Automatic coronary calcium scoring in multi-center lung cancer screening trial

Student: Ewout Reinders (3759512)
Supervisors: Ivana Išgum and Robby T. Tan
Thesis number: ICA-3759512
Institute: University Utrecht

December 19, 2013

Contents

Chapter 1

Abstract

Machine learning plays an essential role in medical image analysis. Supervised machine learning methods typically assume that the training data and task data have an identical distribution. However, this assumption may not always hold. In medical imaging, distributions might change due to for example, variations in the image acquisition protocols or pathology shown. Acquiring a (large) new representative training set for every set of images with a different distribution is often very time consuming or practically impossible. Instead, it would be advantageous to reuse the existing training data to analyze new data having different distribution. Methods enabling such transfer of knowledge are called transfer learning methods. In this thesis, we propose a novel transfer learning method, inspired by TrAdaboost [10], that uses an iterative weighted nearest neighbor classifier to extract knowledge from one distribution (source) to analyze the data originating from another distribution (target). Our method first identifies parts of the source data useful for the analysis of the target data using a small set of the labeled target data. The classifier is subsequently trained using these selected source and target sets. The method was applied to automatic coronary calcium scoring with chest CT scans acquired in a multi-center trial. The source and the target data were represented with scans acquired in two different centers with different CT scanners. Performance of the proposed system was compared with the system described by Išgum et al [14] using standard nearest neighbor classification. The results showed that transfer learning approach improved classification. The achieved sensitivity (detected coronary calcifications) was high, but further investigation might be needed to reduce false positive rate.

Chapter 2

Introduction

Atherosclerosis is a disease in which plaque builds up. Plaque is a combination of fat, cholesterol, calcium, and other substances found in the blood. Plaque in coronary arteries reduces blood flow to the heart muscle. This may lead to cardiovascular diseases and is the leading cause of death worldwide (www.who.int).

In clinical practice, detection of plaques in the coronary arteries is usually performed with ECG-synchronized cardiac CT scans without contrast enhancement. In these CT scans human operators manually search in coronary areas that may represent calcifications. After the calcifications are found, a score is determined called the *calcium score*. There are several ways to express the calcium score, for example by volume, mass or Agatston. The Agatston coronary calcium score is used to determine the risk of cardiovascular and has been proved to be an independent and strong predictor of cardiovascular events [5],[6],[15],[25],[26]. Detected plaque could be quantified with any CT scan that visualizes the heart, for example chest CT scans. However, chest CT is usually not used to detect coronary calcium.

Software packages are used to show areas with high intensity and human operators decide which areas represent plaque or not. This is a time consuming process, especially if the CT scan contains a lot of slices, so it might be interesting to detect cardiovascular diseases automatically. Several methods have been proposed for automatic detection of cardiovascular diseases using cardiac CT [4],[13],[18],[24]. Išgum et al [14] have proposed a machine learning approach for automatic coronary calcium scoring in the low-dose chest CT. This machine learning approach could be divided into three stages; candidate selection to extract candidates, feature computation for the calculation of features for every candidate and a supervised classification to predict whether a candidate is calcium or not. An schematic overview of this system is shown in figure 2.1.

Supervised classification is powerful technique of identifying to which of a set of categories (sub-populations) a new observation belongs, based on data containing observations whose category is known. However, supervised classifiers also have some restrictions. One of these restrictions is that manual annotated observations, known as *training data*, and the new observations, known as new data or the *task*, must be drawn from the same feature space and same distribution. In practice, this assumption does not always hold. If we want to classify a task that is acquired with a different settings, we need to gain new training data that are

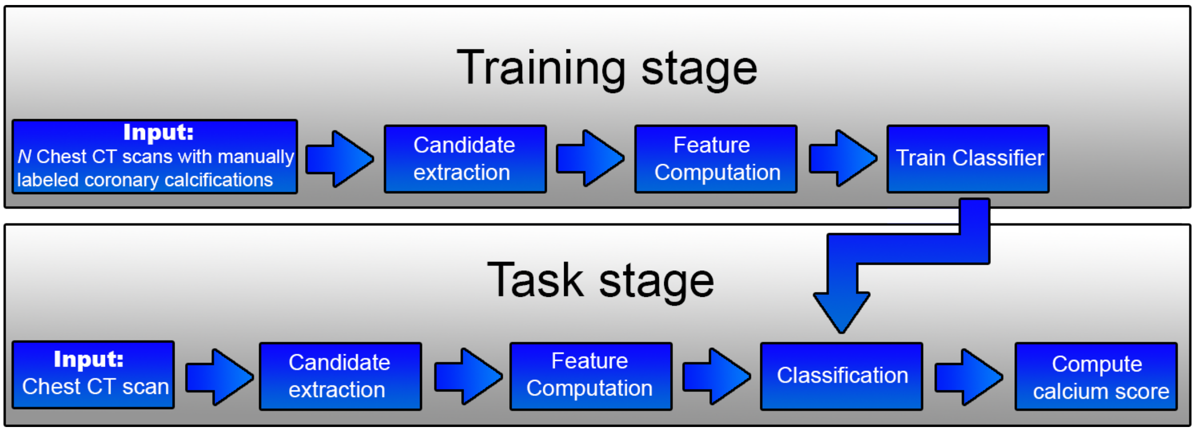

Figure 2.1: *A schematic overview that shows the training and the task stage of system of Išgum et al [14]. The input variable N in the training stage indicates the number of scans that are used for training.*

acquired with the same settings as the task and rebuild the classifier. Especially in medical imaging, acquiring a complete new training set for every task is a time consuming process. Training sets for supervised classification needs to be large, manually annotated and analyzed by human operators.

The goal of this project is to reduce the effort, acquiring a new training set for every task that is acquired with a different settings. We use a complete training set that is acquired with different settings compared to the task, the differently distributed training data, and a small training set that is acquired with the same settings as the task, the same distributed training data. The basic idea of our approach is to find useful data in the different distributed training data, based on the small same distributed training set. Combining the small same distributed training set and useful different distributed training data, we are able to create a new classifier that can be used for classifying the task. The field that works on this problem is called *transfer learning* and we going to apply our method on the data of the NELSON trial. NELSON is a heavy smokers screening trial that investigates whether chest CT screening will decrease lung cancer mortality compared to no screening (www.nelsonproject.nl).

The remainder of this paper is organized as follows. In the second chapter, we provide related work about pattern recognition, machine learning and transfer learning. In the next chapter, we will discuss our approach, implementation and decisions. Our data explanation, results and experiments will be discussed in chapter four. Finally, we will conclude and discuss future work.

Chapter 3

Related work

In this paper, we are going to change the classification stage in the automatic coronary calcium detection system of Išgum et al [14]. In the classification stage, Išgum et al [14] use a machine learning approach to determine whether candidates are calcifications or not. Machine learning makes predictions on the new data using statistical models that are trained on previously collected labeled or unlabeled training data [23]. In machine learning, statistical pattern recognition algorithms are used to assign labels to a given input value. Machine learning can be divided into four groups; supervised, semi-supervised, unsupervised and reinforcement learning.

Supervised learning use labeled training data to train a classifier and use the trained classifier to label new data. The labeled data in the training stage is often manually annotated. Examples of supervised algorithms are SVM [7], kNN (k-nearest neighbor) [8] and decision tree learning [9].

