# Federated Machine Learning

## A research project

Karan Samani
1161081087
Final Year Project BSc. Computer Science (Hons.)
Supervisor: Dr. Derek Bridge
Department of Computer Science
University College Cork
March 1, 2020

# Abstract

In this report we will see what is the motivation behind the need of a federated process for machine learning, a process that preserves privacy. A brief overview of how machine learning and neural networks work will be provided which is required to understand how the federated learning process works. After doing so, we will be moving onto the implementing the federated approach proposed by Google.

Google's approach will be implemented from scratch and also by using Tensorflow's Federated framework. The later being used as a sanity check to confirm that the implementation from scratch is indeed correct, doing so by running a few experiments. After doing so, we will move on to exploring extended approaches to the federated learning idea based on a weighted and selective approach on a global level and a user level.

At the end we will have a few experiments to compare all the approaches, which are, traditional machine learning, federated learning and the several extended approaches that will be explored.

# Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award.

I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award;
- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Signed:..............................................................

Date:................................................................

# Acknowledgements

Thanks to Dr. Derek Bridge for being my supervisor and dealing with me over the last few months, pushing me to my limits and supporting me throughout the year. Ever since first year, I wanted to do my final year project with him and the experience of doing so has lived up to my expectations.

Thanks to the Alexander Baran-Harper's channel on YouTube where I learnt how to use use LaTeX.

# Contents

**Declaration** . . . . . . . . . . . . . . . . . . . . . . . . . . . **i**

**Abstract** . . . . . . . . . . . . . . . . . . . . . . . . . . . . **ii**

**Acknowledgements** . . . . . . . . . . . . . . . . . . . . . . . **iii**

**List of Figures** . . . . . . . . . . . . . . . . . . . . . . . . **vi**

**1 Introduction** . . . . . . . . . . . . . . . . . . . . . . . . . **1**
1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.3 Privacy concerns . . . . . . . . . . . . . . . . . . . . . . 3
1.4 Outline for the fix (rename this) . . . . . . . . . . . . . . 3

**2 Literature Review** . . . . . . . . . . . . . . . . . . . . . . **4**
2.1 Machine learning . . . . . . . . . . . . . . . . . . . . . . 4
2.2 Neural Networks . . . . . . . . . . . . . . . . . . . . . . 7
2.3 Neurons . . . . . . . . . . . . . . . . . . . . . . . . . . 7
2.4 Architecture . . . . . . . . . . . . . . . . . . . . . . . . 8
2.4.1 Dense . . . . . . . . . . . . . . . . . . . . . . . . . 9
2.4.2 Convolutional . . . . . . . . . . . . . . . . . . . . . 9
2.5 Training . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.6 Differential Privacy . . . . . . . . . . . . . . . . . . . . . 11
2.7 Federated Machine Learning . . . . . . . . . . . . . . . . . 12
2.7.1 Benefits . . . . . . . . . . . . . . . . . . . . . . . . 13
2.7.2 Drawbacks . . . . . . . . . . . . . . . . . . . . . . . 13
2.8 Google doing it . . . . . . . . . . . . . . . . . . . . . . . 14
2.9 Privacy . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.10 What has been done by others . . . . . . . . . . . . . . . 14

**3 Core Design** . . . . . . . . . . . . . . . . . . . . . . . . . **15**
3.1 building and designing this . . . . . . . . . . . . . . . . . 15
3.2 Implemenatation . . . . . . . . . . . . . . . . . . . . . . 15
3.3 Global . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
3.4 My implementation . . . . . . . . . . . . . . . . . . . . . 15
3.5 Google's Tensorflow version . . . . . . . . . . . . . . . . . 15

**4 Extended Ideas** . . . . . **16**
4.1 Central Server . . . . . 16
4.1.1 Weighted Average . . . . . 16
4.1.2 Excluding based on std dev . . . . . 16
4.1.3 Local only . . . . . 16
4.2 Peer to Peer . . . . . 16
4.2.1 Weighted Average . . . . . 16
4.2.2 Excluding based on std dev . . . . . 16
4.2.3 Local only . . . . . 16
4.2.4 testing . . . . . 16

