Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Federated Deep Learning for Healthcare Data

Author:
Erik Babu

Supervisor:
Prof. Daniel Rueckert

Second Marker:
Prof. Andrew Davison

June 13, 2020

Abstract

In deep learning, machine learning models typically require large and representative training sets to perform tasks well. If hospitals and clinics aggregate their healthcare data, sufficiently large datasets could be generated to train models to perform highly accurate medical prognosis and diagnosis. However, due to the extremely sensitive nature of medical data, there are strict regulations in place that prevent institutions from collaborating by 'pooling' their data together.

To facilitate collaborative learning on decentralised data without compromising on data privacy, we propose the use of Federated Learning (FL). Showing the viability of this approach could have a significantly positive impact in the medical healthcare space. FL models could potentially uncover patterns between disconnected datasets from several institutions. FL would additionally enable smaller institutions with significantly fewer patients, such as clinics, to also perform accurate diagnosis. This is due to the shared model generalising well, based on the large amounts of data seen from several other institutions.

In this project, we take an existing implementation that achieves state-of-the-art performance when trained on a large chest radiograph dataset (224, 316 images), and redesign it to perform similarly in an FL setting. We implement two FL algorithms, FedAvg and FedProx, and investigate their performance and training overhead when applied to the dataset partitioned between different 'institutions'. Our results show that by employing FL, we are able to achieve within 0.014 AUC (two institutions), 0.008 AUC (five institutions) and 0.014 AUC (ten institutions) of the benchmark model. We conclude that models trained using FL can perform similarly to those trained on aggregated data.

Acknowledgements

First and foremost, I would like to thank Prof. Daniel Rueckert for being a fantastic supervisor throughout the project. During times when I doubted whether the project would be successful, he was extremely understanding and reassuring. Because of his constant guidance, excellent feedback and consistent availability for discussions, we always remained on track.

I would also like to thank Dr. Jonathan Passerat-Palmbach for organising the weekly paper discussions with myself and the other students who undertook Federated Learning projects. He provided additional insights and was always willing to assist us.

Additionally, I am extremely grateful to my family and friends for all their encouragement, love and support, not just during the project, but throughout my degree. I would not have been able to complete this journey without them.

Lastly, I would like to extend my gratitude towards the frontline employees and key workers around the world. Their sacrifices and dedication to keep us protected in these unprecedented times will never be forgotten.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In 2015, a study [1] was conducted on the accuracy of cancer diagnoses. As part of the experiment, 16 testers had to decide whether or not images of breast tissue were cancerous. Independently, the testers correctly assessed 85% of the samples. By pooling the results, i.e. performing majority voting on the independent classifications, the accuracy rate rose to 99%. This experiment was remarkable, not only because of the testers' performance, but also their identity. They were neither oncologists nor pathologists; they were pigeons. Identifying patterns in medical data is not a uniquely human skill.

Because machine learning (ML) algorithms excel at uncovering patterns from data, they have the potential to address problems in healthcare, such as diagnosis [2] and prediction of future health outcomes [3]. However, these algorithms, particularly deep learning approaches, typically require large training sets to achieve good performance. Another obstacle is that labelling medical data requires expert knowledge. Ideally, medical institutions could address these challenge through collaboration i.e. aggregating their anonymised data and annotations to a central location. This would facilitate the creation of sufficiently large datasets, since the healthcare system generates approximately one trillion gigabytes of data each year, and this amount is doubling every two years [4]. However, due to the sensitive nature of healthcare data, there are ethical concerns [5] and data protection regulations such as the General Data Protection Regulation (GDPR) [6] and Health Insurance Portability and Accountability Act (HIPAA) [7] which preclude its sharing, especially among international institutions. Even if it was possible to aggregate all the data at a central location, the significant computational overhead required to process all the data would make the task infeasible.

To address the challenge of utilising large amounts of distributed healthcare datasets in a unified manner while still preserving privacy, we propose the use of Federated Learning (FL); a decentralised solution which "brings" a model to the data, rather than data to a model. FL addresses issues of privacy and data ownership, since the healthcare data never leaves the medical institution.

1.2 Objectives

The primary focus of this project is to develop an FL solution, for the classification of medical imaging data, which performs similarly to its centralised implementation i.e. a model which has access to the aggregated data from every institution.

We will use the CheXpert (Chest Expert) [8] dataset; a collection of 224,316 chest radiographs (2D) of 65,240 patients labeled for the presence of 14 typical chest radiographic observations. Apart from being a medical imaging dataset, CheXpert is also selected because of its size; existing research into the applications of FL on medical imaging data does not focus on datasets of this magnitude. The test set, comprised of 500 studies from 500 unseen patients, is annotated by eight board-certified radiologists. By evaluating our solution against the best available ground truth, we

provide a foundation for the clinical relevance of our results.

From the initial supervisor meetings, the following objectives were agreed:

1. **A centralised model:** This involves training an existing State-Of-The-Art (SOTA) implementation on the entire dataset (representing a central server which has aggregated the data from several institutions). The evaluation of this model serves as the project benchmark.

2. **Institution-specific models:** This entails splitting the dataset into partitions (representing data from each institution), and training a separate model on each partition. The evaluation of these models serve as the baseline.

3. **Models trained using FL:** This requires training a single FL model on each institution's dataset. At the very least, the FL model should outperform the baseline models. This is to justify the use of FL in the first place. For the project to be considered successful, the FL model should perform at nearly the same level as the benchmark model.

4. **Method to compute training overhead metrics:** As FL is a distributed machine learning task, another objective is to report metrics such as communication and computational overhead of training. This is absent from existing research on FL applied to medical imaging.

1.3 Contributions

1. **Implement different FL training techniques:** Instead of utilising an existing FL framework (for reasons discussed in Section 4.1), we write the training procedure ourselves. This includes the implementation of two different FL algorithms, FedAvg and FedProx. More details regarding the algorithms can be found in Section 2.8. We describe their implementation in Sections 4.2 and 4.3.

2. **Investigate the performance of FL applied to two institutions:** Using the algorithms mentioned above, we train FL models on two institutions' data. To simulate institutional data, we split the entire dataset into two non-i.i.d. partitions. It is important to use non-i.i.d. partitions of data to more accurately represent true federated healthcare scenarios. We first experiment on even partitions i.e. each institution contributing 50% of the data. We then experiment on uneven partitions i.e. one institution with 75% of the data, and the other having the remaining 25%. Since this particular experiment involves an imbalance in contributions, we also implement and evaluate the performance of a weighted version of the FedAvg algorithm. We explain the partitioning scheme further in Section 3.5 and analyse the results of these experiments in Sections 5.2 and 5.3.

3. **Experiment with larger number of institutions:** To determine if increasing the number of participating institutions degrades FL model performance, we run a modified version of the experiment above. The only difference is that we partition the dataset between five and ten institutions, each contributing an equal amount of non-i.i.d. data. We analyse the results of these experiments in Sections 5.4 and 5.5.

4. **Report overhead of training FL models:** Although the project does not focus on low-level system and network optimisation, we implement functionality that reports computational and communication overhead of the FL training procedure. This enables our reported metrics to serve as a baseline for any future work on optimisation to be evaluated against. The details of how these metrics are computed, including our modelling assumptions, are explained in Sections 3.6 and 4.5.

5. **Achieve near-benchmark performance with every FL experiment:** Our results show that the models trained using FL achieve similar performance to the benchmark in every set of experiments. This demonstrates the viability of different medical institutions collaborating via FL.

Chapter 2

Background

In this chapter, we provide a brief overview on supervised deep learning i.e. developing predictive models based on both input and output data, followed by a discussion of some applications of computer vision in the real-world. We then survey existing image classification and segmentation algorithms. We conclude the chapter with an analysis of existing state-of-the-art approaches to privacy-preserving deep learning.

2.1 Overview

Deep Learning (DL) is a subset of Machine Learning (ML), which is itself a subset of Artificial Intelligence (AI). Inspired by the human brain, DL algorithms are capable of learning from large amounts of data. DL pipelines typically consist of a neural network, loss function and optimisation method. The neural network builds a mapping from the input to the output. The loss function is a quantitative metric that evaluates how well the algorithm models the data. The goal of the optimisation step is to find the parameters which minimise the loss function.

As increasing amounts of training data are being made available and our ability to perform these computationally intensive tasks is continuously improving, DL techniques have become very successful in computer vision tasks; they achieve near-human performance on image classification [9] and segmentation [10] challenges.

2.2 Real-World Applications of Computer Vision

2.2.1 Retail

In 2018, Amazon opened its first Go store [11] - a concept that enables customers to walk in, pick up their groceries and walk out i.e. without having to pay at a register. Computer vision is used to determine if, and by whom, an item has been picked. The cameras, which track shoppers at all times, ensure they are billed appropriately when leaving the store. Large supermarket chains are currently losing billions of pounds to shoplifters [12]; integrating similar technology could significantly reduce the damage.

2.2.2 Automotive

With advances in DL and computer vision, there has been an emergence of autonomous vehicles on the road. Waymo [13] and Tesla [14] use computer vision for their driver-less software. These algorithms analyse input from 360 degree cameras and other sensors to control the vehicle in different weather conditions, terrains and traffic scenarios. This technology is not only relevant to personal vehicles; it is becoming increasingly applied to logistics [15] and public transport [16].

The World Health Organisation (WHO) report [17] that human error and lack of attention cause the majority of deaths from traffic accidents. They predict that in a decade, such incidents will become the seventh leading cause of death, unless action is taken. The integration of autonomous vehicles on the road will go a long way towards preventing that forecast from becoming a reality.

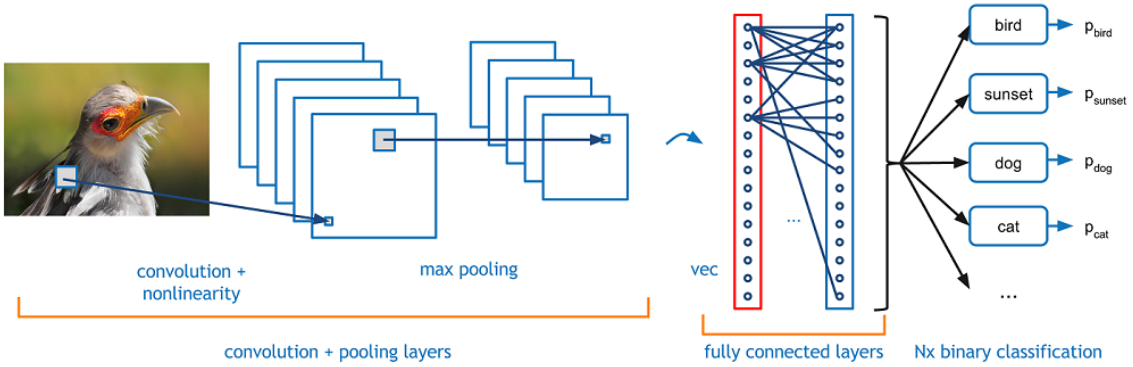


Figure 2.1: Typical CNN architecture [24]

2.2.3 Personal Security

In the last few years it has become possible to unlock mobile phones through facial recognition. This is possible because of computer vision algorithms that are able to accurately identify and authenticate the owner of the phone under different camera angles, variable distances, and contrasting lighting.

Internet of Things (IoT) devices are becoming more prevalent in our day to day lives. There are now smart CCTV surveillance cameras [18] that use computer vision to alert homeowners of any possible threats. These smart-cameras identify movement, detect if it is caused by a human and apply facial recognition to determine if the person is a friend/family of the owner, or an intruder.

2.2.4 Healthcare

This will be the main focus-area of the project. As discussed in Chapter 1, computer vision is being applied to problems in medicine. One such application is health monitoring; Gauss Surgical uses Triton [19], a computer vision-aided platform to monitor blood loss during surgeries.

Another application - medical diagnosis and prognosis - is an area where we are starting to see computer vision algorithms outperform the human experts. Earlier in 2020, Google Health announced that they had developed an AI system which is as good as doctors are at detecting breast cancer [20].

We do not suggest that these computer vision algorithms will replace doctors anytime soon. Instead, our hope is that they will serve as useful assistants, by performing pre-screening or acting as a second opinion. This would enable the medical professionals to deliver healthcare services as efficiently and accurately as possible. Another benefit is that it is easier to train 100 models than 100 doctors; these computer vision algorithms could potentially offer world-class healthcare in countries where access is typically available at a premium.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a form of deep neural network that have significantly contributed to the ability of machines to exceed human performance on imaging challenges. CNNs are similar to multilayer perceptrons (MLPs); they are composed of neurons with learnable weights and biases. The problem with regular MLPs is that they do not scale well for large images. Consider 3D MRI scans, which could have dimensions of 256 x 256 x 12 [21, 22]. In an MLP, a single fully connected neuron in the first hidden layer would therefore have 256 x 256 x 12 = 786,432 weights. As there will most likely be several neurons and layers in the network, the trainable parameters would accumulate quickly and result in overfitting [23].

2.3.1 Structure

Figure 2.1 shows the structure of a CNN used to classify an image. We see that the layers of the CNN arrange neurons in three dimensions. This solution scales well because neurons in a layer are only connected to a small region in the previous layer, unlike in an MLP. CNNs are typically created by stacking the following types of layers:

Input Layer

This layer holds the raw pixel values of the image; with height h, width w and three channels - R, G, B, using Figure 2.1 as an example.
Input Dimension: h x w x 3

Convolution and Activation Layer

The convolutional layer computes the output of the neurons which are only connected to local regions in the input, to produce a feature map. The computation involves "sliding" an n x n kernel/filter over the input and performing element-wise matrix multiplication and summation to get a result. To capture different features, we slide k filters over the input.
Output Dimension: h x w x k

The activation layer applies element-wise activation function to the output of the convolutional layer. This allows the network to model response variables which are calculated using non-linear combinations of the inputs. Below, we describe some common activation functions:

The sigmoid activation function maps inputs in the range (0, 1) and is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The tanh activation function maps inputs to the range (-1, 1) and is defined as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

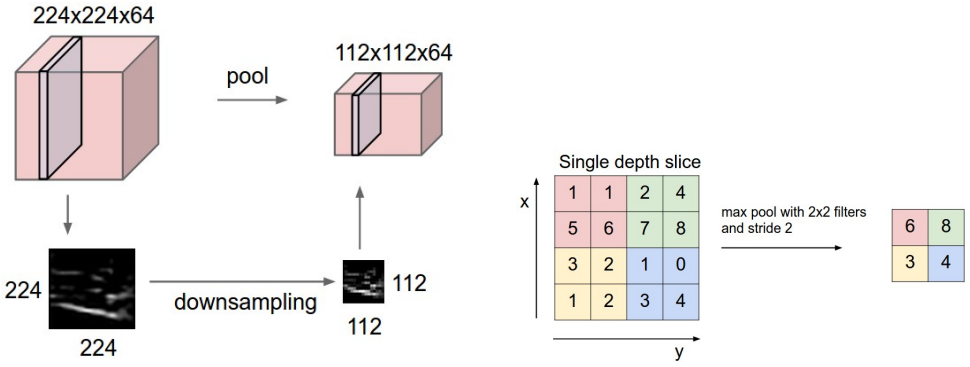
The Rectified Linear Unit (ReLU) is a simple activation function which maps inputs as follows:

$$ReLU(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

The softmax activation function takes a vector x of real numbers and normalises it into a probability distribution proportional to the exponential of the input. Each x_i gets mapped to the range (0, 1) and the sum of the output values $f_i(x)$ is 1.

$$f_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \qquad \text{for } i = 1, ..., J$$

In CNNs, ReLU is typically used as the activation function. Using other activation functions, such as tanh and sigmoid, which map their input to a small output range can introduce the vanishing gradient problem. This is occurs particularly when multiple layers, each using these activation functions, are stacked together. The first layer maps a large input region to a smaller output region, which will be mapped to an even smaller output region by the next layer and so on. This causes the gradients of the network's output, with respect to the parameters in the early layers, to become extremely small. Multiplying n of these small numbers, to compute gradients of the early layers in an n-layer network during back-propagation, results in the value tending to zero i.e. vanishing [25]. Consequently, the network is unable to train effectively.
Output Dimension: h x w x k



(a) Input pooled with 2 x 2 filter, stride 2 (b) Max pooling with 2 x 2 filter, stride 2

Figure 2.2: Pooling layers downsample spatial volume of input [26]

Pooling Layer

This layer is typically placed between successive convolutional layers to gradually reduce the size of the internal representation along the width and height dimensions. This reduces the number of parameters, and therefore computation in the network, and also prevents overfitting. Figure 2.2 shows that when using a 2 x 2 filter "slid" across the input two at a time i.e. stride = 2, only 25% of the activations are preserved. The most common pooling technique is max-pooling; the function selects the MAX value that the sliding window covers. Other pooling functions such as average pooling and L2-norm pooling have been proposed but do not yet achieve comparable performance.
Output Dimension: $\frac{h}{2}$ x $\frac{w}{2}$ x k (for kernel size and stride mentioned above)

Fully Connected Layer

As shown in Figure 2.1, the fully connected input layer takes the output from the layers before it and "flattens" them into a single vector to be fed as input to the next stage. Just as with regular MLPs, each neuron in this layer will be connected to all the neurons in the previous one. This layer will then produce c class scores to be used for classification. In the case of binary classification, sigmoid is typically used as the activation function, and a threshold is applied to determine the class prediction. In the case of multi-class classification, softmax is typically used. The output neuron with the highest response represents the classification.
Output Dimension: 1 x 1 x c

2.3.2 Techniques to Improve Performance

Batch Normalisation

Very deep neural networks involve the composition of several functions or layers. The gradient is used to determine how to update each parameter, under the assumption that the weights in the previous layers to the current layer are fixed. In practice, this assumption does not hold - all layers are changed in an update. This means that the input distribution changes with each step during training (covariate shift) and each intermediate layer is forced to continuously adapt to the changing inputs. Batch normalisation [27] is a technique to reparametrise a model by scaling the output of each layer. This limits internal covariate shift, making the distribution of inputs more stable during the training process. This enables the use of higher learning rates, which leads to faster convergence. Rather than scaling the output to have zero mean and unit variance, batch normalisation allows the network to learn parameters γ and β to scale the output by. This improves the expressive power of the network. The full algorithm is detailed in Figure 2.3.

Attention

In general, attention mechanisms enable neural networks to map the important and relevant features from the input, and assign higher weights to them, thus enhancing accuracy of predictions. Integrating attention modules into neural networks has shown to give state of the art performance in natural language processing tasks such as machine translation [28]. In the context of computer

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 2.3: Batch Normalisation Algorithm [27]

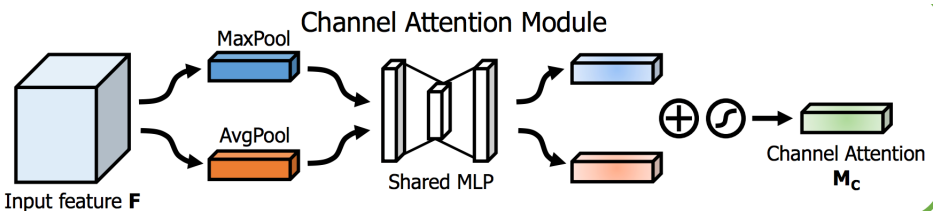


Figure 2.4: Channel Attention Module, adapted from [30]

vision, attention modules equip the neural network with the ability to produce powerful image representations which capture only the properties that are most relevant for the given task, e.g. tumours in cancer diagnosis. In this section, we further discuss the attention mechanisms that we consider using in our models.

Channel Attention Module (CAM) [29]: Given a feature map, CAMs focus on global features. Features are multiplied with a mask of real values between 0 and 1, which are learnt, to appropriately weight individual input channels. This is shown in Figure 2.4.

Spatial Attention Module (SAM) [31]: Given a feature map, SAMs focus on local features. Features are multiplied with a mask of real values between 0 and 1, which are learnt, to appropriately weight patches in the image. Unlike CAMs which output 1-dimensional tensors, SAMs generate masks which have the same dimensions as the feature maps. This is shown in Figure 2.5.

Feature Pyramid Attention (FPA) [32]: Channel-based attention methods are unable to extract multi-scale features effectively. The pyramid structure, shown in Figure 2.6 can extract different scales of feature information. However, the disadvantage of multi-scale representations is that they lack global context. The authors mitigate this by introducing a global average pooling branch, whose output is then combined with the extracted features. Fusing the contextual information from different scales enables better pixel-level attention for high-level feature maps.

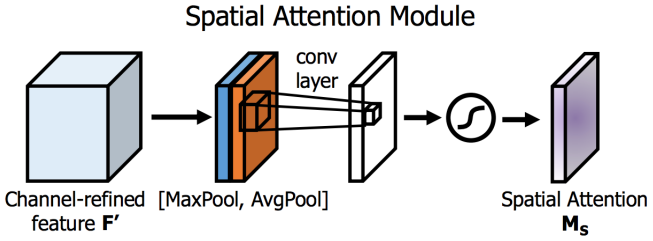


Figure 2.5: Spatial Attention Module, adapted from [30]

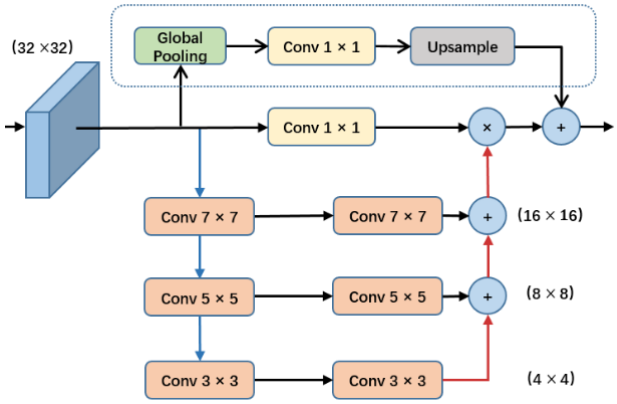


Figure 2.6: Feature Pyramid Attention Module, adapted from [32]

2.4 Image Classification

2.4.1 Overview

Terminology

$\mathbf{x}$: input features i.e. vector representation of images
y : label i.e. meaningful tag telling us what is in the image
Θ : parameters i.e. learnable weights and biases
TP - True Positive : outcome where model correctly predicts the positive class
TN - True Negatives : outcome where model correctly predicts the negative class
FP - False Positives : outcome where model incorrectly predicts the positive class
FN - False Negatives : outcome where model incorrectly predicts the negative class

Goal

Given a training set $T = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ containing m observations, we would like to train a predictor, h_Θ, such that [33]:

$$\forall i[h_\Theta(\mathbf{x}^{(i)}) \approx y^{(i)}]$$

Evaluation Metrics

The **accuracy** is defined as $\frac{TP+TN}{TP+TN+FN+FP}$. In other words, it represents the number of correct classifications divided by the total number of samples in the test set.

The **error rate** is simply defined as $1 -$ accuracy. It represents how often the classifier makes incorrect predictions.

The **precision** is defined as $\frac{TP}{TP+FP}$. It shows the proportion of positive samples that are correctly identified.

The **recall** is defined as $\frac{TP}{TP+FN}$. It shows the proportion of actual positives that are correctly identified.

The **specificity** is defined as $\frac{TN}{TN+FP}$. It shows the proportion of negative samples that are correctly identified.

2.4.2 Popular architecture

The following are CNN-based networks that have achieved state-of-the-art performance on image classification tasks. We consider using these networks to form the building blocks of our FL solutions.

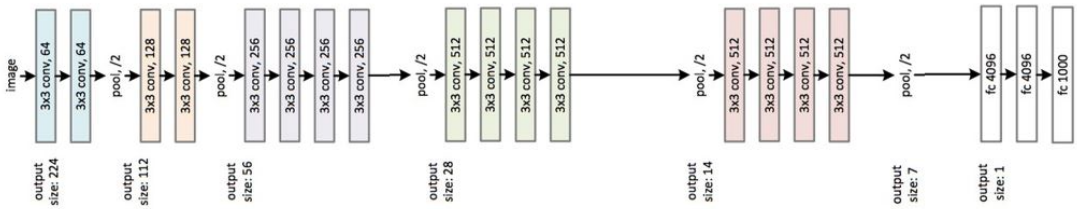
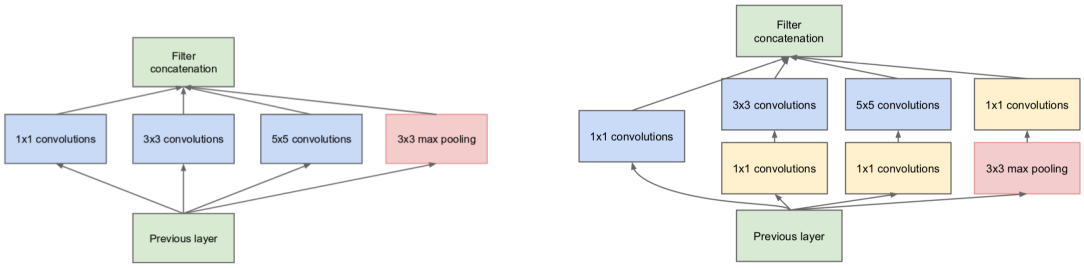


Figure 2.7: VGG-19 architecture, adapted from [35]



(a) Naive version

(b) Dimension reductions

Figure 2.8: Inception Module [37]

VGGNet

Many other models are built on top of VGGNet[34] or use the idea of representing filters with multiple, smaller ones to cover the same effective area. There are several variants of VGGNet, which differ by the total number of layers the network uses. We describe VGG-19, which has 16 convolutional layers and three fully connected layers, as shown in Figure 2.7.

VGGNet was created because CNNs were becoming deeper and using more parameters, leading to longer training times and more overfitting. The solution: ensure that all convolutional kernels are of size 3 x 3. This was unlike the 11 x 11 and 5 x 5 convolutional kernels used by AlexNet [36], the previous state-of-the-art algorithm. This can be done because a 5 x 5 convolutional kernel can be replicated, in terms of receptive field covered, by two stacked 3 x 3 convolutional kernels having stride = 1. Similarly, an 11 x 11 convolutional kernel can be replicated by five stacked 3 x 3 convolutional kernels having stride = 1. As a result, fewer parameters need to be trained: five stacked 3 x 3 convolutional kernels require 5 x 3^2 x $c = 45c$ weights, as opposed to an 11 x 11 convolutional kernel, which requires 11^2 x $c = 121c$ weights.

VGGNet was runner-up in the 2014 ILSVRC [9] classification task, achieving 7.3% top-5 error rate.

GoogLeNet/Inception v1

In image classification tasks, the size of what we are trying to classify can vary considerably. This makes it difficult to decide what kernel size to use; larger ones are preferred for features distributed over large portions of the image, whereas smaller kernels are good at detecting information distributed in a local region. To recognise variable-sized features, we therefore require different-size kernels. Typically, deep CNNs have many stacked layers, using kernels of varied dimensions. This leads to many computationally expensive convolution operations and the risk of overfitting. GoogLeNet [37] mitigates this by implementing multiple kernels of different sizes in the same layer, as shown in Figure 2.8. This makes the network wider, instead of deeper. The 1 x 1 yellow blocks are used for depth reduction i.e. reducing the number of input channels.