In semi supervised learning, there are two training sets; a large unlabeled training set and a small labeled training set. The basic idea of semi supervised learning is to label the large unlabeled training set based on the small labeled training set. When all the training data is labeled, a classifier is created and could be used to label the task. The motivation for this technique is that it may be expensive obtain labeled data, whereas obtaining unlabeled is relatively inexpensive. For example, Tur et al [27] applied semi supervised learning for speech recognition, because speech labeling is a labor intensive and time consuming process.

Unsupervised learning only uses unlabeled data for training. The main goal of unsupervised learning is to find a hidden structure or pattern in the training data. The difference between supervised and unsupervised learning is that unsupervised learning needs to find the labels based on the data structure, while labels are available for supervised and semi supervised learning. One of the most popular unsupervised learning methods is clustering [17]. Over the years, scientist purposed several clustering methods and that were based on the k-means algorithm [3] to solve unsupervised learning problems.

Reinforcement learning is a technique that make decisions based on situations from the past and is often used in robotics. For example, Bakker et al [1] proposed a reinforcement learning method that is able to built a probabilistic model of the environment, based on experiences within the environment.

3.0.1 Transfer learning

As we described in the introduction, obtaining labeled training data for every new task could be costly. Semi supervised learning and unsupervised learning are possible solutions to reduce labeling effort for training data. However, the restriction of these methods is that the distribution and feature space of the task data, also known as the *target domain*, and the training data, also known as the *source domain*, should be identical. If the target domain is obtained with a different protocol or device compared to source domain, distribution or feature space might be different. It would be nice to reduce the need and effort to recollect new training data for the target domain. The field that is working on this problem is called *transfer learning*. According to the survey of Pan et al [22], transfer learning could be divided into three groups; inductive, transductive and unsupervised transfer learning.

Inductive transfer learning assumes that a lot of labeled data from the source domain and some labeled data from the target domain is available. In this setting, labeled data from the target domain is used to determine which parts of the labeled data from the source domain could be reused for the target task. Transductive transfer learning assumes that the source and target tasks are the same, while the source and target domains are different. In transductive transfer learning, a lot of labeled data from the source domain is available, but all the training data of the target domain is unlabeled. In unsupervised transfer learning is similar to inductive transfer learning, except that unsupervised transfer learning has no labeled data available in both source and target domains during training. Unsupervised transfer learning tries to solve these problems by using clustering or dimensionality reduction techniques.

Our approach is an inductive transfer learning approach where a lot of labeled source data and some labeled target data is available. The goal of inductive transfer learning is to determine which parts of the labeled data from the source domain could be reused, based on the some labeled target data, for the target task. Pan et al [22] categorized inductive transfer learning approaches in four categories; feature, parameter, relational-knowledge and instance based transfer learning. Our approach is an instance based transfer learning approach. The basic idea of instance based transfer learning is to use some labeled target data to find overlapping parts in the source data that can be used to extend the labeled target data. Instance based transfer learning does not require a certain representation, the only restriction is that the representation of the target data and source data is identical. An example of an instance based transfer learning algorithm is TrAdaboost [10], which is an extension of the Adaboost framework [12]. Adaboost, a machine learning framework, stands for Adaptive Boosting and aims to boost the accuracy of a weak learner, such as SVM. After each iteration, Adaboost carefully adjusts the weights of training data using a classifier. However, Adaboost is a machine learning framework which assumes that the distribution of the training set and task set is identical. TrAdaboost is the transfer learning variant and uses a small labeled training set that has the same distribution as the task to find useful parts in the differently distributed training set. The authors of TrAdaboost [10] added a mechanism to weaken the impact of the noisy samples and outliers from the source data compared to the labeled target data.

Most transfer learning applications are used for web and text classification [2],[11],[19], [20],[30],[31]. However, there are also some transfer learning applications in the medical imaging field. For example, Wu et al [29] proposed an algorithm for medical image classification according to visual content, such as hand, skull and foot. Their method use multiple kernel learning (MKL) to combine different visual features, and learn the optimal mixing weights for each class adaptively. Van Opbroek et al [28] proposed an instance based transfer learning approach to solve the difference in distribution problem for MRI brain scans and performed experiments with four MRI sets, obtained with four different protocols. They showed that their transfer learning approach increase performance of the classification process and their method is also inspired by the TrAdaboost[10] framework. These approaches use a modified SVM classifier to transfer knowledge from one domain to another. Nguyen et al [21] presented a parameter based transfer learning method of training a cell detector on new data sets with minimal effort. First, they combine the classification rules extracted from existing data with the training samples of new data using transfer learning. Second, a global parameter is incorporated to refine the ranking of the classification rules. Their method achieves the same performance as previous approaches with only 10% of the training effort.

In this paper, we propose an automatic coronary calcium scoring method that uses instance based learning in homogeneous space. We are going to use a weighted nearest neighbor classifier to extract knowledge from the source domain, using some labeled data from the target domain, and use that knowledge for our task in the target domain.

Chapter 4

Theory

In this chapter, we describe our transfer learning approach that can be used for coronary calcium scoring in a multi-center screening trial. Our starting point is the system of Išgum et al [14]. As we described in the introduction, Išgum et al [14] proposed a system for automatic coronary calcium scoring in the low-dose chest CT. The system could be divided into four stages; candidate selection, feature computation, supervised classification and cardiovascular risk calculation. Potential calcifications (candidates) were extracted by thresholding (130 HU) and three dimensional component labeling [15]. The candidates included bony structures (e.g. ribs, sternum, spine), coronary calcifications, aortic calcifications, other type of calcifications (e.g. aortic valve calcifications, calcifications around the trachea) and noise. After candidate selection, a feature vector is computed for every candidate. The feature vector of each candidate contains spatial, texture and size features. The classification stage decides which feature vector is calcium or not. A supervised classification model is used which analyzes manual annotated candidates (feature vectors with a corresponding value that indicates if it is calcium or not) and produces a statistical function, called a *classifier*. This statistical function is used in the classification stage to predict whether a candidate in our task is calcium or not. After classification, the detected positive lesions were quantified by computing volume and Agatston scores. A schematic overview of the system is shown in figure 2.1.

For calcium scoring, Išgum et al [14] showed that supervised classification is a powerful mechanism to identify calcium in chest CT scans. However, such classification strategies also have some restrictions. The restriction of these classification techniques is that the training data and the task must be drawn from the same feature space and same distribution. In medical imaging, this rule does not always hold. If we want to classify scans, we also need to acquire a new training set with the same settings. Acquiring a complete new training set for every task with different settings is a time consuming process.

