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.

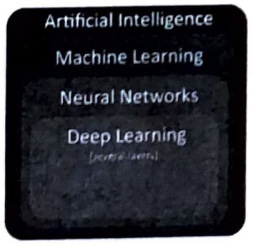

Figure 1: High level visual representation of what fields there are in the study of AI.

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 explained 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.

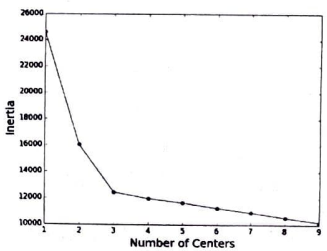

Figure 2: Elbow curve[7].

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 β 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 β 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 βs 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 β combinations. The β 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 β 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 σ (also called the logit function) which

outputs a number between 0 and 1, lets call this h_β.

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

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

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 can't 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.

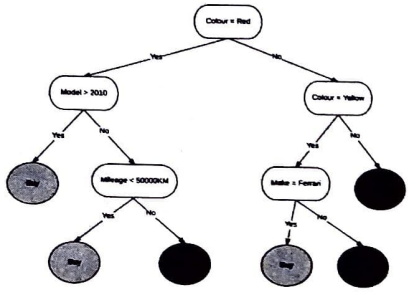

Figure 3: A very simple decision tree[5].

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.

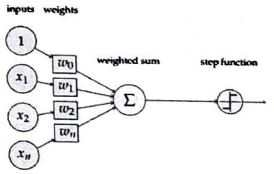

Figure 4: Major components of a neuron[3].

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.

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

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).

2.4.1 Dense Layers

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.

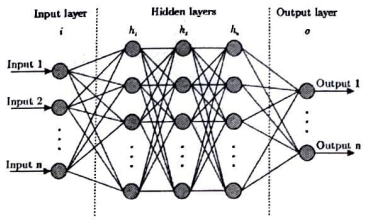

Figure 5: Basic dense neural network [1]

2.4.2 Convolutional Layers

These are more complicated than the previously mentioned dense layers. The input data here is structured and not flattened, as is the case with 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 kernel or a window 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 patterns that the whole layer is trying to learn called feature maps. For instance, with the depth being 3, the convolutional layer has 3 feature maps. Each feature map in a layer is connected to all the feature maps in the subsequent layer. And the layer

Figure 6 does a good job of giving an idea about how this process works. But the main concept remains the same. There are underlying weights that are being tweaked to learn patterns.

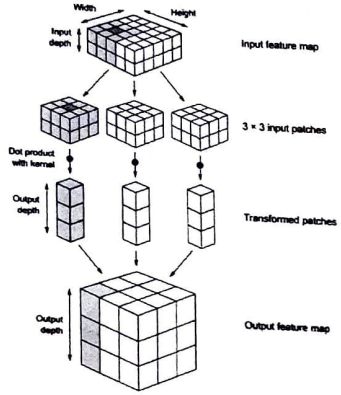

Figure 6: How a convolution works [2].

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

The high level algorithm of training a model is as such. First the training data is fed through the model (initialised with random weights) to make predictions. These predictions are then compared with the actual values. Then, using an algorithm called back propagation, the values of the weights for every neuron are updated in a way so that the predictions are closer to the true values. 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]

2.6 Federated Machine Learning

Traditionally in ML, a server (a central user) would have access to all the data. Presumably collected by everyone sending their data to the server. And the server would train a single model on all the data that it has and people could then access the model to use it.

In Federated ML, there is no concept of having access to all the data. So, to combat this, an identical copy of the model is sent to every participating edge 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). These weights are then sent to a server which would average the weights received from all the users and send the averaged weights back to the users. The users would then 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.

2.6.1 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 just the parameters. Add to that the idea of Secure Aggregation (Sub-Section 3.1) and one can very confidently say that the idea of privacy is held up to the highest standard.

Secondly, the fact that all the training runs on edge devices means that there is no need for an investment into a training infrastructure by a company. The devices will do all the hard work and share the results. So this is a better use of devices that are quite possibly not being used at a given time.

2.6.2 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 that must be decided upon: privacy vs. model performance.

3 Literature Review

How to train Google doing it Privacy

3.1 Secure Aggregation

Essentially where the data is aggregated in stages instead of all averaging being done at the server. Idea of associativity or commutativity? Something like that.

Federated

3.2 Is god real?

3.3 What is the meaning of life?

3.4 42

3.5 Differential privacy

3.6 Google doing it

3.7 Privacy

3.8 What has been done by others