BACHELOR'S THESIS / BIOMEDICAL ENGINEERING 2020

Classification of CMR Images between Healthy and Diseased Myocardium: A Deep Learning Approach

Simran Anand Sahdev

upf. Universitat Pompeu Fabra Barcelona

Bachelor thesis in Biomedical Engineering

Universitat Pompeu Fabra

Classification of CMR images between healthy and diseased myocardium: A deep learning approach

Simran Anand Sahdev

Supervisor: Dr. Amit R. Patel, UChicago Medicine, Chicago, USA

Co-supervisor: Dr. Gemma Piella, Universitat Pompeu Fabra, Barcelona, Spain

July 2020

Universitat Pompeu Fabra Barcelona

This thesis is dedicated to my parents and brother.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors Dr. Amit Patel and Gemma Piella for their guidance, advice and support throughout the project, and especially for their confidence in me. I would also like to thank Amelia Jiménez, a PhD student at the University of Pompeu Fabra, who provided me with constant feedback. Thanks to the full team at the Cardiovascular Imaging department at UChicago Medicine, they made me feel comfortable from day one. Special thanks to Dr. Shuo Wang, for her contribution in manually annotating and segmenting the data I used and for our daily lunch dates! I would also like to thank Dr Emeka Anyawnu and Daksh Chauhan, my AI team mates, who provided me with useful tips to carry out my project. Thanks to Prof. Ralph Andrzejak and Guillermo Jiménez for agreeing to be my reviewers. Finally, I would like to thank my family and friends for their unconditional love and support. It would not have been possible without them.

ABSTRACT

Heart failure is a major cause of morbidity and mortality in the world. Late gadolinium enhanced (LGE) cardiac magnetic resonance (CMR) imaging can be used to directly visualise the presence of myocardial damage, a predictor of heart failure.

Deep Learning (DL) models, specifically convolutional neural networks (CNNs), can help in assisting medical personnel in the analysis and classification of various diseases, including the identification of myocardial damage from CMR images. However, due to the lack of labeled medical images, training DL models is a big challenge. Transfer learning, a technique used for adapting models that have previously been trained on a larger dataset, has proven effective in overcoming this problem.

This study compares three state-of-the-art CNNs (VGG16, VGG19 and Inception V3) in the classification of healthy and diseased myocardium using transfer learning on an imbalanced dataset. To compensate for the imbalance, three subsets of the dataset were created using undersampling and class weighting techniques. These were also used to train the models and subsequently compared. The aim was to see which model could be used in a clinical setting.

Results showed that all models classified a significantly higher amount of times images as healthy, as opposed to diseased and that the balanced dataset provided slightly better outcomes. It was determined that although none of the models were appropriate to assist physicians at the moment, VGG19 could be further tuned to perform better.

KEYWORDS

Heart failure; Myocardial damage; Late gadolinium enhancement; Convolutional neural networks; Transfer learning; Deep Learning; Features; Dataset.

TABLE OF CONTENTS

1. INTRODUCTION

Heart failure is a major cause of morbidity and mortality in the world [37]. Contrast-enhanced **cardiac magnetic resonance** (CMR) imaging can be used to directly visualize the presence of myocardial damage, a predictor of heart failure. The process of diagnosing a patient with heart failure is long and requires experienced physicians. Artificial Intelligence (AI), and in particular Deep Learning (DL) methods, has great potential in medicine and could reduce the time needed to diagnose patients. Through a process of training, DL algorithms can learn from experience (e.g. medical data) to predict and solve complex tasks. The goal of this thesis was to retrain and compare three DL models in the classification of contrast-enhanced CMR images into healthy or diseased myocardium, and analyze whether any of the models could be used in a clinical setting. These DL models were retrained using a technique known as transfer learning.

1.1. Heart Failure

According to the American Heart Association, heart failure is defined as "a chronic, progressive condition in which the heart muscle is unable to pump enough blood to meet the body's need for blood and oxygen" [1].

CMR imaging has been recognized as the gold standard for the assessment of cardiac anatomy and function. **Late gadolinium enhancement** (LGE) CMR is considered the most accurate and highest resolution method for the diagnosis or prognosis of a variety of myocardial diseases including myocardial viability, ischemic and non-ischemic cardiomyopathies, myocarditis and other infiltrative myocardial processes. The pattern of myocardial damage can help determine the cause of heart failure, and the amount of myocardial damage can help identify patients at highest risk of sudden cardiac death [4].

Magnetic resonance imaging (MRI) is a non-invasive imaging technique that uses the intrinsic properties of hydrogen proton spins. Outside a magnetic field, the spin of each hydrogen proton is randomly oriented and provides a net magnetization of zero. However, when a subject is placed within a strong magnetic field, denominated B0, such as the one produced by an MRI scanner, the protons tend to align in the direction of the field producing a net magnetization. When a radio frequency pulse B1 is then applied, also produced by the scanner, it creates a varying magnetic field. The protons absorb the energy from the pulse and flip their spins, eliminating the net magnetization. When the new field is turned off, the protons gradually return to align with the first magnetic field B0. The return process produces a radio signal that can be detected and measured by receivers in the scanner and into an image [2, 3]. During the acquisition of images, different MRI sequences can be achieved by changing the pulses sent to disturb the magnetic field B0, thus allowing for visualization and quantification of different organs.

CMR imaging comprises several techniques of MRI sequences. LGE-CMR relies on the intravenous delivery of gadolinium (Gd) agents to the myocardium [4]. The term late gadolinium enhancement refers to the different wash-in and wash-out kinetics of normal myocardium and tissue with myocardial infarction [5]. In the early stages, 1-3 minutes after intravenous delivery, gadolinium agents reside in the normal myocardium. Due to normal kinetics, in later stages (approximately 10 minutes) it only accumulates in the damaged tissue, allowing physicians to achieve an optimal contrast between healthy and infarcted myocardium at this point. This can then be visualized with MRI using a **phase sensitive inversion recovery** (PSIR) sequence, which nulls the signal from normal myocardium in order to maximize the contrast between diseased and normal myocardium [30]. After acquiring the sequence of PSIR CMR images, physicians have to go through each slice individually to first detect whether there is LGE in the myocardium and then segment the diseased part. Figure 1 illustrates an example of a slice of a PSIR CMR sequence, where the diseased myocardium has been segmented and is shown brighter due to the Gd agents, whilst healthy myocardium is shown in black.

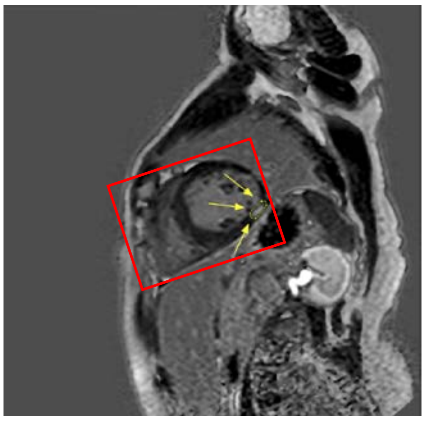


Figure 1. LGE on a short-axis plane PSIR CMR image. The red box corresponds to the heart, where the left and right ventricles can be visualized. The segmented and brighter part corresponds to diseased myocardium, while the black part corresponds to healthy myocardium in the left ventricle.

1.2. Artificial Intelligence

Advances in computational power and an explosion in the generation and availability of data have allowed AI to take center stage in our world. AI is a science that studies how to simulate human intelligence in machines and computers to complete tasks, like self-driving cars. Nowadays, computers are being used to perform a wide-range of complex tasks with high accuracy.

The digitalization of health-related data has allowed the healthcare industry to develop and demonstrate, in recent years, that computers utilizing AI can provide guidance and assistance during image acquisition, in the diagnosis and prognosis of diseases and in many other medical sectors. This could potentially have a significant influence on a physicians workload [6].

Machine Learning

Machine learning (ML), a subset of AI, is the scientific discipline that focuses on how computers automatically learn and improve from data or experiences with the use of mathematics, statistics and computer science. The goal of ML is to create mathematical models that can be trained to generalize and correctly predict, classify or group outcomes for new, unseen data. For machines to learn, they require three components: a dataset, features and an algorithm or model to be trained.

The **dataset** can consist of a large amount of images, numbers, texts or any other kind of data. During the different phases of creating an ML model, the data is usually split into three subsets: *training*, *validating* and *testing* dataset. Throughout the training phase, the model is provided experiences from the training dataset and is tuned to produce accurate prediction from this data by an *optimization algorithm* that adjusts the *parameters* or *weights* of the model. The ability of a model to generalize is usually estimated and corrected during the training phase using the validation data. This information is then used as feedback to further tune and optimize the model. Without the use of the validation set, the model could overfit and not generalize when presented unseen data. After several iterations of training and tuning, the final model is evaluated on the testing dataset, which is used to simulate how the model would perform when faced with new, unseen data.

Features are the key information that allow machines to learn. These features are defined as the characteristics of the data which will tell the machine what to pay attention to during the solving process. The art of choosing the data features is known as feature engineering.

ML **models** can be thought of as a function that accepts data as the input, then manipulates and transforms this data and finally returns a solution or prediction to the task. They are generally classified based on the type of learning technique they use:

- In the case of *supervised learning*, the models are trained on a series of inputs where the outputs or targets are known. This type of data is known as a labeled dataset. This type of learning technique is useful for classification and regression problems. For example, a classification algorithm could consist of a labeled dataset of medical images of tumors and the output would be whether the tumors are benign or malignant.
- In the case of *unsupervised learning*, the algorithm is trained on an unlabeled dataset, where the target is not known. The model looks for clusters or a pattern within the dataset to make sense of what it is seeing to predict. For example, if the computer is shown a sample of different types of flowers it could look for clusters of roses, sunflowers and daisies.
- *Reinforcement learning* is based on trial and error to achieve an objective. Similar to a human learning how to play a game for the first time, the algorithm tries different things and is rewarded or penalized based on whether the outcome is close or not to the objective. A famous ML algorithm which uses reinforcement learning is AlphaGo, developed by DeepMind [7].

For the purpose of this study, the *dataset consisted of CMR images* and *supervised learning was the default learning technique*.

Deep Learning

Prior to deep learning (DL), most ML models often required manual feature extraction to reduce the complexity of the data and make patterns more visible for the algorithm. This process is difficult and time consuming as the expert would have to identify the key features of the data. The performance of most ML models depends on how accurately the features are identified and extracted. In contrast, in DL models, a subfield of machine learning, these features are automatically learnt and extracted from the data without the need of human intervention. DL has shown promising results in a wide range of image classification problems and is currently receiving the most attention.

DL models are based on **artificial neural networks** (ANNs), inspired by the structure and function of the brain. The human brain consists of billions of neurons interconnected to each other that send and process chemical and electrical signals through a process known as synapses. All information that our brain processes and stores is done thanks to these interconnections between neurons and their relative strength. ANNs are based on multiple connected processing units, mimicking neurons, which work together to process information and generate results from it. The results can change based on the parameters of the interconnections between the units or their relative strengths. ANNs can be divided into three parts: an *input layer*, one or more *hidden layers* and an *output layer*. Information flows from the input layer, through the hidden layer to the output layer which then produces a result.

In the healthcare sector, **convolutional neural networks** (CNNs), a DL model, have shown promising results in medical image analysis and classification. As shown in Figure 2, CNNs consist of an input layer, an output layer (prediction) and multiple hidden layers which handle and transform the input. These hidden layers are typically divided in two parts:

1. **Feature Learning**, where the model generates features from the image automatically. This part is composed of,
    - *Convolutional layers*. A convolution is a linear operation that involves the multiplication of $n \times n$ kernels or filters over the input and produces a tensor. These filters look for features or characteristics in the input. Each number in the kernel is known as a weights or parameter, which will be optimized during the training process to look for the important features needed to predict correctly. One example of a filter could be the detection of horizontal lines or edges in an image.
    - *Activation layers*. The tensors obtained in the convolutional layer are then fed through a nonlinear activation function which results in feature maps. One of the

most used nonlinear functions is the simple rectified linear function (ReLU), where all negative values are set to zero and positive values remain as they are.