**5 Datasets used** . . . . . **17**
5.1 Gestures dataset . . . . . 17
5.2 Image dataset . . . . . 17

**6 Experimentation/Testing** . . . . . **18**
6.1 Designing the framework . . . . . 18
6.2 Graphing . . . . . 18

**7 Testing user weights on global data** . . . . . **19**

**8 Conclusions and future work** . . . . . **20**

**9 notes** . . . . . **20**

# List of Figures

1 High level visual representation of what fields there are in the study of AI. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
2 Elbow curve [7]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
3 A very simple decision tree [5]. . . . . . . . . . . . . . . . . . . . . . 6
4 Major components of a neuron [3]. . . . . . . . . . . . . . . . . . . 8
5 Basic dense neural network [1] . . . . . . . . . . . . . . . . . . . . . 9
6 How a convolution works [2]. . . . . . . . . . . . . . . . . . . . . . . 10

# 1 Introduction

## 1.1 Background

Modern day edge devices have a wealth of data on them and have more than enough computation power to run complex calculations on them with ease. These devices can range from a personal computer to a smart phone. In a world where data is power, access to the data on these devices is highly advantageous.

Artificial Intelligence (AI) can be described as the ability for a system to show “intelligence”. Intelligence, as Dr. Derek Bridge put it, is the ability for a system to act autonomously and rationally when faced with disorder, uncertainty, imprecision and intractability. Machine Learning, a branch of AI, is based on the idea of creating a model that recognises patterns from data to be able to solve problems. To be able to do so effectively, one must have access to a lot of data. More data equates to a higher probability of having a more robust and better overall model. If a model can look and learn from more data, the chances are that it can generalise well, and that is the ideal goal.

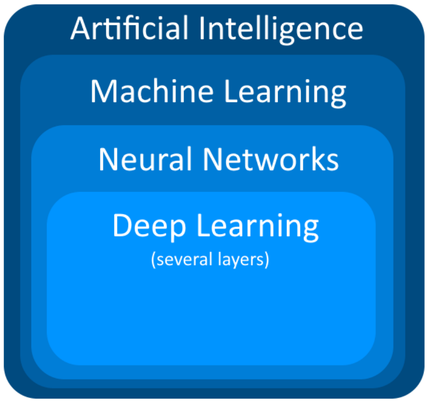

Figure 1: High level visual representation of what fields there are in the study of AI.

The solution to having well trained models seems straight forward. Use the vast amounts data from the edge devices to train a model that, in theory, should be fairly accurate. To do so, the users must give their data to a server so that the server can then train the model on the data that was just provided to it by the edge users. This trained model can then be used by everyone to predict unseen samples. But there are some more complications. Users may not want to share their data but still want the benefits of having a model trained on everyone's data.

## 1.2 Motivation

Artificial Intelligence, specifically Machine Learning, was supposed to be the idea that took the world by storm. A piece of software that could make a machine essentially autonomous. It was supposed to be better in every way, never getting tired, reaching at least the same competency levels as humans and quite possibly even better. So much so that the media started talking about AI putting people out of a job and eventually taking over the world at some point by making far fetched references to the popular movie series Terminator.

Needless to say, this is not accurate. The idea of a general purpose AI, formally called Artificial General Intelligence, is no where near attainable. Current AI applications are good at specific tasks, and only those tasks because they have a narrow scope. Even with their narrow scope, there are more topics that need to be addressed. Such as the security issues that arise with better AI systems. One can use AI to create cyberweapons that can be used for hacking and spreading misinformation. Along with cyberwarfare, AI can be used to make traditional weapons more lethal. For instance with drones using image recognition amongst other things to target specific areas. At first glance this seems like a good idea as it should result in less casualties. But, AI systems are not 100% accurate so they could lead to an accidental strike. And if someone is able to hack into the system, they can make the drones target literally anything and anyone. Even in the right hands this technology can lead to disasters, let alone the wrong hands. Then there is the privacy aspect as well where people may not want to share their potentially sensitive private data under the fear that it can be misused. The idea of privacy will be the focus of this project.

