard for computers to learn because of its large branching factor. In Natural Language Processing (NLP), the few last years have seen a constant flow of record breaking models where models such as ALBERT [18] managed to reach higher than human performance on a standardized reading comprehension test. While some of the successes certainly can be attributed to clever algorithms, many techniques used are not novel but have existed for long periods of time. Instead many believe that it should be attributed to two other large factors: The rise of big data and the development of powerful specialized Graphical Processing Units (GPU). As ML algorithms learn from data, there is almost always a correlation between more data and higher performance and there is a constant growth of both data being generated and data being stored. GPU's are hardware units initially specialized for rendering images. In the act of rendering, a large amount of matrix multiplications must be processed and GPU's are therefore excellent at performing a large number of such operations in parallel, often outperforming Central Processing Units (CPU) by several factors. The most common operation in most ML algorithms is also matrix multiplication and the use of GPU's in ML models therefore enabled both larger models and longer training.
Echocardiography is especially well suited for the use of ML as the resulting images are both used to screen healthy patients and to diagnose patients with cardiovascular

disease. Combined with the fact that the resulting number of produced images per patient is high, this gives us a large dataset with a clear relation between image and output and a balance between both healthy and ill patients.

1.1 Aim

The overarching goal of this project is to explore the possibility of using a machine learning model to quantify the right ventricle attributes from echocardiographic images. In the process of doing so, this project also intends to assess the possibility of using data annotated by another machine learning model as training data.

A successful first goal is of high interest to the medical community as a well-performing model could be used to assist doctors with ultrasound analysis, both speeding up the process and producing a more unified result. While a complete replacement of a practical analysis might never happen, a successful model that offers suggestions during the analysis could both increase accuracy in hard cases as well as changing the doctors task to a sanity check which is faster compared to a full analysis.

Data annotation is a recurring problem in almost any supervised ML situation. In some cases those annotations already exist or are easy to gather but in the most common scenario those annotations have to be manually produced. While simpler tasks such as classifying if an image contains a cat or a dog can still easily be outsourced to external annotators such as AMT [19], this is impossible if the task requires a specialist or if the data cannot be shared because of legal reasons. In the case of echocardiographic analysis, both of these are true and annotation is therefore often a problem.

An automated solution to annotations is therefore of great interest and would be of interest to any supervised ML solution. Imagine that we have a dataset and in that dataset a smaller subset has been annotated by a specialist. A model trained on the subset that can correctly annotate the remaining data could then be used as an automatic annotator. Solving this problem is in the case of supervised learning is meaningless as if it was possible to create a model that could annotate that unseen data correctly, the full problem would already be solved and the remaining data would not be required.

In our case, we have two types of data, a transcript from a doctor analyzing a patient and the ultrasound video that belongs to that same patient. The classification problems, quantifying the size and mobility of the right ventricle, can be solved either by looking at the corresponding ultrasound video or reading the transcript. This means that a model trained on transcript data could generate predictions for the ultrasound videos and those predictions could then be used as annotations. For our annotations to be usable, the transcript model need to be well performing, or the incorrect annotations would make the ultrasound model hard or impossible to train. One then might ask what the point of solving the ultrasound image problem

is if we can create a well performing transcript model without the use of images. The answer is that such a model would be of little to no practical use. To get a prediction on the right ventricle features we would then first need a doctor to analyze the ultrasound images from the patient, feed his analysis into our model which would then generate its own analysis based upon his. Another way of looking at it is to recognize that the transcripts are simply annotations in another form, the transcript model then learns to extract those annotations into a form that our image model can use.

These are the main questions to answer in this project.

- What architecture and settings give the best performance for right ventricle classification on ultrasound videos?
- What architecture and settings give the best performance for right ventricle classification on medical transcripts?
- Does adding additional training data, annotated by a model instead of a trained professional, improve performance?
- How does the final model compare to the inter-observer accuracy of two trained medical professionals?

1.2 Limitations

While many attributes of the heart are interesting for cardiovascular health, this project strictly focuses on evaluating the mobility and the size of the right ventricle. To quantify those attributes, several different classes have been used over the course of the project, each representing a degree of mobility or size. The initial plan was to use four classes for each classification problem but a lack of data in two of the classes resulted in us limiting the classes to two at a later stage. See Table 4.1 for details.

In terms of data selection for training the models, certain limitations have also been put in place. Each examination in the complete dataset has several other types of data that has not been used but in theory could have been useful in an ensemble model. In an ensemble solution metrics such as heart rate, end diastolic volume or ejection fraction could have been used to improve perforce in both the text and image classification problems. Training an ensemble model takes significantly more time as you both need to train a separate model for the new data type and a final model that combines the output from your previous models. This would result in at least three times as many models to train by introducing a new data type which would come at the cost of testing new architectures and iteration upon those and was therefore decided against.

Regarding the architectures and their hyperparameters, limitations on which to test and what hyperparameter values to evaluate have also been used. Certain architectures that could potentially be very high performing such as GPT-2 [20] for text classification and Two-stream I3D [21] for 3D video classification have been excluded due to time constraints and would be interesting to investigate in the

future. The search space for hyperparameters have also been limited for many models if their initial results did not show promise.

1.3 Outline

In the first chapter, we have discussed why the problems we explore in this thesis are important to solve, how a solution could be used and the goals and limitations for the rest of the project.

- In Chapter 2 we will give a more in depth medical background to the problem together with an introduction to machine learning and the two main architectures used in the thesis.
- In Chapter 3 we present the implementation details regarding the models that have been used in the thesis. This includes architectural details, preprocessing techniques and training methods.
- In Chapter 4 we first give an overview of the previous work done in text and image classification in the medical field. After that, a more in-depth analysis of two closely related papers will be presented in separate sections.
- In Chapter 5 the results from both the text classifiers and the image classifiers that have been trained will be presented together with an analysis.
- In Chapter 6 the conclusions that we have drawn from the project will be discussed and presented.

2
Theory

In this chapter we first present a brief medical background on heart physiology and echocardiography to give the reader a better understanding of the different concepts used in those areas that we discuss in the later parts of the thesis. After that we present a technical background that first gives some background to the concept of machine learning and then dives deeper into the two main architectures that have been used in the thesis, the transformer and convolutional neural networks.

2.1 Medical background

A human heart is a double circulatory system that has four chambers: the right ventricle, the left ventricle, the right atrium and the left atrium. The atrium receives the incoming blood and prepares it for the ventricles which then pumps it out again. See Figure 2.1 for an overview. The right side receives deoxygenated blood and pumps it back into the lungs for re-oxygenation while the left side receives oxygenated blood and pumps it through the aorta, the main artery, into systemic circulation. If a ventricle has reduced mobility, this often means that it is unable to pump blood as efficiently as a normal heart and the heart would have to work harder to ensure that the entire body still receives its fill of oxygenated blood. One way for the heart to be able to pump more blood with reduced mobility is to increase the size of the ventricles. There are also other reasons for why the size of the ventricles might be increased in size such as genetic abnormalities, high blood pressure, disease or vascular resistance. In the worst case a ventricle is unable to pump enough blood which then leads to heart failure, an often fatal condition with a 35% risk of death in the first year.

2.1.1 Echocardiography

Echocardiography is the process of using ultrasound to produce an image of the heart. It is one of the most widely used diagnostic tools in cardiology. The images can give the cardiac physiologists a wealth of information such as pumping volume, size of the ventricles, tissue damage and hypertrophy. During a standard echocardiographic procedure, several images are often produced from different view angles to give as much information as possible as each view angle produces an image of a different cross-section of the heart. The four-chamber view, which is the focus in this study, show a cross section where each chamber can be seen clearly and is the one most commonly used to diagnose the right ventricle. See Figure 2.2.

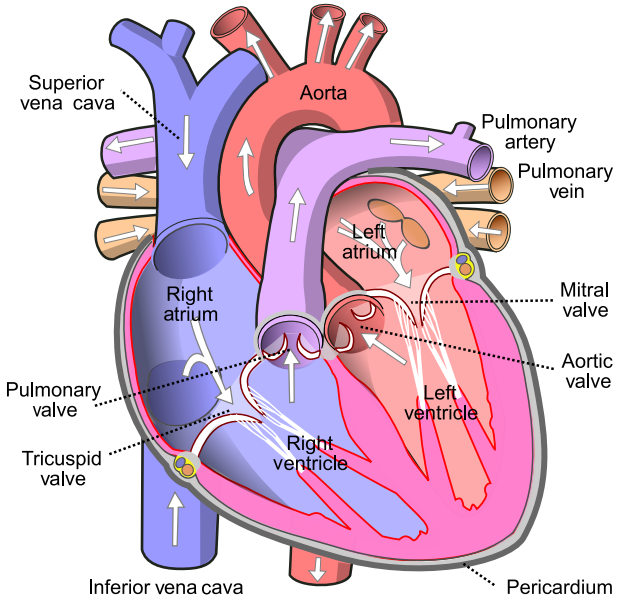

Figure 2.1: A diagram of a normal human heart. Created by Eric Pierce and licensed under Creative Commons Attribution-Share Alike 3.0 Unported.

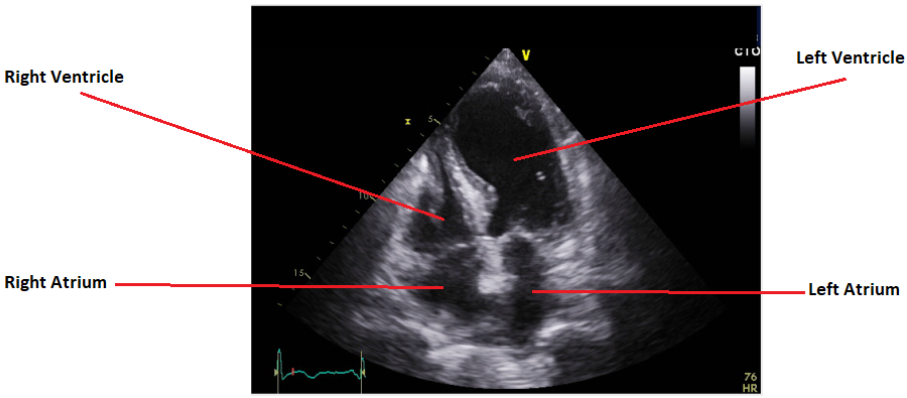

Figure 2.2: A labeled echocardiographic four-chamber view image.

2.2 Machine Learning and supervised learning

Machine learning is the combination of an algorithm and a dataset that produces a model that can closely mimic the relationships between data in the dataset. Such a model can then be used produce predictions or structures given new unseen data. The dataset consists of features and in the case of supervised learning, an output. The features have been created through feature extraction on raw data. As an example, if the raw data is an image, the features could be chosen as the pixel representation of that image and the feature extraction is then the conversion of the image to pixel data.

The algorithm that produces the model can be divided in to three main components: representation, evaluation and optimization. Representation is the space of the possible allowed models that our algorithm can produce. Common representa-

tions are instance models such as KNN and SVM, hyperplanes such as linear and logistic regression and neural network data representations. The second component is evaluation and is a function that given our model and data produces some kind of score that can be measured against previous versions of the model or a benchmark. This function is often known as the objective function, cost function or loss function and denoted by $J(\Theta)$. The type of function depends on what representation that was used for the output. For a discrete classification problem, the function most often used is cross entropy loss while in the case of continuous output, a common cost function is mean squared error. While the ultimate goal of the model is often an objective function such as accuracy, a separate objective function is used internally for the model that has better support for optimization such as being differentiable. The final component is optimization which is the method used to update the model parameters so that the output from the objective function gives us the highest reward (maxima) or smallest error (minima). These methods can be combinatorial such as greedy search, continuous optimization with constraints such as linear programming or without constraints such as gradient descent. The type of optimization method should be chosen in respect to the problem at hand but for more complex problems, stochastic gradient-based methods such as Stochastic Gradient Descent (SGD) [57] and Adam [22] are often selected.

Supervised learning is a subset of machine learning where the data in the dataset consists of an input and output pair. The goal for the final model is to be able to predict the correct output from new input data. The name supervised comes from the fact that the output in many datasets are curated by a human and which in that sense will be supervising the learning by telling the model if the output correct or not.

Most research in the area has been focused on the representation component as it is often the bottleneck for solving complex problems. It does not matter how well or how fast you can learn a representation of a problem if that representation is not close to reality. Depending on the sub-category of ML, different techniques and methods have been developed for better representations but one common factor among all branches is that as time has passed, models have become larger and deeper.

2.2.1 Convolutional neural networks

A Convolutional Neural Network (CNN) is a type of neural network architecture that is typically used for image recognition and classification but has successfully been used in other areas as well. The core of the architecture comes from using a convolutional layer that performs a convolutional operation on input data. In the following examples, 2-dimensional data has been used as input as it is the easiest to visualize but there is no restriction on the dimension that can be used with convolutions. To perform a convolution on input data a convolutional filter (also known as kernel) is required. The filter is a smaller matrix with the same number of dimensions as the input. The operation is done by sliding the filter over each part of the input data. For every step taken, an element-wise multiplication is performed and the values from the resulting matrix are then summed together to produce the output for that step. How far the filter is moved every step is controlled

by a parameter called stride. After all the input has been processed, the resulting matrix is called the feature map. See Figure 2.3 for an example using 5x5 input data, 3x3 filter and stride 2. In the example above, the weights in the filter form a cross. By looking at how the convolution is performed, one can see that if that filter is applied to a part of the input data that has the same shape as the filter, a high value will be produced in the output. A high value output could be understood as the filter giving a strong signal that the feature it represents has been found. For a single convolutional layer, several different filters are applied to the input, each representing its own feature and producing its own feature map.

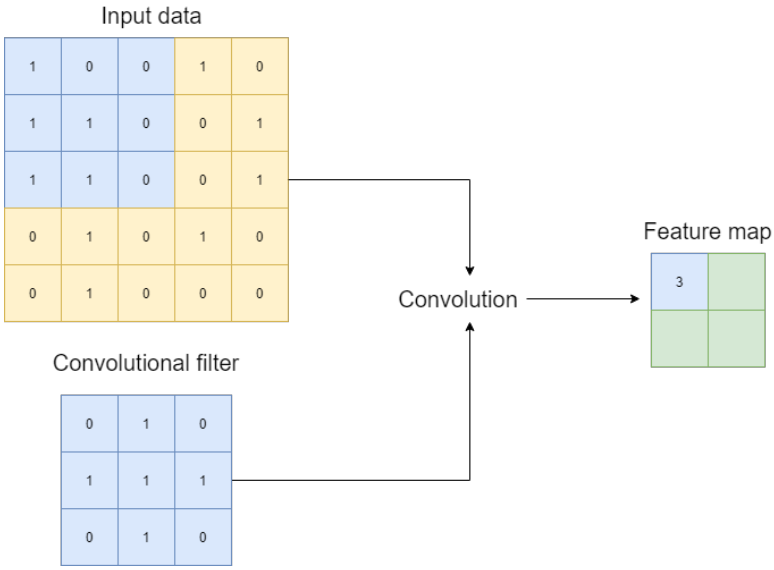

Figure 2.3: A convolutional operation. The filter in this scenario is using a stride of two which produces a resulting feature map of size 2x2.

It is important to note that these filters are not manually programmed but learned through standard optimization techniques. A convolutional layer also uses an activation function on its output, most commonly used a Rectified Linear Unit (ReLU). At this point, we are only using a single convolutional operation directly on our input data and the output will most likely not help us with our actual task unless we want to classify if an image contains a 3x3 cross. If our problem is instead more complicated such as identifying if an image contains a dog or a cat, a single layer of convolutional filters is not enough. By stacking several layers serially, the second convolutional layer with the feature map from the first layer as its input then learns to recognize a feature of a feature of the input and so forth. By stacking the convolutional layers it is therefore possible to represent complex features without having to use an huge amount of filters.

