# TensorFlow Federated - Document Classification Report

Table of Contents

[**1. Introduction**](#_kfs6hwo5ewe1) **2**

[**2. Problem Statement**](#_qlck6bbxpacx) **2**

[**3. Data Extraction and Creation**](#_fmqur2l73ijl) **3**

[**4. Centralized Model Training**](#_5zqocha5z4bp) **4**

[**5. Decentralized Model Training**](#_b7n2ii37t5mu) **5**

[5.1 TFF - Platform setup](#_ruj7cy2uixyi) 5

[5.2 TFF - Model Architecture and Training](#_jfbtx3cr9xxn) 5

[**6. Experiment result**](#_hnlxmltskhgs) **5**

[**7. Conclusion and Future Work**](#_76x6f26dksa0) **6**

[7.1 Architecture](#_ux6r87al0hgc) 6

[7.2 Flow Diagram](#_qc9v5sclnfh) 7

[**8. References**](#_mzj8x6po2inf) **7**

**Abstract**

Conventional Machine Learning approaches require centralizing the data before training a model on that data. Federated Learning is simply the decentralized version of machine learning where a centralized model is trained on decentralized data1. This experiment solves a document classification problem using both the centralized and federated approaches to compare their results in terms of accuracy and computing expenses. The dataset consists of different types of IRS tax forms. The usage of federated learning addresses some of the critical issues associated with data including privacy and integrity while generating similarly accurate results as its centralized counterpart. In this experiment, although the training for federated approach is computationally more expensive than centralized, both the approaches yielded a test accuracy of 100% with the additional advantages that federated learning possesses.

## 1. Introduction

Federated Learning is a technique in Machine Learning that trains a model on data across multiple devices known as edge nodes. This method addresses critical issues related to Data Privacy and Security as the data does not leave the node. Federated Learning is in contrast to the traditional Machine Learning techniques where a centralized approach is followed requiring all the data to be present in a single central server where training is initialized.

An important characteristic of Federated Learning is that the training data are non-iid, that is, data residing inside an edge node locally fail to represent the overall distribution. This provides a challenge to the Federated Learning phenomenon.

In this experiment, a FedAvg algorithm is used which aggregates the results of different models built on individual client nodes and computes a new model through an averaging process1.

## 2. Problem Statement

The goal of this use-case is to classify the different types of IRS tax forms using both the centralized and federated approaches to compare their results. Currently, for the scope of this project, 11 types of tax forms were used. Given that a form is passed as an input to the model, the model would predict the type of the tax form. Below are the types of forms and the number of examples that were available.

|  |  |
| --- | --- |
| **Tax Form Type** | **Count** |
| w2 | 150 |
| 1040 | 150 |
| 1099 int | 112 |
| w4p | 102 |
| w9 | 102 |
| 1040sr | 100 |
| 1099 misc | 98 |
| 1099k | 98 |
| 1099 div | 98 |
| 1099g | 98 |
| w4 | 100 |
| **Total** | **1208** |

Table 1: Types of Tax Forms and number of examples

## 3. Data Extraction and Creation

In order to create the dataset for this use-case, tax form images that were available online as well as programmatically generated datasets were used. W2 and 1040 classes were available online and the remaining classes were generated programmatically using Python to insert text into specific locations of a template tax form thereby creating an example. The size of each image varies from 100 KB to 600 KB.

An example for the class W2 which can be found online is below.

