Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

I need a title

Author:
Tianyang Sun

Supervisor:
Dr Benny Lo

June 2020

Abstract

Your abstract.

Acknowledgments

Comment this out if not needed.

Contents

Chapter 1

Introduction

1.1 Motivation

1.2 Bullshit

1.3 Summarization of out work

In this project, we dealt with two common scenario in medical imaging on segmentation task: (1)only a small set of labelled samples are collected, and (2) a small set of labelled data is available and in addition, a relatively larger amounts of data is collected but not labelled.

For case 1, we explored transfer learning using different pretraining method or available pretrained models. We also tried to understand the network behaviour during transfer learning.

1. First, we pretrained the model on non-Covid Lung volumes to get a pretrained model $F_{pretrained}$, and we also obtained the available pretrained model of Model Genesis [1].
2. Then we perfrom transfer learning using Covid dataset on the two pretrained model weights.
3. We analyzed the results using SVCCA tool to get a further understanding of the Fine-tuned model.

For case 2, we further explored the semi-supervise learning under this typical semisupervise setup. For psuedo labeling, we proposed a method that assign the segmentation label using cosine similarity score from the labelled datset.

1. First, given a pretrained model A, we fine-tune to get A' on the small set of Covid dataset until the model gives relatively good performance (e.g Dice coefficient over 0.75).

2. Next, we random crop 120 68 * 68 patches P_{img} from the labelled datset and store the labels P_{label}. We randomly crop 32 68 * 68 slices from each volume of the unlabelled dataset.

3. Then, we take the encoder part of A'. Given an unlabelled patch $p_{unlabelled}$, we calculate its cosine similarity with each data in P_{img} in the encoded latent space and get the top two most similar labels P_{label_i}, P_{label_j} with similarity score S_i, S_j. We assign the mask with the weighted combinatino of the labels to the unlablled image **Only if both the similarity score is larger than 0.90**.

4. We treated those fake labels in step 3 as 'soft-mask'. We made a copy of A' as A'_{copy} as the student network and trained using the same way as mean-teacher training skeme.

5. We obtained an improvement of ??? percent compared with the transfer learning method.

1.4 Outline of this report

Chapter 2

Background

2.1 Scanning CT

2.1.1 CT and HU values

Computed Tomography (CT) scan leverages X-rays to generate images of the body through rapid rotation of the X-ray tube. Then **attenuation value** of the tissue can be calculated from the intensity reading of the tissue of each voxel to reconstruct the pixels in the images.

Attenuation Value

Given a photon's initial intensity I_0, the attenuation value μ is a linear coefficient describing the change of I_0 after it pass through a uniform object with spacial parameter η. The new intensity can be denote as $I(\eta) = I_0 e^{-\mu\eta}$

Hounsfield Units

Hounsfield units (HU) represents the average attenuation value of each voxel compared to the attenuation value of water. CT numbers can take value between -1000 and 1000 while 2000 shades of grey is out of the capacity of human eyes can distinguish. Thus, only a limited number of HU are displayed for human interpretation. Lung window is normally set to [-1250, 250].

2.1.2 Thick and thin slices

Thin slices are generally regarded as planes representing thickness of less than 3mm [1] In our work, we experienced slice thickness from 1mm to 8mm.
In medical CT scanning, considering the dose of CT, and the equipment limitation, CT slice this varies a lot

[1] http://tech.snmjournals.org/content/36/2/57

2.2 Deep learning fundamentals

In this section, we briefly mention some of the fundementals of Deep Learning and some layer structures used in our experiment, note that most of our notation follows book[2].

2.2.1 Optimizatation in training neural network

Gradient Descent

Fist of all, Gradient Descent Algorithm leverages the first order derivative to modify the weights so that a function gradually reach its local minima. The mathematical definition is:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\alpha} \cdot \nabla \mathrm{J}(\boldsymbol{\theta})$$

in which J denotes the loss function, θ is the trainable parameter and α is the learning rate.

Adam op

2.2.2 Convolutional Neural Network

Convolutional neural networks (CNN) improved deep learning with respect to **Sparse interactions, parameter sharing and equivariant representations** that employ mathematical convolution operation denoted with asterisk symbol $*$. The imaging domain usually make use of discrete convolution:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

We here clarify that in our following notation, we call x the **input**, w **kernel** or **weight**, and s **output** or **feature map**.
In two dimensional case:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n)$$

2.2.3 Activation function

Activation function ϕ (usually non-linear) introduce non-linearity into neural networks. In modern Neural Networks, some of the activation functions are:

- Recitified Linear Unit (RelU): $\sigma(x) = \max(0, x)$
- Eponential Linear Unit (Elu): $\sigma(x) = \begin{cases} \alpha\,(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \mid \alpha \geq 0$
- Leaky Relu: $\sigma(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Softmax: $\sigma(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \mid i \in \{1, \ldots, n\}$ Softmax scale the layer output between 0 and 1 and its sum = 1.

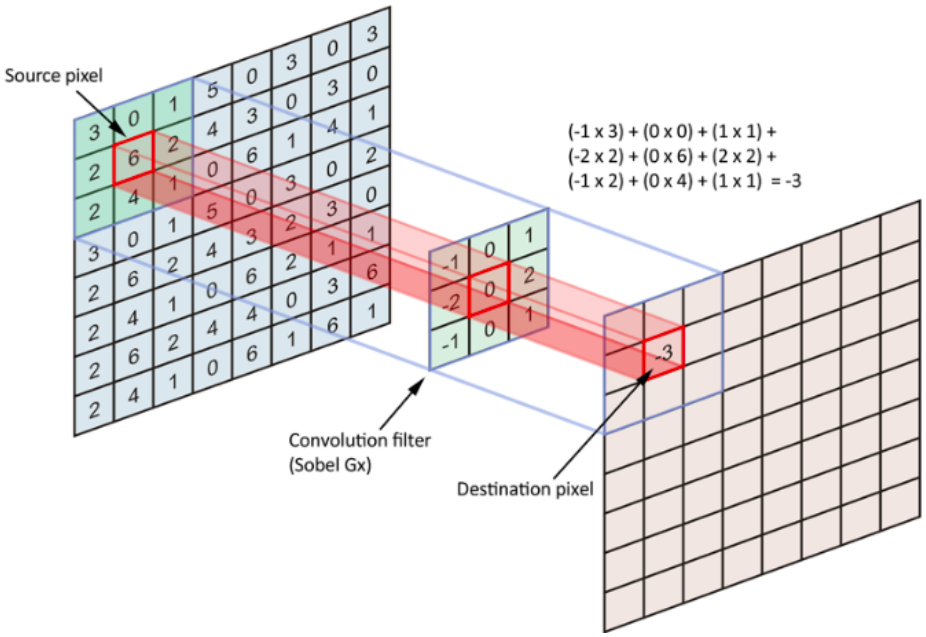

Figure 2.1: An example of convolution using Sobel filter

2.2.4 Pooling

Pooling function modify the output of a layer at a specific location through summarizing its neighboring outputs that helps an approximate invariant. Max pooling is simply the maximum output within a neighborhood. Average pooling takes the average of the neighborhood as output instead of the maximum. An illustration of pooling is shown in figure 2.2 [2]

2.2.5 Batch Normalization

Batch normalization (BN) was proposed to mitigate **internal covariate shift** by fixing the mean and variance of each layer's inputs so that it allows each layer of the network to learn more independently from the rest of the layers.

Batch Normalization adds two trainable parameters to each of the layer. The process of BN is shown below:

[2]https://link.springer.com/article/10.1007/s00521-019-04296-5/figures/1

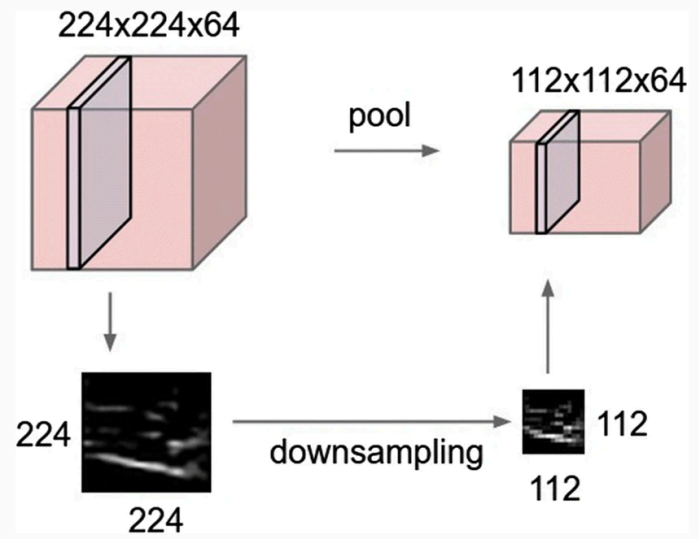

Figure 2.2: Downsampling using pooling

Algorithm 1 Batch Normalisation

Input: Values x over a mini-batch: $Batch = x_{1...m}$, γ, β
Output: $BN_{\gamma,\beta}(x_{1...m})$
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$
▷ Normalizing
$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$
▷ Scale and shift
$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$

2.2.6 Attention Gate

Attention Gate proposed in [] provided a way that we can train the network for segmentation with extra localization objective. Let $\mathbf{x}^l = \left\{\mathbf{x}_i^l\right\}_{i=1}^n$ denotes the activation map over a layer such that x_i^l is the pixel-wise feature vector with dimension equals to the number of feature maps of the output. Attention Gate learn a scaling vector α_i^l as the weight to scale the feature vector, more formally:

$$\hat{\mathbf{x}}^l = \left\{\alpha_i^l \mathbf{x}_i^l\right\}_{i=1}^n$$

The additive attention in the paper, can then be written as:

$$q_{att,i}^l = \psi^T \left(\sigma_1 \left(\boldsymbol{W}_x^T \boldsymbol{x}_i^l + \boldsymbol{W}_g^T \boldsymbol{g} + \boldsymbol{b}_{xg}\right)\right) + b_\psi$$

$$\alpha^l = \sigma_2 \left(q_{att}^l \left(\boldsymbol{x}^l, \boldsymbol{g}; \boldsymbol{\Theta}_{att}\right)\right)$$

in which σ_1 denotes a non-linear activation function such as ReLU and σ_2 is a normalizing function so that $\sum_i e^{q_{att,i}^l} = 1$. In the paper, the author used sigmoid as activation function. Note the the attention gate is usually obtained from a coarser feature map. Figure 2.3 shows the attention gate structure