Although there are several extensions to the original GoogLeNet implementation [38, 39], they all

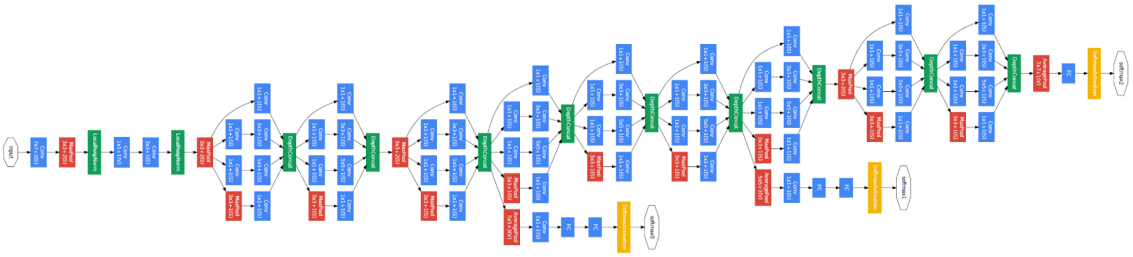


Figure 2.9: GoogLeNet/Inception [37]

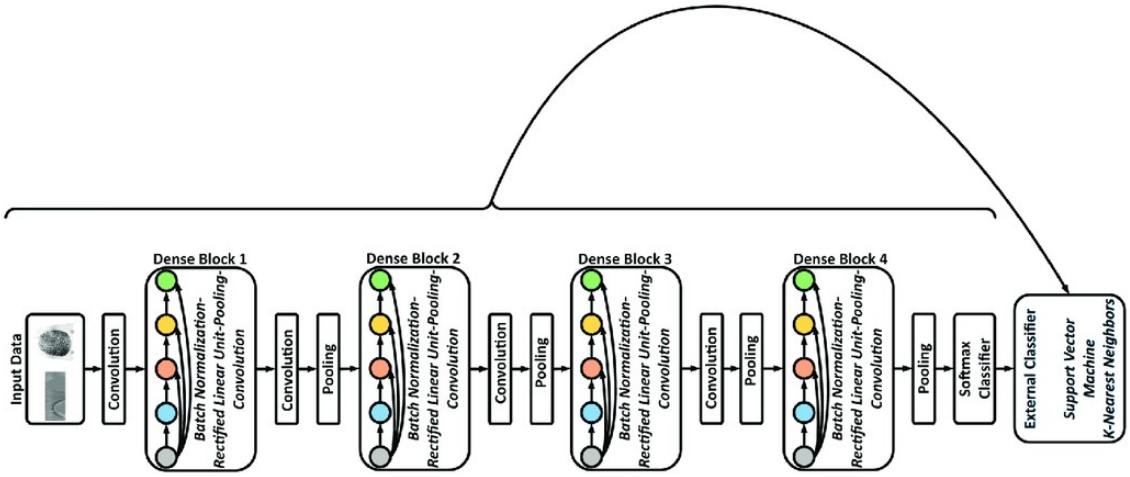


Figure 2.10: DenseNet, adapted from [41]

share a fundamental backbone. At a given level, all extracted features are concatenated and fed into the next inception module. Inception v1 has nine stacked inception modules, containing 22 convolutional layers altogether, as shown in Figure 2.9. To combat the vanishing gradient problem, the authors introduce two auxiliary classifiers; the branches in the network. The total loss function is a weighted sum of the auxiliary loss from the two intermediate inception modules and the real loss. These auxiliary classifiers are only used for training purposes, not during testing or inference.

Instead of stacking fully connected layers after the final convolutional layer, Inception v1 uses a simple global average pooling. This drastically reduces the total number of parameters. The authors are able to remove the fully connected layers without affecting accuracy, due to the depth and width of the network. GoogLeNet won the aforementioned 2014 ILSVRC classification task; it achieves 6.7% top-5 error rate while also being faster than VGGNet.

DenseNet

DenseNet [40] is composed of several dense blocks, as shown in Figure 2.10. In these blocks, each convolutional layer receives additional inputs from all preceding layers. These are concatenated with the layer's own feature maps and forwarded to the convolutional layers ahead of it. This enables the network to be thinner and use fewer channels. Between dense blocks, for the same reason as with GoogLeNet, 1 x 1 convolutions are applied. The outputs from the final dense block are fed into a global average pooling layer and finally, a softmax classifier.

There are several advantages of DenseNet as an image classifier. The errors are propagated to earlier layers more directly, due to the architecture. DenseNet also requires fewer parameters than typical CNNs because it does not have to re-learn feature maps from previous layers. Since each layer receives the previous layers as input, the classifier takes high- and low-level features into account when making predictions, unlike in standard CNNs where the classifier uses the most complex features. DenseNet-264, containing 260 convolutional layers, achieves 5.2% top-5 error rate on ILSVRC.

2.5 Image Segmentation

2.5.1 Overview

Terminology

Ground truth (X): A mask of the image, representing what our model should predict in the segmentation
Predicted truth (Y): A mask of the image, representing the segmentation generated by our model
Upsampling: Technique to increase dimensions of an image representation
Transposed convolution: Upsample an image representation by taking transpose of filter to reverse the convolution

Goal

Given an image as input, the output is a mask, typically of the same dimensions as the input image, in which each pixel is classified to a particular class. In other words, the goal is to perform pixel-level image classification.

2.5.2 U-Net

U-Net is a CNN-based semantic segmentation algorithm that has shown to achieve state-of-the-art performance when applied to medical imaging datasets. Variations [42, 43] of U-Net exist, but they all share the same underlying architecture.

Architecture

The architecture of U-Net is shown in Figure 2.11. It consists of the following two sections, and a layer which mediates between them (bottom-most):

Contraction: Also known as the encoder, this is where regular convolutions and max-pooling are applied. The size of the image representation gradually reduces, as the depth increases. The network has learnt what is in the image but lost information about its position.

Expansion: Also known as the decoder, this is where both transposed and regular convolutions are applied to generate the segmentation mask. The size of the image representation gradually increases, as the depth decreases. The positional information is recovered through upsampling.

Training

One reason that U-Net is appropriate in the context of medical imaging is that it achieves good performance after being trained on only a few images. This is unlike previously discussed Deep CNNs, which require large datasets. This is because of data augmentation techniques applied during the training stage. The loss is defined, for each pixel, such that there is a higher weight at each object's segmentation border. Pixel-wise softmax is applied on the generated image, combined with the cross-entropy loss function; each pixel is classified into one of the classes. This treats the segmentation task as a multi-class classification.

Evaluation Metric

The Dice Coefficient (DC) is a particularly popular metric for evaluating medical imaging segmentation algorithms. It ranges from 0 to 1, with 1 signifying the greatest similarity between predicted and ground truth values. It is defined as:

$$DC = \frac{2 * |X \cap Y|}{|X \cup Y|}$$

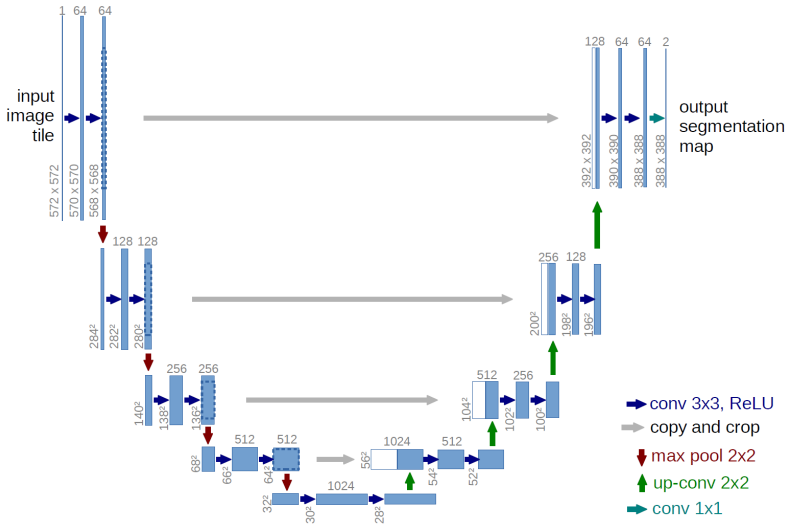


Figure 2.11: U-Net architecture [44]

2.6 Conventional Federated Systems

The concept of federated computing systems are not only specific to applications in machine learning. In the 1990s, a Federated Database System (FDBS) [45] - collection of independent databases co-operating for mutual benefit - was proposed. This does not perform any actual data integration between the independent databases i.e. there is data federation. FDBS provides a uniform interface to allow clients to retrieve data from several, possibly heterogeneous i.e. having different semantics and structure, databases with a single query. A benefit of FDBS is the concept of autonomy; each database owner can still manage data without the FDBS. The concept of autonomy is present in FL as well; participants should be able to associate or disassociate themselves from a network, and participate in more than one FL system. FDBS has some bottlenecks though. Performing schema matching to create a global view of the databases does not scale well with the number of attributes being pair-wise mapped. There were also issues with concurrency control, specifically ensuring global serialisability of transactions. However, this was been addressed with the introduction of commitment ordering.

In the past decade, due to the emergence of cloud computing, there have been studies conducted on federated clouds (FC) [46]. In an FC, multiple external and internal cloud computing services are deployed and managed. They give clients the option to select the best cloud services provider in terms of flexibility, cost and availability, to meet their requirements. As a result, applications run in the most appropriate infrastructure environments. FC also allows an enterprise to distribute workloads around the globe and move data between disparate networks. A drawback of FC is the difficulty in connecting a client with a given external cloud provider; each provider has their own network addressing scheme. The FC has to provide a uniform way for customers to access cloud services without the need for reconfiguration when using resources from different service providers.

2.7 Motivation for Federated Learning

2.7.1 Problem Formulation

FL problems typically involve using data from several remote participants to learn a single, global model. A requirement is that a participant's data must be stored and processed locally i.e. not transmitted to any other participants or a central server. Formally, the goal is to minimise this distributed optimisation problem [47]:

$$\min_{w} F(w), \text{ where } F(w) := \sum_{k=1}^{m} p_k F_k(w)$$

m is the total number of participants. p_k is the relative impact of each participant - this is typically either evenly shared i.e. $p_k = \frac{1}{m}$ or according to the proportion of data provided i.e. $p_k = \frac{n_k}{n}$ where n is the total number of samples and n_k is the number of local samples. We require $\forall k[p_k \geq 0]$ and $\sum_k p_k = 1$. F_k is the local objective function for the k-th participant. It typically represents the empirical risk over the data at that participant. The overall goal of the federated learning algorithm is to find a hypothesis for which the risk, i.e. uncertainty on performance, is minimal. We measure empirical risk on a known set of training data because we do not know the true distribution of the data that the algorithm will work on during inference.

We assume there are m different participants, each denoted by T_i where $i \in [1, m]$. In a non-federated environment, every participant T_i uses only its local data D_i to train a model M_i, having performance P_i. In a federated environment, all participants train a model M_f, having performance P_f. A valid federated system requires that $\exists i \ [P_f > P_i]$ i.e. even though some participants may not receive a better model via FL, there is at least one which achieves better utility [48].

2.7.2 Why Federated Learning?

Cellphones, IoT devices and autonomous vehicles are some of the modern distributed networks creating large volumes of data daily. As there are increasing concerns over transmitting private information over these networks, and the computational power of these devices is continuously improving, an attractive solution is to store data locally and perform training of statistical models directly at the source (edge computing [49]) or close to where it was created (fog computing [50]), instead of at a central server. Recent works have also considered training statistical models centrally but serving and storing them locally [51]. However, such proposals still involve sending data to a central server, and are not feasible when regulations preclude data sharing. FL solutions address these critical issues of data privacy and ownership in a way that can be scalable to a large number of participants, while achieving excellent model performance.

Medical institutions are ideal candidates for FL: they have large volumes of sensitive patient data and the resources to perform on-site computation. The application of FL to medical data could have significant positive impact in the healthcare space. For institutions which contribute significant amounts of data, such as large hospitals, predictive models could be developed that are fine-tuned [52] to their patients. Smaller institutions, such as clinics, could benefit by receiving a model that generalises well enough, based on the data seen from other contributing institutions, to perform diagnosis at an improved accuracy.

2.7.3 FL Systems in Production

Federated Learning Systems (FLS) have already been deployed by major service providers and are playing a crucial part in supporting privacy-sensitive applications. In this section, we discuss the their impact and how to incentivise participants to adopt FL.

Predictive Keyboards: One of the first use cases of federated learning was implemented [53] by Google for Gboard - their predictive keyboard. Due to high regulatory pressure and data overhead, it was no longer feasible to upload all users' text messages centrally to train their word guessing predictive algorithm. Despite the limited memory and computing power of smartphones, Google developed a practical and scalable privacy-preserving FLS.

Healthcare: As previously discussed, training a model to solve complex medical problems requires a lot of data from diverse institutions. These institutions are extremely strict about maintaining control of their sensitive patient data. Assisted by highly traceable technologies, such as distributed ledgers [54], FL has enabled researchers to train predictive models on large amounts of sensitive data in a way that ensures it never leaves the medical institution [52, 55, 56, 57]. These algorithms are able to model heterogeneous data from both pharmaceutical companies and medical institutions.

Transport industry: Due to the large number of self-driving vehicles we will eventually see on the road, and the need for them to quickly respond to real world situations, it is only a matter of time before they adopt FL solutions. FL would enable these systems to to reduce the amount of data

transferred, and accelerate the learning process. An additional benefit of FL over traditional cloud approaches is that they do not provide automakers with real-time access to people's whereabouts, thereby preventing potential safety risks and privacy violations.

Incentive

In real-world applications, parties need to be persuaded to become participants in the FLS. Organisations and companies are typically incentivised to adopt FLS due to regulations. Users of a service are not required to provide their data to improve the models, and must therefore be incentivised to participate. Using the example of Gboard from above, Google cannot prevent users who do not provide data from using the keyboard. However, they can incentivise those who agree to participate by promising a higher accuracy of word prediction. Consequently, more users are motivated to participate, and the performance of the overall model improves.

However, it is still a challenge to design reasonable incentive mechanisms. These are pivotal to the success of the FLS. Apart from the Gboard incentive mechanism of providing better accuracy in exchange for participation, other incentive designs have been proposed to attract participants with high-quality data for FL by offering contract-based rewards [58].

2.8 Federated Learning Algorithms

2.8.1 Terminology

K : total number of participants
n_k : number of training samples at participant k
C : fraction of participants selected for each round
$\ell(w, b)$: loss function for weights w and batch b
w_k^t : model's weight vector for participant k during federated round t

Hyperparameters

B : local minibatch size
E : number of local epochs
η : learning rate
T : number of federated communication rounds

2.8.2 Federated Stochastic Gradient Descent (FedSGD)

In a single step of SGD, gradients are computed using a random subset of the entire dataset. FedSGD is similar, but selects a random subset C of all participants and uses all their data to compute gradients. The central server averages the gradients proportionally to n_k and uses the result to perform gradient descent [59].

2.8.3 Federated Averaging (FedAVG)

Federated Averaging [59] is similar to FedSGD. However, local nodes perform more than one batch update on local data. They also exchange updated weights, instead of gradients, with the central server. This is detailed by the algorithm in Figure 2.12a. The authors show that models trained using FedAvg on the popular CIFAR-10 imaging dataset are able to achieve better accuracy and convergence in significantly fewer communication rounds - this is seen in Figure 2.12b.

In each round of FL, the server selects a random subset S_t of participants. It then sends w_t to all selected participants. Each participant (client k) in S_t updates w_t for E epochs of SGD, with learning rate ℓ to obtain w_k^{t+1}. They send these updated weights back to the server, where they are averaged.

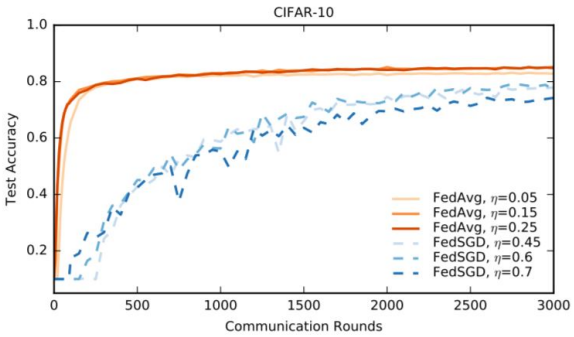
Equal Averaging

The updates from each participant are equally weighted during the aggregation at the server i.e. $w_{t+1} = \frac{1}{K} \sum_{k=1}^{K} w_{t+1}^k$

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **in parallel do**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in \mathcal{B}$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

(a) FedAvg Algorithm



(b) FedAvg vs FedSGD performance on CIFAR-10

Figure 2.12: FedAvg algorithm and its performance benefits [59]

Weighted Averaging

A participant's updates are weighted by the proportion of samples they provide, as shown in Figure 2.12a.

2.8.4 FedProx

FedProx [60] is a generalisation and reparametrisation of FedAvg. The difference is that during training on local nodes, a regularisation term, shown below, is added to the loss function.

$$\text{Loss} = \text{Loss} + \frac{\mu}{2}||w - w_k||^2$$

The μ term is a hyperparameter. The internal term represents the L2-norm of the weights of the locally trained model w_k subtracted from the weights of the global model w. The minor modifications made to the algorithm enable the solution to provide convergence guarantees in environments of non-IID data distributed across the network i.e. displaying statistical heterogeneity. This makes it a more robust method than FedAvg for FL. FedProx does not assume a uniform amount of work will be done by each participant, i.e. it allows for incorporating variable amounts of local work resulting from any system heterogeneity. The authors show that FedProx achieves more stable and accurate convergence in highly heterogeneous settings, improving test accuracy by 22% on average.

2.9 Challenges of FLS

2.9.1 Expensive Communication

Communication overhead is a typical concern in any distributed system. FLS are typically composed of a large number of participants, sometimes in the magnitude of millions [53]. However, for the scope of this project, we assume that only ten participants will be involved in a given round of FL. For large scale FLS to succeed, the systems have to employ communication-efficient methods during the training process. Communication overhead can be reduced by:

Reducing number of federated rounds: As discussed in Section 2.8.3, the use of FedAVG instead of SGD enables convergence in significantly fewer communication rounds.

Reducing the size of transmitted messages: Model compression techniques [61, 62] have been applied to reduce the size of communicated messages at each round. These have still provided convergence guarantees in the presence of non-IID data. These solutions do not take low participation into account; for this project we assume full participation from all parties.

2.9.2 Systems Heterogeneity

In a typical FLS, the storage, communication and computational capabilities of each participant may vary. Participants may also be unreliable and drop out of the network [53], leading to possible biases in the global model. Ideally, an FLS should be robust to potential drop outs. It also should not fully depend on any single party. For the scope of this project, we assume that the medical institutions have similarly high-performing hardware, storage and network capabilities.

2.9.3 Statistical Heterogeneity

In the context of medical data, different hospitals will have different distributions of patient records. This could be due to geography; people in drier countries may suffer fewer cases of pneumonia than those that experience large amounts of rain. As previously discussed, medical institutions have a lot to gain from FLS. If one institution has representative data for one task, and another institution has representative data for a different task, if the institutions agree to participate in FL, they both can potentially receive models capable of improving performance on both tasks. If different sized medical institutions, such as hospitals and clinics, are participating in the FLS, it is likely that the number of data points across each participant will vary. As mentioned in Section 2.8.4, FedAvg diverges when data is not identically distributed across network participants.

The global diagnostic imaging market is dominated by three companies [63]; Siemens (23.2%), General Electric (22.2%) and Philips (19.7%). The imaging devices will produce different scans depending on the manufacturer [64]. For example, Magnetic Resonance Imaging (MRI) signals are not recorded in absolute values, resulting in different intensities for a given contrast, depending on the platform used for acquisition. This introduces further statistical heterogeneity into the FLS. However, we will use a maximally homogenised medical imaging dataset, i.e. one that has been normalised and interpolated to standard resolutions. This facilitates comparison of different algorithms, rather than the efficacy of data pre-processing techniques.

2.9.4 Privacy Concerns

By not transporting raw data, FLS already provide some form of data protection. However, model updates can still leak sensitive information to a third party or the central server. In Section 2.10, we discuss potential attack vectors and privacy-preserving solutions.

2.10 Privacy Mechanisms

For privacy-preserving algorithms to be feasibly incorporated into FLS, they must be efficient with respect to computational and communication overhead. Of significant importance, they should also not overly compromise an algorithm's inference capabilities. These privacy-preserving methods can typically be grouped into the following categories:

Global Privacy: Requires that the model updates generated at each round are private to all untrusted third parties, other than the central server.

Local Privacy: Requires that the updates generated at each round are also private to the central server.

In the rest of this section, we discuss potential attack vectors to FLS, and popular methods of mitigating them. The incorporation of formal privacy-preserving guarantees typically comes at the cost of model accuracy. It is important to find an appropriate trade-off based on nature of the data and requirements of the application.

2.10.1 FLS Attack Vectors

The incorporation of formal guarantees is beyond the scope of this project. In this section we explore the types of attacks that an FLS is generally susceptible to. However, for simplicity, we assume that the aggregator is the only component in the FLS that can compromise the privacy of the other FLS participants, i.e. other parties are honest participants.

Membership inference and model inversion

These can happen when the aggregator is honest-but-curious i.e. follows the protocol correctly but also attempts to infer as much as possible about the data of the parties communicating with it. In a membership inference attack, the attacker can determine if a given individual is present in the training data of an ML model [65]. Unlike model inversion, no additional personal data is learnt about the individual. However, in medical contexts, this information could still potentially be sensitive; it may be possible to determine if an individual is a patient at a particular hospital.

In a model inversion attack, the output of a model is applied to some hidden input, such as an adversarial model, to infer some if its features. In the case of facial recognition systems, a model inversion attack could construct an artificial image of a person. This can be done by using an average of that person's images, which are provided during the training phase. The generated image is not a specific image from the training dataset, unlike with membership inference. Previous work [65] has shown that it is typically difficult to perform such attacks on CNNs. However, Generative Adversarial Networks (GANs) have been used recently [66] to effectively extract such information in collaborative settings. Despite this, it has been shown [67] that the attack has a severe limitation; it is more of a threat when there are only two parties involved. The attack performance significantly drops when the number of participants increases.

Poisoning attacks

This is an active adversarial attack where a malicious participant intentionally "poisons" the training phase in attempt to cause targeted misclassification of the global model [68]. The attacker - a participant in a federated round - controls the local data, training procedure, hyperparameters and the weights submitted to the server for aggregation [69]. In general, this type of attack is not very effective on large-scale FLS. Due to the randomness in choice of participants, the malicious participant is rarely selected for a federated round if there is a large number of them. Updates from all participants are averaged, which could lead to the malicious update having negligible effect on the global model. If the aggregator detects a significant degradation in global model performance after a federated round, it can kick the malicious participant out of the network. In this project, we assume that there are no such adversarial medical institutions.

Byzantine faults

In distributed systems, components may arbitrarily fail or deviate from the protocol without detection; this can lead to misleading results. Existing research [70] shows that it is possible to devise byzantine-resilient algorithms that guarantee convergence for distributed SGD. However, if the FLS uses a central trusted aggregator, this becomes a single point of failure. To keep this project simple, we make the weak assumption that there will be no byzantine faults.

2.10.2 ε-Differential Privacy

This is the most widely used approach in privacy-preserving collaborative machine learning. This is due to the strong theoretical guarantees it provides, the simplicity of the algorithm, and the small systems overhead it incurs. A randomised algorithm M is ε-differentially private [71] if for all $S \subseteq \text{Range}(M)$, when applied to two datasets $D1$ and $D2$ which differ by a single element:

$$\Pr[A(D_1) \in S] \leq e^{\varepsilon} \text{ x } \Pr[A(D_2) \in S]$$

It works by adding sufficiently large, typically Laplace, noise to the result of a computation to hide an individual contribution. This prevents inference about whether a particular sample is used in the training phase. The amount of noise added is dependent on the ε value. Gradients are usually clipped before applying the noise mechanism. Intuitively, adding more noise improves privacy, but can degrade accuracy significantly. This has also proven to be the case when incorporated in federated learning environments, especially when there are a large number of parties, each with limited amounts of data [72, 56].

2.10.3 Secure Multi-Party Computation (SMPC)

With SMPC, two or more participants collaboratively compute an agreed-upon function over a network, on some private input provided by each participant [73]. The output is decrypted by one or more participants. SMPC is a lossless method, and retains original accuracy with high privacy guarantees. However there are some disadvantages; it incurs a large communication overhead. Additionally, even though the inputs from one party cannot be derived by another, information can still be inferred from the decrypted output. Hence, SMPC is vulnerable to inference. Alternative approaches [72] have been proposed to utilise both DP and SMPC to balance their trade-offs and provide stronger privacy guarantees. This has shown to be a scalable approach, as it enables the reduction of injected noise when the number of participants increases, without sacrificing privacy or accuracy.

2.10.4 Fully Homomorphic Encryption

This is a method of securing the learning process by computing on encrypted data [73]; inputs, outputs and intermediate values in FHE are always encrypted. The main benefits of FHE is that it supports arbitrary computations and enables privacy-preserving computation in the presence of an untrusted server. It also involves a much lower communication overhead than SMPC. However, FHE incurs a significantly higher computational overhead. Frameworks such as Zama [74] are being developed to enable fast and accurate inference over encrypted data, while minimising the performance overhead. A drawback of FHE is that it has so far only shown to be applicable in FL when learning linear models [75, 76].

2.10.5 Trusted Execution Environment (TEE)

Distributed and collaborative learning designs are vulnerable to data poisoning and backdoor attacks. In FLS, there is typically a limited sense of accountability; it is difficult to determine which participants are contributors of "bad" training data. A TEE, such as IntelSGX, ensures that computation can take place in a way that guarantees integrity and confidentiality of data. This is done through the provision of a hardware-isolated processing environment [77]. TEEs enable remote attestation; a method by which a participant authenticates its hardware and software to remote participants, i.e. a central server. In decentralised systems, this feature can be utilised to identify corrupt or malicious participants. The drawback of using a TEE is the significant execution overhead, due to the numerous context switches between the TEE and the regular processing environment [78].

CALTRAIN [79] is a TEE-based multi-party collaborative learning system that is able to achieve data confidentiality and model accountability, while overcoming the capacity and performance constraints of secure enclaves. The authors have achieved the same prediction accuracy compared to deep learning models trained in non-protected environments. CALTRAIN supports accountability by maintaining secure links between training instances and the participants that contribute them; the system is able to identify malicious/compromised participants that impact the inference capabilities of the global model.

2.11 Collaborative Learning Applied to Medical Imaging

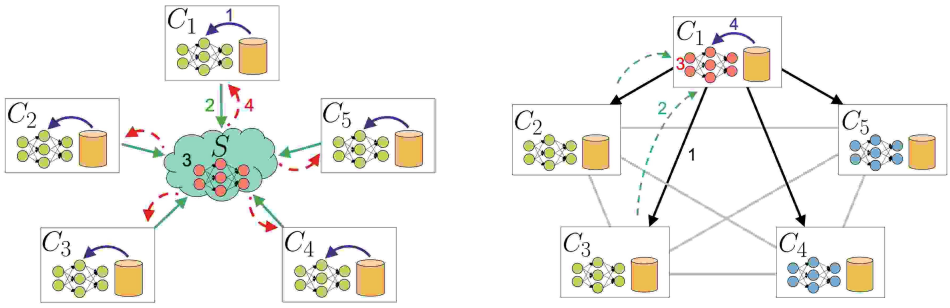
2.11.1 FL vs IIL vs CIIL

In [55], the authors compare the performance of U-Net on the BraTS [80, 81, 82] dataset (further discussed in Section 6.2.5), using three different collaborative learning approaches:

1. A typical FLS, as shown in Figure 2.13a.

2. Institutional Incremental Learning (IIL) - Each institution sequentially trains a model, only once, in any way it chooses.

3. Cyclic Institutional Incremental Learning (CIIL) - Effectively repeats IIL, with each institution training the model for a fixed number of epochs, before passing it to the next institution.



(a) FL with server

(b) BrainTorrent: P2P serverless FL

Figure 2.13: Overview of FL process [57]

IIL and CIIL are completely decentralised approaches; i.e. there is no central aggregator. IIL uses significantly less communication bandwidth than FL because each institution only sends the model it has trained, and receives the initial and final models. However, the IIL technique suffers from "catastrophic forgetting", i.e. the model forgets previously learnt patterns when new data from an institution replaces data from the previous one. As a result, it is the worst performing approach.

CIIL mitigates the issue of catastrophic forgetting by fixing the number of epochs at each institution. However, FL is still a more feasible approach as it effectively does the same thing, but in parallel. This means that FL is better for scaling to a larger number of institutions. Though they both perform similarly on the segmentation task, FL is much more stable; CIIL results show significant variance.

The authors achieve 99% of the centralised model's performance using an FLS. However, a weakness of their study is that they do not evaluate how the model's performance degrades when formal privacy guarantees are incorporated.

