University of Bonn

Master's thesis for obtaining the academic degree
„Master of Science (M.Sc.)“

Visualizing the effects of domain shift on CNN based image segmentation

Author:
Albert Gubaidullin

First Examiner:
Prof. Thomas Schultz

Second Examiner:
Dr. Nils Goerke

Advisor:
Rasha Sheikh

Submitted: March 14, 2022

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Abstract

The motivation of this thesis is to propose a technique to detect a domain shift and create an application that utilizes it in image segmentation task. This application allows to investigate the domain shift problem in pretrained U-Net models. The problem of generalization of segmentation models to unseen data can be difficult because there can be huge differences in image characteristics for different scanners and different medical centers. If a model is not trained properly and a domain shift exists it will not be possible to have a reliable deployment of this model in real-world scenarios. To gain a better understanding of the problem, we developed a tool that can help to detect domain shift problem and identify, at which point model input processing generalization is lost. In this work, we are trying to apply dimensionality reduction techniques to make better visualizations of inner representations of pretrained U-Net models. The final goal of this work is to create a tool that can help to explore the domain shift problem in a given U-Net model.

Contents

List of Figures

List of Tables

1 Introduction

The problem of generalization is a significant challenge in machine learning. Making a trained model perform well on unseen data is not a trivial task on its own, and it gets more difficult if the data belongs to several domains (Quionero-Candela et al. (2009),Torralba and Efros (2011)) When a model, trained on one distribution, is given some data belonging to another distribution, an effect called domain shift might occur and decrease the inference quality. Even small changes in the statistics that would be almost imperceptible to the human eye, can make a well-trained model fail the task it was doing good on the original data. (Kurakin et al. (2017)) In the case of medical imaging, domain shift challenges are typically ascribed to color, brightness, and contrast variations, caused by stain variations and scanner properties (Inoue and Yagi (2020). This problem is common since data in many cases come from different medical centers and scanner vendors. It leads to a major concern of real-world application of Convolutional Neural Networks (CNN) since there are no guarantees that the trained model will generalize to all possible data.

A standard practice of supervised domain adaptation (sDA) is to take some data from the target domain, label it and use it to fine-tune a model that was already pre-trained on the source domain. The main goal of domain adaptation research is to understand how a model should be fine-tuned.

According to some studies, (Shirokikh et al. (2020)) using the transfer learning approach of fine-tuning only the last layers, based on the assumption that low-level features are independent of domain-specific parameters, while high-level ones are more prone to domain shift and should be therefore fine-tuned. However, in some Domain Adaptation articles, experiments show that domain shift is present on the first layers too (Dou et al. (2018),Zhao et al. (2021)).

Thus, we can say that the domain shift problem is probably linked to specific model parameters and optimization scheme. Over-parameterized and high complexity models can become biased to small details of the training set. Developing models that are protected from domain shift problem, is crucial for the real-world application of CNNs. It is important to be able to detect that domain shift exists and track the layer, where it is presented the most.

In this thesis project, we develop a web-based application *DomainVis* that can

help to explore trained CNN-based segmentation models. We propose an approach to investigate the source of domain shift problem in a given model with layer-by-layer visualization and dimensionality reduction techniques. Also, we provide activation and filter visualizations to analyze the complexity of learned features by a CNN. The application provides a set of tools, that can help to understand why the model fails on new data and where it might need a change.

We conduct a series of experiments on the Calgari-Campinas-359 brain imaging dataset and a U-net segmentation model trained on a subset belonging to one domain only.

Chapters content

In Chapter 2 we describe the problem of domain shift, our proposed idea, and algorithms used in the application.

In Chapter 3 we introduce related articles regarding the work that is already done in the field of domain shift detection and CNN Visualisation.

In Chapter 4 we explain modules of the *DomainVis* application, their implementation, and usage. The first section is related to the Reductor module, which performs dimensionality reduction on the outputs of CNN models. In the second section, we describe the Feature map presenter module. The third section is dedicated to the application interface In section four the application usage and instructions are provided.

Chapter 5 is consists of experiments that were done using the proposed approach and the application.

We conclude the thesis in Chapter 6 where we summarize the evaluation results and point out the drawbacks and possible improvements.

2 Background

This chapter reviews the problem of Domain shift, as well as the fundamental background behind the algorithms used in this thesis project.

2.1 Distribution shift in medical imaging

In this section, we review the problem of distribution shift and the sub-problem of Domain shift in particular.

2.1.1 Distribution shift

When training a model, it is assumed that both the training and real-world data distributions are static, meaning that they do not change between the time the model is trained and the time it is deployed. If this statement is false wrong and the distributions change in any way, we must account for it, or at least the probability of it happening. To propose a model like this, one needs first to look into the reasons for the shift. Though there are an unlimited number of possible causes for these shifts, there are many ways to categorize many distinct types of shifts into qualitatively different categories.

A model can be affected by a variety of different distribution shifts: When only the distribution of source data changes and everything else remains the same, it is known as a simple covariate shift. When only the distribution over target data changes and everything else remains the same, this is known as a prior probability shift. When the distributions diverge as a result of an unknown sample rejection procedure, this is known as sample selection bias. Imbalanced data is a type of intentionally shifted dataset used for computational or modeling purposes. Changes in measuring are part of domain shift. Changes in the strength of contributing components are referred to as source component shift.

2.1.2 Domain shift

According to the study of Pooley (2005), manufacturer differences are a major cause of the statistical shift, as different MRI scanners produce images of different

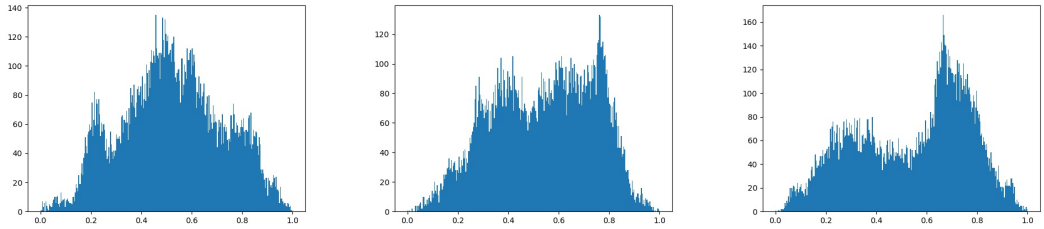


Figure 2.1: Average intensity distributions of normalized 3T MRI slices from scanners of Calgary-Campinas-359 dataset. From left to right: Philips, Siemens, General Electric

characteristics related to manufacturer-specific MRI physics. Thus using multi-center/scanner source data may cause a Domain shift problem to appear.

In Figure 2.1, we can see normalized intensity distributions of normalized 3T MRI slices from different scanners of the Calgary-Campinas-359 dataset (Souza et al. (2018)). There, we can observe a common difficulty with data from many sources - a shift in data distribution caused, in this example, by MRI machine hardware and software settings. When a deep learning model that was trained with images from one domain is used on images from another domain, the performance drops dramatically. The fact that the measurement system, or method of description, can vary is what defines domain shift.

To postulate this, let L be some underlying unchanging latent representation of the covariate space X. We denote a latent variable in this space by x_0. The target variable y is dependent on x_0. The problem is that we never observe x_0. We only observe some map $x = f(x_0)$ that can change between training and test scenarios. Modeling for domain shift involves estimating the map between representations using distributional information.

2.2 Dimentionality reduction

In this section, the idea behind the use of dimensionality reduction is explained. Also, we review the dimensionality reduction algorithms used in the application.