Even when stacking convolutional layers, the number of parameters in complex CNN architectures can still easily go out of control. To combat this problem, a second layer called a pooling layer is often used in between convolutional layers to downsize the output to a more manageable level. The most common type of pooling layer

and the one used in this thesis is called a max pooling layer. To perform it we first specify a spatial neighbourhood such as 2x2 and then similar to our convolution, we slide that pooling window over the feature map. In each location, we only take the max value which then becomes our output. See Figure 2.4 for reference. As the figure shows, applying a 2x2 max pooling operation with stride 2 on a 8x8 feature map produces a resulting output of 2x2, downscaling the resulting output with a factor of 4. Each convolutional filter produces its own unique output and therefore a separate pooling is applied to each filter. A layer with 8 convolutional filters followed by a max pooling layer using the same sizes as above therefore produces an output with the size 8x2x2.

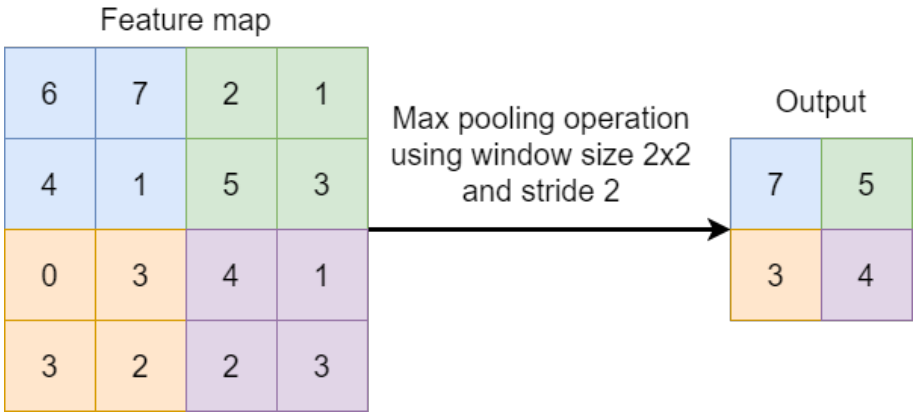

Figure 2.4: A max pooling operation performed on a feature map.

Another common building block in any modern CNN that has been used extensively in this thesis is the Batch Normalization (BN) layer. A BN layer can be used to improve performance, stability and speed for any type of neural network, not only CNN's. It is a relatively new technique that was introduced in 2015 by Sergey Ioffe and Christian Szegedy [26]. One problem when training a DNN is that the distribution of the input to one layer often changes as the parameters in the preceding layer change when the network learns. This phenomenon is known as internal covariate shift and it makes learning slower and more difficult. The BN layer is a way to combat this by normalizing the input to each hidden layer. A layer normally receiving the mini-batch input $X = \{x_1...x_m\}$ would with batch normalization receive $B_{\gamma,\beta}(X)$ as its input where $B_{\gamma,\beta}(X)$ is the batch normalization transform of X with the learnable parameters γ and β. The transform computes the mean and variance for the minibatch and then uses it to normalize the input. The learnable parameters are then used to scale and shift the normalized data which then produces the final output of the layer. When optimizing a network using a BN layer, the loss needs to be backpropagated through the transform and the gradients calculated with respect to its parameters.

The final layer in a CNN architecture is most commonly a fully connected linear

layer. This layer is used to produce a final classification based on the output from the previous layer. Going back to the example with images of cats and dogs, one can think of the previous layers as layers that learn to recognize dog ears, cat ears, dog tails, cat tails and so on. The final layer then adds it all up and learns to realize that if it we found a dog nose, a dog tail and a pair of dog ears in the picture, the image probably contains a dog. In reality, the filters don't learn human recognizable features such as the previous examples and it can be close to impossible to say what kind of feature a trained filter actually has learned to recognize.

3
Related Work

Machine learning in medicine is an area that shows great promise but with a low amount of practical implementation. This is not surprising due to the high requirements regarding safety and the gray zone of responsibility if an algorithm would cause harm to a patient. While machine learning has seen a large increase in popularity over the last 10 years, there has been attempts at at using automated computer systems for medical analysis as far back as the 70's [27]. In this chapter, we first give an overview of medical text and image classification research. Then a more in-depth look at some of the architectures that have been used in this paper and one of the more recent papers on ML in echocardiography.

3.1 Text classifiers

Classifying medical texts using ML has been done before. One recent example is a paper from 2017 by Mark Huges et al [58]. The researchers from that paper used a CNN to classify clinical texts with relatively good results. In another paper from the same year by Wei-Hung Weng et al [59] an RNN was used on a similar dataset. Both of those papers attempted to classify the texts into different medical subdomains such as cardiology or neurology. An even more recent paper from 2019 by Wang, Y et al [60] showed the possibility of analyzing texts for medical information. There a convolutional RNN was used to classify the texts into smoking status and the presence of a hip fracture. They evaluated several different architectures and found that a CNN was the best performing architecture for their problem and data. Another recent development in the area is BioBERT [61], a model based on BERT [23] that has been pre-trained on a large number medical texts. The model was designed for biomedical text mining tasks and outperforms the regular BERT on all fronts in that area. The model was considered for use in this project but believed to be less efficient than a BERT pre-trained on Swedish texts.

3.1.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) [23] is a state of the art language representation model created by Google AI in 2018. It makes use of the Transformer building block [24] that in 2017 showed that one could use a model purely based upon attention mechanisms and produce results superior to the previous best models in NLP. In Figure 3.1, an overview of the encoder Transformer block can be seen which is the main building block in the BERT architecture. While

the model used in the introduction to Transformers showed the high potential of the architecture, BERT took it to the next level with the help of different training techniques, a deeper model and pre-training on a large dataset.

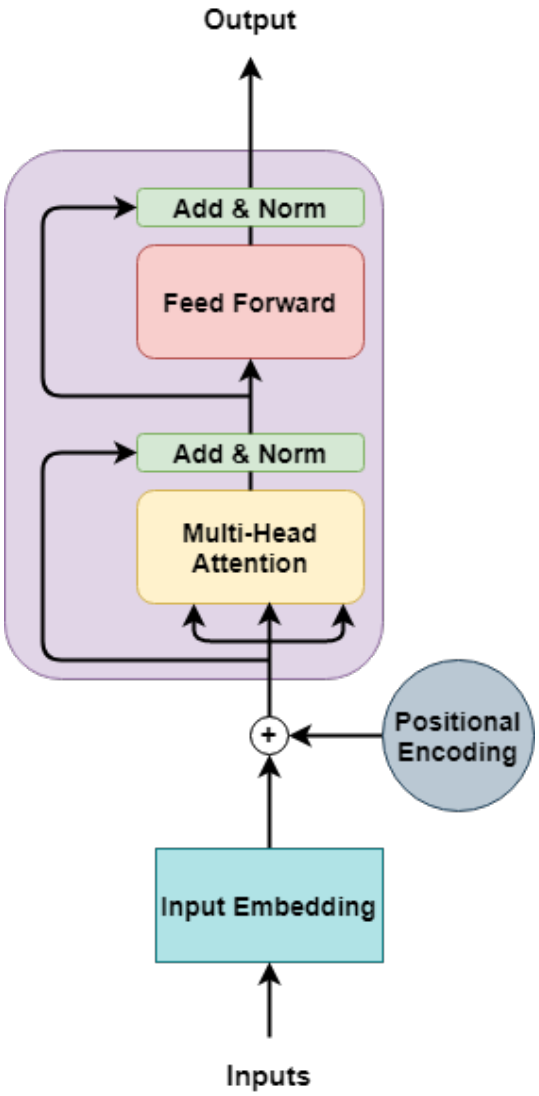

Figure 3.1: The encoder transformer block used by BERT. On subsequent levels, the input to the next encoder block is the output from the previous block.

One difference from many other language representations it that BERT uses a bi-directional representation of the text, looking for context in both directions compared to only looking backwards. To train BERT, two types of pre-training were performed using BooksCorpus [25] and the English Wikipedia as the dataset. The first type is used to train the bi-directionality of the model. It does that by masking out several words in the middle of a sequence and then the task is to predict the masked out words. The other type of pre-training is used to let the model understand the relationship between two sentences. The training picks out two sentences where the second follows the first. The second sentence is then replaced by a random sentence in 50% of the time. The model is then given the two sentences and is asked to predict if the second follows the first. After this pre-training, BERT can then be

finetuned on a number of different tasks such as question/answer, classification or sequence tagging by giving it the appropriate input and output training data.
BERT has two different models, $BERT_{BASE}$ and $BERT_{LARGE}$. The architecture in both are close to identical. Both consist of sequentially ordered Transformer encoders where the input to the first encoder is a combination of token, positional and segment embeddings and the input to the remaining encoders is the output of the preceeding encoder. See Figure 3.2. The only difference between the two versions is in the number of encoders used, where $BERT_{BASE}$ uses 12 and $BERT_{LARGE}$ 24. In this thesis, only the $BERT_{BASE}$ architecture has been used.

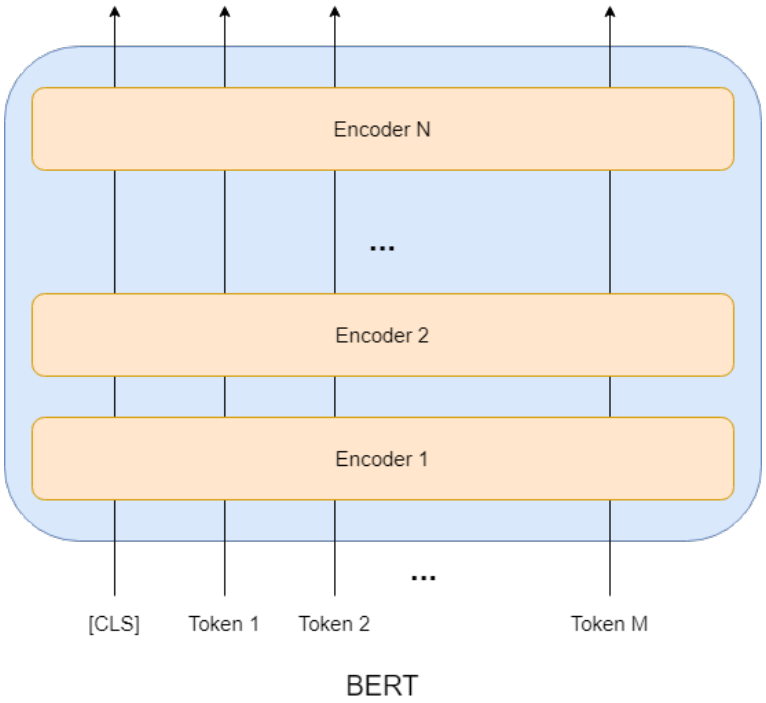

Figure 3.2: The general BERT architecture. For $BERT_{BASE}$, $N = 12$ and for $BERT_{LARGE}$, $N = 24$

3.2 Image Classifiers

Using machine learning to analyze echocardiographic images is relatively new. In 1979, a Fourier analysis [28] on echocardiographic images showed that automated decision making was feasible. After that, many different expert systems were produced such as [29] from 1992, the deformable models [30] in 1996 and an automated endocardial contouring algorithm from 2012 [31]. In recent years, the increase in popularity of machine learning in the world also seems to have spread to the medical research community. The first paper that explored the possibilites of machine learning on echocardiographic images is a paper from 2016 [32] where they evaluated a support vector machine model and k-nearest neighbour model on 15 echocardiography variables. After this paper, many others soon followed. Sukrit Narula et al [33] showed a model that could differentiate between left ventricular hypertrophy in

athletes versus patients with hypertrophic cardiomyopathy. The model they used was trained on data from 139 patients and was an ensemble of a support vector machine, random forest and an artificial neural network. In a more more recent study [34] from 2019 a model was trained on 50000 echocardiographic images of the 2-chamber and 4-chamber views to predict the ejection fraction (EF) of the left ventricle. The model used a combination of linear regression and Bland-Altman analysis and produced very good results.
In all previous work the focus has been on the left ventricle, general heart function, chamber segmentation or view classification. From what we have seen, our study is the first attempt at analyzing right ventricle attributes.

3.2.1 Deep learning interpretation of echocardiograms

One of the most recent papers in this area is called "Deep learning interpretation of echocardiograms" by Gorbani A et al [35]. Their dataset and the methodology used is similar to what has been explored in this thesis which is why a more in depth description and comparison will be made. In their study, their models were trained on several other heart attributes: Pacemaker leads, dilated left atrium, left ventricle hypertrophy, left ventricle systolic volume, left ventricle diastolic volume and ejection fraction.

The dataset used in their study came from 3223 patients where data from 373 patients was used as a test set. The echocardiographic data from those patients was first fed through a view classification model. Initially the authors intended to use every available view but after problems with convergence and training times, a single view, the apical-four-chamber view, was selected to be used used for the training. The view classification model used in their study was from another recent paper [36] with similar performance to the model used in this thesis. After the initial view selection of videos had been made, a pre-processing step was then performed by sampling 20 frames with 10 fps from the original dicom videos and then rescaling those frames into a size of 299x299. The total number of frames (images) in the training set was 1,624,780. The dataset size can be compared to our unmodified training set of 204,520 frames for mobility and 178,880 frames for hypertrophy, the different sizes being due to the fact that not every examination in our dataset had an annotation.

The architecture that was used to train the models on each task was Inception-Resnet-v1 [37] and used no pre-trained weights as it was shown to give no increase in performance. The Inception-Resnet-v1 is a 2D architecture and the models were therefore trained on single frames and predictions were made with a summed average over all frames. During training, significant data augmentation was applied to each frame as it was loaded which gave a minor but reliable performance increase. The data had geometric transformations applied to them such as flipping, rotations and translations. Pixel data was also augmented with the help of changes to contrast,

saturation and brightness.

Their models performed relatively well but differed highly depending on the task that was given. Several tasks were clearly easier for the models to reach a high performance on such as if there is a pacemaker lead (AUC = 0.89), the sex (AUC = 0.88) or left ventricular end systolic volume ($R^2 = 0.74$). Other tasks such as height ($R^2 = 0.33$) and ejection fraction ($R^2 = 0.5$) were clearly harder.

While the datasets, preprocessing and data augmentation are similar with a few differences, the major differences in the studies come from the choice of architecture and classification problems. In their study a 2D model was selected while our best results used a 3D representation of the data. The writers do not mention why a 2D representation was selected over 3D but one can theorize that it is due to the large number of already existing high performing 2D architectures as finding the right architecture is a often a large study in itself.

It is possible that the choice of a 2D representation is reflected in their results as the tasks that the model performed well on are time invariant. For example the presence of a pacemaker lead will be the same regardless of the order or the length of the frames. The same can be said regarding end systolic volume which is the minimum volume of that ventricule. A minimum volume for a ventricle would be the same regardless of the time the minimum was found. Ejection fraction, which is the fraction of blood in that ventricle being pumped out, gave the model a harder challenge. It could be because the amount of blood being pumped out is a function of the volume of the ventricle, how much it contracts, expands and how fast. By looking at an individual frame, most of those factors are impossible to evaluate and to do so we need to look at the relations between the previous frames and the current frame. Because of this, the problem is highly time-variant and a 2D representation is therefore most likely not sufficient to reach great results.

3.2.2 ResNext 3D

In this thesis we used a customized version of ResNext-101 [38] made specifically for video classification problems, denoted as ResNext 3D in this thesis. This customized architecture comes from the paper by Kensho Hara et al [39]. In that study, they tried to replicate the success of 2D CNN's in three dimensions using those two dimensional architectures as base.

A 3D version of ResNet18 was first trained on 4 separate video datasets, UCF101 [40], HMDB-51 [41], ActivityNet [42] and Kinetics [43]. This showed significant overfitting on all datasets except the Kinetics dataset which then became the primary focus for the remainder of the paper. The Kineticts dataset consists of 306,245 videos. Each of the videos show a human action such as cooking, each action is a

class out of the 400 unique classes used in the dataset. Each class has at least 400 videos showing that action and each video is around 10 seconds long.