2.11.2 FL with ε-DP

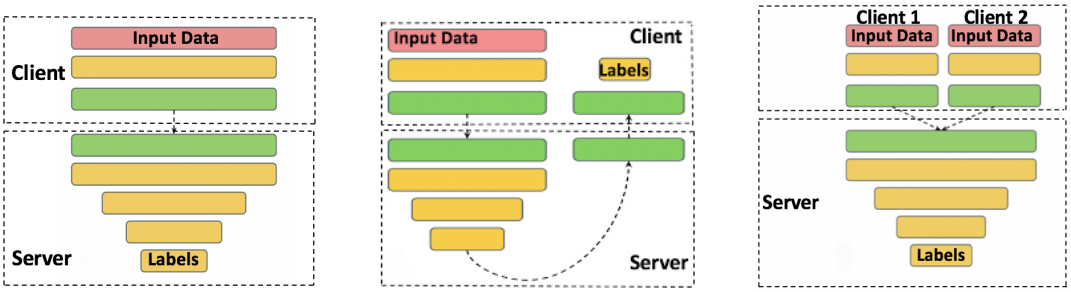
In 2019, a study [56] was conducted by researchers from King's College London, in partnership with NVIDIA, to investigate the feasibility of applying DP techniques to protect patient data in an FLS. Their solution is evaluated on the BraTS dataset, and the general architecture of the system is shown in Figure 2.13a.

At the client-side, a momentum-based SGD is used as the optimiser. At the end of each federated round, the participants' momentum values are reset. When performing the reset operation, the authors observe slightly faster convergence, but similar model performance, compared to when it is not reset. For privacy, a selective parameter update is adopted; to limit the amount of information shared by a participant, only a fraction of the weights - those that are greater than a threshold - are passed to the next stage, where the values are clipped to lie within a fixed range. Before being sent to the server for aggregation, the DP module adds noise to the clipped values via a Laplace mechanism. At the server-side, FedAvg (using weighted averaging) is performed.

The evaluation shows that sharing partial updates causes a noticeable decrease in model performance. Surprisingly, gradient clipping does not affect the convergence speed of the model, and the model performance decrease is almost negligible. The authors show that by sharing a partial model, with reasonable DP guarantees applied to the updates, the performance degradation is reasonable; $DC \approx 0.80$, compared to the FL model with no formal-privacy guarantees having $DC \approx 0.85$. This is a justifiable trade-off for the increased privacy of the healthcare data.

2.11.3 Serverless FL

We have previously only considered FLS with a trusted central server that performs aggregation. With BrainTorrent [52], the authors propose a completely decentralised approach where the medical institutions communicate directly among themselves.



(a) Vanilla split learning

(b) Split learning without label sharing

(c) Split learning for vertically partitioned data

Figure 2.14: SplitNN configurations [57]

Since there is no central server, a new strategy is proposed to coordinate training. All participants are connected directly in a peer-to-peer network, as shown in Figure 2.13b. All N participants maintain a vector $v \in \mathbb{N}^N$ containing their own version and the last versions of models used to create their current models. At the beginning, $v = \overrightarrow{0}$ for all participants. Every time it initiates a training process, it increments its own version number in v. Training at a single participant works as follows:

1. Each participant is locally trained in parallel for a few iterations.
2. A random participant C_i initiates the training process. It asks for the latest models from the other participants to generate v_{new}.
3. All clients C_j with updates (C2 and C3 in Figure) send their weights and number of training samples to C_i.
4. All models sent from the clients C_j are merged with C_i's current model to a single one, using weighted averaging.

While this is an interesting concept, the evaluation itself is particularly weak. The authors claim, using statistically insignificant results, that BrainTorrent outperforms traditional server-based FL approaches. They also never evaluate the performance overhead or how formal privacy-guarantees can be incorporated into this fully decentralised solution.

2.11.4 FL vs Split Learning

The proposed configurations [57] of SplitNN [83], shown in Figure 2.14, facilitate:

- The collaborating institutions needing to perform multiple tasks.
- Performing learning without shared labels.
- Collaborative learning when institutions hold different modalities of patient data.

In the simple vanilla configuration for split learning (Figure 2.14a), each medical institution trains a partial deep network up to a specific layer - the cut layer. The output from this layer is transmitted to a server which completes the training procedure. The gradients are propagated from the last layer of the server to the cut layer. Only the gradients at the cut layer are forwarded back to the medical institutions. This enables a split learning network to be trained without sharing raw data.

In the variant without label sharing (Figure 2.14b), a U-shaped configuration is devised to prevent institutions needing to transmit labels to the server. The output from the final layer of the server is sent back to the institution. This enables the clients to generate the gradients without sharing the corresponding labels. This is ideal in scenarios where labels include highly sensitive information.

The final configuration (Figure 2.14c) has been proposed to enable split learning on vertically partitioned data. Two institutions with overlapping patients, containing different modalities of patient data, train partial models until the cut layers. The output at the cut layer from both institutions is then combined and sent to the central server which completes the training process. A real-world example of this is radiology centres collaborating with pathology test centres, and a server, to perform diagnosis.

The authors show that their SplitNN modifications use significantly lower computational resources and communication bandwidth than FL approaches, while achieving higher accuracy. However, the authors do not address the incorporation of formal-privacy guarantees to the configurations. Additionally, the authors claim that this is "split learning for health" but do not evaluate any of their proposed configurations on medical datasets.

Chapter 3

Benchmark and Baseline Models

This chapter focuses on the work required to develop and train the benchmark (centralised) and baseline (institutional) models that our FL approaches will be evaluated against. We outline the tools and libraries that are crucial to our implementation in Section 3.1. The dataset used to train and evaluate the different approaches, CheXpert, is described in Section 3.2. In Sections 3.3 and 3.4, we detail the implementations of the centralised and institutional models respectively. We then discuss our experimental methodology and address challenges faced during implementation, as well as our strategy to mitigate them, in Section 3.5. Finally, Section 3.6 elaborates on how the computational and communication overhead is measured during training. The evaluation of these models is located in Section 5.

3.1 Key Tools and Libraries

The different approaches, overhead calculation, and model evaluation are implemented in Python 3. Python is a popular choice for implementing industry standard machine learning solutions because of its simplicity, large user community, and vast collection of ML libraries. The reasons for using PyTorch [84] instead of a popular alternative such as TensorFlow [85] are better understood when considered in the context of our FL implementation. We discuss this further in Section 4.1.

The reference implementation[1] we base our FL solution on uses the OpenCV [86] library to load the chest radiographs and perform transformations/image augmentations. Additionally, we utilise Jupyter Notebooks [87] throughout the project. These are used to perform the quantitative sampling of the full dataset, create the baseline data partitions, and visualise the distribution of meta-data and labels. This is further explained in Section 3.5. The benefit of using Jupyter Notebooks for these tasks, as opposed to a python script, is that they make it easier to visualise results and rapidly prototype different ideas.

We initially considered using the LEAF [88] benchmarking framework to measure the training overhead of our models. However, this became infeasible when we committed to using PyTorch - LEAF is only compatible with TensorFlow models. Instead, we use the THOP: PyTorch-OpCounter [89] performance profiling library to measure the computational overhead of training the models. This is further discussed in Section 3.6.1. We could not find a library which met our requirements for measuring the communication overhead of training. Therefore, we implement this functionality, using our own modelling assumptions. We detail this further in Sections 3.6.2 and 4.5.

3.2 Dataset

The CheXpert dataset contains 224, 316 multi-view chest radiographs from 65, 240 patients, labelled for the presence of 14 typical chest radiographic observations. For every scan, each of the 14 labels are classified as positive, negative or uncertain. The statistics are shown in Table 3.1. The authors suggest the following different approaches to deal with the uncertainty labels during training:

[1] `https://github.com/jfhealthcare/Chexpert`

Pathology	Positive	Uncertain	Negative
No Finding	16627	0	171014
Enlarged Cardiomegaly	9020	10148	168473
Cardiomegaly	23002	6597	158042
Lung Lesion	6856	1071	179714
Lung Opacity	92669	4341	90631
Edema	48905	11571	127165
Consolidation	12730	23976	150935
Pneumonia	4576	15658	167407
Atelectasis	29333	29377	128931
Pneumothorax	17313	2663	167665
Pleural Effusion	76696	9419	102526
Pleural Other	2441	1771	183429
Fracture	7270	484	179887
Support Devices	105831	898	80912

Table 3.1: Distribution of observations in dataset, adapted from [8]

- **U-Ignore**: Ignore uncertainty labels when training
- **U-Zeros**: Treat all uncertainty instances as negative
- **U-Ones**: Treat all uncertainty instances as positive
- **U-MultiClass**: Treat uncertainty as a separate class
- **U-SelfTrained**: Use the U-Ignore model to make predictions on uncertain observations. Replace the uncertainty labels with those predictions and then train a model using the updated labels

The task of the classifier is to take a frontal or lateral chest scan as input and predict the probability of a patient displaying the different observations, as shown in Figure 3.1. The authors focus on the evaluation of five observations: Atelectasis, Cardiomegaly, Consolidation, Edema and Pleural Effusion. These are also known as the competition tasks and were selected based on their clinical importance and prevalence. For this reason, we only evaluate our proposed solutions on the competition tasks.

In Figure 3.2, we show the multi-view chest radiographs of a 73 year old female with Atelectasis. This is the collapse/closure of part or all of a lung, resulting in reduced gas exchange [90]. It can be identified on X-Rays by an opaque lung with reduced volume. The Posterior-Anterior (PA) X-Ray technique was used to obtain the frontal view. PA refers to the direction of travel taken by the X-Ray beam i.e. it hits the posterior part of the chest before the anterior part. It is the most common and preferred technique. The meta-data does not specify which techniques were used to obtain the lateral scans in the dataset. It is also worth noting that the dataset contains frontal scans for all patients, but only a smaller proportion have corresponding lateral scans. We show additional radiographs for all competition observations in Appendix B.

Figure 3.3 shows the multi-view chest radiographs of a 66 year old male with Cardiomegaly i.e. an enlarged heart. This can be identified on X-Rays by determining if the ratio of the widest diameter of the heart to the internal diameter of the chest exceeds a certain threshold [91]. The PA technique was used to obtain the frontal view.

The multi-view chest radiographs of a 57 year old female with (pulmonary) Consolidation are shown in Figure 3.4. This is lung tissue which has filled with liquid instead of air [92]. Consolidated lung tissue is usually more opaque than normal - it can typically be identified on an X-Ray by an area of white lung. The PA technique was used to obtain the frontal view.

In Figure 3.5, we show the multi-view chest radiographs of a 29 year old female with (pulmonary) Edema - fluid which has accumulated in the lung tissue [93]. Edema can be a cause of a pulmonary

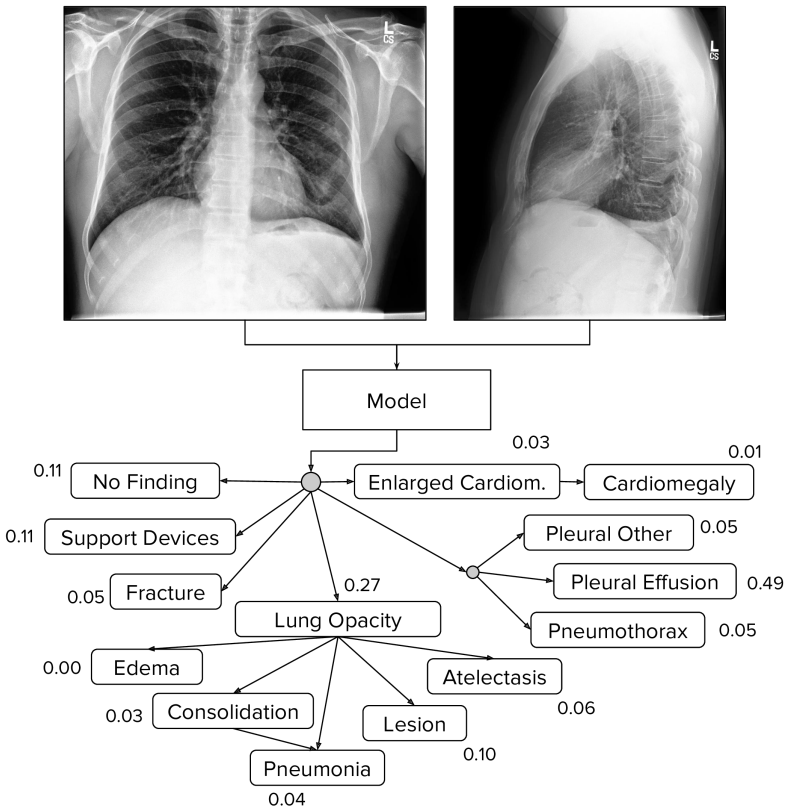
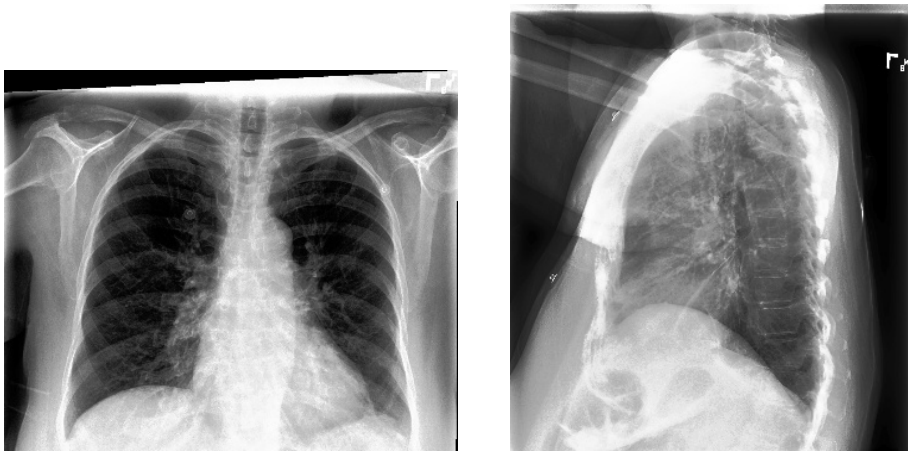


Figure 3.1: CheXpert sample with probability of different observations [8]

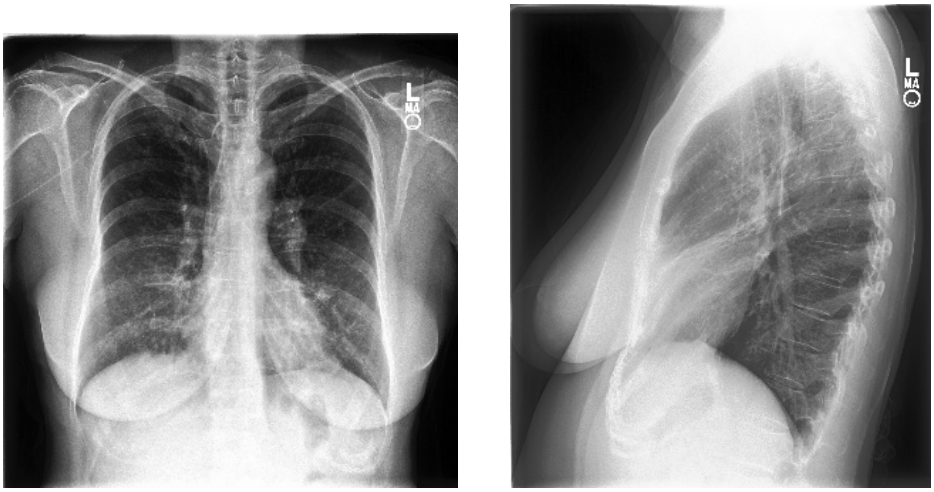


(a) Frontal (b) Lateral

Figure 3.2: Multi-view chest radiographs of Atelectasis patient

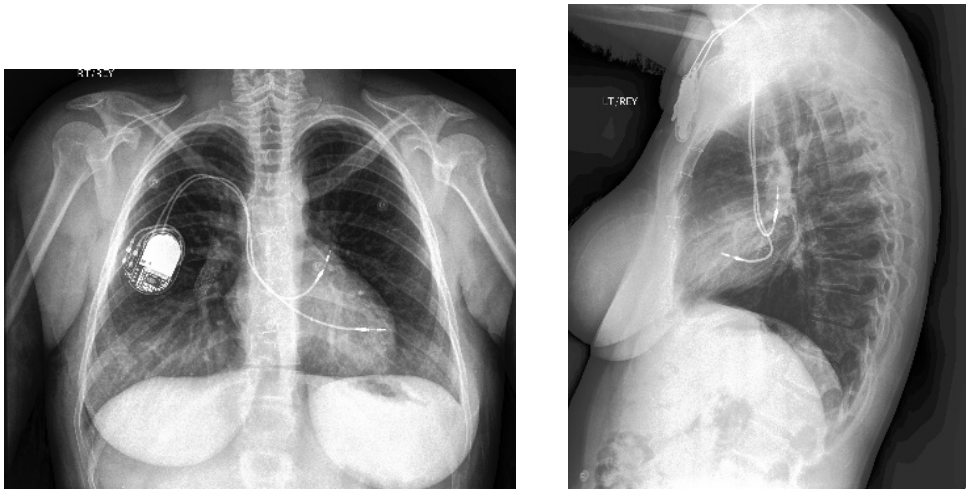
(a) Frontal (b) Lateral

Figure 3.3: Multi-view chest radiographs of Cardiomegaly patient



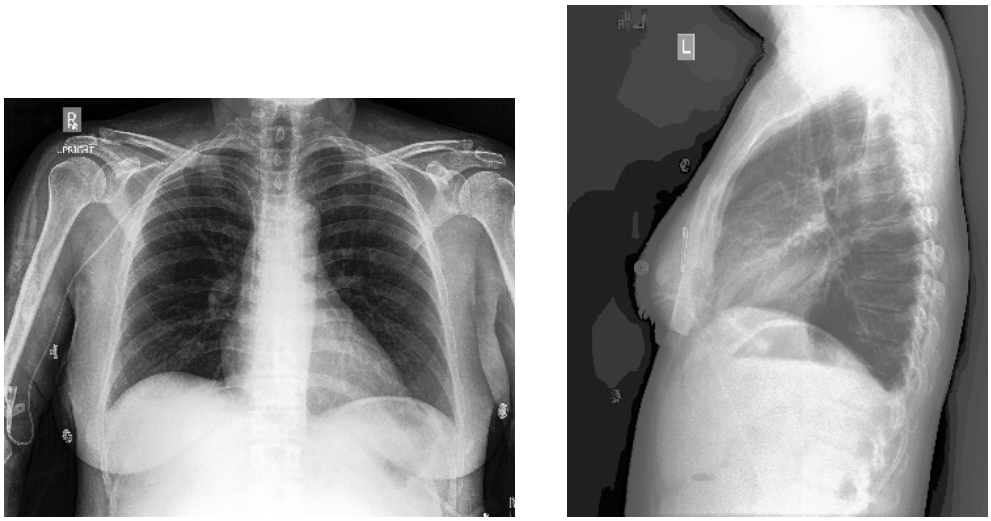
(a) Frontal (b) Lateral

Figure 3.4: Multi-view chest radiographs of Consolidation patient



(a) Frontal (b) Lateral

Figure 3.5: Multi-view chest radiographs of Edema patient



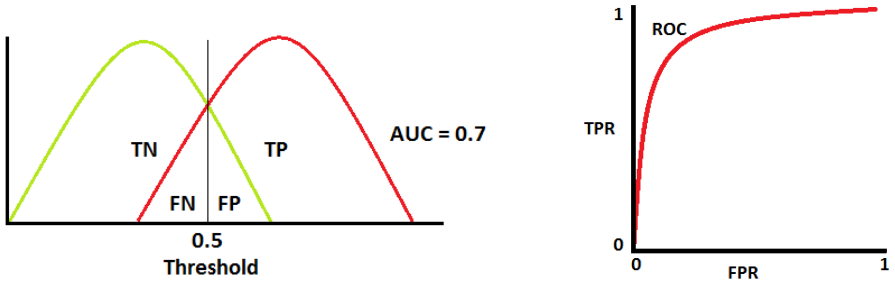
(a) Frontal (b) Lateral

Figure 3.6: Multi-view chest radiographs of Pleural Effusion patient

Consolidation and can be similarly identified on an X-Ray. The PA technique was used to obtain the frontal view. This particular patient also has a pacemaker, which could potentially affect the predictive capability of the classifier.

We show the multi-view chest radiographs of a 50 year old female with Pleural Effusion in Figure 3.6. This is when the pleural cavity, which surrounds the lungs, accumulates excess fluid [94]. This is different to the two previously discussed observations, since this cavity is supposed to contain fluid. Pleural Effusions show up as areas of white in X-Rays, towards the lower section of the pleural cavity. This is because the effusion is more dense than the rest of the lung. Unlike with the previous scans, the Anterior-Posterior (AP) technique was used to obtain the frontal view. This technique is less preferred because the image of the heart tends to be less clear and focused, because it is further away from the detector.

Even with the descriptions of the medical observations, as well as how to detect them from the scans, it is still extremely challenging to identify their presence. This makes it even more impressive that state of the art classifiers perform similarly to the medically trained radiologists [8].



(a) Distribution of negative (green) and positive (red) samples

(b) ROC Curve

Figure 3.7: Distribution curves and the corresponding ROC curve [96]

3.2.1 Evaluating Performance

The authors provide a validation set which contains 200 studies from 200 patients that are randomly sampled from the full dataset. The ground truth is obtained by taking consensus from three board-certified radiologists.

The hidden test set, used to evaluate solutions submitted to the competition leaderboard, consists of 500 studies from 500 previously unseen patients [95]. The ground truth labels are obtained by performing majority voting on the annotations from five out of eight board-certified radiologists. The remaining three radiologists' annotations are used to benchmark radiologist performance. Because we could not get access to the hidden test set, we use the provided validation set for our evaluation, and a subset of the training set for validation and holdout purposes. This is explained further in Section 3.5.

Evaluation Metric

The Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) graph is used to determine the leaderboard. A classifier's performance is determined by its mean AUC score across all observations in the test set. The AUC-ROC curve shows how good a model is at distinguishing between classes.

Suppose we have a classifier with the performance shown in Figure 3.7a. In this example, the model thresholds output such that a score less than 0.5 is classified as negative, and positive for the opposite. When we decrease the threshold, the recall/sensitivity/true positive rate (TPR) increases, while the specificity decreases. The converse occurs when the threshold is increased. The false positive rate (FPR) is defined as 1 - specificity. Therefore, TPR and FPR increase proportionally to each other. In an ROC curve, the TPR is plotted against the FPR at various threshold settings, as shown in Figure 3.7b. An AUC score of 1.0 means the model is a perfect classifier, 0.5 means the model is unable to distinguish between the positive and negative class i.e. no better than random, and 0.0 means the model always makes the opposite decision from the correct one [96]. The ROC-AUC curves of our benchmark model are shown in Figure 5.3.

3.3 Benchmark Implementation

The benchmark represents a centralised training approach. This entails different institutions sending their chest radiographs to a central server. This server then trains a model on the aggregated data and sends the global model back to all institutions. The purpose of the benchmark implementation is to serve as a point of reference - the goal of our FL solutions, which do not require institutions sharing raw patient data, is to achieve similar model performance. From the initial supervisor meetings, it was agreed that our benchmark model should be based on an implementation that achieves SOTA performance when trained centrally on the CheXpert dataset. The implementation we use as our template is the overall fourth best (at the time of writing) performing CheXpert model on the official leaderboard [95] and the highest performing solution with available

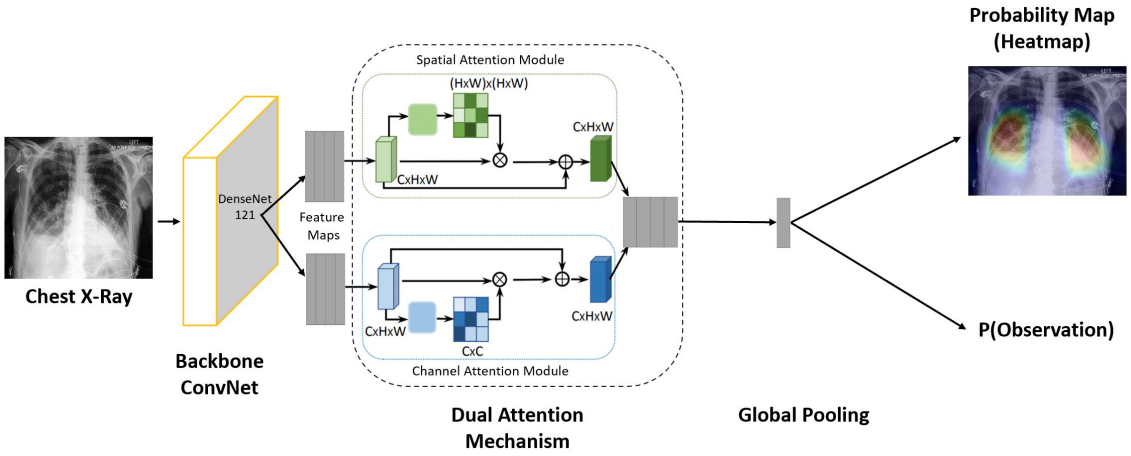


Figure 3.8: Benchmark model structure

source code. Having the source code available is a necessary requirement because of the strict time frame the project operates under. This benefited us because we had an "off-the-shelf" model ready to be trained, evaluated and adapted to perform FL training (the main focus of our project).

Before being fed into the model, the radiographs are first augmented. Figure 3.9g shows that the scans do not have the same orientation, position and scale. To mitigate the effects of this and improve the robustness of the classifier, the authors apply the following `RandomAffine` transformations during training:

- Rotate images randomly between (-15, +15) degrees
- Translate images randomly vertically and horizontally at most 5% of total height and width respectively
- Scale images randomly between 0.95x and 1.05x of original size

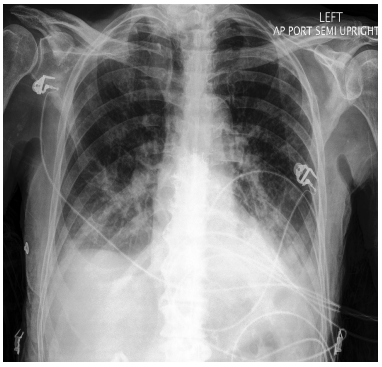
After being transformed, the images are resized to have input size 512 x 512 pixels. If the original image is not large enough, the borders are padded with the mean grayscale value of 128. Finally, the pixel values of the image are normalised according to the mean and standard deviation of the pixel values in the dataset. This is common practice and is most likely done by the authors to improve stability of training.

The authors also modify the labels by applying a hybrid of the U-Zeroes and U-Ones techniques explained in Section 3.2. For Edema and Atelectasis, uncertain labels are mapped to negative observations. For Cardiomegaly, Consolidation and Pleural Effusion, uncertain labels are mapped to positive observations. Looking back at Table 3.1, this is most likely to improve the balance of class distribution in the training samples.

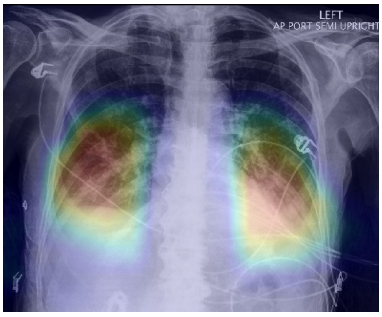
Figure 3.8 shows the structure of the benchmark model. One of the major components of the model is a DenseNet121 backbone, similar to that shown in Figure 2.10. A dual attention mechanism, representing a hybrid between the Channel Attention Module (Figure 2.4) and Spatial Attention Module (Figure 2.5) is also used in the best performing model. The global pooling layer is designed, not only to be used for classification, but also to generate a heatmap i.e. a method of visualising the likelihood and performing localisation of the different observations. Examples of generated heatmaps for some of the observations are shown in Figure 3.9. We evaluate the performance of the benchmark implementation in Section 5.1.