2.2.1 Dimentionality reduction for visualisation

Layers of a CNN model represent input image by high-dimensional feature maps. Such representation is unstructured which makes it hard to explore and visualize. When solving a domain shift problem, it is useful to get a visual representation of data. However, many visualization techniques have limitations in dealing with

large-scale high-dimensional data. The drawback of using dimensionality reduction techniques for visualization is information loss due to reducing the dimensionality of data. In fact, the information loss depends on the used dimensionality reduction techniques and selected parameters. Therefore, it is important to assess dimensionality reduction algorithms to choose the most suitable one with proper parameters.

In this thesis project, we are trying to apply different dimensionality reduction approaches to visualize the difference in activation maps of a U-Net segmentation model given inputs from different domains and thus visualize the domain shift.

2.2.2 Dimentionality reduction approaches

We are using 4 dimentionality reduction algorithms:

- Principal Component Analysis
- ISOMAP
- Locally-Linear Embedding
- T-distributed Stochastic Neighbor Embedding

PCA

Principal components analysis is a well-known technique for determining an optimal orthogonal transformation matrix R such that the points are linearly uncorrelated after transformation. This transformation is learned in a way that the first principal component captures as much variability in the dataset as possible, and each subsequent principal component catches as much leftover variance as possible while remaining orthogonal to the preceding components. Formally, we want to learn R given X such that $X = XR$ yields a more compact data representation.

ISOMAP

PCA has proven to be effective in a variety of applications, but it suffers from focusing solely on maintaining pairwise Euclidean distances and ignores the distribution of nearby data points. PCA algorithm might consider two data points as neighbors if the high-dimensional data is on or near a curved manifold, even if their distance across the manifold is substantially greater than the standard interpoint distance. ISOMAP (Tenenbaum et al. (2000)) is a technique that attempts to retain pairwise geodesic distances between data points in order to overcome this challenge. The geodesic distance is a distance between two places calculated over the manifold. The geodesic distances between data points $x_i (i = 1, 2, ..., n)$ in

ISOMAP are calculated by forming a neighborhood graph G in which each data point x_i is connected to its k nearest neighbors $x_{ij}(j = 1, 2, ..., k)$ in the dataset X. The shortest path between two points in a graph is an estimate of their geodesic distance, and it can be easily determined using Dijkstra's or Floyd's shortest-path algorithms. The pairwise geodesic distance matrix is formed by computing the geodesic distances between all data points in X. By applying classical scaling to the resulting pairwise geodesic distance matrix, the low-dimensional representations y_i of the data points x_i in the low-dimensional space Y are computed.

Local Linear Embedding

A similar approach to ISOMAP is Local Linear Embedding (LLE) (Roweis and Saul (2000)). In this technique, the data points are represented by a graph. In contrast to ISOMAP, it solely tries to preserve the local attributes of the data. As a result, LLE is less vulnerable to short-circuiting than ISOMAP, because short-circuiting only impacts a few local characteristics. Because local properties are preserved, non-convex manifolds can also be successfully embedded. In LLE, the data manifold's local attributes are constructed by expressing high-dimensional data points as a linear combination of their nearest neighbors. In a low-dimensional representation of the data, LLE seeks to keep the reconstruction weights in linear combinations as much as possible.

T-distributed stochastic neighbor embedding

T-distributed stochastic neighbor embedding (T-SNE) (Maaten and Hinton (2008)) is an improved variation of the stochastic neighbor embedding (SNE) (Hinton and Roweis (2002)). To preserve neighborhood identity, t-SNE aims to place a point from a high-dimensional space in a low-dimensional one. The T-SNE technique translates Euclidean distances between high-dimensional data points into conditional probabilities that represent similarities; closer data points indicate a higher degree of similarity. The conditional probability $p_j|i$ represents the similarity of data point x_j to data point x_i. These similarities represent the likelihood that x_i will choose x_j as a neighbor. A similar conditional probability is computed for the low-dimensional equivalents y_i and y_j of the high-dimensional data points x_i and x_j, indicated by $q_j|i$.

The goal of the algorithm is to reduce the mismatch between the high- and low-dimensional representations using conditional probability distributions for the data points. This is done by minimizing the cost function (Equation (2)), using gradient descent.

The cost function is a sum of Kullback–Leibler (KL) divergences over all points. Here P_i represents the conditional probability distribution over all data points

given a data point x_i, and Q_i represents the conditional probability distribution over all other map points given map point y_i.

3 Related Work

This chapter reviews previous related works as well as concurrent to this thesis works.

3.1 Domain shift detection

The problems of domain shift detection have seen considerable progress in computer vision and medical image analysis in recent years. Most of the current solutions presume access to the datasets during training, like in Yu and Aizawa (2019) or validation steps Liang et al. (2018) that do not provide a general solution to domain shift detection.

The authors of Yosinski et al. (2014) analyzed features transferability for the task of image classification. According to their study, only the last levels of the network should be retrained, and the early layers of the network have good transferability.

The authors of (Aljundi and Tuytelaars (2016)) presented a convolutional filter reconstruction to address the problem of domain shift on low-level layers in an unsupervised domain adaptation scenario, demonstrating that the first layers are more prone to domain shift than the later layers.

Surveys of modern techniques for anomaly detection by Wang et al. (2020) and Braei and Wagner (2020) are also can be helpful since their methods might also be used to detect domain shift.

3.2 Domain adaptation

Several studies (Lee et al. (2018), Goyal et al. (2020)) propose utilizing adversarial samples generated via perturbation of the training samples to improve the robustness of their network, but these methods result in longer training times and sub-optimal results.

In the field of medical imaging, domain adaption algorithms solve the domain shift problem differently.

Earlier methods used the same methodology as Shirokikh et al. (2020): fine-tuning the network's later layers while leaving the earliest layers alone. Fine-tuning

of only the last CNN layer is performed in Kushibar et al. (2019), which, according to their findings, enhances the network's accuracy over transferring without adaptation. The authors, on the other hand, make no comparisons to alternative domain adaptation or transferring methodologies.

Several publications (Ghafoorian et al. (2017), Valverde et al. (2019)) compare the results of fine-tuning various layers. Notably, fine-tuning of the entire network is included in the comparison in Ghafoorian et al. (2017), with superior results revealed for a lower number of fine-tuned later layers.

Some later approaches are motivated by (Aljundi and Tuytelaars (2016)), suggesting that low-level structures display domain shift in medical pictures like MRI, whereas high-level structures are similar, hence the initial layers should be targeted.

The work of Shirokikh et al. (2020) compare three methods: fine-tuning the first layers, fine-tuning the last layers, and fine-tuning the entire network. They concluded that fine-tuning the first layers produces better results than the other two methods.

3.3 CNN Feature and Filter Visualisation

One of the challenges of neural networks is figuring out what exactly is going on at each layer of the model. It is known, that each layer extracts higher and higher-level features of the image until the final layer essentially decides what the image represents. The first layer can search for edges or corners. Intermediate layers look for shapes, such as a square or a triangle, by interpreting the basic features. Neurons from the final layers respond to complex items such as faces or driving plates.

Understanding the representation learned by CNN models has been the subject of several studies. Tommasi et al. (2016) presented a visualization technique that exposes the input stimuli that activate specific feature maps. Girshick et al. (2014) used visualization to show which patches in a dataset are primarily responsible for high activations at higher layers of the model. By projecting back from the fully connected layers of the network,Simonyan et al. (2014) shows how saliency maps can be produced from CNN.

The work of Stacke et al. (2019) not only includes the attempts to find domain shift in Mini-GoogLeNet inner representations, they also proposed a metric to measure the domain shift, based on Wasserstein distance.

3.4 DR for CNN Visualization

There is a considerable amount of articles that were trying to apply dimensionality reduction techniques on intermediate layers of CNN models. Cao et al. (2018) tried to apply Multilinear Principal Component Analysis (MPCA) to reduce the dimensionality of later layers of CNN to extract final features before the classification and clusterise it.

