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 8, 2020

Abstract

Ready for review

In deep learning, machine learning models typically require large and representative training sets to perform tasks well. If hospitals and clinics aggregated 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 the 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

Completed

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, the project 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.

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

Ready for review

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 do 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 5.1), 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 Section 5.3.

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 scenarios. We first experiment on even partitions i.e. each institution having 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 4 and analyse the results of these experiments in Sections 6.2 and 6.3.

3. **Experiment with larger number of institutions:** To determine if increasing the number of participating institutions degrades FL model performance, we repeat 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 6.4 and 6.5.

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 assumptions, are explained in Sections 3.5, 4.6 and 5.5.

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 by utilising FL.

Chapter 2

Background

Ready for review

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. The chapter concludes with an analysis of existing state-of-the-art approaches to privacy-preserving deep learning and the dataset we will evaluate our proposed solutions on.

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. The vehicles are able to manoeuvre through 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 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.

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 we are starting to see computer vision algorithms outperform human experts. Just this year, Google Health announced that they have 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 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, 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 created by stacking three main 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

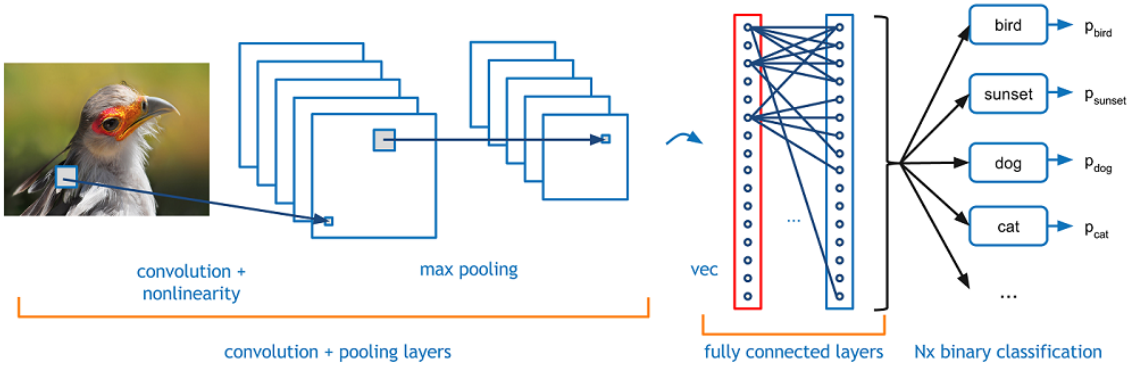


Figure 2.1: Typical CNN architecture [24]

Convolution + Activation Layer

Convolution 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

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. Typically, Rectified Linear Unit (ReLU) is used as the activation function.

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

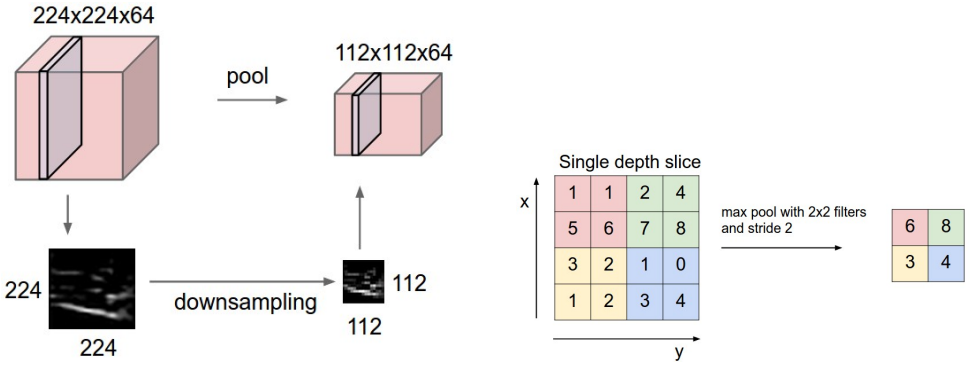
Using other activation functions, such as tanh and sigmoid (detailed in B.1), which map their input to a small output range can introduce the vanishing gradient problem. This is occurs particularly when multiple layers, each using these activation functions, are stacked together. The first layer maps 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 (vanishing) [25]. Hence, the network is unable to train effectively.
Output Dimension: h x w x k

Pooling Layer

This layer is typically placed between successive convolutional layers to gradually reduce the size of the 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 2 x 2 pooling filter)

Fully Connected Layer

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, as shown in Figure 2.1. As with regular MLPs, each neuron in this layer will be connected to all the neurons in the previous one. This layer will



(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]

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]

then produce c class scores to be used for 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 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 and 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 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 will

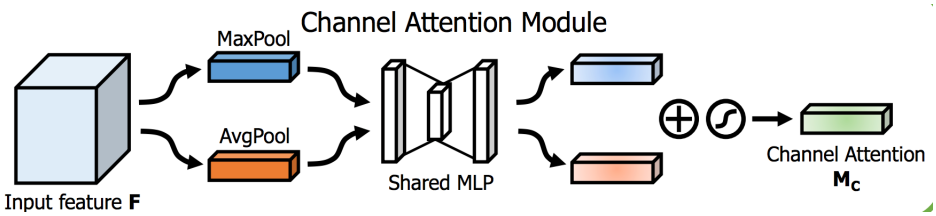


Figure 2.4: Channel Attention Module, adapted from [30]

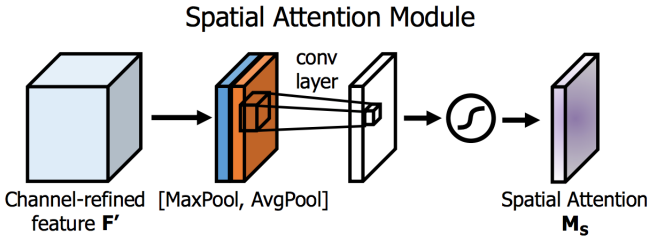


Figure 2.5: Spatial Attention Module, adapted from [30]

be used 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 and lose pixel-wise information. 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.

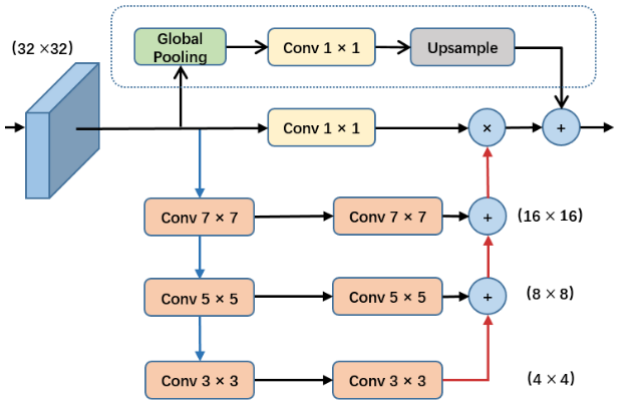


Figure 2.6: Feature Pyramid Attention Module, adapted from [32]

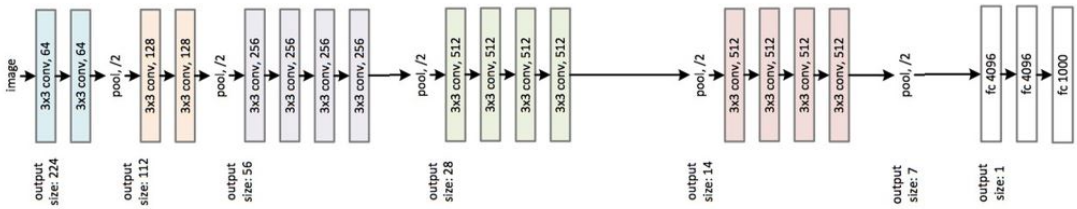


Figure 2.7: VGG-19 architecture, adapted from [35]

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

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]:

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

2.4.2 Popular architecture

The following are CNN-based networks that have achieved state-of-the-art performance on image classification tasks. These networks will also form the building blocks of our federated solutions.

VGGNet

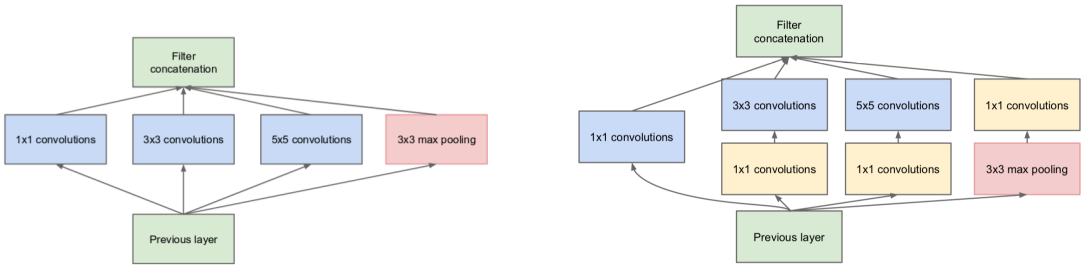
Many other models are built on top of VGGNet[34] or use the idea of representing filters with multiple, smaller ones to cover the same effective area. There are several variants of VGGNet, which differ by the total number of layers the network uses. We describe VGG-19, which has 16 convolutional layers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



(a) Naive version (b) Dimension reductions

Figure 2.8: Inception Module [37]

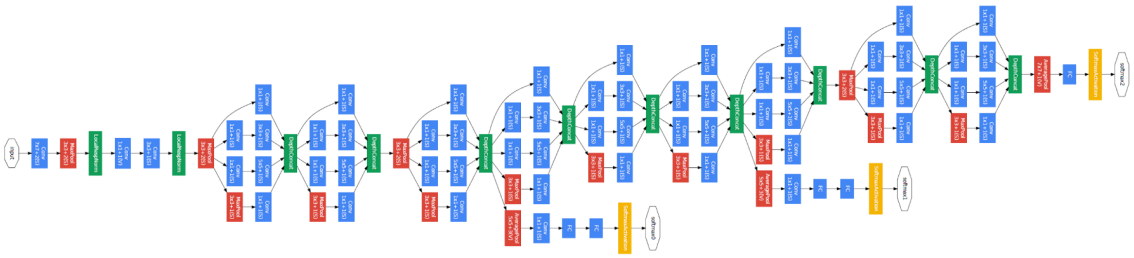


Figure 2.9: GoogLeNet/Inception [37]

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 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

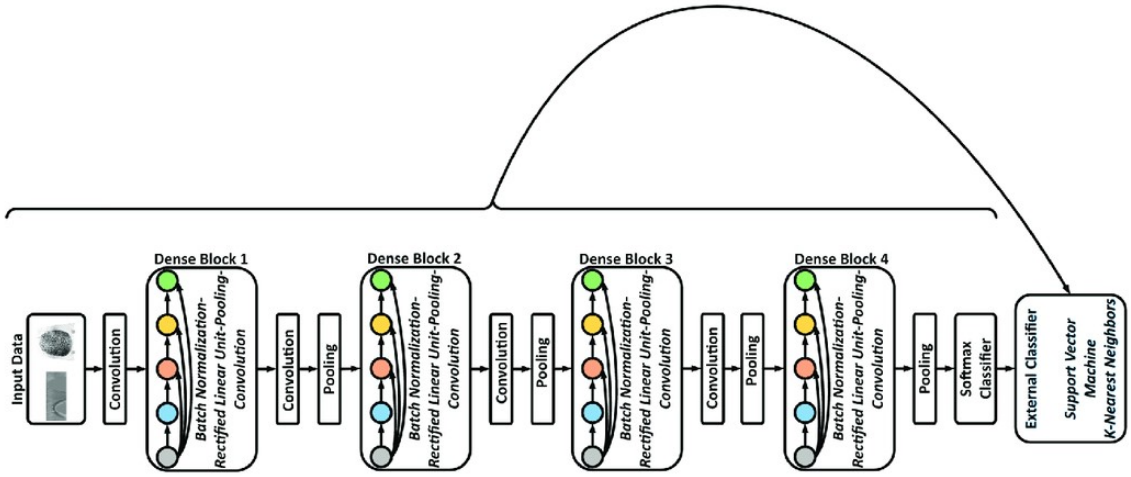


Figure 2.10: DenseNet, adapted from [41]

account, 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: A mask of the image, representing what our model should predict in the segmentation.
Upsampling: Technique to increase size of an image.
Transposed convolution: Converting an image from low- to high-resolution. Done by taking transpose of filter to reverse the convolution.

Goal

Given an image as input, the output is an image, 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 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 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 gradually increases, as the depth decreases. The positional information is recovered through upsampling.

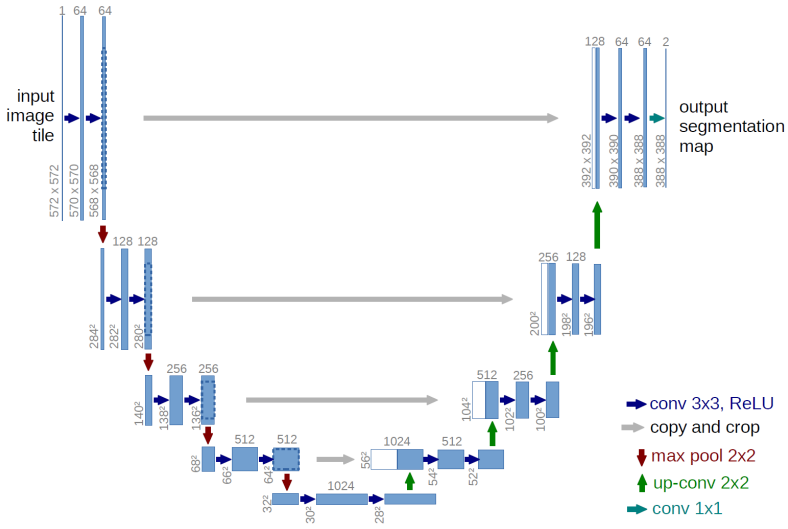


Figure 2.11: U-Net architecture [44]

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 required 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.

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.

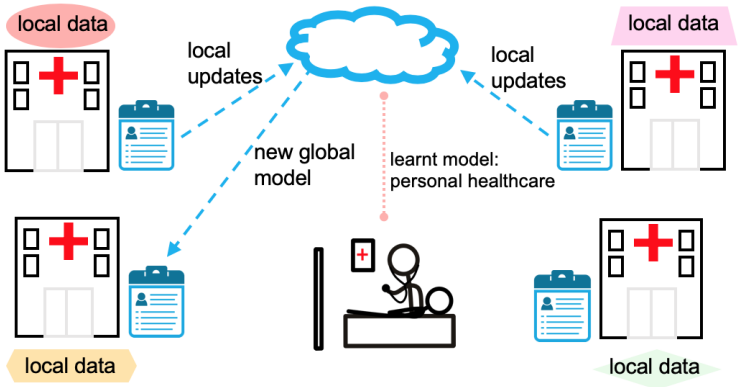


Figure 2.12: FL for healthcare via heterogeneous patient records from multiple institutions [47]

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, as shown in Figure 2.12. 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 [48]:

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

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 [49].

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 [50]) or close to where it was created (fog computing [51]), instead of at a central server. Recent works have also considered training statistical models centrally but serving and storing them locally [52]. 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.

Relevance to Healthcare Data

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. Models could potentially uncover patterns between previously disconnected datasets from several institutions. For institutions which contribute large amounts of data, predictive models could be developed that are fine-tuned [53] to their patients. Smaller institutions such as clinics could still receive a model that generalises well enough, based on the data seen from every contributing institution, to benefit their patients as well.

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 [54] 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: 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 [55], 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 [56, 57, 58, 53]. These algorithms are able to model heterogeneous data from both pharmaceutical companies and medical institutions.

Transport industry: Due to the potentially large number of self-driving vehicles we will see on the road and the need for them to quickly respond to real world situations, it is only a matter of time before FL solutions are used 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 when and where people are going, 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 [59].