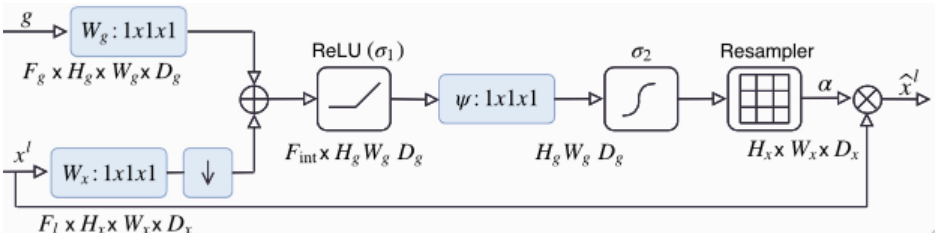

Figure 2.3: Attention Gate structure proposed in []

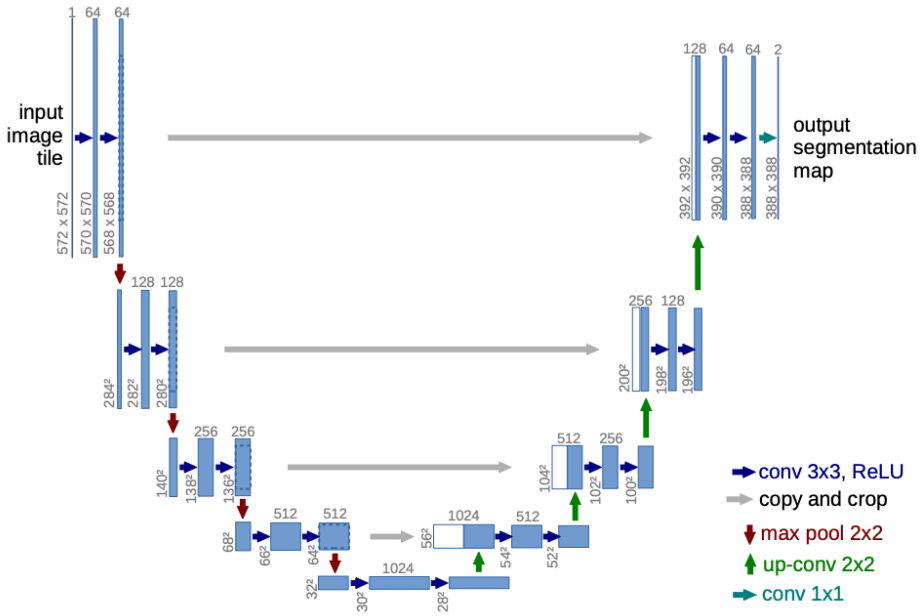

Figure 2.4: Original Unet architecture in [3]

2.2.7 Encoder decoder architecture

2.2.8 Unet & Friends

This section we introduce several well known methods in medical segmentation. Unet [3] and its variations plays a dominant role in current medical segmentation tasks, and it is often used as a baseline model for performance evaluation in the literature.

Unet [3] is among one of the most widely used medical segmentation models since the day it was proposed. The original Unet consists of a contracting path followed by an expansive path that gives a ”U” shaped architecture. The network architecture is shown in figure 2.4. Later this 2D model was extended to 3D version in [4] for for Kidney segmentation tasks so that the model learn features from information implicated between slices.

VNet [5] is another 3D variation of Unet, and the network performed evaluation on prostate dataset. Each block of convolution has a residual feature that the input of

the block is added to the last convolutional layer. The author argued that leverage residual structure enables network convergence in a fraction of the amount of time other network used.

2.3 Small Sample Segmentation

2.3.1 Data augmentation

In medical domain, huge dataset consists of large numbers of carefully labelled samples is rarely available due to heavy workload for annotations, rarity of disease, ethic issues of data acquire process and data privacy. Furthermore, different data acquire protocols (i.e. CT machines used by different hospitals) brings difficulties to clinical practice for good accuracy of existing pre-trained models. As a result, few shot learning and/or few shot segmentation has been explored in recent years. In this section, we focus on a few approaches that has been used in current literature which aim to explore the potential of existing training samples through various augmentation methods to alleviate the insufficient training samples in medical imaging. We discuss data augmentation method as well as the amount of data used in each work. Table 2.1 provide an overview of each method.

2.3.2 Traditional Data Augmentation

Traditional Data augmentation method in imaging domain refers to the process that does not require such training data to learn a transformation.

[6] investigated data augmentation methods under 3D medical domain of MR and ultrasound images. The data augmentation process consists of a sequence of traditional transformation techniques. The paper argued that sharpness in medical images during training process limits model generalization thus applying gaussian filter to images take noise into consideration. Brightness and contrast difference caused by variations in scanning protocols brings potential domain shift thus a sequenced random shift followed gamma correction and random linear transform in intensity are reasonable data augmentation methods. Finally spatial transformations including rotation, flipping, scaling and deformation is added to the augmentation process. The source domain in this method is Prostate dataset [3] which consists 48 4D volumes. We argue here that the stacked transformation is a physical transformation process independent to the size of dataset because no learning or training process is required in this augmentation method thus might bring benefits to our task. However, although Deep learning models (Convolutional Neural Networks) are scale and rotation invariant, medical images differs from natrual images that scanning was conducted with a certain position, i.e. patients usually lie on a CT bed facing up for CT scanning. Thus flipping the lung volumes or rotating the lungs more than 5

[3] http://medicaldecathlon.com/index.html#tasks

Paper	Method	Dataset	Number of samples
[6]	Stacked traditional transform	Prostate dataset	48 4D volumes
[7]	mixup	CIFAR 10	Huge
[8]	mixup	Knee MR images	88 3D MRI
[10]	Asymmetric mixup	BraTS	not specified

Table 2.1: Data Augmentation methods

degrees seems too 'violent'.

Another method "mixup" by [7] based on generic vicinal distribution, which generates new samples through interpolation between two existing data. The author argued that this method works as a regularizer which encourages linear behavior between training samples. In terms of imaging, the augmentation is applied to CIFAR 10 (2D non medical) Dataset. The calculation method in paper follows the following equations where x_i and x_j are training examples and λ is usually sampled from beta distribution.

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j$$

The work was originally implemented on training GANs. Later in medical domain, [8] further investigated mixup method on Knee MR images on OIA database [4] that consists of 88 3D MRI scans. The work showed mixup improves generalization under their experiment setup while having risk of slight underfitting due to to the strong regularization. The author further mentioned not using weight decay in the experiment solve the underfitting issue. [9] summarized mixup augmentation method gives "soft labels". Variations of the mixup method utilize asymmetric is further explored in [10] trained on Brain MR images, and the method reports huge gain under their experiment setup.

2.3.3 Learning for augmentation

Following the traditional augmentation methods, we here want to discuss a few medical image augmentation based on deep learning. Specifically we focus on those who used one or a few data during the augmentation which is close to the problem we are facing.

Adversarial defense was deployed in [11] for the augmentation of small samples in Brain MR segmentation. Adversarial samples generated though Fast Gradient Sign Method[12] and then added as training data to improve the robustness. So far we believe this method worth trying because the experiment was trained on 7 brain volumes, which is close to our task.

[4] http://www.oai.ucsf/.edu/

2.3.4 Network

2.3.5 Transfer Learning

In medical domain, few shot learning mainly focus on transfer learning from pre-trained networks that leverage both medical and non-medical datasets.

In Lung CT segmentation area, Sports-1M dataset has been used as source domain to train a multi-task learning model for nodule malignancy prediction and rating [13]. The author reported significant improvement in the prediction accuracy, however, did not mention the proportion of data used for transfer training.

People tend to choose datasets from closer domain for transfer learning. It is reasonable to consider methods that transfer across disease in the same structure under the same modality. In our case, we might want to investigate transfer learning from NSCLC Dataset to Covid segmentation set given that both of them are lung CT scans.

Recent work explored several across disease transfer learning training techniques under MRI domain [14]. The paper evaluated three transfer learning methods trained on 3D U-Net by Fine tuning the last three layers, Fine tuning the decoder and Fine tuning all model parameters. The Source Dataset: Multiple Sclerosis Dataset consists 3630 MRI volumes and used Brain Tumor Dataset as Target dataset including 210 high-grade glioma (HGG) and 75 low-grade glioma (LGG) Brain MRI scans. The training target is a decaying weighted categorical cross entropy loss weighted by relative voxel. Their best validation performance of pre-trained network achieved validation performance AUC 0.77. Experiment result on 20, 50, 100 and 150 samples during Fine tuning respectively showed that Fine tuning all parameters out performed the rest methods in most cases.

One potential drawback is that compared to the our task, the target training set is relatively larger, the performance is expected to be less ideal when using ”fine tune all” method using 4 or less volumes in our case.

2.3.6 Special network design

Transfer learning methods usually require small samples to update millions of parameters that take the risk of overfitting [15]. The design of multiple branch network, usually includes conditioner arm and segmentation arms inspired the deep learning in medical domain to go beyond transfer learning while encourage a stronger between-arms interaction to compensate the lacking in pre-trained model [16]. The proposed method perform 3D volume segmentation at test time while use 2D images during training. However the method requires start and end slice to be indicated for each query volume and still not achieving good dice score.

Paper [11] trained segmentation of Brain MR image on 7 brain volumes after Adversarial defense augmentation. The work first split the segmentation from easy to hard

Paper	Method	Domain Details	Task
[15]	Design Conditional Branch	Target PASCAL-5	Few shot segmentation
[17]	Guidance network	Target PASCAL VOC	Few shot segmentation
[13]	Non medical to medical transfer	Source: Sports-1M; Target: Lung nodule	Multi-task learning: prediction and rating
[14]	Across domain transfer	Source: MSD; Target: Brain Tumor Dataset	Segmentation
[15]	Augmentation+pixel dense segmentation	Only trained on 7 brain Volumes	Dense segmentation
[11]	Branch network design	–	Segmentation

Table 2.2: Small sample methods in medical and non-medical domain

into 2 individual classifiers then joint learning with dense pixel segmentation. The author reported that the result outperformed Unet and Vnet method trained from scratch. We argue that the augmentation provide good result in the segmentation accuracy while the segmentation part is not well explained in the paper. We have emailed the author for further details.

2.3.7 Learning with unlabelled data

Another common scenario with medical imaging is that, a small set of labelled annotations is available and a relatively larger set of data was collected but not labelled. In this section, we briefly mention three types of work that leverage unlabelled data: training encoder, noise removing, and semi-supervise learning. Both training encoder and noise removing serves as a pretrained network that are usually fined-tuned using the methods we described in the previous section. Semi-supervise learning however, How to describe this

Training encoder as initialization

Training encoder part without expert labeling usually make use of spatial information of the image such as slice order [**?**] and direction [**?**].