In Pan et al. (2008), authors propose a Maximum Mean Discrepancy Embedding reduction technique to learn a low-dimensional latent space common to two given domains.

Hoyt and Owen (2021) has done extensive work on applying T-Sne on layers of VGG-15 dataset trained on CIFAR-10 data.

4 Application

This chapter consists of four parts. The first part is dedicated to the *Reductor* module of the application. In the second one, we describe the Feature map explorer module. The third part has information about the interface of the application and available visualization tools. The last part consists of general information on the installation of the application as well as connecting other databases and CNN-based architectures.

4.1 Reductor

In this section, we will describe the *Reductor* module. Given the outputs of one of the layers, this module is applying chosen dimensionality reduction technique to visualize differences between source and target domains.

The initial dimensionality of the input for this module is:

$$N \times S \times M \times \ P1 \times P2 \tag{4.1}$$

Where N is a number of domains, S is a number of samples for each domain, M is the number of feature maps, $P1$ and $P2$ are dimensionalities of each feature map.

Using this module we can compare layer by layer outputs of a model for source and target domains. It reveals the differences between domains showing separate data clusters for each domain.

4.1.1 Reduction modes

There are different ways to shape the data for a reduction, in this work we are concentrated on two options: reduction of feature maps and reduction of pixels.

Feature Map reduction

Feature Map reduction is the first and default mode of operation for dimensionality reduction in our application. In this mode, we reduce the dimensionality of feature

maps from the layer outputs to 2.

$$N \times S \times M \times \ P1 \times P2 \rightarrow N \times S \times \ P1 \times P2 \times 2 \tag{4.2}$$

Reducing N-dimensional feature map space to 2 dimensions allows us to visualize this data on a 2D plot. By investigating resulting clusters and use of additional tools, we might be able to answer some questions, like :

- Does network segment data on this layer?
- What data does the network see and find important?
- Is there is a domain shift presented on this layer?

We process the data for this reduction mode by first rearranging the initial data with dimensions of Formula 4.2 to a 2D array, that has $N \times Stimes\ P1 \times P2$ rows and M columns by simply stacking it. Then this data is being processed by one of the algorithms. If labels for the data are also given, additional steps take place. First, we reshape the data back to its original form, but with $M := 2$. With this step, the information about pixel coordinates and the source domain and sample is restored. Then we calculate the *Evaluation Map* to be able to link the resulting data to a true label from the dataset. More on this is explained in Section 4.3.

Pixel reduction

The second type of dimensionality reduction used in this project is pixel reduction. In this operation mode, we are trying to capture the differences between feature maps by reducing the dimensionality of coordinate space.

We take each activation map and reduce it to a 2D coordinate.

$$N \times S \times M \times \ P1 \times P2 \rightarrow N \times S \times M \times 2 \tag{4.3}$$

By this reduction, we are trying to find similarities between feature maps for each domain.

Pixel reduction can help to understand the relationship between the complexity of the data and the complexity of the network. If, for example, a number of points, representing several feature maps can be assigned to a cluster, perhaps they can be replaced just by one map. Also, it can help to find the existence of a domain shift on a layer, if corresponding data points for target and source domain have a shift on a final plot.

4.1.2 Additional tools

There are certain challenges while processing high dimensional weakly unstructured data such as CNN feature maps. The Reduction module has a number of additional parameters that can be utilized.

Subsampling

dimensionality reduction techniques can take a considerate amount of time to perform. To solve this problem, the application allows data subsampling before passing it to the *Reductor* module. The subsampling is performed by the Nearest Neighbor Scaling algorithm.

It's a straightforward image interpolation method that takes the value of the nearest input image pixel and assigns it to the scaled image pixel. The nearest neighbor algorithm chooses the value of the nearest known pixel without taking into account the values of other pixels nearby, resulting in a piecewise constant function (Jenkins et al. (1985)). Such algorithm was chosen in attempt to save exact data values and not to blur images. This is important in case binary images like masks are being used.

Background remove

The application also provides the functionality of removing a background from the original image, this can help to reduce the amount of data passed to the *Reductor*. This also can increase the precision of dimensionality reduction, since no background data is involved in the reduction. In case masks are given, this tool will simply remove background data. If masks are not available, this tool can cut the background according to the predicted image.

4.1.3 Visualisation

In the application, the *Reductor* module is represented by a *Reductor* page (Fig 4.1). This page consists of a settings sidebar, two plots, and a table. Plots on this page are: dimensionality reduction plot on the left and a Model input plot on the right.

Settings sidebar

Setting sidebar provides important visualisation settings,including:

- Algorithm

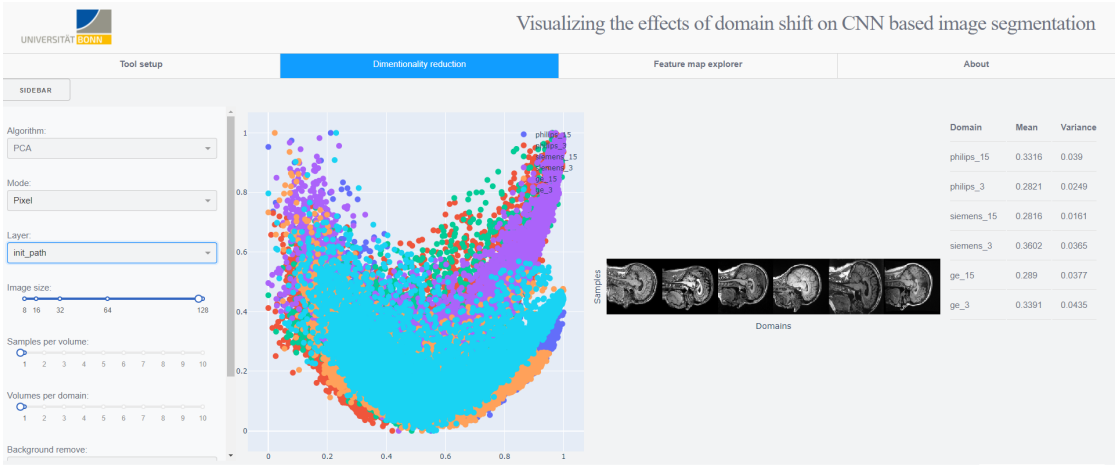


Figure 4.1: *Reductor* page of the application

- Layer
- Image size
- Volume per domain
- Sample per volume
- Mask cut

The algorithm drop menu allows the user to choose one of the implemented algorithms from chapter 2.

The layer drop menu provides a dynamically created list of all layers of the given model.

The image size slider regulates the size of the image passed to the dimensionality reduction algorithm. This slider corresponds to the Subsampling function explained above.

Volume per domain and Sample per volume sliders define the amount of data being passed to the dimensionality reduction algorithm. Under volume in this context, we assume each separate file chosen on Tools setting page. Sample here is a 2D slice from each volume.

Background remove drop menu correspond to the background remove function explained above. There are three options to this tool.

- None
- Prediction
- True mask

The first option will turn off the tool completely. The second option will remove the data, according to provided labels, if they were given. A third option removes the data, based on the output of the network.

Some settings appear only if a certain algorithm is chosen. If the user choosing *T-SNE*, the following settings appear:

- Number of iterations slider
- Perplexity slider

These options are responsible simply for a number of iterations of *T-SNE* algorithms and the perplexity value respectively.

If the user choosing *ISOMAP* or *LLE* algorithms, the slider for choosing a number of nearest neighbors will appar

The settings sidebar can be removed by clicking on the *sidebar* button on top of it.