## 1.3 Privacy concerns

There are numerous examples where people may not want to share their personal data. For instance, for the training of a model that deals with predictive text, the input data would require essentially everything that a user may type into their device. It is pretty obvious to see why some people may not want to share the messages and other content that they type on their devices. It is a clear invasion of their privacy.

Another application could be training on images for classification purposes. People may not want to share images, which may include sensitive images, that they have stored on their devices with a third party. This can be extended to an even more sensitive topic of medical imaging where people may not want to share something like the X-Rays of their bodies.

In general, people are sceptical of sharing personal data. But there is still the need to train a model that has had exposure to as much data as possible.

## 1.4 Outline for the fix (rename this)

This was the motivation behind the idea of Federated Learning which was an idea proposed by Google in 2016 [4]. The idea of not having to share your data with someone else and yet still have the benefits of having a model that has exposure to their data. This will be the main point of interest in this report.

Google's approach towards federated learning will be further explain in later chapters. Following Google's approach, several extended ideas based on weighted and selective approaches will be discussed and their implementation explained. Along with that, outputs from some experiments to show how the extensions compare with Google's approach of federated learning and the traditional approach of machine learning in this context.

# 2 Literature Review

## 2.1 Machine learning

Machine learning is a very broad field which includes a lot of different learning methodologies. Learning can take place in a supervised context where the dataset is labelled, or in an unsupervised context where the data is not labelled.

For unsupervised learning, the most well known algorithm is k-means clustering. This algorithm aims to find structure within the dataset and aims to cluster the closely related groups together into a total of $k$ clusters. The caveat here is that the value of $k$, the number of clusters, needs to be known before hand. Although there is a way to find an optimal value for $k$ when its value it not known. This is done by producing a graph of Inertia vs k, where inertia is the measure of how dense are the clusters formed. This graph is called an elbow curve because it produces a curve that looks like an elbow. And the point at which the graph turns, the elbow (3 in Figure 2), can be thought of as a good value of $k$ for the dataset. An extension to k-means is k-means++ where through clever selection of the initial points, a better result is obtained.

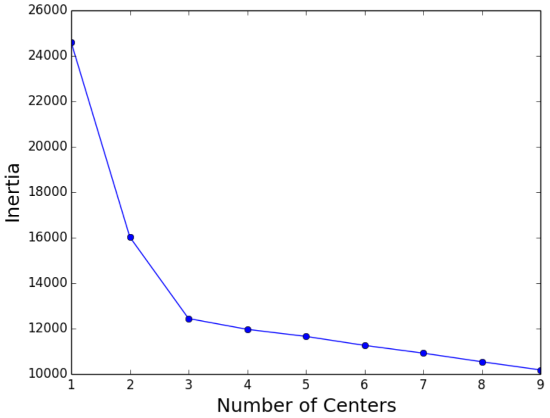

Figure 2: Elbow curve [7].

In supervised learning, the task is to learn a function that maps an input to an output, based on sample input-output pairs. In this project, the focus is on supervised learning where the aim is not to find structure within a dataset but rather trying to learn how to map the input data to a desired output. There are quite a few methods of finding the right function that fits the dataset, ranging from simple functions to very complicated functions.

One of the simplest approaches for supervised learning is called Linear Regression, which aims to solve regression problems. The data that is provided to it are pairs consisting of input features (as a vector) and the desired output. Based on the set of input pairs provided, linear regression tries to find a linear function, which is a set of coefficients $\beta$ for all the features, that fits the dataset well. The set of input pairs provided is called the training set. To find the best possible function to fit the data, the idea of a loss function is used. This essentially says how distant the predictions are from the desired output. Loss in this case is usually mean squared error (MSE). The error $e$, as shown below, is the difference between the actual value and the predicted value.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

The values of $\beta$ that fit the dataset the best would be the one with the lowest value for MSE on the training data. A naive way to solve this problem would be to iterate over all the infinitely many $\beta$s and run predictions on the training set to give an output. Use that output and the desired output to calculate the loss values for all the $\beta$ combinations. The $\beta$ values that give the lowest MSE value (loss) would be chosen as the solution. But there are more sophisticated methods of solving this like calculus and gradient descent. The latter can intuitively be thought as taking, usually small, steps in the direction that reduces the loss.