2.8 Federated Learning Algorithms

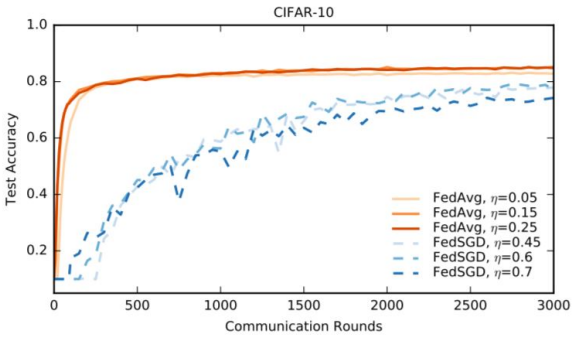
2.8.1 Terminology

K : total number of participants
n_k : number of training samples at participant k
C : fraction of participants selected for each round

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 2.13: FedAvg algorithm and its performance benefits [60]

$\ell(w, b)$: 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

2.8.2 Federated Stochastic Gradient Descent (FedSGD)

With SGD, gradients are computed using a random subset of the entire dataset, and used to perform a step of gradient descent. 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 [60].

2.8.3 Federated Averaging (FedAVG)

Federated Averaging [60] 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 2.13a. 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 2.13b.

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 in S_t updates w_t for E epochs of SGD on client k 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. $w_{t+1} = \sum_{k=1}^{K} \frac{1}{K} w_{t+1}^k$

Weighted Averaging

A participant's updates are weighted by the proportion of samples they provide (weighted FedAvg), as shown in Figure 2.13a.

2.8.4 FedProx

FedProx [61] 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.

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

The μ term is a hyperparameter. The internal term represents the L2-norm of the weights of the locally trained model w^t 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 in highly heterogeneous settings; FedProx achieves more stable and accurate convergence, improving test accuracy by 22% on average in highly heterogeneous settings.

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 [54]. However, for the scope of this project, we assume that only 15 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 2.8.2, the use of FedAVG instead of SGD enables convergence in significantly fewer communication rounds.

Reducing the size of transmitted messages: Model compression techniques [62, 63] 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 also 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 [54], 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 the two 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 2.8.2, FedAvg diverges when data is not identically distributed across network participants.

The global diagnostic imaging market is dominated by three companies [64]; Siemens (23.2%), General Electric (22.2%) and Philips (19.7%). The imaging devices will produce different scans depending on the manufacturer [65]. 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, for our project we will be using maximally homogenised medical imaging datasets i.e. those that have been normalised and interpolated to standard resolutions to enable comparison between different methods.

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 computation 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 not the main goal 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 - 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 was present in the training data of an ML model [66]. 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, using an average of the images of that person which were provided during the training phase. The generated image does not produce a specific image from the training dataset, unlike with membership inference. Previous work [66] has shown that it is typically difficult to perform such attacks on CNNs. However, Generative Adversarial Networks (GANs) have been used recently [67] to effectively extract such information in collaborative settings. Despite this, it has been shown [68] 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 [69]. The attacker - a participant in a federated round - controls the local data, training procedure, hyperparameters and the weights submitted to the server for aggregation [70]. 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 [71] 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 then 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 [72] 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 [73, 57].

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 [74]. 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 [73] 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 [74]; 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 [75] 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 [76, 77].

2.10.5 Trusted Execution Environment (TEE)

Distributed 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 [78]. 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 [79].

CALTRAIN [80] 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 [56], the authors compare the performance of U-Net on the BraTS [81, 82, 83] dataset, using three different collaborative learning approaches:

1. A typical FLS, as shown in Figure 2.14a.

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.

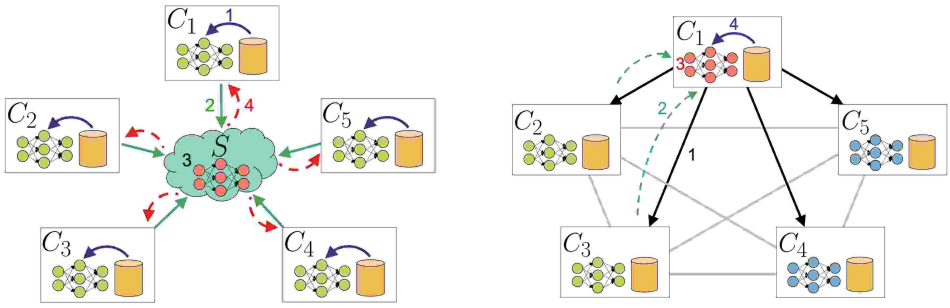
IIL and CIIL are completely decentralised approaches; i.e. there is no central aggregator. IIL uses significantly less communication bandwidth than FL because each institution only sends the model it has trained, and receives the initial and final models. However, the IIL technique suffers from "catastrophic 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 does not perform as well as either of the other approaches.

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 tasks, FL is much more stable; CIIL results showed 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

Last year, a study [57] 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 2.14a.



(a) FL with server

(b) BrainTorrent: P2P serverless FL

Figure 2.14: Overview of FL process [58]

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, weighted FedAvg 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 [53], the authors propose a completely decentralised approach where the medical institutions communicate directly among themselves.

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 2.14b. 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:

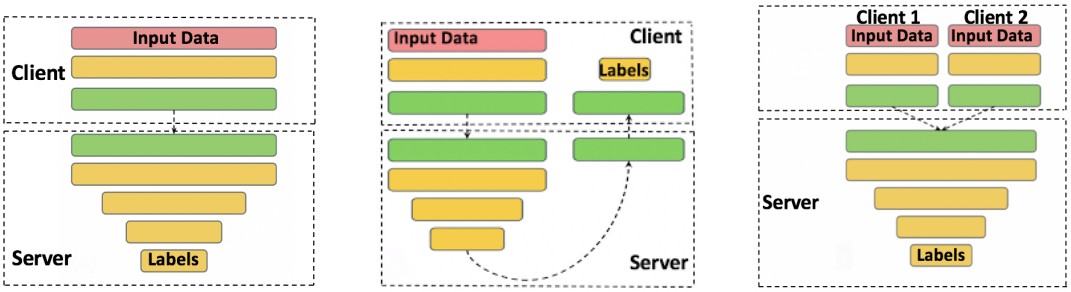
1. Each participant is locally trained in parallel for a few iterations.

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.



(a) Vanilla split learning

(b) Split learning without label sharing

(c) Split learning for vertically partitioned data

Figure 2.15: SplitNN configurations [58]

2.11.4 FL vs Split Learning

The proposed configurations [58] of SplitNN [84], shown in Figure 2.15, 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 2.15a), each medical institution trains a partial deep network up to a specific layer - the cut layer. The output from this layer are transmitted to a server to complete 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 2.15b), a U-shaped configuration is devised to prevent institutions needing to transmit labels to the server. The outputs from the final layer of the server are 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 2.15c) 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 upto the cut layers. The output at the cut layer from both institutions is then combined and sent to the central server to complete 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

Centralised Model (Benchmark)

WORK IN PROGRESS

Introduce chapter (mention the purpose of the centralised model as a point of reference to the federated learning implementation) with possibly a diagram detailing how the centralised model would operate in a hospital setting, mention tools and libraries used throughout the implementation and evaluation of the centralised model. Provide elaborate breakdown of how the benchmark model was trained. Discuss experimental methodology to configure and evaluate model. Include challenges faced and how they were addressed. Discuss how we measure computational and communication overhead. Mention that the evaluation of the centralised model is located in Evaluation section of report.

3.1 Key Tools and Libraries

Python3, pytorch, opencv for image manipulation, thop (profiling performance), jupyter notebooks (specify to perform stratified sampling to ensure 5% and 20% samples of the data followed the same distribution), matplotlib for visual analysis. Discuss alternative tools and libraries considered and why it was decided not to proceed with them. Also maybe mention the setup of a virtual environment and directory in bitbucket to allow experiments to either run on standard doc machines, as well as the gpu cluster.

3.2 Proposed Dataset

Include a few examples of how the dataset looks in general, e.g. one picture of each pathology (with frontal and lateral), to show how difficult it is to classify the images, pictures of the 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 task is to predict the probability of these different observations from multi-view chest scans, as shown in Figure 3.1.

Reference models

Since the training labels for each study in the dataset have either 0, 1 or u to represent negative, positive and uncertain observations respectively, the authors explore different approaches to using the uncertainty labels when training their reference models.

U-Ignore: Ignore u labels when training
U-SelfTrained: Use model trained by U-Ignore to make predictions on unlabelled observations. Each uncertainty label is replaced with the model's prediction.
U-Zeros: Treat all u instances as 0 (negative)
U-Ones: Treat all u instances as 1 (positive)
U-MultiClass: Treat u as separate class.

Evaluation

The authors focus on the evaluation of five observations which they call the competition tasks, selected based of clinical importance and prevalence: Atelectasis, Cardiomegaly, Consolidation,

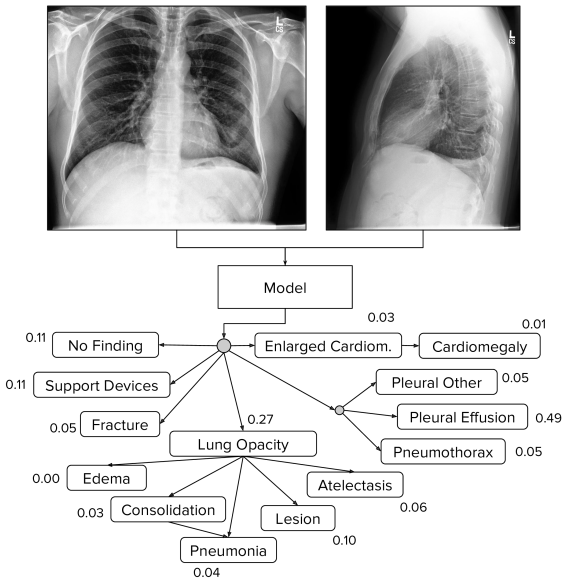


Figure 3.1: CheXpert sample with probability of different observations [8]

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]

Edema, and Pleural Effusion.

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

The test set consists of 500 studies from 500 previously unseen patients [85]. 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. Since we could not get access to the test set for the project, the validation set will be used for final evaluation purposes and a subset of the training set will be used as validation data.

3.3 Implementation

Address how we could only analyse the meta-data available to us, they don't give information about radiologist/institution, which may also introduce implicit bias into the model

Source the repo used (https://github.com/jfhealthcare/Chexpert). Mention that this was found after checking https://stanfordmlgroup.github.io/competitions/chexpert/ website for submissions with available source code. It was discussed in early supervisor meeting that our benchmark should be based on atleast one of the top 10 performing models. Describe the pipeline in an end-to-end fashion, e.g. image transformations etc. Discuss the overall structure of the deep NN used and include diagram for simpler explanation. Include description of dual attention (SAM) mechanism used in best performing models. Address how the lack of domain knowledge was sometimes a problem. Implementation used sometimes leveraged domain-specific knowledge to enhance the algorithms' performance. For example, they used a mix of U-Ones and U-Zero in the implementation, and we had to take that at face value. Sufficient domain knowledge would potentially assist us with adapting their implementation to perform even better in a federated setting, or enable us to make informed justifications as to why it may not be possible.

3.4 Experimental Methodology

Fixing seeds in RNG to ensure reproducibility of results. Also mention the configuration files for each experiment are also placed in the repository for reproducibility, as well as the log files showing the training results, for added transparency.

Include pictures of heatmap or no?

Mention that I had to register for the dataset, and that this was done early in the second term, to ensure there were no delays and enable experiments to run as quickly as possible. Talk about the challenge of long training times due to the size of the CheXpert dataset (∼11GB) and likely complexity of the deep neural networks developed, models could take up to four days to train.

Another mitigation to the long training times was to sample 5% of overall training data, such that distribution of labels (and meta-data) in subset matches that of overall training data. Used this, on sota config, to make sure everything worked end-to-end. Then extend to 20% and see that results are meaningful. When stratified sampling was done when configuring the early models, not sure if introducing any implicit bias as a result. Mitigated by comparing performance and also ensuring distribution of meta-data was similar, not just the labels. Then train on full dataset and compare performance to best performing sota model trained on best-weights, to ensure it is configured similarly, making comparison of results valid.

Hyperparamters (specify) were also tuned on this 5% of data, as it would take too long to perform hyperparameter searches on the full dataset, for all the different experiments being run! I suppose this is a negative, as you can never be sure if it's fully representative. Talk about especially with number of epochs, and how this was alleviated as an issue in the final model by setting up early stopping on the validation data, to prevent overfitting. Talk about how validation data was created by taking 500 images from training data from same distribution as training set. This had to be done because we did not have access to the offical test set, therefore the provided validation set was used as a test set.

Discuss the challenge of experiments getting killed on the department ray machines (Cuda out of memory errors) and how I was losing all the results because of that. Mention that the code was refactored to save checkpoints after each epoch to resume training. And another solution was to run the jobs on the department gpu cluster, but mention the constraints with that (max 2 jobs at a time that take no longer than 2 days each to complete, and queue time associated before a job would run).

3.5 Measuring Overhead

Talk about how and why this was done. Mention that it was one of the objectives of the project and that previous research into FL applied to medical imaging typically does not include both these metrics. Another reason that we report these metrics is so that future work can be done, regarding low level optimisations, and the results can be compared to our experiments and serve as a baseline. Mention that we initially considered using the LEAF (https://github.com/TalwalkarLab/leaf) benchmarking framework, but found out it was only compatible with Tensorflow models, and that we could generate the metrics ourselves, as described below.

3.5.1 Computation

Discuss why thop was chosen versus existing tools / manually computing them myself. Discuss the changes that had to be made to allow it to integrate with this model. Mention how the overall calculations were performed in an entire training process.

3.5.2 Communication

Ask Daniel if the centralised model should have the overhead of having to download all the data from each institution, but the individual institutions should have no communication overhead as they do not communicate with each other. This could be another case for using FL, as the communication overhead would be lower, since only updates are being transferred, not the images themselves. Also talk about the drawbacks of using my own method to calculate communication overhead (not very robust, serves as a lower bound, as it does not factor other network communication costs to transfer the data / updates). Don't know how accurate it is. Add x% overhead to final result to account for any further networking overhead.

Take raw size of images, and state the assumptions e.g. no networking overhead. State that if different modelling application was used, can measure the overhead in latency / other ways. We assumed the overhead is defined as follows.

Chapter 4

Institutional Models (Baseline)

WORK IN PROGRESS

Introduce chapter (mention what the institutional models represent, as a baseline i.e. the federated learning solution should outperform ideally all institutional models and in the worst case, at least the worst performing individual institution to justify the use of FL in the first place). Talk about how the data splits were decided and in each section, mention why they are relevant or what we expect to learn by running experiments on the different splits. Mention tools and libraries used throughout the implementation and evaluation of the institutional models. Discuss experimental methodology to configure and evaluate institutional models. Include challenges faced and how they were addressed. Mention that the evaluation of the institutional models is located in Evaluation section of report.

4.1 Key Tools and Libraries

Jupyter notebooks and matplotlib for analysing distribution of labels and meta-data, and generating the data partitions.

4.2 Two Institution Split

Mention that our initial goal was to investigate how effective FL is when applied to two medical institutions. Then explain we wanted to further the investigation by experimenting with different proportions of data at the institutions and why this was done.

4.2.1 Equal Split

Mention how the data was split into 2 non-iid partitions, each with 50% data. Show some of the meta-data and label distribution breakdown here. Talk about how this is to see if two equally-sized institutions with similar amounts of data can collaborate to produce a model better than they could individually.

4.2.2 Unequal Split

Mention how the data was split into 2 non-iid partitions, in a 75% / 25% split. Show some of the meta-data and label distribution breakdown here. Talk about how this is to see if a large and small institution can collaborate to produce a model that would help the smaller institution perform diagnosis at a similar level to the larger institution. And also to see if their combined model outperforms the larger institution.