- *Pooling layers*. The feature map is reduced in size to remove redundant features either by using a max pooling or average pooling function.

2. **Classifier**, which is usually composed of *fully connected layers* or *dense layers.* The feature maps from the previous layer are flattened into a vector, and each unit is connected to every unit in the next layer allowing the machine to learn nonlinear combinations of the features. Each unit represents the probability that a certain feature belongs to a label or class. As with the kernels, the weight of connection between each unit is tuned and optimized during the training phase. This layer is known as the classifier, as this is where the decision is made. In the case of a classification problem, the output of the network is a vector where each element is the probability of the input image being of a certain class.

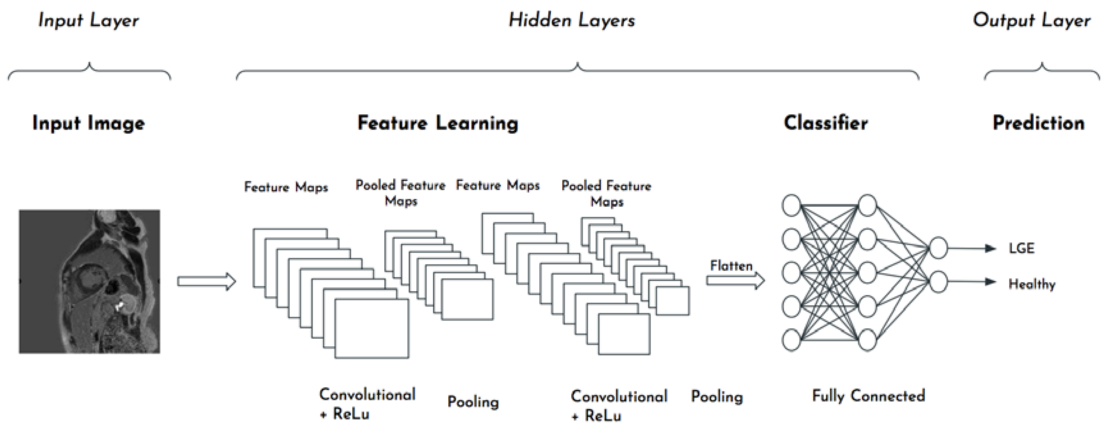


Figure 2. Convolutional Neural Network (CNN) framework. A CNN takes the CMR image as an input, which is then passed through a series of convolutional, activation and pooling layers to learn features. The feature maps are then flattened into a vector, to be fed into the fully connected layers which will then provide a prediction of whether it thinks the image corresponds to a healthy or diseased (presents LGE) heart.

An important concept in CNNs is the notion of **hierarchical layers** during the feature learning phase. The model uses multiple feature learning layers to progressively extract higher level features from the dataset. For example, in the lower layers or the first layers the model looks for general features such as edges and colors from the data, whilst higher layers could detect more complex concepts such as letters or representation of faces.

The combination of these hidden layers gives rise to different **CNN architectures**, from shallow models (small number of hidden layers) to very deep models (large number of hidden layers). Depending on the type of task we want to solve, the CNN architecture will be different.

In the ML section, an overview of the training phase was explained. In the case of CNNs, **backpropagation** is the method used to train models for supervised learning and adjust the weights. This method can be divided into 4 parts:

1. *Forward propagation*, where the training dataset is passed through the whole network to obtain all the predicted values.
2. Calculation of the *cost* or *loss function*. This step evaluates the predicted outputs from the forward propagation against the expected predictions (labeled dataset values). The error rate is calculated using a cost or loss function, and will give the model a sense of how well or badly it is doing. Different formulas can be used to calculate the error, such as the mean squared error or cross-entropy.
3. *Backpropagation* or *backward propagation of errors*. The aim of this step is to minimize the cost function by adjusting the weights. During this part, the error rate is propagated backwards to previous layers where the level of adjustment of weights in that layer is determined by computing the gradients of the cost function with respect to those weights. The gradients will give the model a sense of how much each weight is affecting the cost function.
4. *Weight update*. An optimizer is used to change the weights by using the gradients calculated previously. Some examples of optimizers are stochastic gradient descent (SGD), RMSprop or Adam [38]. Using any of these optimizers and a learning rate, which controls how much they should be adjusted, the weights are updated.

These 4 parts form one training iteration. Models are usually trained during multiple iterations, and so the weights are updated constantly until the desired result is reached.

1.3. Transfer Learning

One of the main obstacles of training DL models for medical problems is the lack of labeled medical data. Although more and more data is being gathered, DL models need huge amounts of information to be able to accurately predict. Transfer learning can be a big help in overcoming this obstacle and achieving optimal results.

Transfer learning is a common strategy to train a neural network with a small dataset using an existing model which was pretrained on a large dataset. The underlying assumption is that lower-level features (e.g. edges and colors) learned on a large dataset can be useful for other tasks. In this case, the weights from the feature learning layers can be reused and the classifier weights can be retrained to adapt to the new dataset.

1.4. State of the Art

Given enough training data, DL models can achieve comparable or even better performance than experts in the diagnosis or prediction of certain diseases, e.g. predicting cardiac arrest [14]. The only problem being the lack of large amounts of high-quality labeled medical data.

Transfer learning on state-of-the-art DL models such as AlexNet [10], Inception V3 [11, 12], VGG16 and VGG19 [13], pretrained on the 1000 class ImageNet [8] database has proven to be effective when they are retrained on new datasets for different tasks. Kermany et al. [15] achieved 92.8% accuracy in the detection of pneumonia on a small dataset of 5,232 chest X-ray images using this technique. Vianna [16] also demonstrated the performance of models when using transfer learning in the prediction of pneumonia. Esteva el at. [17] demonstrated that retraining the Inception V3 on skin images to detect the presence of a benign or malignant tumor gave results which were on par with expert dermatologists. Shu [19] showed that using a small dataset of 6,000 images on pretrained DL models avoided overfitting and gave accurate results as long as the appropriate modifications were done.

Nevertheless, transfer learning will not always be the correct strategy to choose, as shown in the paper of Ezqeuiel de la Rosa et al. [18]. They proposed a new automatic method for myocardial infarction quantification on LGE-CMR images, but the first part of their method was to detect the presence of LGE. They compared a proposed CNN to

VGG19 and although the results obtained were very similar, they decided to use their model as it had slightly better results than using transfer learning.

1.5. Objective

Nowadays, physicians have to manually detect whether there is myocardial damage on CMR images, which requires time and experience. In this study we implemented and compared three state-of-the-art CNN models (Vgg16, Vgg19 and Inception V3) in the discrimination between healthy and diseased myocardium on LGE-CMR images with a small imbalanced dataset. The models were pretrained on the ImageNet dataset and transfer learning was used to retrain them. These architectures were chosen after an extensive research of various well-established models and their performance when adapted to a variety of medical image classification scenarios. To compensate for the imbalance and also study the effects of the quality of the dataset when retraining models, three subsets of data were created from the original one using undersampling and class weighting techniques. The end goal was to determine whether any of the chosen models could be used in a clinical setting. Furthermore, the following two questions were answered during the study:

- Would a deeper CNN architecture, such as Inception V3, allow the model to detect and represent more complex relations between the features and thus achieve better performance?
- Would a balanced dataset be necessary to show meaningful results, as the model would be trained on equal amounts of data of both classes?

2. MATERIAL AND METHODS

2.1. Programming Environment

The main programming language used during the project was Python [27]. Tensorflow [20] and Keras [21], two deep learning libraries, were used to develop and train the models. Tensorflow is an open source python ML library for performing high-end numerical computations. It is a framework that involves defining and running computations involving tensors [22]. Keras is a high-level neural networks API, written in Python and capable of running on top of Tensorflow. Thanks to its easy and fast coding approach, it is widely used by beginners.

As stated in the introduction, transfer learning was used on three state-of-the-art pretrained models, VGG16, VGG19 and InceptionV3. These models were available in the Keras library and could easily be modified and retrained to perform image classification on new datasets.

To be able to retrain the models on Keras the following inputs were required:

- **Batch size.** Number of training or validating examples that the model used before updating its weights. Due to the memory requirements, using fewer samples of the training and validating dataset was necessary so that the model took up less space on the machine.
- **Epochs**. Number of complete passes through the entire training and validating dataset. One epoch is accomplished when a complete training and validating dataset is cycled forward and backwards through the CNN.
- **Labeled Dataset**. To train and evaluate the models, the data had to be stored in a particular format to be fed into the CNNs (see Figure 3). ImageDataGenerator and flow from directory, two functions available on Keras, were used to load, preprocess and train the models with the data in batches. For this study, the dataset folder consisted of CMR images that were divided into three subfolders: training, validating and testing. Each of these subfolders were then divided into two folders, corresponding to the classes LGE (i.e. diseased myocardium) and healthy.

- **Pretrained model**. Load the Vgg16, Vgg19 or InceptionV3 models with their corresponding weights from the ImageNet dataset.
- **Defining which layers to be retrained**. In the case of this study, the feature learning layers of each model were frozen, meaning the weights obtained on the ImageNet dataset of each model were not to be updated, whilst the classifiers were retrained.
- **Create a new classifier**. For every model, a new classifier outputting 2 classes was created. In the last layer, where the prediction is made, a softmax function was used to turn the values into probabilities ranging from 0 to 1.
- **The "class weight" function values**. This function was used during the training phase on two of the datasets (explained in section 2.2. Dataset). Here, each class is assigned a weight which will cause the model to "pay more attention" to examples from the under-represented class, in this case LGE, when optimizing the weights. For example, if the LGE class (1) is twice less represented than the healthy class (0), class weight could be defined as {0: 0.5, 1: 1}.
- **Compile function**. This is where the loss function, optimization function and metric used to monitor training were defined. All models used a categorical cross entropy loss function and Adam optimizer. In those cases where the "class weight" function was used, the loss function applied was a weighted cross-entropy. The metric calculated was accuracy, which would allow us to obtain epoch vs loss graphs and epoch vs accuracy graphs.

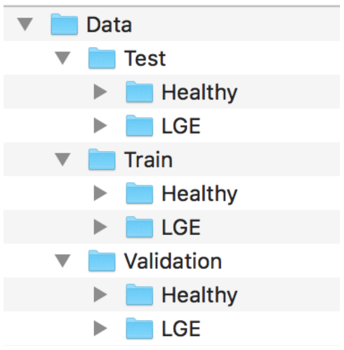


Figure 3.View of the structure of the folders where the data was stored.

2.2. Dataset

The dataset consisted of short-axis stress CMR images of 405 patients (59 ± 14 years SD, 54% male) acquired at the UChicago Medicine Duchossois Center for Advanced Medicine (DCAM). All examinations were performed using a 1.5 Magnetic Resonance Imaging System (Achieva, Philips Medical Systems). On average, a short-axis image stack had around 12 slices or images. For each patient, the PSIR technique was used to analyze the images and visualize LGE in the damaged myocardium.

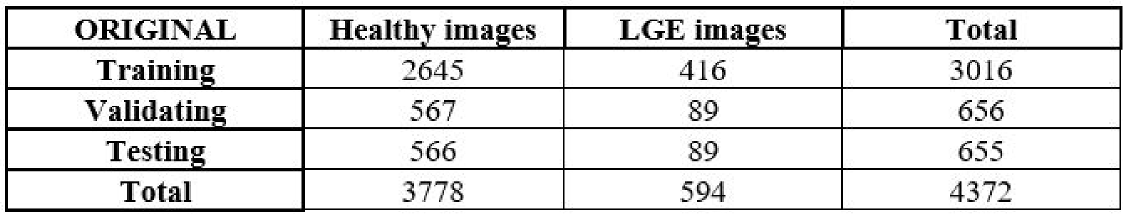
Manual image annotation and segmentation of LGE was done by a trained physician, from UChicago Medicine, using Arterys [23], an AI assisted software for medical images. All the CMR images of the patients were uploaded onto Arterys and analyzed individually for LGE. Arterys also anonymized the data, by changing the real name of the patient to a phonetic id (e.g. Bisunay) making it easier to use this data in a secure way afterwards. Out of the 405 patients, 135 presented myocardial damage on some of their CMR images. These were then segmented and labeled as LGE on the software.