Logistic Regression follows the same basic principle as linear regression but is used for classification purposes instead. Classification problems are about predicting if a sample belongs to a certain class or not. The input data for this is similar to linear regression with it being a pair of input features (as a vector) but the desired output being a label representing a class of objects (like “dog”). Logistic regression still works off of building linear models using $\beta$ under the hood and predicts numbers that are probabilities of a certain input being part of a certain class. For the prediction, input features are passed through a sigmoid function $\sigma$ (also called the logit function) which outputs a number between 0 and 1, lets call this $h_\beta$.

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

$$h_\beta(x) = \sigma(x\beta)$$

These numbers are interpreted as the probability of the input being part of a class, usually the positive class (a class which requires action and is labelled as 1). Based on the probability, the input is classified to a class $\hat{y}$. The idea of reducing the loss still applies, but a more complication loss function is used in this case.

$$\hat{y} = \begin{cases} 0 & \text{if } Prob(\hat{y} = 1|x) < 0.5 \\ 1 & \text{if } Prob(\hat{y} = 1|x) \geq 0.5 \end{cases}$$

Linear regression and logistic regression are both based off of a linear function so they cant deal with complex data very well. Decision trees are an alternative that can better fit complex datasets can be used for both, regression and classification problems. They are very readable compared to other approaches. At a very high level, the structure of the tree dictates what path a sample input should take. The inner nodes in split the data based on conditions and the leafs represent the decisions made on the samples. A different loss function is used to optimise the answer. But even decision trees are not complex enough for the needs of this project.

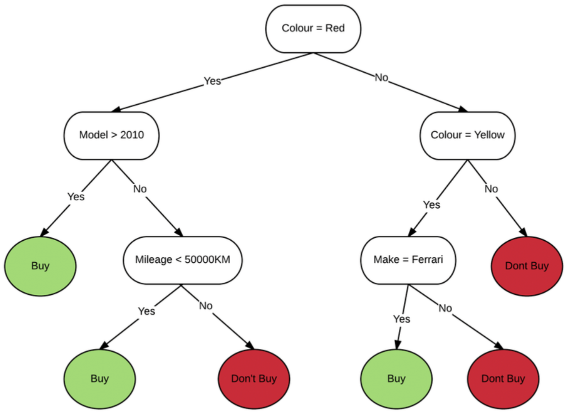

Figure 3: A very simple decision tree [5].

Neural networks have become the go to solution in recent times for pretty much all problems. They can cater to a wide range of problems, including very complex problems such as image classification, image localisation and natural language processing. They outperform pretty much any other approaches on almost every task. This is why they will be used exclusively in this project.

## 2.2 Neural Networks

Neural networks, as seen before, are a subset of machine learning. Neural networks are not a new idea, they have been around for decades. But they only really took off in recent years with the emergence of better and more affordable hardware. The basic idea behind the workings of a neural network are quite straight forward. A model is defined and data is passed through it to make predictions. Then, based on the loss, some adjustments are made to the parameters learnt. This whole process is repeated by iterating over the dataset a set number of times during the training process. To start off, the idea of a neuron must be explained which are the basic building blocks of a neural network.

## 2.3 Neurons

A neuron can be thought of as a node that takes in a weighted sum of its inputs, passes it through a function (called the activation function) and outputs a value to be used later. The inputs received are from either the input layer neurons or the hidden layer neurons. The activation function used below is a step function that outputs a 1 if the weighted sum is more than a certain value (0 in this case), otherwise it outputs a 0. The input data point labelled 1 in Figure 4 is used to represent a bias $b$. Because the input is 1, multiplying it with a weight is guaranteed to give a value which will act as a number that is always used in the summing process later before the value is passed into the activation function.