4.3 Five Institution Split

Mention how, as part of stretch goal, the data was split into five non-iid partitions, each with 20% data. Show some of the meta-data and label distribution breakdown here. Mention that I considered running further experiments on the larger number of institutions (e.g. uneven proportion of data splits: 50/20/15/10/5 with 5 institutions, if 20/20/20/20/20 didn't perform close to

centralised model, to investigate if performance is tied to institution with highest proportion of data, but that there wasn't enough time to run all these experiments due to the high competition for resources.

4.4 Ten Institution Split

Mention how, as extension to stretch goal, data was split into ten non-iid partitions, each with 10% data. Show some of the meta-data and label distribution breakdown here. Mention that we would have extended the experiments to an even larger number of institutions but explain that we couldn't because that would require running n individual experiments on each partition. And this would not have been possible due to the high competition for resources. And as this was a stretch goal, experiments were run towards the end of May, when everyone wanted to use the GPU cluster and ray/gpu machines.

4.5 Experimental Methodology

Discuss experimental methodology for generating the institutions' data partitions. Since the CheXpert dataset does not distinguish by radiologist / institution, artificial partitions had to be created to simulate a federated environment. Mention the challenges of partitioning the data in a non-IID manner and why non-IID was selected (more clinically representative). Talk about how random state was fixed when partitioning the data to ensure reproducibility of results. Once again, talk about how this was first run on 5% then 20% samples of the full training set before running the experiments on the whole dataset.

4.6 Measuring Overhead

Mention that computational overhead is calculated in the same way as with the centralised model, but that the institutional models have no communication overhead, since in this scenario, the institutions do not communicate with each other.

Chapter 5

FL Models

WORK IN PROGRESS

Introduce chapter. Talk about the goal of the FL model to at least outperform the worst institutional model, and in the best case scenario to match the performance of the benchmark model. Mention tools and libraries used throughout the implementation and evaluation of the FL models. Discuss experimental methodology to configure and evaluate FL models. Include challenges faced and how they were addressed. Mention that the evaluation of the FL models is located in Evaluation section of report.

5.1 Key Tools and Libraries

Talk about frameworks that were considered, but ultimately not used (and why they weren't used)

- TensorFlow Federated / TFF: https://www.tensorflow.org/federated Provides high- and low-level interfaces (to express custom federated algorithms). Decided not to use it because the project was not focused on low-level optimisation, and I had more familiarity with pytorch.
- NVIDIA Clara SDK Framework: https://devblogs.nvidia.com/annotate-adapt-model-medical-imaging-clara-train-sdk/ The Clara Train SDK, part of the Clara AI toolkit, gives data scientists and developers the tools to accelerate data annotation, development, and adaptation of AI algorithms for medical imaging Provides pretrained models, high and low level APIs Integrates with Keras for 'bring your own model' Did not use it because we could not find a lot of guides / tutorials online.
- FATE: https://github.com/FederatedAI/FATE Supports standalone and distributed runtime architecture includes implementation of many common machine learning algorithms as well as necessary utility tools Not selected because it was not as popular as the other choices, and the framework was not as mature.
- Substra: Uses Docker, which requires sudo permissions, meaning I wouldn't be able to configure it to run on the department machines. Also not many examples available on how to configure it.
- OpenMined/Pysyft: https://github.com/OpenMined/PySyft Has a lot of documentation and useful tutorials Supports methods of secure computation This was initially selected as our framework of choice but as the project went further, we ran into issues with it (master branch having issues, not supporting momentum in learning stage, etc) and the benefits it brought (methods of secure computation, websocket workers) were becoming less relevant to the project.
- Mention that there are existing frameworks that incorporate DP into the training processs (and refer to fb library `https://github.com/facebookresearch/pytorch-dp`) and show that it's straight out of the box and ready to use. it made training 3x slower and didn't have a lot of documentation
- Manually writing the FL training loops: Explain why we decided to pursue this option.

5.2 FL Algorithms

5.2.1 Federated Averaging (FedAvg)

Mention what changes had to be made to the source repo / config files (new fields + local epochs hyperparameter). Can probably show source code for algorithm here because it's so short.

Weighted FedAvg

Mention what changes had to be made to the source repo/ config files (new fields). Can probably show source code for algorithm here because it's so short.

5.2.2 FedProx

Mention what changes had to be made to the source repo/ config files (fields + new mu hyperparameter). Can probably show source code for algorithm here because it's so short.

To serve as a comparison to the performance of FedAvg, each experiment is repeated using FedProx, a recently developed FL algorithm. FedProx has shown to outperform FedAvg in non-IID settings. . fill in with correct section

5.3 Implementation

Talk about how the source repo had to be modified to support FL training. Mention challenges faced and how they were addressed (e.g. trade-off between using my own FL implementation rather than an off-the-shelf framework, slow training times for fedprox, dataparallel model loading in evaluation section which gave poor performance etc).

5.4 Experimental Methodology

Mention that the outline for our FL implementation was sourced from (`https://github.com/IBM/FedMA/blob/master/language_modeling/language_main.py`). Talk about how the FL experiments were run on the 5% and 20% institution splits, and once we were happy with results, extended onto the full dataset. We ran the 2-institution experiments, then the 5 and finally 10 institution experiments. Once we acheieved SOTA performance, we attempted to improve it by using FedProx, which has shown to outperform FedAvg in non-iid settings. When this was implemented, all experiments were run again using FedProx as the FL algorithm.

5.5 Measuring Overhead

Explain that computational overhead is calculated as before, and that communication overhead is determined, at each institution, as the number of bytes they upload to and download from the aggregator during a federated round of learning . Also draw attention to the benefit of FL compared to centralised model, since FL institutions only transmit updates, rather than the entire data, to the server. This is less expensive on the network. Report comms overhead as bytes uploaded + downloaded

The communication overhead of the institutions participating in FL is constant, regardless of the number of participants. The communication overhead of an FL aggregator scales linearly with the number of participating institutions, regardless of how much data each institution contributes. Explain why

Chapter 6

Evaluation

Ready for review

In this section, we discuss and evaluate 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 show charts describing the distribution of meta-data, labels and observation-specific performance for only the benchmark model. For all the other experiment, this information is placed in Appendix A. In 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.

6.1 Benchmark

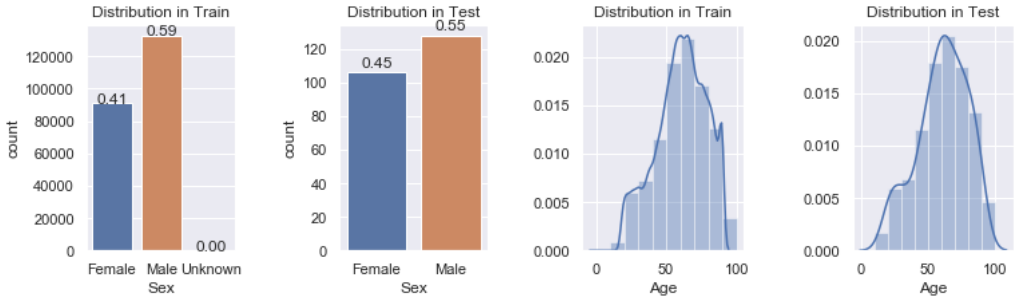
As described in Section 3, this represents a model trained on centrally aggregated data from multiple institutions.

The distribution of meta-data between the training and test data is shown in Figure 6.1. 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 6.2. 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.

Place correct section

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 . 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 model output to classify an observation as positive. 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 [85]. The performance of the benchmark model on the different observations is shown in Figure 6.3. 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 [86], 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).

It is worth noting that the network cost of downloading the full model (1624 MB) is not factored into the reported metrics on communication overhead. This is because every single institution and aggregator would need to incur the cost once and the size of the model never changes.



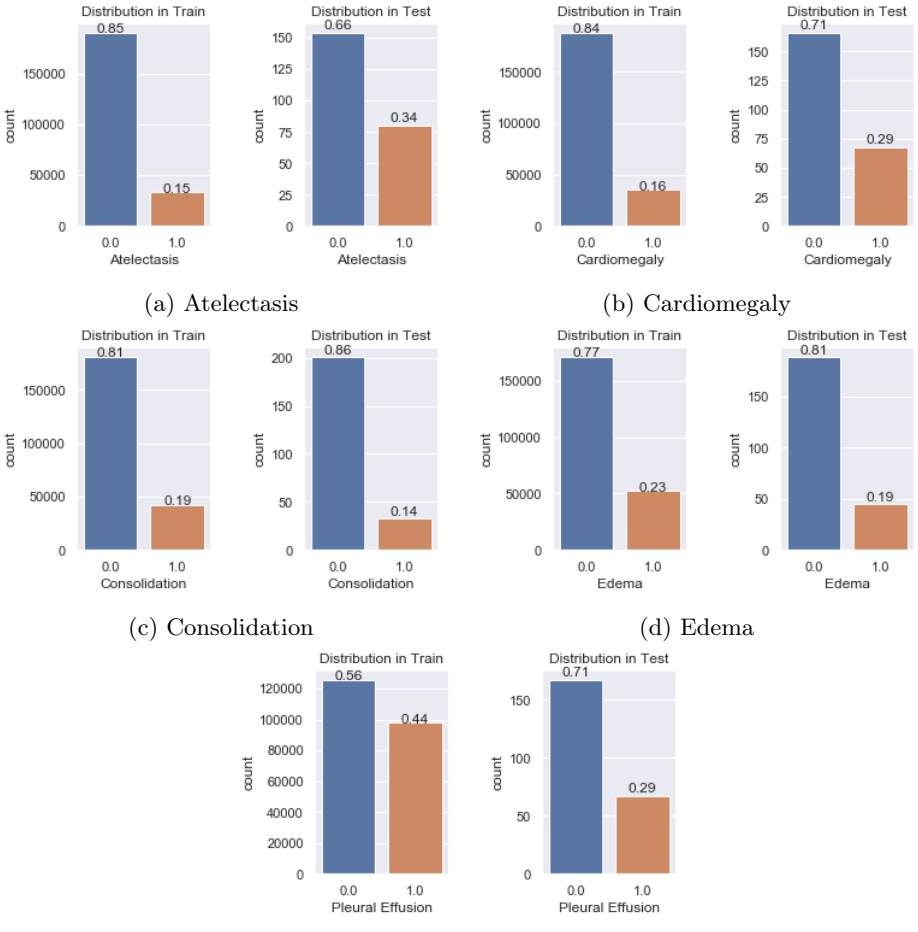
(a) Gender

(b) Age

(c) Scan Orientation

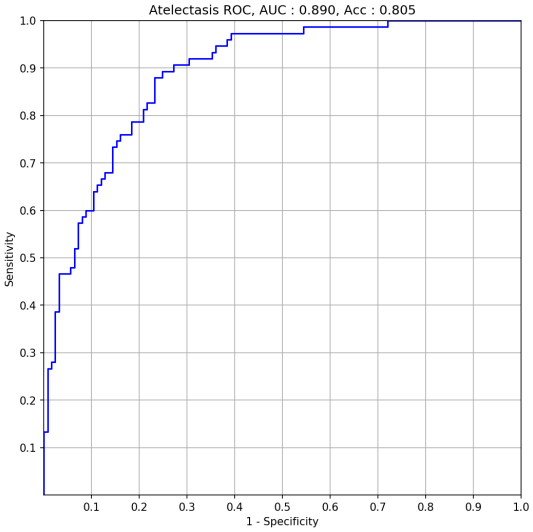
(d) Radiograph Technique

Figure 6.1: Distribution of meta-data between train and test sets

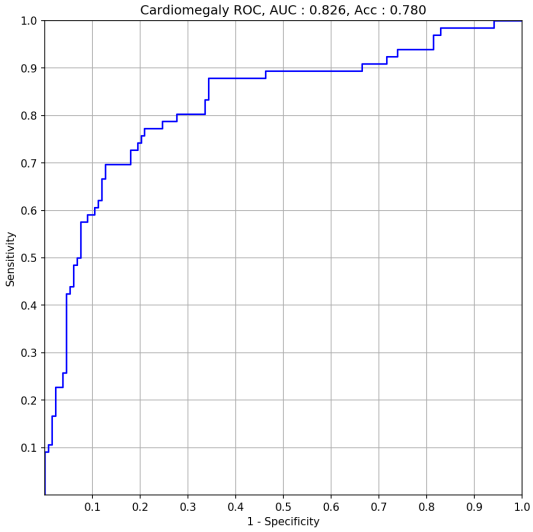


(e) Pleural Effusion

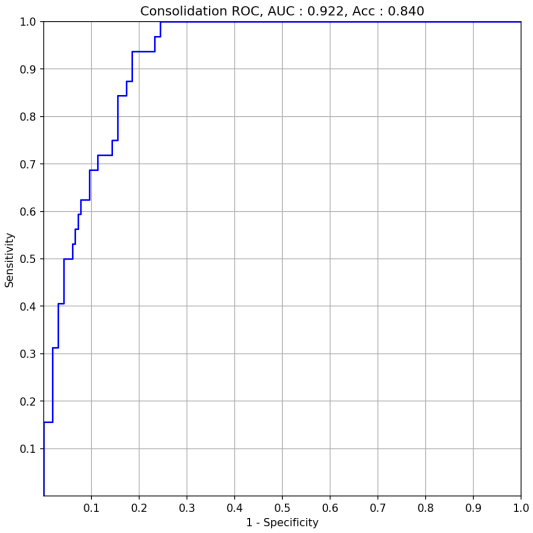
Figure 6.2: Distribution of labels between train and test sets



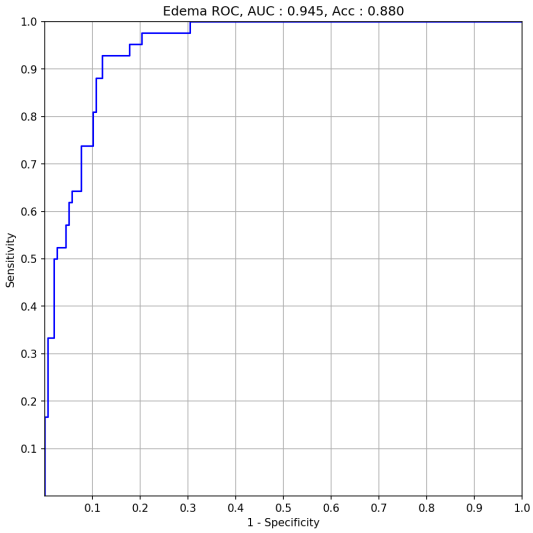
(a) Atelectasis



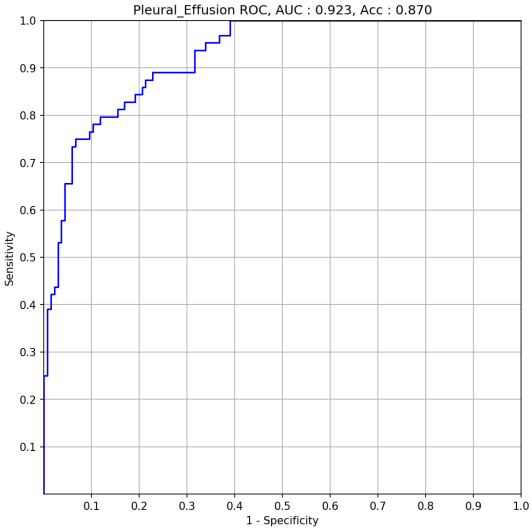
(b) Cardiomegaly



(c) Consolidation



(d) Edema



(e) Pleural Effusion

Figure 6.3: Performance of benchmark model on the different observations

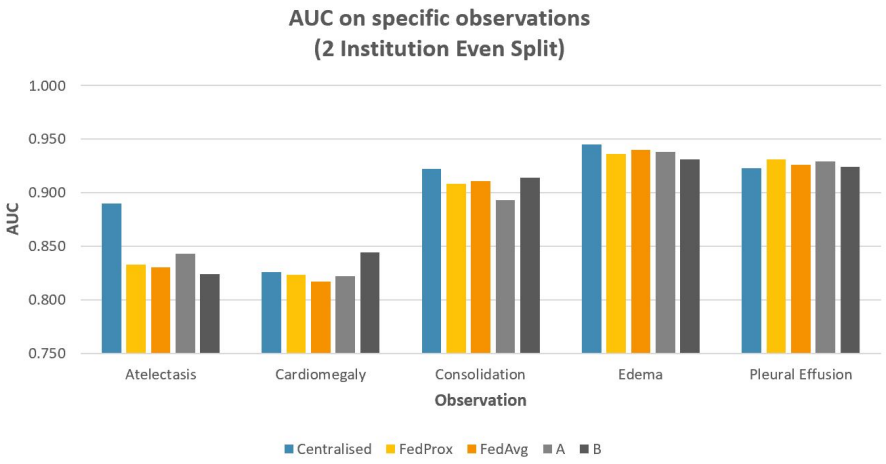


Figure 6.4: AUC on each observation using different approaches - 2 institutions (even)

6.2 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 6.4, we compare model performance of each technique on the different observations. We colour code the centralised, FL and baseline approaches to assist with interpretation. We see that the benchmark model significantly outperforms all the other approaches regarding Atelectasis classification. It is surprising 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 6.5, 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 6.6, 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 takes 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 6.7, we compare the communication overhead when training the models using the different approaches. As explained in Section 4.6, 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.