Dimensionality Reduction Plot

The left plot is a 2D representation of one of the model's layer outputs.

Depending on the number of domains in the provided database and current settings, outputs from multiple input slices can be plotted at the same time. This allows us to see the correlations between multiple slices within one domain and multiple domains at the same time.

A number of tools are presented on the top of the plot, including zoom in and out, pan, autoscale, download, lasso, and box tool

The most important of above are *lasso* and *box* tools. They allow the user to select multiple points on the plot and it will highlight the corresponding areas on the Model input plot. This tool is able to provide a background on selected clusters of data.

Model input plot

Model input plot consists of multiple input images that were given to a model. Amount of images based on samples, volumes, and domains chosen. Rows on this plot correspond to samples and volumes, columns correspond to domains. As in another plot, a number of tools are presented on the top of this one, including zoom in and out, pan, autoscale, download, lasso, and box tool.

The main function, this plot provides is an ability to highlight areas, corresponding to chosen by lasso or box tools data points. It is also possible to click on the areas of this plot to highlight corresponding data points on the dimensionality reduction plot.

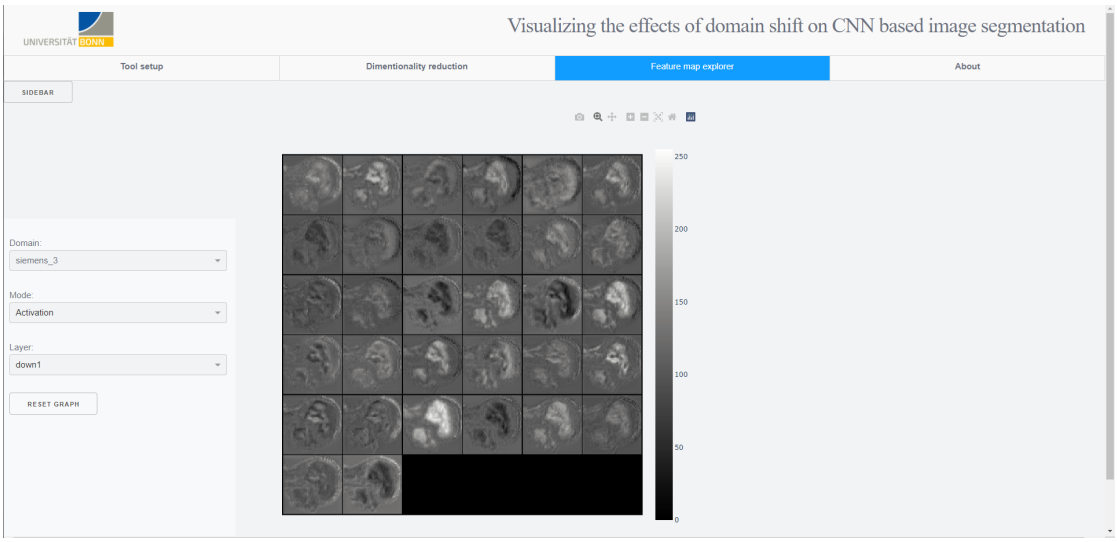


Figure 4.2: Feature map explorer page of the application

The table

The table on this page simply shows mean and variance values for the domains, presented on Dimensionality Reduction Plot. This statistical values are helpful, when searching for a good parameters of dimentionality reduction algorithm.

4.2 Feature Map Explorer

In this section, we will describe the Feature map explorer module. This module can visualize feature maps of a network and provides some additional tools for feature map exploration.

4.2.1 Feature Map Explorer modes

There are two modes of the Feature map explorer mode: filter and feature map visualization.

4.2.2 Visualisation

The Explorer module is represented by the Feature map explorer page (Fig 4.2) a. On this page, the user can visualize learned filters and activations outputs for one of the input images. The page consists of a settings sidebar and a plot.

Settings sidebar

Setting sidebar provides important visualisation settings,including:

- Domain
- Mode
- Layer

The domain drop menu allows choosing one of the domains existing in the provided database. Mode drop menu consists of two option, filter, and activation that defines, what is going to be presented on the plot. The layer drop menu provides a dynamically created list of all layers of the given network. Outputs of the chosen layer will be visualized on the 2D plot.

A number of tools are presented on the top of the plot, including zoom in and out, pan, autoscale, and download.

4.3 Additional pages

In this section, we will describe the interface of the additional pages of the application.

The application consists of four pages. Two pages for each of the modules and two for the application initialization setup and user manual.

4.3.1 Application initialisation

First page of the application is a initialisation page. Here user can choose a model and a database to work with (Fig 4.3). The initialisation page consists of three input fields:

- Model selection
- Database selection
- Label selection

Each input field has a button underneath that opens a file upload interface.
More information on field requirements can be found in subsection 4.4.6

4.3.2 User Manual

The user manual is the last page of the application. It has information on how to operate the application.

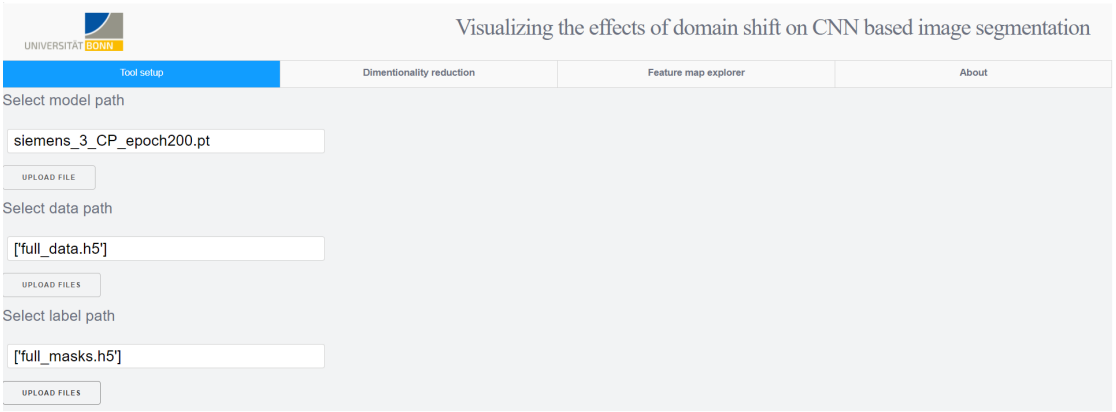


Figure 4.3: Feature map explorer page of the application

4.4 Installation and usage

In this section, we will describe the installation and workflow of the application

4.4.1 Libraries used

Database processing in the application is mostly done using Nibabel, Numpy, and Scikit-image libraries. Nibabel is used as a main reader for the raw medical data. Numpy and scikit-image libraries provide an image manipulation toolset. Database storage and further access are implemented using the h5py library and allow easy and fast access to the preprocessed data.

dimensionality reduction techniques are mostly implemented using the scikit-learn library. T-SNE algorithm requires an openTSNE library.

CNN models that the application is using have to be implemented using the PyTorch library.

To make an interface for this thesis project we decided to create a web application using the Dash framework.

4.4.2 Model requirements

For the application to work properly, a model is required to be implemented using a PyTorch Framework.

If any *Sequential* layers are used, they will be treated as a single layer. The outputs of the last layer in the stack will be treated as output.

4.4.3 Dataset requirements

Since domain list is created dynamically, datasets also have to follow strict rules. They are the same for the main dataset and the label if any provided. The file format has to follow the rule: *domain name_id.**. The *id* value is not used anywhere in the program, but the *domain name* value is used in a domain list creation. There can be several files presented for each domain, they will be treated as separate volumes by the program. There is also an option of direct choice of $h5$ files. They will not have any preprocessing steps and considered already ready to use.

