電腦網路實驗實驗報告

< 智慧聯網霧端運算-應用聯盟學習於邊緣網路 >

姓名: 翁佳煌　　學號: 409430030

1. 實驗名稱

Federated Learning

2. 實驗目的

本實驗旨在了解並探索聯邦學習（Federated Learning，FL）的概念及其不同類型。實驗的目標如下：
準備輸入數據：從數據集中打印指定用戶及其對應標籤的數據。

探索聯邦學習中的異質性：
a. 印出前一位用戶、當前用戶和下一位用戶的數據分布。
b. 印出前一位用戶、當前用戶和下一位用戶的平均圖像。
c. 根據結果推斷所執行的聯邦學習類型，並提供說明原因。

在聯邦數據上訓練模型：在聯邦數據集上訓練模型，並印出 30 個訓練輪次的結果，包括準確度和損失值。

在 TensorBoard 中顯示模型指標：將訓練結果（包括準確度和損失）在 TensorBoard 中進行可視化。

讓我們深入了解聯邦學習的概念，以及它在保護數據隱私的同時，在大型分散數據集上訓練模型的能力。該實驗還旨在展示聯邦學習中的異質性，通過分析不同類型的聯邦數據集和相應的訓練結果。最後，將模型指標在 TensorBoard 中顯示，以便更好地可視化和分析訓練過程。

3. 實驗設備

Linux 作業系統之電腦。
Google Colab
Tensorflow 套件

4. 實驗步驟

1. Prepare the input Data

a. 印出前 50 組第 X 位用戶的資料，要求: X 為學號後三碼，資料標籤用 tittle 顯示:

實作問題 a，如下圖 1，首先，我們創建了一個名為 example_dataset 的數據集。該數據集是從 emnist_train 數據集中為特定客戶端（用戶）創建的。在這裡，我們選擇了第 30 個用戶（也就這問題 a 要求的學號）的數據集，即 emnist_train.client_ids[30]。

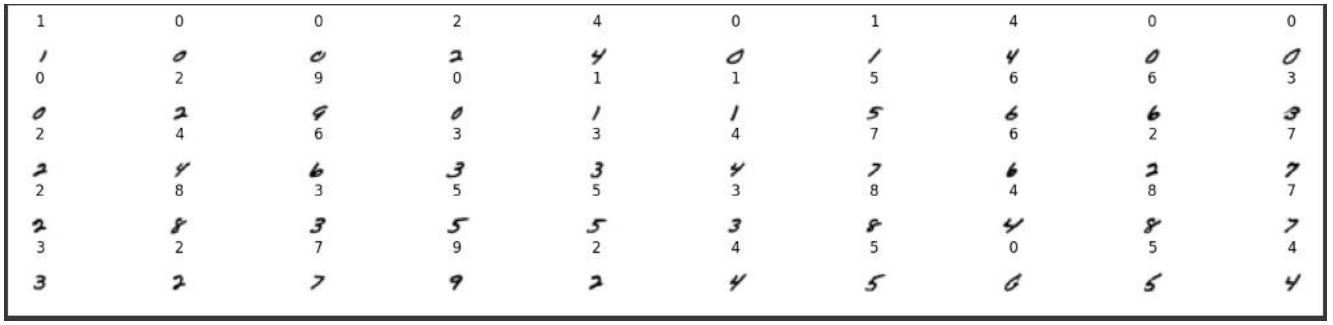
接下來，我們使用 next(iter(example_dataset))獲取 example_dataset 中的下一個元素，也就是該客戶端（用戶）的一個數據樣本。這個數據樣本是一個字典，包含了兩個鍵值對：'pixels'和'label'。

'pixels'鍵對應的值是代表圖像像素的數組。
example_element['pixels'].numpy()將該數組轉換為 NumPy 數組的形式。
'label'鍵對應的值是該圖像的標籤。example_element['label'].numpy()將該標籤轉換為 NumPy 數組的形式。
然後，我們使用 Matplotlib 庫將前 50 個數據樣本顯示出來。通過迭代 example_dataset.take(50)，我們獲取前 50 個數據樣本，並使用 plt.subplot 在一個大圖中建立 50 個小的子圖。對於每個數據樣本，我們使用 plt.imshow 將圖像數據以灰階的形式顯示出來，並使用 plt.title 添加標題，該標題為該數據樣本的標籤。最後，我們使用 plt.axis('off')關閉軸刻度，使圖像更加整潔。