After analyzing all 405 patients, Arterys provided to the team the exported database in the form of two JSON files: a study file, which contained the information of the patients anonymized and the different MRI sequences acquired, and a workspace file, which had the modifications (segmentations and labels) done on every slice. Both files also included those patients whose images did not present any myocardial damage (no LGE label) and, so, had not been annotated or segmented.

The DL models used on Keras required that the data be labeled and stored in a particular format, as mentioned in the previous section (2.1. Programming Environment). To do this, an algorithm was created, using Python, to separate those images that had been labeled from those that had not and then be placed in two folders: Healthy folder, which had those images that had not been labeled, and LGE folder, which had those images that had been labeled. As the JSON files contained all the database of every patient, the algorithm first had to look through both JSON files and link each patient to its workspace. From there, the algorithm had to look for the correct PSIR sequence and check whether any of the slices had been labeled. If any of the slices had been annotated, they were saved in the LGE folder. Those slices that had not been annotated were saved in the Healthy folder. Figure 3 shows the format in which the data was stored.

Once every image had been saved in their respective folder (Healthy or LGE), they were randomly separated into three subsets. 70% of both classes were included in the training dataset, 15% in the validation dataset and 15% in the testing dataset.

Table 1. Original dataset. Number of healthy and LGE images for the training, validating and testing folders.



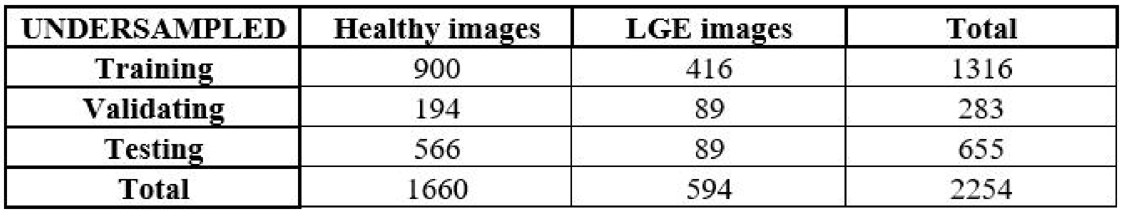
ORIGINAL	Healthy images	LGE images	Total
Training	2645	416	3016
Validating	567	89	656
Testing	566	89	655
Total	3778	594	4372

It was clear from Table 1 that the data was imbalanced. The percentage of LGE cases from the total number of slices was 13%. As stated in the objective section (1.5.Objective), one of the questions was if a balanced dataset would be necessary to train the models correctly. To get an answer, the following datasets with their characteristics were used:

- **Original** or **imbalanced dataset** (see Table 1), which included all the healthy and LGE images of all 405 patients.
- **Undersampled** or **balanced dataset** (see Table 2), which had all the LGE images, but 65% of the healthy slices were randomly deleted from the original training and validating folders so that the ratio between both classes was 1:2 during the training phase. The reason why the data was not balanced to have a ratio of 1:1 was because we wanted the model to learn from a real life scenario, where most images would be from the healthy class and second because of the lack of data. If more images had been deleted, the models would not have had enough training data to learn from. The testing folder remained the same, so that we could compare the results to the other datasets.

- **Original dataset with the use of "class weight" function**. This consisted of the original dataset with the use of the function "class weight" during the training phase. Throughout the study this third dataset will be referred to as "weighted".
- **Undersampled dataset with the use of the "class weight" function.** This consisted of the undersampled dataset with the use of the function "class weight" during the training phase. Throughout the study this fourth dataset will be referred to as "undersampled + weighted dataset".

Table 2. Undersampled dataset. Number of healthy and LGE images for the training, validating and testing folders. 65 % of healthy images, from the original training and validating folders, were randomly eliminated to balance the data.



UNDERSAMPLED	Healthy images	LGE images	Total
Training	900	416	1316
Validating	194	89	283
Testing	566	89	655
Total	1660	594	2254

2.3. Architecture CNNs

VGG16, VGG19 and Inception V3 were the three state-of-the-art DL architectures, included in the Keras library, that were chosen for this project as they showed good results in other medical classifications tasks. All these models were previously trained on the ImageNet database of 1000 classes.

VGG16 and **VGG19** were developed by the Visual Geometry Group from University of Oxford in 2014 [13]. Compared with VGG16, VGG19 tends to be slightly better but requires more memory. VGG16 consists of 16 weighted layers including thirteen convolutional layers with filter size 3x3, and 3 fully connected layers. All convolutional layers are divided into 5 groups, which are followed by a max pooling layer. VGG19 has the same architecture as VGG16, the only difference being that instead of having thirteen convolutional layers, it has sixteen convolutional layers [24]. The fully connected layers of both models consist of two fully connected layers with 4026 nodes each, followed by a 1000 classifier layer corresponding to the number of classes in the ImageNet dataset. In the case of this project, the classifier was retrained and modified to have a 2 classifier layer, corresponding to LGE or healthy (see Appendix 3 for summary of both models).

The use of multiple small filters allows both models to cover the full area that larger filters would cover. For example, two layers of 3x3 filters would be equal to using one 5x5 filter in that area. This would also mean that fewer parameters are needed. Two layers of 3x3 filters would be equal to 2x3x3 = 18 parameters, whilst one layer of a 5x5 filter would be equal to 1x5x5 = 25 parameters. This is good for a better convergence and decreasing the chances of overfitting.

InceptionV3 was developed by a research team at Google in 2014 [11, 12]. Up until the creation of this model, most CNNs had what is known as a sequential architecture, meaning that the layers with different sized filters were stacked one after the other. This meant that different sized features were learnt at different levels of the model. Based on the idea of “inception modules”, Google introduced multi-level feature extractors that use different size filters, ranging from 1x1 to 5x5, within the same module to extract different size features [24] at the same time. The resulting tensors of each filter are then concatenated together and sent to the next layer. The model has 42 layers, the deepest model which was used for this project, but fewer parameters than other CNNs. Whilst VGG16 and VGG19 have very good accuracy on the ImageNet dataset, its huge computational requirements, in terms of memory and time, is a problem. Inception V3 also proved the efficacy of using very deep networks [25]. During the retraining phase, four fully connected layers were added to this model, with the last one consisting of 2 nodes (see Appendix 3 for model summary).

2.4. Implementation Details

In the case of this study, the training and validating batch size was set to 25, and the testing batch size was set to 1 for all the models. The numbers of epochs done during the training phase was set to 50. The learning rate was set to the default of the Adam optimizer, which was 0.001.

Regarding the original dataset with the use of “class weight” function, the values were set to {0:0.58, 1:3.68}, where 0 was the healthy class and 1 the LGE class. For the undersampled dataset with the use of “class weight” function, the values were set to {0:0.73, 1:1.59}. The difference in values for the two datasets was due to the fact that the original one had a far greater number of healthy images, and so the weight given when classifying LGE images in the loss function had to be bigger. For the balanced dataset, as the ratio was reduced to 1:2, the weight given to the LGE class could be reduced.

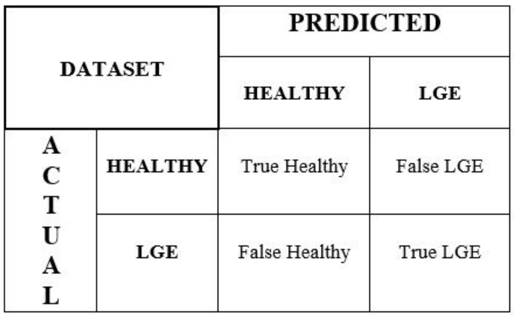
When feeding the models the data, the images first had to be preprocessed. Using the ImageDataGenerator function, they were first rescaled to a 1 - 255 color scale. Then, for the case of VGG16 and VGG19, the images were resized to 224 x 224 pixels, and for the case of Inception V3 to 229 x 229 pixels. One requirement was that all images had to have 3 channels, which was not the case for this data. The flow-from-directory function automatically created 3 channels for the images when the color mode was defined as “rgb”.

2.5. Evaluation Metrics

As imbalanced datasets were used during this study, it was important to look at the correct evaluation metrics which would allow us to analyze the effect of having more data of the healthy class. For most tasks, accuracy is used as a performance measure. In the case of imbalanced data, accuracy alone will not give a correct measurement of the overall performance as the data of the majority class will overwhelm the minority class, meaning that an accuracy score of 0.9 which might seem very good could mean that only the majority class is being correctly classified. Due to this, the following evaluation metrics were used to compare the models:

- **Confusion Matrix:** a table that provides information on which classes are being predicted correctly, incorrectly and what type of errors are being made. In the case of this project, which had a binary classification task, the confusion matrix can be seen in Table 3.

Table 3. Confusion Matrix.



DATASET		PREDICTED	
		HEALTHY	LGE
ACTUAL	HEALTHY	True Healthy	False LGE
	LGE	False Healthy	True LGE

From this table we see how many times the model predicted healthy or LGE correctly, corresponding to True healthy and True LGE, and how many times it predicted these classes incorrectly, seen as False healthy and False LGE.

For the performance metrics of both classes (LGE or healthy), the following notation was used:

· $\mathrm{True\ Healthy} = \mathrm{N(H, detected)}$

· $\mathrm{False\ LGE} = \mathrm{N(H, missed)}$

· $\mathrm{N(H, total) = N\ (H, dected) + N(H, missed)}$

· $\mathrm{True\ LGE} = \mathrm{N(LGE, detected)}$

· $\mathrm{False\ Healthy} = \mathrm{N(LGE, missed)}$

· $\mathrm{N(LGE, total) = N\ (LGE, dected) + N(LGE, missed)}$

· $\mathrm{N(total) = N\ (H, total) + N(LGE, total)}$

- **Accuracy**: ratio of correctly predicted observations to the total number of observations. The value will always be between 0 and 1.

$$A = \frac{N(H, detected) + N(LGE, detected)}{N(total)}$$

As mentioned before, for imbalanced data this is not an appropriate measurement.

- **Precision**: ratio of correctly predicted observations to the total predicted observations of a class. The value will always be between 0 and 1. Of all predicted healthy / LGE images, how many actually are healthy / LGE?

$$P(class) = \frac{N(class, detected)}{N(class, detected)\ +\ N(other\ class, missed)}$$

- **Recall (Sensitivity)**: ratio of correctly predicted positive observations to all the observations in that class. The value will always be between 0 and 1. Of all true healthy / LGE cases, how many did we predict correctly?

$$R(class) = \frac{N(class, detected)}{N(class, detected) + N(class, missed)}$$

For medical tasks, this metric is the most important one as it is more valuable to know that the model is classifying all the correct images as LGE, even if some healthy images are classified as LGE which can then be revised by the physician. The important thing is not to miss out on the classification of diseased myocardial images due to false negatives.

- **F1-score**: a single score based on the weighted average of precision and recall. The value will always be between 0 and 1.

$$F1 - score\ (class) = 2 \times \frac{(R(class) \times P(class))}{(R(class) + P(class))}$$

3. RESULTS

Training time for each model was significantly reduced thanks to the use of transfer learning. On average, each model took around 2-3 hours to train. Once the models had been retrained, the confusion matrices and performance measures were obtained on the testing dataset to analyze the classifying capacity of each model trained on the 4 datasets. The testing dataset consisted of 655 images in total, out of which 566 images were labeled as healthy and 89 images were labeled as LGE. The same testing images were used for all models to be able to correctly analyze and compare the models.