This design was chosen due to the usual structure of medical datasets, where multiple 3D volumes are given.

4.4.4 Dataset processing

Before direct usage, all files are preprocessed and compiled in one or two, if labels are provided, $h5$ files, and then stored on the server. Only a number of slices from the center of each volume is taken.

In case of $h5$ file chosen on previous step, the will be uploaded to the server and all the slices will be taken.

4.4.5 Application configuration

Some of the applications setting are located directly in program files in a config file. This file includes such settings as:

- Upload directory
- Model directory
- Data directory
- Mask directory
- Processed directory
- Slices

The first five options are responsible for various files directories explained in previous sections. The Slices variable defines the number of slices that are being taken from each MRI scan.

4.4.6 Application workflow

When a user enters the web application, the *Tool Setup* page appears. There a model and a database have to be chosen at this point, or further work with the application is not possible.

After choosing model, database files, and possibly label files in the user system, they are uploaded to the server

Then the work with the application can start. The user can choose any of the other pages. Each page when visited will try to visualize the default values. Changing any of the values on setting toolbars will refresh r=the page and change the plot results.

5 Experiments

In order to investigate the quantification of domain shift, the cross-dataset generalization for brain segmentation was evaluated by training the U-Net models to segment brain on an MRI slice. By evaluating how well models generalize to new target domains we quantify the domain shift in each scenario. Then we quantify the domain shift in the inner representations of the the U-Net model using algorithms described in chapter 4 and searching where the domain shift is presented the most. Then, we investigate the correlation between new representations and the model accuracy. Finally, we detect layers with the most domain shift presence and determine best algorithms and their parameters to detect it.

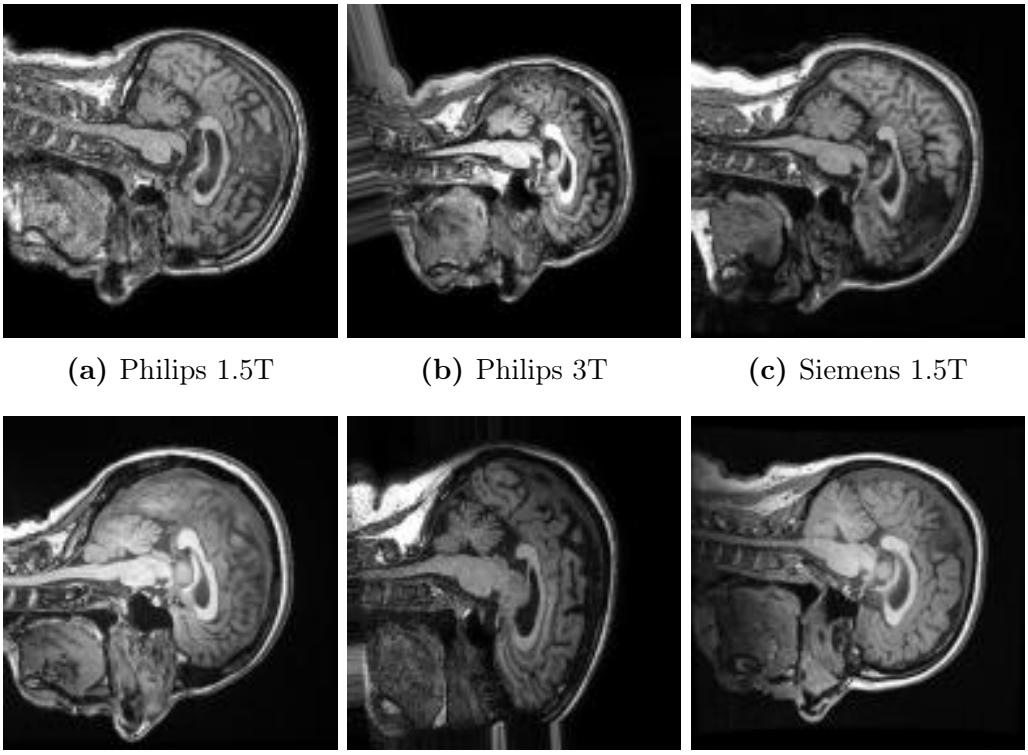
In this section, the datasets as well as the experimental setup are described

5.1 Dataset

A publicly available training subset of the Calgary-Campinas (CC-359) dataset was used for training and all experiments. The CC-359 dataset (Souza et al. (2018)) is an open, multi-vendor, multi-field strength magnetic resonance (MR) T1-weighted volumetric brain imaging dataset. It consists of photos of healthy persons 29–80 years old taken using 1.5 T and 3 T scanners from three vendors (Siemens, Philips, and General Electric). CC-359 has a total of 359 MRI scans, 60 per vendor, and magnetic field strengths. It also generates consensus brain extraction masks for all supervised classification volumes.

To distribute data equally across all vendors, 55 scans were used for training and 4 were used for testing of the model. Both 1.5T and 3T scans were used for each vendor. Initially, each MRI scan of the CC-359 dataset has dimensionality of $[150 \times 288 \times 288]$. To process this data we transformed all the scans to the equal voxel resolution of $[1 \times 1 \times 1]$ mm using interpolation of the slices.

Only 40 middle slices were taken to keep only the center of the 3D scan. The resulting images were normalized to 0-1 range and then, their size were reduced to $[128 \times 128]$.



(a) Philips 1.5T **(b)** Philips 3T **(c)** Siemens 1.5T

(d) Siemens 3T **(e)** General Electric 1.5T **(f)** General Electric 3T

Figure 5.1: Examples of MRI slices from CC-359 Dataset for each domain

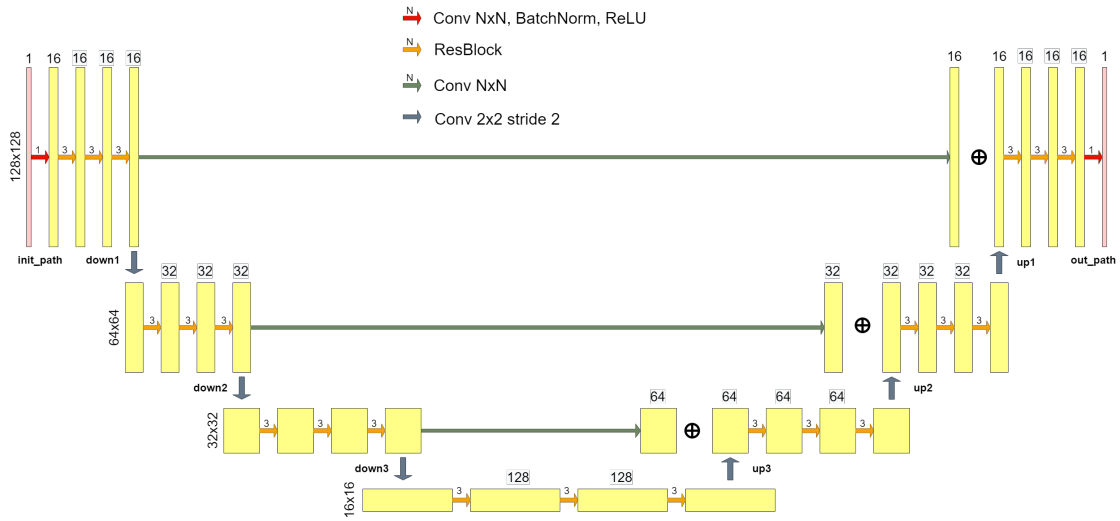


Figure 5.2: U-Net structure

5.2 Model structure

The model used for this dataset is similar to Shirokikh et al. (2020). It is an original 2D U-Net model (Ronneberger et al. (2015)) with a minor changes. Instead of convolutions we used a series of residual blocks (Ronneberger et al. (2015)). We also apply convolutional layer with $[1 \times 1 \times 1]$ kernel to the skip-connections. We change the channel-wise concatenation at the end of the skip-connections to the channel-wise summation – it reduces memory consumption and preserves the number of channels in the following residual blocks. Exact details of the model can be fount on Figure 5.2

5.3 Model training

Data propagation

On this data set we evaluated cross-domain transferability of the model. To do so, we trained a model for each domain separately, then test each one of them on the remaining domains. Three-fold cross validation was used to determine a model's test score on the source domain.