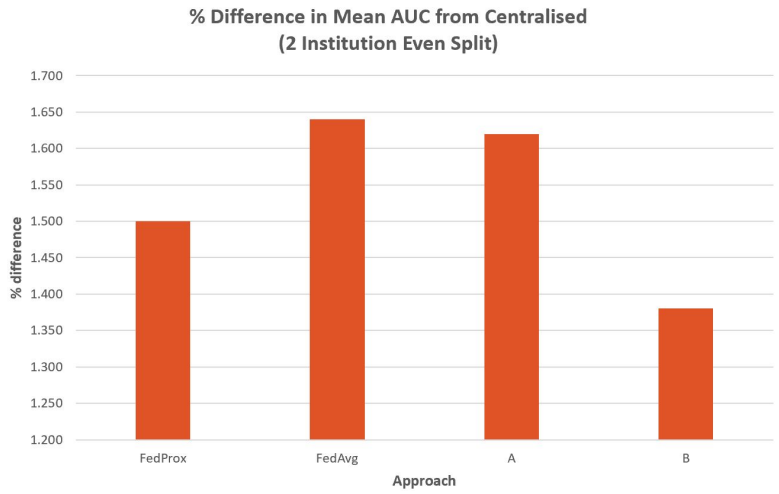


Figure 6.5: Mean AUC % offset from benchmark model - 2 institutions (even)

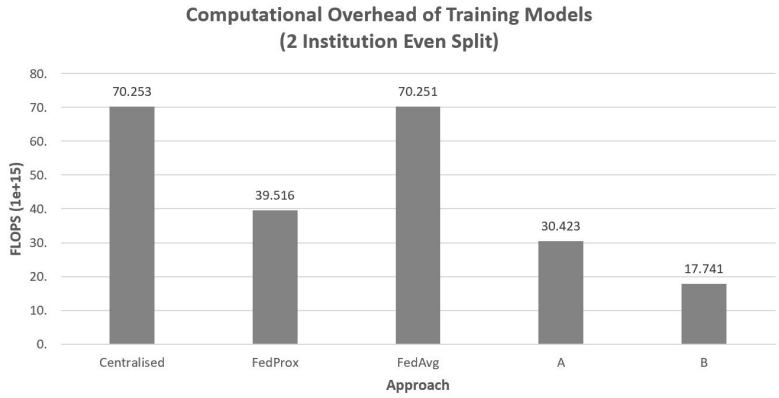


Figure 6.6: Computational overhead using the different approaches - 2 institutions (even)

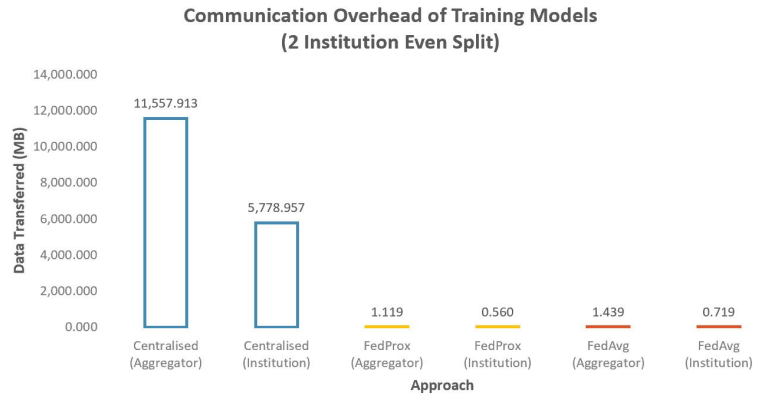


Figure 6.7: Communication overhead using the different approaches - 2 institutions (even)

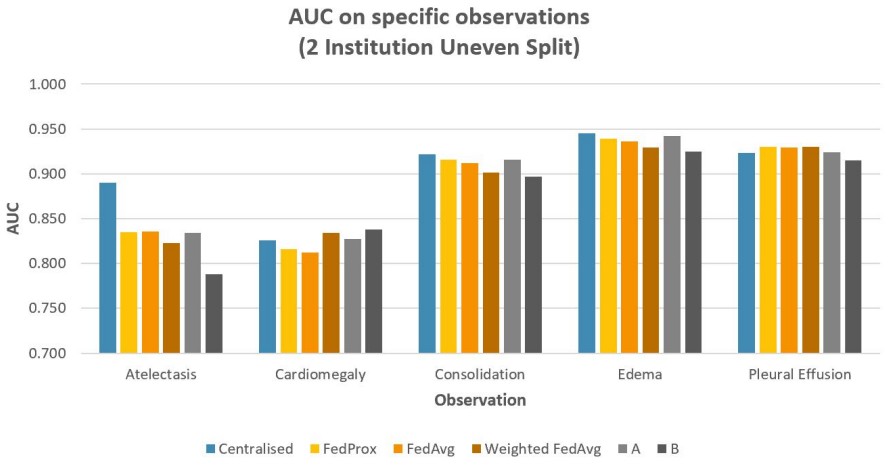


Figure 6.8: AUC on each observation using different approaches - 2 institutions (uneven)

6.3 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 6.9, we see that the institutional model (B) trained on 25% performs noticeably worse overall compared to the other approaches. On further inspection, looking at Figure 6.8, 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% 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 6.10 and 6.11, 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.

6.4 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

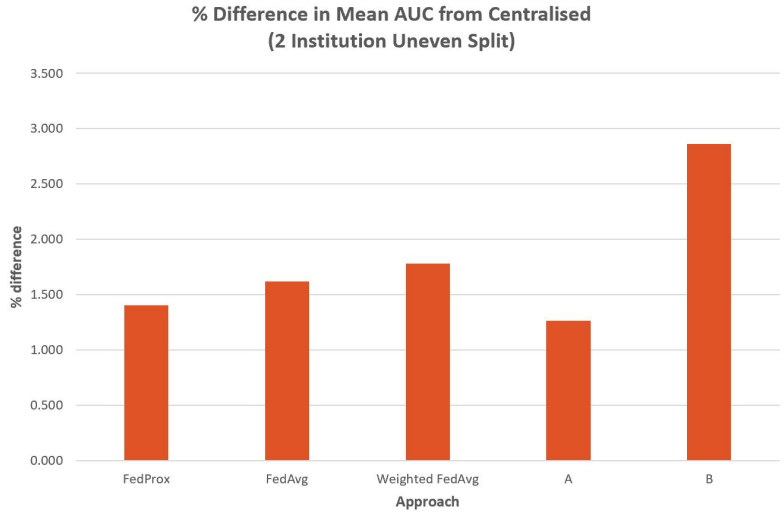


Figure 6.9: Mean AUC % offset from benchmark model - 2 institutions (uneven)

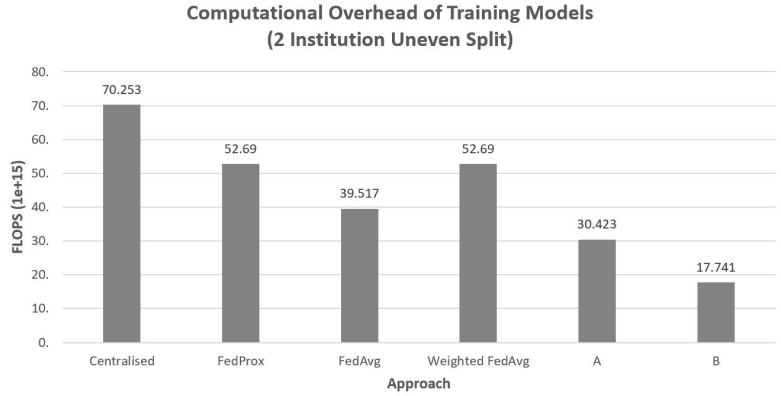


Figure 6.10: Computational overhead using the different approaches - 2 institutions (uneven)

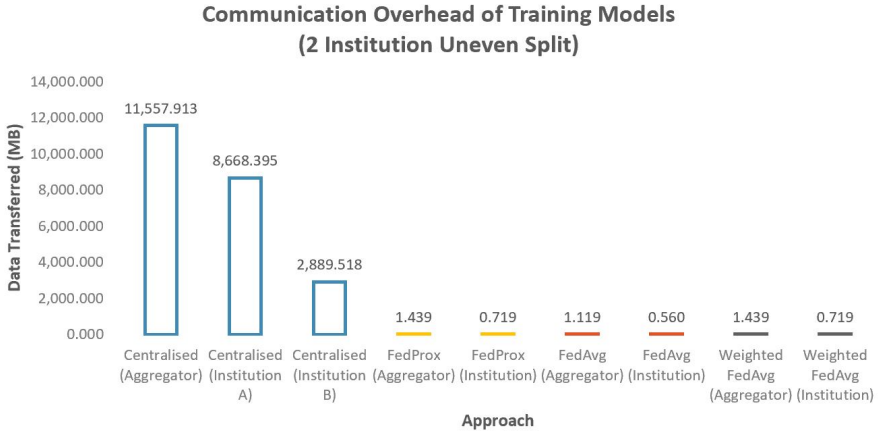


Figure 6.11: Communication overhead using the different approaches - 2 institutions (uneven)

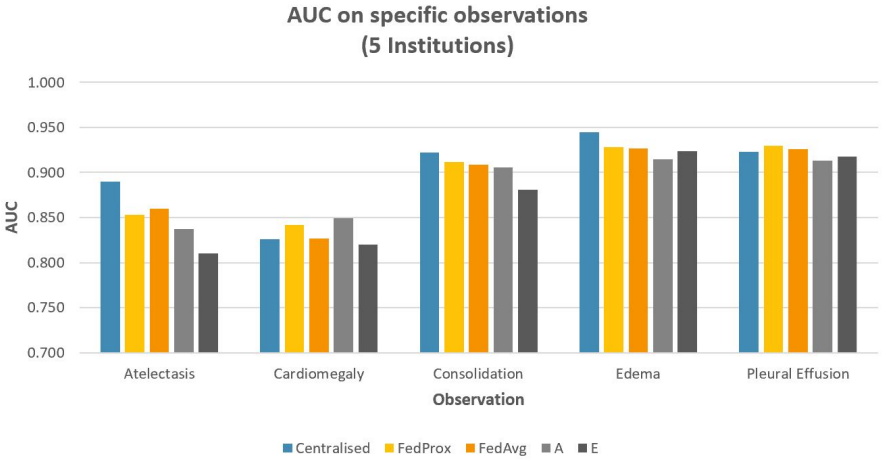


Figure 6.12: AUC on each observation using different approaches - 5 institutions

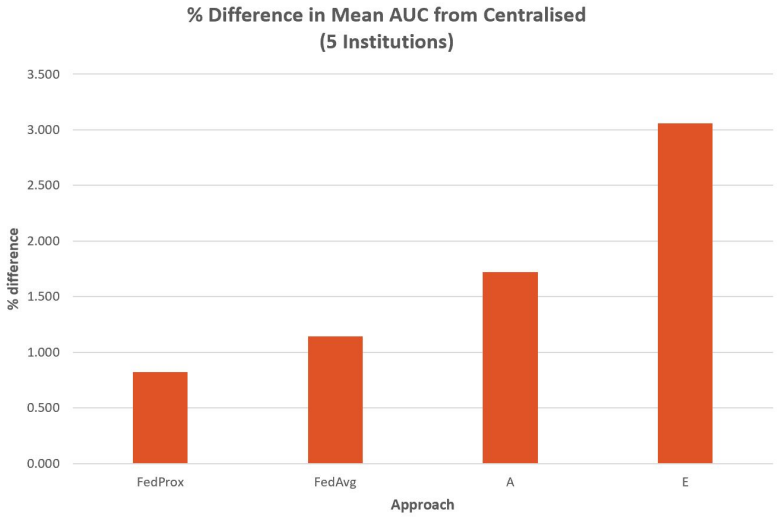


Figure 6.13: Mean AUC % offset from benchmark model - 5 institutions

models, is placed in Appendix A.3.

From Figure 6.13, we see that the FL approaches outperform all the baseline models. On further inspection, Figure 6.12 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 6.14 and 6.15. 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 local rounds of training n times, where n is the number of communication rounds needed to reach convergence. For reasons already described, FedProx has a lower computational and communication overhead than FedAvg.

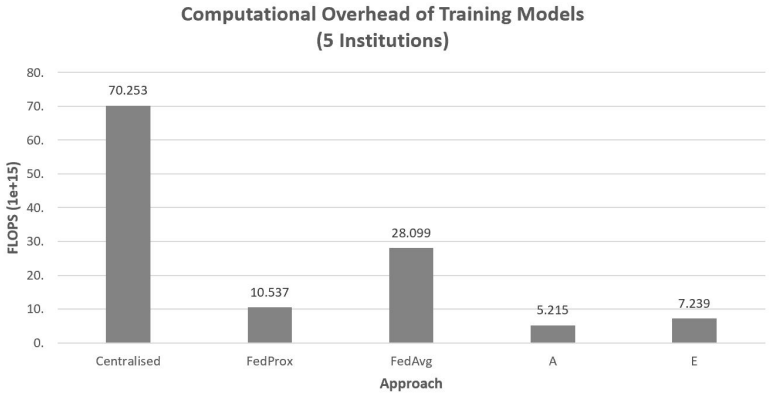


Figure 6.14: Computational overhead using the different approaches - 5 institutions

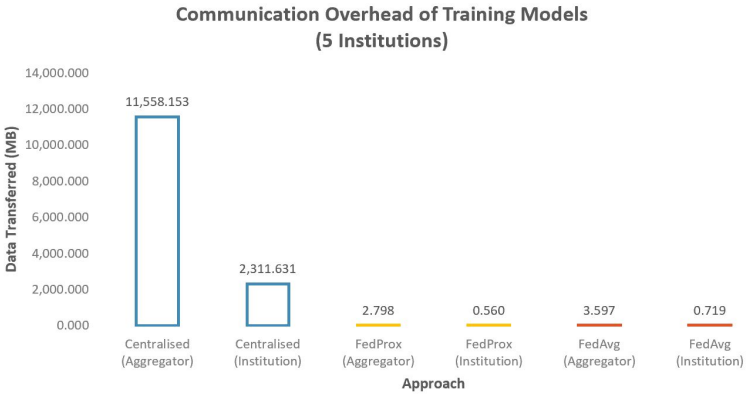


Figure 6.15: Communication overhead using the different approaches - 5 institutions

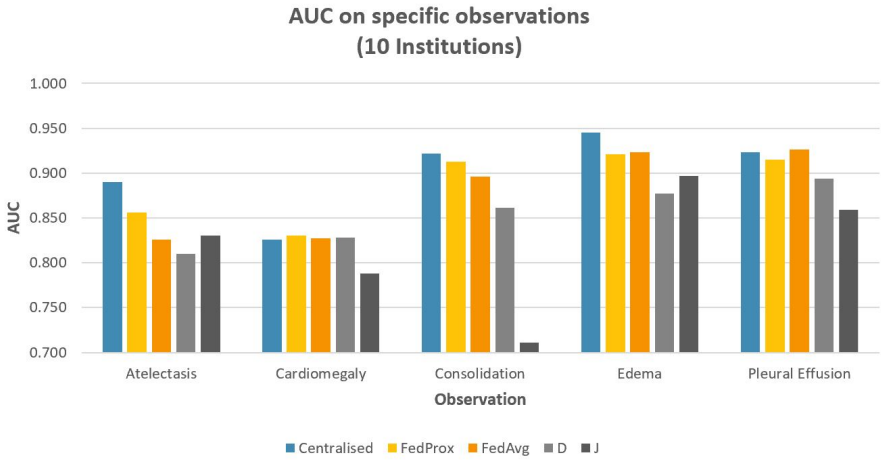


Figure 6.16: AUC on each observation using different approaches - 10 institutions

6.5 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 6.17 shows that both FL approaches also outperform all the baseline models in a ten-institution setting. Apart from the mean AUC performance, Figure 6.16 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 with lower data contributions, 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 6.18 and 6.19. 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 5.5.