3.1. Evaluation Metrics

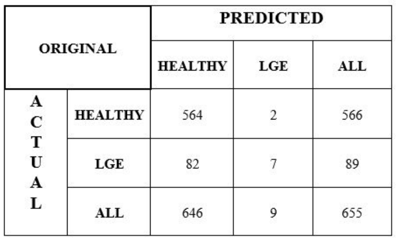
The confusion matrices for each mode with their corresponding datasets can be seen in Tables 4, 5 and 6, whereas the performance measures (accuracy, precision, recall and F1-score) for both LGE and healthy class are illustrated in Table 7. Accuracy and loss functions for the training and validating phases have been added in Appendix 1, these were useful for seeing whether or not the model was overfitting and whether or not it was starting to generalize its prediction capacity.

Ideally, the antidiagonal values (false healthy and false LGE) of the confusion matrices would be zero, as this would mean that every image is classified in the correct class. In this study, the results show that the models classified correctly those images that were labeled as healthy significantly more when compared to the LGE images. When looking at the confusion matrices of all the models, VGG16 (see Table 4) appeared to be the model that classified most of the images as healthy for all datasets. VGG19 (see Table 5) classified slightly more times LGE images correctly then the other models for most datasets. Inception V3 provided very varied results for each dataset, specifically when looking at the confusion matrix of the weighted dataset (bottom left matrix in Table 6) where most of the testing data was classified as LGE, a stark comparison to all the other models, which usually classified most images as healthy, and at the original dataset (see top left matrix in Table 6) where no image was classified as LGE.

As explained previously, accuracy is not an appropriate measurement for this task and so, a result of 0.87 (see Table 7, VGG16 retrained on the original dataset) could be considered good, but in reality when looking at the confusion matrices it is not classifying LGE images properly. The precision, recall and F1-score for both classes showed what was happening in reality. The values obtained for the healthy class were much higher compared to the LGE class, where values seemed to stay on average under 0.5, except for the case of Inception V3 retrained on the weighted dataset. The undersampled dataset provided slightly better performance values for the LGE class in general. VGG19 showed overall better performance values for the LGE class, as the F1-score (average of recall and precision) for this model and all datasets was always above average.

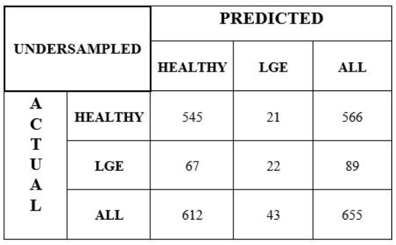
Table 4. From a to d: Confusion matrices for the pretrained VGG16 model retrained on the original, undersampled, weighted and undersampled + weighted datasets.

VGG16



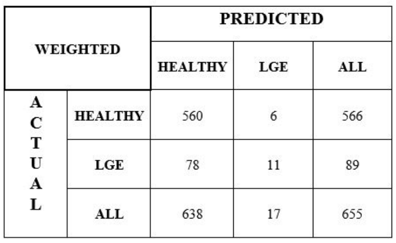
ORIGINAL		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	564	2	566
	LGE	82	7	89
	ALL	646	9	655

(a)



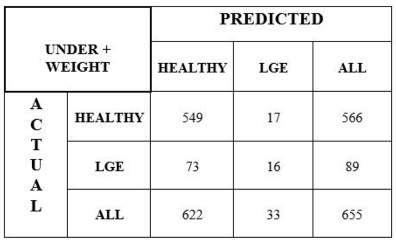
UNDERSAMPLED		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	545	21	566
	LGE	67	22	89
	ALL	612	43	655

(b)



WEIGHTED		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	560	6	566
	LGE	78	11	89
	ALL	638	17	655

(c)

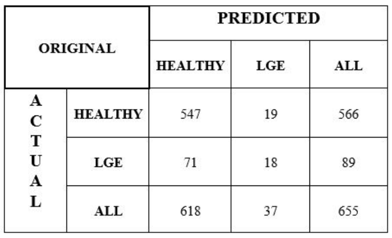


UNDER + WEIGHT		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	549	17	566
	LGE	73	16	89
	ALL	622	33	655

(d)

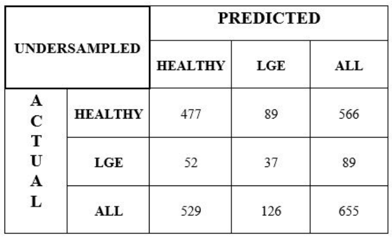
Table 5. From a to d: Confusion matrices for the pretrained VGG19 model retrained on the original, undersampled, weighted and undersampled + weighted datasets.

VGG19



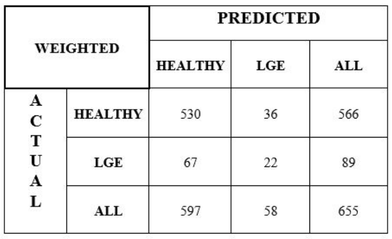
ORIGINAL		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	547	19	566
	LGE	71	18	89
	ALL	618	37	655

(a)



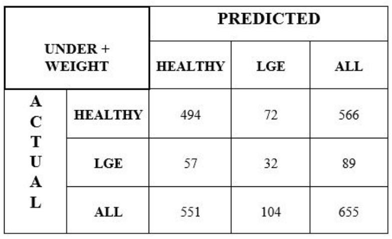
UNDERSAMPLED		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	477	89	566
	LGE	52	37	89
	ALL	529	126	655

(b)



WEIGHTED		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	530	36	566
	LGE	67	22	89
	ALL	597	58	655

(c)



UNDER + WEIGHT		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	494	72	566
	LGE	57	32	89
	ALL	551	104	655

(d)

Table 6. From a to d: Confusion matrices for the pretrained Inception V3 model retrained on the original, undersampled, weighted and undersampled + weighted datasets.

INCEPTION V3

ORIGINAL		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	566	0	566
	LGE	89	0	89
	ALL	655	0	655

(a)

UNDERSAMPLED		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	557	9	566
	LGE	86	3	89
	ALL	643	12	655

(b)

WEIGHTED		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	92	474	566
	LGE	9	80	89
	ALL	101	554	655

(c)

UNDER + WEIGHT		PREDICTED		
		HEALTHY	LGE	ALL
ACTUAL	HEALTHY	492	74	566
	LGE	65	24	89
	ALL	557	98	655

(d)

Table 7. Evaluation metrics (accuracy, precision, recall and F1-score) for the three pretrained models (VGG16, VGG19 and Inception V3) retrained on the four datasets.

DATASET	MODEL		HEALTHY			LGE		
		ACCURACY	PRECISION	RECALL	F1-SCORE	PRECISION	RECALL	F1-SCORE
ORIGINAL	VGG16	0.87	0.87	1.00	0.93	0.78	0.08	0.14
	VGG19	0.86	0.89	0.97	0.92	0.49	0.20	0.29
	INCEPTION V3	0.64	0.86	1.00	0.93	0	0	0
UNDERSAMPLED	VGG16	0.86	0.89	0.96	0.93	0.51	0.25	0.33
	VGG19	0.78	0.90	0.84	0.87	0.29	0.42	0.34
	INCEPTION V3	0.85	0.87	0.98	0.92	0.25	0.03	0.06
WEIGHTED	VGG16	0.87	0.88	0.99	0.93	0.65	0.12	0.21
	VGG19	0.84	0.89	0.94	0.91	0.38	0.25	0.30
	INCEPTION V3	0.26	0.91	0.16	0.28	0.14	0.90	0.25
UNDER + WEIGHT	VGG16	0.86	0.88	0.97	0.92	0.48	0.18	0.26
	VGG19	0.80	0.90	0.87	0.88	0.31	0.36	0.33
	INCEPTION V3	0.78	0.88	0.087	0.88	0.24	0.27	0.26

3.2. Visualization of Results

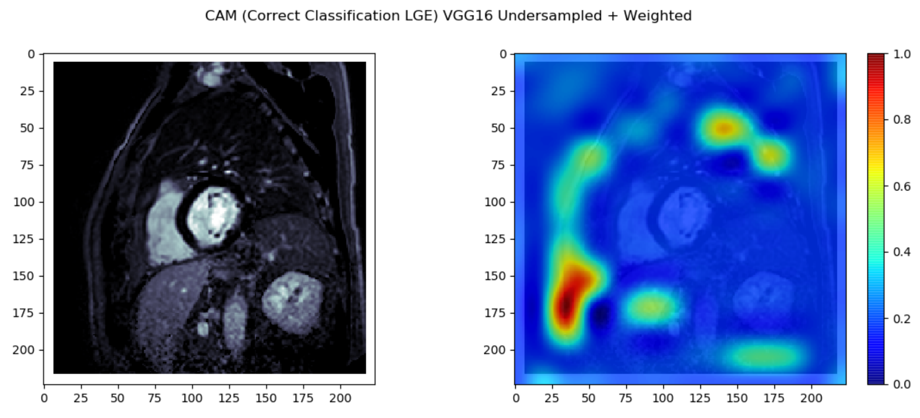
There are a number of tools to visualize the results better and understand what exactly each model is looking at when classifying images. For the case of this project, Class Activation Maps (CAMs) and Activation Maximization (AM) of the last fully connected layer of each model were obtained [26]. CAMs produce a gradient image that tells us what part of the image the model is looking at when classifying an image. AM can be used to get a representation of the input image that maximizes the final dense layer output corresponding to a certain class. In other terms, the map will show us which features the network expects to be able to identify a certain class. For example, if one of the classes was a bird and the model had been trained correctly, the AM would generate an input image which would be representative of what the model is expecting to see when classifying a bird (e.g. eyes and beaks) as it would have learnt the correct characteristics of what it means to be a bird.

Figure 4 illustrates the CAMs of an image that every model classified correctly as LGE when retrained on the undersampled and weighted dataset. Figure 5 shows what each model, also retrained on the undersampled and weighted dataset, was looking at when classifying an image incorrectly as LGE when it was actually healthy, except for the case of Inception V3 where it correctly predicted the image as healthy. Most CAMs showed that VGG19 was the only model that was looking at the heart to classify the images. VGG16 would tend to look at other parts of the image, mainly the borders to classify the images. Finally, Inception V3 looked at a broader section of the image, sometimes even looking at the full image itself and not a specific part.

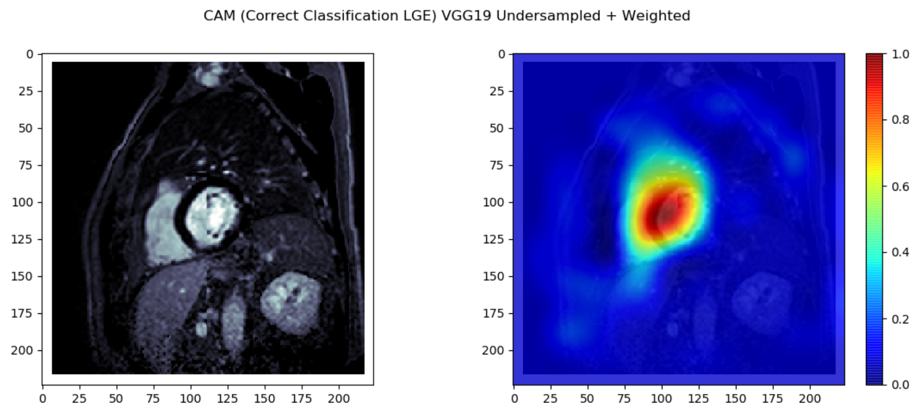
AM maps were obtained for both classes of all models retrained on the four datasets to see what features every model had learnt to differentiate between LGE and healthy images. Figure 6 and 7 represent the AM maps of both classes for the VGG19 model retrained on the undersampled dataset. In the case of this model (see Appendix 2 Figure 13 for more images), the AM maps for both classes have certain similarities, such as circular forms possibly belonging to the heart (one of the most relevant features) as VGG19 did pay more attention to it when classifying the images (shown in the CAMs). However, the AM map for the healthy class seems to show more defined shapes than the LGE map which seems to have blurred features. The colors on these maps are the result of the different combinations of the patterns obtained in the lower layers, where simple features such as edges and colors were extracted.