3.4 Baseline Implementation

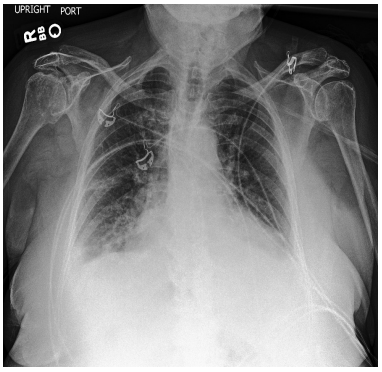
The baseline represents institutions that do not send any data to an aggregator or any other institution i.e. they train models locally. The goal of the project is to investigate how the performance



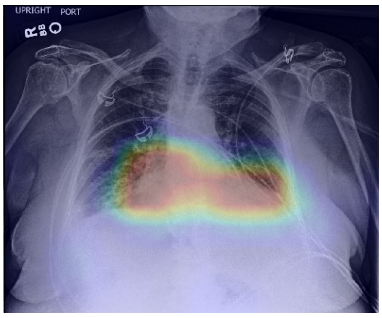
(a) Atelectasis - Radiograph



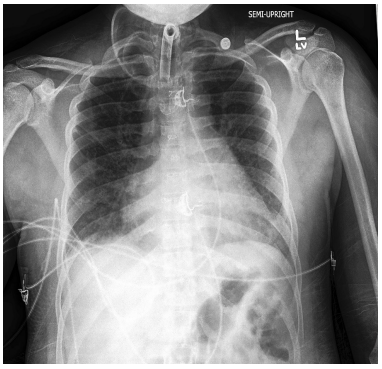
(b) Atelectasis - Heatmap



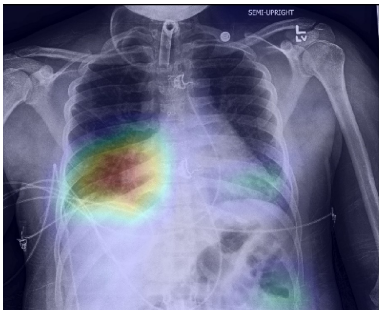
(c) Cardiomegaly - Radiograph



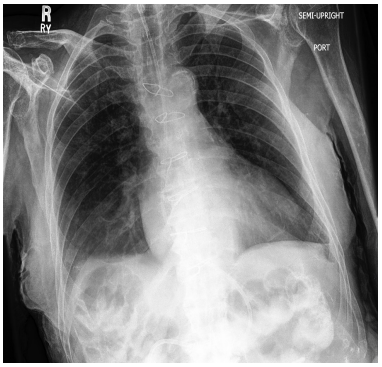
(d) Cardiomegaly - Heatmap



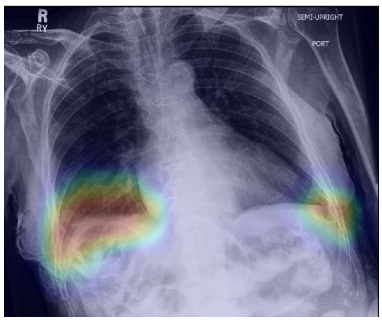
(e) Consolidation - Radiograph



(f) Consolidation - Heatmap



(g) Pleural Effusion - Radiograph



(h) Pleural Effusion - Heatmap

Figure 3.9: Heatmaps generated by global pooling, adapted from [97]

of the baseline and FL approaches change relative to the benchmark when the dataset is partitioned between n institutions. To accurately simulate federated medical scenarios, the data is distributed between the institutions in a non-i.i.d. manner. Further details of the data-partitioning scheme can be found in Section 3.5. In this section, we state and explain the different investigations carried out. The results of our experiments are placed in Section 5.

Our initial goal is to investigate the potential benefits, if any, of FL applied to **two institutions** when they each contribute equal amounts of data to the training process. For an FL technique to be deemed successful, it would need to outperform both the baseline models and simultaneously perform similarly to the benchmark. This would show that the FL model is better at generalising on new data than all the models trained individually by the institutions. At the very least, we would require the FL approach to outperform the worst performing baseline model. This would justify the use of FL, since it would mean at least one institution would be receiving a shared model which is better at generalising on unseen data. If the FL model is unable to outperform the worst institutional model, we conclude that it is not an appropriate approach for the particular use case. We run an additional experiment involving two institutions, where one contributes 75% of the training data, while the other contributes the remaining 25%. The purpose of this experiment is to investigate the impact on shared model performance when institutions contribute an uneven proportion of data. This would be analogous to a large hospital collaborating with a clinic.

A stretch goal of the project is to investigate model performance and training overhead when the number of participating institutions scales up, simultaneously reducing the proportion of data contributed by each institution. We investigate performance when the dataset is partitioned between **five institutions**, each contributing 20% towards training. This is analogous to several medium-sized hospitals collaborating. We considered investigating uneven splits on larger numbers of institutions, but decided against it. This is because it would involve running multiple extra sets of experiments - we would not have had time to complete them due to the intense competition for computational resources on the department GPU cluster. We extend the stretch goal by investigating performance when **ten institutions** contribute equal amounts of data. This is analogous to multiple clinics collaborating. For the same reason as in the aforementioned experiment, we do not investigate more than ten participating institutions.

The baseline models use the same underlying structure as the centralised version described in Section 3.4. They only differ from the benchmark implementation on certain hyperparameter values. We describe our hyperparameter tuning strategy in Section 3.5.

3.5 Experimental Methodology

When it was agreed that CheXpert would be used for training and evaluation, we immediately registered with the Stanford University School of Medicine to download the dataset. It was important to do this early, because we did not know how long the approval process would take. Once the request was approved, we had the choice of using the lower resolution (∼11GB) or full resolution (∼439GB) dataset. We use the downsampled version to train our models, since it facilitates faster training while also having a considerably lower storage overhead.

Due to the sensitive nature and volume of the dataset, it was not feasible to perform training on public clouds such as Amazon Web Services (AWS), Microsoft Azure or Google Cloud Platform (GCP). Therefore the decision was taken to store the dataset on the department's `/vol/bitbucket/` network share. This also required us to modify the folder permissions to ensure no other users could access the data. A benefit of storing the data on the network share is that it is accessible from any Department of Computing (DoC) computer. For this reason, we also set up our Python virtual environment and repository[2] on the network share. This enabled us to run experiments from any of the DoC machines, including the GPU cluster[3], with minimal additional configuration required.

To ensure experimental reproducibility of results, we fix seeds in any random number generator (RNG) used by our implementation. This is highlighted in Listing 3.1. To further aid reproducibil-

[2] `https://github.com/erikbabu/federated-learning-healthcare-data`

[3] `http://www.imperial.ac.uk/computing/people/csg/guides/hpcomputing/gpucluster/`

ity, we provide the configuration files and detailed documentation on how to configure training. For transparency, the repository also includes log files for every single experiment. These report the evolving performance of the model on the holdout set during training.

```
# Fix numpy random seed
import numpy as np
np.random.seed(0)

# Seed RNG for all torch devices (both CPU and CUDA)
import torch
torch.manual_seed(0)

# Invocation of pandas dataframe sampling function (using fixed RNG)
partition = meta_data.sample(n=num_records_to_sample, random_state=0)
```

Listing 3.1: Seeded RNGs in our implementation

To ensure we correctly configured the SOTA implementation, we did not immediately train the model on the full dataset. This is because training the benchmark model on the entire dataset for 10 epochs on a GPU takes two full days. To mitigate the effects of long training times, we use functionality provided by the scikit-learn [98] library to perform quantitative sampling of the dataset. We generate a 5% sample which has the same distribution of meta-data and labels as the full dataset. The same technique is used to obtain representative samples of 1000 radiographs, representing our validation (500) and holdout (500) set. The SOTA implementation is trained on this 5% sample and evaluated on the test set. After obtaining meaningful results, we were confident that the implementation was configured correctly. We then repeated the sanity check on a 20% sample before training the model on the full dataset. The model checkpoint we use for the final evaluation is that which has the highest mean AUC on the holdout set. The authors of the SOTA implementation report a mean AUC of 0.913. Our benchmark implementation achieves 0.012 AUC less. This slight discrepancy is most likely due to the fact that their model is trained on the full resolution dataset. To further mitigate the effects of long training times, we attempted to further downsample the images. However, this idea was discarded, because it led to significant degradation of model performance.

Hyperparameters, such as the number of epochs, learning rate, momentum, attention technique and model backbone (Densenet vs VGG vs Inception) are tuned on the validation set. The authors, who train the SOTA implementation on four GPUs in parallel, use a batch size of 56. We are constrained to use a batch size of eight or less when training our models. This is because if higher values are used, the GPU runs out of memory and we enounter `RuntimeError: CUDA error: out of memory` errors. Because hyperparameter searches are expensive to perform, we only run them on the 5% samples of the dataset and use the best resulting configuration to train the final model on the entire dataset. The drawback of doing it this way is that we cannot be sure if the sample is fully representative. For example, the radiographs are captured by different radiologists, each potentially using a different brand of MRI device. The meta-data does not specify this information. Consequently, although we ensure the distribution of available meta-data and labels is the same, our samples may still introduce implicit bias. We perform separate hyperparameter searches for the baseline and FL approaches (further discussed in Section 4.4) in every set of experiments. This is because the distribution of data at each institution is different from the centralised setting i.e. they do not necessarily share the same optimal hyperparameter configurations. Another reason is that we assume there is no centralised solution when we run the baseline and FL experiments. Therefore, it would invalidate experimental integrity to train the other approaches using the optimal hyperparameters of the centralised approach.

Before generating the institutions' partitions, we re-label the competition observations specified in the meta-data to be consistent with the heuristic applied to the benchmark implementation (described in Section 3.3). Because all the radiographs in the CheXpert dataset are collected from Stanford Hospital between 2002 and 2017, we have to create our own partitions to simulate federated institutional data. In medical settings, the prevalence of observations between institutions will be different. To ensure our experiments accurately depict this, we distribute the data between the institutions in a non-i.i.d. manner. We generate the first partition by sampling (without replacement) from the full dataset. The next partition is generated by sampling (without replacement)

from the remaining records. To ensure different distributions for each institution, we instruct the sampling algorithm to assign different likelihoods/weights to the five observations. Appendix A contains charts that show the distribution of labels between the institutions, for all sets of experiments.

Throughout the project, there was intense competition for computational/GPU resources. A drawback of training models on the department's GPU cluster is that it only allows users to run up to two concurrent jobs. It also terminates processes after four days of run-time. When demand exceeds available resources, there is also an associated queuing time before a job is scheduled to run. To avoid using the cluster, we tried to run experiments on the DoC `ray` machines. These machines do not have a limit on job run-time. For this reason, they were desirable to a lot of other users. This made it challenging to find available machines. Even when we did find one, there was no guarantee that our experiment would complete - we often encountered more GPU `out of memory` errors because another user would run their experiment on the same machine. Consequently, we would lose all training progress. We mitigated this by implementing model checkpointing. This enabled us to resume training after a job was terminated. Despite this, it became frustrating to continuously have to resume jobs, particularly when experimenting on the ten institution split. This is because we would need to run 12 simultaneous experiments - ten for the baseline models and two for the different FL approaches. We therefore decided to persevere with running experiments on the GPU cluster.

3.6 Measuring Overhead

We implement a way to measure the computational and communication overhead of the training process. This is because existing research on FL applied to medical imaging tends to focus solely on model performance. Since FL is a distributed learning problem, we agreed in the initial supervisor meetings that our approach should investigate the viability of FL, not only in terms of model performance, but also regarding training overhead. We address further benefits of reporting these metrics in Section 6.2.4 and show our results in Section 5.

3.6.1 Computation

For the benchmark, our overhead model assumes that all training is performed at the central server - it incurs all the computational overhead and the institutions incur none. Given a model and the inputs for a single training batch, the THOP library computes and returns the number of Multiply-Accumulate (MAC) operations performed on that batch of data. A single MAC has one multiplication and one add operation i.e. it has the form $a \leftarrow a + (b \text{ x } c)$. We therefore approximate the number of Floating Point Operations (FLOPs) by doubling the MACs.

Although the THOP library abstracts away the complexity of the calculations for the most common layers e.g. `conv2d`, it has a provision to supply `custom_ops`. These enable the library to calculate the overhead for layers it has not encountered before. In Listing 3.2, we show an example calculation for the sigmoid activation. Because the exponential operation is expensive to compute, it can be approximated as $\exp(x) = 1 + x + \frac{x^2}{2!} + ... + \frac{x^9}{9!}$. The factorials are considered as constants. Therefore, applying the sigmoid activation to a single element involves 10 add and 9 division operations - seen in Line 5.

```
1 def count_sigmoid(model, x, y):
2     # sigmoid f(x) = 1/(1+exp(x))
3
4     nelements = x[0].numel()
5     total_ops = 19 * nelements  # Because sigmoid is element-wise
6     model.total_ops += torch.DoubleTensor([int(total_ops)])
```

Listing 3.2: Example computational overhead calculation for sigmoid layer

The total computational overhead of training the benchmark model is calculated as follows. We determine the number of batches in an epoch. This is multiplied by the total number of epochs to determine the total number of batches being trained. We then multiply the MACs performed in a single batch by the total number of batches, to obtain the total MACs. This value is doubled

to obtain the FLOPs approximation. The same calculation is performed when evaluating the computational overhead of the baseline models. However, the baseline implementation has no notion of a central aggregator - each institution incurs computational overhead since they train the models themselves.

3.6.2 Communication

We report communication overhead as the total bytes communicated i.e. the sum of bytes uploaded and downloaded. In the centralised implementation, from the institutions' point of view, they upload raw images and labels to the aggregator. When the model has been trained, they download the model updates from the aggregator. From the aggregator's point of view, it has to download the raw images from every institution - the full CheXpert dataset ($\sim$ 11GB) in this case, train the model and upload the weights to each institution. In the baseline implementations, we assume that there is no communication overhead. This is because the institutions train their own models in isolation.

For all experiments, we assume there is no additional networking overhead. This not very robust because, in reality, there will be packet overhead and re-transmission. By ignoring these, we acknowledge that our reported values will not be as accurate. However, they do serve as a lower bound on the communication overhead. It is also worth noting that the network cost of downloading the full model (1624 MB) is not factored into the reported metrics on communication overhead for any of the experiments. This is because to train or use the model, every institution and aggregator would need to download the full version once, and its size never changes.

Chapter 4

FL Models

This chapter focuses on the work required to develop and train our FL solutions. We begin by outlining the key tools and libraries used, in Section 4.1. In Section 4.2, we demonstrate the simplicity of the FL algorithms by showing their source code. We then explain how we modify the centralised implementation to facilitate FL, in Section 4.3. This is proceeded by a discussion of our experimental methodology and how we address challenges encountered, in Section 4.4. The minimum target of our FL solutions is to at least outperform the worst baseline model. Our goal is also to achieve similar performance to the benchmark model. The evaluation of our FL solutions is located in Section 5.

4.1 Key Tools and Libraries

Before working on the benchmark and baseline implementations, we had to first decide which FL framework to use. This is because certain frameworks only have support for models implemented using a particular library. Once a framework was selected, we then searched for a SOTA model, implemented in the appropriate library, to use as a template in our benchmark (previously discussed in Section 5.1). The following FL frameworks were considered:

- **TensorFlow Federated (TFF)** [99]: This framework provides high- and low-level interfaces to express custom FL algorithms. We decided not to proceed with this framework due to our unfamiliarity with TensorFlow - it notoriously has a steep learning curve. Another reason is that none of the top-10 performing CheXpert models (with available source code) are implemented in TensorFlow; we require our benchmark model to achieve SOTA performance.

- **FATE** [100]: This framework is developed by WeBank's AI team to support federated AI ecosystems. It also implements several formal privacy-preserving protocols. Because it is still a fairly new framework, it has not built a large enough community of users. For this reason, we did not proceed with the framework. It would have been difficult to find support online if we ran into any potential difficulties during our implementation.

- **NVIDIA Clara SDK** [101]: This framework gives data scientists and developers the tools to accelerate the development of privacy-preserving AI algorithms for medical imaging. It provides pre-trained models and is highly customisable. However, we decided against utilising this framework because it uses Docker and Kubernetes, which require root privileges. We do not have these permissions on the DoC machines and running the experiments on a personal computer was infeasible, due to slower training times when executed on a CPU.

- **PySyft** [102]: Pysyft provides abstractions that make it simple to implement FL, ε-DP and SMPC. Another benefit is that is provides an API similar to PyTorch, which we have a lot of familiarity with. The library is well documented and there are also several tutorials[1] provided. PySyft was initially our framework of choice. However, as we got further into the implementation, we ran into several issues. To mitigate this, we decided to implement the FL training loops ourselves. This is further discussed in Section 4.3.

[1] `https://github.com/OpenMined/PySyft/tree/master/examples/tutorials`

4.2 FL Algorithms

In this section we show source code for the FL algorithms we evaluate as part of our project.

4.2.1 Federated Averaging (FedAvg)

This is described in Section 2.8.3 as Equal Averaging. We accumulate the weights from all participants across all weight parameters, proportional to the total number of participants in the training process, as shown in Listing 4.1.

```
def fed_avg(weights):
    """
    Performs federated averaging, giving equal importance to each participant

    weights: The updates sent from all participants, placed in a list
    """

    # Determine number of clients and averaging factor
    n_clients = len(weights)
    factor = float(1 / n_clients)

    # Perform averaging
    avg = weights[0]
    for k in avg.keys():     # iterate through all weight parameters
        for i in range(1, n_clients):
            avg[k] += weights[i][k]
        avg[k] = torch.mul(avg[k], factor)

    # Averaged weights to be sent back to participants
    return avg
```

Listing 4.1: FedAvg implementation

Weighted FedAvg

This is described in Section 2.8.3 as Weighted Averaging. We accumulate the weights from all participants across all weight parameters, proportional to the amount of data they contribute to the training process, as shown in Listing 4.2.

```
def weighted_fed_avg(weights, factors):
    """
    Performs weighted federated averaging
    Gives importance to each participant
    weighted on number of data points
    contributed

    weights: The updates sent from all participants, placed in a list
    factors: The proportion of data each participant contributes, as a list
    """

    # Determine number of clients
    n_clients = len(weights)

    # Perform weighted averaging
    w_avg = weights[0]
    for k in w_avg.keys():
        w_avg[k] = torch.mul(weights[0][k], factors[0])

    for k in w_avg.keys():
        for i in range(1, n_clients):
            w_avg[k] += torch.mul(weights[i][k], factors[i])

    # Averaged weights to be sent back to participants
    return w_avg
```

Listing 4.2: Weighted FedAvg implementation

We discuss further changes required by the repository to facilitate FL training in Section 4.3.

4.2.2 FedProx

The details of the algorithm are described in Section 2.8.4. FedProx enables the use of any aggregation function. Unlike the previously discussed FL approaches, it is implemented on the client-side. It works by adding a regularisation term to the training loss value, based on how different the local weights are from the global ones. Listing 4.3 shows that apart from adding the regularisation term, the training process occurs as usual. We implement FedProx to serve as comparison to the performance of FedAvg. It is worth noting that training models using FedProx takes approximately twice as long, compared to those trained using FedAvg. This additional latency is due to the overhead of performing the expensive computation of the regularisation term for each batch of data.

```
# Get loss as usual
loss = get_loss(output, target)

# Compute FedProx regularisation term and update loss value
reg = 0.0
for param_index, param in enumerate(model.parameters()):
    reg += ((mu / 2) * torch.norm((param - global_weights[param_index]))**2)

loss += reg

# Compute gradient of loss as usual
loss.backward()
```

Listing 4.3: FedProx implementation

4.3 Implementation

The initial plan was to import our benchmark model into PySyft and utilise the abstractions provided by the framework to perform FL, and eventually integrate formal privacy guarantees into the solution. As mentioned in Section 4.1, we abandoned PySyft mid-way through the project. Before the implementation stage, when working through the tutorials to familiarise ourselves with the framework, we encountered our first issues. Because PySyft is not as mature as the more established frameworks, it is still being constantly updated and the API is always evolving. This meant that some of the tutorials were outdated - certain functions either did not work, or demonstrated unexpected behaviour. During our implementation phase (March 2020), we encountered further issues with the framework. It provides a `federate()` method, which distributes the data from a `FederatedDataLoader` across the participants. The problem with this function is that it randomly assigns data to the participants i.e. we cannot specify which data should go to a particular institution. The function also does not have a provision to support uneven distribution of data between participants - a key requirement for one of our experiments. Another serious issue was that the framework did not support the momentum argument in the optimiser. This would likely mean our FL models would require even more epochs to converge. For reasons explained in Section 4.4, we would not have been able to perform FL training for more epochs. When it became apparent that formal-privacy guarantees were beyond the scope of the project, we decided that we were no longer receiving sufficient benefits from PySyft to justify its use. In this section, we describe how we manually implemented the FL training loops instead.

Only the training process needed to be modified to facilitate FL. We use the implementation of the benchmark as a template and make slight modifications. The outline of our FL implementation is inspired by an open source version[2] released by IBM, which also manually performs the FL training loops. We create a dictionary that maps clients to their corresponding `DataLoader` and another field, to track the communication overhead incurred. We also track the communication overhead of the aggregator. Details on how the overhead is calculated is explained in Section 4.5. Note that the FL institutions have the same partitions of data as the baseline, to make comparisons between their performance valid.

Another modification made to the benchmark implementation is to wrap the training procedure in an external loop. The iterations in the outer-loop represent federated communication rounds. The inner-loop now represents the number of local epochs to train the model for. All institutions

[2] `https://github.com/IBM/FedMA/blob/master/language_modeling/language_main.py`

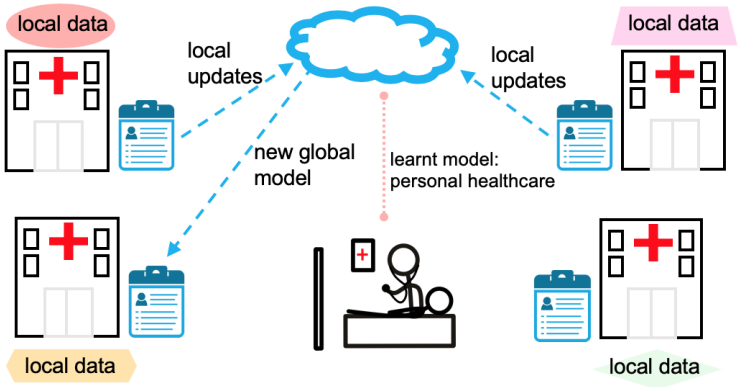


Figure 4.1: FL for healthcare via patient radiographs from heterogeneous institutions [103]

train the model locally for the same number of local epochs. At the start of the training process, the aggregator randomly initialises model weights and sends it to all institutions, updating its communication overhead in the process. All institutions load these weights, perform local training, send their updates to the aggregator and update their overhead. The aggregator receives updates from all participating institutions, performs the averaging (as described in Section 4.2), sends the new shared weights back to the institutions and updates its overhead. At the end of a federated round of communication, we evaluate the shared model. If the new version outperforms the previous best on the holdout set, we checkpoint it to be used for evaluation. Figure 4.1 summaries the overall FL training process.

4.4 Experimental Methodology

Just as in Section 3.5, we first trained our FL implementations on the 5%, followed by 20%, samples of the dataset to ensure everything was configured correctly. Once meaningful results were obtained, we ran the experiments on the full dataset. This approach proved to be beneficial, since our initial FL implementation had an error in the model checkpointing code. Because of the error, the model was not being correctly deserialised during the evaluation stage. Since we ran an end-to-end verification of the implementation after the first experiment, we isolated this and fixed the issue immediately. If we had only performed the evaluation after all experiments completed, we would have realised this issue too late - all the models would have required re-training.

As mentioned in Section 3.5, we also perform hyperparameter searches when training our FL approaches. The FL-specific hyperparameters we tune are the number of local epochs, global communication rounds and scaling factor μ of the regularisation term (only with FedProx). The first full set of FL experiments were run on the two institution splits, using the optimal hyperparameters found. FedProx was implemented to serve as a point of reference to the performance of FedAvg. When the FL solution achieved our model performance targets, we performed the stretch goal experiments - using five and ten participating institutions.

It is important to note that the limit on job run-time on the department's GPU cluster might influence the experimental results. The benchmark and baseline models all converged after being trained for less than 3 days. However, the FL experiments were terminated after the four day limit had been reached. There is a possibility that given more training time, both FedAvg and FedProx would perform slightly better than reported in our evaluation section.

4.5 Measuring Overhead

The computational overhead of training the FL models is calculated in a similar way to the benchmark and baseline approaches (described in Section 3.6.1). The only difference is that the total

batches are calculated by multiplying the number of batches in an epoch by the number of local epochs and the number of global communication rounds. The FL computational overhead metrics we report in Section 5 represent the average computation performed by all institutions during training. Because the FL aggregators simply perform averaging of updates at the end of each communication round, we treat their computational overhead as negligible.

In a federated round of communication, the aggregator uploads the initial global weights (n bytes) to the k different institutions. At the end of the round, it downloads the updates (n bytes) from each institution and performs the aggregation. This is repeated for T communication rounds. At the end of training, the aggregator sends the final updates to each institution. The total communication overhead of the aggregator, in bytes, can therefore be calculated as:

$$(2 \cdot n \cdot k) \cdot T + n \cdot k$$

In our modelling assumption, the size of the updates is constant. However, it is worth noting that the aggregator overhead increases linearly with the number of institutions. This is because it needs to transmit updates to/from a larger number of participants. Realistically though, we would not expect more than 100 medical institutions to collaborate in an FL setting. Depending on how slowly the model converges, T could potentially have the strongest impact on the FL aggregator overhead.

The total communication overhead of a participating institution is simply $1/k$ of the aggregator's communication overhead i.e. $(2{\cdot}n){\cdot}T{+}n$. The overhead of each institution does not directly depend on the number of participating institutions. However, more participating institutions may lead to slower convergence, which would increase the communication overhead. The communication overhead metrics of the FL institutions, reported in Section 5, represent the average across all institutions during training. Our results show that FL puts the network under less strain than the centralised approach. This is because weight updates are transmitted, rather than raw images.

Chapter 5

Evaluation

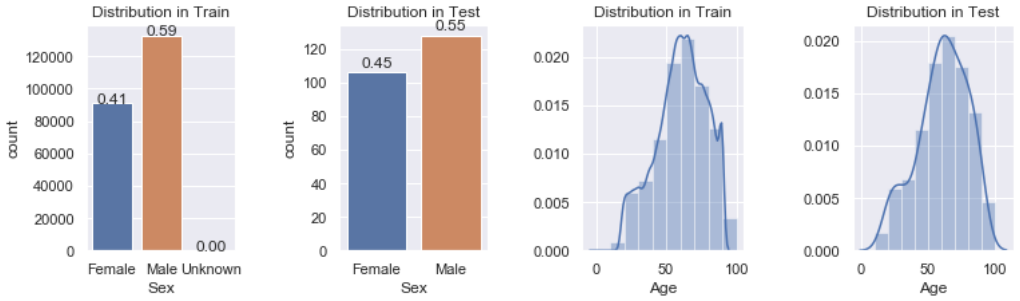
In this section, we discuss, evaluate and compare the results of our experiments on different FL techniques applied to varying numbers of institutions. To prevent the main body of the report from becoming cluttered, we only show charts describing the distribution of meta-data, labels and observation-specific performance for the benchmark model. For all the other experiments, this information is placed in Appendix A. At the end of each experiment, we compare the performance of the models, as well as the computational and communication overhead of the different approaches. At the very least, an FL technique needs to outperform one of the baseline models in each experiment. Ideally, an FL technique outperforms every baseline model in each experiment. In the best case, an FL technique is also able to achieve similar performance to the benchmark model in each experiment.

5.1 Benchmark

As described in Section 3, this represents a model trained on centrally aggregated data from multiple institutions.

The distribution of meta-data between the training and test data is shown in Figure 5.1. From this, we see that there is a minor difference in male/female ratios. Apart from that, the rest of the meta-data follows the same distribution. The distribution of labels is shown in Figure 5.2. The label having a value of 1.0 indicates a positive observation and the label having a value of 0.0 indicates a negative observation. From this, we notice that for all observations, there is a surprising difference in the distribution of labels between the training and test data. This is unusual because we typically expect the training data to be representative of what the model encounters in the "real world". This is only mentioned for analytical purposes; we decide not to modify either dataset to preserve experimental integrity. In reality, the test set would be hidden anyway and we would not be aware of this discrepancy.