After the initial evaluation of the datasets, several high performing 2D architectures were then converted into 3D versions and trained on the Kinetics dataset. The conversion was made by replacing each individual 2D layer with a 3D layer of the same relative size. For example, a 2D convolutional layer with 128 filters of size 3x3 was converted into a 3D convolutional layer with the same amount of filters that had a size of 3x3x3. The study focused on different sized ResNet [54] architectures as well as extensions to the ResNet architecture such as WideNet [46], ResNext [38] and DenseNet [47].

The results showed that ResNext 3D was the highest performing model by a small margin on the test set used in the study. Both a top-1 and a top-5 accuracy metric is used in the study. This is because of the non-exhaustive annotation used in the Kinetics dataset. A video might for example show a person brushing teeth while driving a car but the video will only be classified as one of those two actions even if both are the right answer. This is similar to ImageNet [16] that suggests using a top-5 metric with the same reasoning, an image might include several objects but will only be annotated with one class. The top-1 accuracy for the ResNext 3D was 65.1% and top-5 was 85.7%, both respectable considering the high number of classes. The models were then also tested on two of the previous datasets that were excluded from training, UCF101 and HMDB-51. ResNext 3D were once again the top performer with a 90.7% accuracy on the UFC101 dataset and 63.8% accuracy on HMDB-51. These results were then compared to prior state of the art method for classifying 3D data using both 2D and 3D representations such as C3D [44], P3D [45] and two-stream CNN [21]. The comparisons showed that the 3D architectures outperform even complex 2D architectures, specialized in 3D classification.

4
Methods

In this chapter we first present general information regarding how the different models have been trained and evaluated and the full dataset used in the thesis. This is then followed by a section on text classification with a subsection for every model that has been trained on that classification problem, the data used to train them and the preprocessing done on that data. The fourth section is regarding the annotation of images in two ways, first with the help of a view annotation model, secondly with the help of a text classifier. The final section describes the image classification task, the details regarding the different models, the data that was used and the preprocessing that was performed.

4.1 General training and evaluation

Several general techniques were used on every classification problem and model. For any model, validation on the test set ran at the end of every epoch with all layers and gradients frozen in evaluation mode. If the chosen performance metric was higher than the previous maximum, the model weights were checkpointed and saved as the current best model. If the validation loss had not been lowered for several epochs, training was stopped. At the end of the training, the checkpointed best model was loaded into memory and used for any end of training metrics and evaluation.

Performance of the different models has been evaluated on a test set, a subset of the entire dataset that was set aside and not used during training. The performance a model has on that test set gives an indication on how good the model performs on new unseen data. Any reference to model performance in this and following sections therefore refers to performance on the corresponding test set unless otherwise specified.

4.2 Data

The full dataset that has been used in this project consists of 12504 examinations made on 6252 healthy and 6252 unhealthy patients. Each examination is identified with an anonymized userID tag. Every examination contains many different types of data such as heart rate, age and sex, but in this project only two types of data have been used. Those are a transcript of the analysis from the doctor's examinination and the echocardiographic videos of the heart. Each examination had a different

number of echocardiograhic videos that belonged to it, the total number of videos for all examinations is 244572. More details regarding the transcript data can be found in section subsection 4.3.1 and the video data in section subsection 4.5.1.

4.3 Text Classification

The tasks for the text classifier were to classify medical transcripts from a cardiovascular diagnosis on right ventricle size and right ventricle mobility. Each classification task was done by a separate model with no relation to the model doing the other task. Both classification tasks initially used 6 classes to describe the different levels of size and mobility of the right ventricle, see table 4.1. The initial plan was to use four classes but the last two classes (4 and 5) had to be added to capture transcripts that did not fit the first four classes. Those were transcripts that either described the reduction in mobility or the increase in size in very general terms or transcripts that did not describe them at all. After results on all models showed poor performance in relation to the classes with low membership, the classes were then instead reduced to 3. See the next section and Table 4.3 for details.
Several architectures were evaluated for the classification tasks and their details are described in the coming sections. They were all trained on the same training data and evaluated on the same test set, hidden from all models until evaluation.

Class	Right ventricular mobility	Right ventricular size
0	Normal Mobility (NM)	Normal Size (NS)
1	Slightly Reduced Mobility (SRM)	Slightly Increased Size (SIS)
2	Moderately Reduced Mobility (MRM)	Moderately Increased Size (MIS)
3	Greatly Reduced Mobility (GRM)	Greatly Increased Size (GIS)
4	Reduced Mobility (RM)	Increased Size (IS)
5	No Information (NI)	NI

Table 4.1: Classes used by the text classifiers

4.3.1 Data

This dataset is a subset of the entire set of 12504 examinations. It consists of transcript data from 1561 patients that had been examined for possible heart related diseases. The transcript is the final analysis of the patient from the diagnosing doctor. Each transcript is a string of text and to each transcript two class labels were attached, one for mobility and one for size. The labels were added to the data by a medical professional trained in cardiology that analyzed the transcripts using the annotation software Prodigy [48]. The text in the transcripts are in Swedish and contains heavy use of medical jargon, numbers and measurements, see an artificially constructed example below which are similar to the real transcripts. The actual transcripts cannot be shared due to privacy laws.

"Dilaterad vä kammare med lätt nedsatt systolisk funktion och generell hypokinesi. Skattat PA tryck 50-55 mmHg. Höger kammare utan anmärkning. EF ca 45%. Slagvolymer är normala av doppler att döma. Fyllnadstryck går ej att bedöma. Normalstora förmak. Normalt CVT."

The full dataset was balanced in terms of healthy and sick patients but not in terms of the classes defined for the classification problems. The initial size of the dataset was 1197 data points but after the class imbalance was revealed, a second selection of data was made with the help of the best performing BERT model that had been trained on the initial dataset. The trained model was fed all un-annotated transcripts and then predicted classes for them. The 364 data points that had the highest output for class 2 and class 3 for size and mobility were then selected for a second round of manual annotation. For the full class distribution in the dataset see Table 4.2.

	Classification problem			
Class	Size (Initial)	Size (2nd sel)	Mobility (Initial)	Mobility (2nd sel)
Class 0	676	726	684	714
Class 1	120	158	154	176
Class 2	0	9	17	65
Class 3	21	66	23	47
Class 4	141	210	222	335
Class 5	239	392	97	224
Total	1197	1561	1197	1561

Table 4.2: Class distribution of the annotated transcript data

After initial training results showed poor performance on the classes that had low membership even after the second selection, the data was instead divided into 3 classes instead of 6. All classes describing either a generic or specific deviation from normal were consolidated into one single class. See Table 4.3 for for details.

Class	Right ventricular mobility	Right ventricular size
0	NM	NS
1	RM	IS
2	NI	NI

Table 4.3: The three classes used by the final text classifier

4.3.2 Preprocessing

Several steps were taken convert the transcripts and labels into data that could be used to trained the different classifiers. For the labels, a simple lookup table was used to convert the strings into their numerical representations as seen in Table 4.1. The BERT models used their own pre-trained tokenizers to convert the transcripts

into numerical data while the linear classifier and the CNN were trained on transcripts that had been converted to n-grams.

To convert transcripts into n-grams a three step process was used. First each string was tokenized, the stop words and punctuation was then removed before feeding the text into the NLTK [49] tokenizer. Secondly the tokenized strings were converted into n-grams using the NLTK n-gram method. Finally a vocabulary was generated from all n-grams that had been created. In terms of n for the n-grams generated, n-grams of one up to five were tested in initial testing where it showed that 2-grams and 3-grams were far superior to 1-grams, 4-grams and 5-grams with 2-grams being the best performing.

The data was then split into a test set and a training set where the test set consisted of 15% of the data and the training set the remaining data. The split was done using stratified sampling to ensure that both the training set and the test set had the same class balance. To perform stratified sampling the dataset was divided into six groups, one for each class. Then random sampling without replacement was performed on each separate group until 15% of the data in each group had been selected. The remaining data in each group was used as the training set and the sampled data was set aside as the test set. A static seed value for the sampling was used during iterations to ensure that the test and training set stayed the same.

After the split, upsampling was performed on the training set to ensure that each class was equally weighted. The upsampling was done by finding highest membership count among all classes and then duplicating data in all classes until the number of members in that class was the same. This introduced no new data to the training set but ensured that the frequency that every class would be seen by the model was equal among all classes.

4.3.3 Linear classifier

The linear classifier was a single linear layer with an input size equal to the length of the n-gram vocabulary and an output size equal to the number of classes. Deeper models with several linear layers separated by ReLU layers were also tested but did not improve performance. The model was trained for 50 epochs and the loss was calculated using cross entropy loss. The weights were updated using the Adam optimizer with a learning rate of 2×10^{-4} and a weight decay parameter of 5×10^{-2}. The size of the model was relatively small which meant that a batch size of 128 could be used to train and validate it.

4.3.4 CNN classifier

The CNN model that was used was based on a paper by Yoon Kim from 2014 [50]. This model uses three 2D convolutional layers with 128 filters and filter sizes of 3,4 and 5 respectively. Each 2D convolutional layer uses a ReLU activation function and a max pooling layer that has the same size as the text to reduce the size of the

output. Before the input is sent to the convolutional layers, it is first fed through an embedding layer with an input size equal to the size of the vocabulary and output size equal to 128. After the input has passed through the embedding layer and all three convolutional layers, it is then fed into a linear fully connected layer with the output size equal to the number of classes. The model was trained for 30 epochs and as with the linear classifier, cross entropy was used to calculate loss and the Adam optimizer to update weights. The Adam optimizer used a learning rate of 1e-3 and a weight decay parameter of 1e-3. A batch size of 2 was used as improvements to performance could be seen with a lower batch size.

4.3.5 BERT

BERT [23] is a highly sophisticated ML architecture that makes use bi-directional training of the transformers building block [24] and were described in section 3.1.1. The BERT model comes in two different sizes, a 12-layer version and a 24-layer version. All our models were trained on the 12-layer version because of memory constraints. The architecture was imported from the HuggingFace Transformer library [51] and used a version specifically designed for sequence classification, see Figure 4.1.

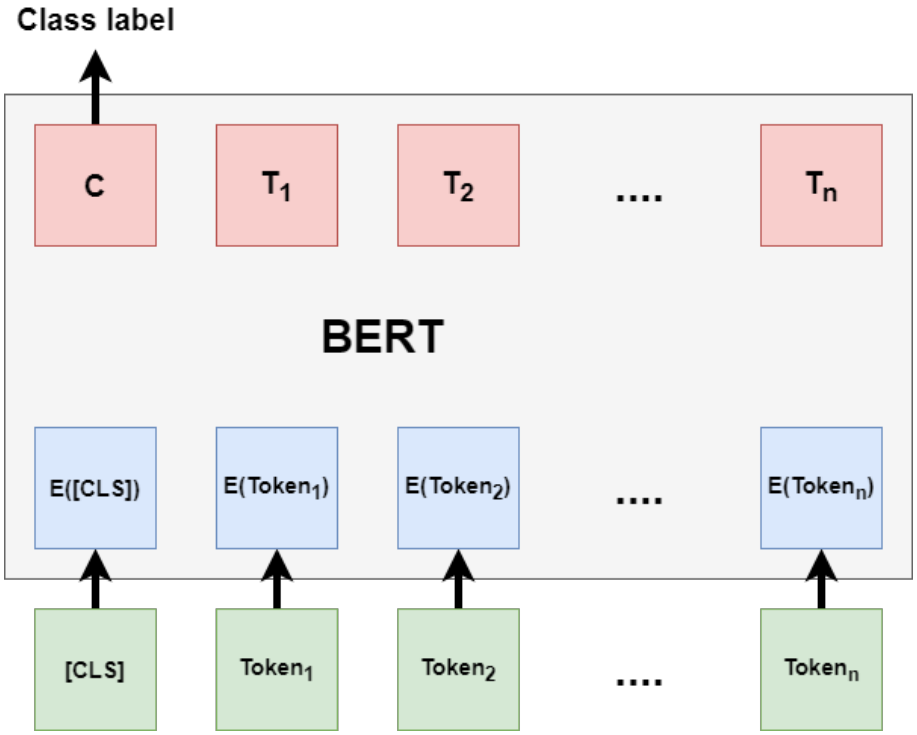

Figure 4.1: A BERT model finetuned for sequence classification. The [CLS] token is a special token added in front of every input.

The library offers several versions of pre-trained weights and two of those, "bert-base-multilingual-cased" and "bert-base-swedish-cased", have been tested and evaluated in this project. The models were trained for 25 epochs with a batch size of 8, a learning rate of 2e-5 and an epsilon of 1e-8 using the AdamW optimizer [52]. The

AdamW optimizer is also from the HuggingFace library and is similar to the normal Adam optimizer from PyTorch but with a different implementation of weight-decay. It should be noted that in appendix A.3 of the BERT paper [23], batch size, learning rate and epochs to train are suggested for fine-tuning BERT with new data. A lower batch size than the suggested 16 or 32 was used as it was otherwise impossible to hold both input data and model in CUDA memory at the same time. As BERT has a max sequence length of 512, longer sequences have to be truncated or split into several. This was avoidable in our situation as every transcript had a shorter length. All suggested learning rates were tested and evaluated together with other both lower and higher learning rates and 2e-5 was found to be the most optimal. The largest deviation from the suggested hyperparameters was in the number of epochs that was trained. Initially the model was trained for only 2-4 epochs but the performance was extremely low. Increasing the training length also increased the model performance up to the maximum found at 25 epochs.

4.4 Model annotation

Two kinds of annotations performed by a trained ML model were performed and used in this project. The first task were to annotate the different dicom videos belonging to a single examination into the different view angles the videos were recorded from. The second type of annotation is to annotate the same examination with the correct right ventricle mobility and size, based on transcript data. The following sections describe those in more detail and in Figure 4.2

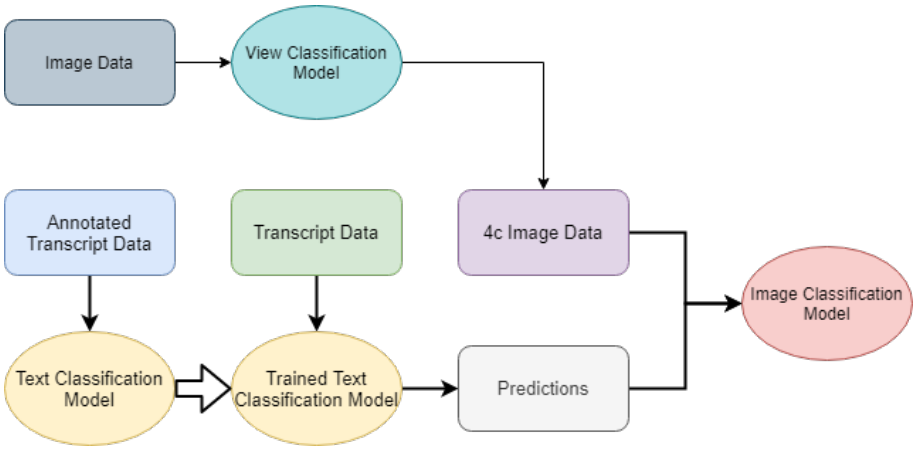

Figure 4.2: The project process. Flow of data as input and output to the different models used.

4.4.1 View annotation

In each examination several ultrasound videos are produced from different view angles. The different view angles highlight different structures and features of the heart. When analyzing the right ventricle, the primary view angle that is used is the four chamber view. See Figure 2.2. While it is technically possible to train

a model on all different view angles at the same time, it is unlikely that the view angles showing no information regarding the right ventricle would have no adverse effect on training. The original dicom files don't contain data regarding what view angle the video was recorded from and the data therefore had to be classified into the different views before usage.

To classify the views, an already existing model that had been trained on a subset of the same data was used. The model was developed by Elin Björnsson and Jan Liu [13] and is a CNN based on the ResNet34 architecture. As we were interested in one view in particular, the most important factors was the model's recall and precision for that class. We wanted to find as many of the actual 4-chamber view videos as possible as they would make up our dataset. At the same time, another view being incorrectly classified as a 4-chamber view would give us irrelevant data and had to be avoided as well. Their research showed that model performed extremely well at classifying this particular class which we also could confirm with sampling. Based on their test set results, their model managed to find almost every single 4-chamber view with very low confusion in regards to other classes with one small exception, the r view. The r view is according to the medical professionals we consulted an extremely similar view that shows the same parts of the heart with a slight variation to angle. According to them, it could be used to classify the right ventricle as well and would often be used in conjunction together with the 4 chamber view.
The trained view classification model was fed all 244572 ultrasound videos. The output of the model was fed through a Softmax layer to convert them into probabilities. For each examination, the instance ID of the file that had the highest probability of the 4-chamber view was saved together with the actual class prediction and the probability.