Even if the training set is not fully representative of the test data, there are likely certain parts of the data that can still be reused. Both domains, the source and target domain, contain common knowledge, namely the knowledge whether candidates are calcium or not. The major challenge is to identify the common knowledge between auxiliary or source tasks and apply such knowledge to new or target tasks. By identifying parts of the source domain data that we can reuse in the target domain, these methods typically find parts of a domain that are shared with the target domain. The field that tries to find the useful parts in the

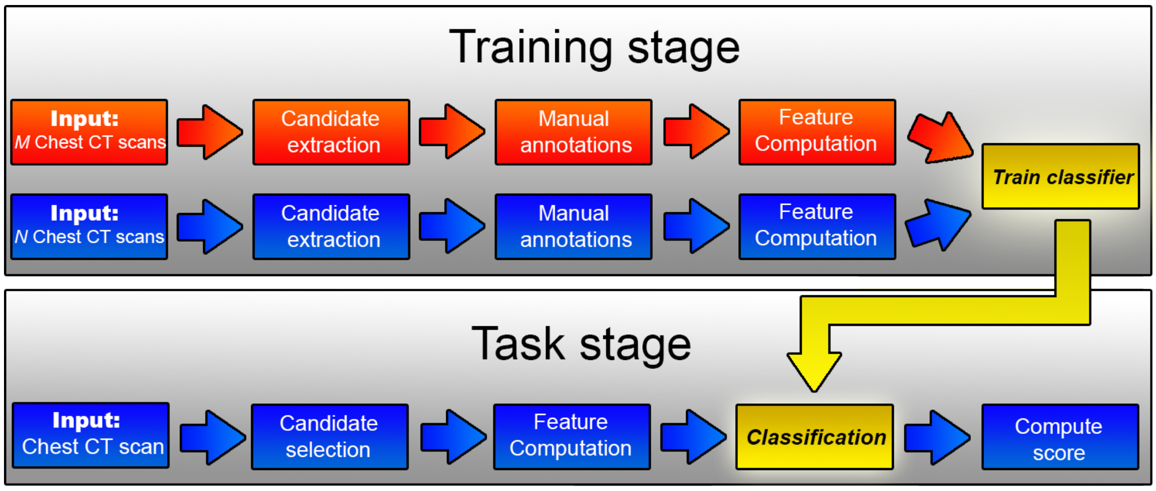

Figure 4.1: *A schematic overview that shows the training stage and the task stage of the transfer learning framework. Yellow shows the parts in the system that are modified compared to the system of Išgum et al [14] in figure 2.1.*

source domain is called instance based transfer learning. TrAdaboost [10] is a instance based transfer learning approach which uses a small amount of labeled task data to find useful parts in the source domain. In the experiments, TrAdaboost [10] used SVM with a linear kernel as basic learner in their framework. However, several experiments of Išgum et al[14] showed that the nearest neighbor classifier has a better performance for calcium scoring than the SVM classifier. Given this information, we decided to create a method that is based on the nearest neighbor classifier.

Our transfer learning approach, inspired by TrAdaboost [10], contains two stages; the training stage and task stage. We train a new classifier based on two labeled data sets and use that classifier for our task. Figure 4.1 shows a schematic overview of the modified system. We have made changes in the yellow blocks 'train classifier' and 'classification' compared to the original system of Išgum et al [14]. The orange blocks is the source domain which contains the data that we want to reuse as much as possible. The blue blocks is our target domain, the domain that we want to classify. In our case, we want to keep N, the number of scans in the target domain in the training stage, as low as possible.

4.0.2 Training classifier

According to figure 4.1 we have two labeled data sets to train the classifier. First, we have a complete labeled data set from the source domain, the domain where we want to learn form. Second, we have some labeled data from the target domain, the domain that we want to classify. Both domains are defined in the same feature space, because same candidate extraction and feature computation techniques were used for both domains. The train classifier is responsible for extracting common knowledge between the target and source domain. The train classifier stage could be divided in three steps:

- Feature selection.

- Extract knowledge from the source domain using the feature representation in the first step.
- Weighted nearest feature selection.

Feature selection

In the first step, we employ feature selection between the labeled target data and the source domain. Our goal is to determine an optimal feature space between the labeled target data and the source domain and eliminate features that have a bad impact on classifying the labeled target data. We use a floating forward feature selection (SFFS) [16] strategy with the nearest neighbor classifier to determine the optimal feature representation. In this step, a feature representation is determined that minimizes the prediction error on the labeled target data using the source domain. The result of this step is a feature representation that minimizes the prediction error on the labeled target data.

Extracting knowledge

After the optimal feature representation was determined for classifying the labeled target domain using the source domain, an iterative approach with a weighted nearest neighbor classifier is used to determine which samples from the source domain could be reused for classifying the target domain. In our approach, all labeled candidates in the training stage receives a *weight*. The weight indicate how a candidate fits in the target domain. The weight is a value between zero and one, one means that the candidate fits well and zero means that the candidate does not fit well in the target domain. Labeled candidates of the target domain always have a weight of one, because these candidates are from the same domain that we want to classify.

The weights are determined by a *weighted nearest neighbor* classifier which is based on the nearest neighbor principle. The most straightforward implementation of nearest neighbor algorithm can be described in 3 steps:

- Calculate all distances of all neighbors to the input sample using a distance metric.
- Sort all distances.
- Select k samples with the shortest distances to the sample that we want to classify (k is defined by the user).
- Calculate the probability of each class using the k samples that have the shortest distances. The class that has the largest probability is assigned to the input sample.

However, the nearest neighbor classifier only takes the distance (e.g. Eucledian distance, Manhattan distance) into account. We also would like to take the weight in consideration, because candidates in the source domain with a large weight (the useful samples) should have more impact than the candidates in the source domain with a low weight (the outliers). We need to weaken the impact of outliers and increase the impact of useful candidates. Therefore, the nearest neighbor strategy is modified into a weighted nearest neighbor strategy. The weighted nearest neighbor classifier is implemented as follows

- Calculate all distances of all neighbors to the input sample using a distance metric.
- Sort all distances.
- Select the k shortest distances (k is defined by the user).
- Get the weight of the k samples that have the shortest distance.
- Calculate the total weight for each class. The class that has the largest weight is assigned to the input sample.

To decide which candidates in the source domain are useful in the target domain, we use our weighted nearest neighbor in an iterative approach. A formal description of the iterative weighted nearest neighbor approach is shown in table 4.1 and an example of the algorithm is shown in figure 4.2. As it can be seen from the algorithm, if the probability of a candidate in the source domain is equal to the threshold of the current iteration (p_t), the weight of that candidate will be equal to that threshold. The threshold is determined by the k value, so this means that the number of iterations can never be larger then the k value. In the first iteration, we only validate on the labeled target domain candidates, because those candidates have a weight of one and candidates of the source domain have a weight of minus one. The weight of a source candidate will only change if it has the probability for the same class equal to the threshold. In the first iteration, the threshold will be one and source candidates will receive a weight of one when all neighbors in the labeled target domain candidates are from the same class as the source candidate. In the second iteration, we search for source domain candidates that has a lower threshold ($p_t = 1 - (t \times \frac{1}{k})$) for the same class than the last iteration. In this iteration, we validate the source domain candidates against the labeled target domain candidates plus the source domain candidates that received a weight in the previous iteration. After several iterations, the source domain candidates that fit in the labeled target domain candidates will have larger weights, while the source domain candidates that are dissimilar to the labeled target domain candidates will have lower weights.

Figure 4.2 shows an example how the algorithm works. The output of figure 4.2 shows that the red samples, source domain samples, have a weight between 0 and 1. However, some samples still have a weight of minus one. This happens when the probability of the source domain candidate in the target domain is zero. In this way, outliers are identified in the classification model. Candidates that contain a weight of minus one in the classification model are ignored during classification of the task.

Weighted nearest neighbor feature selection

After we obtained the classification model, we know which samples from the source domain are useful and outliers. In the first feature selection stage, we used all samples from the source domain. Because the impact of outliers are decreased in the classification model, other features might improve classification performance. Again a new feature selection is performed with the new classification model that that minimize the prediction error on the labeled target data using the hypothesis function $h_N(x_i)$, obtained from the 'training classifier' stage. Again, a floating forward feature selection (SFFS) [16] strategy using the prediction function $h_N(x_i)$ to determine the optimal feature representation.