In the work [**?**], authors trained a network to predict a transformation of orientation by rotating or flipping the input slice, and then fine tune the network to classify retinal imaged of diabetes patients. One potential use of this type of initialzation for segmentation task is that we can append the Up-Convolution then transfer on segmentation labels.

Paper [**?**] trained a encoder that, given a reference slice and a prediction slice, predict the prediction slice is above of behind the reference slide so that the network can learn spatial information using the unlabelled images.

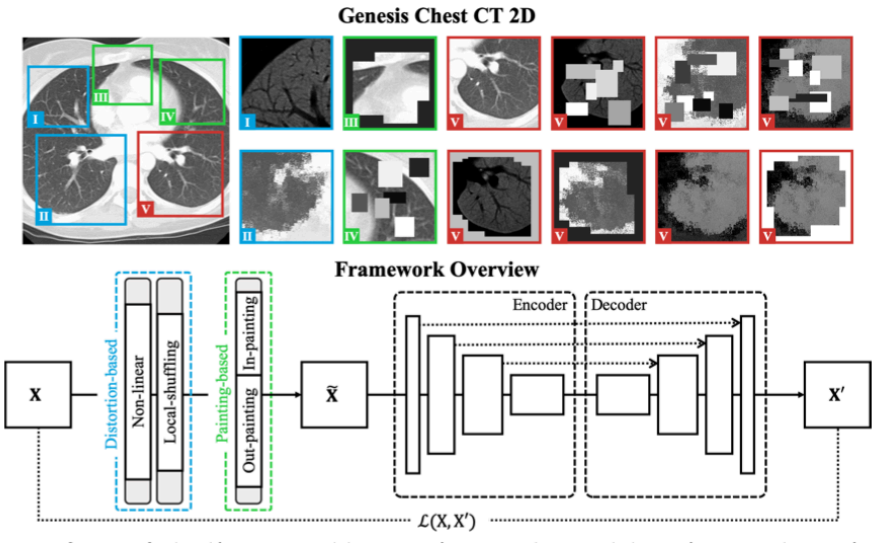

Figure 2.5: An illustration of Model Genesis pre-training (Figure from the original paper [1])

Noise removing

Instead of training a surrogate task such as training encoder leveraging spatial or location information as a pretraining task, Zongwei Zhou [1] provided another probability for pretraining both encoder and decoder that used together can serve as weight initialization for a segmentation model, and encoder used on its own is a pretrained feauture extractor for classification task.

The work trained a Unet-style network that given a randomly cropped and sliced image, first destroy the image through a sequence of random non-linear transformation, local shifting, out/in-painting then forward the image to restore the original image by minimizing the mean-square-error during training. The author argue that in that way, both the generalizability and the detail feature encoding can be learnt which can be leveraged in later transfer learning process. Figure 2.5 showed the restoration process. The model was trained on Lung CT images which is close to our task while the provided model was in 3D version, later in the experiment stage, we actually flattened the pretrained weight into 2D for future training.

Semi-supervise

Semi supervise approach usually add a prediction consistency loss during training. [18] propsed a teacher-student training strategt such that two networks are train together. The proposed so called 'Mean-teacher' network trained two network simultaneusly. First the author train a model on the fully labeled dataset, that serve as the teacher model. In the beginning of the semi-supervise training, a copy of the teacher model is also created called the student network. Now given a new set of unlabelled raw images, the student network are trained using the weight of psuedo-label cost and the prediction consistency cost. The teacher model weights

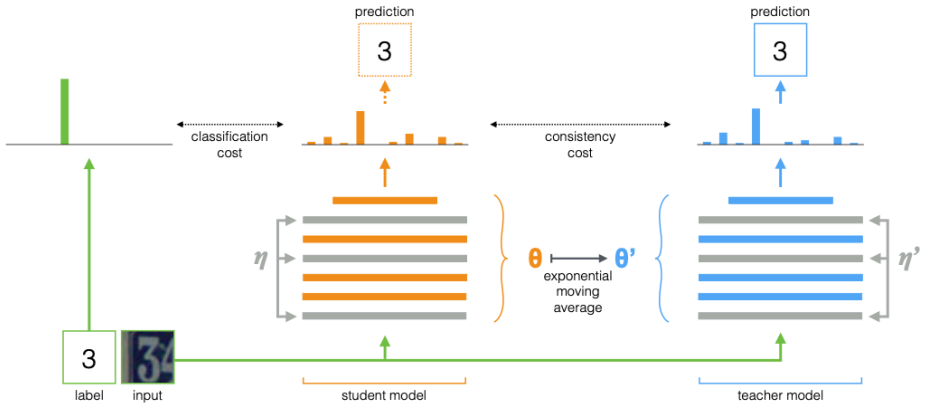

Figure 2.6: An illustration of mean teacher for semi-supervise learning (Figure from the original paper [18])

are updated periodically using an exponential moving average of the student network. Figure 2.6 explains the training process.

We here further gives the defnition in the original work [18]. More formally, define the consistency of the prediction as:

$$J(\theta) = \mathbb{E}_{x,\eta',\eta}\left[\|f\left(x,\theta',\eta'\right) - f(x,\theta,\eta)\|^2\right]$$

where x is the input data (image), θ, η denotes the weight and noise in the teacher model and θ', η' denotes the weight and noise in the student model. The weight update in the teacher model can then be written as:

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$$

in which α is the smoothing parameter showing how much the teacher want to move given the new student network.

We argue here that this training strategy can help our task 2 given a relatively larger amount of unlabelled samples are available.

2.4 Evaluation

2.4.1 Quantifying Segmentation prediction

Loss function or objective function is a crutial component in neural network training. Segmentation tasks usually make use of *Distribution loss*, *Region based loss* and *boundary-based* loss for training and evaluation of segmentation performance. Recent work in [19] summarized some common loss functions for segmentation.

Cross entropy (CE) measures the dissimilarity between the learned distribution and target distribution.

$$CE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} w_c \cdot (y_i^c \log p_i^c)$$

where y_i^c indicates the prediction result (correct or wrong) and p_i^c denotes the predicted probability of pixel i for class c, w_c is now 1 for original cross entropy loss. Unet [3] training extend the CE by adding weight w_c. A common example for weight measurement is through the inverse proportion of observed class frequency. This modification potentially deal with imbalance class which is very common in medical domain.

Dice loss is a region based loss function that learn to optimize the Dice Coefficient (D). Vnet [5] first brought the Dice loss into machine vision community to solve the problem of highly biased prediction towards dominant area (e.g background) in medical segmentation.

$$D = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

Assume we segment N samples, p_i denotes the prediction volume and g_i denotes the ground-truth volume. Dice loss usually require the label to be one hot encoded during training. One benefit is that Dice loss does not requires class balance methods such as weighting method in CE loss.

Hausdorff Distance loss(HD) aims to minimize the boundary distance between prediction and ground-truth segmentation masks. Similar to Dice loss, it also alleviate the class imbalance issue during training. However,directly minimizing Hausdorff Distance is intractable while an approximation (HD Loss) through distance transform (gray-level intensities of points inside foreground regions are changed to show the distance to the closest boundary from each point [5]).

$$L_{HD_{DT}} = \frac{1}{N}\sum_{i=1}^{N}\left[(s_i - g_i)\cdot\left(d_{Gi}^2 + d_{Si}^2\right)\right]$$

Paper [19] proposed that so far none of the papers in the literature provide a comprehensive comparison of the loss functions for segmentation task. Selecting loss function is still based on empirical comparison. For example, [**?**] used compound loss function that combined CE and Dice together as training objectives overall gives good performance compared to individual loss functions.

2.4.2 Evaluating Transfer Learning

Loss function quantifies the transfer learning prediction accuracy. We further want to understand the model behaviours. For example: How weights are updated during fine tuning? Is Pretraining compared to random initialization result in different

[5]https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm

weights such that the the latent space are apart from each other?

Singular Vector Canonical Correlation Analysis (SVCCA) [20] developed by Google Brain provide a method that compares the latent space of different models or different layers. The method is more quite simple while useful for comparing high dimensional latent features.

First let l denotes the output of layer over a dataset D, The paper perform the following SVCCA steps:

- Given two layers of output l_1, l_2 that represent the learnt subspace on the dataset, perform singular value decomposition then select the new subspace l_1', l_2' which preserve 99% of the original variance.
- Perform Canonical Correlation Analysis on the two subspace l_1', l_2' so that the two new subspaces are as correlated as possible through transfomation. Each direction has a correlation ρ_i
- The correlation of two subspaces is the average of each output correlation $\rho_{l_1,l_2} = \frac{1}{N}\sum_{i=0}^{N}\rho_i$

SVCCA interpret the result of any two learnt subspace over a dataset. Later in work [21], the author utilized the method to understand the behavior of transfer learning in rather larger amounts of data on medical classification task.

2.5 Ethics and professional considerations

Chapter 3

Data

3.1 Data description

One of the most straightforward way to deal with the limit amount of available data is to leverage as much related Medical Data as possible. We thus investigated several public available dataset for Lung CT scans in addition to the Covid-CT dataset.