4.4.2 Transcript annotation

After all text classifiers had been trained on the annotated data, the best performing model (BERT-Swedish) was selected to annotate the remainder of the examinations with right ventricle size and mobility. Each transcript was fed into the trained BERT model and the maximum output used as class prediction and saved to file together with the user ID and instance ID. The annotations were then used together with the image data to train image classifiers, see next section.

4.5 Image Classification

The task was to classify right ventricle size and right ventricle mobility from multi-frame images (videos) that were generated from echocardiography. The initial plan was to use the original four classes that were also used in text classification (class 0 to 4). After looking at the class balance of the BERT annotations and some initial tests with four classes, the amount of data in class 2 and 3 was deemed too small to train a high performing model. The classification problem was therefore restructured to a binary classification problem with the classes normal and not normal. Several architectures were evaluated for the classification tasks. They were all trained on

1. Filter files on view annotations
2. Filter files on transcript annotations
3. Read full-size videos
4. Conversion to grayscale
5. Rescaling and padding size
6. Downscaling fps
7. Truncating and padding length
8. Downscaling video size
9. Save processed video on disk
10. Down/Upsampling

Figure 4.3: The pre-processing done before training in order.

the same training data and evaluated on the same test set, hidden from all models until evaluation.

4.5.1 Data

The data consisted of 12622 videos in Dicom format, which is a medical imaging data format and the corresponding labels for mobility and size. A dicom file contains the video data together with medical data, administrative data and metadata. The video data was saved with three channels (RGB) but shot in different sizes, with different length and varying frames per second (fps). In terms of size, a very large majority was shot in 484x636 with only a few outliers. The length of the videos had a much larger variance and there was no standard that could be found. Regarding the fps, a large variance could also be found here but a majority was captured in 25 fps.

4.5.2 Preprocessing

Significant pre-processing on the dicom files was done prior to training, see List 4.3. This was both to reduce file size to a smaller size and to reduce the required file-processing operations that had to be performed during training. The smaller size made us able to save the videos on an SSD-drive instead of a larger mechanical drive. The faster disk, smaller file size and the lowered number of operations that ran before loading data into CUDA reduced the wait for data during training to a negligible amount.

Before any video processing was done, the files were filtered on view annotations and transcript annotations. While the view annotations showed the instance ID of the video that had the highest probability of being a 4-chamber view, it was not certain that it predicted it as the 4-chamber view. This could be because not all examination had a corresponding 4-chamber view or because it confused it with another view. A selection here was therefore done to only choose the videos that the view classification model had predicted as either the 4-chamber view or the r-view, the reasoning behind also including the r-view being that it is extremely similar and also could be used to analyze the right ventricle.

After the files had been filtered on views, the remaining files were filtered on transcript annotations. Here, all examinations being classified as 5 in the 6-class case or 2 in the 3-class case were excluded from the data processing. Those classes corresponded to "no information" in regards to information regarding the right ventricle in the transcript and could therefore not be used as data. As each examination had two different transcript annotations, one for mobility and one for size, this meant that separate filters were used for the different classification problems and resulting in two separate datasets. In this process, we also paid attention to whether the annotation was human made or model generated. The videos with human made annotations were set aside and a balanced test set of 300 videos was selected from that subset of data as we wanted to avoid the risk of using data with incorrect model-generated annotations for evaluating our models. The remaining videos in that subset not used for a test set were used at a later stage in training to evaluate performance using training data from a combination of model and human annotated data. All model annotated data was used as a training set. After this process, the mobility dataset consisted of 300 videos in the test set and 10226 in the training set. The size dataset consisted of 300 videos in the test set and 8944 videos in the training set.

The first step after filtering the video data was to convert the RGB (3 channels) videos to grayscale (1 channel). The echocardiography that was used produces videos in grayscale but there are some colours from the added video data that are not from the ultrasound such as the heartrate graph. As the colour of the heartrate graph is not of any relevance to our classification task, no important data was lost in this conversion. The actual grayscale conversion was done by multiplying the video data matrix with [0.2989, 0.5870, 0.1140] which are standard luminence values for most video and TV systems.

Most of the videos used the same standard size for the frames but for those that didn't, scaling and padding was then applied. For frames smaller than the standard size, zero-padding was done by adding pixels around the borders of the frame with 0 as pixel data. For frames larger than the standard size, the resize method from the OpenCV library [53] was used to rescale each frame down to the required size.

Each video was then downscaled to the same fps. The initial fps values that were used to record the videos had to be reduced because of memory constraints. A standard was also chosen because it meant that each frame in each video was of equal length in terms of milliseconds. While no tests were run to confirm this, we speculated that training could be harder if the frame real life length differed for each video. The standard fps was selected to 12 fps, half of the most common fps value among all videos. For videos with an fps higher than 12, the processed video contained of frames selected with equal intervals from the original video, the remaining frames were dropped. For videos with lower fps than 12, frames were instead duplicated until the target fps was reached.

1. Image Centering
2. Grayscale Normalization
3. Gaussian Noise
4. Brightness
5. Vertical Transpose
6. Horizontal Transpose

Figure 4.4: The data augumentation done during training in order.

As the length of the videos had a large variance, a standard of 20 frames was set as the models require input with uniform dimensions. This was chosen based on an estimate of the mean length and the target fps that was selected for fps rescaling. Any video with a length longer than 20 frames was truncated down to the 20 first frames. The videos that had a length shorter than 20 frames were instead padded by adding additional frames with 0's as pixel data.

Finally each video frame was rescaled to a lower resolution that was 35% of the original size making the final frame size 169x222. This was so that both model and input could fit inside CUDA memory. After this step, the resulting video was saved to disk as a numpy array.

Early iterations on the original distribution of the datasets showed poor performance due to the lack of data in class RM and IS and separate versions of the datasets were therefore constructed for training using downsampling and upsampling of data. These techniques enforce class balance in the training set but at the expense of a smaller training dataset for downsampling and longer training times when using upsampling. For iterating on hyperparameters and architecture, the downsampled version was used. This dataset was constructed by sampling datapoints from the common class (NM or NS) until the amount was equal to the amount of members in the rare class (RM or IS). This ensured balance but excluded a large part of the data from the common class. Once the best architecture was found, training was performed on the upsampled version of the original dataset. This dataset was constructed by duplicating data in the rare class until the amount was equal to the amount of members in the common class. Again, this ensured balance but at the expense of a dataset close to twice the size.

4.5.3 Data Augumentation

Several types of data augumentation techniques were applied every time a mini-batch was read from disk during training, see List 4.4. The use of data augumentation increases the effective size of the dataset and reduces overfitting.
The first step in the data augumentation process is to center the image on the right ventricle. This is not actual data augumentation as it was done in the same way every time the process ran. Instead, the operation was performed to reduce memory usage and data size without affecting the relevant image data. In every image, the right ventricle was positioned in the top left corner. The image was therefore roughly centered on the right ventricle by removing the bottom 20% (33) rows and

the right 35% (77) columns of pixels. This was far from a perfect centering but more of a rough trim of the sides where we knew that the right ventricle never appeared. While this did not directly increase performance, it enabled us to run larger models which in turn improved performance.

The second step was to normalize the grayscale values to values between 0 and 1. As standard grayscale ranges between 0 and 255, the pixel values in each image were then divided by 255 to produce the transformed image. Scaling down the possible valules to this range both improve performance and time to converge for most ML models.

The third step was to transform the image by adding Gaussian noise. The transformation was done by creating a matrix of equal size to the image matrix. Each element in the matrix was populated with a number drawn from a Gaussian distribution with mean 0 and variance 0.1. The image was then transformed by adding the Guassian noise matrix element wise to the image matrix.

The fourth step was to transform the image by reducing or increasing the brightness by a randomized amount. Here a single value was sampled from a Gaussian distribution of mean 0 and variance 0.1 and that value was then added to every element in the image matrix.

The fifth and sixth steps transformed the image by transposing the image vertically and horizontally. To do so, a random value between -15 and 15 was sampled with equal probability. The resulting value then determined how many pixels the image would be shifted. For example, a value of -3 in the vertical transpose transform meant that the image would be shifted 3 pixels to the left. The new empty elements after the shifts were populated with black (0).

The last three steps in the data augumentation were mainly done to reduce overfitting and performance on the test set as it ensured that the probability of repetition in training data was extremely low.

4.5.4 Custom CNN

One of the models that was evaluated for image classification is a custom built CNN architecture. The architecture consists of 4 convolutional blocks in serial connection followed by a fully connected layer at the end. Each convolutional block has four layers in the same order: A 3D Convolutional Layer (3DConv), a batch normalization layer, a leaky ReLU [55] activation layer and finally a max pooling layer. Two versions of this architecture were trained and evaluated, one using 3D layers and one using 2D layers with the input data flattened to 2D. The 2D version was relatively quickly discarded as inferior and therefore the focus in this section will be on the 3D version but results from the 2D version can be found in section 5.

The main difference between the blocks is in the convolutional layer that is used in that block. The 3DConv layer in the first block has 48 filters with size $7 \times 11 \times 13$, the layer in the second block has 48 filters with size $5 \times 9 \times 11$, the layer in the third block has 128 filters with size $3 \times 7 \times 9$ and the layer in the final fourth block has 128 filters with size $3 \times 5 \times 7$. All 3DConv layers used the same stride and padding, both equal to one. See Figure 4.5.

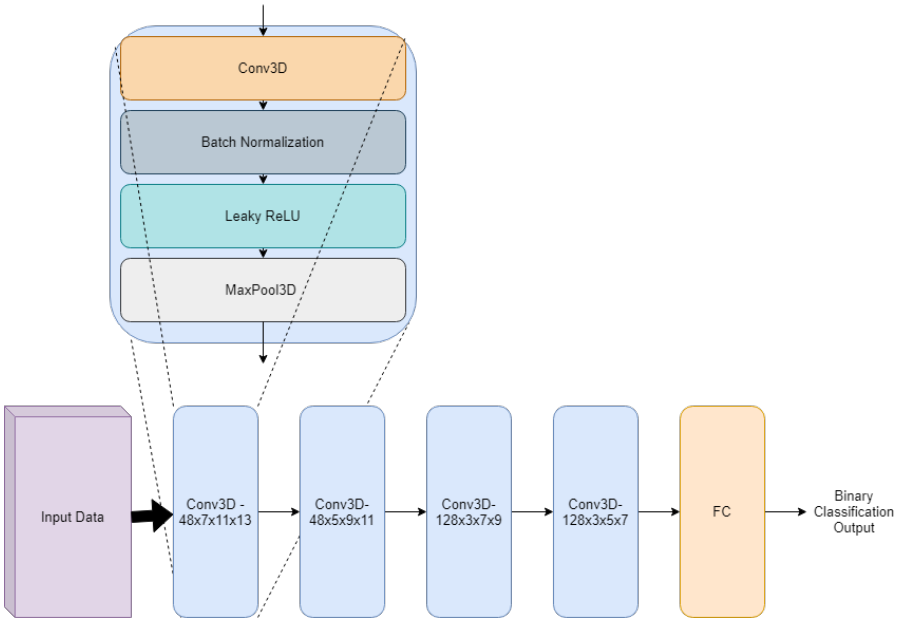

Figure 4.5: The architecture for the best performing custom built 3D CNN architecture.

The batch normalization layers were also identical apart from their input size which was dependent on the number of filters in the corresponding 3DConv layer. The ReLU and max pooling layers was the same for all blocks, the pooling layer using a $3 \times 3 \times 3$ kernel with a stride of 2 and padding of 1.
Multiple different versions of this architecture were tested and evaluated with changes to the number of convolutional blocks, number of filters, size of filters, stride and padding. The prior configuration was the highest performing one.
The model was trained for 30 epochs using cross entropy loss and the Adam optimizer with a learning rate of 1e-3 and a weight decay of 1e-4.

4.5.5 ResNext 3D

The ResNext-101 architecture described in section 3.2.2 was also evaluted. The implemented models were identical to the one in the paper except a few minor changes. The batch size was reduced to 16 and the number of classes, the output of the final fully connected layer, was changed from 400 to 2. Similar to the original model our model used a cardinality of 32 and 4 layers of bottleneck blocks. The ResNext 3D had pretrained weights available but they were not used as it was believed that they would not give any increase in performance due to the difference in task and data. Models were then trained on each of the separate classification

tasks. Similar to the other architectures, cross entropy loss and an Adam optimizer with weight decay 1e-4 and learning rate 1e-3 was used for optimization.

4.5.6 2D Image classification

In the field of image classification, a majority of the work and research has gone towards 2-dimensional problems. This has lead to 2D architectures being more sophisticated and tested while most of the 3D architectures are an existing 2D architecture restructured into an extra dimension such as the ResNext 3D that was presented in the prior section. Therefore, we have tested two different techniques for reshaping our 3D problem into a 2D problem so that we can make use of those well-performing models.

In both cases we have used the same 2D architecture and hyperparameters except for batch size. The architecture that was used was ResNet18 [54] with the final layer replaced with a fully connected layer that has binary output. As in the previous models, the model was trained using cross entropy loss and the Adam optimizer with a learning rate of 1e-3 and weight decay of 1e-4.

No pre-trained weights were used as tests showed no improvements to the model using them. The pre-trained weights were evaluated by loading the lower layers with the pre-trained weights and then freezing the gradients for those layers. The final layer was then trained for 20 epochs with 1e-2 as the learning rate. The lower layers were then unfrozen and the entire model was again trained for 30 epochs with a learning rate of 1e-3.

Additional preprocessing was done on the data for these models as they required a 3-channel input which meant that we could not use the same grayscale input as in our other models. Two solutions for that were tested, the first being to simply duplicate the grayscale channel two times, making each channel identical and using grayscale values. The second solution was to skip the grayscale conversion in the initial pre-processing step, making use of the fact that our dicom videos already came with 3-channels. No difference in performance between these two solutions was found and we therefore used the grayscale duplication method as it meant that we did not have to keep two different datasets on disk at the same time.
The pretrained models also expected the data to be normalized with a mean of [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225]. This normalization was kept even after we stopped using pre-trained weights as it seemed to help with performance.

4.5.6.1 Flattened

This variant of input reshaped the videos from 3D to 2D by flattening them in the height dimension. This meant that the new dimensions for a single video that previously had dimensions L*H*W*C (length, height, width, channels) after reshaping became (H*L)*W*C, see Figure 4.6.