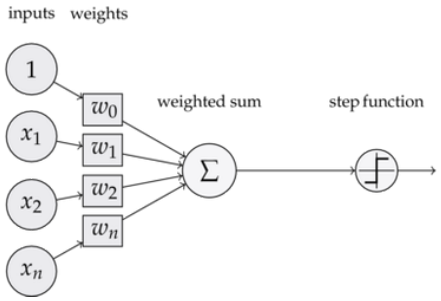

Figure 4: Major components of a neuron [3].

Note that, for brevity, the weighted sum can be written in a vectorised format.

$$w \cdot x \equiv \sum_j w_j x_j$$

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x \leq 0 \\ 1 & \text{if } w \cdot x > 0 \end{cases}$$

The activation function can be swapped out from the step function that was being used earlier with some other activation function such as ReLU (Rectified Logic Unit), sigmoid function, etc. ReLU takes the max of either 0 or the weighted sum and uses that as its output.

$$\text{output} = \max\{0, w \cdot x + b\} \qquad (1)$$

## 2.4 Architecture

In a neural network, there are layers of such neurons. These are usually broken down in three parts, the input layer, the hidden layer and then the output layer. The input layer is not actually a layer of neurons but rather just a representation of the input data as a layer that connects to the hidden layers. The hidden layers can contain any number of layers with any number of neurons for the layers.

After the hidden layers is the output layer. This layer is responsible for outputting the predictions. The output layer usually has its own activation function which depends on the application, i.e., if it is a classification problem or a regression problem. Since this project focuses on multi-class classification problems, we will be using the softmax activation function for the output layer.

The layered structure described above is called the architecture of the model. Some of the common used architectures are densely connected layers (Section 2.4.1) and convolutional layers (Section 2.4.2). More information on the aforementioned and more architectures can be found in the book about Deep Learning by F. Chollet [2].

### 2.4.1 Dense

These are one of the most straight forward architectures and are generally the way most neural networks end. They are quite useful when placed as the last few layers, especially for classification purposes. In these, all the nodes are connected to every node in the subsequent layer. The input for these layers are flattened data, which can be thought of as a list of input data where nested lists are not allowed. An example can be seen in Figure 5.

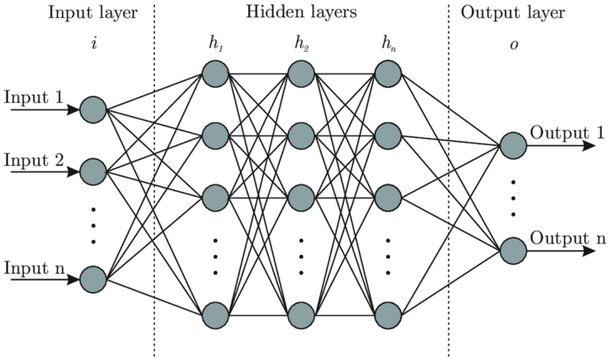

Figure 5: Basic dense neural network [1]

### 2.4.2 Convolutional

These are more complicated than the previously mentioned dense layers. The input data here is structured and not flattened, unlike dense nodes. Their basic idea is to find patterns in the input data and make them more abstract as the layer count increases. They indicate the presence of certain shapes. Most common use for convolutional layers is in image classification.

A convolutional layer has a window (called the kernel) that is used to look over the input data and recognise patterns localised in that window. Every layer has a "depth" number of sub-layers which can be thought of as the number of patterns that the layer is trying to recognise and they are called the feature maps of the layer. For example if the depth is 3, the convolutional layer has 3 feature maps and will look at 3 kinds of patterns. The nodes in the following convolutional layer are connected to every node in the window of the previous layer, including the sub-layers as well. The set of weights that connect a node to the sub-layer nodes in the following layer are the same within the window. Figure 6 does a good job of giving an idea about how this looks.