3.1.1 NSCLC Dataset

NSCLC Dataset contains 402 Lung CT scans, of which 78 cases are anotated with left lung, right lung and pleural effusion area. A sample annotation is shown in figure 3.1

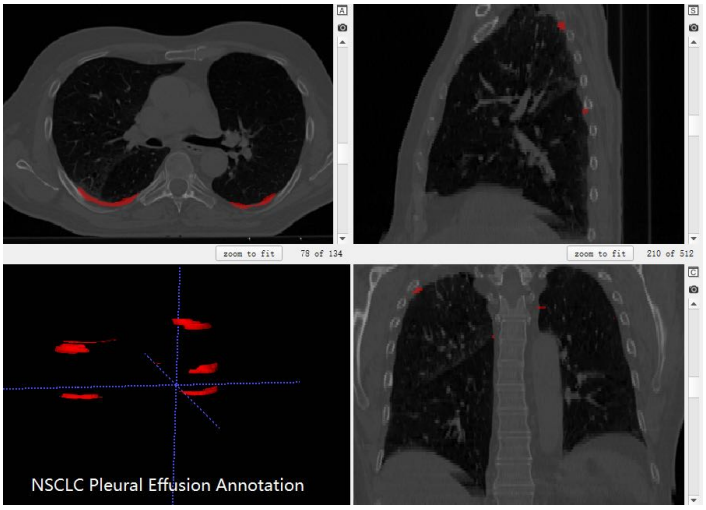

Figure 3.1: An example volume from NSCLC Dataset with its annotation

3.1.2 MSD Lung Tumor

MSD Lung Tumor contains 63 Lung CT scans, annotating the Lung Cancer area. An example shown in figure 3.2

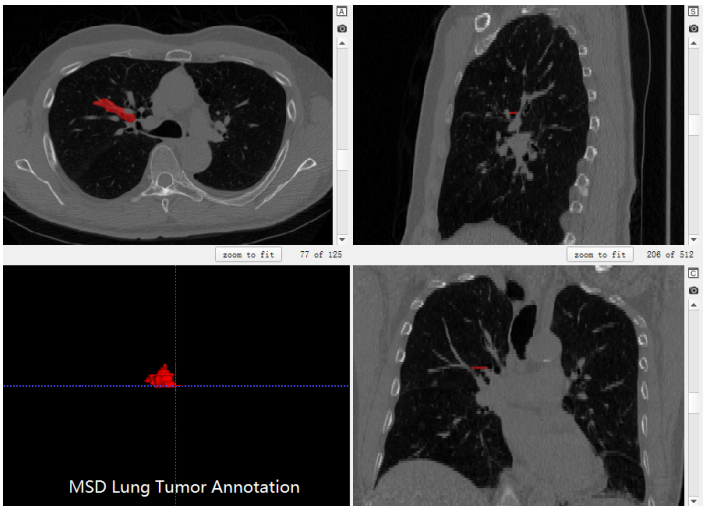

Figure 3.2: An example volume from MSD Dataset with its annotation

3.1.3 MosMed Dataset

MosMed Dataset Contains 50 Annotated thick-slice Covid CT scans, as well as around 300 unannotated Lung CTs. We'd like to report here that we used only 200 unannotated slices because downloading keep giving me error due to location restriction. An example shown in figure 3.3

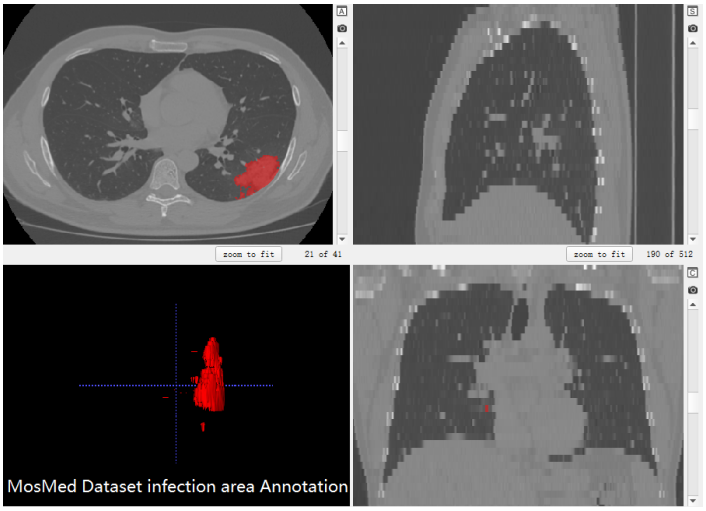

Figure 3.3: An example volume from MosMed Dataset with its annotation

3.1.4 Covid Segmentation Benchmark

Covid Segmentation Benchmard contains 20 CT scans from 2 radiometric centers, of which 10 volumes thin-slice CT volumes and 10 thick slice CT volumes.

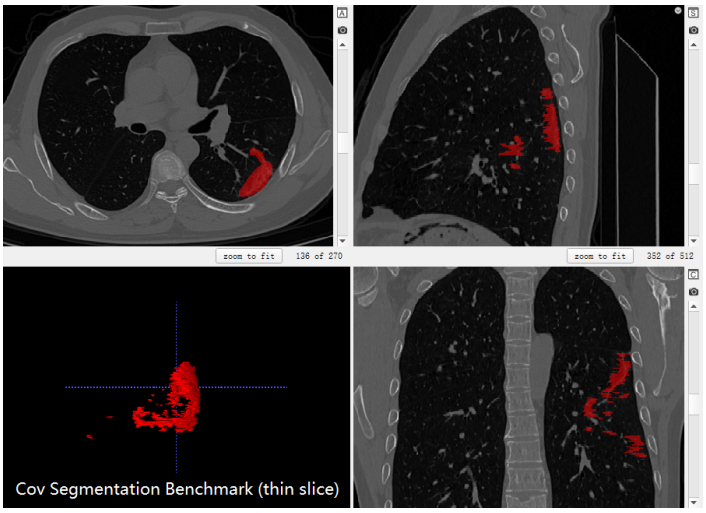

Figure 3.4: An example volume from Cov Segmentation Benchmark (thin slice) with its annotation

3.2 Data preprocessing

The dataset used in this paper was collected from multiple centers with different equipment setup. Structseg2019, NSCLC, MSD Lung Tumor set are thin slice CT scans. MosMed contained 50 thick slice scans and the Covid segmentation benchmark are multi-center dataset from different centers. We argue that deep learning algorithms, especially segmentation tasks are sensitive to this difference. To make most use of the data, we first extract the lungs from the CT volumes. Then a sequence of preprocessing was performed including resampling, histogram equalization, and mean variance normalization. Most implementation used SimpleITK dataset

3.2.1 Data gathering and cleaning

3.2.2 Processing into functional features

Lung CT scans images includes the lung tissue as well as bones and meshes that influence the preprocessing such as normalization as well as future segmentation. We intended to first segment the lung tissue out for better focus.
Lung segmentation using watershed algorithm

Lung segmentation Using pre-trained Deep learning models
Deep learning method provide finer results when facing severe pathology. Work in [22] provided a promising result for Lung segmentation. In addition, they further improve their lung segmentation model with Covid-19 Lung data.

We leveraged their models provided in their github repository [1]. Original volumes from the Lung datasets went through the model, we threshold the lung out and

[1]https://github.com/JoHof/lungmask

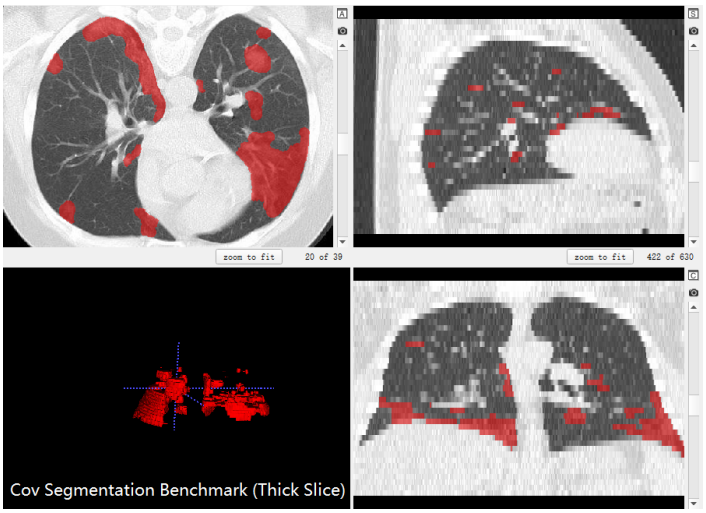

Figure 3.5: An example volume from Cov Segmentation Benchmark (thick slice) with its annotation

set the uninterested area (background) as 0. Figure 3.6 showed and example lung volume before and after filtering the lung tissue.

3.2.3 Resampling

Figure 3.7 showed the diveristy in spacing from different datsets. To deal with the different spacing for multi-domain data, we resampled the data to spacing (1, 1) in the Axial view and remain not changed in the Z axis.

3.2.4 Mean Variance Normalization

We performed Mean Variance Normalization (Z score normalization) for the lung tissue voxels by subtracting each volume with its mean and divided by its standard deviation. So that the data has zero mean and standard deviation of 1.

$$z = \frac{x - \mu}{\sigma}$$

of which μ is the mean and σ is the variance. Figure 3.8 showed the intensity range before and after normalization

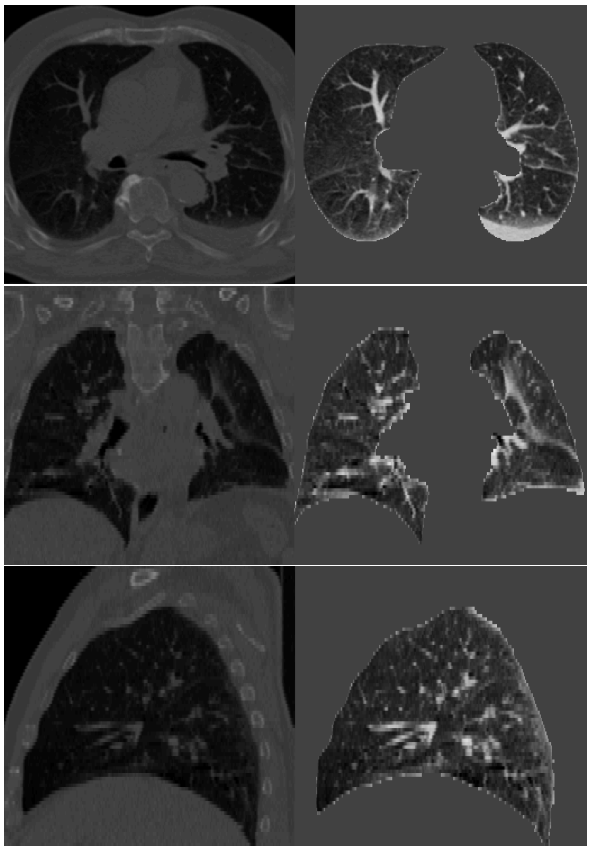

Figure 3.6: An example from NSCLC Pleural Effusion dataset showing the volumes before and after lung filtering. (Top: Axial view, Middle: Coronal view, Bottom: Saggital View)

3.3 Data Augmentation

For data augmentation, we investigated some of the augmentation techniques in medical imaging domain, each of the augmentation techniques was carefully chosen matching real medical situation.

Rotation: We performed a small random rotation between [-3, 3] degree considering the pose variation when lying down on the scanner.

Elastic Transformation: We performed a small elastic transformation considering lying down and holding breath when scanning Lung CT might brings shape change to the lungs tissue.

Random Gamma and Gaussian Noise: We performed a random gamma correction to simulate the variation generate due to different equipment. We also added a random gaussian noise for a more robust training.