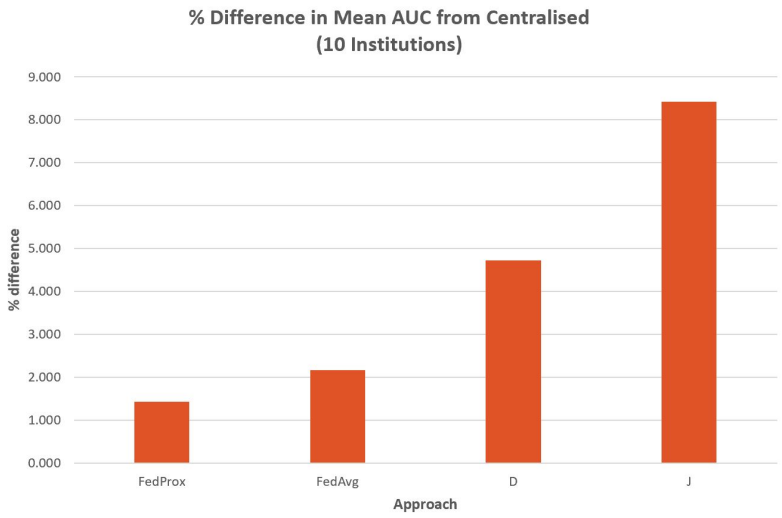


Figure 6.17: Mean AUC % offset from benchmark model - 10 institutions

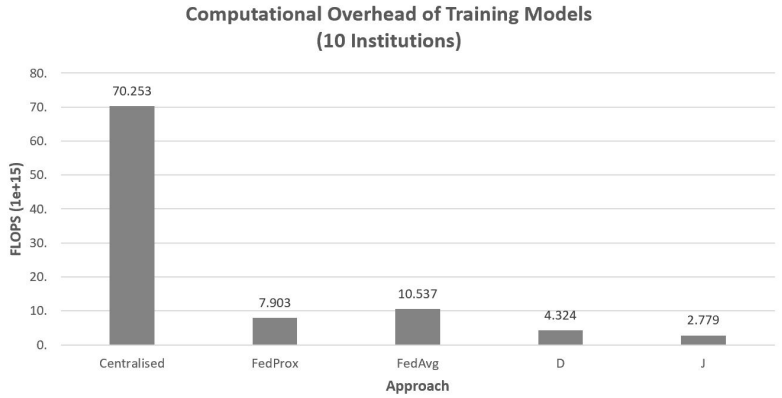


Figure 6.18: Computational overhead using the different approaches - 10 institutions

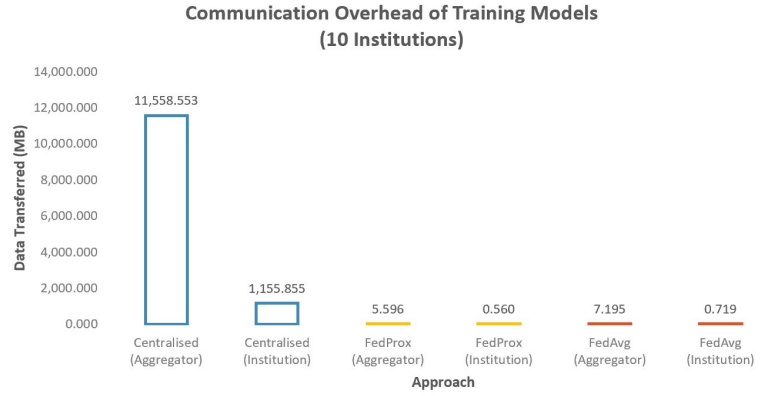


Figure 6.19: Communication overhead using the different approaches - 10 institutions

Chapter 7

Conclusion and Future Work

Ready for review

7.1 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 smaller than the centralised approach. The FL participants also incur a negligible 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 individually achieve. 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.

7.2 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.

7.2.1 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 typically selected during training.

7.2.2 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. We did not perform this experiment ourselves due to unfamiliarity with the domain of multi-task learning and also because of our limited time frame. Another reason this experiment would be challenging is that the radiographs tend to contain more than one positive observation; there may not be enough samples which are only positive for a specific observation.

7.2.3 Integrate Formal Privacy Guarantees

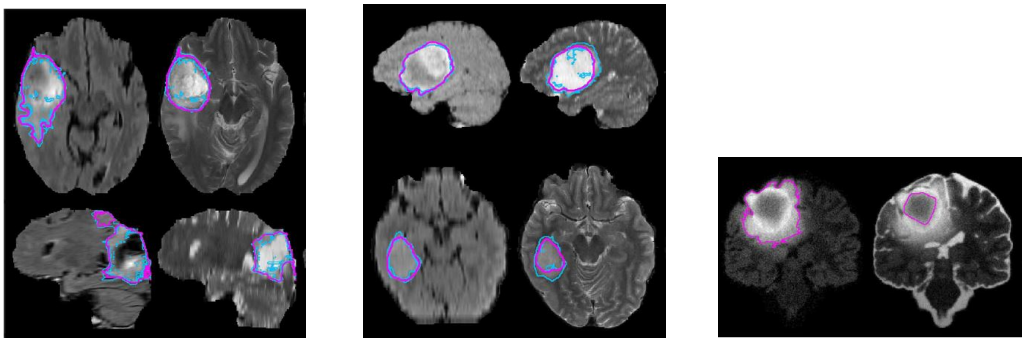
Another relevant experiment would have been to investigate the trade-off between data privacy and model performance, through the incorporation of ε-Differential Privacy (ε-DP) and/or Secure Multi-Party Computation (SMPC); concepts previously discussed in Chapter 2.10. Although we wrote our own FL training procedures, if formal privacy guarantees are to be incorporated, it is advised to use one of the frameworks which provide all this functionality, such as PySyft (discussed in Section 5.1). This investigation was not performed during the course of the project because formal privacy guarantees were not our primary focus. 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.

7.2.4 Optimise Training Overhead

Given that incorporating privacy preserving techniques and utilising FedProx significantly slow down the training process in FL, a possible future work would be to reduce the system performance overhead. This would likely involve using a library, such as TensorFlow Federated Core (mentioned in Section 5.1), 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.

7.2.5 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 7.1. The blue outlines



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

Figure 7.1: BraTS whole tumour volume dataset samples [81]

represent the individual experts' annotations and the magenta lines represent the consensus segmentation. There are no individual annotations for 7.1c, 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 [87] 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 - it is easy to overfit on the limited amount of data. Performance will likely degrade with more institutions added, since each will contribute fewer images to the model. 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. 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

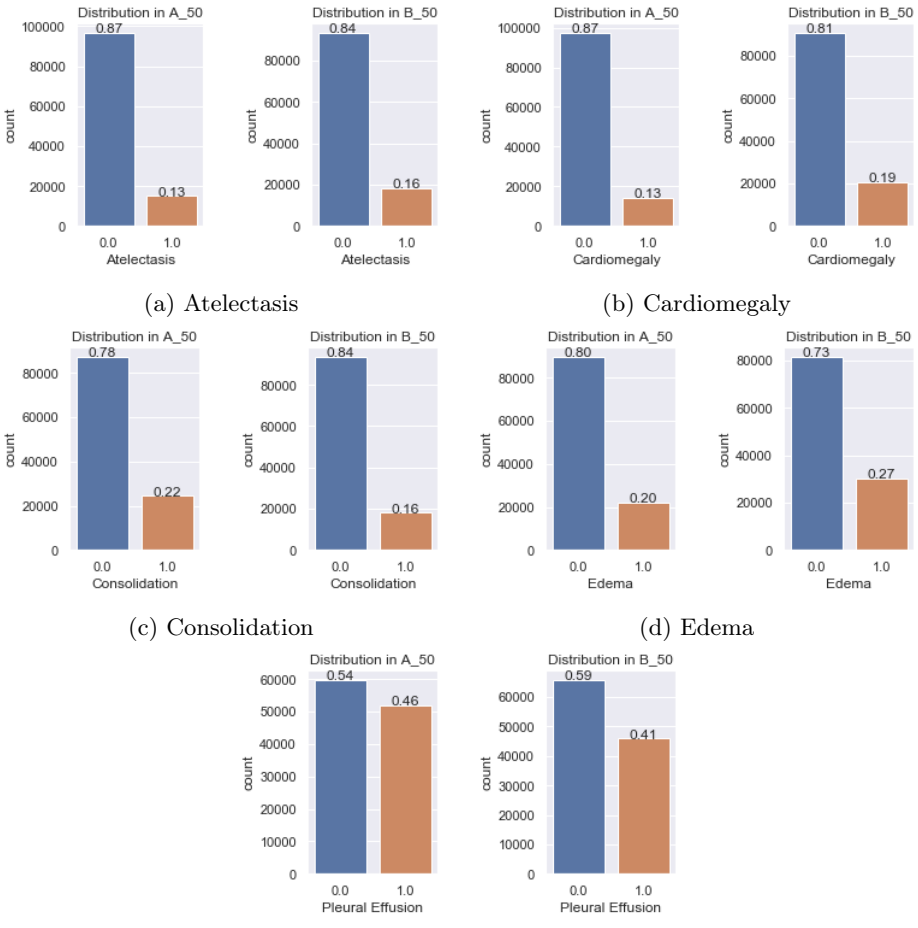
Remaining Analysis and Results

Ready for review

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

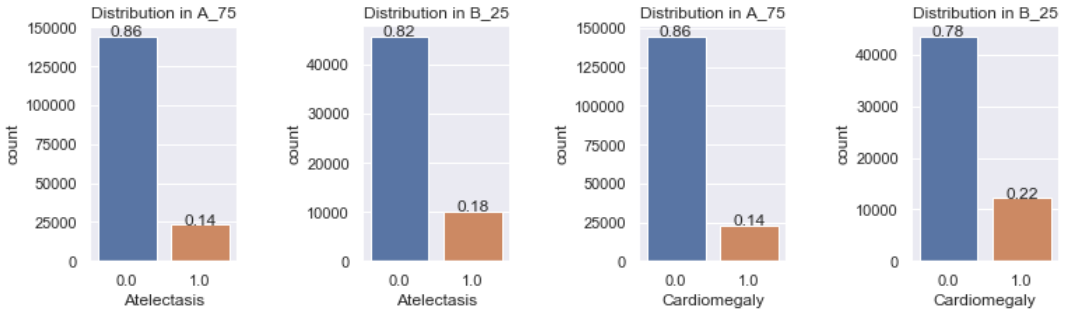
A.1.2 Model Performance

	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)

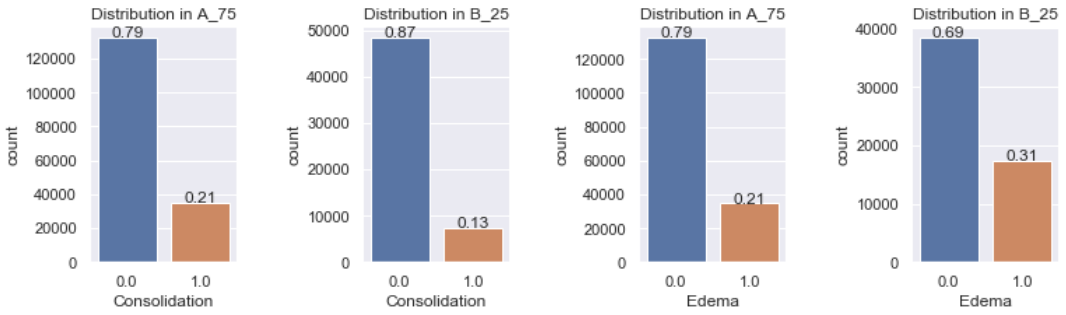
A.2 Two Institution Uneven Split

A.2.1 Distribution of Labels



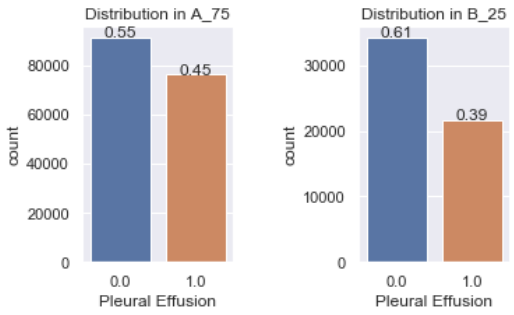
(a) Atelectasis

(b) Cardiomegaly



(c) Consolidation

(d) Edema



(e) Pleural Effusion

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

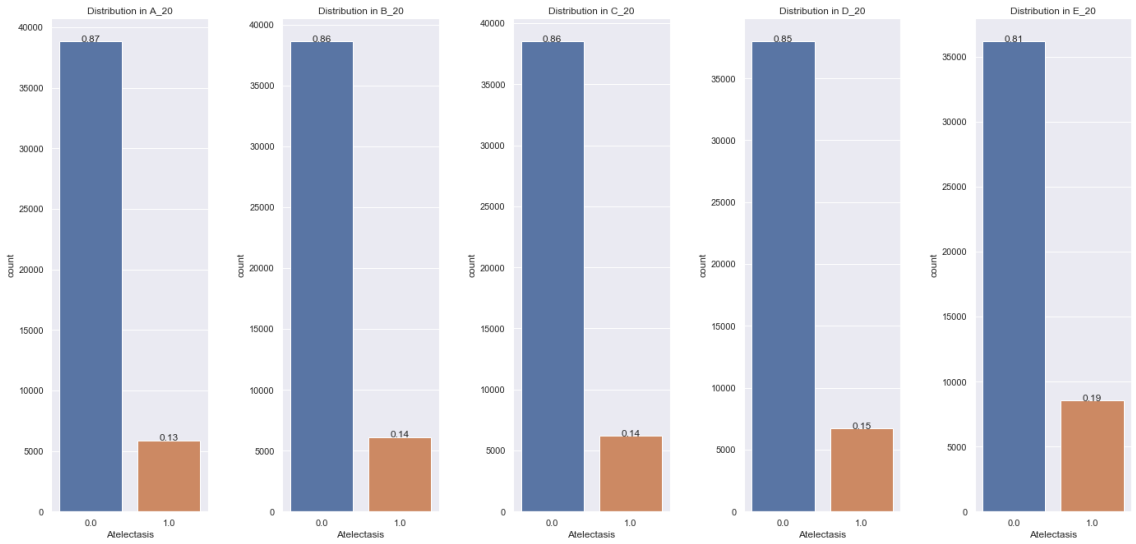
A.2.2 Model Performance

	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

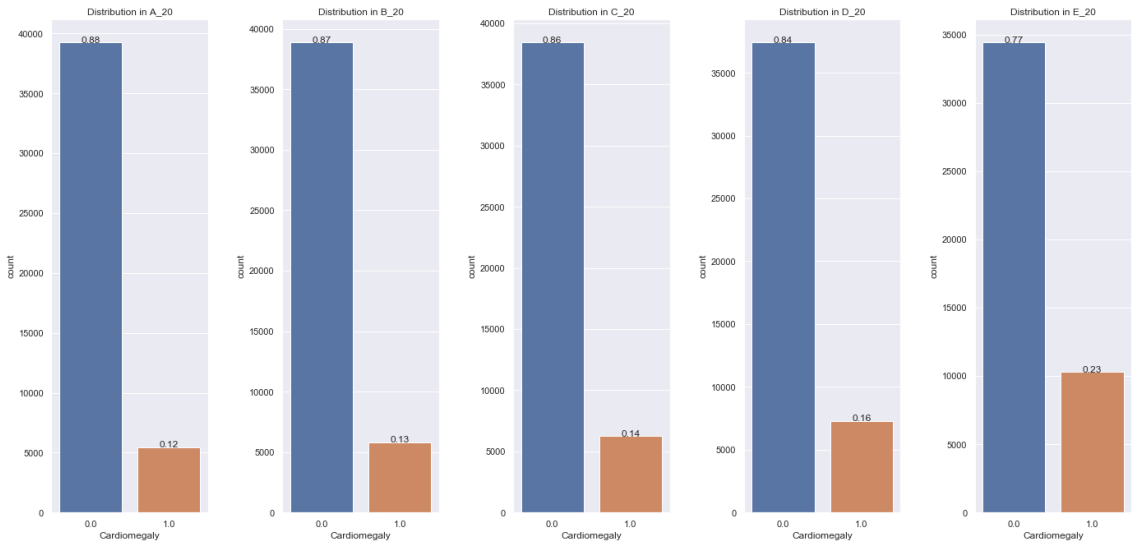
Table A.2: AUC on each observation using different approaches - 2 institution (uneven)