For all experiments, the key performance indicator of the model is the ROC-AUC score. Further details on how to interpret the scores are provided in Section 3.2.1. In simple terms, the higher the AUC, the better the model is at distinguishing between positive and negative classes. We do not focus on the accuracy values, as they change depending on the threshold - for all experiments, we set a threshold of 0.5 for the model to classify its output as a positive observation. We could have determined the optimal threshold for each observation in each experiment, but this was deemed low priority, since the AUC score is a more comprehensive measure and is also the metric used in the official CheXpert leaderboard [95]. The performance of the benchmark model on the different observations is shown in Figure 5.3. We see that the model nearly acheives an AUC score of 0.9 on Atelectasis, and achieves AUC scores of more than 0.9 for Consolidation, Edema and Pleural Effusion. Consistent with the reference implementation [97], Cardiomegaly is the most difficult observation to diagnose. The success of the FL approaches will depend on how close they get to the mean AUC score of the benchmark model (0.901).



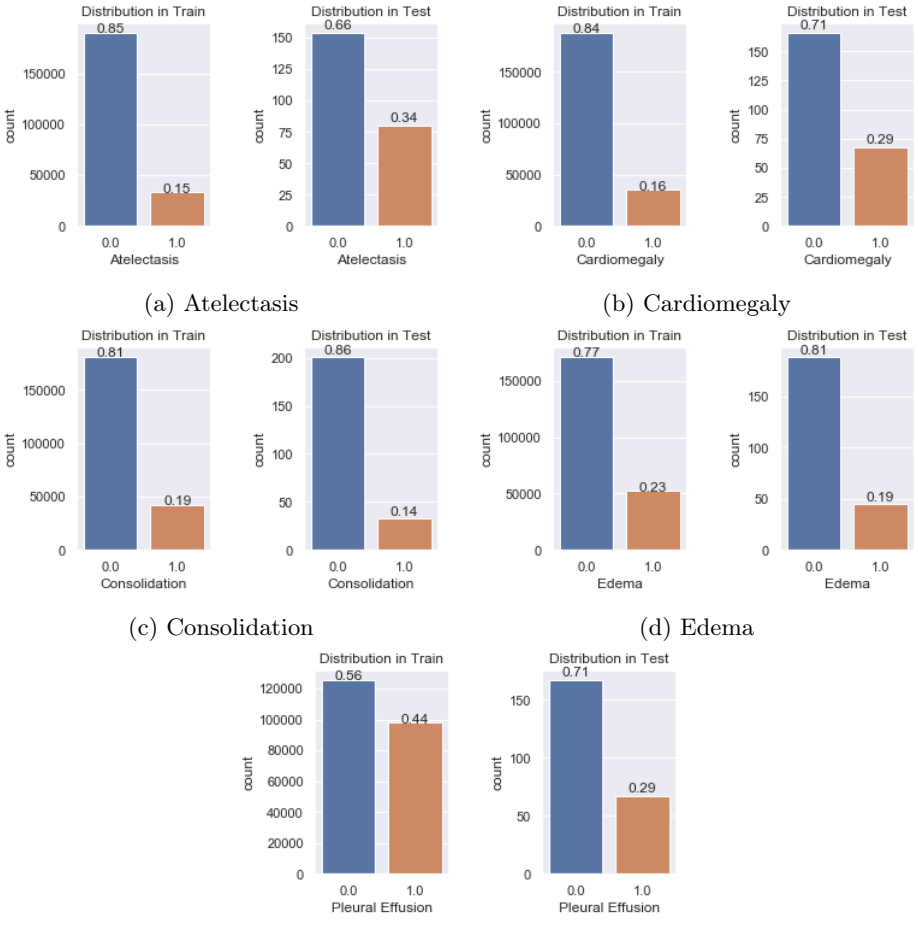
(a) Gender

(b) Age

(c) Scan Orientation

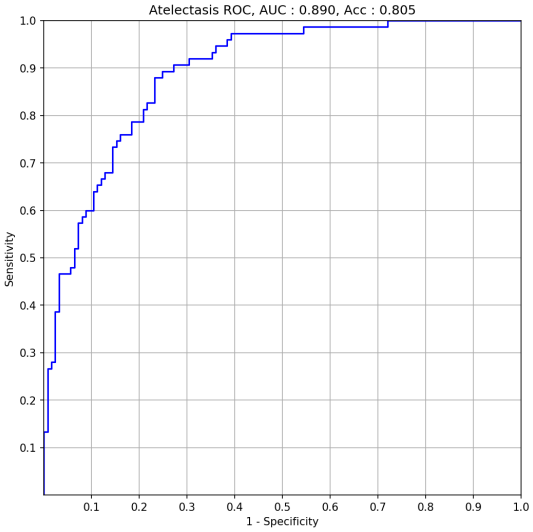
(d) Radiograph Technique

Figure 5.1: Distribution of meta-data between train and test sets

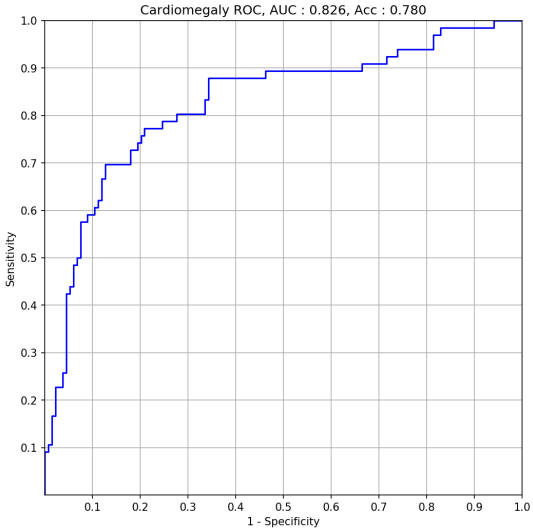


(e) Pleural Effusion

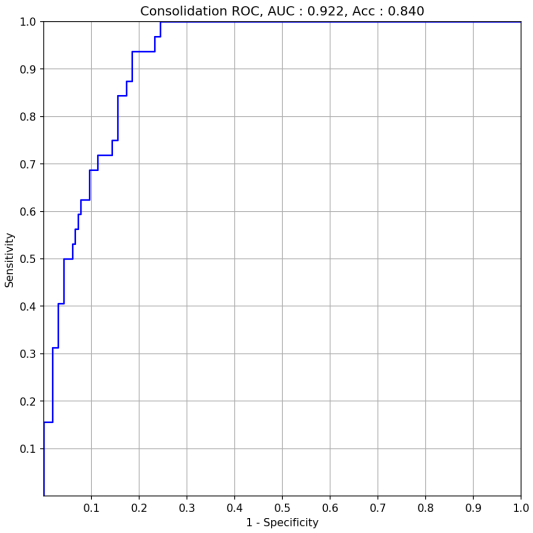
Figure 5.2: Distribution of labels between train and test sets



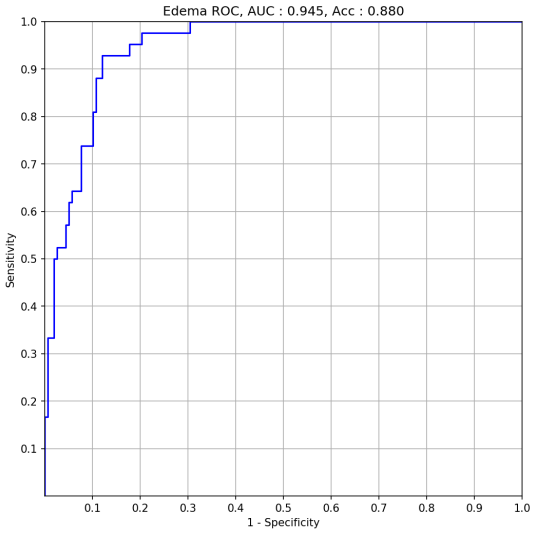
(a) Atelectasis



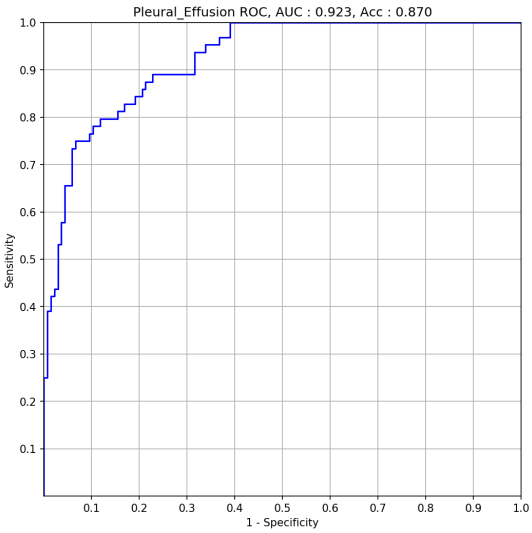
(b) Cardiomegaly



(c) Consolidation



(d) Edema



(e) Pleural Effusion

Figure 5.3: Performance of benchmark model on the different observations

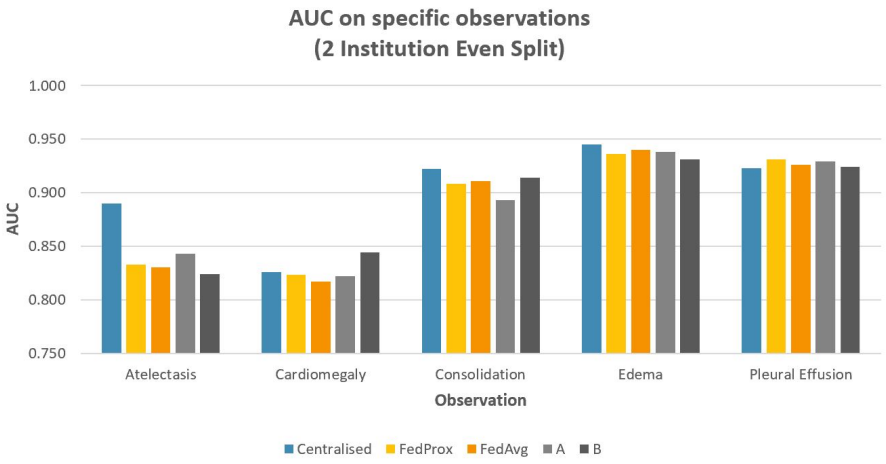


Figure 5.4: AUC on each observation, colour coded (blue, amber, gray) by the 3 approach classes

5.2 Two Institution Split (50/50)

In this experiment, the training data is partitioned between two institutions such that each one contributes 50% of the data during the training phase. The full set of results for this experiment is placed in Appendix A.1.

In Figure 5.4, we compare model performance of each technique on the different observations. We colour code the centralised (blue), FL (amber) and baseline (gray) approaches to assist with interpretation. We see that the benchmark model significantly outperforms all the other approaches regarding Atelectasis classification. This is unexpected and there is no clear reason why it should perform so much better than the other approaches. It is also interesting that the benchmark model does not necessarily outperform the other approaches for every observation. This is most likely due to the fact that the dataset is very large; 50% of data at each institution is probably enough to saturate performance of each baseline model. In Figure 5.5, we compare the % offset in mean AUC between the different approaches to the centralised model. We see that FedAvg is the worst performing approach. This is most likely because the partitions are non-i.i.d., making it more difficult for the algorithm to converge. FedProx, which has better convergence properties in non-i.i.d. settings, outperforms both FedAvg and one of the baseline models. This meets the minimum requirement of our experiment. The model trained using FedProx also achieves within 1.5% of the benchmark, therefore satisfying the requirement that it performs similarly to the centralised model. The model trained at institution B performs closest to the centralised model, slightly outperforming FedProx, by 0.12%. However, as previously mentioned, the reason for it outperforming the FL models may be due to the effects of data saturation.

In Figure 5.6, we compare the computational overhead when training the models using the different approaches. As FedAvg has weaker convergence properties than FedProx, there is no surprise that it has a larger computational overhead; it requires more federated rounds, and thus computational operations, to train the model to achieve good performance. It is interesting that the model trained at B outperformed the one trained at A and has nearly half its computational overhead, even though they are both trained on the same amount of data. The data partition at institution B is likely more representative, leading to faster convergence and better performance. In Figure 5.7, we compare the communication overhead when training the models using the different approaches. As explained in Section 3.6.2, the baseline models have no communication overhead. The centralised aggregator has significant overhead because it needs to download all the images from the institutions. Similarly, the institutions in the centralised approach also have a large overhead because they need to upload all the images to the server. In comparison, both FL solutions utilise very little of the network. This is because participating institutions send model updates, rather than entire datasets, to the aggregator. Consequently, the FL aggregators also do not have the overhead of downloading the large volume of images. It is expected that FedProx has a slightly smaller communication overhead than FedAvg because it typically converges in fewer communication rounds.

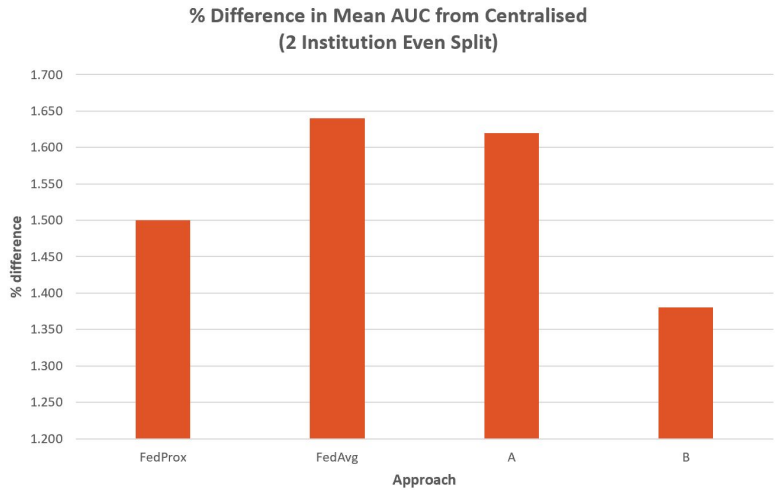


Figure 5.5: Mean AUC % offset from benchmark model - 2 institutions (even)

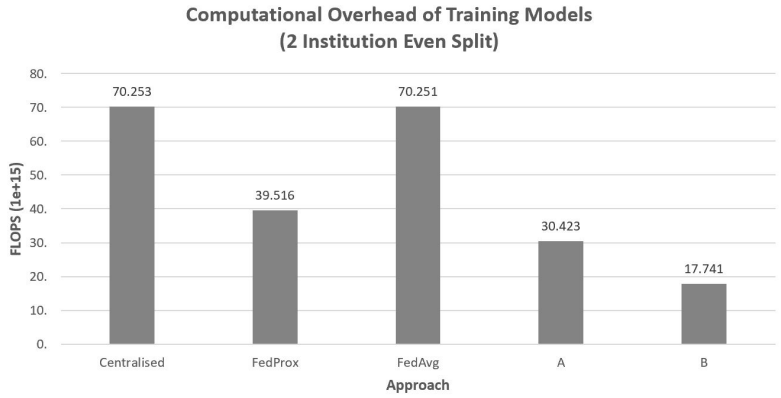


Figure 5.6: Computational overhead using the different approaches - 2 institutions (even)

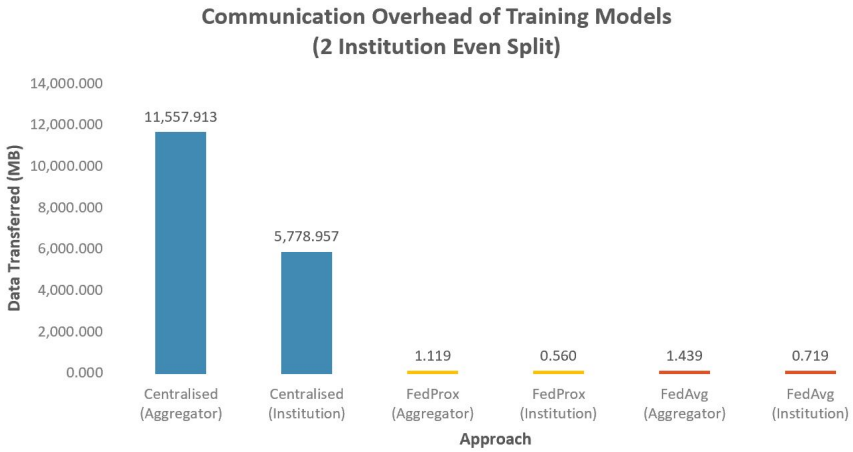


Figure 5.7: Communication overhead using the different approaches - 2 institutions (even)

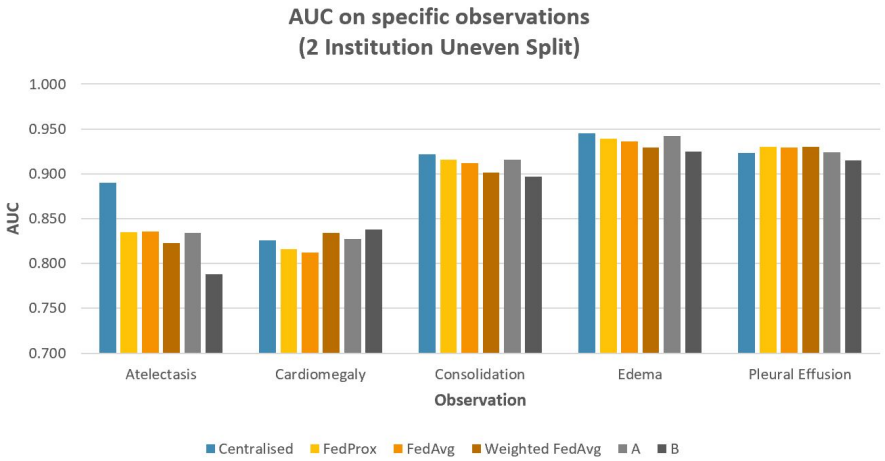


Figure 5.8: AUC on each observation using different approaches - 2 institutions (uneven)

5.3 Two Institution Split (75/25)

In this experiment, the training data is partitioned between two institutions such that one contributes 75% of the data and the other contributes the remaining 25% during the training phase. The full set of results for this experiment is placed in Appendix A.2.

From Figure 5.9, we see that the institutional model (B) trained on 25% of the data performs noticeably worse overall compared to the other approaches. On further inspection, looking at Figure 5.8, we see that B performs worse than all the other approaches for every observation apart from Cardiomegaly. This is likely because, unlike the previous experiment where each institution had 50% of the data, the model performance does not saturate on 25% of the data. We meet the minimum requirement for this experiment since all FL approaches outperform B. The best performing FL approach performs within 1.4% of the benchmark, therefore satisfying the requirement that it performs similarly to the centralised model. The institutional model (A) trained on 75% of the data performs closest to the centralised model, marginally outperforming FedProx, by 0.14%. It is unexpected that the Weighted FedAvg approach performs worse than default FedAvg. A possible reason for this is that the weighted approach takes longer to converge. This could not be further investigated because it would have involved running longer experiments. Due to the constraints on the length of job run-times on the GPU cluster, this was not possible.

In Figures 5.10 and 5.11, we compare the overhead when training the models using the different approaches. Surprisingly, FedAvg converged in fewer rounds than FedProx. This is reflected by its lower computational and communication overhead. A similar argument is made for FedAvg compared to Weighted FedAvg. Institution B has a third of the data compared to institution A, hence fewer operations are needed to train the model to convergence. This is also why the institutions in the centralised approach have disparate communication overheads; institution A needs to upload three times more data to the aggregator.

For both experiments involving 2 institutions, we meet all but one of our targets; the best performing FL model still does not outperform all baseline models. We validate our hypothesis that this is due to data saturation by training each institutional model on smaller proportions of data in the proceeding experiments.

5.4 Five Institution Split

In this experiment, the training data is partitioned between five institutions such that each one contributes 20% of the data during the training phase. For ease of interpretation, in this section we only include baseline results for the institutional models with the best and worst performing mean AUC. The full set of results for this experiment, including those of the other institutional

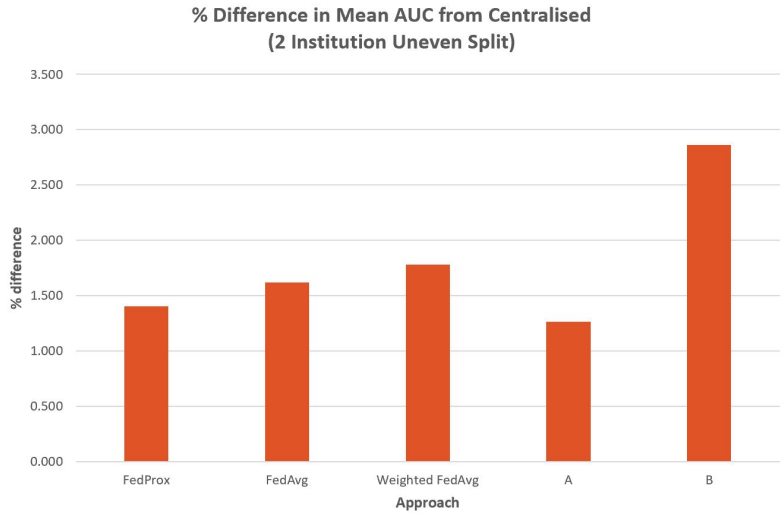


Figure 5.9: Mean AUC % offset from benchmark model - 2 institutions (uneven)

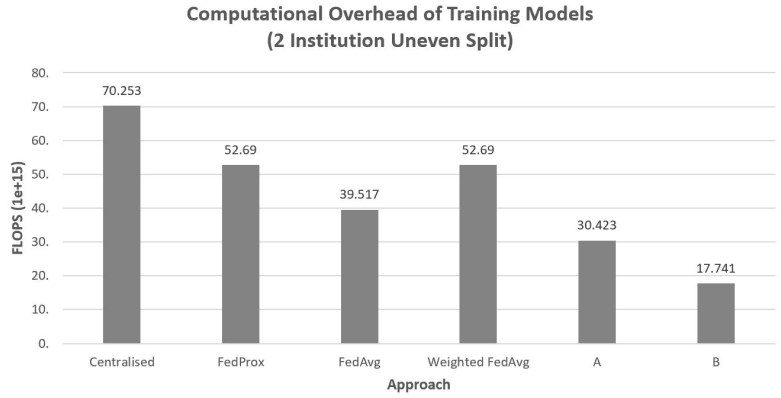


Figure 5.10: Computational overhead using the different approaches - 2 institutions (uneven)

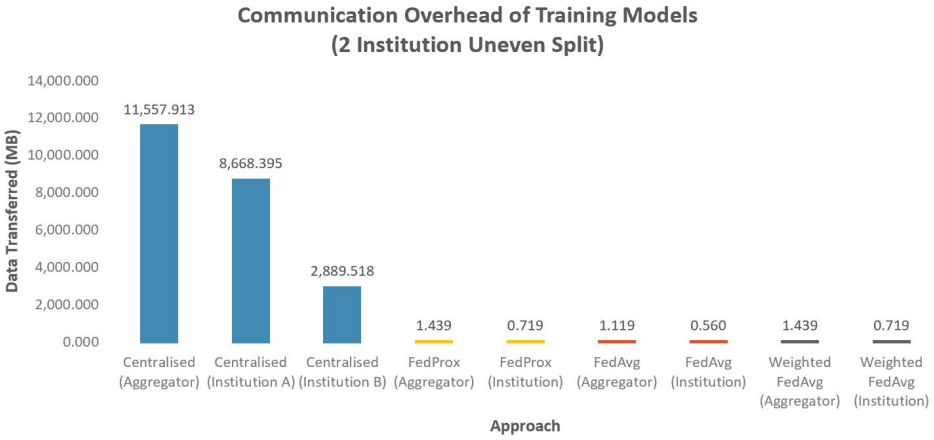


Figure 5.11: Communication overhead using the different approaches - 2 institutions (uneven)

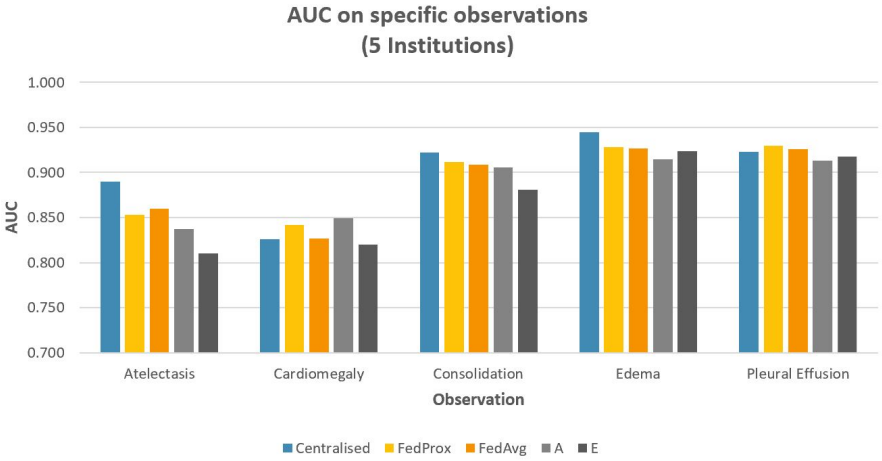


Figure 5.12: AUC on each observation using different approaches - 5 institutions

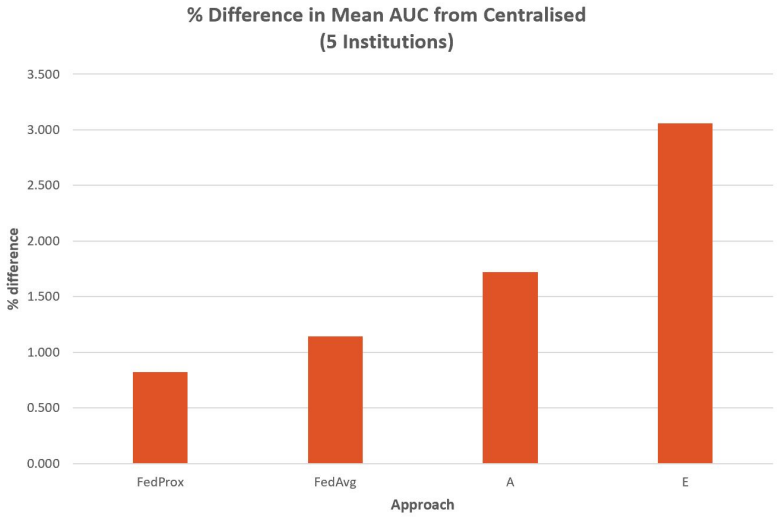


Figure 5.13: Mean AUC % offset from benchmark model - 5 institutions

models, is placed in Appendix A.3.

From Figure 5.13, we see that the FL approaches outperform all the baseline models. On further inspection, Figure 5.12 and Table A.3 show that the FL approaches outperform the baseline models on all observations apart from Cardiomegaly (A). Out of all the experiments, the FedProx model from this investigation performs closest to the benchmark, achieving within 0.8% of the centralised implementation. Because our FL approach outperforms every baseline model, and also performs similarly to the benchmark, we state that all the model performance targets for this experiment have been met.

The overhead of training the models is compared in Figures 5.14 and 5.15. The results are in line with what we expect. The centralised model performs the most computation and has the largest communication overhead - it is trained on the entire dataset. The institutions in the centralised approach have a smaller overhead than in the previous experiments because they contribute less data. However, this is still a significant communication overhead compared to the other approaches. The baseline models perform the least computation - they are trained on a fraction of the dataset. The institutions participating in FL perform more computation than the baseline models because they perform E local rounds of training T times, where T is the number of global communication rounds needed to reach convergence. For reasons already described, FedProx has a lower computational and communication overhead than FedAvg.