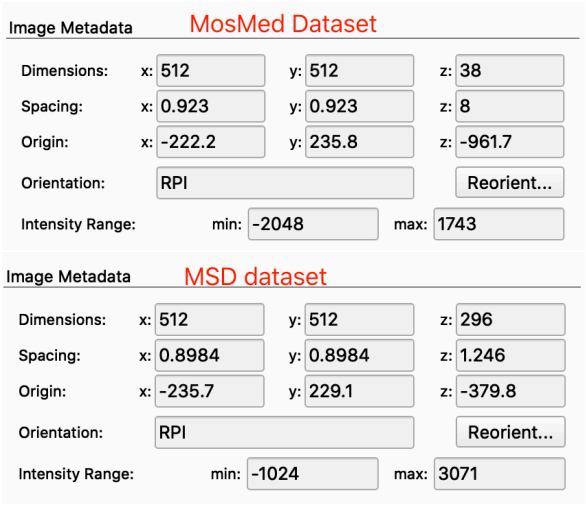

Figure 3.7: An example of different spacing information from MosMed and MSD dataset

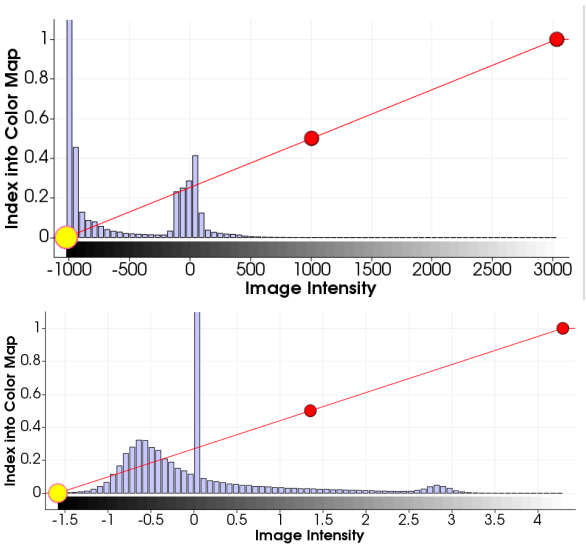

Figure 3.8: An example showing the intensity range before and after normalization (Top: before normalization; Bottom: after normalization)

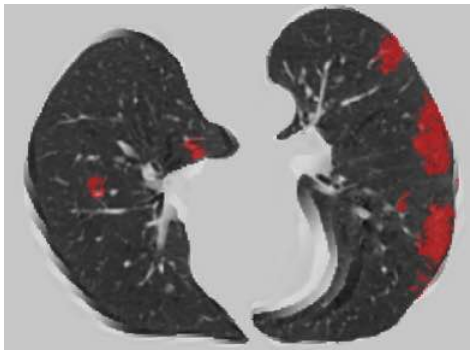

Figure 3.9: An example of rotation using MosMed Dataset

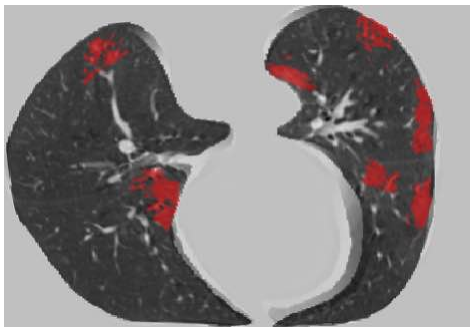

Figure 3.10: An example of elastic transformation using MosMed Dataset

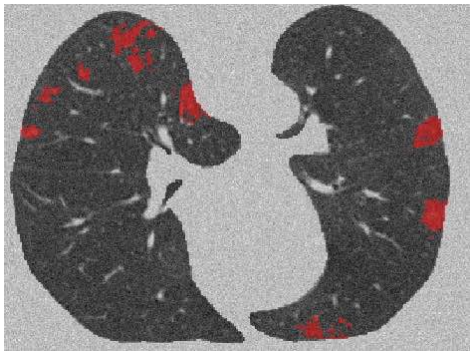

Figure 3.11: An example of random Gamma and Gaussian Noise using MosMed Dataset

Chapter 4

Deep learning architecture

4.1 Network Architecture and Methodology

Recall that in our project, due to large variation in the axial axis, we specified our problem into 2D segmentation task.

4.1.1 Unet

We implemented a 2D Unet model as the baseline model, the original Unet structure is described in 4.1. We further added a Batch Normalization layer before each activation better to alleviate covariance shift. The input image channel is one in our case because CT volumes are grey scale images.
We selected the input as original size because we want to reserve as much detail as possible. The input dimension is then (batch-size $\times$ 1 $\times$ 512 $\times$ 512), and the output dimension after softmax is (batch-size $\times$ 2 $\times$ 512 $\times$ 512) for the binary segmentation task, and we take an argmax to get the binary image the same size as the input.

4.1.2 Attention Gated Unet

We used the original Attention Gated Unet architecture recommended in the paper [**?**] that we added the Gate block before the skip connection concatenate to the upconvolution result in the corresponding layer. An illustration is shown in figure 4.1.

4.1.3 Loss function

training objectives

In this project, we used a combination of Foreground Soft Dice loss and Cross Entropy Loss. The justification of these combination is:

Layer	input	output	channel
Conv 3x3 + RELU	572	570	64
Conv 3x3 + RELU	570	568	64
Max Pool 2 × 2	568	284	64
Conv 3x3 + RELU	284	282	128
Conv 3x3 + RELU	282	280	128
Max Pool 2 × 2	280	140	128
Conv 3x3 + RELU	140	138	256
Conv 3x3 + RELU	138	136	256
Max Pool 2 × 2	136	68	256
Conv 3x3 + RELU	68	66	512
Conv 3x3 + RELU	66	64	512
Max Pool 2 × 2	64	32	512
Conv 3x3 + RELU	32	30	1024
Conv 3x3 + RELU	30	28	1024
up-conv 2 × 2	28	56	512
convs 1x1	56	54	512
convs 1x1	54	52	512
up-conv 2 × 2	52	104	256
convs 1x1	104	102	256
convs 1x1	102	100	256
up-conv 2 × 2	100	200	128
convs 1x1	200	198	128
convs 1x1	198	196	128
up-conv 2 × 2	196	392	64
convs 1x1	392	390	64
convs 1x1	388	388	2

Table 4.1: The Original Unet Architechture

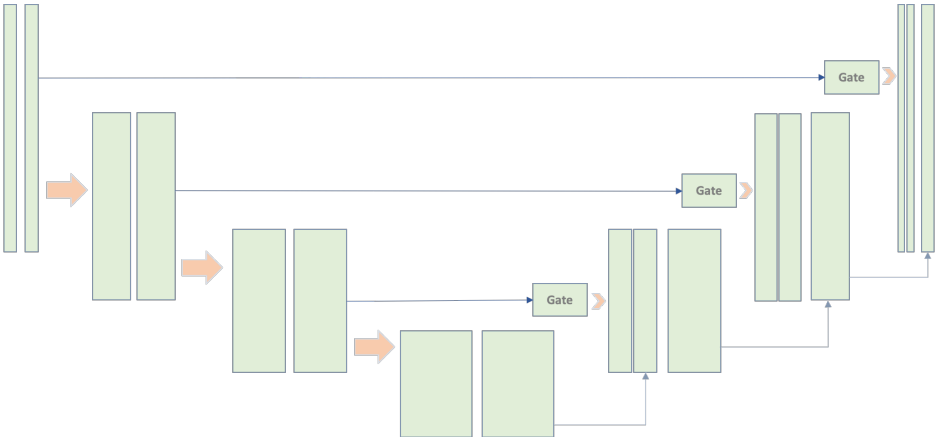

Figure 4.1: An illustration of the attention block added in the Unet

- We consider CE loss converge fast for the background to learn the rough location of the segmentation
- We consider Foreground Dice only because for our highly imbalanced segmentation problem, adding background will vanish the loss and our segmentation target will not learn well.

Mathematically, we write our objective function as:

$$Loss = \frac{1}{I}\sum_{i=1}^{I} 1 - \frac{2\sum_{l=1}^{L}\sum_{c=1}^{C} p_l^c r_l^c}{2\sum_{l=1}^{L}\sum_{c=1}^{C} p_l^c + r_l^c} - -\frac{1}{N}\sum_{c=1}^{N}\frac{1}{w_c}\sum_{l=1}^{L} r_l^c \log\left(p_l^c\right)$$

Validation and testing scenario

For Validating or testing the network, we used the foreground hard Dice Loss calculating the range of overlap.

$$DiceLoss = \frac{2|A \cap B|}{|A| + |B|}$$

4.1.4 Transfer Learning

As we discussed in the background chapter, the pretrained network from a source domain can serve as a good starting point for the issue especially when training data is not that sufficient. We make use of two different initializations one trained on our own and another one downloaded online, and fine tune the model on both the plain Unet and Attention Unet.

Pretraining

For pretraining, we leverage two model initializations. We pretrained the model with the mixed Non-Covid lung Dataset, and we also used the weight initialization of the Model Genesis Dataset.

- First the we pretrained the segmentation model using non-Covid dataset based on the assumption that the weights serves as good starting point for the fine-tuning stage.
- Paper [1] pre-trained a 3D Unet-like model using several public dataset in the way that trained the model to restore image using destroyed image. The model was published in 3D [1]. Since we did out experiment in 2D, we first load the 3D model and extract the layer weight. Then we flatten the weights into 2D as the initialization.

[1] https://github.com/MrGiovanni/ModelsGenesis/tree/master/pytorch

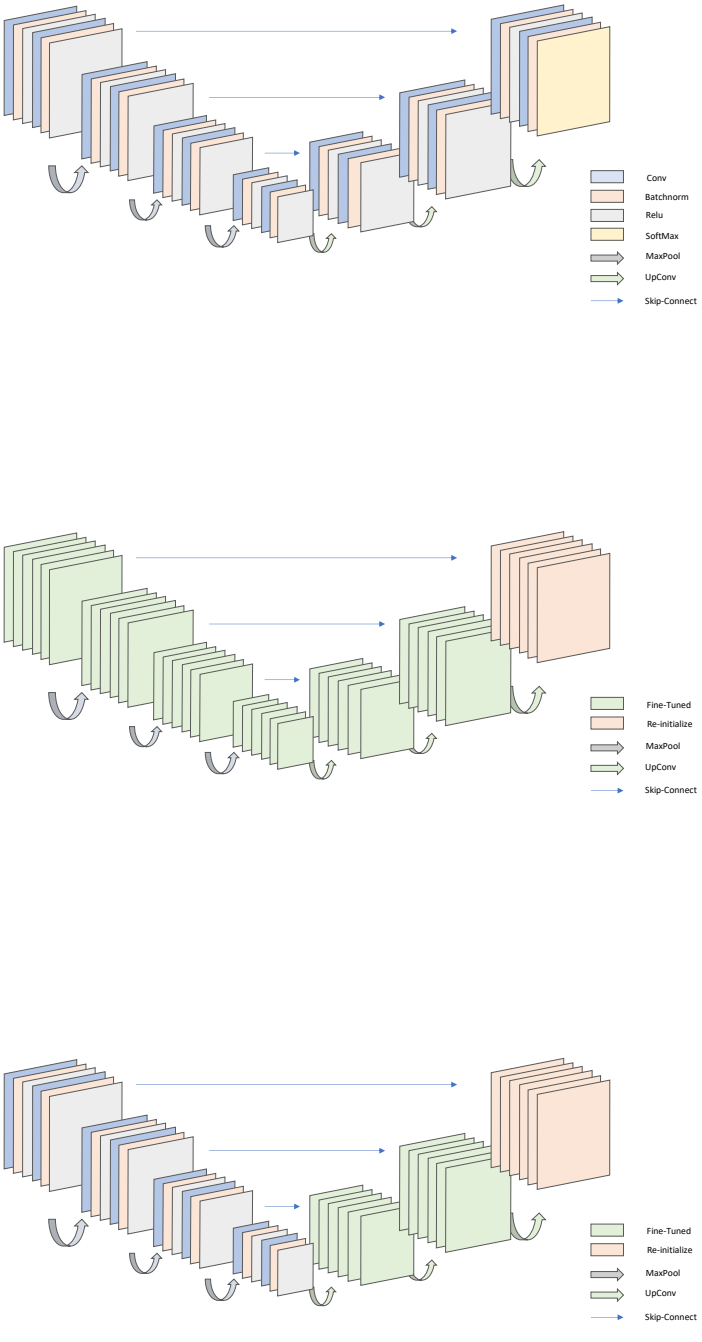

Figure 4.2: An illustration of Network structure used for pretraining and transfer-learning