A.3 Five Institution Split

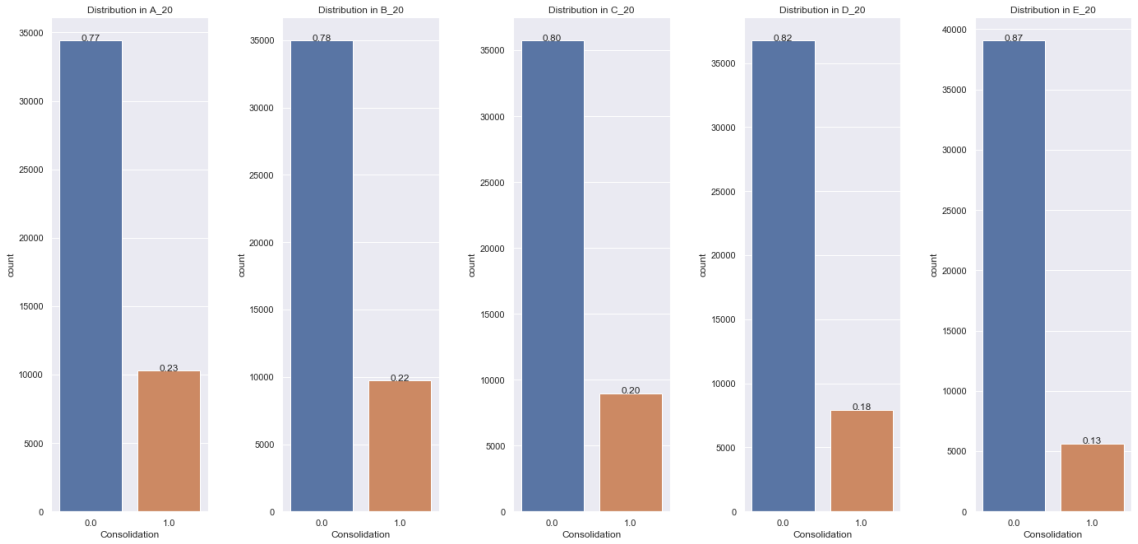
A.3.1 Distribution of Labels



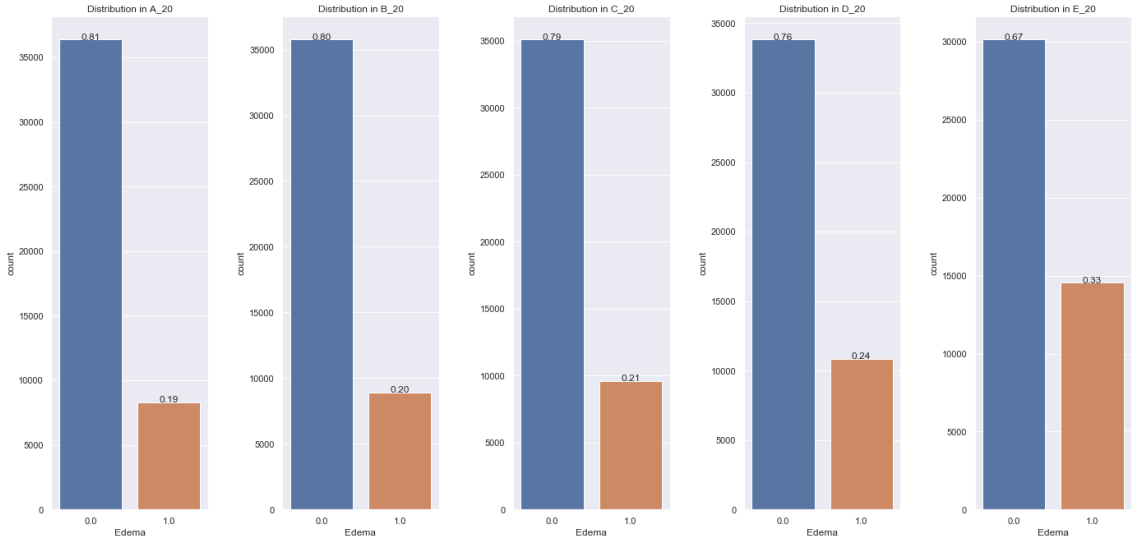
(a) Atelectasis



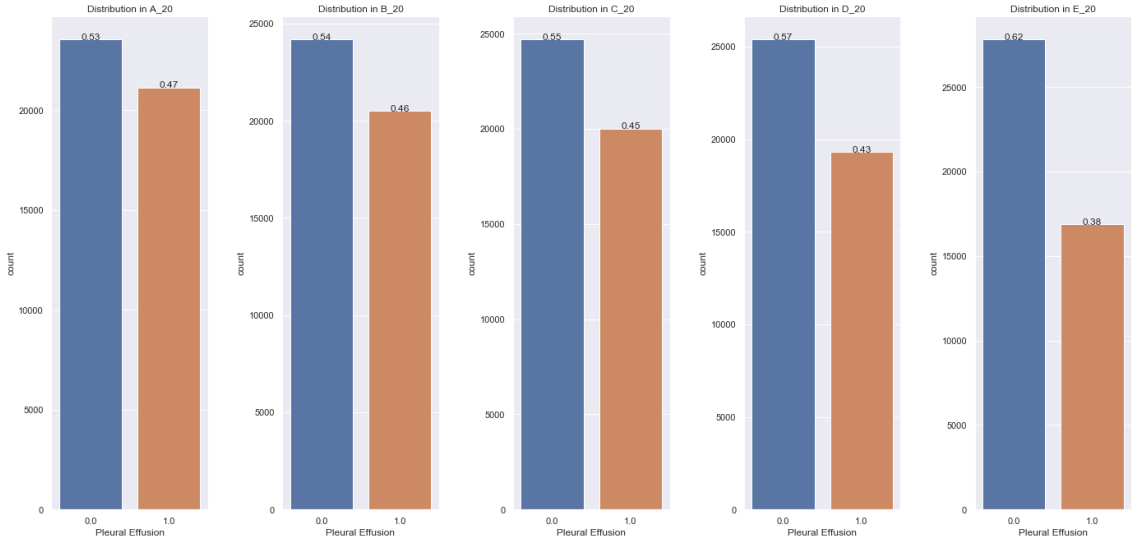
(b) Cardiomegaly



(c) Consolidation



(d) Edema



(e) Pleural Effusion

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

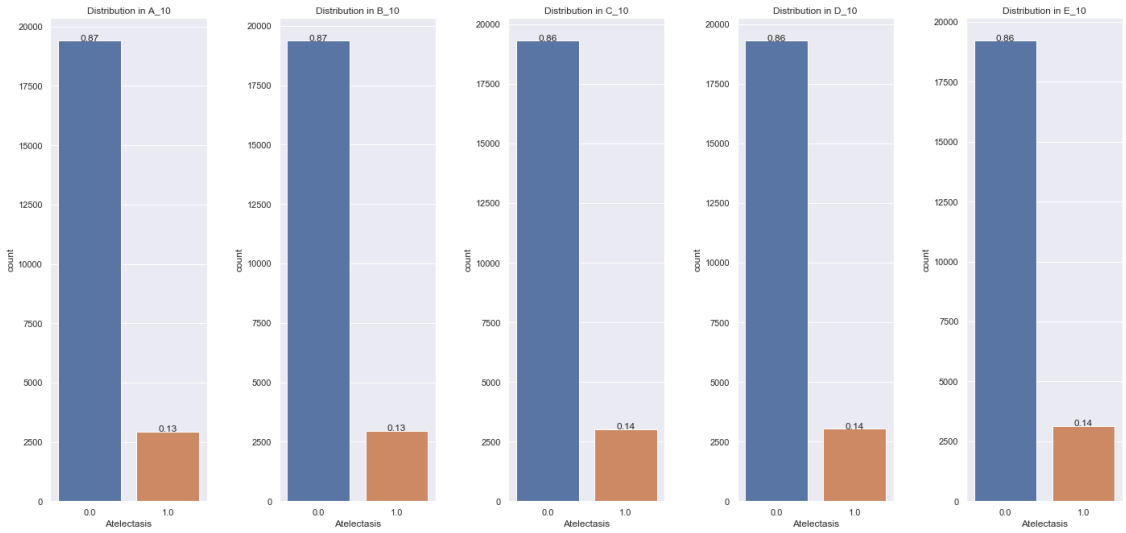
A.3.2 Model Performance

	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

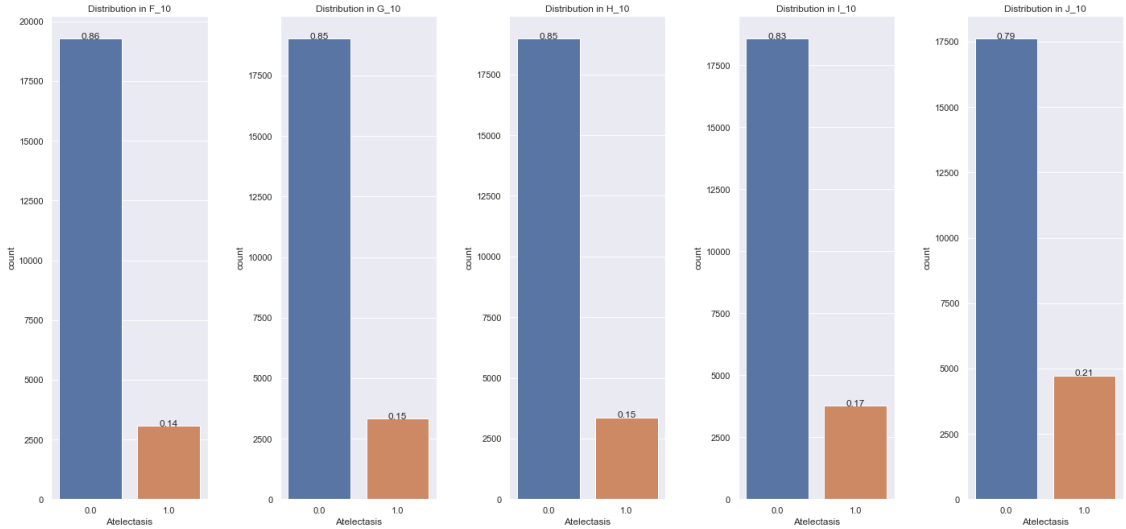
Table A.3: AUC on each observation using different approaches - 5 institutions