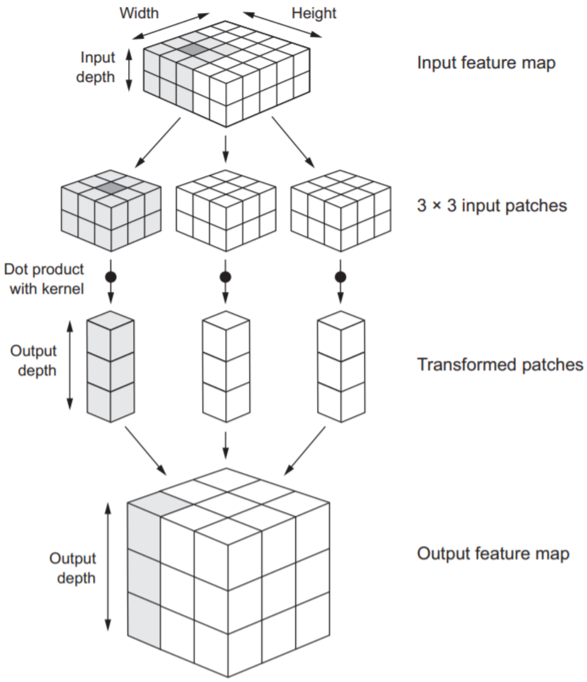

Figure 6: How a convolution works [2].

Convolutional layers are often followed by some maxpooling layers, which reduce the output size from the convolutional layers. This can be done for many reasons, such as saving memory and making the computations required less intensive. At the end of a series of convolutional layers, there is a series of dense layers that are used to give the predicted output(s).

## 2.5 Training

For a neural network to work, the weights with which the nodes are connected need to be tweaked and the process of doing so is called training the model. The model is trained on a training set, which is a subset of the whole dataset. This is done to avoid leakage between the testing data and the training data. We will go through the high level algorithm with which neural networks are trained. More information on this and the idea of neural networks can be found in the (online) book “Neural Networks and Deep Learning, Michael A. Nielsen” [6] and the book “Deep Learning with Python, Francois Chollet” [2].

The model is initialised with random weights before training begins. After that, the training data is then through the model to make predictions. These predictions are then compared with the actual values, and the loss is calculated using a loss function. For regression, mse is used and for classification sparse categorical crossentropy is used. The details in which they work are not required, but the idea of finding out how far off the predictions were from the actual value still holds. Using an methods like gradient descent or Adam (based on gradient descent and other methods, an algorithm called back propagation tweaks the weights in a way that would hopefully reduce the loss. These updates can be made based on every example, batches of examples or the entire dataset as a whole. Generally, the batched approach is taken and the entire dataset is broken down into batches of training data.

This whole process of predicting, calculating loss and tweaking the weights can be done several times by iterating over the whole training set several times. The number of times that the entire training set has been gone over is called the number of epochs that the network was trained on. This needs to be tweaked manually as it could lead to under fitting or over fitting the model. Which means, the model is too general or too specific, respectively, to be useful in actual usage.

## 2.6 Differential Privacy

linkage attacks, netflix and imdb data to identify only good for large dataset, else bad complex easier to do real data and anonymise

deidentify data outliers fields

want to decouple learning about an individual and learning about the population

analysis output is not dependant on a particular individual and will be the sameif they are not incuded

adjacent dataset n+1 and n sized dataset

algo m

## 2.7 Federated Machine Learning

Traditionally in ML, a server (lets call this this central agent) would have access to all the data. Presumably collected by all the edge users (lets call these just users) sending their data to the central agent. The central agent would then train a single model using all the data that it has and anyone could then access the model to use it.

In Federated ML, there is no concept of having access to all the data. So as a workaround, an identical copy of the model is sent to every participating user. All the users then train their local copy of the model on just their own local data. This would usually be done without the user explicitly running the training command. The devices would start the training process for their model based on certain conditions such as whether or not the devices are currently charging, if they are on WiFi, have been idle for some time (for instance night time), etc. After the training process has been completed, the model would have learnt some parameters (the weights for the neuron connections, as seen in Section 2.2). These weights are then sent to a central agent which would average the weights received from all the users and send the averaged weights back to the users. Instead of sending all the weights, an alternative would be to send only the changes made to the weights. The latter is what Google does. After receiving the updated weights from the central agent, the users would update their local models with these new averaged weights and start the training process again. This back and forth of training, averaging and training again can take place several times. At the end (if there is an end), the resulting product can be pretty close to the traditional ML approach. Although, it must be noted that depending on the application, it may not be close or some changes may be required.