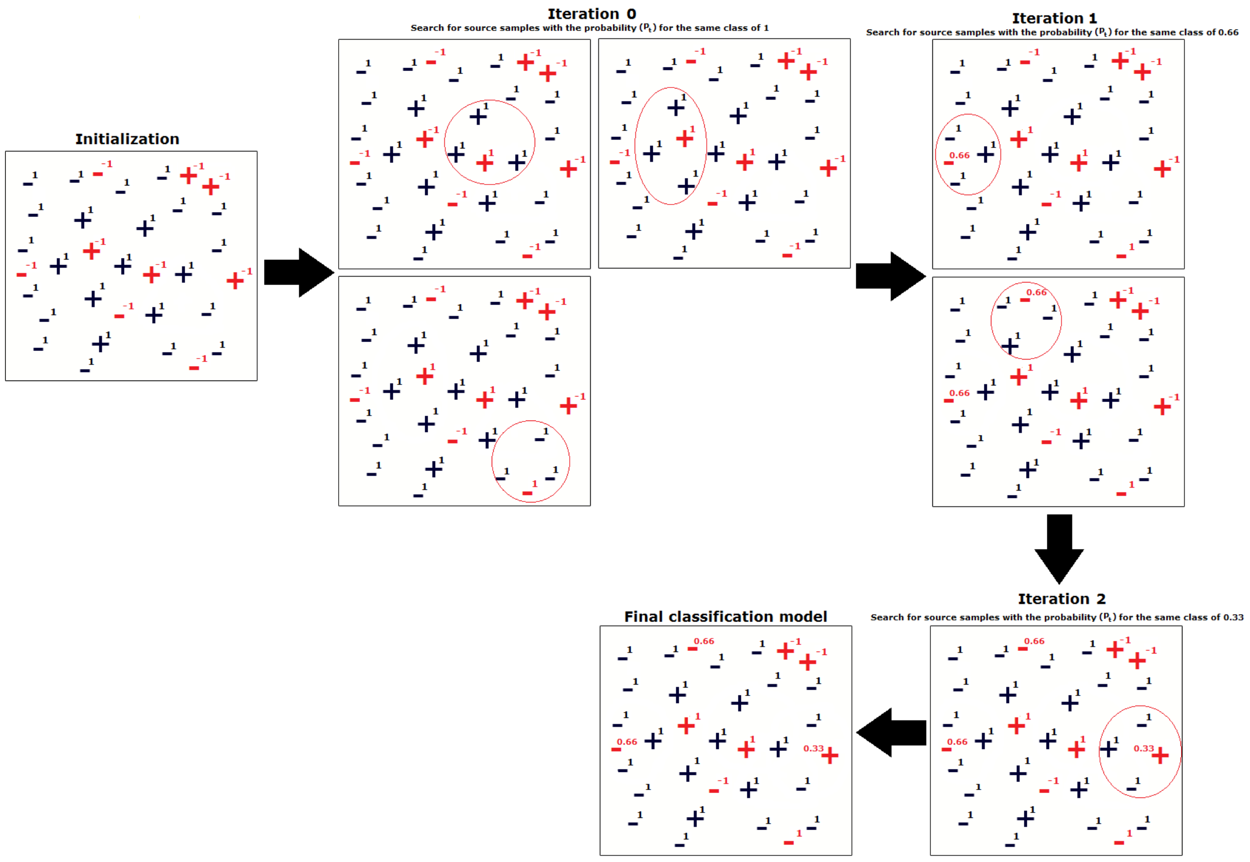
Initialization
Iteration 0
Search for source samples with the probability (P$_t$) for the same class of 1
Iteration 1
Search for source samples with the probability (P$_t$) for the same class of 0.66
Iteration 2
Search for source samples with the probability (P$_t$) for the same class of 0.33
Final classification model

Figure 4.2: *The process of how the classifier is trained. Black samples are samples from the target domain and red are samples from the source domain. In each iteration, we assign weights to candidates that has a certain probability for the same class. This probability is determined as follows:* $p_t = 1 - (t \times \frac{1}{k})$ *where t is the iteration number and k is value that is used for the nearest neighbor approach. In this example, the k value is three and we also use three iterations. The output is a classification model where each candidate contain a weight.*

Table 4.1: *Our proposed transfer learning algorithm explained in pseudo code*

```
Algorithm in pseudo code

Input the following parameters: $T_{different}$, $n$, $T_{same}$, $m$, $N$, $Classifier$ (Weighted Nearest Neighbor Classifier), $k$
Initialize:
  - The weight vector $w^1$ (weight vector at iteration 1) = ($w_1^1$ till $w_{m+n}^1$). 1 till $m$ is initialized with the value 1, $m$ till $m+n$ is initialized with the value -1.
  - Create training set $T_{merged} = T_{same} \cup T_{different}$ .

For $t = 0$ till $N$.
  1. Calculate the probability interval of iteration $t$: $p_t = 1 - (t \times \frac{1}{k})$
  2. Apply weight vector $w^t$ and training set $T_{merged}$ on the $Classifier$. The output is a hypothesis function at iteration $t$: $h_t(x_i)$
  3. For every sample $x_i$ in $T_{different}$.
    - Validate the sample with the hypothesis function $h_t(x_i)$.
    - When $h_t(x_i) > p_t$ is true, change the weight of sample $x_i$ to the output of the hypothesis function. $w_{m+i}^t = h_t(x_i)$
  4. When $w^t$ has changed in iteration $t$, go to step 2.

Output: A new hypothesis function at the $N$-th iteration: $h_N(x_i)$
```

4.0.3 Classification

We also decided to use a two stage classification approach, because Išgum et al[14] obtained the best results with this strategy. The goal of the first classification stage was to reduce the number of candidates by discarding candidates with a high probability for being in a negative class. In the second stage we our hypothesis function $h_N(x_i)$ to determine more difficult identifiable coronary calcifications. An example that shows how the weighted nearest neighbor classification works and the difference between the regular nearest neighbor approach is shown in figure 4.3.