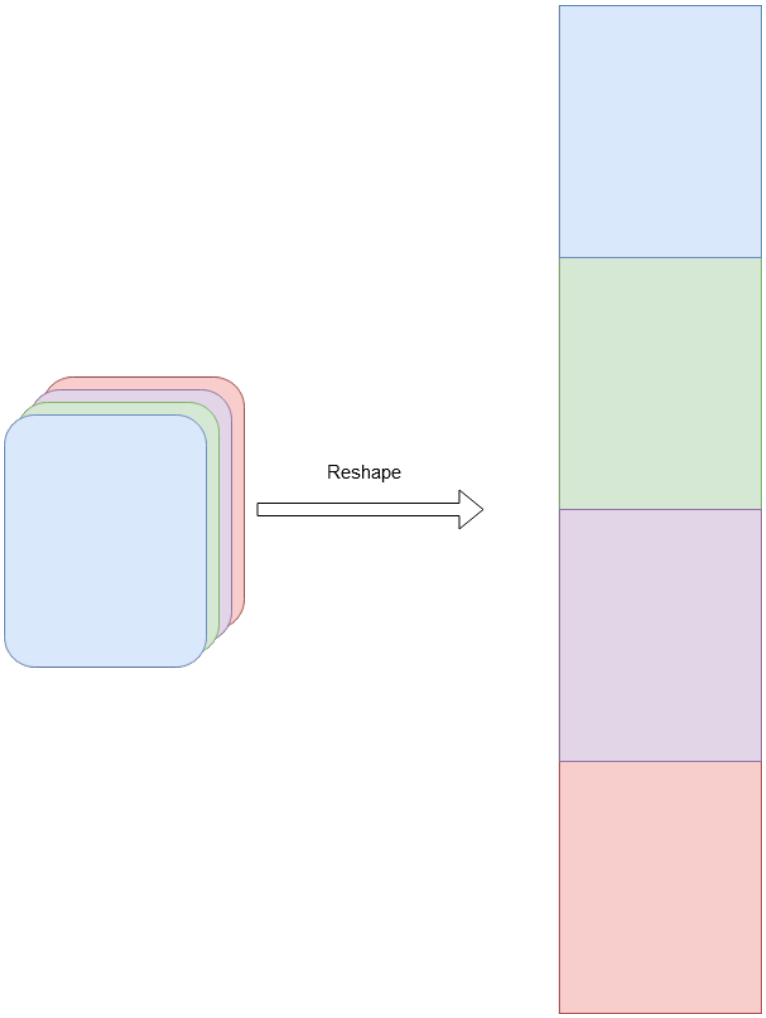

Figure 4.6: An example of a 4 frame video being flattened into a single 2D image.

4.5.6.2 Single Frame

The other approach used for training a 2D model on 3D data were to train the model on individual frames. This has been used successfully in at least one other machine learning solutions such as the models developed by Ghorbani A et al [35] that was discussed in section 3.
When training the models, a batch of B videos of size $L * H * W * C$ was first loaded from disk. The data was then restructured into a larger batch with shape $(B * L) * C * H * W$. Every frame from the same video used the same label. The models were then fed the larger batch of size $B * L$ as its input and trained. The trained model could then be used predict the class of a new video. To do so the data was loaded and restructured the same way as in training and fed into the model. The prediction was then generated by summing up the output from all individual frames that belonged to the same video and class with the maximum value was used as the model's prediction.

5
Results and discussion

This chapter will present the results of the thesis together with a discussion regarding them. Initially the results from text classification will be presented with a larger focus on BERT-Swedish as it was the most successful for text classification and therefore the model selected for video annotation. After that the results from the different image classifiers will be presented together with their analysis.

5.1 Experiment setup

All models were trained on a single computer using a Nvidia RTX 2080 with 8 GB VRAM. The results below are from the models that performed the best on the test set used for that classification task. The time it took to train the models before convergence were reached differed from model to model, with around 1 hour for the fastest and 8-10 hours for the slowest.

5.2 Text Classification

An overview of the results from all different text classifiers can be found in Table 5.1. A combination of F1 score and test accuracy was primarily used to gauge the performance of the classifiers but as shown in the table, they follow closely and there is no case where a higher F1 score does not also indicate a higher accuracy. The F1 score used here is the macro average over all classes: $F1 = \frac{F1_0 + ... + F1_n}{n}$, where n is the number of classes.

Problem	Model	F1 score	Average Test Accuracy
Size	Linear Classifier	0.49	69%
Size	CNN	0.64	77%
Size	BERT-Multilingual	0.86	94%
Size	BERT-Swedish	**0.89**	**95**%
Mobility	Linear Classifier	0.68	75%
Mobility	CNN	0.72	79%
Mobility	BERT-Multilingual	0.86	90%
Mobility	BERT-Swedish	**0.88**	**92**%

Table 5.1: Comparison of all text classifiers using 6 classes. The F1 score used here is the macro average over all classes.

5.2.1 Linear classifier

The linear classifier did not perform as well as we estimated at the beginning of the project. This is most likely due to the fact that it is limited by simply weighting each trigram. In many cases, the N-grams cannot capture enough context, for example the three words "kraftigt nedsänkt höger" could both come from the sentence "Rörlighet kraftigt nedsänkt höger kammare" and also from "vänster kammares rörlighet är kraftigt nedsänkt. höger kammare normal". Increasing the number of words included in the N-grams would capture more context but also make it harder to train as the number of N-Grams in the vocabulary would increase significantly because of combinatorics. Our tests showed that bigrams was the most optimal number of words to capture but with performance still not comparable to the more advanced architectures.

5.2.2 CNN text classifier

The results from the CNN text classifier were not particularly impressive but it's possible that changes to the representation of data and the architecture could have given a significant increase in performance. Instead of letting an N-gram model represent the words, we would instead use a simple vocabulary and let the model itself learn the important pairings. The N-gram model was chosen because of the good performance increase it gave the linear classifier. Regarding the architecture, the 2-dimensional convolutional layers were used and could have been changed to 1-dimensional layers, reducing the complexity. It is however our belief that the CNN text classifiers performance even after implementing those changes would have a very hard time reaching the performance level of BERT which is why our focus instead went towards improving and iterating upon the BERT model.

5.2.3 BERT

Of all the text classifier architectures tested in this project, BERT performed the best for both problems. Let us start by looking at the classification problems using 6 different classes. In figure 5.1 and 5.2, confusion matrices can be found for both mobility and size and a more detailed table of the results can be found in 5.2.

It is clear that the mobility model cannot classify GRM with great precision and this would lead to a high number of incorrect annotations for that class. This is the class that had the lowest number of data points for the mobility problem and we believe that the low precision is most likely due to the lack of data in that class rather than that specific class being harder to classify. With that in mind, it is surprising to see that the model was able to predict MRM with a relatively high precision even though it also suffered from a similar lack of data. This can be attributed either to luck as the number of datapoints in the test set is also low or it might be the case that this class is slightly easier to classify and it reached the threshold of data required to be able to learn how to predict it.

Shifting the focus towards size as our classification problem, we can see similar results. Here, both of the classes with the lowest precision are the two classes with the smallest number of data points and it is our belief that it is also the primary reason for it. From the confusion matrices for both mobility and size, we can see the the largest confusion comes from transcripts being incorrectly classified as either class RM, IS or NI. Regarding class RM, there is a clear overlap between that class and the classes SRM, MRM and GRM as a transcript describing the right ventricle as having a certain degree of reduced mobility also describes it as having reduced mobility in the general sense and the same can be applied to the corresponding classes for size. The transcripts incorrectly classified as class NI are often from the classes with the poorest performance. This makes sense as the models are not good at recognizing the information in transcripts that would classify them with the correct class, then class NI is chosen as no information could be found.

As it was clear that the models had problems classifying the rarer classes, we opted to reduce the number of classes as low performance in our text classifier would indirectly affect the performance of our image classifier. The three classes used can be found in Table 4.3 and in Figure 5.4 and 5.3. Confusion matrices can be found for the respective classification problems and a more detailed table of the results can be found in 5.3.

We can see that reducing the amount of classes removed much of the weakness from the models. Focusing on mobility as our first classification problem, the first result that stands out is the relatively low sensitivity for class NI. This means that several of the transcripts that contain no information are incorrectly classified as either normal mobility or reduced mobility. Looking at the confusion matrix we can see that the majority of them are classified as normal mobility and using the same reasoning as in the 6 class case, we believe this to be the actual case in reality. There is very little confusion between class NM and class RM and most of the remaining misclassifications comes from class NM or RM being classified as class NI. This means that a transcript with information about the right ventricle would be tagged having no information. As ultrasound examinations with a corresponding transcript annotation of class NI are excluded from the training data, this would mean that we would have slightly less data to train on because of these misclassifications.

The results from the size model are very similar to the mobility model but better. It does not suffer as much as mobility from low class NI sensitivity and all misclassifications there are tagged with class IS. The remaining misclassifications are almost all transcripts of class IS being classified as class NS. This would again lead to ultrasound videos of class IS being excluded from the training set of the image classifier. While the number of items being excluded is relatively low, as class IS is our least populous class, model performance most likely suffers more from items in that class being excluded compared to items in class NS.

Class	Problem	Precision	Sensitivity	Specificity	f1-score
NM	Mobility	0.97	0.97	0.98	0.97
SRM	Mobility	0.87	1.0	0.98	0.93
MRM	Mobility	0.91	1.0	0.99	0.95
GRM	Mobility	0.58	1.0	0.98	0.74
RM	Mobility	0.95	0.82	0.99	0.88
NI	Mobility	0.87	0.79	0.98	0.83
NM	Size	0.97	0.96	0.97	0.97
SIS	Size	0.96	0.92	0.99	0.94
MIS	Size	0.5	1.0	0.99	0.67
GIS	Size	0.83	1.0	0.99	0.91
IS	Size	0.94	0.91	0.99	0.92
NI	Size	0.93	0.93	0.98	0.93

Table 5.2: Performance metrics from the BERT models pre-trained on a Swedish corpus using 6 classes.

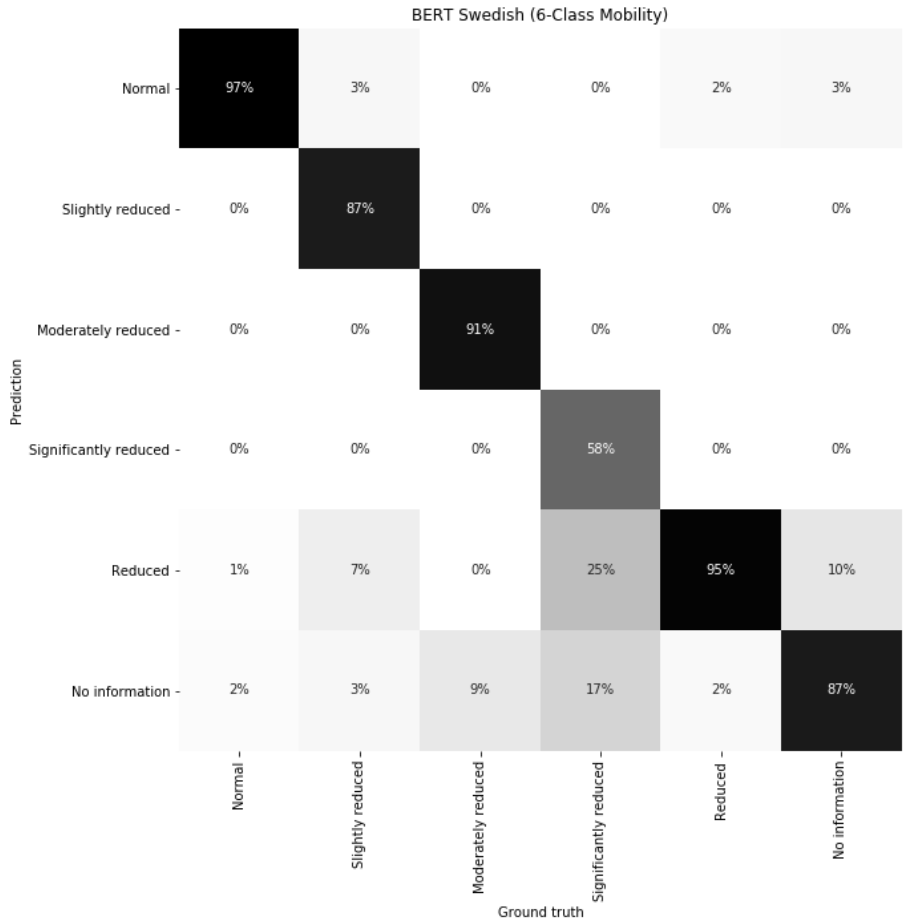

Figure 5.1: Confusion matrix of the BERT model pre-trained on a Swedish corpus and finetuned on transcripts using 6 classes for mobility.

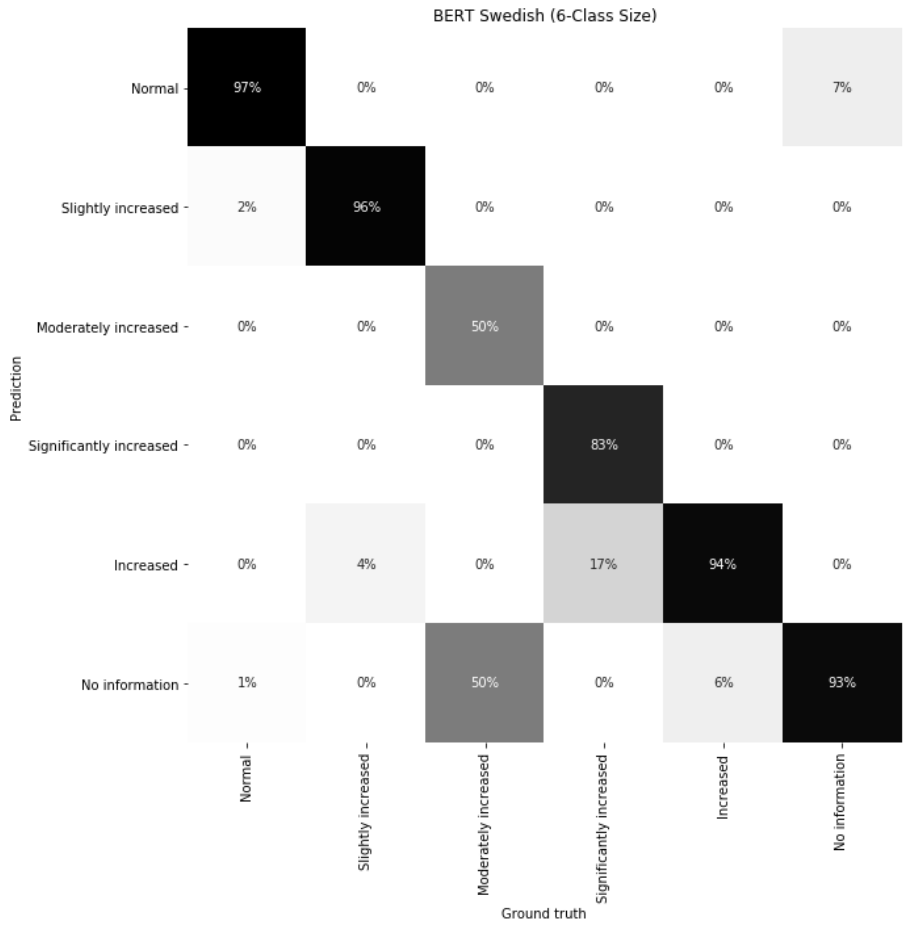

Figure 5.2: Confusion matrix of the BERT model pre-trained on a Swedish corpus and finetuned on transcripts using 6 classes for size.

Class	Problem	Precision	Sensitivity	Specificity	f1-score
NM	Mobility	0.97	0.97	0.98	0.97
RM	Mobility	0.95	99	0.96	0.97
NI	Mobility	0.90	0.79	0.99	0.84
NS	Size	0.98	0.95	0.98	0.97
IS	Size	0.94	0.98	0.98	0.96
NI	Size	0.92	0.92	0.97	0.92

Table 5.3: Performance metrics from the BERT models pre-trained on a Swedish corpus using 3 classes.

5.3 Image classification

The overall best results from all models trained on the image classification problem can be found in Table 5.4. As our test set was perfectly balanced in this scenario, we opted to use accuracy as our way of comparing separate models and architectures.