The parameters used during training are presented in Table 5.1

We used a the surface Dice score to measure segmentation quality. Results of this experiment can be seen in Table 5.2

Parameter	Value
Optimiser	Stochastic Gradient Descent
Number of epochs	200
Batch size	32
Learning rate	0.001
Training images	1980
Validation images	220

Table 5.1: Training parameters for the U-net models

	Ph 1.5T	Ph 3T	Sm 1.5T	Sm 3T	GE 1.5T	GE 3T
Ph 1.5T	0.9447	0.9031	0.9384	0.7953	0.9303	0.8286
Ph 3T	0.3661	0.8409	0.3246	0.2596	0.516	0.2173
Sm 1.5T	0.7480	0.8636	0.9067	0.6421	0.8164	0.6866
Sm 3T	0.8231	0.9704	0.9274	0.9556	0.7674	0.9187
GE 1.5T	0.7931	0.8781	0.8701	0.4560	0.9572	0.6126
GE 3T	0.7188	0.9320	0.9162	0.9183	0.8268	0.9429
MSE	0.0770	0.0051	0.0570	**0.1432**	0.0447	0.1192

Table 5.2: Cross-domain model accuracy comparison. Column names are the source domains which the model is trained on, the row name is the target domains which the model is tested on. Sm, GE, Ph correspond to vendors, i.e. Siemens, GE and Philips. Results are given in surface Dice Score. The last row is designated to Mean Squared Errors (MSE)

5.4 Concept proof experiment

In the first experiment the model with the most domain shift problem was chosen. To determine it, we calculated Mean Squared Error for each model across all domains from the baseline dataset:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2 \tag{5.1}$$

Here X_i accuracy of the model on dataset i, $\hat{X}$ is accuracy for baseline dataset, n is number of datasets. Under baseline dataset we consider an oeiginal dataset, the model was trained on.

Results of this calculation can be seen on Table 5.2 in the MSE row. The biggest value belongs to the Siemens 3T model.

Domain shift quantification for the inner representations of U-Net is done in two steps. First, we process the activations outputs of the model through PCA, LLE and ISOMAP dimentionality reduction algorithms to reduce the dimentionality of feature maps to two dimensions.

Then we are calculating the Sliced Wasserstein (SW) distance proposed in Bonneel et al. (2014), between source and domain distributions across all layers and algorithms.

Sliced Wasserstein distance The SW metric is a computationally cheaper alternative to Wasserstein distance. It is defined as an expectation over random projections, and approximated by Monte Carlo. In original form, the Wasserstein distance between the distributions is given by:

$$W(p_s, p_t) = \inf_{\pi \in \Gamma(p_s, p_t)} \int_{RxR} |(x-y)| d\pi(x,y) \tag{5.2}$$

Where by p_s we denote the distribution of source domain dataset $X_s = x_{s1}, ..., x_{sm}$ after the dimentionality reduction. A second dataset, $X_t = x_{t1}, ..., x_{tm}$ similarly generates p_t. As $\pi(p_s, p_t)$ we denote the joined distributions with margins p_s and p_t.

This metric, commonly known as the earth mover's distance, calculates the low cost way to translate one distribution to another, taking into consideration both the amount and the distance involved. The distance is calculated based on the absolute values of the generated distributions, and measures both the shape mismatch and the distance between two distributions.

	Ph 1.5T	Ph 3T	Sm 1.5T	Sm 3T	GE 1.5T	GE 3T
init_layer	0.12(0.83)	0.09(0.62)	0.12(0.84)	0(0)	0.15(1)	0.1(0.65)
down1	0.09(1)	0.05(0.55)	0.04(0.44)	0(0)	0.05(0.61)	0.03(0.39)
down2	0.06(0.54)	0.06(0.56)	0.07(0.69)	0(0)	0.08(0.76)	0.11(1)
down3	0.06(0.59)	0.1(1)	0.07(0.71)	0(0)	0.07(0.71)	0.07(0.74)
up3	0.04(0.45)	0.06(0.69)	0.09(1)	0(0)	0.07(0.79)	0.03(0.28)
up2	0.04(0.53)	0.03(0.35)	0.08(0.96)	0(0)	0.04(0.45)	0.08(1)
up1	0.06(0.53)	0.11(1)	0.05(0.41)	0(0)	0.05(0.48)	0.04(0.38)

Table 5.3: Cross domain distances for every inner U-Net layer produced by PCA algorithm

5.4.1 Evaluation

On Tables 5.35.5 we can see the resulting distance measurements for PCA, ISOMAP and LLE algorithms respectively. In the brackets the normalised values across domain are provided. The T-SNE algorithm was not used during these evaluations due to it's inability to preserve the distances between data samples. Examples of this algorithm performance can be found in additional materials of Experiment 2.

To measure the similarity between resulting vectors of distances and model accuracy values for each algorithm, we use the cosine similarity metric.

$$CS = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.3}$$

where A_i and B_i are components of vector A and B respectively.

Results of this metric are presented on table 5.6. In the brackets, there are values, calculated for normalised distances.

In each cell we can see a measurement of how good the algorithm performs on a model layer.

5.4.2 Experiment discussion

As seen in the table, there is a strong correlation, between Wasserstein distance and model performance. It could be explained by its inherent quality of comparing both the form and position of the distributions.

	Ph 1.5T	Ph 3T	Sm 1.5T	Sm 3T	GE 1.5T	GE 3T
init	0.03(0.32)	0.04(0.44)	0.1(1)	0(0)	0.06(0.56)	0.04(0.39)
down1	0.08(0.7)	0.07(0.61)	0.09(0.75)	0(0)	0.08(0.67)	0.12(1)
down2	0.03(0.24)	0.15(1)	0.06(0.42)	0(0)	0.1(0.71)	0.12(0.81)
down3	0.05(0.36)	0.15(1)	0.06(0.4)	0(0)	0.09(0.59)	0.05(0.32)
up3	0.06(0.66)	0.09(1)	0.04(0.41)	0(0)	0.05(0.59)	0.04(0.39)
up2	0.05(0.6)	0.06(0.7)	0.05(0.55)	0(0)	0.08(1)	0.06(0.7)
up1	0.02(0.19)	0.12(1)	0.01(0.11)	0(0)	0.04(0.34)	0.01(0.1)

Table 5.4: Cross domain distances for every inner U-Net layer produced by ISOMAP algorithm

	Ph 1.5T	Ph 3T	Sm 1.5T	Sm 3T	GE 1.5T	GE 3T
init_layer	0.04(0.52)	0.02(0.24)	0.02(0.23)	0(0)	0.07(1)	0.06(0.83)
down1	0.12(0.83)	0.14(0.98)	0.15(1)	0(0)	0.1(0.67)	0.12(0.79)
down2	0.07(0.93)	0.08(1)	0.08(0.96)	0(0)	0.07(0.86)	0.06(0.73)
down3	0.03(0.58)	0.05(1)	0.04(0.81)	0(0)	0.04(0.78)	0.05(0.87)
up3	0.02(0.18)	0.09(1)	0.03(0.3)	0(0)	0.04(0.44)	0.03(0.34)
up2	0.03(1)	0.02(0.58)	0.01(0.31)	0(0)	0.02(0.56)	0.01(0.21)
up1	0.04(0.78)	0.01(0.24)	0.01(0.2)	0(0)	0.05(1)	0.03(0.61)