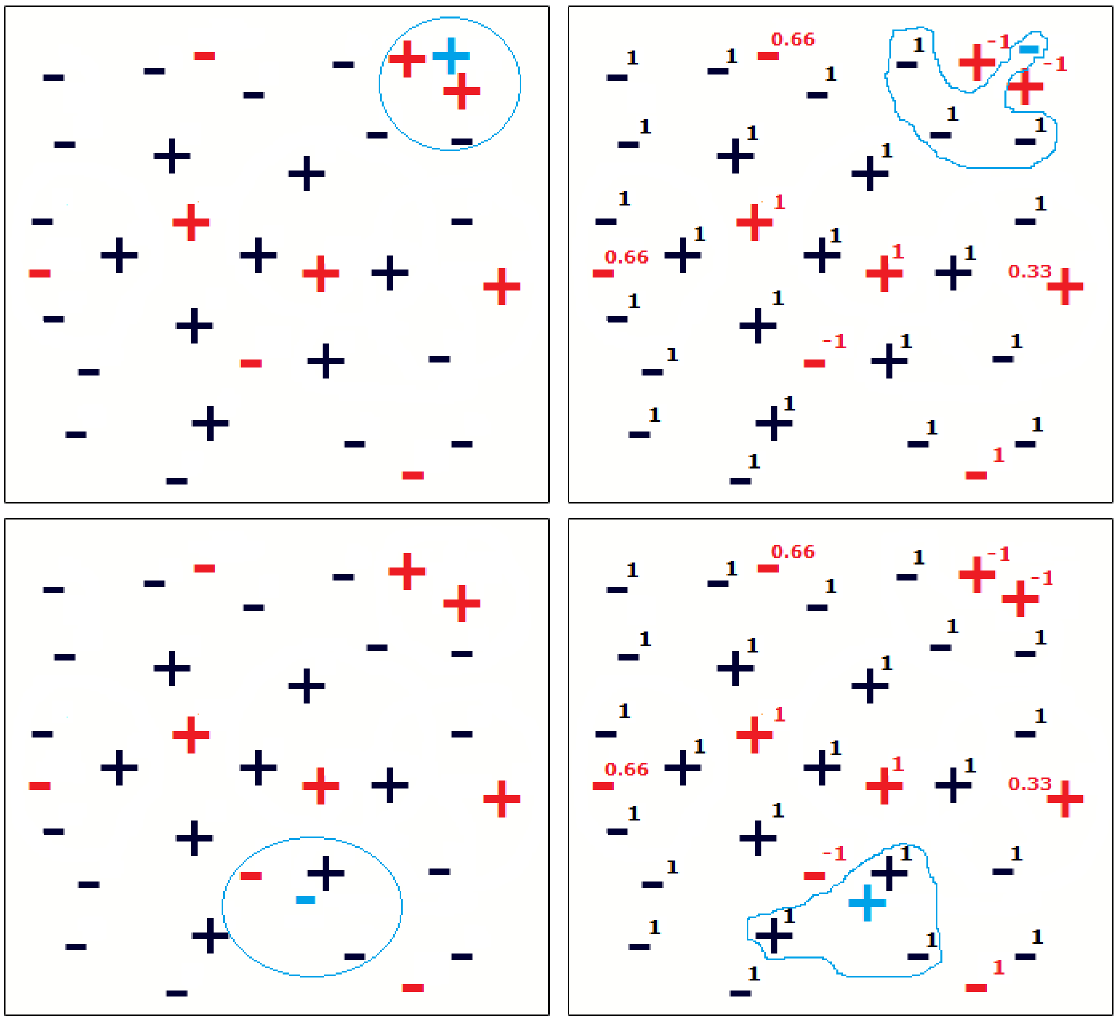

Figure 4.3: *A comparison between regular nearest neighbor algorithm (left) and weighted nearest neighbor algorithm (right). Regular nearest neighbor predicts the class of the red candidate by only using the shortest distances. Weighted nearest neighbor predicts the class of the red candidate using the shortest distances and the weight value. Candidates that contain a weight of -1 are ignored during the selection of the shortest distances. In this example, the k value is three.*

Chapter 5

Experimentation

5.0.4 Data

For our experiments we used chest CT scans from the NELSON trail. The NELSON trail is a heavy smokers screening trial that investigates whether low-dose 16-detector multi-slice chest CT (16 x 0.75 mm, 30 mAs, no IV contrast, no ECG synchronization) screening will decrease lung cancer mortality compared to no screening (www.nelsonproject.nl). Išgum et al [14] also used the NELSON trail for their experiments.

For our experiments we included 229 baseline scans from the Nelson trial. Hundred scans were acquired in UMC Utrecht (UMCU) using a Philips CT kernel B scanner. The other images were made in UMC Groningen (UMCG) using a Siemens CT kernel B30f scanner. All UMCU and a variable number of UMCG scans (10, 15, 20 and 25) were used for training, and the remaining 204 UMCG images were used for testing. All scans were reconstructed to 3.1 mm-sections with 1.4 mm increment. The reference standard was set manually by experts and was used for training and evaluation. A scan can example of both domains is shown in figure 5.1.

5.0.5 Blurring

Compared to the UMCU scans, we noticed that a lot more candidates were extracted from the UMCG scans in the candidate extraction stage. The bottom images of figure 5.1 show the extracted candidates of the top slices. In the 204 UMCG scans, an average of 30640 candidates were extracted per scan while the 100 UMCU training scans have an average of 5985 candidates per scan. We decided to apply a Gaussian blur function with a scale of 0.5 on all UMCG scans to reduce the number of candidates. After blurring, an average of 26334/30640 candidates were discarded per scan and 4.5% of the total calcium volume is discarded in the 204 UMCG scans. Figure 5.2 shows an example of an UMCG scan before and after blurring. After classification, results are translated back to the original image for calculating scores.

5.0.6 Experiments

For every experiment, calculation of the sensitivity is performed. The sensitivity for the volume and lost candidates is calculated as follows