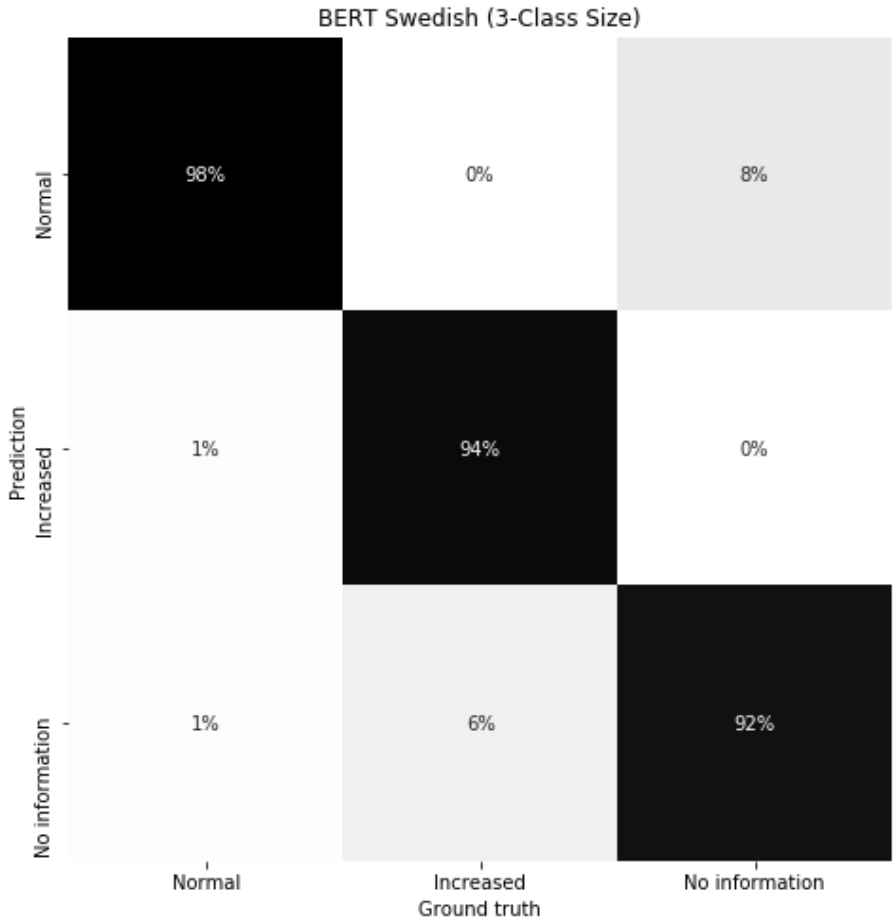

Figure 5.3: Confusion matrix of the BERT model pre-trained on a Swedish corpus and finetuned on transcripts using 3 classes for size.

Overall we can see that the models using 3-dimensional representations of the data performed better than the 2-dimensional models, in particular on the mobility classification task. It should also be noted how small the difference between the results from mobility and size is. It might reflect the similarity of the two problems: both look at the same ventricle and the mobility of a ventricle is related to change in size of that ventricle.

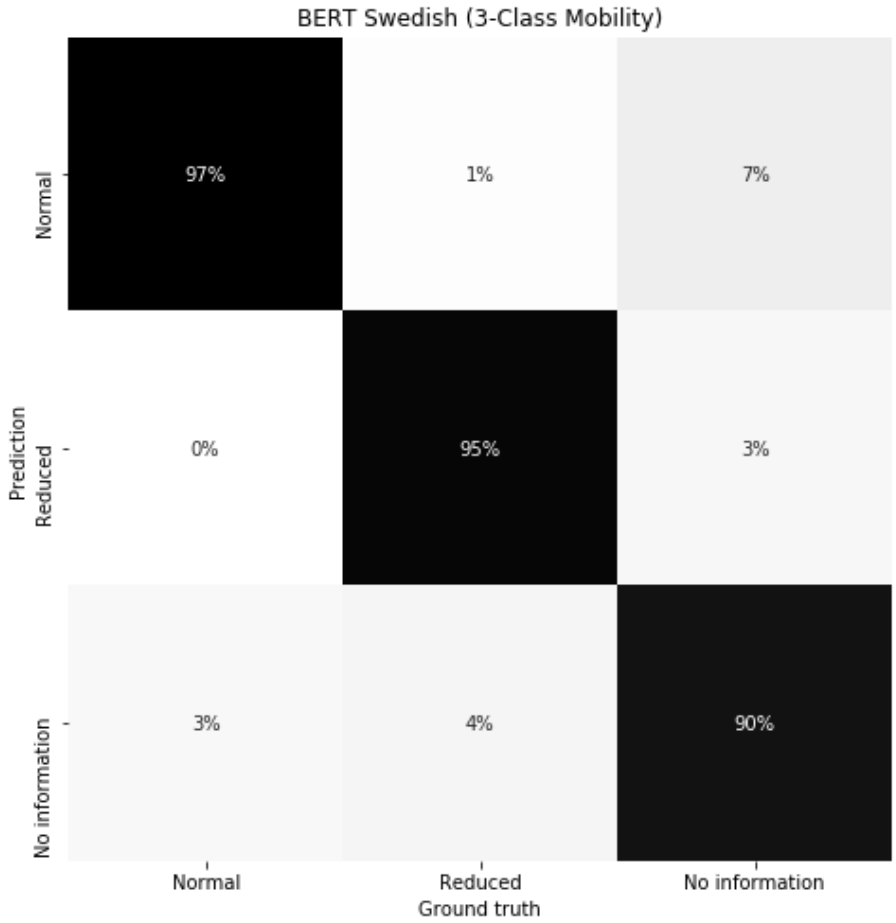

Figure 5.4: Confusion matrix of the BERT model pre-trained on a Swedish corpus and finetuned on transcripts using 3 classes for mobility.

Problem	Model	Average Test Accuracy
Size	ResNet18 (flattened video)	71%
Size	ResNet18 (single frame)	79 %
Size	ResNext 3D	80%
Size	Custom 3D CNN	**83**%
Size	Custom 2D CNN	73%
Mobility	ResNet18 (flattened video)	72%
Mobility	ResNet18 (single frame)	76%
Mobility	ResNext 3D	81%
Mobility	Custom 3D CNN	**82**%
Mobility	Custom 2D CNN	74%

Table 5.4: Comparison of the average test accuracy of all image classifiers using 2 classes

5.3.1 ResNet18

It is clear that training on a single frame was superior to training on a video flattened to 2D. It is possible that the resulting images after flattening are too large in relation to the filters. It should be noted that the single frame training accuracy actually was lower (ca 66%) than the resulting test accuracy. This shows that summing the results from multiple frames not only works but increases performance significantly. It is possible that by using more frames either by increasing fps or length, an even higher accuracy could have been reached. It is not surprising that size was the metric the single frame model performed best on as we believe size to be the least time dependent of the two. This is since the definition of size that the physiologists use relates to the maximum size of the ventricle at any given time.

5.3.2 ResNext 3D

The ResNext 3D architecture performed relatively well and is the other architecture that reached over 80% accuracy in models for both mobility and size. Comparing it to the ResNet18 single frame model, the ResNext 3D architecture is similar but uses 3D layers and the additional cardinality parameter. The difference in results from the two architectures was very small in the case of size. As discussed in the previous section, we believe size to be the least time dependent and it is therefore not strange that the models had similar performance. The difference here is probably rather due to the cardinality. Comparing the mobility architectures, we see that the difference between ResNet18 and ResNext 3D is much larger. Some of the difference should still be attributed to the addition of cardinality but a larger part can most likely be related to the 3D-representation used in the ResNext 3D model.

5.3.3 Custom CNN 3D

The custom CNN architecture with 3D layers was the best performer on both classification problems and we will therefore go more in depth when analyzing the results. The overall performance of the architecture compared to the other architectures was slightly higher but not extremely significant. During the iteration on the architecture, we could see that increasing the number of filters in especially the early layers led to significant performance increases. With more memory available, we believe that performance could be increased further simply by increasing the number of filters even further. Experiments that reduced the number of layers or the size of the filters for the sake of increasing the amount of filters were not successful and instead reduced performance. The current architecture had the best balance in terms of filter amount, filter size and depth in relation to the available memory. In Table 5.5, a detailed breakdown of the performance metrics can be found for the best trained models with this architecture. Focusing first on the mobility model, we can see that it has low precision and specificity but very high sensitivity in terms of class RM. This means that the model finds almost every case that has RM but incorrectly classifies some patients that actually belong to the class NM with RM instead. For medical purposes, high sensitivity for the class related to abnormal conditions is preferable as low sensitivity might lead to patients with actual health problems not

getting the help they need.

Shifting to size as our classification problem, the results are more tightly grouped across the board. For the class IS, we now instead have low sensitivity and high precision and specificity. While the differences are not as large as in the mobility case, this is not preferable as it could lead to patients with a right ventricle that has increased size being incorrectly classified as normal.

Class	Problem	Precision	Sensitivity	Specificity	f1-score
NM	Mobility	0.91	0.72	0.93	0.80
RM	Mobility	0.77	0.93	0.72	0.84
NS	Size	0.81	0.85	0.80	0.83
IS	Size	0.84	0.80	0.85	0.82

Table 5.5: Performance metrics from the Custom CNN 3D models.

5.3.3.1 Sensitivity Maps

To help visualize what the models had learned, sensitivity maps have been generated. A sensitivity map is created by plotting the gradient of the activation function with respect to the input. This can be understood as plotting the degree of change in output if in the input was changed slightly. As the change is different in different areas, the areas which are sensitive to change in input are of high importance to the final output. These sensitivity maps can be very noisy and we have therefore used a technique called SmoothGrad [56] to reduce the noise. The technique has two major components: adding gaussian noise to the input image and doing multiple samples of the sensitivity map, averaging the result. This reduces the noise in the produced sensitivity maps and highlights the important areas even further.

In Figure 5.5 the original echocardiographic image can be seen. Both the input and the output from the model is in 3D and the correct sensitivity map would therefore be a video and not a single image. In the Appendix, the full frame sequence can be found for the sensitivty maps and the inputs. As we center the images around the right ventricle, this is the actual input to the model and also the input that the sensitivity maps should be compared to. The centered version of the original image can be seen in Figure 5.6 (**A**).

In relation to this input the mobility model produced a sensitivity map which can also be seen in Figure 5.6 (**B**). There are two major areas that have strong influence on the result. The first one seems to line up with the wall between the right and left ventricle. The second area appears to correlate to the wall at the bottom of the right atrium. It is interesting that the model seemed to have learned to classify mobility by looking at the structures surrounding the right ventricle and not the right ventricle itself. It is unclear why the outer left wall of the right ventricle seemed to be of no importance.

Using the model classifying size, another sensitivity map was produced with the same input. The sensitivity map can be found in Figure 5.6 (**C**). Here the highlighted areas seem to correlate with the right ventricle, the wall between the ventricles and parts of the right atrium. The tricuspid valve and the mitral valve, the valves between the atriums and ventricles are also partly highlighted. Similar to mobility, the model have seemed to learn to relate size to the structures surrounding the right ventricle that shift and move when the right ventricle size changes.

While the exact highlighted areas differ both from frame to frame and from input video to input video, the one consistent theme that was found after analyzing several of these sensitivity maps is that the structures close to or related to the movement of the right ventricle often have a high gradient.

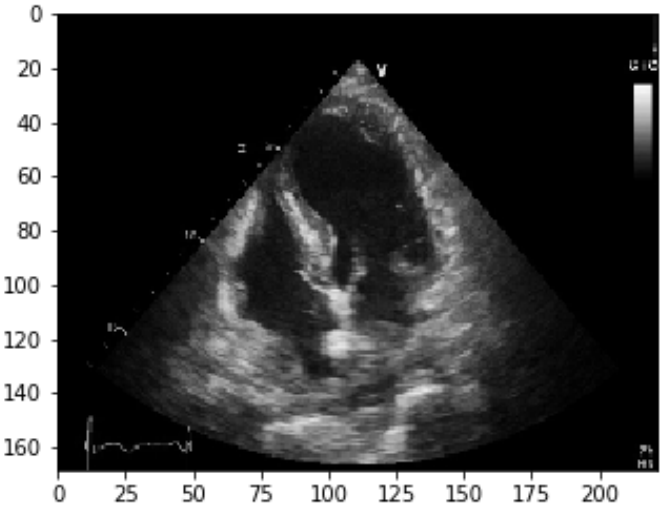

Figure 5.5: The original echocardiographic image.

5.3.4 Model vs Human annotated data

In this section we will compare the results of the best performing architecture (Custom CNN 3D) trained on only model-annotated data, only human annotated data and a mixture of both. In Table 5.6 the best accuracies using either human-annotated, model-annotated or a combination of both types of data as the dataset. In Figure 5.7 and 5.8, the best accuracies from training the architecture on model-annotated and human-annotated data plotted as a function of the size of the respective datasets can be seen. Included are also the best results from using a dataset with a combination of both types of annotations. By training on subsets of constantly increasing size, we can see how model performance changes with training data size for both model-annotated and human-annotated. While the best results are what we compare, it is important to know how that could change with the addition of more data.

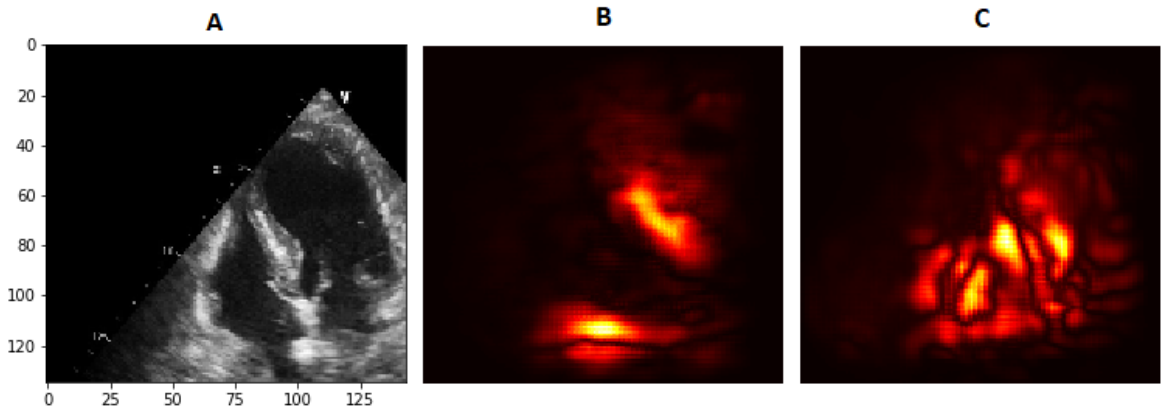

Figure 5.6: **A.** The echocardiographic image centered on the right ventricle after data augumentation. **B.** The sensitivity map for the Custom CNN 3D model trained on mobility using A as input. **C.** The sensitivity map for the Custom CNN 3D model trained on size using A as input.

From both graphs it is clear that the performance scores when using either type of annotation are relatively close. While the datasets are larger when using model-annotated data, it seems to be offset by the fact that some of the annotations are incorrect. We can also see that the increase in performance due to increases in training data have started to taper off, in particular for the size model. The models training on a dataset that uses data with both types of annotations are better in all cases. Upsampling did help slightly with performance for the size model but the downsampled version actually performed better for the mobility model. It is possible that the annotations for the added data, which mostly consisted of model-annotated data in class NM, were poor enough that the performance actually suffered because of it.

Problem	Annotations	Dataset Size	Average Test Accuracy
Size	Human	1561	80%
Size	Model	11061	78 %
Size	Combination	12622	**83%**
Mobility	Human	1561	81%
Mobility	Model	11061	80%
Mobility	Combination	12622	**82%**

Table 5.6: Comparison of the best average test accuracy of the Custom 3D CNN using a dataset with only human-annotations, only model-annotations or a combination of both.

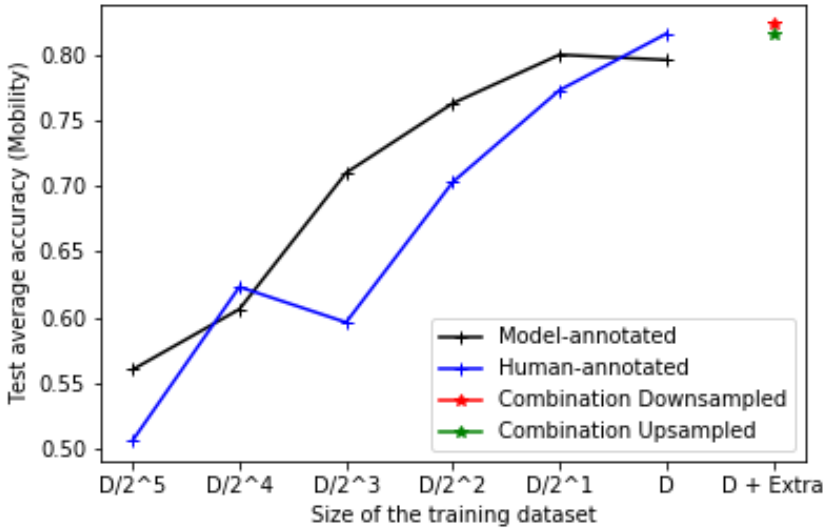

Figure 5.7: Best test accuracy for a Custom CNN 3D mobility model trained on different sized subsets of the full respective dataset. Also included in the graph are the results from the training set using a combination of both human-annotated and model-annotated data. D is the full size of the respective dataset, human-annotated or model-annotated.