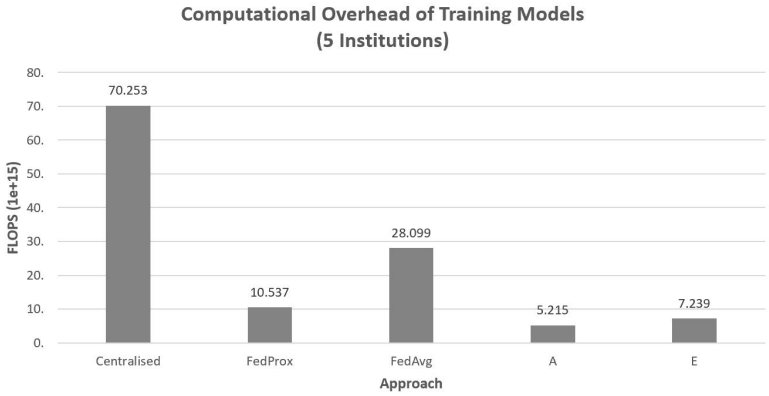


Figure 5.14: Computational overhead using the different approaches - 5 institutions

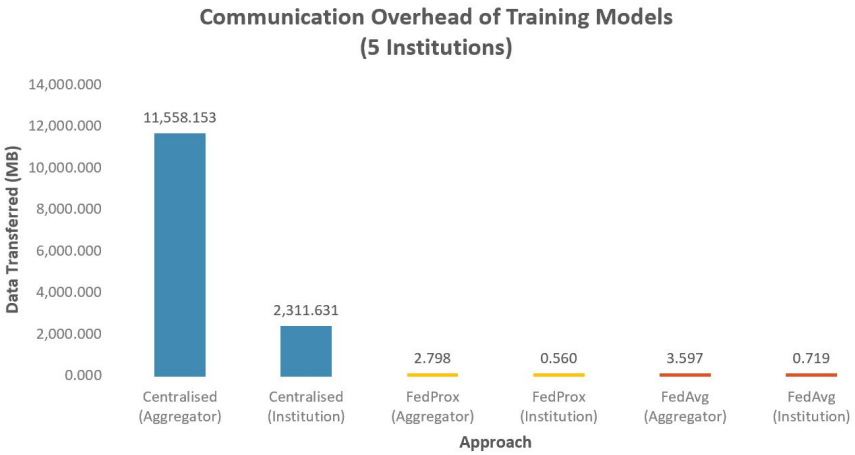


Figure 5.15: Communication overhead using the different approaches - 5 institutions

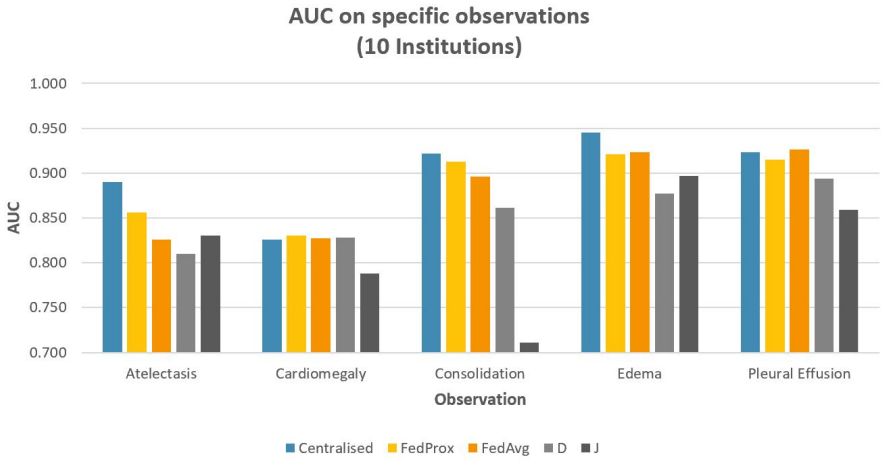


Figure 5.16: AUC on each observation using different approaches - 10 institutions

5.5 Ten Institution Split

In this experiment, we further investigate the effects of increasing the number of participating institutions while simultaneously reducing the amount of data they each contribute. The training data is partitioned between ten institutions such that each one contributes 10% of the data during the training phase. As with the previous experiment, in this section we only include baseline results for the institutional models with the best and worst performing mean AUC. The full set of results for this experiment, including those of the other institutional models, is placed in Appendix A.4.

As with five institutions, Figure 5.17 shows that both FL approaches also outperform all the baseline models in a ten-institution setting. Apart from the mean AUC performance, Figure 5.16 and Table A.4 show that the FedProx outperforms the baseline models on all observations apart from Cardiomegaly (F). FedAvg outperforms the baseline models on all observations apart from Atelectasis (J), Cardiomegaly (F, D) and Consolidation (B). Given that the overall performance of the institutional models degrades when they have less data, it is impressive that the best FL approach achieves within 1.4% of the benchmark.

The overhead of training the models is compared in Figures 5.18 and 5.19. These results follow the same trend as in the five-institution setting. Once again, the institutions in the centralised approach have a smaller, but significant, overhead because they contribute less data. As expected, we also notice that the communication overhead of the FL aggregators increases with more participants. The reason for this is explained in Section 4.5.

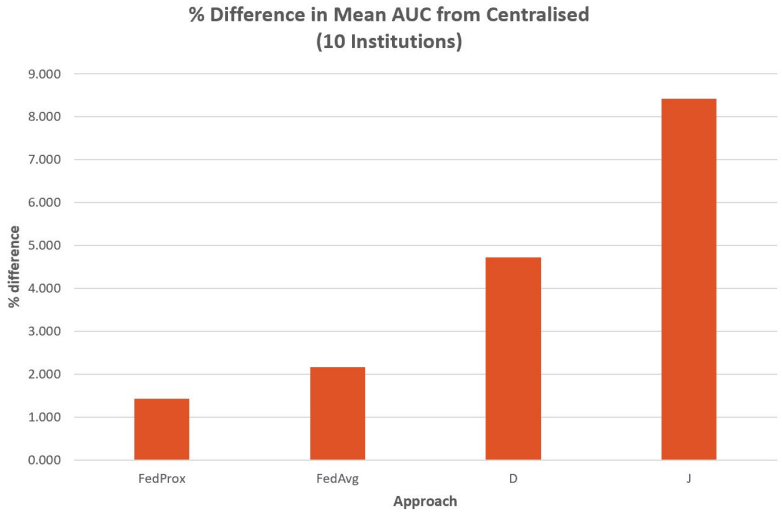


Figure 5.17: Mean AUC % offset from benchmark model - 10 institutions

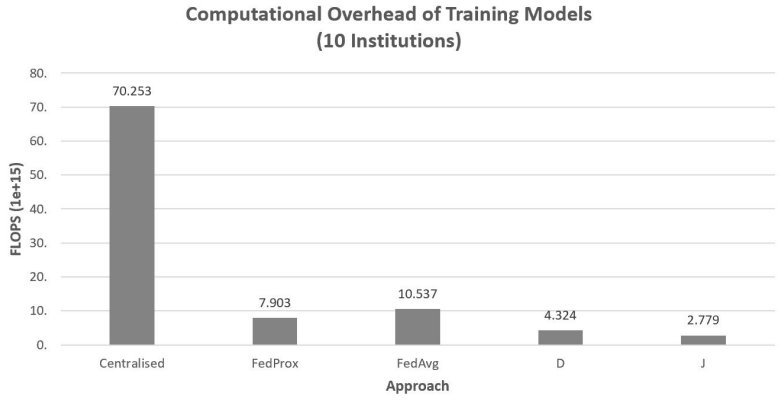


Figure 5.18: Computational overhead using the different approaches - 10 institutions

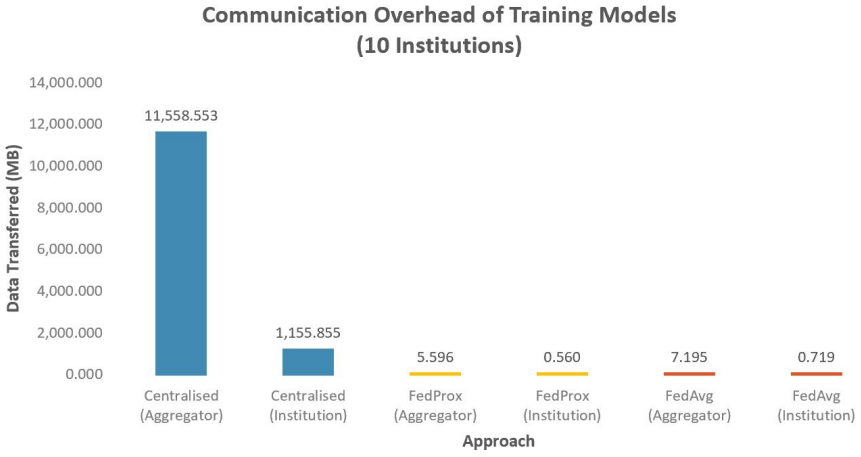


Figure 5.19: Communication overhead using the different approaches - 10 institutions

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In the course of this project, we have successfully implemented a federated learning environment that achieves near state-of-the-art results when trained on a large and distributed dataset of chest radiographs.

This was done by using the implementation of one of the top-5 (at the time of writing) performing CheXpert models as a template for our benchmark. The implementation was then modified to facilitate the training and evaluation of two FL algorithms - FedAvg (including a variant) and FedProx. For the baseline models, the training data was partitioned in a non-i.i.d. manner, to simulate medical institutions having different distributions of observations. We investigated the performance of FL applied to two institutions. When both institutions contributed equal amounts of data, our best performing FL model achieved within 0.015 AUC of the benchmark model. In the case where the contributions were uneven, our best performing FL model achieved within 0.014 AUC of the benchmark. To determine if there was a performance degradation with an increasing number of participants, we investigated the effects of increasing the number of participating institutions, simultaneously reducing the proportion of data they contributed. Although the performance of the baseline models degraded, our best performing FL models achieved within 0.008 AUC (five institutions) and 0.014 AUC (ten institutions) of the benchmark.

As part of the project, we also modelled the computational and communication overhead involved when using the various approaches to train the networks. Our results show that the centralised approach achieves the best model performance, but incurs a substantial computational and communication overhead at the aggregator, and a large communication overhead at the participating institutions. This approach is also the least feasible, due to data privacy regulations. The baseline models incur no communication overhead, and have the lowest computational overhead out of all approaches investigated. However, as previously stated, the performance of these baseline models degrades when the institutions do not have sufficient training data. Factoring overall model performance and the training overhead, our experimental results show that FL is a suitable bridge between the two approaches. Although each FL participant incurs more computational overhead compared to the baseline, it is still much lower than the centralised approach. The FL participants also incur a minor communication overhead. Although the communication overhead of the FL aggregators increases linearly with the number of participating institutions, it is unlikely to be a problem for our proposed use case. Realistically, no more than 100 institutions would be collaborating.

To summarise, we have shown that FL enables large institutions to help smaller ones perform more accurate diagnosis in a privacy-preserving manner. We have also shown that FL enables several small institutions to collaborate and perform diagnosis at higher accuracy than they would be able to achieve individually. There is still a lot of work that needs to be carried out before FL can be adopted in a medical setting. However, our results represent a promising first step and provide a foundation for future work to be built on top of.

6.2 Future Work

Although the excellent experimental results show the project has been successful, there is still room to improve and extend the research. In this section, we elaborate on possible avenues to follow.

6.2.1 Scale up Participating Institutions

For the scope of this project, we assume that no more than ten medical institutions will be collaborating. An interesting experiment would be to investigate the effects on FL model performance and training overhead when the number of FL participants scales to an extra order of magnitude e.g. 100 institutions, using even and uneven splits. We could not perform this investigation ourselves because it would have involved training 100 baseline models. This was not possible due to the competition for department compute resources. Because our project dealt with a small number of participating institutions, 100% of participants were always selected in each federated round of training. Typically, only a fraction are selected. A further investigation could be to determine if performance degrades when institutions with a large amount of data are not regularly selected during training.

6.2.2 Apply FL to Multi-Task Learning Problems

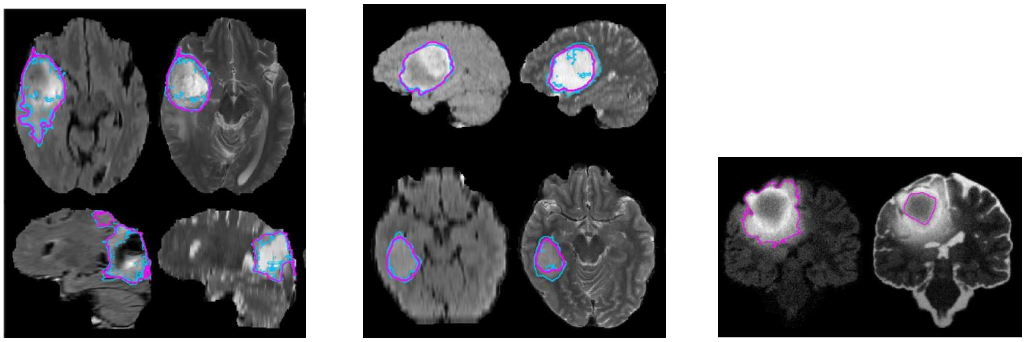
Although we experimented on non-i.i.d. splits of the dataset, each partition always had a sufficient number of samples for each observation. A possible future work could be to partition the CheXpert dataset in such a way that each institution only has positive samples for a particular observation. This could show the relevance of medical institutions that specialise in particular chest observations collaborating to produce a model that can generalise well on all observations. This would require using a different reference model which is capable of performing multi-task learning. Upon inspection of the meta-data, we realise that a lot of scans are positive for more than one observation. To determine if there are enough samples that are only positive for a specific observation, we first apply the labelling heuristic described in Section 3.3 and investigate how many of the samples are only positive for one of the competition observations. To determine the total positive samples for each observation, we apply the labelling heuristic to values reported in Table 3.1. From our analysis, shown in Table 6.1, we conclude that there are sufficient samples with unique observations to perform this experiment. However, due to unfamiliarity with the domain of multi-task learning and also because of a limited time frame, we do not carry out the investigation in the course of this project.

Observation	**Total**	**Unique**
Atelectasis	29333	8833
Cardiomegaly	29599	8768
Consolidation	36706	13128
Edema	48905	12752
Pleural Effusion	86115	36213

Table 6.1: Total vs Unique occurrences of the competition observations

6.2.3 Integrate Formal Privacy Guarantees

Another relevant experiment would have been to investigate the trade-off between data privacy and model performance, through the incorporation of ε-Differential Privacy (ε-DP) and/or Secure Multi-Party Computation (SMPC); concepts previously discussed in Chapter 2.10. Although we wrote our own FL training procedures, if formal privacy guarantees are to be incorporated, it is advised to use one of the frameworks which provide all this functionality. We did not perform this investigation because formal privacy guarantees were beyond the scope of the project. Incorporating ε-DP and SMPC would have also significantly slowed down the training process; our vanilla FL experiments were already on the limit of allocated run-time on the department GPU cluster.



(a) High-grade tumours (b) Low-grade tumours (c) Synthetically generated tumours

Figure 6.1: BraTS whole tumour volume dataset samples [80]

6.2.4 Optimise Training Overhead

Given that incorporating privacy preserving techniques and utilising FedProx significantly slow down the training process in FL, a possible future work could be to reduce the system performance overhead. This would likely involve using a library, such as TensorFlow Federated Core (mentioned in Section 4.1), which provides an API that enables the implementation of low-level optimisations. Due to a strict time frame, and low familiarity with distributed systems and TensorFlow, we agreed that this is beyond the scope of the project. However, our reported system metrics can serve as a reference point for any future optimisations to be evaluated against.

6.2.5 Evaluate Performance on Different Datasets

To verify the generality of the FedProx algorithm on medical imaging data, a possible future work could be to repeat the experiments on the Brain Tumour Segmentation (BraTS) dataset; a collection of multi-modal and multi-institutional MRI scans of 285 subjects with low-grade (non-cancerous) and high-grade (cancerous) brain tumours, as shown in Figure 6.1. The blue outlines represent the individual experts' annotations and the magenta lines represent the consensus segmentation. There are no individual annotations for 6.1c, as they represent the synthetic cases generated by software. As a starting point, a potential FL solution could be trained on this version of U-Net [1] which has been modified [104] to maximise brain tumour segmentation performance.

Although this is a segmentation task, we do not expect that modifications would need to be made to the FedProx algorithm. The main challenge of using BraTS is that it only consists of 285 images. In a federated healthcare scenario, if one institution has a large portion of the data, it will be easy for the other institutions to overfit on their significantly smaller amounts of data, if trained using the baseline approach. Performance will likely degrade with more institutions added, since each will have even fewer training images. Our hypothesis is that an FL model would generalise well across all of the different institutions' data. During the project, we decided against evaluating FedProx on the BraTS dataset due to the unexpected competition for compute resources on the department GPU cluster. Instead, we focused solely on CheXpert to ensure a more comprehensive set of standalone results.

[1] `https://github.com/pykao/Modified-3D-UNet-Pytorch`

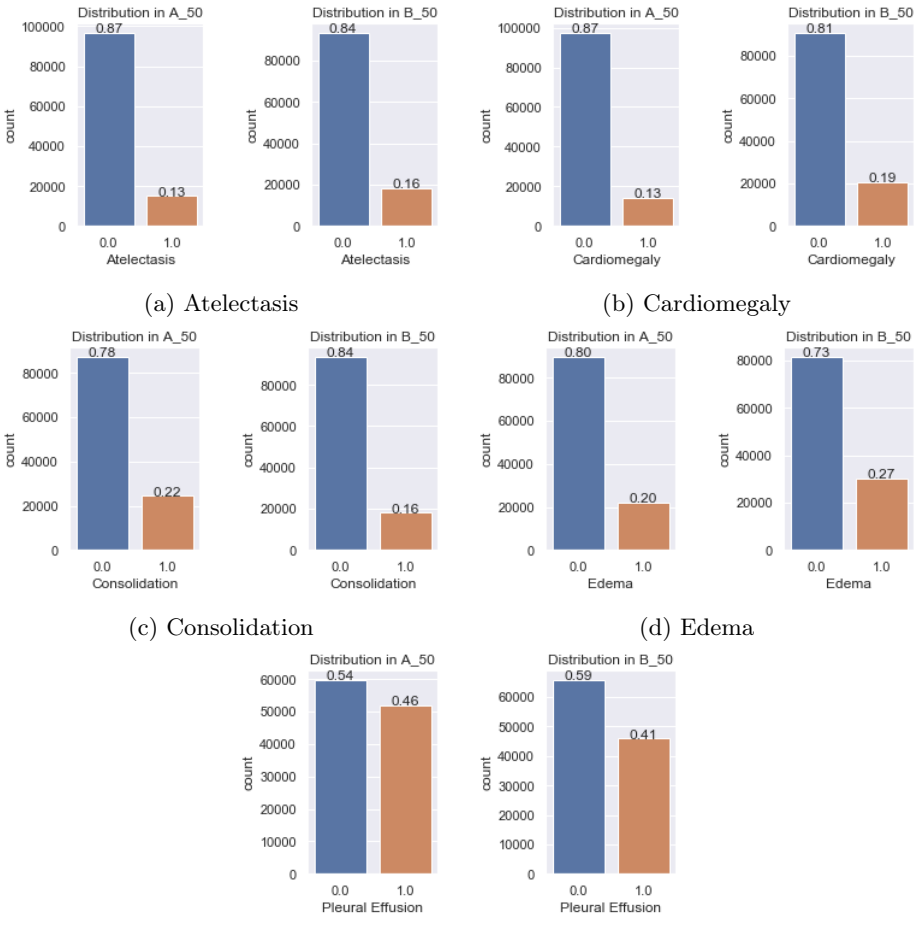
Appendix A

Remaining Analysis and Results

This section contains the distribution of labels between partitions and ROC-AUC metrics for all the experiments run as part of the project. They are placed here, instead of the main body of the report, to facilitate easier reading, while providing the same level of transparency about our experimental results. The bold AUC values in the tables represent the best score achieved for an observation across all **non-centralised** approaches.

A.1 Two Institution Even Split

A.1.1 Distribution of Labels



(a) Atelectasis

(b) Cardiomegaly

(c) Consolidation

(d) Edema

(e) Pleural Effusion

Figure A.1: Distribution of labels between partitions in 2 institution even split

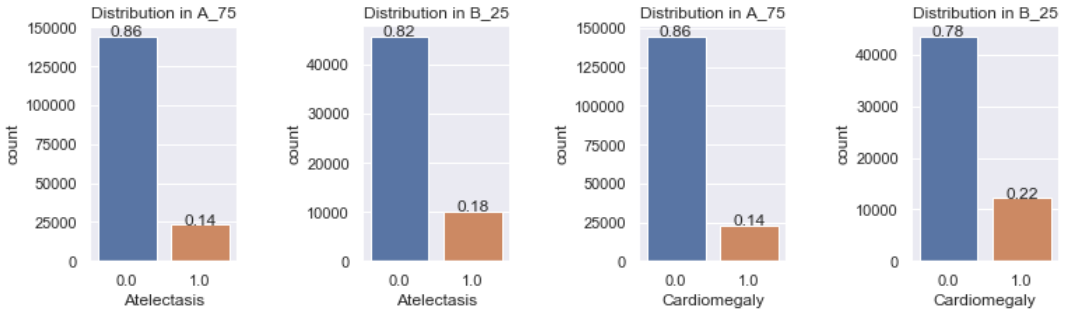
A.1.2 Model Performance

	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion
Centralised	0.890	0.826	0.922	0.945	0.923
FedProx	0.833	0.823	0.908	0.936	**0.931**
FedAvg	0.830	0.817	0.911	**0.940**	0.926
A (50%)	**0.843**	0.822	0.893	0.938	0.929
B (50%)	0.824	**0.844**	**0.914**	0.931	0.924

Table A.1: AUC on each observation using different approaches - 2 institution (even)

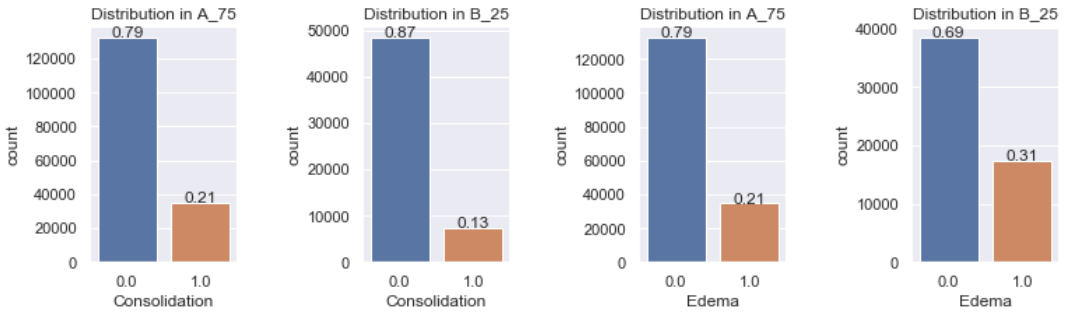
A.2 Two Institution Uneven Split

A.2.1 Distribution of Labels



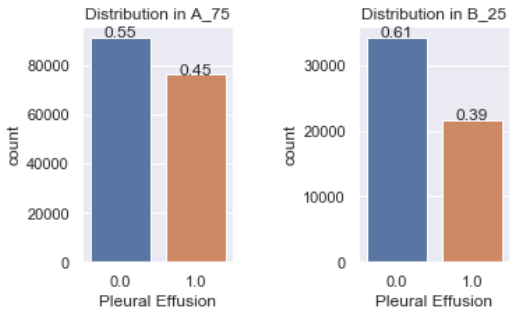
(a) Atelectasis

(b) Cardiomegaly



(c) Consolidation

(d) Edema



(e) Pleural Effusion

Figure A.2: Distribution of labels between partitions in 2 institution uneven split

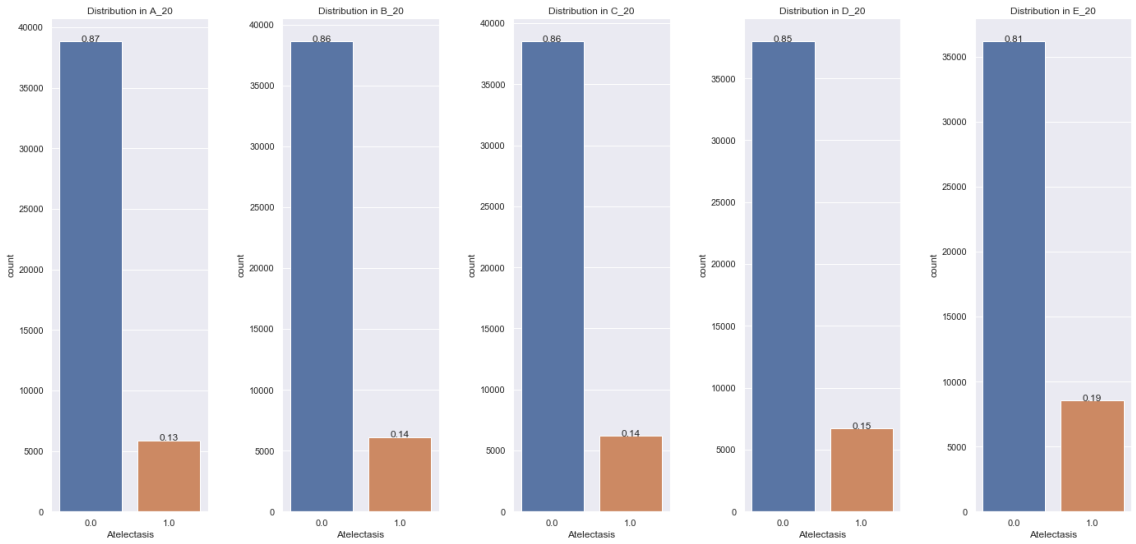
A.2.2 Model Performance

	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion
Centralised	0.890	0.826	0.922	0.945	0.923
FedProx	0.835	0.816	**0.916**	0.939	**0.930**
FedAvg	**0.836**	0.812	0.912	0.936	0.929
Weighted FedAvg	0.823	0.834	0.901	0.929	**0.930**
A (75%)	0.834	0.827	**0.916**	**0.942**	0.924
B (25%)	0.788	**0.838**	0.897	0.925	0.915

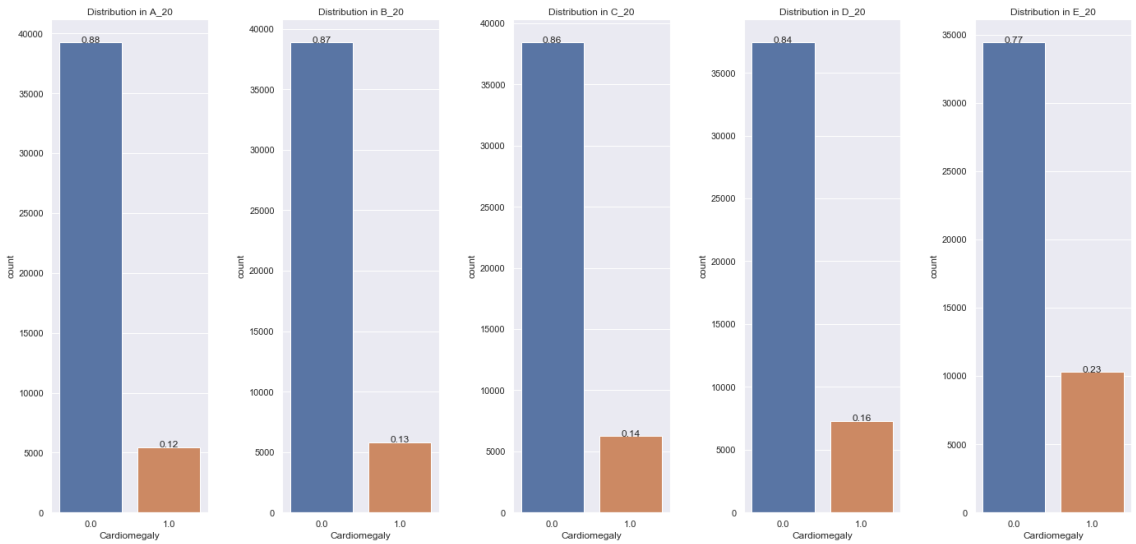
Table A.2: AUC on each observation using different approaches - 2 institution (uneven)