Fine-Tuning

We fine tuned the two pre-trained model using both of the initialization to train the Unet and the Attention Gate Unet architecture. We did experiment that allow the network to fine tune all or we freeze the encoder part. The justification of freezing the encoder can be further explained in the experiment section that SVCCA showed that the encoder showed less movement during fine-tuning.

1. Freeze encoder: We freeze the first half of the encoder, and fine tuned the decoder except that we re-initialized the output layer.

2. Fine-Tune-All: The whole network is fine tuned and the last block of convolutional layers was reinitialized.

- Fine-Tuning Unet: We directly load the trained model from the image and train the network and did the experiment with the two Fine-Tuning methods mentioned above.

- Fine-tuning Attention Gate: We consider that the attention gate for different domain are different, so for both of the fine-tuning methods, we take the encoder part only and append new decoder and attention gate to train from scratch.

4.2 SVCCA implementation

4.3 presenting with unlabelled data

Given that a larger amount of unlabelled data was published later in our project, we further investigated the semi-supervise segmentation area leveraging unlabelled dataset. We constructed a semi-supervise learning framework including psuedo labeling, and mean-teacher training with our loss function.

4.3.1 Psuedo labeling

We developed a new psuedo labeling framework that assign a reasonable label to the unlabelled samples.

We sampled both the labelled samples and the unlabelled images. For the labeled samples, we first get the bounding box of the segmentation mask, then enlarge the mask by a maximum of 10 pixels each side, and we sampled images within that bounding box. For unlabeled images, we use the coarse 3D segmentation to generate the bounding box, and the bounding box was enlarged maximum of 15 pixels, we sample patches within the coarse bounding box. To assign a noisy psuedo label to the cropped images, we take the encoder part of the trained 2D segmentation model and append a Global Average Pooling function. We then encode a dictionary of the

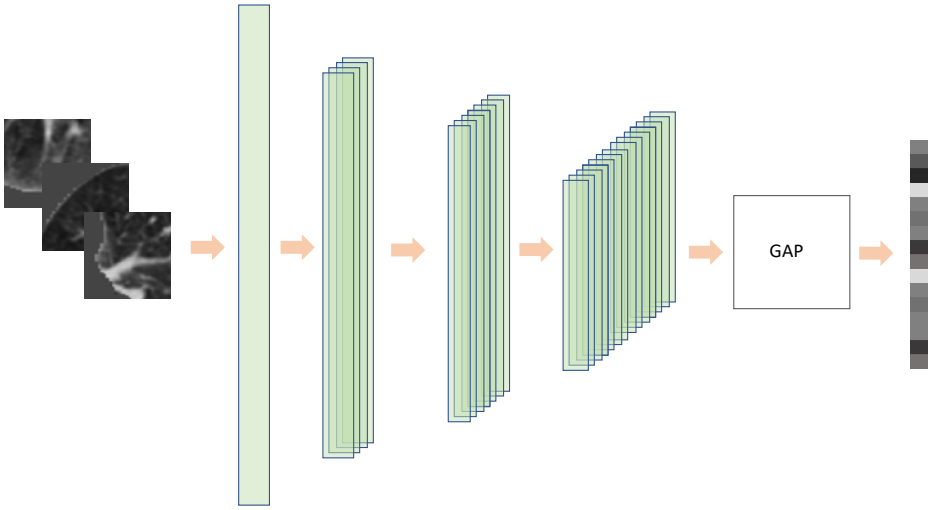

Figure 4.3: An illustration of feature encoding process

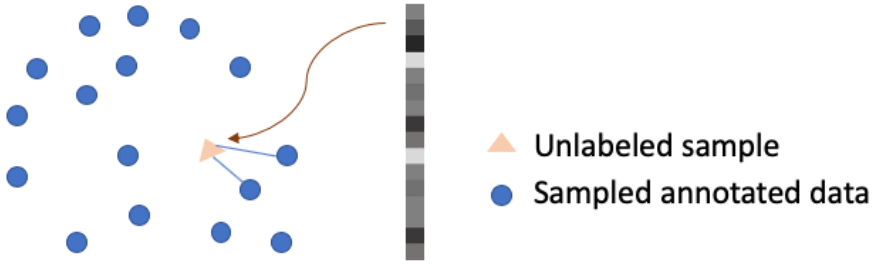

Figure 4.4: An illustration of label assigning to unlabelled samples

annotated samples in the feature space.

For assigning noisy psuedo labels, given an unlabeled sample $I_{unlabeled}$ and its latent representation $Latent(I_{unlabelled})$, we calculated the pairwise cosine similarity and return the maximum cosine similarity score as the weight for this sample.

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

An illustration of the label assigning process is shown in figure 4.4

4.3.2 Semi-supervise architecture

After assigning reasonable psuedo labels to the samples, as we described in Chapter 2, we designed a Mean-teacher style model for the semi-supervise task.

The teacher model is the 2D Unet we described in section 4.1 transfered usining the non-covid dataset because we found that transfer learning give a better result as a intialization (detail in chapter 5), except that we only update the weight for no more than 10 epochs becuase we want the teacher model to be a bit more noisy than the converged model.

We first copied the trained weights to get the student model. For updating weights, we let the same patch go through the teacher and student model, and calculate the prediction consistency, we calculate a degraded dice score on the student network prediction with the pseudo label, and we set the weight to 1 * Dice Loss for a lighter penalization. The weight decreased every 5 epochs by 0.8. The student network weights are updated per batch (or observation) and the teacher model is update once per epoch.
Figure 4.5 showed an illustration of the weight update process using this framework.

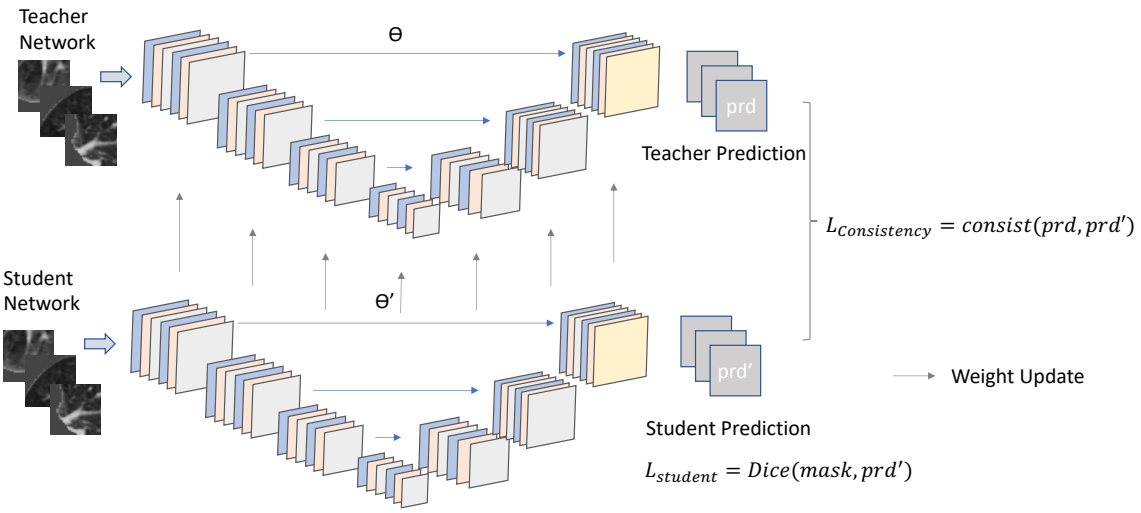

Figure 4.5: The design of network model for our semi-supervise setup

4.3.3 Loss function for training

Mathemartically, we write the Loss function as

$$L_{train} = L_{consistency} + L_{segmenation}$$

$$L_{train} = CE(prd_{teacher}, prd_{student}) + Dice(prd_{student}, mask)$$

4.3.4 Validation and testing

For the evaluation, we used the Dice Loss as we did in section 4.1.

Chapter 5

Experiment

5.1 Experiment

A good model, especially facing with small sample segmentation, should generalize well instead of overfitting on a small number of training data. The whole point of data augmentation, adding noise, etc, is to improve the generalizability of the trained neural network so that for the unseen data, the model predict reasonably well.

Due to the limited ability of the GPU resource available, tuning hyper-parameters using grid search is too inefficient. Thus in our experiment, we leveraged the optimizer scheduler in PyTorch so that starting from the learning rate starts from $lr = 2 \times 10^{-4}$ and decreased to 0.8 * lr every 25 epochs so that the learning rate approach 0 in the later training stage.

1. **learning rate**

2. **Optimizer:** We used Adam optimizer considering that Model Genesis, winning solution for NSCLC segmentation as well as other well known solutions used the model optimizer.

3. **Epochs:** The maximum number of epochs for training is 500. However, each epoch we validate the result and the best model was saved throughout the training process. We count the number of continued non-improving epochs, and when it reached 30 epochs, the training progress terminated automatically.

5.2 With Fully labelled data

We first consider training on a small dataset and all of them are labelled with segmentation masks, because this is the case during the first few months of this project when only 20 volumes of the Covid lungs from the Covid Segmentation benchmark were publicly available, later in July 2020, MosMed published a new dataset containing 50 labeled slices.