A.4 Ten Institution Split

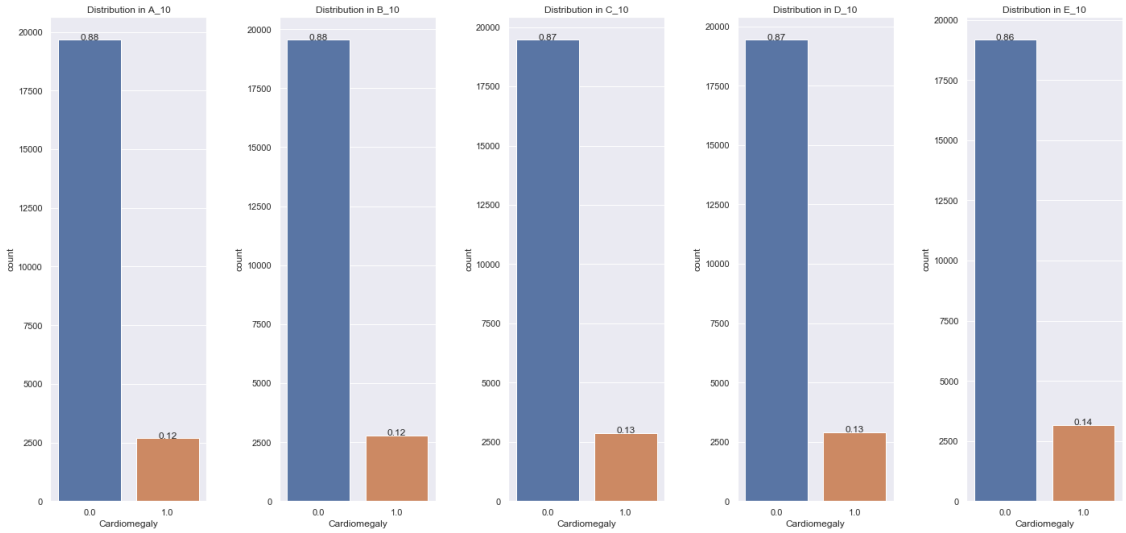
A.4.1 Distribution of Labels



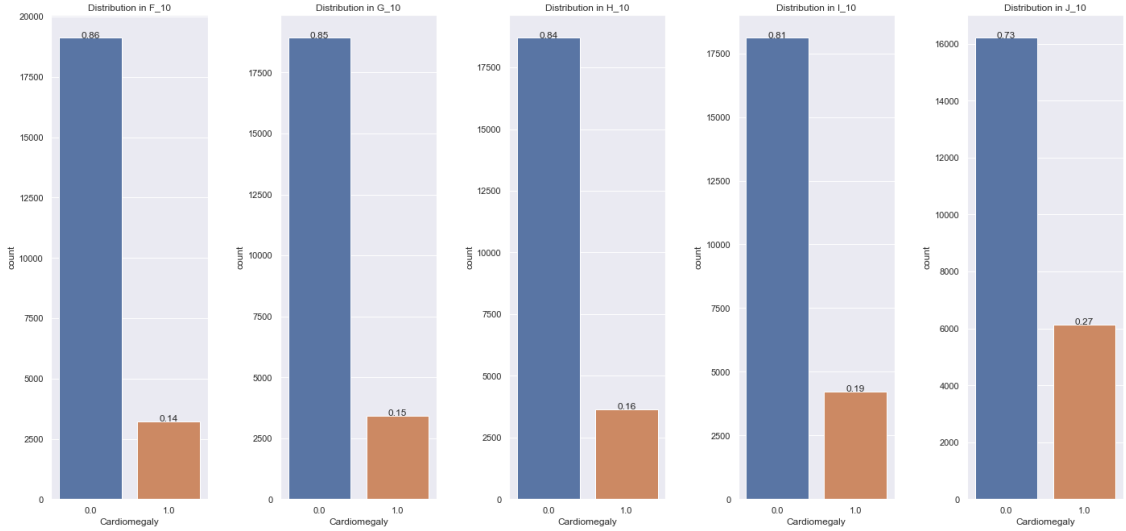
(a.1) Atelectasis: A - E



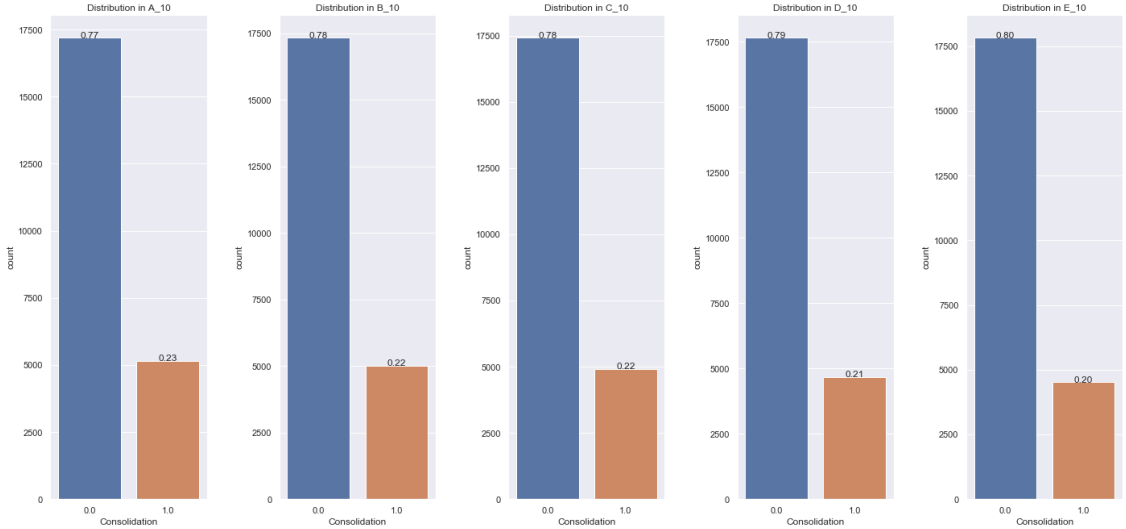
(a.2) Atelectasis: F - J



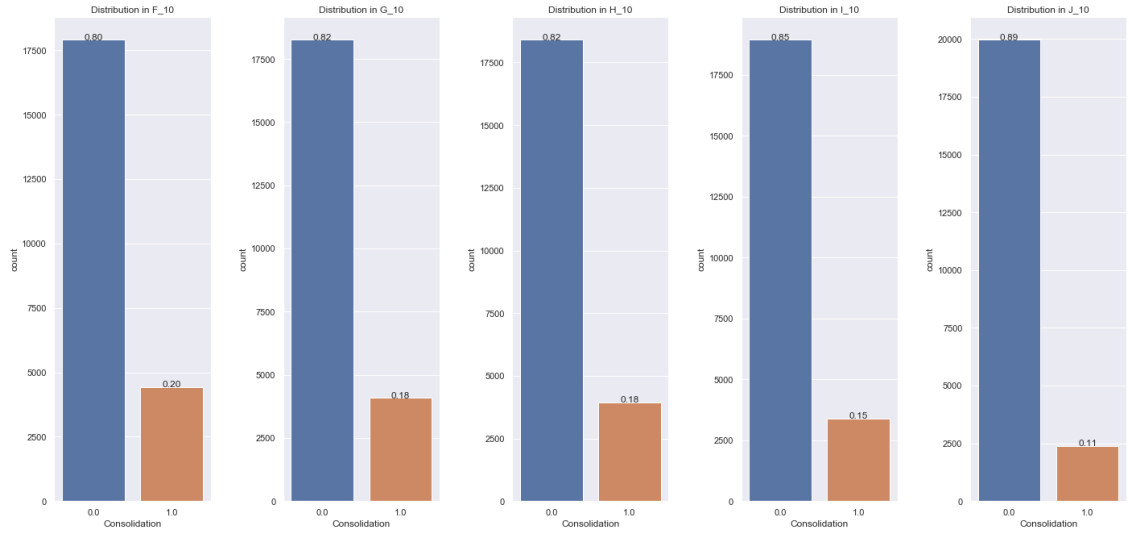
(b.1) Cardiomegaly: A - E



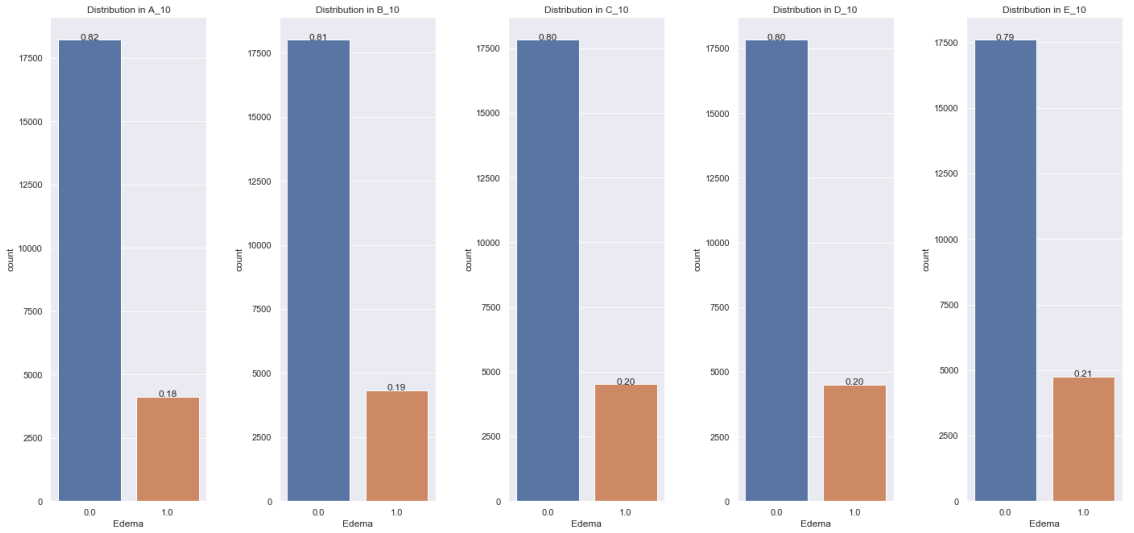
(b.2) Cardiomegaly: F - J



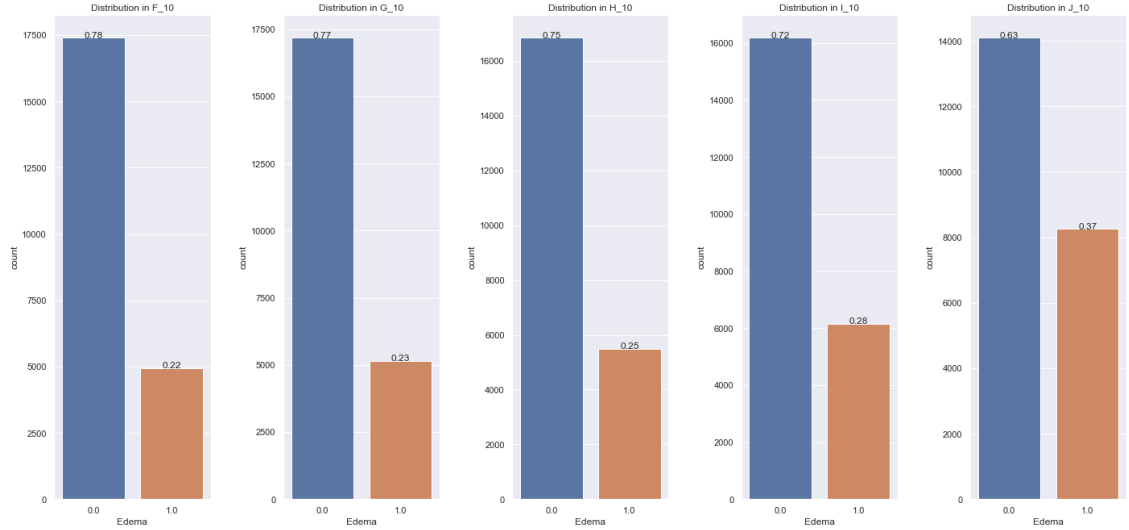
(c.1) Consolidation: A - E



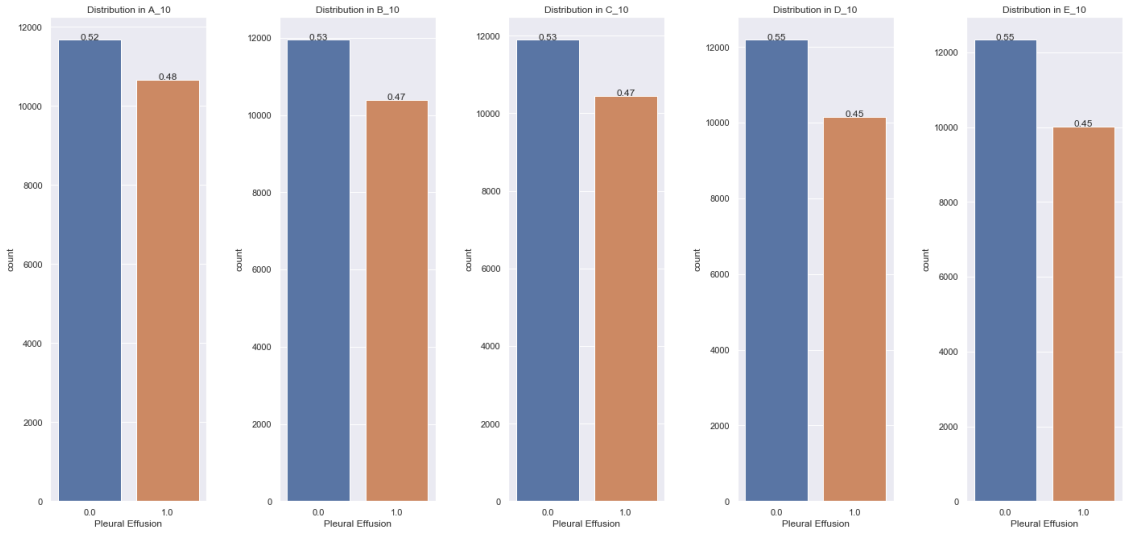
(c.2) Consolidation: F - J



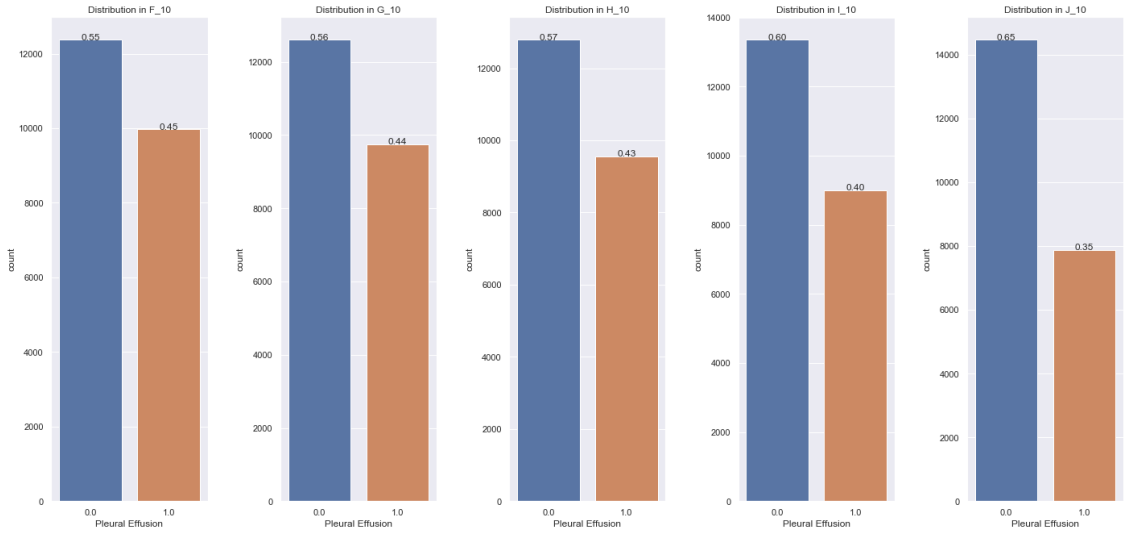
(d.1) Edema: A - E



(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

Miscellaneous

Probably going to remove this entire section and place in the background instead

B.1 Activation Functions

B.1.1 Sigmoid

Maps input to the range (0, 1).

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

B.1.2 Tanh

Maps input to the range (-1, 1).

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

B.1.3 Softmax

Given a vector, x, of real numbers, normalises it into a probability distribution proportional to the exponential of the input. i.e. each x_i gets mapped to (0, 1) and the sum of the output x values is 1.

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

B.2 Evaluation Metrics

B.2.1 ROC AUC

When we need to check or visualise the performance of a multi-class classification problem, we use ROC AUC. It measures performance at various thresholds settings. ROC is a probability curve and AUC represents the degree or measure of separability.

Refine this section with more detail

It shows how good a model is at distinguishing between classes. A score of 1.0 means it is perfect. A score of 0.5 means the model is unable to distinguish between the positive and negative class i.e. no better than random. A score of 0.0 means the model always makes the opposite decision from the correct one.

Terminology

AUC - Area Under the Curve. Represents the degree or measure of separability
ROC - Receiver Operating Characteristics. It is a probability curve

B.2.2 Accuracy

The accuracy is the number of correct classifications divided by the total number in the dataset.

Terminology

TP - True Positives
TN - True Negatives
FP - False Positives
FN - False Negatives

Formula

$$ACC = \frac{TP + TN}{TP + TN + FN + FP}$$

B.2.3 F1 score

The F1 score conveys the balance between the precision and the recall of classifications.

Terminology

Precision = $\frac{TP}{TP+FP}$

Recall = $\frac{TP}{TP+FN}$

Formula

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

B.2.4 Dice Coefficient (DC)

This 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 true values.

Terminology

A - Ground Truth
B - Predicted Truth

Formula

$$DC = \frac{2 * |A \cap B|}{|A \cup B|}$$

Bibliography

[1] Levenson RM, Krupinski EA, Navarro VM, Wasserman EA. Pigeons (Columba livia) as Trainable Observers of Pathology and Radiology Breast Cancer Images. PLOS ONE. 2015 Nov;10(11):e0141357. Available from: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0141357.

[2] Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017 Feb;542(7639):115–118. Available from: https://www.nature.com/articles/nature21056.

[3] Ayer T, Alagoz O, Chhatwal J, Shavlik JW, Kahn CE, Burnside ES. Breast cancer risk estimation with artificial neural networks revisited. Cancer. 2010 Jul;116(14):3310–3321. Available from: https://acsjournals.onlinelibrary.wiley.com/doi/10.1002/cncr.25081.

[4] Stern J. The Fragmentation of Health Data – Datavant;. Available from: https://datavant.com/2018/08/01/the-fragmentation-of-health-data/.

[5] Frank O. Opinion | Donate Your Health Care Data Today. The New York Times. 2019 Oct;Available from: https://www.nytimes.com/2019/10/02/opinion/health-care-data-privacy.html.

[6] Wikipedia F. General Data Protection Regulation. Wikipedia; 2020. Page Version ID: 933500662. Available from: https://en.wikipedia.org/w/index.php?title=General_Data_Protection_Regulation&oldid=933500662.

[7] Wikipedia F. Health Insurance Portability and Accountability Act. Wikipedia; 2019. Page Version ID: 932269315. Available from: https://en.wikipedia.org/w/index.php?title=Health_Insurance_Portability_and_Accountability_Act&oldid=932269315.