5.3.5 Interobserver comparison

In this section we have compared the transcript annotations against image annotations done by medical professionals done directly on the ultrasound videos instead of transcript data and against model predictions.
Two medical professionals trained in cardiology, Ola and Eva, were given 100 ultrasound videos that had been human-annotated through reading the transcripts and asked to annotate the right ventricle size and mobility as either normal or abnormal based on the video content. The dataset was perfectly balanced in the case of mobility, 50/50, and roughly balanced for size , 60/40 in favour of class NS.
According to the specialists in cardiology that we talked to, the transcript annotations should be seen as the ground truth in in this case. The reason for this is because the transcripts are written by an experienced professional that has not only analyzed the echocardiographic videos from all different view angles, he also consulted all other related data in the examination. While the step of converting the transcript into an annotation does introduce a layer of insecurity, as the doctor annotating the transcripts could make a mistake when analyzing them, the reading comprehension task is was deemed extremely simple and therefore this factor should be small. The transcripts where there was any uncertainty in the annotation were classified with NI and excluded from test sets, training sets and this inter-observer analysis.
The results from comparing the transcript annotations against the two different medical professionals that annotated the ultrasound videos and the best performing models can be seen in Table 5.7. In Figure 5.9 and 5.10 the confusion matrices for

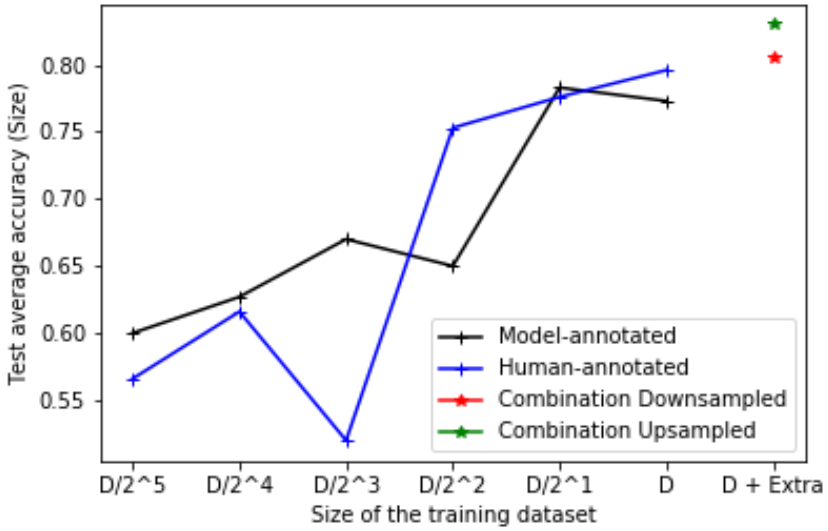

Figure 5.8: Best test accuracy for a Custom CNN 3D size model trained on different sized subsets of the full respective dataset. Also included in the graph are the results from the training set using a combination of both human-annotated and model-annotated data. D is the full size of the respective dataset, human-annotated or model-annotated.

the same agreements can be found.

We can see that the expert annotations done directly on the ultrasound video follow each other closely both in terms of agreement versus the ground truth and agreement between themselves. The fact that they agree with each other more than they agree with the transcript could be because the analysis made for the ground truth had access to more examination data. This means that when looking only at the images, Eva and Ola agreed with their assessment but it is possible that the assessment would have changed with full access to the examination data and bringing them closer to the ground truth. The second factor that could attribute to the relatively large agreement between Ola and Eva is due to the fact that Ola is one of the persons that trained Eva in echocardiographic analysis. This could give them a similar style and judging criteria when it comes to evaluating these metrics.

Comparing the expert agreement with the model agreement, we see that they are similar in terms of mobility but the agreement between the model and the ground truth is clearly higher when it comes to size. While we earlier mentioned that the lack of data could be one reason for the low agreement rate between experts and ground truth, the model and the experts used the same data when giving the predicts (only ultrasound videos) which reinforces that our model achieves higher than human performance on this particular task.

Interobserver	Problem	Agreement
Transcript (GT) vs Ola (Pred)	Mobility	83%
Transcript (GT) vs Eva (Pred)	Mobility	80%
Transcript (GT) vs Model (Pred)	Mobility	80%
Ola (GT) vs Eva (Pred)	Mobility	86%
Transcript (GT) vs Ola (Pred)	Size	72%
Transcript (GT) vs Eva (Pred)	Size	71%
Transcript (GT) vs Model (Pred)	Size	83%
Ola (GT) vs Eva (Pred)	Size	85%

Table 5.7: Interobserver agreement scores.

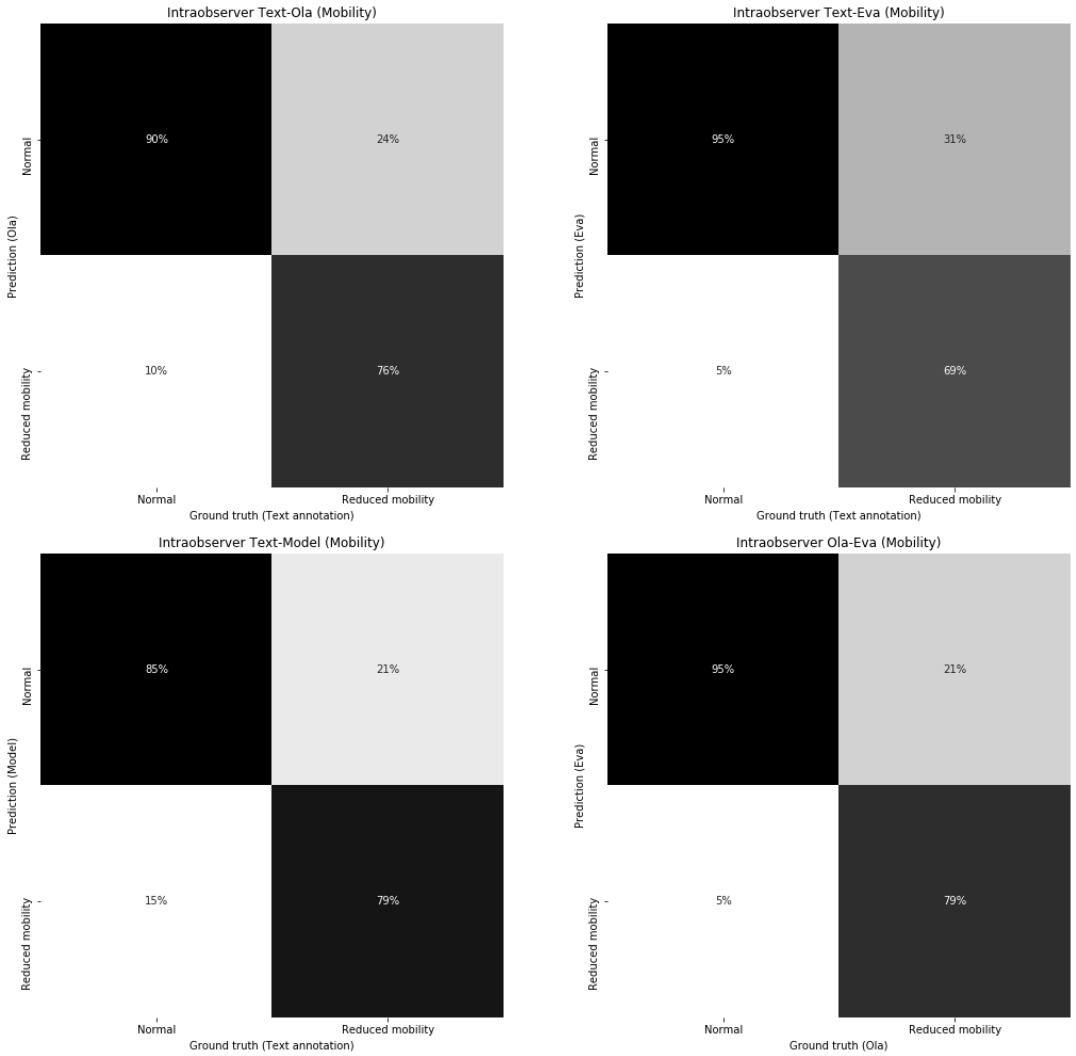

Figure 5.9: Interobserver agreement confusion matrices for mobility

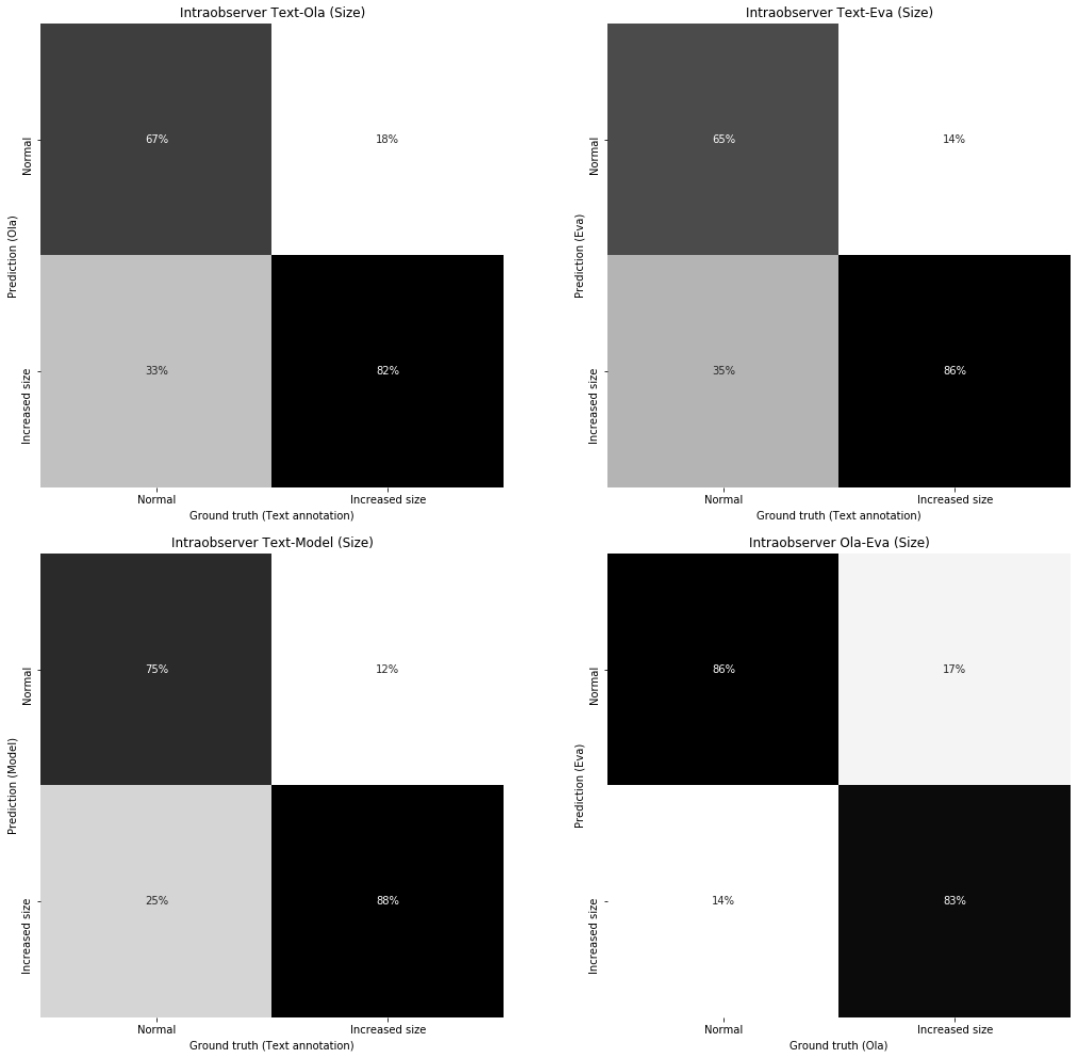

Figure 5.10: Interobserver agreement confusion matrices for size

6

Conclusion

6.1 Text Classification

The results showed that the BERT architecture is currently in its own class when it comes to text classification on this particular task. While the different pre-trained weights made a slight difference, the strong performance is most likely due to the architecture. The analysis of the transcripts, both by hand and by BERT, showed that the different classes that the medical professionals in theory used to evaluate mobility and size were hardly used in practice. Instead most right ventricles were either classified as normal or slightly abnormal in terms of size and mobility. While this could be because the doctors have a hard time classifying the right ventricle correctly, the more likely scenario is that the large majority of patients have a normal right ventricle, even if they have other health problems.

6.2 Image Classification

The results showed that 3D representations were superior when classifying 3D data in this particular case. The difference was smaller in the case where we believe the data to be more time invariant (size). The single frame 2D model performed surprisingly well on the size task and the technique should be taken into consideration depending on the classification problem at hand. The extra dimension in data also meant larger inputs which affected the size of the architectures and the amount of pre-processing that had to be done. Iterations were a constant battle against GPU-memory and it is likely that when using more hardware, performance could be significantly increased by using deeper models, more filters, higher fps or larger frames. The difference between human-annotated and model-annotated data was very slight but the combination of both produced the best results. We believe that it is no coincidence it was our size model that performed the best compared to the human interobserver agreement considering that it was our size text classifier that reached the highest accuracy. By iterating more upon the model-annotating text classifiers or increasing their training data, it is likely that the mobility image classifier would also have reached higher than human performance without any change to it's architecture. That being said, the performance of the classifiers trained only on human-annotated data were still very close to the performance of the best models and there is a lot of added work to gain those extra percentages of accuracy.

6.3 Future Work

A continuation of the work done in this thesis could be focused in several different directions. It would be interesting to see the 2-Stream I3D [21] implemented and trained on the training set as it has shown better performance than the ResNext 3D on most 3D classification public datasets. Including additional parameters and attributes from the examinations as extra input could lead to better models. One example that was considered for this project is the inclusion of heart rate. Heart rate could help especially the mobility models as the heart rate is one consideration when evaluating it. The interactions between the two different classification tasks could also be explored. Does increased size often imply reduced mobility? If we let the model know that the mobility is reduced, would that lead to a better prediction rate for size? Those are some of the questions that one could try to answer by conjoining models and data from the two problems.

Bibliography

[1] Alexander Papolos et al (2016) *U.S. Hospital Use of Echocardiography: Insights From the Nationwide Inpatient Sample*. Journal of the American College of Cardiology, 67(5):502-11.

[2] Anthony N.DeMaria et al (1979) *Systematic correlation of cardiac chamber size and ventricular performance determined with echocardiography and alterations in heart rate in normal persons*, The American Journal of Cardiology, Volume 43, Issue 1.

[3] Massimiliano Cantinotti and Martin Koestenberger (2017) *Quantification of Left Ventricular Size and Function by 2-Dimensional Echocardiography: So Basic and So Difficult*, Circulation: Cardiovascular Imaging, Volume 10, No. 11.

[4] S.K.Karna, M.K.Rohit and A.Wanchu (2015) *Right ventricular thickness as predictor of global myocardial performance in systemic sclerosis: A Doppler tissue imaging study*, Indian Heart Journal, Volume 67, Issue 6.

[5] Maceira AM, Prasad SK, Khan M and Pennell DJ. (2006) *Reference right ventricular systolic and diastolic function normalized to age, gender and body surface area from steady-state free precession cardiovascular magnetic resonance*, European Heart Journal, 27(23):2879-88.

[6] François Haddad, Sharon A. Hunt, David N. Rosenthal and Daniel J. Murphy (2008) *Right Ventricular Function in Cardiovascular Disease*, Circulation, Volume 117, No. 11.