For the Inception V3 model AM maps (see Appendix 2 Figure 14), multiple blurred bright spots can be seen, possibly referring to different organs in the CMR images and not the heart. All AM maps for this model seemed to produce blurry features, verifying that the model had not captured the correct notion of what the heart or what myocardial damage was. VGG16 (see Appendix 2 Figure 12) seemed to use the shape of the heart as a feature when classifying images as healthy, but not when classifying images as

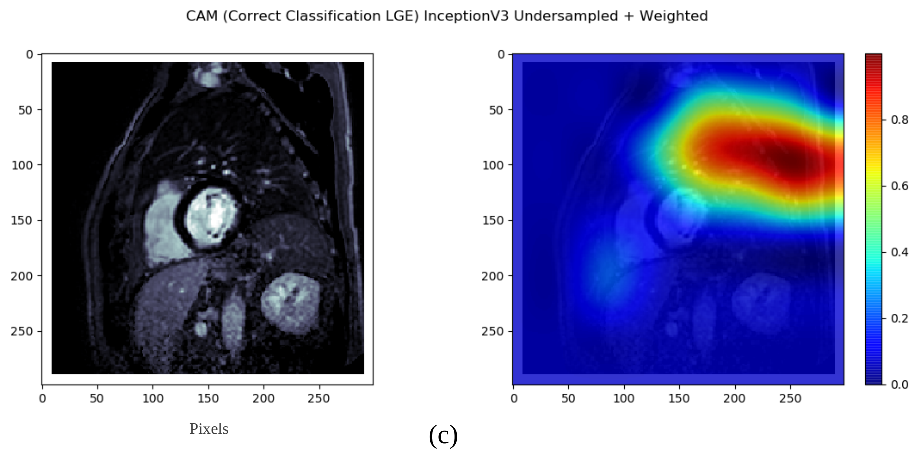
LGE. In contrast, as mentioned previously, VGG19 did capture the shape of the heart for both classes. However, more research is needed to properly understand these maps to be able to give more appropriate explanations as to what they mean.


CAM (Correct Classification LGE) VGG16 Undersampled + Weighted

(a)

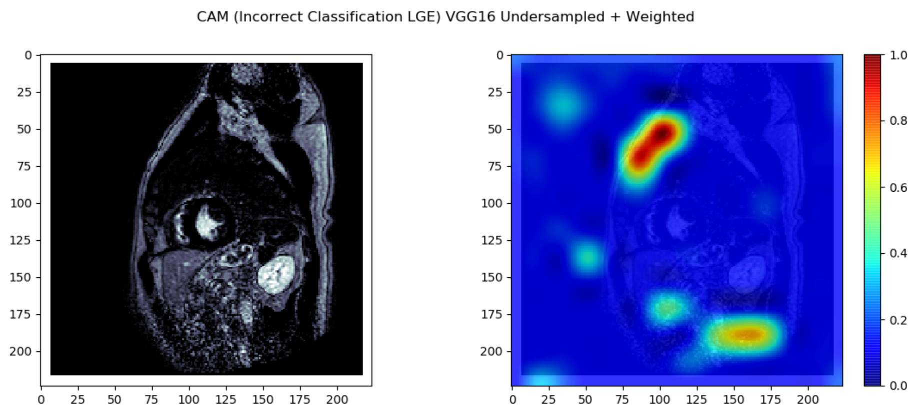

CAM (Correct Classification LGE) VGG19 Undersampled + Weighted

(b)

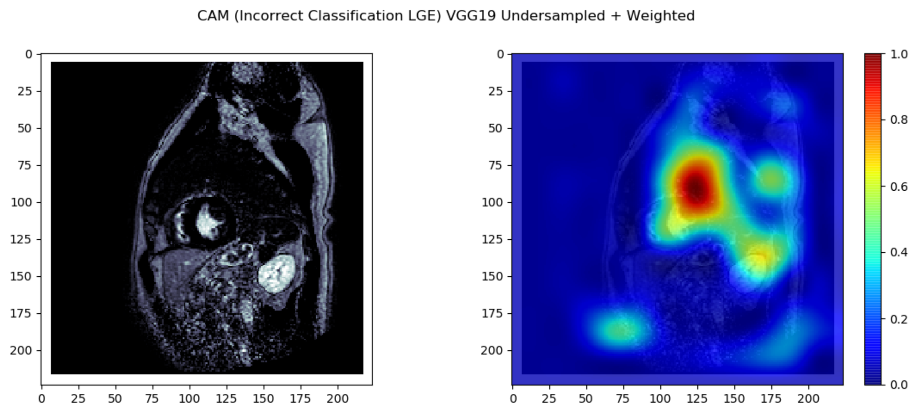

CAM (Correct Classification LGE) InceptionV3 Undersampled + Weighted
Pixels

(c)

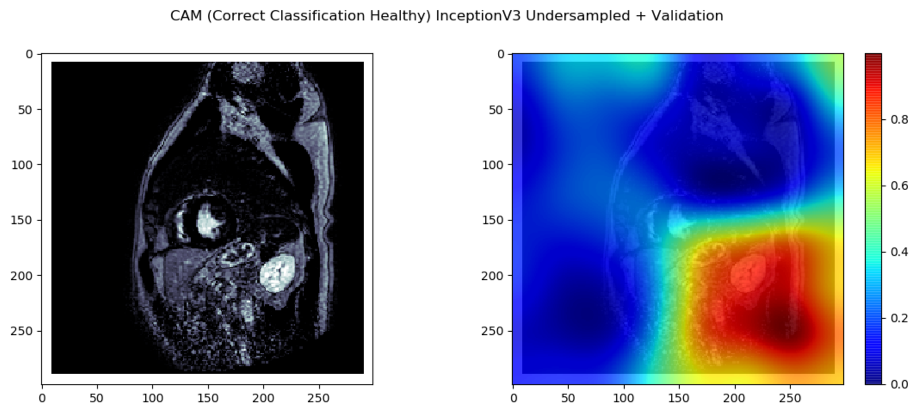
Figure 4. From a to c: CAMs of the VGG16, VGG19 and Inception V3 models, retrained on the undersampled and weighted dataset, for an image that had been correctly classified as LGE. The x and y axes correspond to the pixels of the image resized.


CAM (Incorrect Classification LGE) VGG16 Undersampled + Weighted

(a)


CAM (Incorrect Classification LGE) VGG19 Undersampled + Weighted

(b)


CAM (Correct Classification Healthy) InceptionV3 Undersampled + Validation

(c)

Figure 5. From a to c: CAMs of the VGG16, VGG19 and Inception V3 models, retrained on the undersampled and weighted dataset, for an image that had been classified incorrectly as LGE when it was healthy. Except for Inception V3, which classified the image correctly. The x and y axes correspond to the pixels of the image resized.

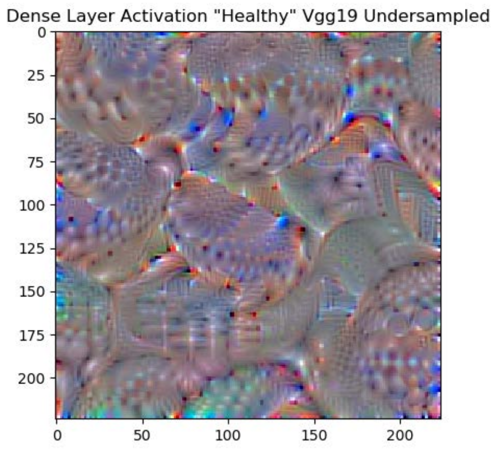


Figure 6. AM map of the healthy class for the VGG19 model trained on the undersampled dataset.

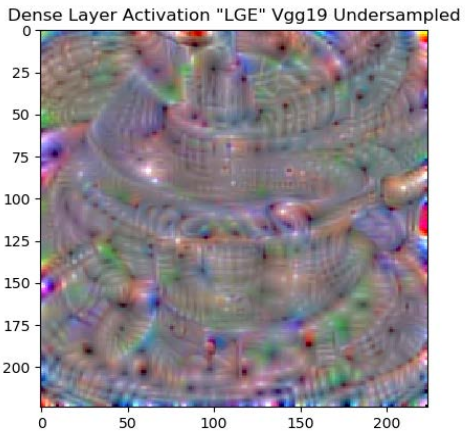


Figure 7. AM map of the LGE class for the VGG19 model trained on the undersampled dataset.

4. DISCUSSION

At the beginning of the study, two questions were proposed to be answered (1.5. Objective).

The **first question** was whether the deeper CNN architecture, Inception V3, would result in the model detecting and representing more complex relations between features and, so, produce better predictions. Based on the results obtained, this was not the case. When comparing the confusion matrices, Inception V3 tended to have the most questioning and varying values for all four datasets. The CAMs (3.2.Visualization of Results) obtained were an important part in understanding what the models were looking at when classifying. In the case of Inception V3, it was clearly not looking at the heart when classifying, proving that this model was not learning the important features. The AM maps (see Appendix 2 Figure 14) for this model provided clues as to what features it was expecting to see for each class. In this situation, the AM maps showed blurry shapes and were very different to the other two models, more evidence that the model had not learnt to detect the appropriate features (i.e. the heart and myocardial damage). The explanation as to why the deeper model did not prove to be the better one can be separated based on the datasets it was retrained on.

In the case of Inception V3 being retrained on the original dataset, the accuracy and loss graph (Appendix 2), seemed to show that the model was *overfitting*. This could be why the retrained model did not classify any image as LGE, as it had learnt to only classify images as healthy due to having more training examples of this class.

For the other three datasets on which the model was retrained, one possible reason as to why this deeper architecture did not provide better results could be the *vanishing gradient problem* [39], where, as the weights are updated from the higher layers to the lower layers during backpropagation, the gradients become smaller till a point where they are no longer changing the value of the weights of the lower layers and so, results in the network never reach to a good solution. In other terms, the model does not find the optimal weights to get proper results. Another reason could be the reverse, an *exploding gradient problem*, where the gradients keep getting higher values and so the weights are updated with huge values. The updated weights are so different to previous weights, that the optimal values are never achieved because the proportion to which the weights become updated during each epoch is too large.

The **second question** was whether a balanced data would be necessary to achieve more meaningful results, as the model would be trained on equal amounts of images of both classes. From the results shown in Tables 4, 5, 6 and 7, it is clear to see that using the imbalanced datasets results in the models usually classifying most images as healthy due to the small number of LGE samples to learn from. The use of the undersampled dataset allowed the models to classify images as LGE at a slightly higher amount of time. One possible reason as to why the undersampled dataset did not give significantly better results was that when this dataset was created, 65 % of the healthy samples were eliminated, and so, less images were used to train the models overall. If large amounts of images, of both classes, were gathered the models could potentially be trained to classify accurately at a higher rate.

Interestingly, using the class weight function on the original dataset, gave similar results to the undersampled dataset, meaning a higher amount of images were classified as LGE. So, although a balanced dataset is always better, using the class weight function on the original dataset and properly tuning the value of the weight the model gives each class during the training phase could lead the models to classify images as accurately as with the balanced dataset.

The **aim of the study** was to compare the models and see whether any of them could be used in a clinical setting. Out of all three models, VGG19 appeared to be the only model that looked at the heart in the images, although it is not clear whether it was looking at the diseased part of the myocardium to classify the images as LGE. If this was the case, we would expect the model to predict the images correctly, as it would have learnt to detect the enhanced pattern in the myocardium. It is also the model that provides recall values above the average, an important metric to ensure the models are not missing out on classifying actual diseased images.