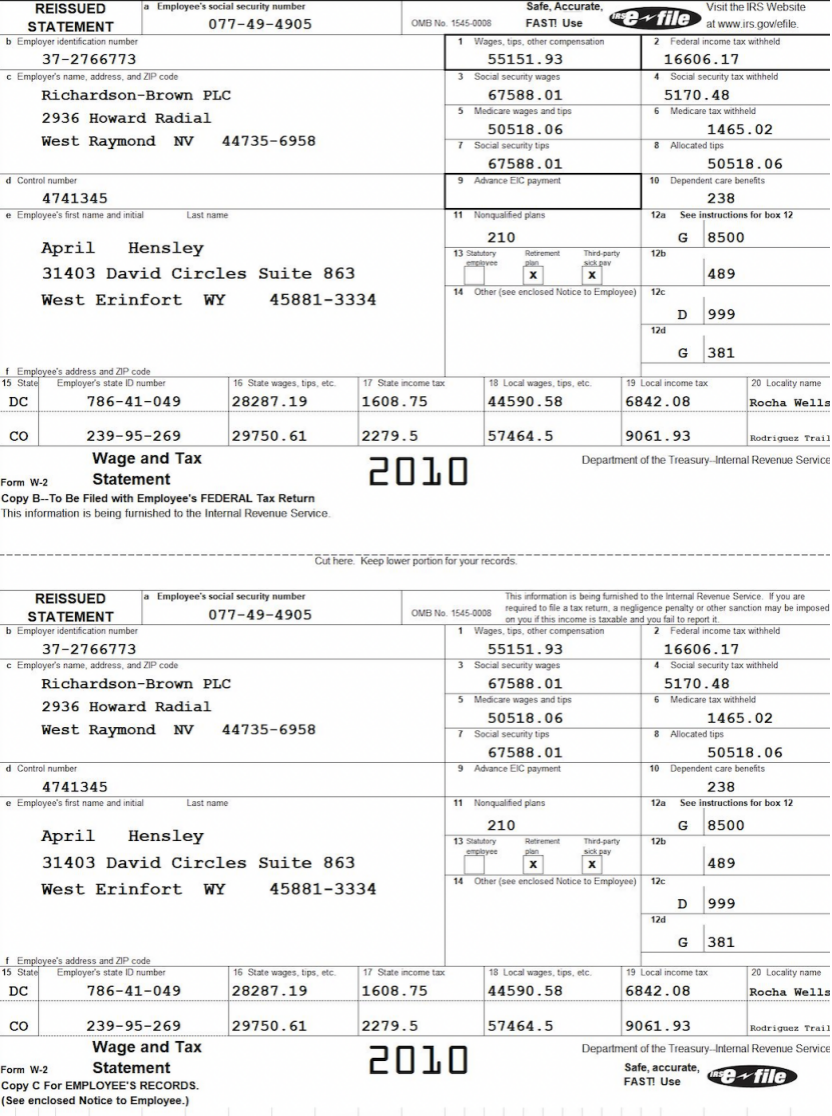


Figure 1: Tax image of W2 found online

An example for 1099-MISC tax form which is generated programmatically.

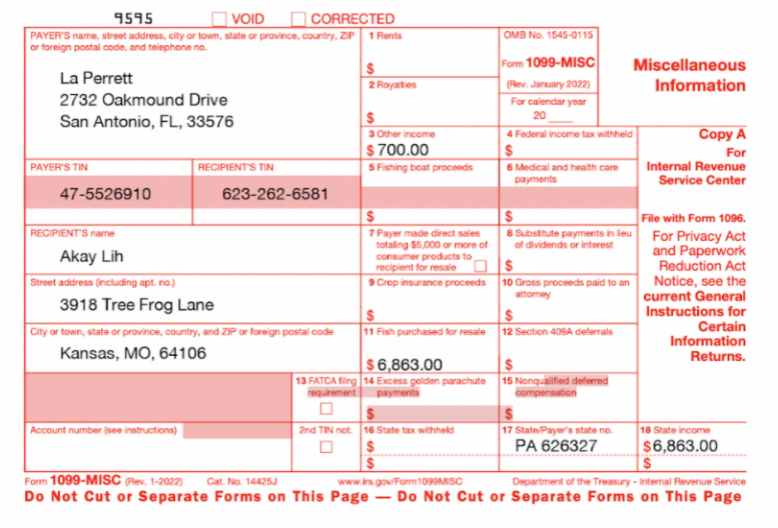


Figure 2: Tax image of 1099-MISC generated by the program

Optical Character Recognition (OCR) is then used to extract the text from the images and generate the dataset. After the dataset is generated, it is split into Train (85%) and Test (15%) datasets. The train dataset is transformed using count vectorizer and TF-IDF to obtain the frequency distribution of the tokens. This training dataset is then used to train a Keras model by using both the centralized as well as the decentralized approach and their accuracy is compared.

## 4. Centralized Model Training

A Neural Network algorithm in Keras is used to train a model to classify the tax forms. Adam algorithm is used as the gradient descent optimizer and for each training batch, 8 images are used. The model training is then initialized for 10 epochs. Below are the specifications of the centralized model.

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer Type** | **Number of Units** | **Input Shape** | **Activation Function** |
| Input Layer | 128 | (None,10667) | Relu |
| Output Layer | 11 | (None,128) | Softmax |

Table 2: Centralized Model Specifications

## 5. Decentralized Model Training

#### 5.1 TFF - Platform setup

In order to simulate the real world application of training in a federated environment, 10 VM instances are created to represent client devices or edge nodes. gRPC servers are started in these edge nodes to establish a connection with the orchestrator or server. The orchestrator serves as the control center. It defines the model architecture to be used on each client node and aggregates the results of the individual models through federated averaging. The training happens on each edge node locally so that the data does not leave the device and ensures data integrity. The experiment is repeated with the same dataset that was used in the centralized approach. The train set is shuffled to introduce randomness and divided into 10 equal sized partitions to be fed into 10 VM instances that acted as edge nodes.

#### 5.2 TFF - Model Architecture and Training

The orchestrator is set up to perform model training on each of the client nodes individually with a batch size of 2 and the results of the models are aggregated through a Federated Averaging algorithm to develop a holistic model.