5.2.1 Experiment Setup

To fully leverage the Covid Dataset, we combined the 20 volumes in Covid Segmentation bechmark and the 50 volumes MosMed Dataset. We randomly selected 20 volumes for testing, and further split the 50 remaining volume into 5 fold for cross validation. Smaller sample: One fold (10 volumes) for training; Normal training: Four fold (40 Volumes) for training. Note that since we cannot do anything to the various slice thickness, we sliced the volume into 2D.

5.2.2 Best results

For the segmentation of infection area using **fully labelled sample only**, we achieved Dice score (DSC) on the training set (50 volumes sliced in 2D) of 99.0321%, 80.3601% on the validation set and 79.9356% on the testing set. All the volumes was preprocessed as described in Section 3 and augmented the data using random rotating, and elastic transformation and mix up augmentation on the training set only.

5.2.3 Training

We compared the model trained from scratch and with transfer learning in this experiment. More specifically, we experimented two transfer learning intialization in this section. First we trained the Binary segmentation task with the images from NSCLC and MSD dataset, then the model was fine tuned using two different methods suggested in work [14]: Fine-Tuning all layers (reinitialize the last layer), and Fine-Tuning only the Decoder, shown in figure 4.2.

Unet		Train	Validation	Test
From Scratch	10-40 split	0.9894	0.7012	0.6893
	40-10 split	0.9705	0.8119	0.8012
Fine Tune Decoder	10-40 split	0.9771	0.7963	
	40-10 split	0.9693	0	
Fine Tune All	10-40 split	0.9802		
	40-10 split	0.9562		
Attention gated Unet		Train	Validation	Test
From Scratch	10-40 split	0.9904		
	40-10 split	0.9710		
Fine Tune	10-40 split	0.9521		
	40-10 split	0.9512		
Fine Tune All	10-40 split	0.9608		
	40-10 split	0.9611		

Table 5.1: Training with fully labeled Dataset

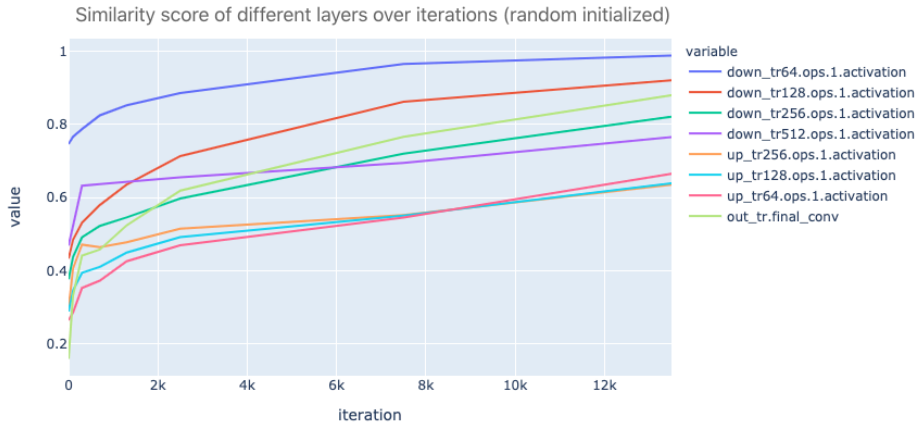

Figure 5.1: CCA similarity score of each layer over training iterations (randomly initialized)

5.2.4 SVCCA analysis on transfer learning

Network convergence during training

We compared the convergence on each layer throughout the training process on the segmentation model. [**?**] reported that the network converged bottom up for a classification task during training. However, this is not exactly the case in our segmentation model.

Network convergence with random initialization

We first want to analyze the change of network layers from random initialization towards convergence using the SVCCA tool. We iterate through the dataloader and validate every 100 epochs and save the model if the result yield better performance.

Figure 5.2 plots the similarity of latent space of each network layer using random initialization.

- We observed that, in general, encoder converged faster than decoder.
- Suprisingly, the first down convolution layer moved less comparing the initialization to the converged model.
- The similarity score in the output layer converged slightly faster than the other decoder part. Our explanation is that, the model performance (accuracy) increased faster in the first few number of iterations then slowly improved throughout the training.
- Although the similarity of the several Up-Convolution in the decoder part kept changing in the later stage of training, the output layer(out_tr.finalconv in figure 5.2) did not moved much. One implication is that, Neural Network may learn different latent feature space while giving similar performance with respect to the accuracy.

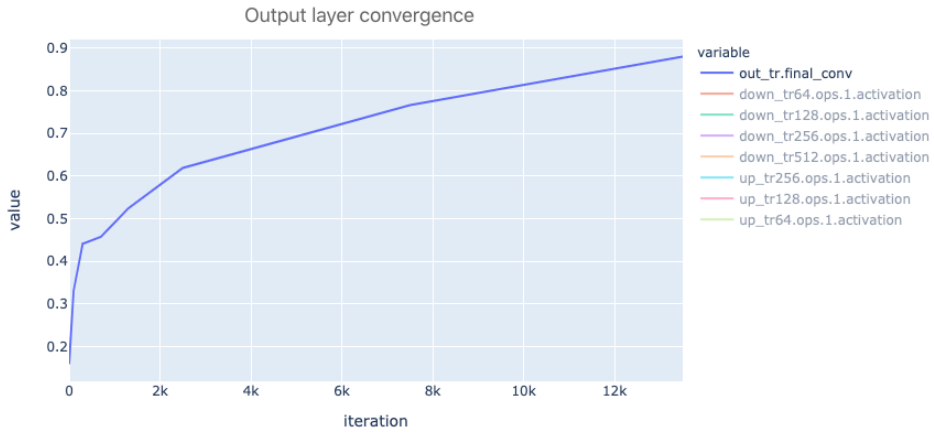

Figure 5.2: Filtering out the output layer convergence

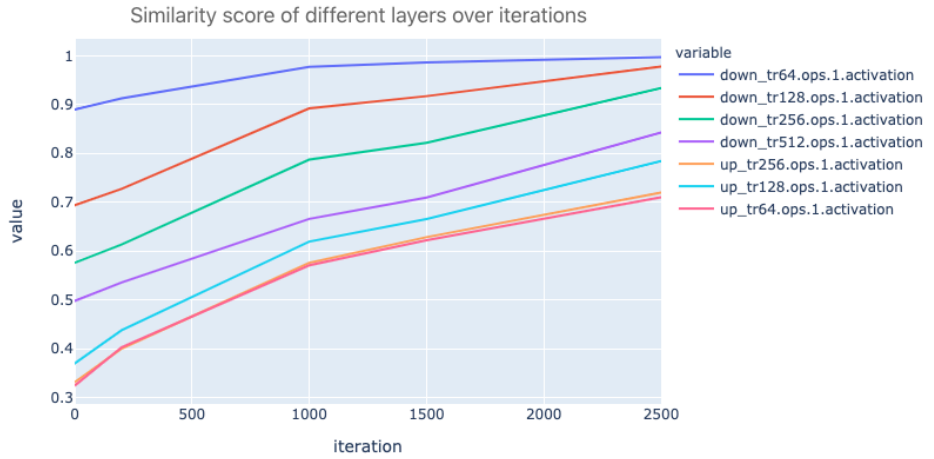

Figure 5.3: CCA similarity score of each layers over iterations during Fine-tuning

Network convergence during Fine-tuning

The question we asked is: **Is the convergence similar to random initialization when we have a pre-trained model?** To test the convergence, we repeated the experiment in the previous section: Starting from a pre-trained model, we iterate through the Dataloader (containing 40 volumes sliced into 2D from the Axis view) and validate every 100 iterations. We saved the model of that epoch if the validation accuracy reported a better performance. Figure 5.3 plots the CCA similarity score of each saved model compared with the converged model.
Comparing the similarity of feature map in the segmentation model, we observed the following behaviour:

- In the Encoder part, lower layers converged faster compared to higher layers. Specifically, down covolution reported higher similarity score from the beginning compared with the similarity score obtained when we trained from scratch

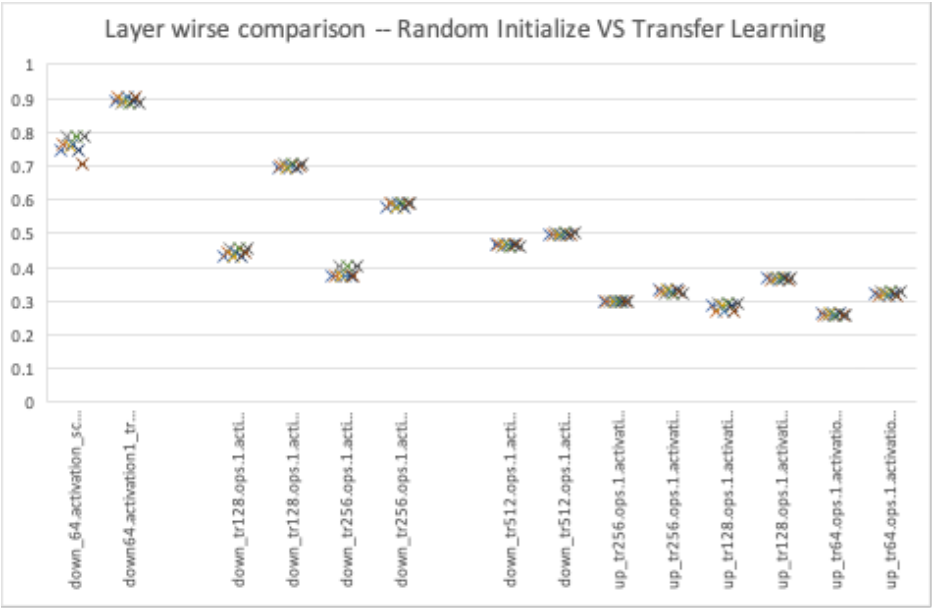

Figure 5.4: Layer-wise Comparison during training

- Layers in the Decoder observed convergence almost at the same time.

Feature space comparison for transfer learning

Another question we want to discuss is: **How much did the model move given the amount of data it observed during the Fine-Tuning?**

A larger amounts of data:
Figure 5.3 shows that, although all layers output seen a higher similarity score as the iteration number increase, Down-Convolution layers in the Encoder moved much less compared to the rest of the model, yielding a slightly better feature reuse during Fine-Tuning, and the feature reuse decreased from lower layers to higher layers. The Decoder, however, gives a much lower similarity score (lower than 0.5) comparing the pre-trained initialization to the converged model. Figure 5.4 compared the layer-wise feature space similarity which showed the similar obeservation.
We then trained the model using only 5 volumed sliced into 2D.....