A.3 Five Institution Split

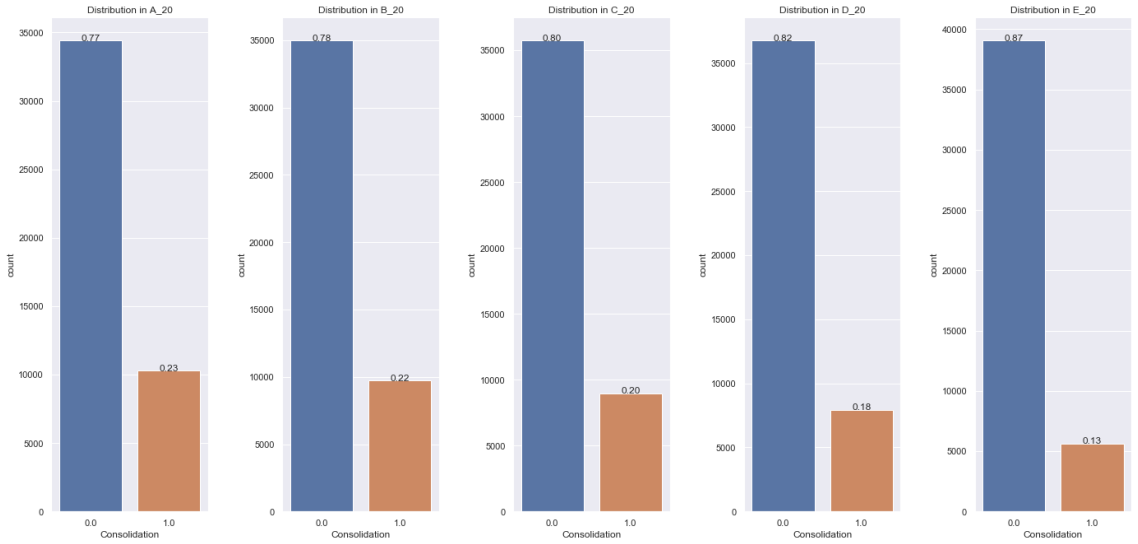
A.3.1 Distribution of Labels



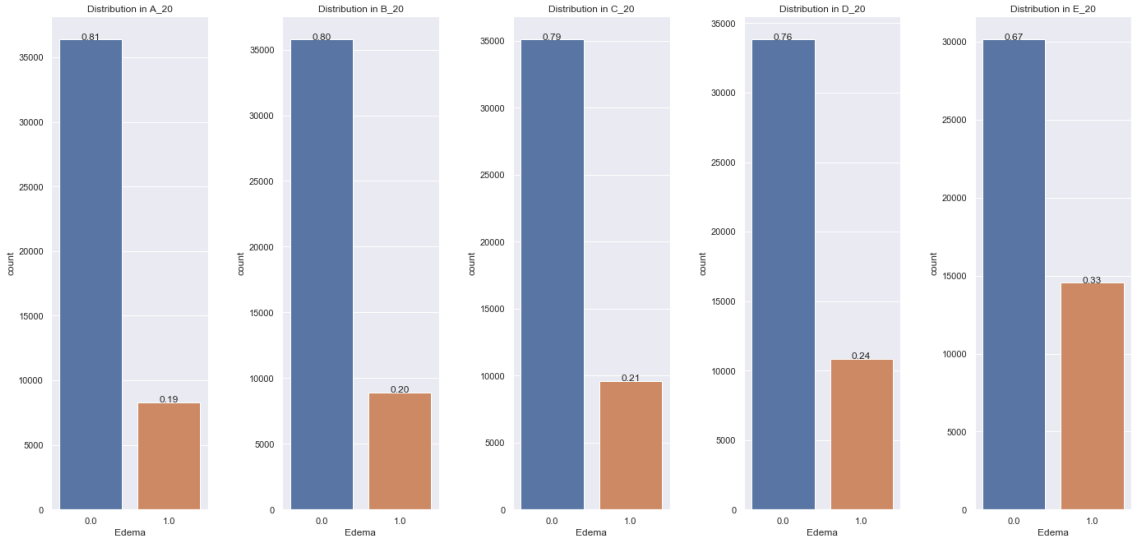
(a) Atelectasis



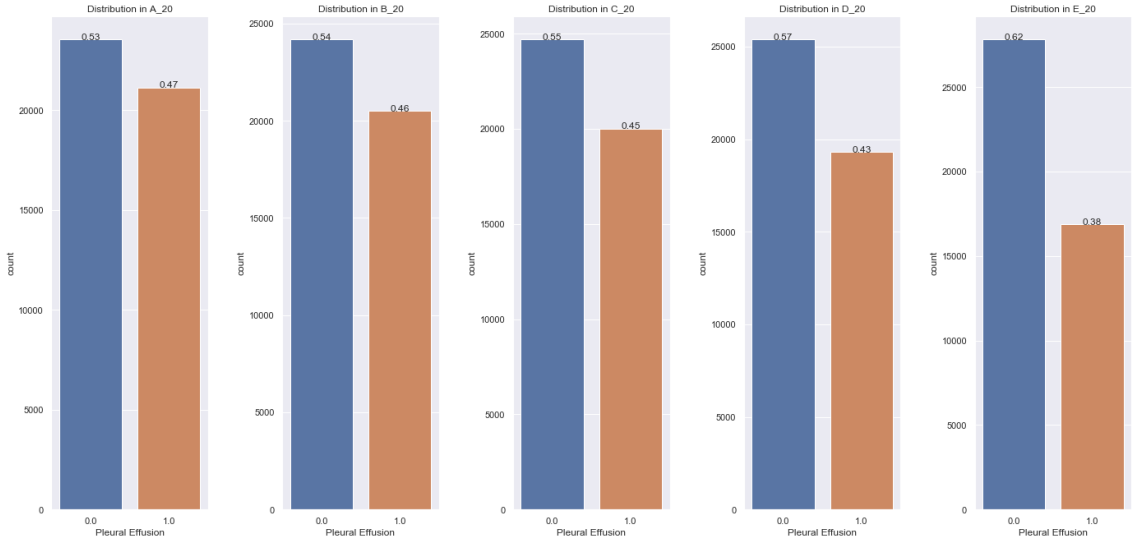
(b) Cardiomegaly



(c) Consolidation



(d) Edema



(e) Pleural Effusion

Figure A.3: Distribution of labels between partitions in 5 institution split

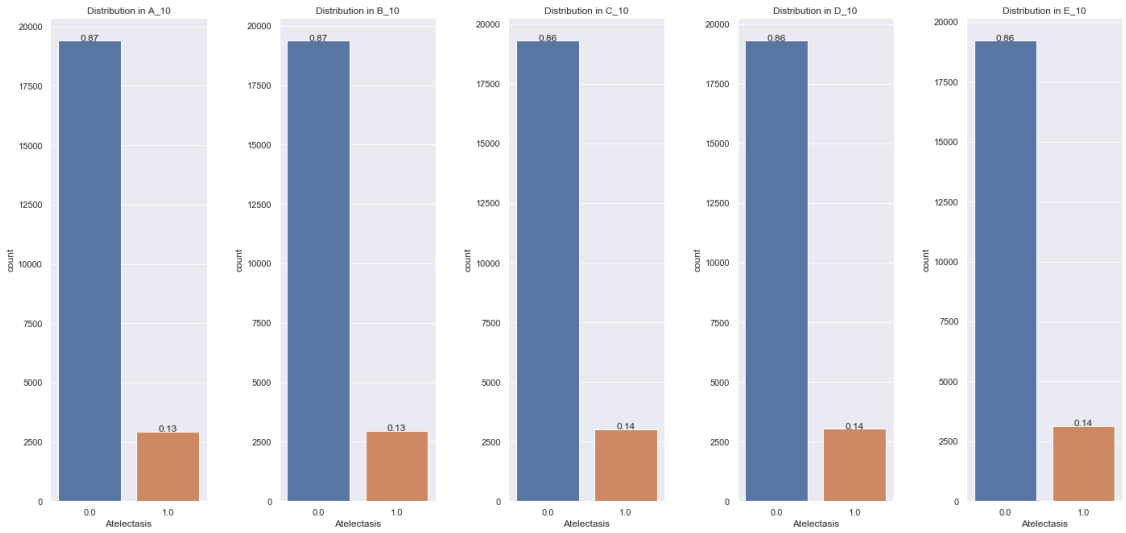
A.3.2 Model Performance

	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion
Centralised	0.890	0.826	0.922	0.945	0.923
FedProx	0.853	0.842	**0.912**	**0.928**	**0.930**
FedAvg	**0.860**	0.827	0.909	0.927	0.926
A (20%)	0.837	**0.849**	0.906	0.915	0.913
B (20%)	0.820	0.814	0.892	0.918	0.912
C (20%)	0.828	0.821	0.905	0.910	0.924
D (20%)	0.828	0.809	0.868	0.927	0.917
E (20%)	0.810	0.820	0.881	0.924	0.918

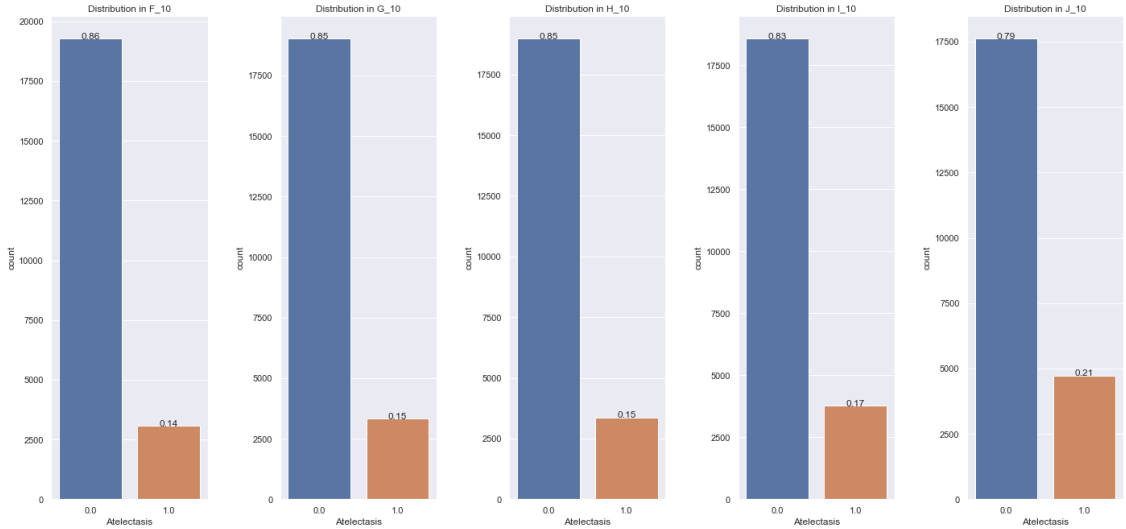
Table A.3: AUC on each observation using different approaches - 5 institutions

A.4 Ten Institution Split

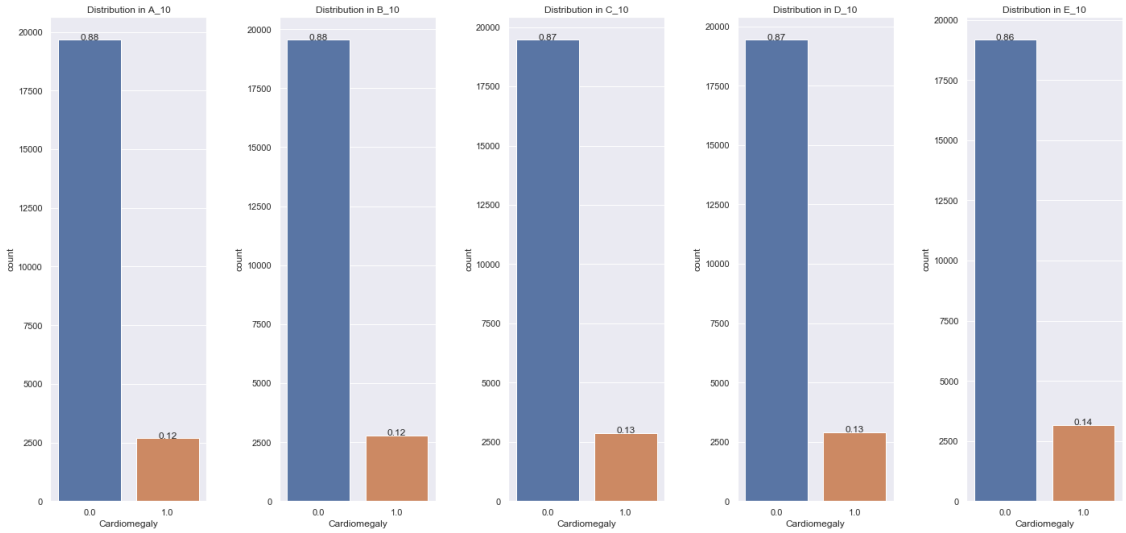
A.4.1 Distribution of Labels



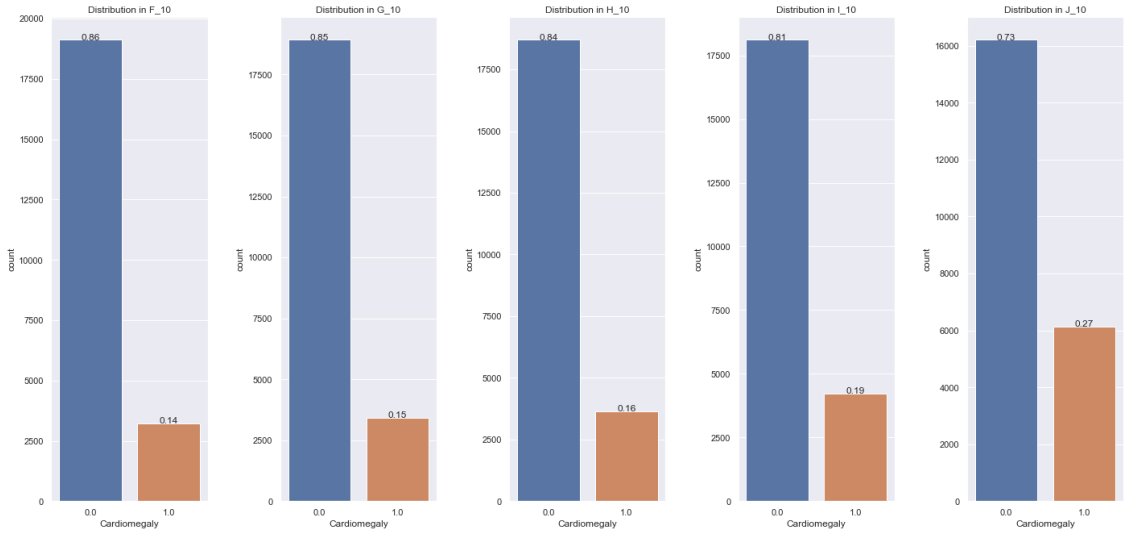
(a.1) Atelectasis: A - E



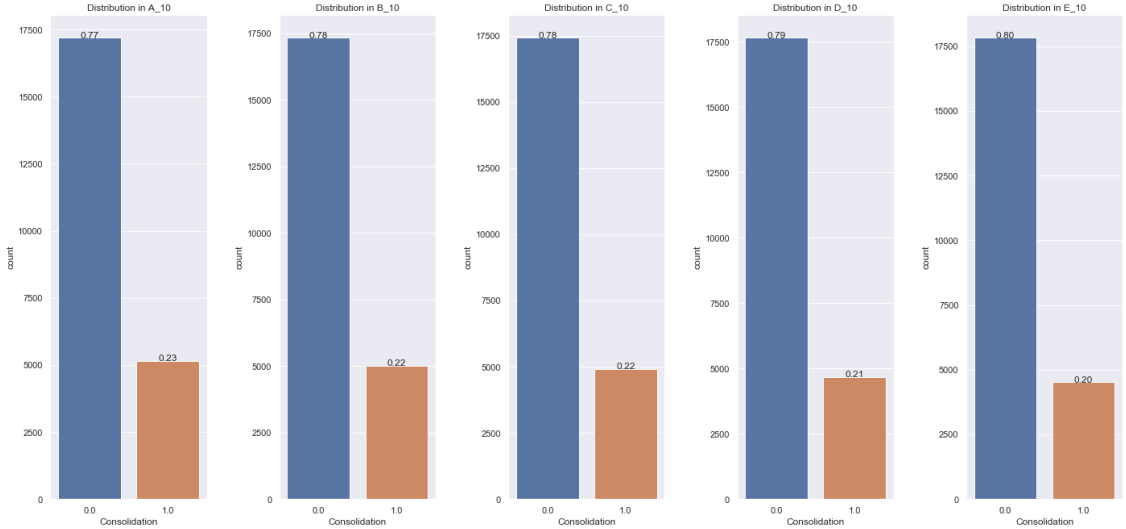
(a.2) Atelectasis: F - J



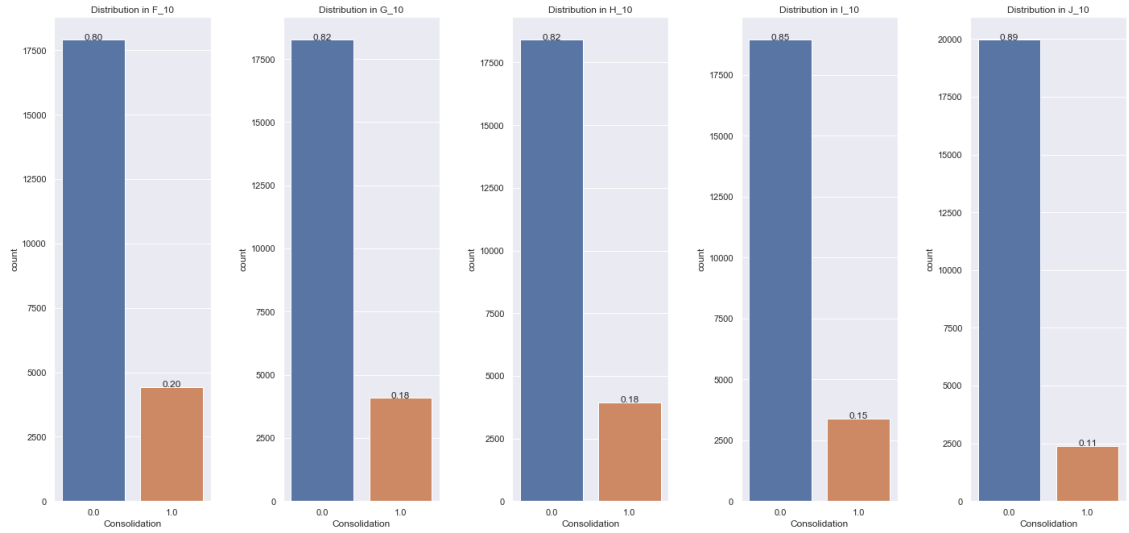
(b.1) Cardiomegaly: A - E



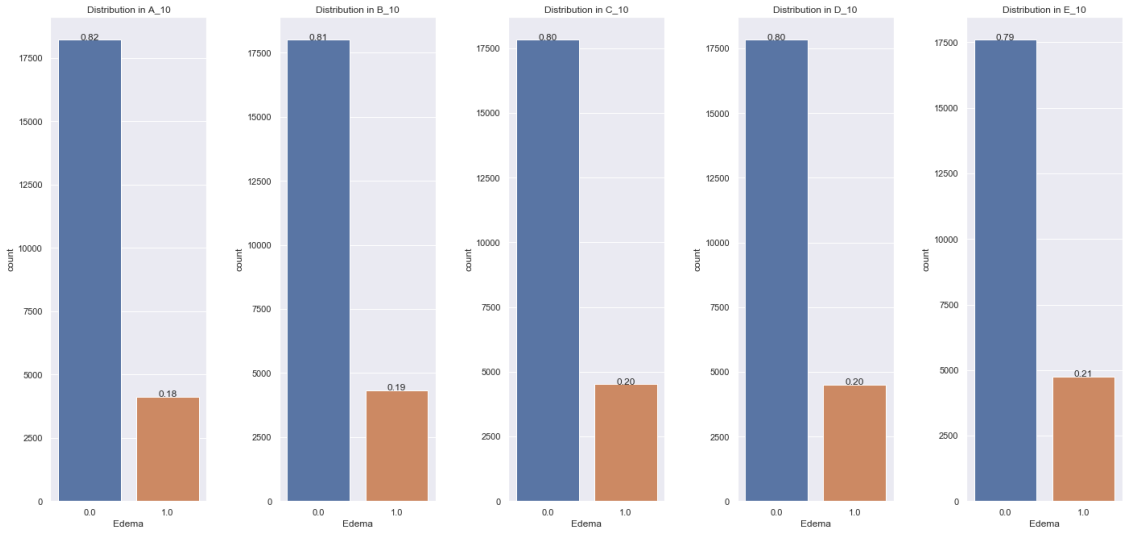
(b.2) Cardiomegaly: F - J



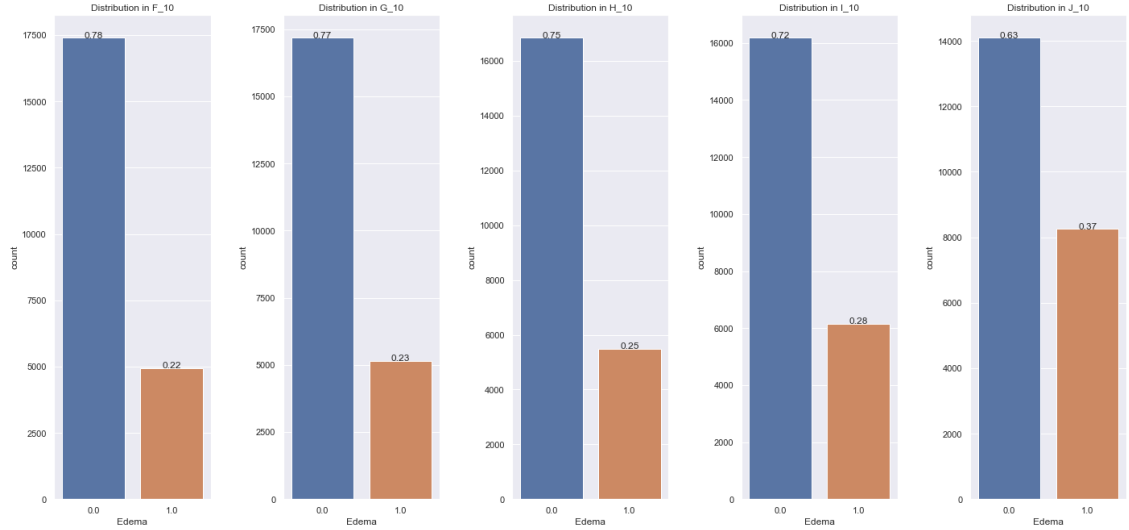
(c.1) Consolidation: A - E



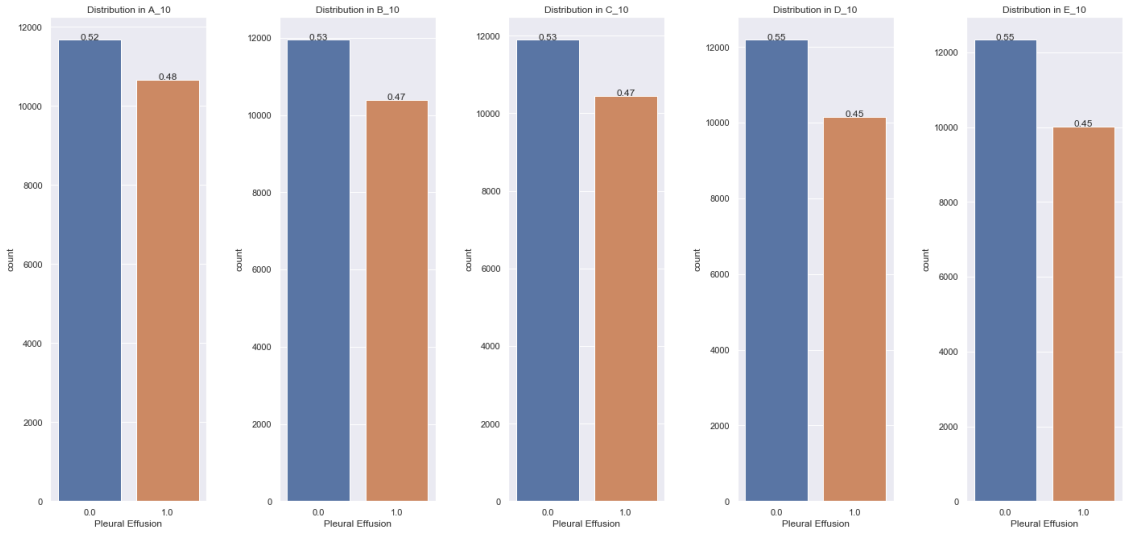
(c.2) Consolidation: F - J



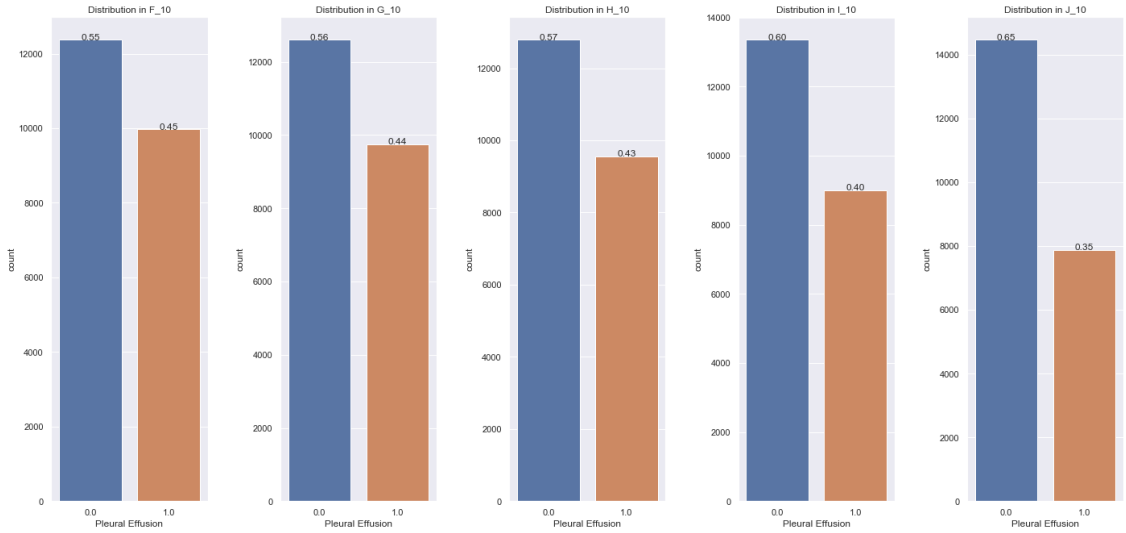
(d.1) Edema: A - E



(d.2) Edema: F - J



(e.1) Pleural Effusion: A - E



(e.2) Pleural Effusion: F - J

Figure A.4: Distribution of labels between partitions in 10 institution split

A.4.2 Model Performance

	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion
Centralised	0.890	0.826	0.922	0.945	0.923
FedProx	**0.856**	0.830	**0.913**	0.921	0.915
FedAvg	0.826	0.827	0.896	**0.923**	**0.926**
A (10%)	0.779	0.753	0.850	0.885	0.867
B (10%)	0.803	0.816	0.901	0.901	0.847
C (10%)	0.730	0.724	0.889	0.888	0.873
D (10%)	0.810	0.828	0.861	0.877	0.894
E (10%)	0.738	0.772	0.859	0.896	0.840
F (10%)	0.745	**0.831**	0.873	0.895	0.894
G (10%)	0.800	0.778	0.870	0.897	0.845
H (10%)	0.789	0.768	0.876	0.891	0.841
I (10%)	0.735	0.818	0.834	0.911	0.871
J (10%)	0.830	0.788	0.711	0.897	0.859

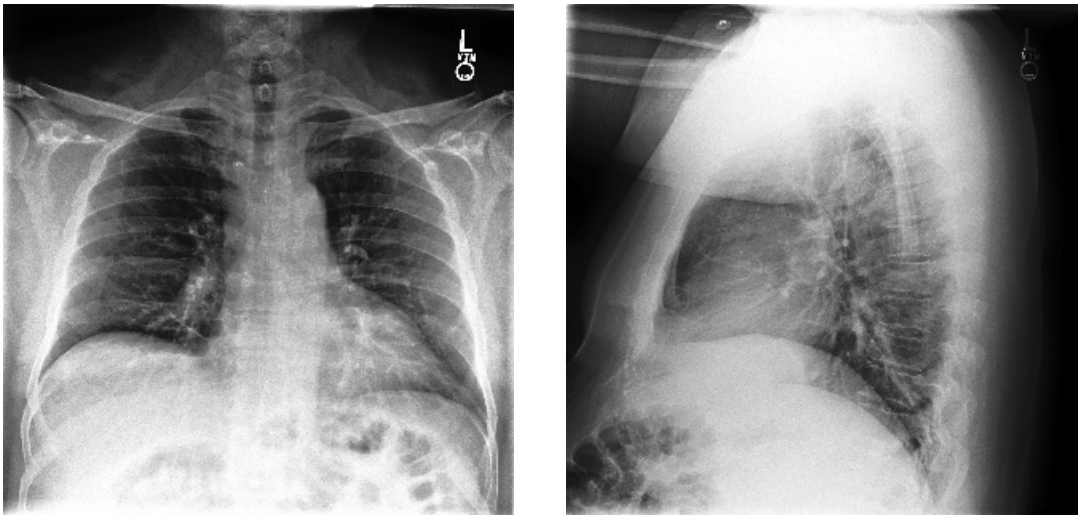
Table A.4: AUC on each observation using different approaches - 10 institutions

Appendix B

Additional Chest Radiograph Samples

B.1 Atelectasis

These are the multi-view scans of a 53 year old male. The PA technique was used to obtain the frontal view.