These findings have allowed both questions to be answered, the first being that a deeper model did not perform better and that a balanced dataset would be optimal if the use of class weight was not available or the values could not be properly defined. This study was done to see whether any of the models could potentially be used in a clinical setting. At the moment, none of the models can be used, but properly modifying and retraining VGG19 on a cleaner dataset could result in the model giving desirable results.

4.1. Future Work

DL models require lots of trial and error, and so, multiple modifications can be proposed to make the models train better. In the case of using a pretrained model to discriminate between healthy and diseased myocardium, there is still lots of work to be done. Below, a list of possible projects and tasks that have not been tackled or discussed and would be interesting to do are proposed:

- *Deleting those slices that could confuse the model*. When preparing the dataset, all PSIR images acquired were used, even those that did not show any section of the heart. This was because the aim was to try and simulate a real life scenario where the minimum human intervention was needed, and in this situation a physician would be able to send all the slices without deleting any. Because the results showed poor performance, by deleting these images (examples shown in Figure 8), the models might be able to properly train. These were correctly classified as healthy, but seeing as they had nothing to do with what the algorithm should be looking at (i.e. the heart), it could confuse the model as to what features to pay attention to.

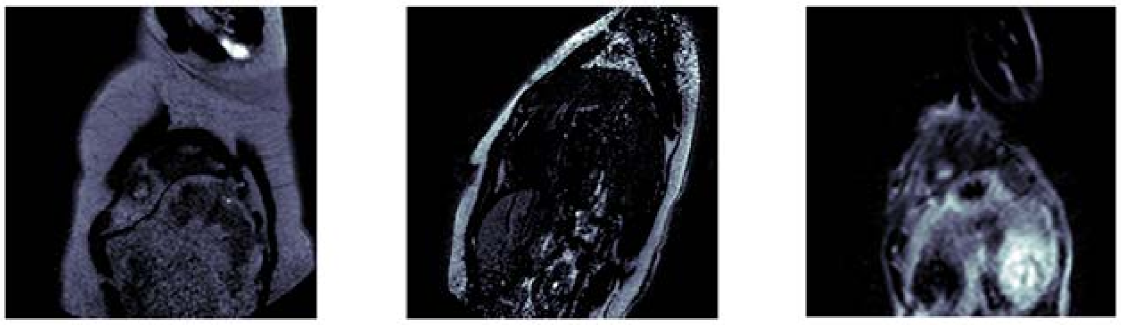


Figure 8. CMR images of the top and bottom part of the sequence stack that could be misleading for the models

- *Retraining some of the feature learning layers*. During this study, only the classifiers were retrained and adapted to the new classification problem. Trying to retrain a few of the convolutional layers, could allow the model to find new features specifically targeted for this type of data (i.e. CMR images) and then relate them in the classifier.
- *Changing the learning rate*. During the weight updating process, the learning rate was set to 0.001, by trying different values the results can change.
- *Cropping the CMR images to only show the heart.* As the models seemed to be getting confused on what to focus on, feeding the models images of only the heart region could prove to be effective. Same as the first point, the aim was to not need any human intervention finally.
- *Using the segmentations to train the models*. The physician who manually labeled each LGE-CMR image also segmented the LGE pattern. This segmentation could be

used to help train the models, as we would be able to show the model where there is myocardial damage. This point can be used simultaneously with the previous point.

- *Applying image data augmentation*. One downside of using the undersampled dataset was that less training samples were used. One famous method, included in the Keras library, is data augmentation which artificially increases the diversity of the data by applying transformation functions on the dataset. This has been proven to be effective in multiple scenarios [35], but the correct techniques (e.g. rotation or brightness) must be chosen.

4.2. Conclusion

DL is proving to be a good technique in assisting medical personnel in various departments. However, a big obstacle is the need of large amounts of data to properly train the models. Transfer learning on pretrained models has helped to overcome this problem by retraining models on new and small datasets. The main goal of this study was to compare three state-of-the-art DL models in the classification of images between healthy and diseased (LGE) myocardium with the use of transfer learning. Four datasets were used to retrain the models: the original imbalanced dataset (Table 1), an undersampled or balanced dataset (Table 2), the original dataset with the use of the “class weight” function and the undersampled dataset with the use of the “class weight” function. After training all three models on the four datasets, the results on the testing images were compared and analyzed. VGG16 and VGG19 were chosen because of their simplicity. Both models, created by the same research group, were identical except for the number of convolutional layers they had. In the case of VGG16, the model was composed of 13 convolutional layers and 3 fully connected layers. VGG19 had 16 convolutional layers and 3 fully connected layers. Inception V3 was chosen due to its use of inception modules, which would allow for a multi-feature layer extractor and its 42 layer depth. Results showed that VGG19 performed better on average and that training the models on a balanced dataset slightly improved their performance. Nevertheless, further work is needed to demonstrate that DL can be used in this type of medical classification problem.

5. LIST OF FIGURES

Page

6. LIST OF TABLES

Page

7. BIBLIOGRAPHY

[1] American Heart Association. (2017). Heart. *What is heart failure*. Retrieved April 22, 2020, from https://www.heart.org/en/health-topics/heart-failure/what-is-heart-failure

[2] Vassiliou, V. S., Cameron, D., Prasad, S. K., & Gatehouse, P. D. (2018). Magnetic resonance imaging: Physics basics for the cardiologist. *JRSM Cardiovascular Disease*.

https://doi.org/10.1177/2048004018772237

[3] Murphy, A., Dr. Ray Ballinger, J. et al. Radiopaedia. *MRI physics* (n.d.). Retrieved February 10, 2020, from https://radiopaedia.org/articles/mri-physics

[4] Satoh, H., Sano, M., Suwa, K., Saitoh, T., Nobuhara, M., Saotome, M., Urushida, T., Katoh, H., & Hayashi, H. (2014). Distribution of late gadolinium enhancement in various types of cardiomyopathies: Significance in differential diagnosis, clinical features and prognosis. *World Journal of Cardiology*, 6(7), 585–601.

https://doi.org/10.4330/wjc.v6.i7.585

[5] Kellman, P., & Arai, A. E. (2012). Cardiac imaging techniques for physicians: late enhancement. *Journal of magnetic resonance imaging: JMRI*, 36(3), 529-542.

https://doi.org/10.1002/jmri.23605

[6] Siegersma, K. R., Leiner, T., Chew, D. P., Appelman, Y., Hofstra, L., & Verjans, J. W. (2019). Artificial intelligence in cardiovascular imaging: state of the art and implications for the imaging cardiologist. *Netherlands heart journal : monthly journal of the Netherlands Society of Cardiology and the Netherlands Heart Foundation*, 27(9), 403–413. https://doi.org/10.1007/s12471-019-01311-1

[7] DeepMind. (n.d.). *AlphaGo*. Retrieved June 8, 2020, from https://deepmind.com/research/case-studies/alphago-the-story-so-far

[8] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *IJCV*. https://doi.org/10.1007/s11263-015-0816-y

[9] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *IEEE Xplore*, 86(11), 2278-2323.

https://doi.org/10.1109/5.726791

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *NIPS,* 25 1090-1098.
https://doi.org/10.1145/3065386

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9. https://doi.org/10.1109/CVPR.2015.7298594

[12] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*), 2818-2826.
https://doi.org/10.1109/CVPR.2016.308

[13] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR).* https://arxiv.org/abs/1409.1556

[14] Kwon, JM., Lee, Y., Lee, Y., Lee, S. & Park, J. (2018). An algorithm based on deep learning for predicting in-hospital cardiac arrest. *J Am Heart Assoc*. 2018;7.

https://doi.org/10.1161/JAHA.118.008678

[15] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. S., Liang, H., Baxter, S. L.,McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M. K., Pei, J., Ting, M. Y. L., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., Shi, A., Zhang, R., Zheng, L., Hou, R., Shi, W., Fu, X., Duan, Y., Huu, V. A. N., Wen, C., Zhand, E. D., Zhang, C. L., Li, O., Wang, Z., Singer, M. A., Sun, X., Xu, J., Tafreshi, A., Lewis, M. A., Xia, H. & Zhang, K. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*. 172(5), 1122–1131. https://doi.org/10.1016/j.cell.2018.02.010

[16] Vianna, V. P. (2018). Study and development of Computer-Aided Diagnosis system for classification of chest x-ray images using convolutional neural networks pre-trained for ImageNet and data augmentation. https://arxiv.org/abs/1806.00839

[17] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M. & Thrun, S. (2017) Dermatologist-level classification of skin cancer with deep neural networks. *Nature*. 542(7639), 115–118. https://doi.org/10.1038/nature21056.

[18] Rosa, E., Sidibé, D., Decourselle, T., Leclercq, T., Cochet, A. & LAlande, A. (2019). Myocardial Infarction Quantification from Late Gadolinium Enhancement MRI using top-hat transforms and neural networks. Submitted to *IEEE*. https://arxiv.org/abs/1901.02911

[19] Shu, Mengying. (2019). Deep learning for image classification on very small datasets using transfer learning. *Creative Components*. 345. https://lib.dr.iastate.edu/creativecomponents/345

[20] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: A system for large-scale machine learning. Software available from tensorflow.org

[21] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras. Software available from https://keras.io

[22] Roy, R. (n.d.). GeeksforGeeks. *Best Python libraries for Machine Learning*. Retrieved February 10, 2020, from https://www.geeksforgeeks.org/best-python-libraries-for- machine-learning/

[23] Arterys Inc. (2018). Medical Imaging Cloud Ai. Retrieved from https://arterys.com

[24] Rosebrock, A. (2017) pyimagesearch. *ImageNet: VGGNet, ResNet, Inception, and Xception with Keras*. Retrieved March 15, 2020, from https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

[25] Raj, B. (2018). TowardsDataScience. *A Simple Guide to the Versions of the Inception Network.* Retrieved March 15, 2020 from https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[26] Purkait, N. (2019). Convolutional Neural Networks. In Packt Publishing Ltd (Eds.), *Hands-On Neural Networks with Keras: Design and create neural networks using deep learning and artificial intelligence principles* (pp. 159 - 163). Retrieved from https://books.google.es/books?id=lc6PDwAAQBAJ

[27] Van Rossum, G. & Drake, F.L. (2009). Python Software Foundation. Python Language Reference,version 3.0. Software available from https://www.python.org

[28] Sharma, V. (2018). VinodsBlog. *Deep Learning - Introduction to Convolutional Neural Networks*. Retrieved January 26, 2020 from

https://vinodsblog.com/2018/10/15/everything-you-need-to-know-about-convolutional-neural-networks/

[29] Dertat, A. (2017). TowardsDataScience. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. Retrieved January 26, 2020 from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[30] Elster, A., D. (n.d.). MRIquestions. *Phase Sensitive Inversion Recovery (PSIR).* Retrieved January 23, 2020 from http://mriquestions.com/ps-phase-sensitive-ir.html

[31] Kellman, P., Arai, A. E., McVeigh, E. R., & Aletras, A. H. (2002). Phase-sensitive inversion recovery for detecting myocardial infarction using gadolinium-delayed hyperenhancement. *Magnetic resonance in medicine, 47*(2), 372–383. https://doi.org/10.1002/mrm.10051

[32] Jay, P. (2017). Medium. *Transfer Learning using Keras.* Retrieved March 25, 2020 from https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8

[33] Araújo, J., R. (2018). Medium. *First steps with Transfer Learning for custom image classification with Keras.* Retrieved April 2, 2020 from https://medium.com/abraia/first-steps-with-transfer-learning-for-custom-image-classification-with-keras-b941601fcad5