Table 5.5: Cross domain distances for every inner U-Net layer produced by LLE algorithm

	init_layer	down1	down2	down3	up3	up2	up1
ISOMAP	0.63(0.46)	0.59(0.13)	0.83(0.85)	0.93(0.99)	0.97(0.96)	0.73(0.67)	0.86(0.92)
LLE	0.5(0.36)	0.71(0.69)	0.73(0.81)	0.76(0.84)	0.51(0.4)	0.64(0.57)	0.95(0.97)
PCA	0.66(0.45)	0.62(0.41)	0.6(0.26)	0.79(0.93)	0.88(0.93)	0.45(0.18)	0.75(0.73)

Table 5.6: Cosine distance between vectors for each layer of the model calculated fo ISOMAP, LLE and PCA

There are some fluctuations in the similarities based on the layer and algorithm used, but the main trend is visible: Domain shift changes the data propagation among all the layers, but in the later ones, it correlates more with the model accuracy.

In this setting both LLE and ISOMAP had a number of nearest neighbours $n := 5$. Changing this parameter slightly changes the performance of the algorithms, while keeping trend the same.

5.5 Experiment via Application

The *DomainVis* application provides a good visualisations, that allows to see the domain shift without complicated metrics. The experiment is conducted on the same data, which allows to compare the performance of the algorithms.

First step of the experiment is to upload the data, labels, and model into the DomainVis tool. Then on the second page we can choose an algorithm and layer to see the 2d representation of the layer outputs. For the the first example the LLE representation of init_path layer was chosen, due to it's poor performance during first experiment.

Trying different neighbours parameters, the best representation was found, based on mean and variance table as well as the plot visualisation. On Figure 5.3we can see the DomainVis tool window with these parameters. The domain distributions are too dense to investigate them at the same time, so we can investigate them pairwise, based on the Siemens 3T baseline domain. Using lasso tool on the first plot, we can select an area of interest on the representation plot to see the source of the points highlighted on the input image plot. On Figure 5.4 we can see pairs of representation with area selected and highlighted input slices. As we can see, the lasso tool allows a clear separation between domains within certain areas. This proves the existence of the domain shift on this layer.

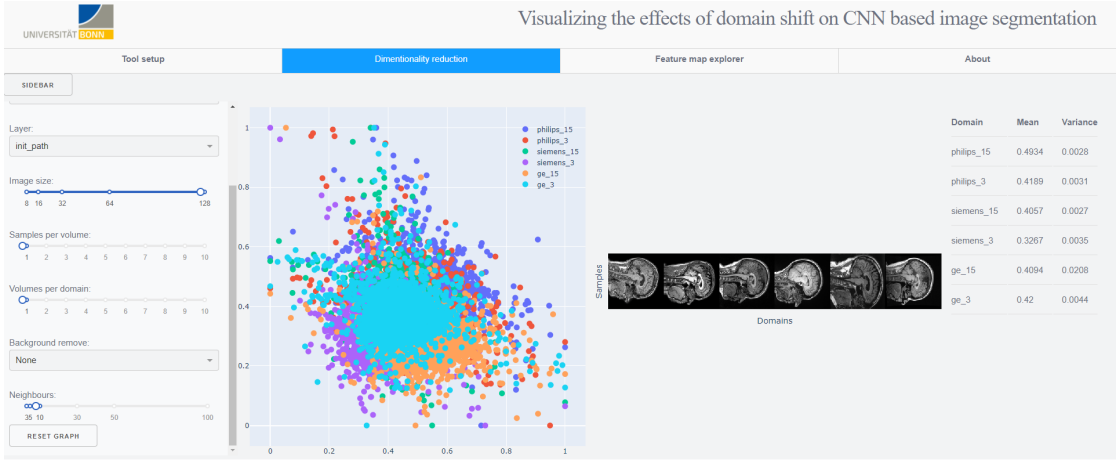
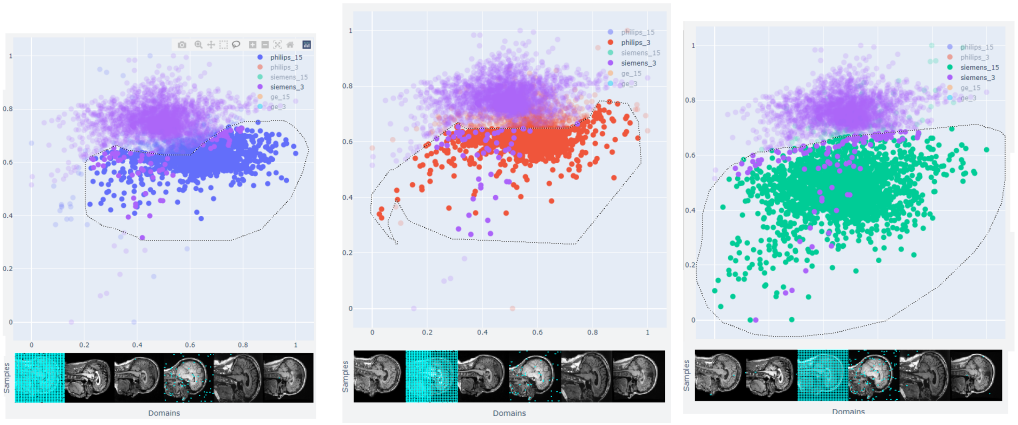
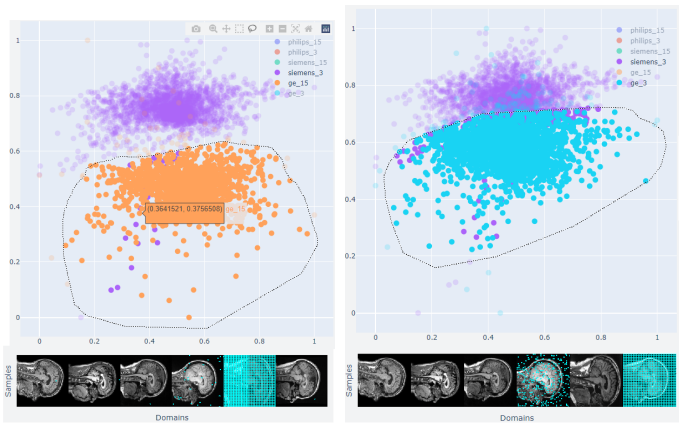


Figure 5.3: The DomainVis representation of init_path layer after LLE algorithm processing



(a) Siemens 3T vs Philips 1.5T **(b)** Siemens 3T vs Philips 3T **(c)** Siemens 3T vs Siemens 1.5T



(d) Siemens 3T vs General Electric 1.5T **(e)** Siemens 3T vs General Electric 1.5T

Figure 5.4: Pairwise application of LLE on itit_path layer of the U-Net. Top part is the 2d representation of Feature map space and area, chosen with lasso tool.Bottom part is input slice for each domain with highlighted pixels that represent chosen area on top image.

6 Conclusion

Achievement

In this work, we proposed a technique for domain shift detection in image segmentation problem. We proved that this is a reliable solution for a domain shift detection problem. Based on the conducted experiments, the technique is accurate.

We developed a DomainVis tool that utilises this technique and allows to investigate pretrained U-Net models. The application is optimised and provides a good visualisation tools. Subsampling option allows faster calculations when, there is a lot of input data.

Further improvement

The developed tool demonstrates efficient and reliable behavior and can be further improved with a perspective for practical applications.