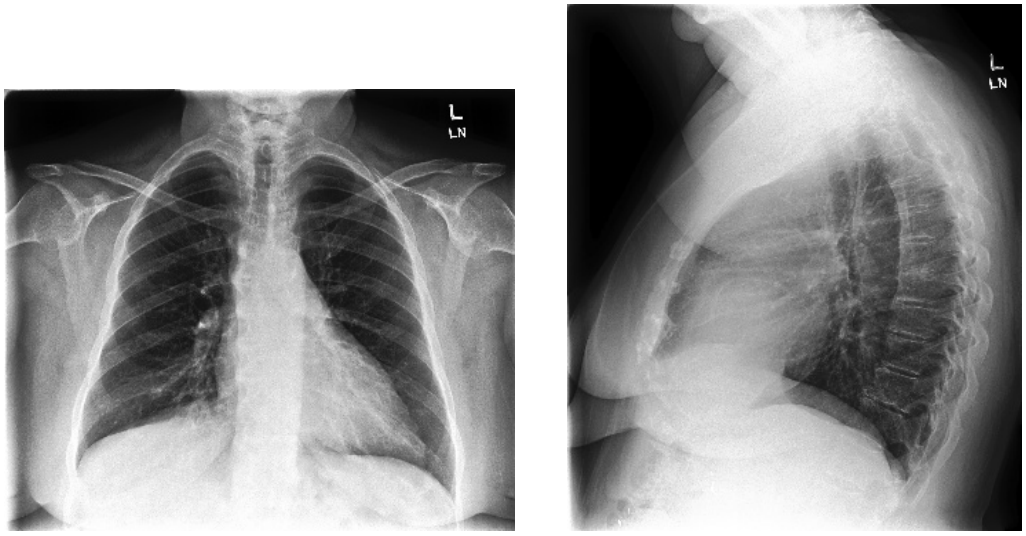


(a) Frontal (b) Lateral

Figure B.1: Chest radiographs of different Atelectasis patient

B.2 Cardiomegaly

These are the multi-view scans of a 64 year old female. The PA technique was used to obtain the frontal view.



(a) Frontal (b) Lateral

Figure B.2: Chest radiographs of different Cardiomegaly patient

B.3 Consolidation

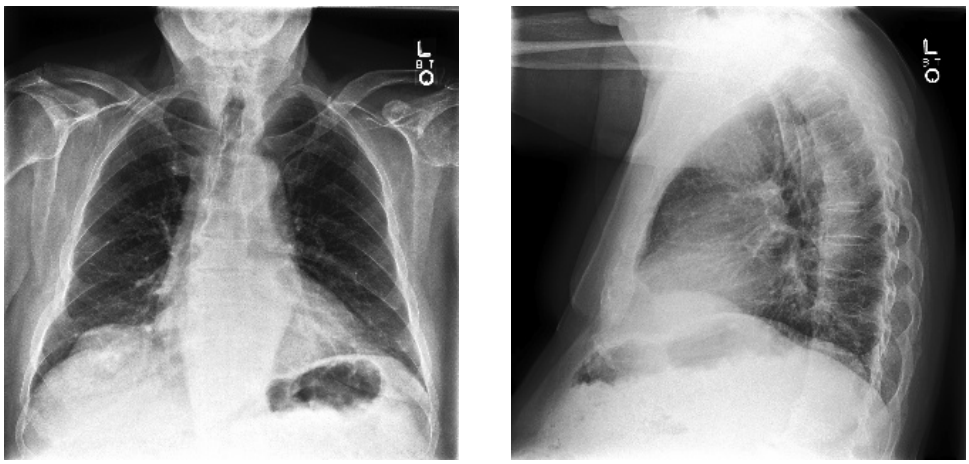
These are the multi-view scans of a 51 year old female. The PA technique was used to obtain the frontal view.

(a) Frontal (b) Lateral

Figure B.3: Chest radiographs of different Consolidation patient

B.4 Edema

These are the multi-view scans of a 88 year old male. The PA technique was used to obtain the frontal view.

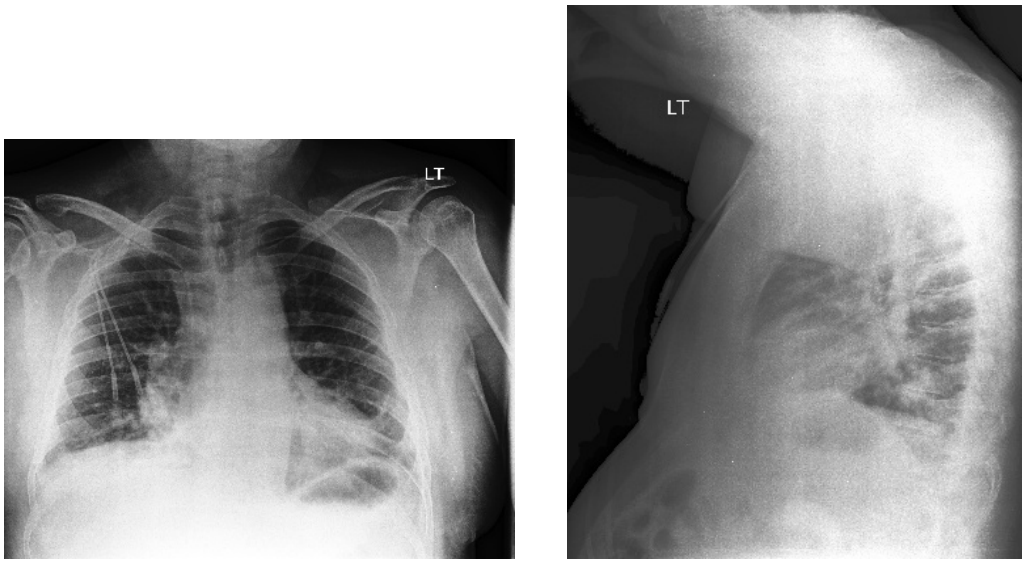


(a) Frontal (b) Lateral

Figure B.4: Chest radiographs of different Edema patient

B.5 Pleural Effusion

These are the multi-view scans of a 46 year old male. The PA technique was used to obtain the frontal view.



(a) Frontal (b) Lateral

Figure B.5: Chest radiographs of different Pleural Effusion patient

Bibliography

[1] Richard M. Levenson et al. "Pigeons (Columba livia) as Trainable Observers of Pathology and Radiology Breast Cancer Images". en. In: *PLOS ONE* 10.11 (Nov. 2015), e0141357. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0141357. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0141357 (visited on 01/02/2020).

[2] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". en. In: *Nature* 542.7639 (Feb. 2017), pp. 115–118. ISSN: 1476-4687. DOI: 10.1038/nature21056. URL: https://www.nature.com/articles/nature21056 (visited on 01/02/2020).

[3] Turgay Ayer et al. "Breast cancer risk estimation with artificial neural networks revisited". en. In: *Cancer* 116.14 (July 2010), pp. 3310–3321. ISSN: 1097-0142. DOI: 10.1002/cncr.25081. URL: https://acsjournals.onlinelibrary.wiley.com/doi/10.1002/cncr.25081 (visited on 01/02/2020).

[4] Jacob Stern. *The Fragmentation of Health Data – Datavant*. URL: https://datavant.com/2018/08/01/the-fragmentation-of-health-data/ (visited on 01/02/2020).

[5] Oren Frank. "Opinion | Donate Your Health Care Data Today". en-US. In: *The New York Times* (Oct. 2019). ISSN: 0362-4331. URL: https://www.nytimes.com/2019/10/02/opinion/health-care-data-privacy.html (visited on 01/02/2020).

[6] Foundation Wikipedia. *General Data Protection Regulation*. en. Page Version ID: 933500662. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=General_Data_Protection_Regulation&oldid=933500662 (visited on 01/02/2020).

[7] Foundation Wikipedia. *Health Insurance Portability and Accountability Act*. en. Page Version ID: 932269315. Dec. 2019. URL: https://en.wikipedia.org/w/index.php?title=Health_Insurance_Portability_and_Accountability_Act&oldid=932269315 (visited on 01/02/2020).

[8] Jeremy Irvin et al. "CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison". In: *arXiv:1901.07031 [cs, eess]* (Jan. 2019). arXiv: 1901.07031. URL: http://arxiv.org/abs/1901.07031 (visited on 01/02/2020).

[9] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *arXiv:1409.0575 [cs]* (Jan. 2015). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575 (visited on 01/17/2020).

[10] Tao Zeng, Bian Wu, and Shuiwang Ji. "DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation". en. In: *Bioinformatics* 33.16 (Aug. 2017), pp. 2555–2562. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx188. URL: https://academic.oup.com/bioinformatics/article/33/16/2555/3096435 (visited on 01/17/2020).

[11] Amazon. *Amazon Go*. URL: https://www.amazon.com/b?ie=UTF8&node=16008589011 (visited on 01/18/2020).

[12] Alex Moshakis. "Nation of shoplifters: the rise of supermarket self-checkout scams". en-GB. In: *The Observer* (May 2018). ISSN: 0029-7712. URL: https://www.theguardian.com/global/2018/may/20/nation-of-shoplifters-supermarket-self-checkout (visited on 01/18/2020).

[13] Waymo. *Waymo - We're building the World's Most Experienced Driver*. en. URL: https://waymo.com/ (visited on 01/18/2020).

[14] Tesla. *Autopilot*. en-GB. URL: https://www.tesla.com/autopilot (visited on 01/18/2020).

[15] VolvoTrucks. *Automated Trucks | Volvo Trucks*. English. URL: https://www.volvotrucks.com/en-en/about-us/automation.html (visited on 01/18/2020).

[16] EricssonIOT. *Self-driving buses in Stockholm, Sweden*. en. Jan. 2018. URL: https://www.ericsson.com/en/internet-of-things/trending/driverless-buses-in-stockholm-sweden (visited on 01/18/2020).

[17] WHO. *Road traffic injuries*. en. Dec. 2018. URL: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (visited on 01/18/2020).

[18] Google. *Nest Cam IQ Outdoor - Outdoor Home CCTV System - Google Store*. URL: https://store.google.com/gb/product/nest_cam_iq_outdoor (visited on 01/18/2020).

[19] Gauss. *Triton^{TM} AI-enabled platform for real time monitoring of surgical blood loss*. en-US. 2019. URL: http://www.gausssurgical.com (visited on 01/18/2020).

[20] Scott Mayer McKinney et al. "International evaluation of an AI system for breast cancer screening". In: *Nature* 577.7788 (Jan. 2020), pp. 89–94. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1799-6. URL: https://doi.org/10.1038/s41586-019-1799-6.

[21] J. Anthony Seibert. *Archiving, Chapter 2: Medical Image Data Characteristics - Society for Imaging Informatics in Medicine*. en. URL: https://siim.org/page/archiving_chapter2 (visited on 01/16/2020).

[22] Isaac Bankman. *Handbook of Medical Imaging: Processing and Analysis Management; pp772*. en. 2nd ed. Academic Press, Oct. 2000. ISBN: 978-0-08-053310-0.

[23] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *arXiv:1511.08458 [cs]* (Dec. 2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458 (visited on 06/12/2020).

[24] Adit Deshpande. *A Beginner's Guide To Understanding Convolutional Neural Networks*. July 2016. URL: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/ (visited on 01/15/2020).

[25] Chigozie Nwankpa et al. "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". In: *arXiv:1811.03378 [cs]* (Nov. 2018). arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378 (visited on 01/17/2020).

[26] Andrej Karpathy. *CS231n Convolutional Neural Networks for Visual Recognition*. 2015. URL: http://cs231n.github.io/convolutional-networks/ (visited on 01/16/2020).

[27] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167 [cs]* (Mar. 2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167 (visited on 05/29/2020).

[28] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv:1810.04805 [cs]* (May 2019). arXiv: 1810.04805 version: 2. URL: http://arxiv.org/abs/1810.04805 (visited on 05/30/2020).

[29] Jie Hu et al. "Squeeze-and-Excitation Networks". en. In: *arXiv:1709.01507 [cs]* (May 2019). arXiv: 1709.01507. URL: http://arxiv.org/abs/1709.01507 (visited on 05/30/2020).

[30] Yan Kang. *Further Attention Utilization – Efficience & Segmentation*. Library Catalog: sisyphus.gitbook.io. 2019. URL: https://sisyphus.gitbook.io/project/deep-learning-basics/classification/further-attention-utilization-classification-and-segmentation (visited on 05/30/2020).

[31] Jun Fu et al. "Dual Attention Network for Scene Segmentation". In: *arXiv:1809.02983 [cs]* (Apr. 2019). arXiv: 1809.02983. URL: http://arxiv.org/abs/1809.02983 (visited on 05/30/2020).

[32] Hanchao Li et al. "Pyramid Attention Network for Semantic Segmentation". In: *arXiv:1805.10180 [cs]* (Nov. 2018). arXiv: 1805.10180. URL: http://arxiv.org/abs/1805.10180 (visited on 05/30/2020).

[33] Ben Glocker. *Lecture 2: Introduction to Machine Learning. Presented at Imperial College London*. en. Jan. 2020. (Visited on 01/14/2020).

[34] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 01/17/2020).

[35] Ahmed Abdelbaki. *Computer Vision Lab SS16 - P-CNN features for Action Recognition*. Oct. 2016. DOI: 10.13140/RG.2.2.14789.86247.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (visited on 01/18/2020).

[37] Christian Szegedy et al. "Going Deeper with Convolutions". In: *arXiv:1409.4842 [cs]* (Sept. 2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842 (visited on 01/17/2020).

[38] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *arXiv:1512.00567 [cs]* (Dec. 2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567 (visited on 01/18/2020).

[39] Christian Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *arXiv:1602.07261 [cs]* (Aug. 2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261 (visited on 01/18/2020).

[40] Gao Huang et al. "Densely Connected Convolutional Networks". In: *arXiv:1608.06993 [cs]* (Jan. 2018). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993 (visited on 01/18/2020).

[41] Milad Salem, Shayan Taheri, and Jiann-Shiun Yuan. "Utilizing Transfer Learning and Homomorphic Encryption in a Privacy Preserving and Secure Biometric Recognition System". In: *Computers* 8 (Dec. 2018), p. 3. DOI: 10.3390/computers8010003.

[42] Vladimir Iglovikov and Alexey Shvets. "TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation". In: *arXiv:1801.05746 [cs]* (Jan. 2018). arXiv: 1801.05746. URL: http://arxiv.org/abs/1801.05746 (visited on 01/19/2020).

[43] Ozgun Cicek et al. "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation". In: *arXiv:1606.06650 [cs]* (June 2016). arXiv: 1606.06650. URL: http://arxiv.org/abs/1606.06650 (visited on 01/19/2020).

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 01/19/2020).

[45] Amit P. Sheth and James A. Larson. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases". In: *ACM Comput. Surv.* 22.3 (Sept. 1990), pp. 183–236. ISSN: 0360-0300. DOI: 10.1145/96602.96604. URL: https://doi.org/10.1145/96602.96604.

[46] Tobias Kurze et al. "Cloud federation. Cloud Computing". In: (2011), pp. 32–38.

[47] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *arXiv:1908.07873 [cs, stat]* (Aug. 2019). arXiv: 1908.07873. URL: http://arxiv.org/abs/1908.07873 (visited on 01/09/2020).

[48] Qinbin Li et al. "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection". In: *arXiv:1907.09693 [cs, stat]* (Dec. 2019). arXiv: 1907.09693. URL: http://arxiv.org/abs/1907.09693 (visited on 01/09/2020).

[49] Pedro García López et al. "Edge-centric Computing: Vision and Challenges". In: *Computer Communication Review* 45 (2015), pp. 37–42.

[50] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In: *MCC '12*. 2012.

[51] Tsvi Kuflik, Judy Kay, and Bob Kummerfeld. "Challenges and solutions of ubiquitous user modeling". In: *Ubiquitous display environments*. Springer, 2012, pp. 7–30.

[52] Abhijit Guha Roy et al. "BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning". In: *arXiv:1905.06731 [cs, stat]* (May 2019). arXiv: 1905.06731. URL: http://arxiv.org/abs/1905.06731 (visited on 01/02/2020).

[53] Keith Bonawitz et al. "Towards Federated Learning at Scale: System Design". en. In: *arXiv:1902.01046 [cs, stat]* (Mar. 2019). arXiv: 1902.01046. URL: http://arxiv.org/abs/1902.01046 (visited on 01/02/2020).

[54] Hannah Kuchler. *Pharma groups combine to promote drug discovery with AI*. en-GB. June 2019. URL: https://www.ft.com/content/ef7be832-86d0-11e9-a028-86cea8523dc2 (visited on 01/20/2020).

[55] Micah J. Sheller et al. "Multi-Institutional Deep Learning Modeling Without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation". In: *arXiv:1810.04304 [cs, stat]* (Oct. 2018). arXiv: 1810.04304. URL: http://arxiv.org/abs/1810.04304 (visited on 01/02/2020).

[56] Wenqi Li et al. "Privacy-preserving Federated Brain Tumour Segmentation". In: *arXiv:1910.00962 [cs]* (Oct. 2019). arXiv: 1910.00962. URL: http://arxiv.org/abs/1910.00962 (visited on 01/02/2020).

[57] Praneeth Vepakomma et al. "Split learning for health: Distributed deep learning without sharing raw patient data". In: *arXiv:1812.00564 [cs, stat]* (Dec. 2018). arXiv: 1812.00564. URL: http://arxiv.org/abs/1812.00564 (visited on 01/02/2020).

[58] Jiawen Kang et al. "Incentive Design for Efficient Federated Learning in Mobile Networks: A Contract Theory Approach". In: *arXiv:1905.07479 [cs]* (Oct. 2019). arXiv: 1905.07479. URL: http://arxiv.org/abs/1905.07479 (visited on 01/20/2020).

[59] H. Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *arXiv:1602.05629 [cs]* (Feb. 2017). arXiv: 1602.05629. URL: http://arxiv.org/abs/1602.05629 (visited on 01/20/2020).

[60] Tian Li et al. "Federated Optimization in Heterogeneous Networks". In: *arXiv:1812.06127 [cs, stat]* (Dec. 2018). arXiv: 1812.06127. URL: http://arxiv.org/abs/1812.06127 (visited on 01/20/2020).

[61] Hanlin Tang et al. "Communication Compression for Decentralized Training". In: *arXiv:1803.06443 [cs, stat]* (Jan. 2019). arXiv: 1803.06443. URL: http://arxiv.org/abs/1803.06443 (visited on 01/20/2020).

[62] Raphael Bost et al. "Machine learning classification over encrypted data." In: vol. 4324. 2015, p. 4325. URL: https://eprint.iacr.org/2014/331.pdf.

[63] Matej Mikulic. *Diagnostic imaging market share top medtech companies 2017 and 2024*. en. Aug. 2019. URL: https://www.statista.com/statistics/331739/top-global-companies-by-diagnostic-imaging-market-share/ (visited on 01/21/2020).

[64] Olivier Potvin et al. "Measurement Variability Following MRI System Upgrade". English. In: *Frontiers in Neurology* 10 (2019). ISSN: 1664-2295. DOI: 10.3389/fneur.2019.00726. URL: https://www.frontiersin.org/articles/10.3389/fneur.2019.00726/full (visited on 01/21/2020).

[65] Reza Shokri et al. "Membership Inference Attacks against Machine Learning Models". In: *arXiv:1610.05820 [cs, stat]* (Mar. 2017). arXiv: 1610.05820. URL: http://arxiv.org/abs/1610.05820 (visited on 01/22/2020).

[66] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning". In: *arXiv:1702.07464 [cs, stat]* (Sept. 2017). arXiv: 1702.07464. URL: http://arxiv.org/abs/1702.07464 (visited on 01/22/2020).

[67] Luca Melis et al. "Exploiting Unintended Feature Leakage in Collaborative Learning". In: *arXiv:1805.04049 [cs]* (Nov. 2018). arXiv: 1805.04049. URL: http://arxiv.org/abs/1805.04049 (visited on 01/02/2020).

[68] Arjun Nitin Bhagoji. *Lecture: IBM Research - Model Poisoning Attacks in Federated Learning. Presented at Princeton University*. 2018. URL: http://www.princeton.edu/~abhagoji/files/SecML_2018_fed_learn_poison.pdf (visited on 01/22/2020).

[69] Eugene Bagdasaryan et al. "How To Backdoor Federated Learning". In: *arXiv:1807.00459 [cs]* (Aug. 2019). arXiv: 1807.00459. URL: http://arxiv.org/abs/1807.00459 (visited on 01/22/2020).

[70] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. "Machine learning with adversaries: Byzantine tolerant gradient descent". In: *Advances in Neural Information Processing Systems*. 2017, pp. 119–129. URL: https://papers.nips.cc/paper/6617-machine-learning-with-adversaries-byzantine-tolerant-gradient-descent.pdf.

[71] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3–4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[72] Stacey Truex et al. "A Hybrid Approach to Privacy-Preserving Federated Learning". In: *arXiv:1812.03224 [cs, stat]* (Aug. 2019). arXiv: 1812.03224. URL: http://arxiv.org/abs/1812.03224 (visited on 01/02/2020).

[73] Naranker Dulay. *Lecture: Privacy Engineering Part II. Presented at Imperial College London*. Nov. 2019. (Visited on 01/22/2020).

[74] *Zama*. en. 2019. URL: https://zama.ai/ (visited on 01/22/2020).

[75] Stephen Hardy et al. "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption". In: *arXiv:1711.10677 [cs]* (Nov. 2017). arXiv: 1711.10677. URL: http://arxiv.org/abs/1711.10677 (visited on 01/22/2020).

[76] V. Nikolaenko et al. "Privacy-Preserving Ridge Regression on Hundreds of Millions of Records". In: *2013 IEEE Symposium on Security and Privacy*. May 2013, pp. 334–348. ISBN: 978-0-7695-4977-4. DOI: 10.1109/SP.2013.30.

[77] M. Sabt, M. Achemlal, and A. Bouabdallah. "Trusted Execution Environment: What It is, and What It is Not". In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. Aug. 2015, pp. 57–64. ISBN: 978-1-4673-7952-6. DOI: 10.1109/Trustcom.2015.357.

[78] J. Ekberg, K. Kostiainen, and N. Asokan. "The Untapped Potential of Trusted Execution Environments on Mobile Devices". In: *IEEE Security & Privacy* 12.4 (Aug. 2014), pp. 29–37. ISSN: 1558-4046. DOI: 10.1109/MSP.2014.38.

[79] Zhongshu Gu et al. "Reaching data confidentiality and model accountability on the caltrain". In: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 336–348. URL: https://arxiv.org/abs/1812.03230.

[80] Bjoern H. Menze et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)". In: *IEEE Transactions on Medical Imaging* 34.10 (Oct. 2015), pp. 1993–2024. ISSN: 1558-254X. DOI: 10.1109/TMI.2014.2377694.

[81] Spyridon Bakas et al. "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features". eng. In: *Scientific Data* 4 (2017), p. 170117. ISSN: 2052-4463. DOI: 10.1038/sdata.2017.117.

[82] Spyridon Bakas et al. "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge". In: *arXiv:1811.02629 [cs, stat]* (Apr. 2019). arXiv: 1811.02629. URL: http://arxiv.org/abs/1811.02629 (visited on 01/07/2020).

[83] Otkrist Gupta and Ramesh Raskar. "Distributed learning of deep neural network over multiple agents". In: *arXiv:1810.06060 [cs, stat]* (Oct. 2018). arXiv: 1810.06060. URL: http://arxiv.org/abs/1810.06060 (visited on 01/23/2020).

[84] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[85] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16*. 2016, pp. 265–283.

[86] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[87] Thomas Kluyver et al. "Jupyter Notebooks – a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press, 2016, pp. 87–90.

[88] Sebastian Caldas et al. "LEAF: A Benchmark for Federated Settings". In: *arXiv:1812.01097 [cs, stat]* (Dec. 2019). arXiv: 1812.01097. URL: http://arxiv.org/abs/1812.01097 (visited on 01/02/2020).

[89] Ligeng Zhu. *Lyken17/pytorch-OpCounter*. original-date: 2018-01-26T06:20:22Z. June 2020. URL: https://github.com/Lyken17/pytorch-OpCounter (visited on 06/11/2020).

[90] Jeremy Jones. *Atelectasis (summary) | Radiology Reference Article | Radiopaedia.org*. en-GB. Library Catalog: radiopaedia.org. URL: https://radiopaedia.org/articles/atelectasis-summary (visited on 06/10/2020).

[91] Radswiki. *Cardiomegaly | Radiology Reference Article | Radiopaedia.org*. en-GB. Library Catalog: radiopaedia.org. URL: https://radiopaedia.org/articles/cardiomegaly?lang=gb (visited on 06/10/2020).

[92] Frank Gaillard. *Air space opacification | Radiology Reference Article | Radiopaedia.org*. en-GB. Library Catalog: radiopaedia.org. URL: https://radiopaedia.org/articles/air-space-opacification-1?lang=gb (visited on 06/10/2020).

[93] Yuranga Weerakkody. *Pulmonary oedema | Radiology Reference Article | Radiopaedia.org*. en-GB. Library Catalog: radiopaedia.org. URL: https://radiopaedia.org/articles/pulmonary-oedema?lang=gb (visited on 06/10/2020).

[94] Jeremy Jones. *Pleural effusion (summary) | Radiology Reference Article | Radiopaedia.org*. en-GB. Library Catalog: radiopaedia.org. URL: https://radiopaedia.org/articles/pleural-effusion-summary?lang=gb (visited on 06/10/2020).

[95] *CheXpert: A Large Dataset of Chest X-Rays and Competition for Automated Chest X-Ray Interpretation*. URL: https://stanfordmlgroup.github.io/competitions/chexpert/ (visited on 01/02/2020).

[96] Sarang Narkhede. *Understanding AUC - ROC Curve*. en. Library Catalog: towardsdatascience.com. May 2019. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (visited on 06/09/2020).

[97] Ye Wenwu et al. *jfhealthcare/Chexpert*. original-date: 2019-11-03T09:10:37Z. June 2020. URL: https://github.com/jfhealthcare/Chexpert (visited on 06/06/2020).

[98] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[99] Google. *TensorFlow Federated*. en. Library Catalog: www.tensorflow.org. URL: https://www.tensorflow.org/federated (visited on 06/12/2020).

[100] FedAI. *An Industrial Grade Federated Learning Framework*. zh-CN. Library Catalog: fate.fedai.org. URL: https://fate.fedai.org/ (visited on 06/12/2020).

[101] Nvidia. *NVIDIA Clara*. en. Library Catalog: developer.nvidia.com. May 2019. URL: https://developer.nvidia.com/clara (visited on 06/12/2020).

[102] Theo Ryffel et al. "A generic framework for privacy preserving deep learning". In: *arXiv:1811.04017 [cs, stat]* (Nov. 2018). arXiv: 1811.04017. URL: http://arxiv.org/abs/1811.04017 (visited on 01/02/2020).

[103] Machine Learning Department University Carnegie Mellon. *Federated Learning: Challenges, Methods, and Future Directions*. en-US. Nov. 2019. URL: https://blog.ml.cmu.edu/2019/11/12/federated-learning-challenges-methods-and-future-directions/ (visited on 01/20/2020).

[104] Fabian Isensee et al. "Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge". In: *arXiv:1802.10508 [cs]* (Feb. 2018). arXiv: 1802.10508 version: 1. URL: http://arxiv.org/abs/1802.10508 (visited on 01/19/2020).