[34] Schindler, A., Lidy, T., & Rauber, Andreas. (2016). Comparing Shallow versus Deep Neural Network Architectures for Automatic Music Genre Classification. In *9th Forum Media Technology (FMT2016)*, 1734, 17–21. https://publik.tuwien.ac.at/files/publik_256008.pdf

[35] Shorten, C., & Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J Big Data* 6, 60. https://doi.org/10.1186/s40537-019-0197-0

[36] Graziani, M., Andrearczyk, V., & Müller, H. (2018). Visual interpretability for patch-based classification of breast cancer histopathology images.

https://openreview.net/pdf?id=S1PTal9sz

[37] Mayo Clinic. (2020). Heart failure. *Heart failure – Symptoms and causes*. Retrieved April 1, 2020, from https://www.mayoclinic.org/diseases-conditions/heart-failure/symptoms-causes/syc-20373142

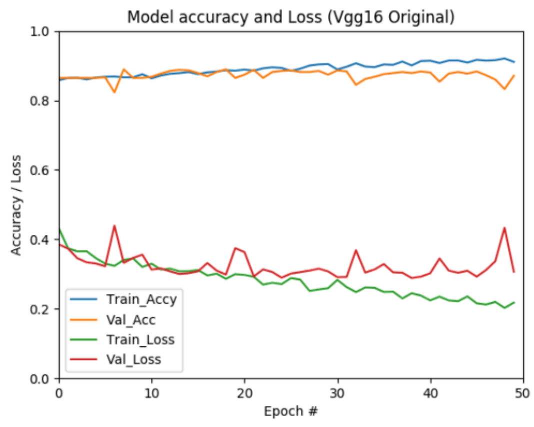
[38] Sebastian Ruder. (2016). *An overview of gradient descent optimization algorithms*. Retrieved March 5, 2020, from https://ruder.io/optimizing-gradient-descent/

[39] Wang, Chi. (2019). TowardsDataScience. *The Vanishing Gradient Problem*. Retrieved May 25, 2020, from https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

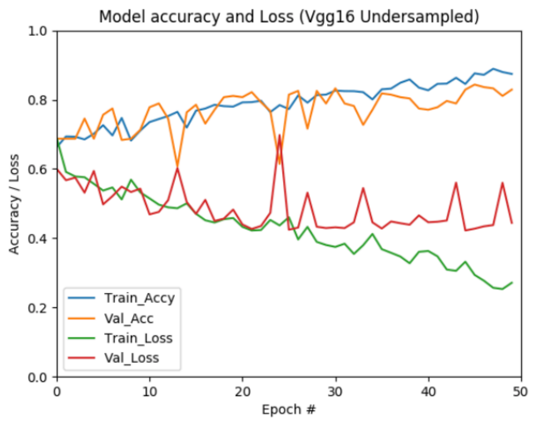
8. APPENDICES

8.1. Appendix 1

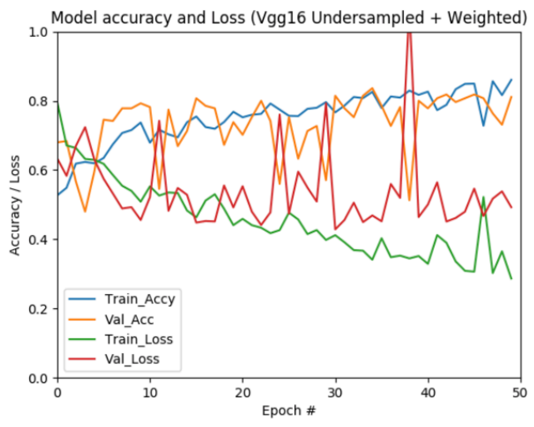
VGG16



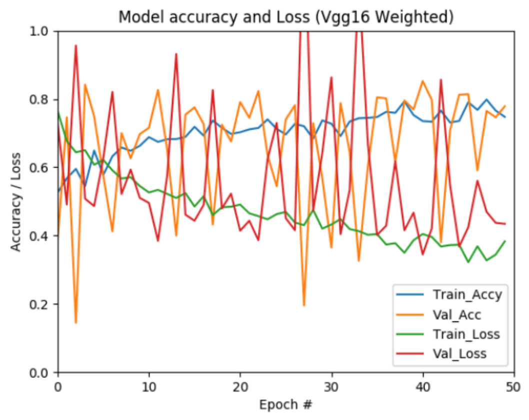
(a)



(b)



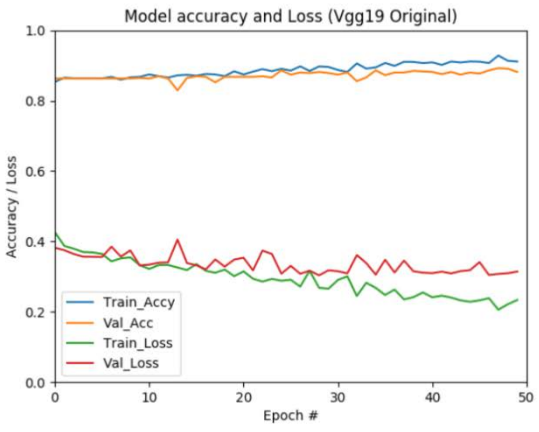
(c)



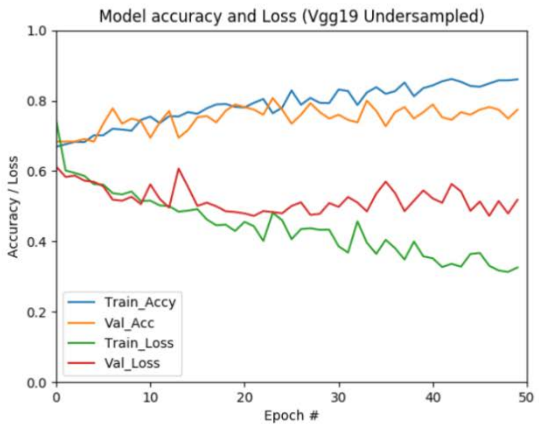
(d)

Figure 9. VGG16 accuracy and loss graphs obtained during the training and validating phase on all four datasets. From top left to bottom right, graphs for VGG16 model retrained on: original dataset undersampled dataset, original dataset with class weight function and undersampled dataset with class weight function.

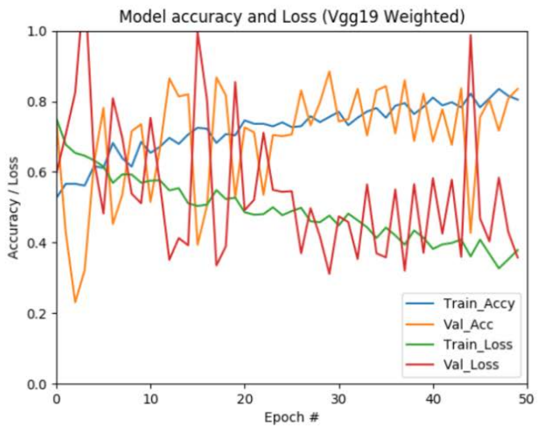
VGG19



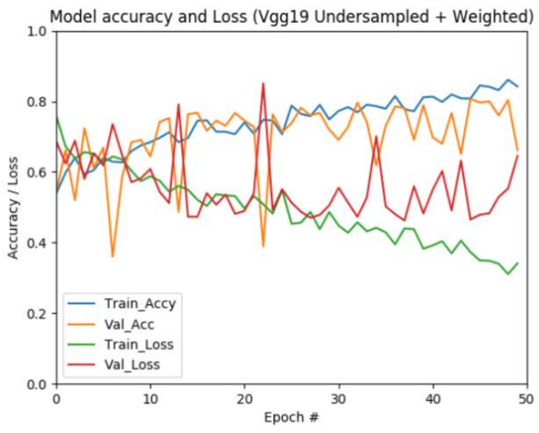
(a)



(b)



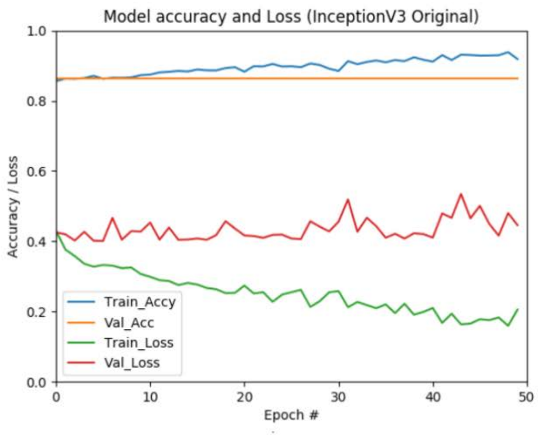
(c)



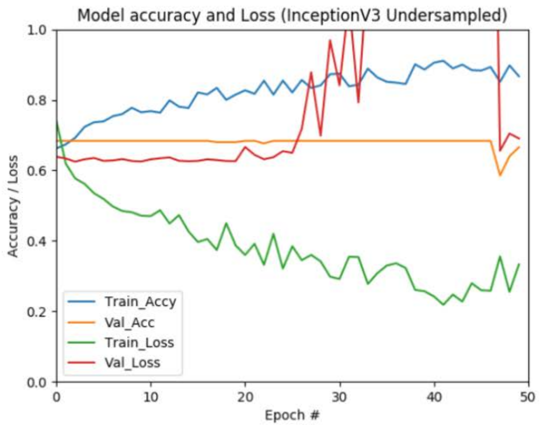
(d)

Figure 10. VGG19 accuracy and loss graphs obtained during the training and validating phase on all four datasets. From top left to bottom right, graphs for VGG19 model retrained on: original dataset undersampled dataset, original dataset with class weight function and undersampled dataset with class weight function.

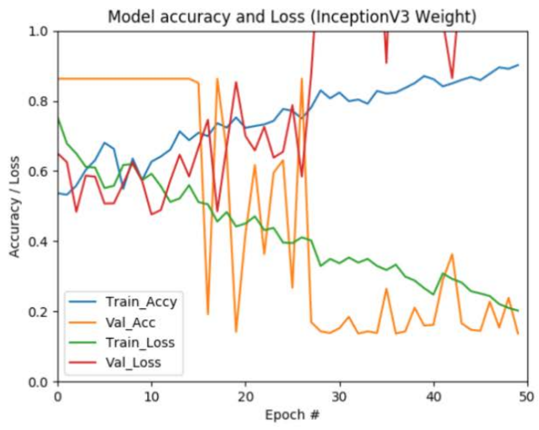
INCEPTION V3



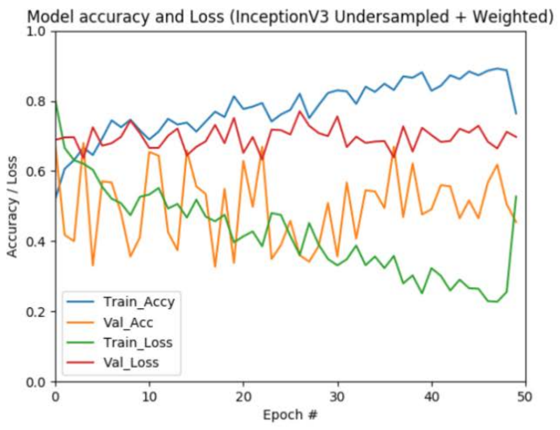
(a)



(b)



(c)



(d)

Figure 11. Inception V3 accuracy and loss graphs obtained during the training and validating phase on all four datasets. From top left to bottom right, graphs for Inception V3 model retrained on: original dataset undersampled dataset, original dataset with class weight function and undersampled dataset with class weight function.