### 2.7.1 Benefits

Federated machine learning has a few benefits. Firstly and most importantly, all the training takes places on the only the edge devices. This means that the users do not have to share their data with anyone. The only data they share are the parameters learnt from the training process that took place on their local data. And it is impossible to recreate the original data from the just the parameters. Add to that the idea of Secure Aggregation [8] and one can very confidently say that the idea of privacy is held up to the highest standard. Secure aggregation is where the data is aggregated in stages. Instead of all averaging being done at the server where some data can possibly be reverse engineered, there are intermediary steps where the data is averaged and then the central agent finally averages those averages. But we will not be focussing on secure aggregating in this project.

Secondly, the fact that all the training runs on edge devices means that there is no need for an investment into building a large training infrastructure by a company. The edge devices will do all the hard work of the model training and share the results which a central agent would then use quite trivially. So this is a better use of resources as a whole as idle devices would not just stay idle and would take part in the training process.

### 2.7.2 Drawbacks

As mentioned before, the federated approach can lead to similar performance as the the traditional approach. But this depends on the distribution of the data as we will see in later sections (**??**. If the dataset amongst all the users is very similar, then the overall result of the federated process can be very similar to the traditional approach. But if the users have skewed datasets, as is the case in real life, the traditional approach is generally better. It is a trade-off of privacy vs. model performance that must be decided upon when deciding between the federated approach and the traditional approach respectively.

## 2.8 Google doing it

## 2.9 Privacy

## 2.10 What has been done by others

# 3 Core Design

## 3.1 building and designing this

## 3.2 Implemenatation

## 3.3 Global

## 3.4 My implementation

## 3.5 Google's Tensorflow version

Sanity check was close enough to mine

# 4 Extended Ideas

## 4.1 Central Server

### 4.1.1 Weighted Average

### 4.1.2 Excluding based on std dev

### 4.1.3 Local only

## 4.2 Peer to Peer

### 4.2.1 Weighted Average

### 4.2.2 Excluding based on std dev

### 4.2.3 Local only

### 4.2.4 testing

# 5 Datasets used

Dataset and expereiments the same

Or diff chapters for images • Comoing up with decent model • Traditional ml results

## 5.1 Gestures dataset

found issues so moved on

## 5.2 Image dataset

h5 files

artificial users

# 6 Experimentation/Testing

## 6.1 Designing the framework

## 6.2 Graphing

For each dataset show the otuput we got

# 7 Testing user weights on global data

Analysis
Design
Implementation
Evaluation
Conclusions

third person
present or past tense

design talks about apckages and stuff. convince them its good describe test set Can just saw 6 other graphs were xyz for whatever reason Give a reason to why im shoeing something

# 8 Conclusions and future work

can use ”i” in here

# 9 notes

Not written in the passice Name the agent Describe the system Give it a name User Name parts of the system Edge agent, central agent Present and past

# Bibliography

[1] F. Bre, J. M. Gimenez, and V. D. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using artificial neural networks,” *Energy and Buildings*, vol. 158, pp. 1429–1441, 2018.

[2] F. Chollet, *Deep Learning with Python*, 1st. USA: Manning Publications Co., 2017, ISBN: 1617294438.

[3] JustinB. (2017). Introduction to perceptron: Neural network.

[4] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, *Federated optimization: Distributed machine learning for on-device intelligence*, 2016. arXiv: `1610.02527 [cs.LG]`.

[5] A. Moodley, “Language identification with decision trees: Identification of individual words in the south african languages,” PhD thesis, Jan. 2016. DOI: `10.13140/RG.2.2.25539.81445`.

[6] M. A. Nielsen, “Neural networks and deep learning,” 2015.

[7] RJ Nowling. (2015). Customer segmentation with the bigpetstore multinomial model.

[8] A. Segal, A. Marcedone, B. Kreuter, D. Ramage, H. B. McMahan, K. Seth, K. Bonawitz, S. Patel, and V. Ivanov, “Practical secure aggregation for privacy-preserving machine learning,” in *CCS*, 2017.