[8] Irvin J, Rajpurkar P, Ko M, Yu Y, Ciurea-Ilcus S, Chute C, et al. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. arXiv:190107031 [cs, eess]. 2019 Jan;ArXiv: 1901.07031. Available from: http://arxiv.org/abs/1901.07031.

[9] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. arXiv:14090575 [cs]. 2015 Jan;ArXiv: 1409.0575. Available from: http://arxiv.org/abs/1409.0575.

[10] Zeng T, Wu B, Ji S. DeepEM3D: approaching human-level performance on 3D anisotropic EM image segmentation. Bioinformatics. 2017 Aug;33(16):2555–2562. Available from: https://academic.oup.com/bioinformatics/article/33/16/2555/3096435.

[11] Amazon. Amazon Go;. Available from: https://www.amazon.com/b?ie=UTF8&node=16008589011.

[12] Moshakis A. Nation of shoplifters: the rise of supermarket self-checkout scams. The Observer. 2018 May;Available from: https://www.theguardian.com/global/2018/may/20/nation-of-shoplifters-supermarket-self-checkout.

[13] Waymo. Waymo - We're building the World's Most Experienced Driver;. Available from: https://waymo.com/.

[14] Tesla. Autopilot;. Available from: https://www.tesla.com/autopilot.

[15] VolvoTrucks. Automated Trucks | Volvo Trucks;. Available from: https://www.volvotrucks.com/en-en/about-us/automation.html.

[16] EricssonIOT. Self-driving buses in Stockholm, Sweden; 2018. Available from: https://www.ericsson.com/en/internet-of-things/trending/driverless-buses-in-stockholm-sweden.

[17] WHO. Road traffic injuries; 2018. Available from: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[18] Google. Nest Cam IQ Outdoor - Outdoor Home CCTV System - Google Store;. Available from: https://store.google.com/gb/product/nest_cam_iq_outdoor.

[19] Gauss. Triton^{TM} AI-enabled platform for real time monitoring of surgical blood loss; 2019. Available from: http://www.gausssurgical.com.

[20] McKinney SM, Sieniek M, Godbole V, Godwin J, Antropova N, Ashrafian H, et al. International evaluation of an AI system for breast cancer screening. Nature. 2020 Jan;577(7788):89–94. Available from: https://doi.org/10.1038/s41586-019-1799-6.

[21] Seibert JA. Archiving, Chapter 2: Medical Image Data Characteristics - Society for Imaging Informatics in Medicine;. Available from: https://siim.org/page/archiving_chapter2.

[22] Bankman I. Handbook of Medical Imaging: Processing and Analysis Management; pp772. 2nd ed. Academic Press; 2000.

[23] O'Shea K, Nash R. An Introduction to Convolutional Neural Networks. ArXiv e-prints. 2015 Nov;.

[24] Deshpande A. A Beginner's Guide To Understanding Convolutional Neural Networks; 2016. Available from: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/.

[25] Nwankpa C, Ijomah W, Gachagan A, Marshall S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. arXiv:181103378 [cs]. 2018 Nov;ArXiv: 1811.03378. Available from: http://arxiv.org/abs/1811.03378.

[26] Karpathy A. CS231n Convolutional Neural Networks for Visual Recognition; 2015. Available from: http://cs231n.github.io/convolutional-networks/.

[27] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:150203167 [cs]. 2015 Mar;ArXiv: 1502.03167. Available from: http://arxiv.org/abs/1502.03167.

[28] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:181004805 [cs]. 2019 May;ArXiv: 1810.04805 version: 2. Available from: http://arxiv.org/abs/1810.04805.

[29] Hu J, Shen L, Albanie S, Sun G, Wu E. Squeeze-and-Excitation Networks. arXiv:170901507 [cs]. 2019 May;ArXiv: 1709.01507. Available from: http://arxiv.org/abs/1709.01507.

[30] Kang Y. Further Attention Utilization – Efficience & Segmentation; 2019. Library Catalog: sisyphus.gitbook.io. Available from: https://sisyphus.gitbook.io/project/deep-learning-basics/classification/further-attention-utilization-classification-and-segmentation.

[31] Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, et al. Dual Attention Network for Scene Segmentation. arXiv:180902983 [cs]. 2019 Apr;ArXiv: 1809.02983. Available from: http://arxiv.org/abs/1809.02983.

[32] Li H, Xiong P, An J, Wang L. Pyramid Attention Network for Semantic Segmentation. arXiv:180510180 [cs]. 2018 Nov;ArXiv: 1805.10180. Available from: http://arxiv.org/abs/1805.10180.

[33] Glocker B. Lecture 2: Introduction to Machine Learning. Presented at Imperial College London; 2020.

[34] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:14091556 [cs]. 2015 Apr;ArXiv: 1409.1556. Available from: http://arxiv.org/abs/1409.1556.

[35] Abdelbaki A. Computer Vision Lab SS16 - P-CNN features for Action Recognition; 2016.

[36] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems 25. Curran Associates, Inc.; 2012. p. 1097–1105. Available from: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[37] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going Deeper with Convolutions. arXiv:14094842 [cs]. 2014 Sep;ArXiv: 1409.4842. Available from: http://arxiv.org/abs/1409.4842.

[38] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. arXiv:151200567 [cs]. 2015 Dec;ArXiv: 1512.00567. Available from: http://arxiv.org/abs/1512.00567.

[39] Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv:160207261 [cs]. 2016 Aug;ArXiv: 1602.07261. Available from: http://arxiv.org/abs/1602.07261.

[40] Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely Connected Convolutional Networks. arXiv:160806993 [cs]. 2018 Jan;ArXiv: 1608.06993. Available from: http://arxiv.org/abs/1608.06993.

[41] Salem M, Taheri S, Yuan JS. Utilizing Transfer Learning and Homomorphic Encryption in a Privacy Preserving and Secure Biometric Recognition System. Computers. 2018 Dec;8:3.

[42] Iglovikov V, Shvets A. TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. arXiv:180105746 [cs]. 2018 Jan;ArXiv: 1801.05746. Available from: http://arxiv.org/abs/1801.05746.

[43] Cicek O, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. arXiv:160606650 [cs]. 2016 Jun;ArXiv: 1606.06650. Available from: http://arxiv.org/abs/1606.06650.

[44] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:150504597 [cs]. 2015 May;ArXiv: 1505.04597. Available from: http://arxiv.org/abs/1505.04597.

[45] Sheth AP, Larson JA. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Comput Surv. 1990 Sep;22(3):183–236. Available from: https://doi.org/10.1145/96602.96604.

[46] Kurze T, Klems M, Bermbach D, Lenk A, Tai S, Kunze M. Cloud federation. Cloud Computing. 2011;p. 32–38.

[47] University MLD Carnegie Mellon. Federated Learning: Challenges, Methods, and Future Directions; 2019. Available from: https://blog.ml.cmu.edu/2019/11/12/federated-learning-challenges-methods-and-future-directions/.

[48] Li T, Sahu AK, Talwalkar A, Smith V. Federated Learning: Challenges, Methods, and Future Directions. arXiv:190807873 [cs, stat]. 2019 Aug;ArXiv: 1908.07873. Available from: http://arxiv.org/abs/1908.07873.

[49] Li Q, Wen Z, Wu Z, Hu S, Wang N, He B. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. arXiv:190709693 [cs, stat]. 2019 Dec;ArXiv: 1907.09693. Available from: http://arxiv.org/abs/1907.09693.

[50] López PG, Montresor A, Epema DHJ, Datta A, Higashino T, Iamnitchi A, et al. Edge-centric Computing: Vision and Challenges. Computer Communication Review. 2015;45:37–42.

[51] Bonomi F, Milito RA, Zhu J, Addepalli S. Fog computing and its role in the internet of things. In: MCC '12; 2012. .

[52] Kuflik T, Kay J, Kummerfeld B. Challenges and solutions of ubiquitous user modeling. In: Ubiquitous display environments. Springer; 2012. p. 7–30.

[53] Roy AG, Siddiqui S, Pölsterl S, Navab N, Wachinger C. BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning. arXiv:190506731 [cs, stat]. 2019 May;ArXiv: 1905.06731. Available from: http://arxiv.org/abs/1905.06731.

[54] Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, et al. Towards Federated Learning at Scale: System Design. arXiv:190201046 [cs, stat]. 2019 Mar;ArXiv: 1902.01046. Available from: http://arxiv.org/abs/1902.01046.

[55] Kuchler H. Pharma groups combine to promote drug discovery with AI; 2019. Available from: https://www.ft.com/content/ef7be832-86d0-11e9-a028-86cea8523dc2.

[56] Sheller MJ, Reina GA, Edwards B, Martin J, Bakas S. Multi-Institutional Deep Learning Modeling Without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation. arXiv:181004304 [cs, stat]. 2018 Oct;ArXiv: 1810.04304. Available from: http://arxiv.org/abs/1810.04304.

[57] Li W, Milletarì F, Xu D, Rieke N, Hancox J, Zhu W, et al. Privacy-preserving Federated Brain Tumour Segmentation. arXiv:191000962 [cs]. 2019 Oct;ArXiv: 1910.00962. Available from: http://arxiv.org/abs/1910.00962.

[58] Vepakomma P, Gupta O, Swedish T, Raskar R. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv:181200564 [cs, stat]. 2018 Dec;ArXiv: 1812.00564. Available from: http://arxiv.org/abs/1812.00564.

[59] Kang J, Xiong Z, Niyato D, Yu H, Liang YC, Kim DI. Incentive Design for Efficient Federated Learning in Mobile Networks: A Contract Theory Approach. arXiv:190507479 [cs]. 2019 Oct;ArXiv: 1905.07479. Available from: http://arxiv.org/abs/1905.07479.

[60] McMahan HB, Moore E, Ramage D, Hampson S, Arcas BAy. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:160205629 [cs]. 2017 Feb;ArXiv: 1602.05629. Available from: http://arxiv.org/abs/1602.05629.

[61] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated Optimization in Heterogeneous Networks. arXiv:181206127 [cs, stat]. 2018 Dec;ArXiv: 1812.06127. Available from: http://arxiv.org/abs/1812.06127.

[62] Tang H, Gan S, Zhang C, Zhang T, Liu J. Communication Compression for Decentralized Training. arXiv:180306443 [cs, stat]. 2019 Jan;ArXiv: 1803.06443. Available from: http://arxiv.org/abs/1803.06443.

[63] Bost R, Popa RA, Tu S, Goldwasser S. Machine learning classification over encrypted data. vol. 4324; 2015. p. 4325. Available from: https://eprint.iacr.org/2014/331.pdf.

[64] Mikulic M. Diagnostic imaging market share top medtech companies 2017 and 2024; 2019. Available from: https://www.statista.com/statistics/331739/top-global-companies-by-diagnostic-imaging-market-share/.

[65] Potvin O, Khademi A, Chouinard I, Farokhian F, Dieumegarde L, Leppert I, et al. Measurement Variability Following MRI System Upgrade. Frontiers in Neurology. 2019;10. Available from: https://www.frontiersin.org/articles/10.3389/fneur.2019.00726/full.

[66] Shokri R, Stronati M, Song C, Shmatikov V. Membership Inference Attacks against Machine Learning Models. arXiv:161005820 [cs, stat]. 2017 Mar;ArXiv: 1610.05820. Available from: http://arxiv.org/abs/1610.05820.

[67] Hitaj B, Ateniese G, Perez-Cruz F. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. arXiv:170207464 [cs, stat]. 2017 Sep;ArXiv: 1702.07464. Available from: http://arxiv.org/abs/1702.07464.

[68] Melis L, Song C, De Cristofaro E, Shmatikov V. Exploiting Unintended Feature Leakage in Collaborative Learning. arXiv:180504049 [cs]. 2018 Nov;ArXiv: 1805.04049. Available from: http://arxiv.org/abs/1805.04049.

[69] Bhagoji AN. Lecture: IBM Research - Model Poisoning Attacks in Federated Learning. Presented at Princeton University; 2018. Available from: http://www.princeton.edu/~abhagoji/files/SecML_2018_fed_learn_poison.pdf.

[70] Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How To Backdoor Federated Learning. arXiv:180700459 [cs]. 2019 Aug;ArXiv: 1807.00459. Available from: http://arxiv.org/abs/1807.00459.

[71] Blanchard P, Guerraoui R, Stainer J, others. Machine learning with adversaries: Byzantine tolerant gradient descent. In: Advances in Neural Information Processing Systems; 2017. p. 119–129. Available from: https://papers.nips.cc/paper/6617-machine-learning-with-adversaries-byzantine-tolerant-gradient-descent.pdf.

[72] Dwork C, Roth A. The Algorithmic Foundations of Differential Privacy. Found Trends Theor Comput Sci. 2014 Aug;9(3–4):211–407. Available from: https://doi.org/10.1561/0400000042.

[73] Truex S, Baracaldo N, Anwar A, Steinke T, Ludwig H, Zhang R, et al. A Hybrid Approach to Privacy-Preserving Federated Learning. arXiv:181203224 [cs, stat]. 2019 Aug;ArXiv: 1812.03224. Available from: http://arxiv.org/abs/1812.03224.

[74] Dulay N. Lecture: Privacy Engineering Part II. Presented at Imperial College London; 2019.

[75] Zama; 2019. Available from: https://zama.ai/.

[76] Hardy S, Henecka W, Ivey-Law H, Nock R, Patrini G, Smith G, et al. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv:171110677 [cs]. 2017 Nov;ArXiv: 1711.10677. Available from: http://arxiv.org/abs/1711.10677.

[77] V Nikolaenko, U Weinsberg, S Ioannidis, M Joye, D Boneh, N Taft. Privacy-Preserving Ridge Regression on Hundreds of Millions of Records. In: 2013 IEEE Symposium on Security and Privacy; 2013. p. 334–348.

[78] M Sabt, M Achemlal, A Bouabdallah. Trusted Execution Environment: What It is, and What It is Not. In: 2015 IEEE Trustcom/BigDataSE/ISPA. vol. 1; 2015. p. 57–64.

[79] J Ekberg, K Kostiainen, N Asokan. The Untapped Potential of Trusted Execution Environments on Mobile Devices. IEEE Security & Privacy. 2014 Aug;12(4):29–37.

[80] Gu Z, Jamjoom H, Su D, Huang H, Zhang J, Ma T, 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. p. 336–348. Available from: https://arxiv.org/abs/1812.03230.

[81] Menze BH, Jakab A, Bauer S, Kalpathy-Cramer J, Farahani K, Kirby J, et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). IEEE Transactions on Medical Imaging. 2015 Oct;34(10):1993–2024.

[82] Bakas S, Akbari H, Sotiras A, Bilello M, Rozycki M, Kirby JS, et al. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. Scientific Data. 2017;4:170117.

[83] Bakas S, Reyes M, Jakab A, Bauer S, Rempfler M, Crimi A, et al. Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge. arXiv:181102629 [cs, stat]. 2019 Apr;ArXiv: 1811.02629. Available from: http://arxiv.org/abs/1811.02629.

[84] Gupta O, Raskar R. Distributed learning of deep neural network over multiple agents. arXiv:181006060 [cs, stat]. 2018 Oct;ArXiv: 1810.06060. Available from: http://arxiv.org/abs/1810.06060.

[85] CheXpert: A Large Dataset of Chest X-Rays and Competition for Automated Chest X-Ray Interpretation.;. Available from: https://stanfordmlgroup.github.io/competitions/chexpert/.

[86] Wenwu Y, Yao J, Xue H, Li Y. jfhealthcare/Chexpert. JF Healthcare; 2020. Original-date: 2019-11-03T09:10:37Z. Available from: https://github.com/jfhealthcare/Chexpert.

[87] Isensee F, Kickingereder P, Wick W, Bendszus M, Maier-Hein KH. Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge. arXiv:180210508 [cs]. 2018 Feb;ArXiv: 1802.10508 version: 1. Available from: http://arxiv.org/abs/1802.10508.