8.2. Appendix 2

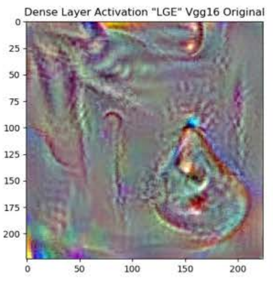
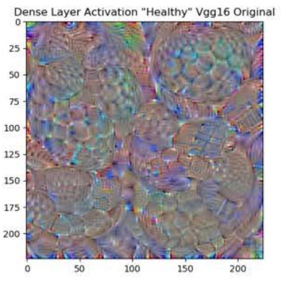
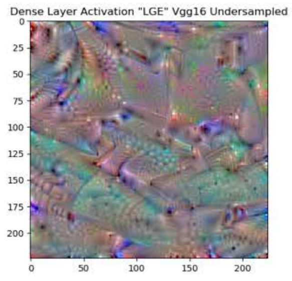
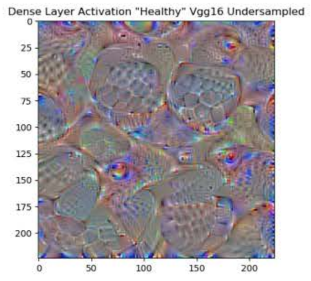
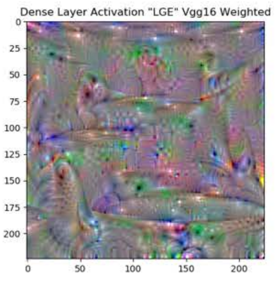
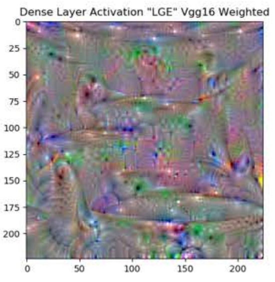
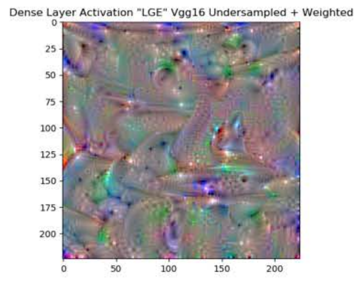
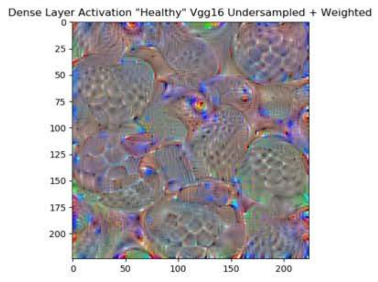
VGG16		
DATASET	**LGE**	**HEALTHY**
Original		
Undersampled		
Weighted		
Undersampled + Weighted		

Figure 12. AM maps of both classes (LGE and healthy) for the VGG16 model retrained on all four datasets.

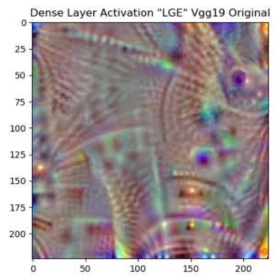
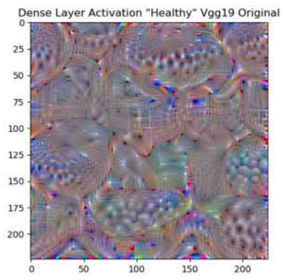
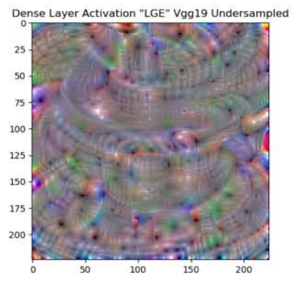
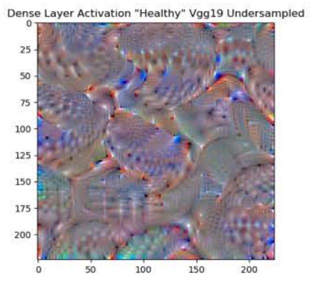
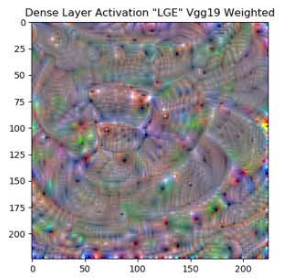
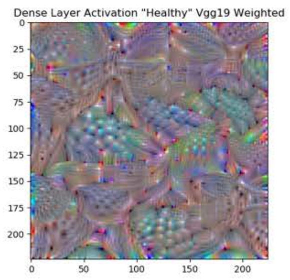
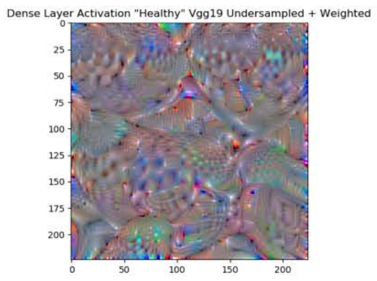
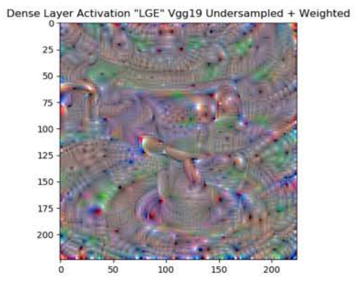
VGG19		
DATASET	**LGE**	**HEALTHY**
Original		
Undersampled		
Weighted		
Undersampled + Weighted		

Figure 13. AM maps of both classes (LGE and healthy) for the VGG19 model retrained on all four datasets.

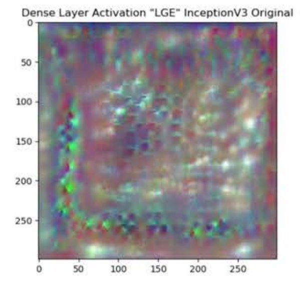
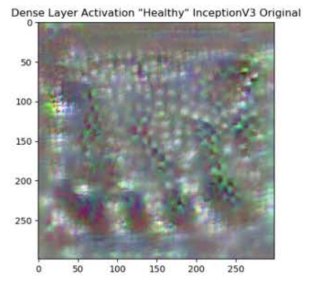
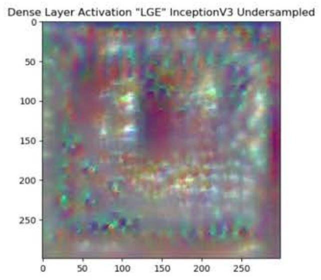
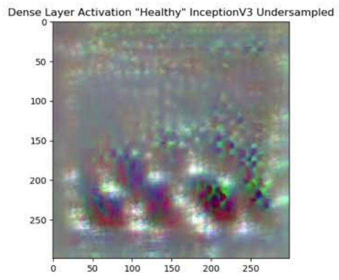
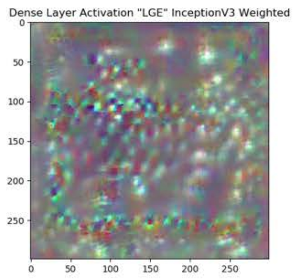
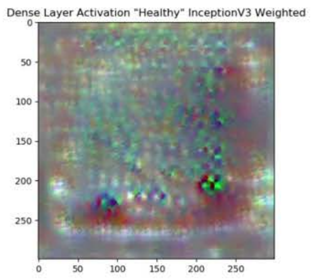
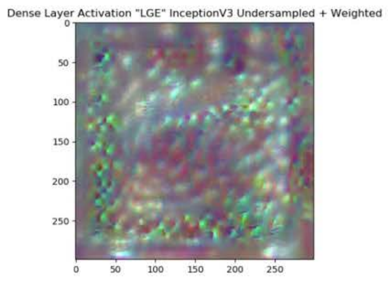
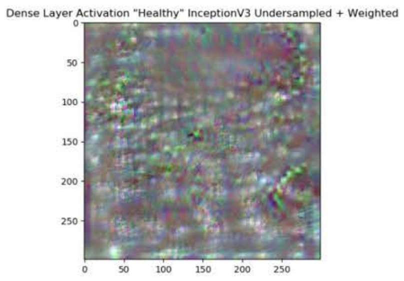
INCEPTION V3		
DATASET	**LGE**	**HEALTHY**
Original		
Undersampled		
Weighted		
Undersampled + Weighted		

Figure 14. AM maps of both classes (LGE and healthy) for the Inception V3 model retrained on all four datasets.

8.3. Appendix 3

Summary of the architecture of the three models with their new classifier, highlighted in blue, have been added below.

VGG16

```
Layer (type)                   Output Shape                 Number Parameters
==============================================================================
Input Layer                    (None, 224, 224, 3)           0

block1_conv1 (Conv2D)          (None, 224, 224, 64)          1792

                                        .
                                        .
                                        .

block5_conv3 (Conv2D)          (None, 14, 14, 512)           2359808

block5_pool (MaxPooling2D)     (None, 7, 7, 512)             0

global_average_pooling         (None, 512)                   0

dense_1 (Dense)                (None, 4096)                  2101248

dense_2 (Dense)                (None, 1024)                  4195328

dense_3 (Dense)                (None, 2)                     1025
==============================================================================
Total parameters: 21,013,312
Trainable parameters: 6,298,626 → weights that were retrained on the four datasets.
Non-trainable parameters: 14,714,688 → weights of the frozen layers that were not updated.
```

Figure 15. Summary of the VGG16 model with the new classifier, highlighted in blue. The models final layer was a 2 node classifier, corresponding to LGE and healthy class. The number of parameters that were retrained was 6.298.626, equal to the sum of the number of parameters of the 3 fully connected layers.

VGG19

```
_____________________________________________________________________________
Layer (type)                    Output Shape              Number Parameters
=============================================================================
Input Layer                  (None, 224, 224, 3)                0
_____________________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)              1792
_____________________________________________________________________________
                                      .
                                      .
                                      .
_____________________________________________________________________________
block5_conv4 (Conv2D)        (None, 14, 14, 512)              2359808
_____________________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)                   0
_____________________________________________________________________________
global_average_pooling          (None, 512)                      0
_____________________________________________________________________________
dense_1 (Dense)                 (None, 4096)                  2101248
_____________________________________________________________________________
dense_2 (Dense)                 (None, 1024)                  4195328
_____________________________________________________________________________
dense_3 (Dense)                  (None, 2)                      2050
=============================================================================
Total parameters: 26,323,010
Trainable parameters: 6,298,626 → weights that were retrained on the four datasets.
Non-trainable parameters: 20,024,384→ weights of the frozen layers that were not updated.
```

Figure 16. Summary of the VGG19 model with the new classifier, highlighted in blue. The models final layer was a 2 node classifier, corresponding to LGE and healthy class. The number of parameters that were retrained was 6.298.626, equal to the sum of the number of parameters of the 3 fully connected layers.

Inception V3

Layer (type)	Output Shape	Number Parameter	Connected to
Input Layer	(None, 299, 299, 3)	0	
conv2d_1(Conv2D)	(None,149,149,32)	864	input_1[0][0]
	⋮		
activation_94(Activation)	(None,8,8,192)	0	batch_normalization_94[0][0]
mixed10 (Concatenate)	(None, 8,8, 2048)	0	activation_86[0][0] mixed9_1[0][0] concatenate_2[0][0] activation_94[0][0]
Global_average_pooling	**(None,2048)**	**0**	**mixed10[0][0]**
dense_1(Dense)	**(None,1024)**	**2098176**	**global_average_pooling[0][0]**
dense_2(Dense)	**(None,1024)**	**1049600**	**dense_1[0][0]**
dense_3(Dense)	**(None,512)**	**524800**	**dense_2[0][0]**
dense_4(Dense)	**(None,2)**	**1026**	**dense_3[0][0]**

Total parameters: 25,476,386
Trainable parameters: 3,673,602 → weights that were retrained on the four datasets.
Non-trainable parameters: 21,802,784→ weights of the frozen layers that were not updated.

Figure 17. Summary of the Inception V3 model with the new classifier, highlighted in blue. The models final layer was a 2 node classifier, corresponding to LGE and healthy class. The number of parameters that were retrained was 3.673.602, equal to the sum of the number of parameters of the 4 fully connected layers.