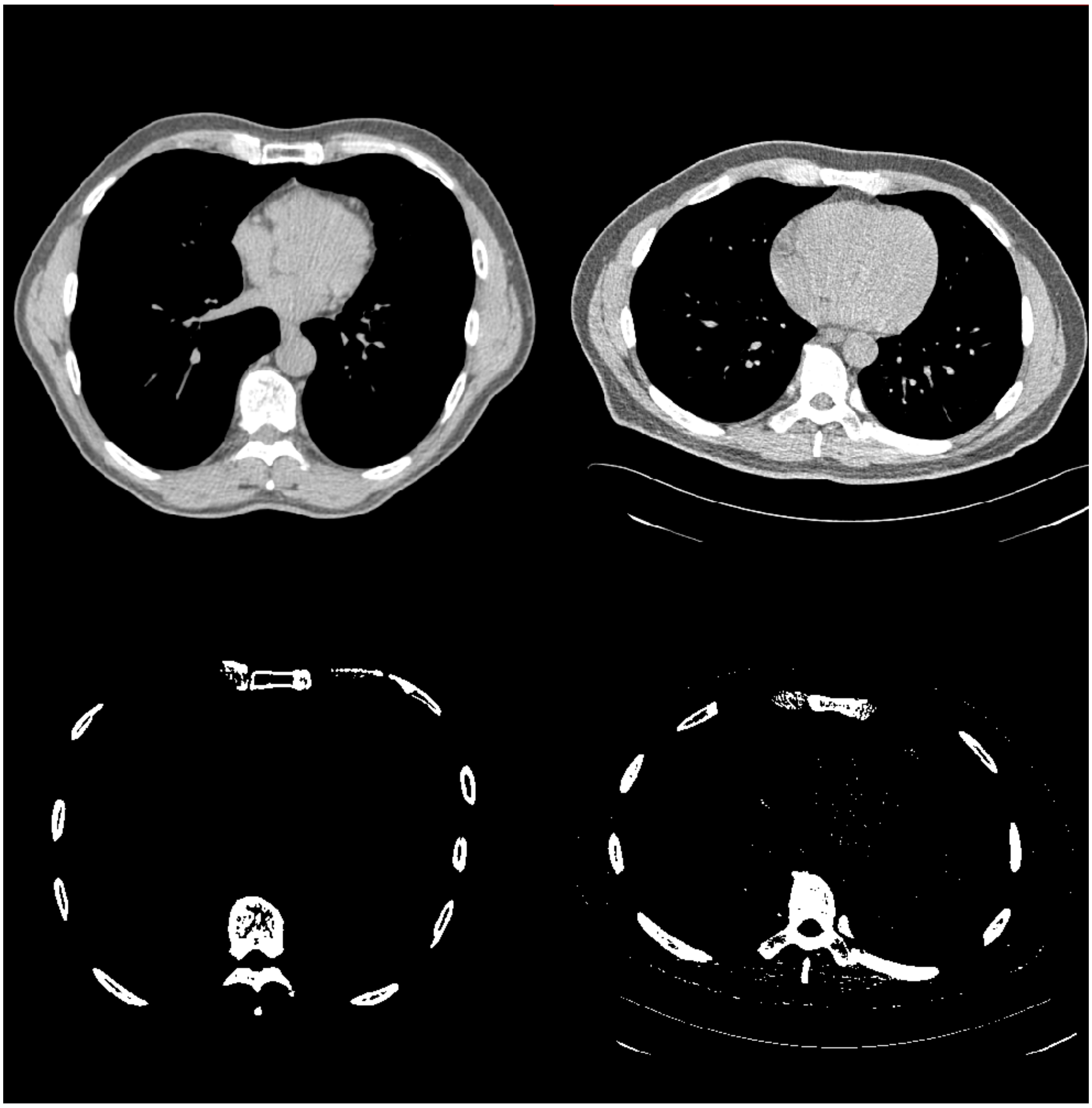

Figure 5.1: *Two slices from two different scans. Left scan is acquired from the UMCU and right scan is aquired from the UMCG.*

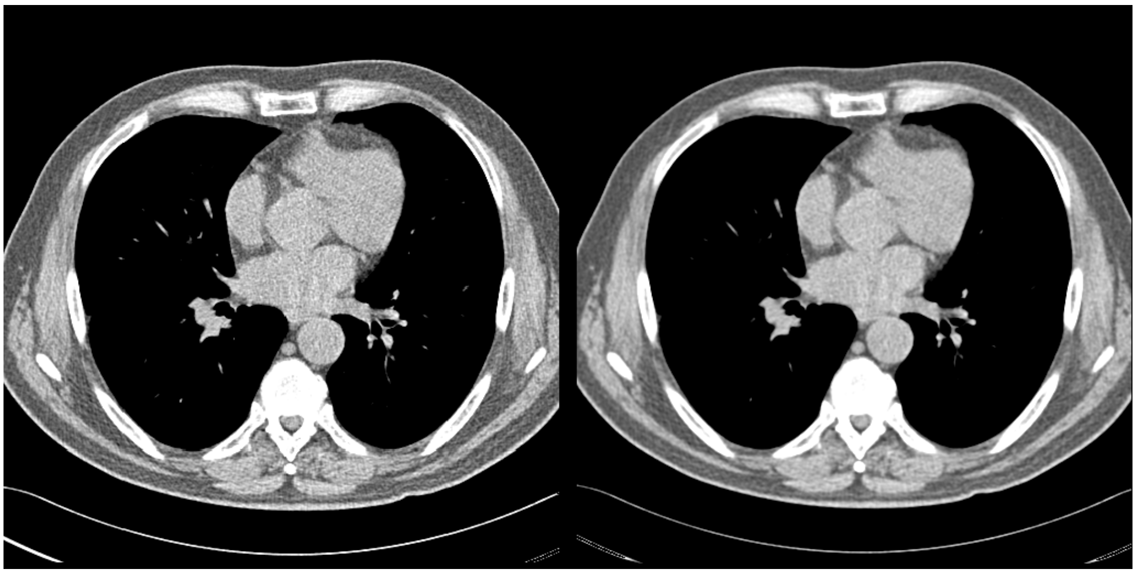

Figure 5.2: *A slice of an UMCG scan without blurring (left) and a slice of the same UMCG scan after blurring (right) with a scale of 0.5*

$$sensitivity\ for\ candidates = \frac{number\ of\ true\ postives}{number\ of\ postives}$$

$$sensitivity\ for\ volume = \frac{volume\ of\ true\ postives}{volume\ of\ all\ postives}$$

$$lost\ calcium\ stage\ n = \frac{number\ of\ false\ negatives\ stage\ n}{number\ of\ postives}$$

$$lost\ calcium\ volume\ stage\ n = \frac{false\ negatives\ volume\ stage\ n}{number\ of\ postives}$$

For the first experiment, we used the system of Išgum et al [14] using two stage classification where the first stage uses a 100 nearest neighbor classifier and the second stage a 11 nearest neighbor classifier. For training, we use 100 UMCU scans for and the 204 UMCG scans were used for validation. The goal of this experiment is to get more insight which stage goes wrong during classification in the system of Išgum et al [14]. We determined the ratio of the candidates and volume that is lost in every stage. The first row of table 5.1 shows the results per stage. Based on these results, we measured that the first stage discard 41% of the total calcium candidates for further analysis. In the second stage, 19% of the calcifications were classified as negative. Based on the result of the first experiment, we can conclude that the classification result of the UMCG data using the nearest neighbor classifier that is trained on UMCU data is worse. The main problem seems to be in the first stage of classification namely 41% of the calcified lesions were lost in that stage.

In the next experiments, we add some UMCG scans additional for training. We used different sizes (10, 15, 20 and 25 scans) and merged the candidates of the UMCG together with the candidates of the 100 UMCU scans and used them for training. The results can be also found in tables 5.1 and 5.2. It seems that the first stage does not improve and still discards a lot of calcium in the first stage. The results show that adding some additional UMCG scans for training does not improve classification performance. We also decided to

Table 5.1: *Setup with the results performed with the method described in [14]. This table shows the sensitivity for candidates per stage and the total sensitivity of the candidates.*

Training set setups	**Calcium lesions lost**		**Sensitivity**
	First stage	**Second stage**	
100 UMCU scans	0.41	0.19	0.40
100 UMCU and 10 UMCG scans	0.42	0.33	0.16
100 UMCU and 15 UMCG scans	0.43	0.20	0.37
100 UMCU and 20 UMCG scans	0.53	0.06	0.39
100 UMCU and 25 UMCG scans	0.39	0.21	0.38
10 UMCG scans	0.59	0.17	0.24
15 UMCG scans	0.55	0.13	0.31
20 UMCG scans	0.65	0.01	0.34
25 UMCG scans	0.48	0.19	0.33

Table 5.2: *Setup with the results performed with the method described in [14]. This table shows the sensitivity for volume per stage and the total sensitivity of the volume.*

Training set setups	**Calcium volume lost (mm3)**		**Sensitivity**
	First stage	**Second stage**	
100 UMCU scans	0.48	0.24	0.27
100 UMCU and 10 UMCG scans	0.46	0.37	0.25
100 UMCU and 15 UMCG scans	0.47	0.27	0.25
100 UMCU and 20 UMCG scans	0.47	0.09	0.44
100 UMCU and 25 UMCG scans	0.46	0.29	0.24
10 UMCG scans	0.52	0.28	0.19
15 UMCG scans	0.53	0.22	0.24
20 UMCG scans	0.61	0.01	0.37
25 UMCG scans	0.52	0.27	0.21

perform the same experiments where the small UMCG training scans are used for training. Again we used different training set sizes for training. However it seems that classification of the first stage still not improves. This behavior is expected, because the number of scans we used for training is relative low.