DomainVis can be extended in multiple ways. The Reductor module can have more dimentionality algorithms to work with, including more complicated ones, like Autoencoders, trained for the task of domain shift detection. The Presenter module lacks some functionality, like filter exploration tools. The additional modules can be developed: we can utilise Gradient ascend to be able to investigate, what filters in the network respond to. This could help to find filters, that cause the dimension shift to appear. Optimisation of already used algorithms can also help this tool.Tensor based T-Sne algorithm, instead of original T-Sne could improve the calculation time drastically.

References

Aljundi, R. and T. Tuytelaars (2016). "Lightweight Unsupervised Domain Adaptation by Convolutional Filter Reconstruction". In: *ECCV Workshops* (cit. on pp. 9, 10).

Bonneel, N., J. Rabin, G. Peyré, and H. Pfister (2014). "Sliced and Radon Wasserstein Barycenters of Measures". In: *Journal of Mathematical Imaging and Vision* 51, pp. 22–45 (cit. on p. 27).

Braei, M. and S. Wagner (2020). "Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art". In: *ArXiv* abs/2004.00433 (cit. on p. 9).

Cao, Z., S. Mu, Y. Xu, and M. Dong (2018). "Image retrieval method based on CNN and dimension reduction". In: *2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pp. 441–445 (cit. on p. 11).

Dou, Q., C. Ouyang, C. Chen, H. Chen, and P.-A. Heng (2018). "Unsupervised Cross-Modality Domain Adaptation of ConvNets for Biomedical Image Segmentations with Adversarial Loss". In: *IJCAI* (cit. on p. 1).

Ghafoorian, M., A. Mehrtash, T. Kapur, N. Karssemeijer, E. Marchiori, M. Pesteie, C. R. G. Guttmann, F.-E. Leeuw, C. M. Tempany, B. van Ginneken, A. Y. Fedorov, P. Abolmaesumi, B. Platel, and W. M. Wells (2017). "Transfer Learning for Domain Adaptation in MRI: Application in Brain Lesion Segmentation". In: *MICCAI* (cit. on p. 10).

Girshick, R., J. Donahue, T. Darrell, and J. Malik (June 2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 10).

Goyal, S., A. Raghunathan, M. Jain, H. V. Simhadri, and P. Jain (2020). "DROCC: Deep Robust One-Class Classification". In: *ArXiv* abs/2002.12718 (cit. on p. 9).

Hinton, G. E. and S. T. Roweis (2002). "Stochastic Neighbor Embedding". In: *NIPS* (cit. on p. 6).

Hoyt, C. R. and A. B. Owen (2021). "Probing neural networks with t-SNE, class-specific projections and a guided tour". In: *ArXiv* abs/2107.12547 (cit. on p. 11).

Inoue, T. and Y. Yagi (2020). "Color standardization and optimization in whole slide imaging". In: *Clinical and diagnostic pathology* 4 (cit. on p. 1).

Jenkins, W. K., B. Mather, and D. C. Munson (1985). "Nearest neighbor and generalized inverse distance interpolation for Fourier domain image reconstruction". In: *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing* 10, pp. 1069–1072 (cit. on p. 15).

Kurakin, A., I. J. Goodfellow, and S. Bengio (2017). "Adversarial examples in the physical world". In: *ArXiv* abs/1607.02533 (cit. on p. 1).

Kushibar, K., S. Valverde, S. González-Villà, J. Bernal, M. Cabezas, A. Oliver, and X. Lladó (2019). "Supervised Domain Adaptation for Automatic Sub-cortical Brain Structure Segmentation with Minimal User Interaction". In: *Scientific Reports* 9 (cit. on p. 10).

Lee, K., K. Lee, H. Lee, and J. Shin (2018). "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *NeurIPS* (cit. on p. 9).

Liang, S., Y. Li, and R. Srikant (2018). "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *arXiv: Learning* (cit. on p. 9).

Maaten, L. van der and G. E. Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605 (cit. on p. 6).

Pan, S. J., J. T. Kwok, and Q. Yang (2008). "Transfer Learning via Dimensionality Reduction". In: *AAAI* (cit. on p. 11).

Pooley, R. A. (2005). "AAPM/RSNA physics tutorial for residents: fundamental physics of MR imaging." In: *Radiographics : a review publication of the Radiological Society of North America, Inc* 25 4, pp. 1087–99 (cit. on p. 3).

Quionero-Candela, J., M. Sugiyama, A. Schwaighofer, and N. Lawrence (2009). "Dataset Shift in Machine Learning". In: (cit. on p. 1).

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI* (cit. on p. 25).

Roweis, S. T. and L. K. Saul (2000). "Nonlinear dimensionality reduction by locally linear embedding." In: *Science* 290 5500, pp. 2323–6 (cit. on p. 6).

Shirokikh, B., I. Zakazov, A. Chernyavskiy, I. Fedulova, and M. Belyaev (2020). "First U-Net Layers Contain More Domain Specific Information Than The Last Ones". In: *DART/DCL@MICCAI* (cit. on pp. 1, 9, 10, 25).

Simonyan, K., A. Vedaldi, and A. Zisserman (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *CoRR* abs/1312.6034 (cit. on p. 10).

Souza, R., O. Lucena, J. Garrafa, D. Gobbi, M. Saluzzi, S. Appenzeller, L. Rittner, R. Frayne, and R. Lotufo (2018). "An open, multi-vendor, multi-field-strength brain MR dataset and analysis of publicly available skull stripping methods agreement". In: *NeuroImage* 170. Segmenting the Brain, pp. 482–494. ISSN: 1053-8119. URL: `https://www.sciencedirect.com/science/article/pii/S1053811917306687` (cit. on pp. 4, 23).

Stacke, K., G. Eilertsen, J. Unger, and C. F. Lundström (2019). "A Closer Look at Domain Shift for Deep Learning in Histopathology". In: *ArXiv* abs/1909.11575 (cit. on p. 10).

Tenenbaum, J. B., V. D. Silva, and J. C. Langford (2000). "A global geometric framework for nonlinear dimensionality reduction." In: *Science* 290 5500, pp. 2319–23 (cit. on p. 5).

Tommasi, T., M. Lanzi, P. Russo, and B. Caputo (2016). "Learning the Roots of Visual Domain Shift". In: *ECCV Workshops* (cit. on p. 10).

Torralba, A. and A. A. Efros (2011). "Unbiased look at dataset bias". In: *CVPR 2011*, pp. 1521–1528 (cit. on p. 1).

Valverde, S., M. Salem, M. Cabezas, D. Pareto, J. C. Vilanova, L. Ramió-Torrentà, À. Rovira, J. Salvi, A. Oliver, and X. Lladó (2019). "One-shot domain adaptation in multiple sclerosis lesion segmentation using convolutional neural networks". In: *NeuroImage : Clinical* 21 (cit. on p. 10).

Wang, Z., G. Yuan, H. Pei, Y. Zhang, and X. Liu (2020). "Unsupervised learning trajectory anomaly detection algorithm based on deep representation". In: *International Journal of Distributed Sensor Networks* 16 (cit. on p. 9).

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson (2014). "How transferable are features in deep neural networks?" In: *ArXiv* abs/1411.1792 (cit. on p. 9).

Yu, Q. and K. Aizawa (2019). "Unsupervised Out-of-Distribution Detection by Maximum Classifier Discrepancy". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9517–9525 (cit. on p. 9).

Zhao, X., A. Sicilia, D. S. Minhas, E. E. O'Connor, H. Aizenstein, W. E. Klunk, D. L. Tudorascu, and S. J. Hwang (2021). "Robust White Matter Hyperintensity Segmentation On Unseen Domain". In: *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 1047–1051 (cit. on p. 1).