最後執行結果如下圖 2，實現了從數據集中準備輸入數據並印出指定用戶及其對應標籤的功能。

```python
#1.Prepare the input data

example_dataset = emnist_train.create_tf_dataset_for_client(
       emnist_train.client_ids[30])

example_element = next(iter(example_dataset))

example_element['label'].numpy()

# from matplotlib import pyplot as plt
# plt.imshow(example_element['pixels'].numpy(), cmap='gray', aspect='equal')
# plt.grid(False)
# _ = plt.show()


## Example MNIST digits for one client
figure = plt.figure(figsize=(20, 4))
j = 0

for example in example_dataset.take(50):
    plt.subplot(5, 10, j+1)
    plt.imshow(example['pixels'].numpy(), cmap='gray', aspect='equal')
    plt.title(example['label'].numpy()) #add title
    plt.axis('off')
    j += 1


```

▲圖 1



▲圖 2

b. 當執行 emnist_train.element_type_structure 指令時,會產生出下圖的輸出,請解釋這行輸出的意義:

如下圖 3 紅框框為執行結果，當執行 emnist_train.element_type_structure 指令時，會產生一個類型為 OrderedDict 的輸出。該輸出指示了 emnist_train 數據集中每個元素（數據樣本）的結構。

具體來說，emnist_train.element_type_structure 輸出的意義如下：

'label'：這個鍵對應的值指示了數據樣本的標籤。在這個數據集中，標籤是一個沒有特定形狀的整數（tf.int32），即 TensorSpec(shape=(), dtype=tf.int32, name=None)。

'pixels'：這個鍵對應的值指示了數據樣本的像素數據。在這個數據集中，像素數據是一個形狀為(28, 28)的浮點數數組（tf.float32），即 TensorSpec(shape=(28, 28), dtype=tf.float32, name=None)。

簡單來說，emnist_train.element_type_structure 輸出的 OrderedDict 表示了 emnist_train 數據集中每個數據樣本的結構，其中包含了標籤和對應的像素數據。這將在後續的操作中用於讀取和處理數據集中的數據。

```
#environment setup
!pip install --quiet --upgrade tensorflow-federated

%load_ext tensorboard

import collections
import numpy as np
import tensorflow as tf
import tensorflow_federated as tff
from matplotlib import pyplot as plt

np.random.seed(0)

tff.federated_computation(lambda: 'Hello, World!')()

#download dataset
emnist_train, emnist_test = tff.simulation.datasets.emnist.load_data()
emnist_train.element_type_structure


OrderedDict([('label', TensorSpec(shape=(), dtype=tf.int32, name=None)),
             ('pixels',
              TensorSpec(shape=(28, 28), dtype=tf.float32, name=None))])
```

▲圖 3

2. Explore Heterogeneity in FL

要求 2 中要回答:

a. 印出第 X-1, X, X+1 位用戶的資料分布，要求: X 為學號後三碼。

b. 印出第 X-1,X,X+1 位用戶的 Mean Image。

c. 根據結果推論本次實驗是執行何種類型的 FL,並說明原因。

共有這三個問題，首先先看到下圖 4 的 code。

首先，根據指定的學生學號的後三位數字，將其分別命名為 X。然後，從 emnist_train 數據集中選擇第 X-1、X 和 X+1 位用戶，將它們存儲在 clients_to_plot 列表中。

接下來，針對每個用戶，我們創建了對應的客戶數據集 client_dataset，該數據集是從 emnist_train 數據集中為指定用戶創建的。然後，我們創建了一個 defaultdict(list)的 plot_data 字典，用於存儲每個標籤的計數。