Based on the previous experiments, we can conclude that the first stage of two stage classification has a big impact on the final classification result, while the second stage classifies most calcium correctly. We decided to replace the first stage classification of Išgum et al [14] with another first stage classification strategy. Instead of merging the 100 UMCU scans and labeled UMCG scans together, candidates are validated against the source domain, UMCU scans, and the small labeled target domain, UMCG scans separately. If the candidate has a probability for the negative class more then 0.8 in the source domain and 0.96 for a negative class in the labeled data of the target domain, the candidate will be discarded for further analysis. Results of this experiment can be found in tables 5.3 and 5.4. Compared to the original system of Išgum et al [14], the results show that less calcium is discarded for further analysis in all cases. However, the sensitivity is still not optimal in terms of candidates and

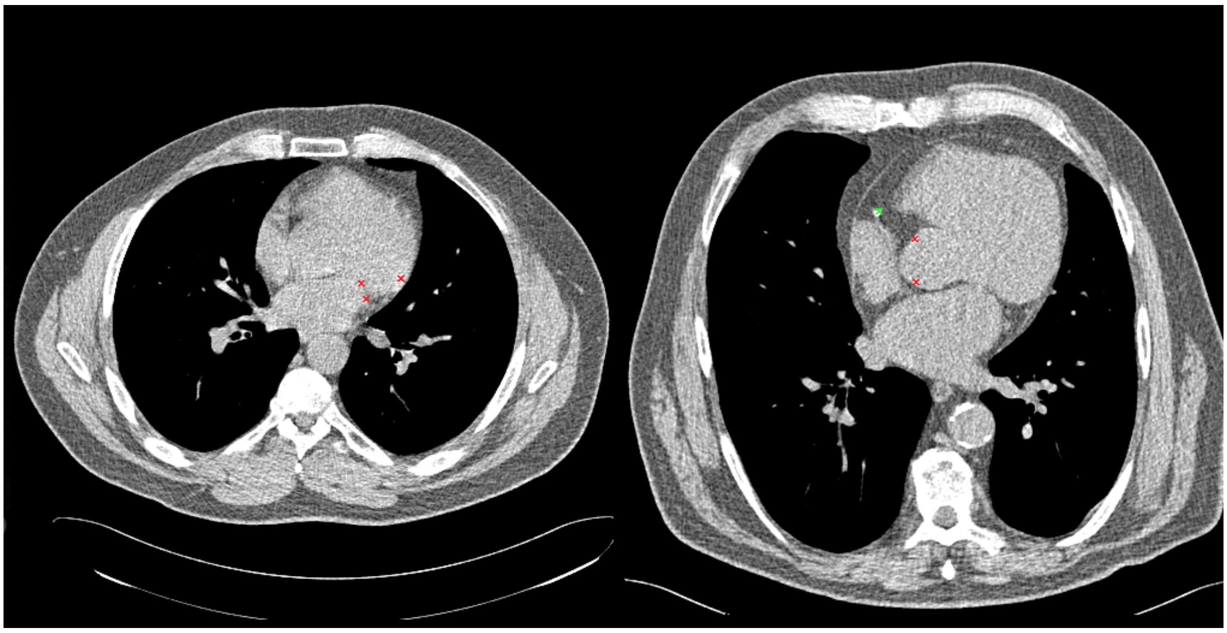

Figure 5.3: *Two slices from chest CT scans showing true positives (green crosses) and false positives (red crosses). False positives often represents noise (left) or small calcifications in the ascending aorta (right)*

Table 5.3: *Setup with the results performed with the method described in [14], where we changed the first classification stage. This table shows how much calcium candidates were lost per stage. The last column show the total sensitivity of the candidates.*

Training set setups	**Calcium lesions lost**		**Sensitivity**
	First stage	**Second stage**	
100 UMCU and 10 UMCG scans	0.14	0.57	0.29
100 UMCU and 15 UMCG scans	0.12	0.56	0.32
100 UMCU and 20 UMCG scans	0.11	0.43	0.46
100 UMCU and 25 UMCG scans	0.13	0.41	0.45

volume. These results show that the performance of the second stage has the biggest impact on the total result and needs to change to improve the classification performance for calcifications.

After several experiments were performed with the system of Išgum et al [14], we used the same first stage approach, separate validations against the source and target domain, and use our transfer learning approach for the second stage. In previous small experiments, a k value of eleven and five iterations gave the best performance, so we decided to use the same settings for all experiments. Results can be found in table 5.5 and 5.6. Comparing with the previous results, more calcium is predicted correctly which results in higher sensitivity. However, table 5.7 shows that the transfer learning method also produces on average more FP. We analysed FP in 20 of the 204 scans and it seems that FP occur in the region of the heart. FP often represent noise or calcifications in the ascending aorta. Figure 5.3 shows examples of FP that occur after classification in our system.

Table 5.4: *Setup with the results performed with the method described in [14], where we changed the first classification stage. This table shows how much calcification volume is lost per stage. The last column show the total sensitivity of the volume.*

Training set setups	**Calcium volume lost (mm^3)**		**Sensitivity**
	First stage	**Second stage**	
100 UMCU and 10 UMCG scans	0.11	0.71	0.18
100 UMCU and 15 UMCG scans	0.11	0.70	0.18
100 UMCU and 20 UMCG scans	0.04	0.68	0.27
100 UMCU and 25 UMCG scans	0.06	0.66	0.28

Table 5.5: *Setup with the results performed with our transfer learning system. This table shows the sensitivity for calcium candidates per stage and total sensitivity of the calcium candidates.*

Training set setups	**Calcium lesions lost**		**Sensitivity**
	First stage	**Second stage**	
100 UMCU and 10 UMCG scans	0.14	0.35	0.51
100 UMCU and 15 UMCG scans	0.12	0.29	0.59
100 UMCU and 20 UMCG scans	0.11	0.17	0.64
100 UMCU and 25 UMCG scans	0.13	0.23	0.54

Table 5.6: *Setup with the results performed with our transfer learning system. This table shows the sensitivity for volume per stage and the sensitivity of the volume.*

Training set setups	**Calcium volume lost (mm^3)**		**Sensitivity**
	First stage	**Second stage**	
100 UMCU and 10 UMCG scans	0.11	0.31	0.58
100 UMCU and 15 UMCG scans	0.11	0.29	0.59
100 UMCU and 20 UMCG scans	0.04	0.17	0.79
100 UMCU and 25 UMCG scans	0.06	0.22	0.72

Table 5.7: *An overview that shows the results on average between the method described in [14] and the transfer learning method using 20 scans from the target domain.*

	Method described in [14]	**Transfer learning**
Average calcium volume present (reference)	660 mm^3	660 mm^3
Average detected volume (TP)	180 mm^3	522 mm^3
Average undetected volume (FN)	480 mm^3	138 mm^3
Average non-calcium volume detected as calcium (FP)	19 mm^3	220 mm^3

Chapter 6

Conclusions and future work

We proposed an algorithm for transferring knowledge from one domain to another domain by using an iterative weighted nearest neighbor approach. The basic idea is to select source domain candidates that fits well in the labeled data from the target domain and train a classifier. We compared the results of our algorithm with the system of Išgum et al [14] using their optimal settings (two stage classification with the nearest neighbor classifier). The results show that our method improves classification in terms of identifying coronary calcium (sensitivity). However, we can also conclude that the results of this implementation does not give satisfactory results for application. The main drawback of this method is that it also produces significantly more false positives. Future work needs to be done to improve the method to reduce the number of false positives.

6.0.7 Future work

Like we described in the experiments section, the first stage of the classification has a bad influence on the final results. We also made a workaround for this problem by using the training sets separately instead of using a merged training set for the first stage. However, this does not solve the actual problem. The problem could be related to the fact that the UMCG domain has more negative candidates extracted per scan. It is possible that unremoved noise after blurring will disturb the first classification stage.