The specifications of the holistic model that is built are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer Type** | **Number of Units** | **Input Shape** | **Activation Function** |
| Input Layer | 20 | (None,10667) | Relu |
| Output Layer | 11 | (None,20) | Softmax |

Table 3: Federated Model Specifications

This model is then used to evaluate the held out test Dataset that resides in the orchestrator.

## 6. Experiment result

The model trained using the centralized approach obtained 100% accuracy within 10 epochs while the federated approach reached the same accuracy within 10 epochs for 10 federated learning rounds. A learning round plays the role of a global epoch which aggregates the results from all edge nodes. Below is a table that summarizes the results.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Training Loss** | **Train Accuracy** | **Test Accuracy** |
| Centralized Model | 0.034 | 100% | 100% |
| Federated Model | 0.004 | 100% | 100% |

Table 4: Comparison of Results

## 7. Conclusion and Future Work

While the centralized model enjoys a slight advantage over the federated model in terms of compute optimization, from a data privacy standpoint, the federated model provides a lot of incentives. Hence, in order to dive deeper into this approach, an architecture is designed as shown below to scale this use-case to automate the pipeline and accommodate more client nodes. The flow diagram below reveals the individual steps in more detail. Important parts of this architecture will include automating the scaling up and down of virtual machines that will serve as edge nodes. Other parts that will need automation are the data upload process, starting up the gRPC connections on each client and finally deploying a clean up mechanism for freeing up memory and stopping the running servers.

#### 7.1 Architecture



Figure 3: TFF Architecture Diagram in GCP

#### 

#### 

#### 

#### 7.2 Flow Diagram

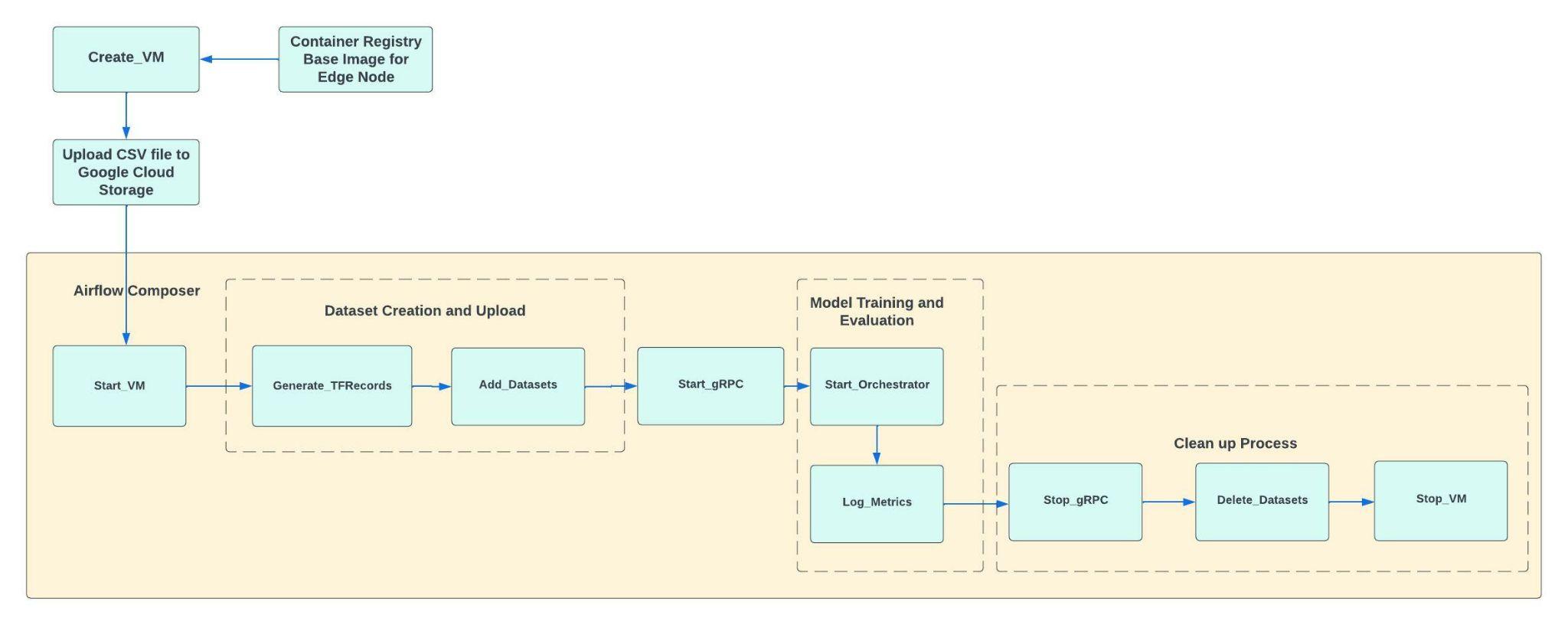


Figure 4: TFF - Flow Diagram

## 8. References

1. Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B., “Communication-Efficient Learning of Deep Networks from Decentralized Data”, <https://arxiv.org/pdf/1602.05629.pdf> , 2016.