5.3 With unlabeled data

We then consider leveraged unlablled data because in mid July, MosMed published unlabelled CT volumes. Thus, we continued our experiment to leveraged the unlabeled dataset. We first fine tune a pre-trained model on the annotated dataset, take the encoder of the network. We cropped patches from the unannotated images, assign noisy label to them and train a mean-teacher style network using those cropped patches

5.3.1 Experiment Setup

Apart from the labelled datset described in section 5.2, we downloaded 200 unlabeled volumes and first preprocessed the CT volumed as we described in chapter 3.

5.3.2 Training a coarse 3D segmentation

The purpose of leveraging unlabelled data during training is to improve the generalization of the network so that it generalize better on unseen data. We want to crop more infection area to guide the segmentation model, so we first train a coarse 3D segmentation to generate a 'rough mask' of the image to guide the segmentation.

5.3.3 Transfer learning 2D segmentation

5.3.4 Psuedo Label Assignment – Cosine Similarity in the feature space

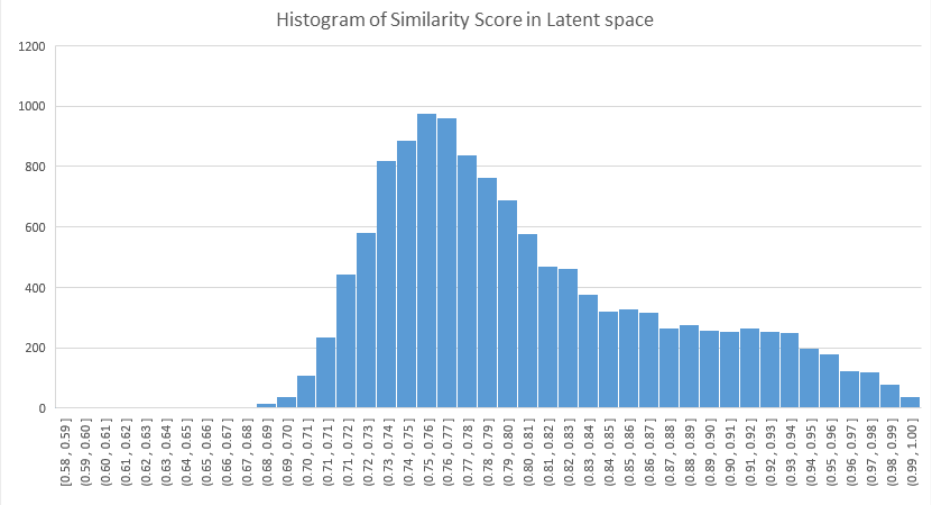

Figure 5.5: Histogram of Cosine Similarity score

Figure 5.5 counts the occurence of similarity score over all sampled unlabeled patches. We took those samples with similarity score between 0.87 and 0.96 (sim $\in$ [0.87, 0.96]), and we assign the label of the annotated samples with highest similarity score. Figure 5.6 showed some examples of noisy labels assigned using this method.

5.3.5 Mean teacher training

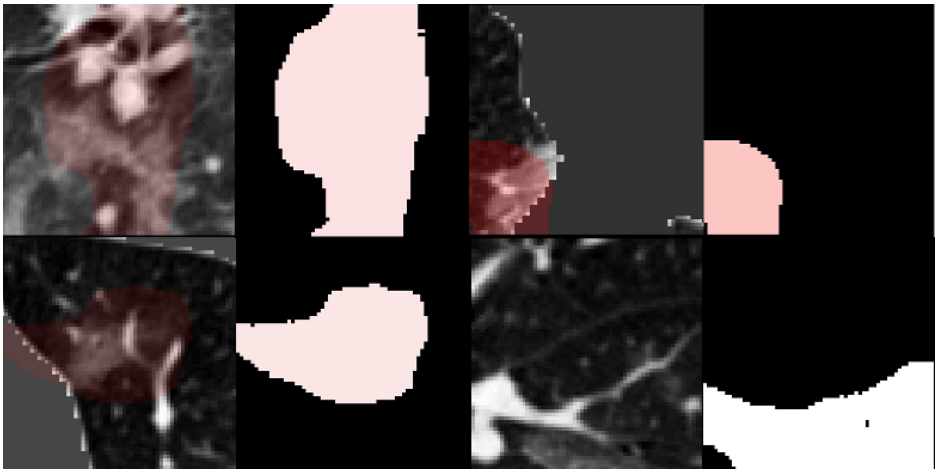

Figure 5.6: Example patches and noisy masks

Chapter 6

Discussion and conclusion

6.1 Conclusion

6.2 Future work

Chapter 7

Appendix A: Ethics Checklist

This project does not involve direct human participant, while uses personal data from BraTs19 competition collected by Center for Biomedical Image Computing & Analytics (CBICA). Furthermore, the CBICA dataset used in this project were directly downloaded from the competition. This project only uses the MRI image offered by the competition while it also offering anonymised data about the age and survival of patients. Moreover, although the dataset is officially public, we made sure that we had all the necessary authorizations to use it, and we strictly follow the BCS code and IET rules of conduct.

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		X
Does your project involve the use of human embryos?		X
Does your project involve the use of human foetal tissues / cells?		X
Section 2: HUMANS		
Does your project involve human participants?		X
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from Human Embryos/Foetuses i.e. Section 1)?		X
Section 4: PROTECTION OF PERSONAL DATA		
Does it involve the collection and/or processing of sensitive personal data (e.g.health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	X	
Does it involve processing of genetic information?		X
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		X
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing datasets?		X
Section 5: ANIMALS		
Does your project involve animals?		X
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		X
If your project involves low and/or lower-middle income countries, are any benefitsharing actions planned?		X
Could the situation in the country put the individuals taking part in the project at risk?		X
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		X
Does your project deal with endangered fauna and/or flora /protected areas?		X
Does your project involve the use of elements that may cause harm to humans, including project staff?		X
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		X

Section 8: DUAL USE		
Does your project have the potential for military applications?		X
Does your project have an exclusive civilian application focus?		X
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		X
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		X
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		X
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		X
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g.privacy, stigmatization, discrimination), if misapplied?		X
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		X
Section 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		X
Will your project use or produce goods or information for which there are data protection, or other legal implications?		X
Section 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		X

Chapter 8

Appendix

Bibliography

[1] Z. Zhou, V. Sodha, M. M. R. Siddiquee, R. Feng, N. Tajbakhsh, M. B. Gotway, and J. Liang, "Models genesis: Generic autodidactic models for 3d medical image analysis." [Online]. Available: http://arxiv.org/abs/1908.06912 pages

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. pages

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation." [Online]. Available: http://arxiv.org/abs/1505.04597 pages

[4] . iek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2016*, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds. Springer International Publishing, vol. 9901, pp. 424–432, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-46723-8_49 pages

[5] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation." [Online]. Available: http://arxiv.org/abs/1606.04797 pages

[6] L. Zhang, X. Wang, D. Yang, T. Sanford, S. Harmon, B. Turkbey, H. Roth, A. Myronenko, D. Xu, and Z. Xu, "When unseen domain generalization is unnecessary? rethinking data augmentation." [Online]. Available: http://arxiv.org/abs/1906.03347 pages

[7] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization." [Online]. Available: http://arxiv.org/abs/1710.09412 pages

[8] E. Panfilov, A. Tiulpin, S. Klein, M. T. Nieminen, and S. Saarakkala, "Improving robustness of deep learning based knee MRI segmentation: Mixup and adversarial domain adaptation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, pp. 450–459. [Online]. Available: https://ieeexplore.ieee.org/document/9022164/ pages

[9] N. Tajbakhsh, L. Jeyaseelan, Q. Li, J. N. Chiang, Z. Wu, and X. Ding, "Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation," vol. 63, p. 101693. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S136184152030058X pages

[10] Z. Li, K. Kamnitsas, and B. Glocker, "Overfitting of neural nets under class imbalance: Analysis and improvements for segmentation." [Online]. Available: http://arxiv.org/abs/1907.10982 pages

[11] X. Ren, L. Zhang, D. Wei, D. Shen, and Q. Wang, "Brain MR image segmentation in small dataset with adversarial defense and task reorganization," in *Machine Learning in Medical Imaging*, H.-I. Suk, M. Liu, P. Yan, and C. Lian, Eds. Springer International Publishing, vol. 11861, pp. 1–8, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-32692-0_1 pages

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples." [Online]. Available: http://arxiv.org/abs/1412.6572 pages

[13] S. Hussein, K. Cao, Q. Song, and U. Bagci, "Risk stratification of lung nodules using 3d CNN-based multi-task learning," vol. 10265, pp. 249–260. [Online]. Available: http://arxiv.org/abs/1704.08797 pages

[14] B. Kaur, P. Lematre, R. Mehta, N. M. Sepahvand, D. Precup, D. Arnold, and T. Arbel, "Improving pathological structure segmentation via transfer learning across diseases," in *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*, Q. Wang, F. Milletari, H. V. Nguyen, S. Albarqouni, M. J. Cardoso, N. Rieke, Z. Xu, K. Kamnitsas, V. Patel, B. Roysam, S. Jiang, K. Zhou, K. Luu, and N. Le, Eds. Springer International Publishing, vol. 11795, pp. 90–98, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-33391-1_11 pages

[15] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation," in *Procedings of the British Machine Vision Conference 2017*. British Machine Vision Association, p. 167. [Online]. Available: http://www.bmva.org/bmvc/2017/papers/paper167/index.html pages

[16] A. G. Roy, S. Siddiqui, S. Plsterl, N. Navab, and C. Wachinger, "'squeeze & excite' guided few-shot segmentation of volumetric images." [Online]. Available: http://arxiv.org/abs/1902.01314 pages

[17] K. Rakelly, E. Shelhamer, T. Darrell, A. Efros, and S. Levine, "Few-shot segmentation propagation with guided networks," p. 10. pages

[18] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." [Online]. Available: http://arxiv.org/abs/1703.01780 pages

[19] J. Ma, “Segmentation loss odyssey.” [Online]. Available: http://arxiv.org/abs/2005.13449 pages

[20] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, “SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability.” [Online]. Available: http://arxiv.org/abs/1706.05806 pages

[21] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, “Transfusion: Understanding transfer learning for medical imaging.” [Online]. Available: http://arxiv.org/abs/1902.07208 pages

[22] J. Hofmanninger, F. Prayer, J. Pan, S. Rohrich, H. Prosch, and G. Langs, “Automatic lung segmentation in routine imaging is a data diversity problem, not a methodology problem.” [Online]. Available: http://arxiv.org/abs/2001.11767 pages