Our proposed algorithm for transferring knowledge from one domain to another domain uses two parameters, namely the k value and the number of iterations. For our experiments, we used a k value of eleven and five iterations because in previous experiments, these settings gave the best results. However, the k value of the iterative weighted nearest neighbor approach is an important parameter for the learning process and for classification. We used for all our experiments the same value for k but the k value should be adjusted to the size of the training set from the target domain. When our labeled data of the target domain is very small and we use a large k, transfer learning will not be optimal. Another approach is to use a small k for training and use a large k for feature selection after training the classifier and classification. More experiments with different k values needed to be performed to check the influence of the k value during training and classification.

Another parameter of the transfer learning approach is the number of iterations. If the amount of iterations is equal to the k value, all samples in the source domain receive a weight between one and zero. If the amount of iterations is lower than the k value, some source domain candidates that has a probability for the same class. For example, if the k value is ten and the amount of iterations is 5, only source domain candidates that contain a probability for the same class between one and 0.5 are used. Candidates with a probability for the same class lower than 0.5 are ignored and still have a weight of minus one. More experiments needed to be done to check whether it is more optimal to keep the number of iterations equal to the k (assign a weight to all candidates in the source domain) or keep the number of iterations lower than the k value (only assign a weight to candidates in the source domain that have a specific probability).

Another important aspect is to get more insight information about the kernels that were used for the UMCU and UMCG data. Based on figure 5.1, it seems that UMCG scan is sharper than the UMCU scan. Due the difference in sharpness, it could be possible that different settings should be used in the extraction stage or feature computation stage for the UMCG scans. For example, potential calcifications (candidates) were extracted by thresholding and three dimensional component labeling [15]. For the UMCU and UMCG scans, a threshold of 130 HU was used. However, the scans of UMCG seems to be sharper, so it might be possible to increase this threshold. Using a larger threshold, the number of potential candidates will reduce. Increasing the threshold may reduce the number of samples that represent noise and may finally lead to a more accurate classification model.

Bibliography

[1] Bram Bakker, Viktor Zhumatiy, Gabriel Gruener, and Jürgen Schmidhuber. Quasi-online reinforcement learning for robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2997–3002. IEEE, 2006.

[2] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics, 2006.

[3] Hans-Hermann Bock. Clustering methods: a history of k-means algorithms. In *Selected contributions in data analysis and classification*, pages 161–172. Springer, 2007.

[4] Gerd Brunner, Deepak R Chittajallu, Uday Kurkure, and Ioannis A Kakadiaris. Toward the automatic detection of coronary artery calcification in non-contrast computed tomography data. *The International Journal of Cardiovascular Imaging*, 26(7):829–838, 2010.

[5] Matthew J Budoff and Khawar M Gul. Expert review on coronary calcium. *Vascular health and risk management*, 4(2):315, 2008.

[6] Matthew J Budoff, Leslee J Shaw, Sandy T Liu, Steven R Weinstein, Tristen P Mosler, Philip H Tseng, Ferdinand R Flores, Tracy Q Callister, Paolo Raggi, and Daniel S Berman. Long-term prognosis associated with coronary calcificationobservations from a registry of 25,253 patients. *Journal of the American College of Cardiology*, 49(18):1860–1870, 2007.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[9] A Criminisi, J Shotton, and E Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*, 5(6):12, 2011.

[10] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[11] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[12] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[13] Ivana Išgum, Annemarieke Rutten, Mathias Prokop, and Bram van Ginneken. Detection of coronary calcifications from computed tomography scans for automated risk assessment of coronary artery disease. *Medical physics*, 34:1450, 2007.

[14] Ivana Išgum, Mathias Prokop, Meindert Niemeijer, M Viergever, and Bram van Ginneken. Automatic coronary calcium scoring in low-dose chest computed tomography. *Medical Imaging, IEEE Transactions on*, 31(10):2322–2334, 2012.

[15] Peter C Jacobs, Mathias Prokop, Yolanda van der Graaf, Martijn J Gondrie, Kristel J Janssen, Harry J de Koning, Ivana Isgum, Rob J van Klaveren, Matthijs Oudkerk, Bram van Ginneken, et al. Comparing coronary artery calcium and thoracic aorta calcium for prediction of all-cause mortality and cardiovascular events on low-dose non-gated computed tomography in a high-risk population of heavy smokers. *Atherosclerosis*, 209(2):455–462, 2010.

[16] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 1997.

[17] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[18] Uday Kurkure, Deepak R Chittajallu, Gerd Brunner, Yen H Le, and Ioannis A Kakadiaris. A supervised classification-based method for coronary calcium detection in non-contrast ct. *The International Journal of Cardiovascular Imaging*, 26(7):817–828, 2010.

[19] Xiao Ling, Gui-Rong Xue, Wenyuan Dai, Yun Jiang, Qiang Yang, and Yong Yu. Can chinese web pages be classified with english data source? In *Proceedings of the 17th international conference on World Wide Web*, pages 969–978. ACM, 2008.

[20] Lilyana Mihalkova, Tuyen Huynh, and Raymond J Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pages 608–614, 2007.

[21] Nhat H. Nguyen, Eric Norris, Mark G. Clemens, and Min C. Shin. Rapidly adaptive cell detection using transfer learning with a global parameter. In *Proceedings of the Second international conference on Machine learning in medical imaging*, MLMI'11, pages 209–216, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[23] Weike Pan, Erheng Zhong, and Qiang Yang. Transfer learning for text mining. In *Mining Text Data*, pages 223–257. Springer, 2012.

[24] Stefan C Saur, Hatem Alkadhi, Lotus Desbiolles, Gábor Székely, and Philippe C Cattin. Automatic detection of calcified coronary plaques in computed tomography data sets. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008*, pages 170–177. Springer, 2008.

[25] Leslee J Shaw, Paolo Raggi, Tracy Q Callister, and Daniel S Berman. Prognostic value of coronary artery calcium screening in asymptomatic smokers and non-smokers. *European heart journal*, 27(8):968–975, 2006.

[26] Leslee J Shaw, Paolo Raggi, Enrique Schisterman, Daniel S Berman, and Tracy Q Callister. Prognostic value of cardiac risk factors and coronary artery calcium screening for all-cause mortality1. *Radiology*, 228(3):826–833, 2003.

[27] Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, 2005.

[28] Annegreet van Opbroek, M Arfan Ikram, Meike W Vernooij, and Marleen de Bruijne. Supervised image segmentation across scanner protocols: A transfer learning approach. In *Machine Learning in Medical Imaging*, pages 160–167. Springer, 2012.

[29] Hong Wu, Hao Zhang, and Chao Li. Medical image classification with multiple kernel learning. In *Proceedings of the Second International Conference on Internet Multimedia Computing and Service*, pages 189–192. ACM, 2010.

[30] Tianbao Yang, Rong Jin, Anil K Jain, Yang Zhou, and Wei Tong. Unsupervised transfer classification: application to text categorization. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1159–1168. ACM, 2010.

[31] Dequan Zheng, Chenghe Zhang, Geli Fei, and Tiejun Zhao. Research on text categorization based on a weakly-supervised transfer learning method. In *Computational Linguistics and Intelligent Text Processing*, pages 144–156. Springer, 2012.