[7] Dr. Bassem Sobhi Ibrahim (2016) *Right ventricular failure* , E-Journal of Cardiology Practice, Volume 14.

[8] Federico M. Asch et al (2019) *Automated Echocardiographic Quantification of Left Ventricular Ejection Fraction Without Volume Measurements Using a Machine Learning Algorithm Mimicking a Human Expert*, Circulation: Cardiovascular Imaging, Volume 12, No. 9.

[9] Matthias Schneider et al (2019) *Visual assessment of right ventricular function by echocardiography: how good are we?*, The International Journal of Cardiovascular Imaging, Volume 35, Issue 11.

[10] João Figueira Silva, Jorge Miguel Silva, António Guerra, Sérgio Matos and Carlos Costa (2018) *Ejection Fraction Classification in Transthoracic Echocardiography Using a Deep Learning Approach*, 2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS).

[11] World Health Organization (2017) *https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)*.

[12] Ali Madani, Jia Rui Ong, Anshul Tibrewal and Mohammad R. K. Mofrad (2018) *Deep echocardiography: data-efficient supervised and semi-supervised deep learning towards automated diagnosis of cardiac disease*, npj Digital Medicine, Volume 1, Article number 59.

[13] Elin Björnsson and Jan Liu (2019) *Automatic assessment of cardiac ultrasound images using deep learning*, Departement of Electrical Engineering, Chalmers.

[14] Xiaoxuan Liu et al (2019) *A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis*, The Lancet, Volume 1, ISSUE 6, Pe271-e297.

[15] Karen Simonyan and Andrew Zisserman (2015) *Very Deep Convolutional Networks for Large-Scale Image Recognition*, ICLR Conference.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei (2009) *ImageNet: A large-scale hierarchical image database* 2009 IEEE Conference on Computer Vision and Pattern Recognition.

[17] David Silver et al (2016) *Mastering the game of Go with deep neural networks and tree search* Nature, volume 529, pages 484–489.

[18] Zhenzhong Lan et al (2019) *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations* arXiv:1909.11942.

[19] Amazon Mechanical Turk `https://www.mturk.com/`

[20] Alec Radford et al (2019) *Language Models are Unsupervised Multitask Learners* OpenAI, `https://openai.com/blog/better-language-models/`

[21] Joao ao Carreir and Andrew Zisserman(2018) *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset* arXiv:1705:07750.

[22] Diederik P. Kingma and Jimmy Ba (2014) *Adam: A Method for Stochastic Optimization* arXiv:1412.6980.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2018) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* arXiv:1810.04805.

[24] Ashish Vaswani et al (2017) *Attention Is All You Need* arXiv:1706.03762

[25] Yukun Zhu et al (2015) *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books* arXiv:1506.06724.

[26] Sergey Ioffe and Christian Szegedy (2015) *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* arXiv:1502.03167.

[27] R. A. Miller (2010) *A History of the INTERNIST-1 and Quick Medical Reference (QMR) Computer-Assisted Diagnosis Projects, with Lessons Learned* Yearb Med Inform 2010; 19(01): 121-136.

[28] W. K. Chu et al (1979) *Echocardiogram analysis in a pattern recognition framework* American Association of Physicists in Medicine, Volume 6, Issue 4, Pages 267-271.

[29] Julio E. Pérez et al (1992) *On-line assessment of ventricular function by automatic boundary detection and ultrasonic backscatter imaging* Journal of the American College of Cardiology, Volume 19, Issue 2.

[30] Tim McInerney and Demetri Terzopoulos (1996) *Deformable models in medical image analysis: a survey* Medical Image Analysis, Volume 1, Issue 2, Pages 91-108.

[31] Paaladinesh Thavendiranathan et al (2012) *Feasibility, Accuracy, and Reproducibility of Real-Time Full-Volume 3D Transthoracic Echocardiography to Measure LV Volumes and Systolic Function* JACC: Cardiovascular Imaging, Volume 5, Issue 3.

[32] Partho P. Sengupta et al (2016) *Cognitive Machine-Learning Algorithm for Cardiac Imaging* Circulation: Cardiovascular Imaging, Volume 9, No. 6.

[33] Sukrit Narula et al (2016) *Machine-Learning Algorithms to Automate Morphological and Functional Assessments in 2D Echocardiography* Journal of the American College of Cardiology, Volume 68, No 21.

[34] Federico M Asch et al (2019) *Automated Echocardiographic Quantification of Left Ventricular Ejection Fraction Without Volume Measurements Using a Machine Learning Algorithm Mimicking a Human Expert* Circ Cardiovasc Imaging. 2019;12(9).

[35] Amirata Ghorbani et al (2020) *Deep learning interpretation of echocardiograms* npj Digital Medicine volume 3, Article number: 10.

[36] Jeffrey Zhang et al (2018) *Fully Automated Echocardiogram Interpretation in Clinical Practice* Circulation: Cardiovascular Imaging, Volume 138, No. 16.

[37] Christian Szegedy et al (2014) *Going Deeper with Convolutions* arXiv:1409.4842.

[38] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu and Kaiming He (2016) *Aggregated Residual Transformations for Deep Neural Networks* arXiv:1611.05431.

[39] Kensho Hara, Hirokatsu Kataoka and Yutaka Satoh (2017) *Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?* arXiv:1711.09577.

[40] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah (2012) *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild* arXiv:1212.0402.

[41] Hilde Kuehne et al (2011) *HMDB51: A Large Video Database for Human Motion Recognition*. IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011.

[42] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem and Juan Carlos Niebles (2015) *ActivityNet: A large-scale video benchmark for human activity understanding* 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[43] Will Kay et al (2017) *The Kinetics Human Action Video Dataset* arXiv:1705.06950.

[44] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani and Manohar Paluri (2014) *Learning Spatiotemporal Features with 3D Convolutional Networks* arXiv:1412.0767.

[45] Zhaofan Qiu, Ting Yao and Tao Mei (2017) *Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks* arXiv:1711.10305.

[46] Sergey Zagoruyko and Nikos Komodakis (2017) *Wide Residual Networks* arXiv:1605.07146.

[47] Gao Huang, Zhuang Liu, Laurens van der Maaten and Kilian Q. Weinberger (2016) *Densely Connected Convolutional Networks* arXiv:1608.06993.

[48] *Prodigy - An annotation tool for AI, Machine Learning and NLP* `https://prodi.gy/`
[49] *Natural Language Toolkit* `http://www.nltk.org/index.html`
[50] Yoon Kim (2014) *Convolutional Neural Networks for Sentence Classification* arXiv:1408.5882.
[51] *Hugging Face Transformer Library* `https://huggingface.co/`
[52] Ilya Loshchilov and Frank Hutter (2018) *Decoupled Weight Decay Regularization* ICLR 2019 Conference.
[53] *OpenCV* `https://opencv.org`
[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun (2015) *Deep Residual Learning for Image Recognition* arXiv:1512.03385.
[55] Bing Xu, Naiyan Wang, Tianqi Chen and Mu Li (2015) *Empirical Evaluation of Rectified Activations in Convolutional Network* arXiv:1505.00853.
[56] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas and Martin Wattenberg (2017) *SmoothGrad: removing noise by adding noise* arXiv:1706.03825.
[57] Herbert E. Robbins (2007) *A Stochastic Approximation Method* Annals of Mathematical Statistics, Volume 22.
[58] Mark Hughes, Irene Li, Spyros Kotoulas and Toyotaro Suzumura (2017) *Medical Text Classification Using Convolutional Neural Networks* Stud Health Technol Inform. 2017;235:246-250.
[59] Wei-Hung Weng, Kavishwar B. Wagholikar, Alexa T. McCray, Peter Szolovits and Henry C. Chueh (2017) *Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach* BMC Med Inform Decis Mak 17.
[60] Wang, Y., Sohn, S., Liu, S. et al (2019) *A clinical text classification paradigm using weak supervision and deep representation.* BMC Med Inform Decis Mak 19.
[61] Jinhyuk Lee et al (2019) *BioBERT: a pre-trained biomedical languagerepresentation model for biomedical text mining* Bioinformatics, 1–7.

A

Appendix 1

A.1 Sensitivity maps over time

Sensitivity maps over the entire duration of the input can be found in Figure A.1 and A.2.

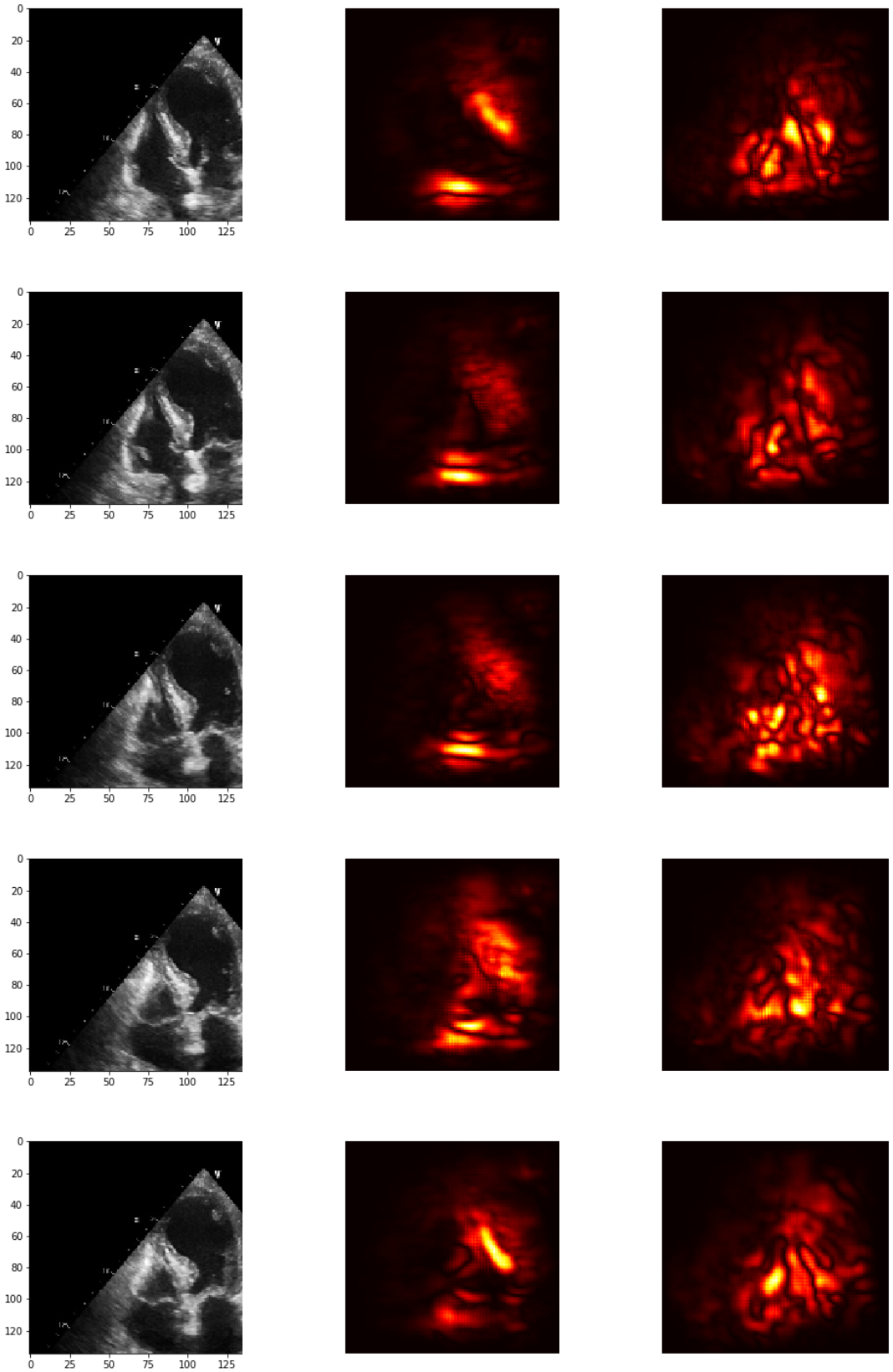

Figure A.1: Sensitivity map over time for the mobility and size models. Frames 1 to 5

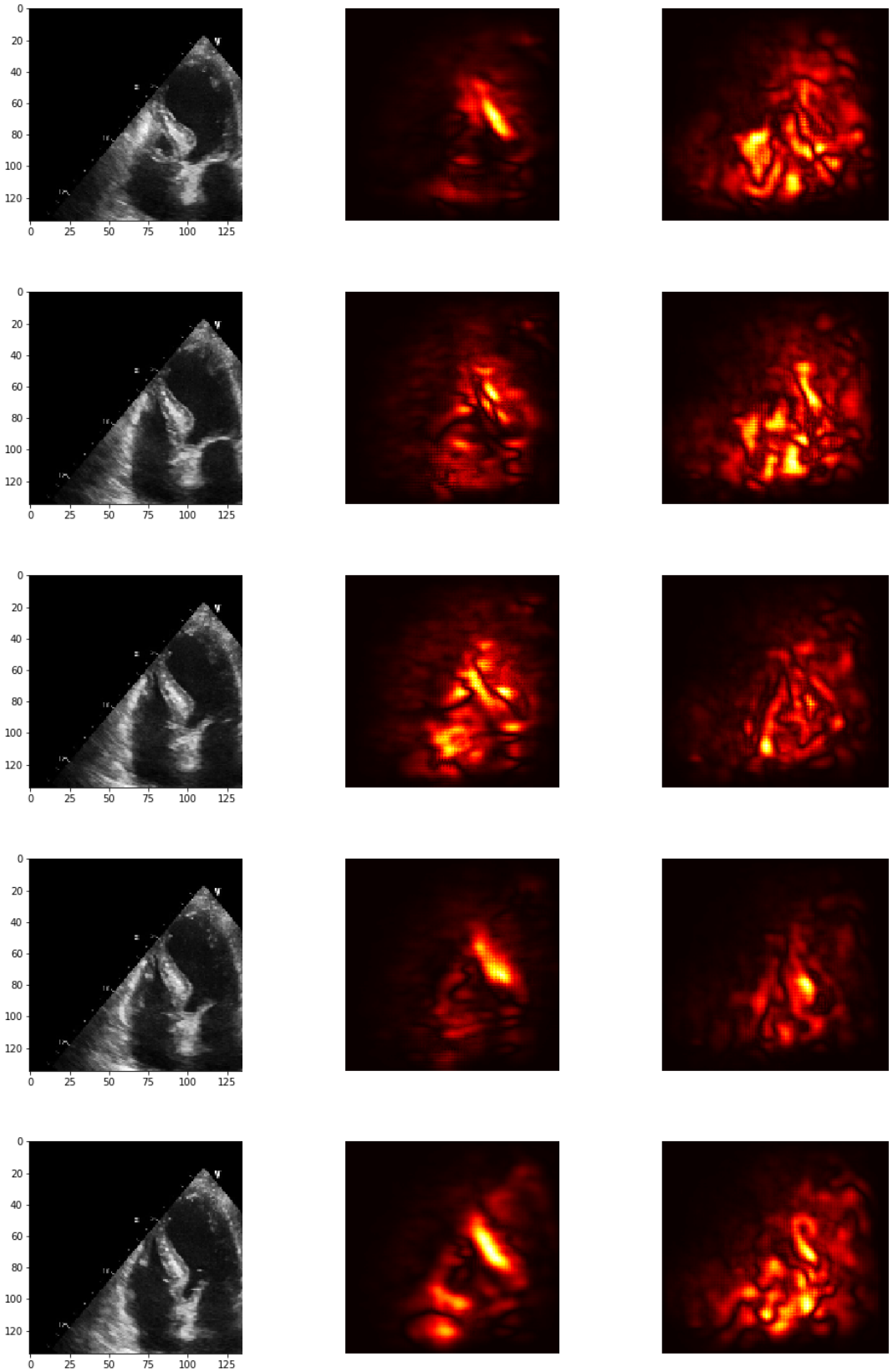

Figure A.2: Sensitivity map over time for the mobility and size models. Frames 6 to 10