在第 a 部分中，我們使用 plt.hist 將每個用戶的標籤計數可視化。對於每個用戶，我們迭代其對應的客戶數據集，並將每個數據樣本的標籤添加到相應的 plot_data 列表中。然後，我們使用 plt.hist 繪製每個標籤的直方圖，以顯示該用戶的標籤分佈情況。這樣可以觀察不同用戶之間的標籤分佈差異。

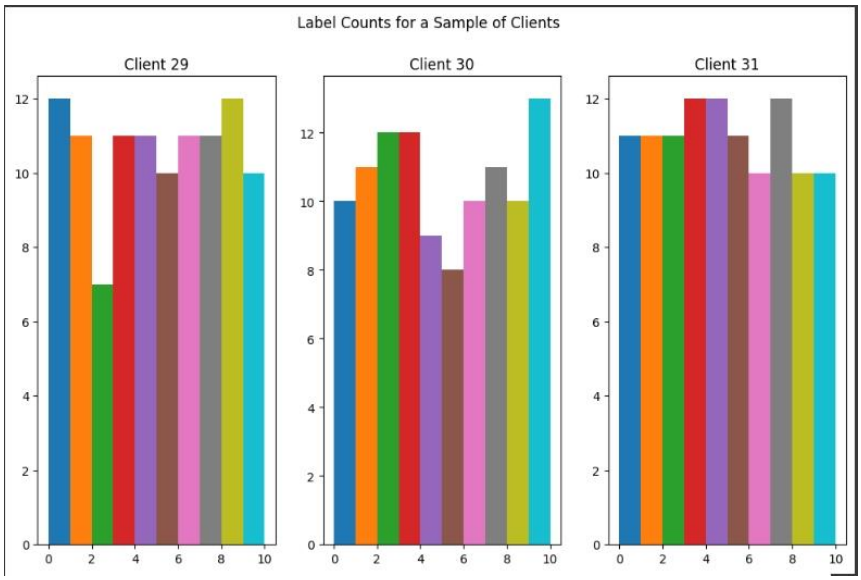
在第 b 部分中，我們計算並繪製每個用戶每個標籤的平均圖像。對於每個用戶，我們迭代其對應的客戶數據集，並將每個標籤的像素數據添加到相應的 plot_data 列表中。然後，我們計算每個標籤的像素數據的平均值，並使用 plt.imshow 將其以圖像的形式顯示出來。這樣可以觀察不同用戶對於每個標籤的平均圖像之間的差異。

下圖 5 和圖 6 為執行結果。

```
#2.Exploring heterogeneity in FL

X = 30   # Replace with the last three digits of your student ID
clients_to_plot = [X-1, X, X+1]

# Number of examples per layer for a sample of clients
f = plt.figure(figsize=(12, 7))
f.suptitle('Label Counts for a Sample of Clients')
for i in range(3):
    client_dataset = emnist_train.create_tf_dataset_for_client(
            emnist_train.client_ids[clients_to_plot[i]]) # use client 29 30 31
    plot_data = collections.defaultdict(list)
    for example in client_dataset:
        # Append counts individually per label to make plots
        # more colorful instead of one color per plot.
        label = example['label'].numpy()
        plot_data[label].append(label)
    plt.subplot(1, 3, i+1)
    plt.title('Client {}'.format(clients_to_plot[i]))
    for j in range(10):
        plt.hist(
                plot_data[j],
                density=False,
                bins=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
plt.show()

# Each client has different mean images, meaning each client will be nudging
# the model in their own directions locally.

for i in range(3):
    client_dataset = emnist_train.create_tf_dataset_for_client(
            emnist_train.client_ids[clients_to_plot[i]]) #here
    plot_data = collections.defaultdict(list)
    for example in client_dataset:
        plot_data[example['label'].numpy()].append(example['pixels'].numpy())
    f = plt.figure(i+1, figsize=(12, 5))
    f.suptitle("Client #{}'s Mean Image Per Label".format(clients_to_plot[i])) #here
    for j in range(10):
        mean_img = np.mean(plot_data[j], 0)
        plt.subplot(2, 5, j+1)
        plt.imshow(mean_img.reshape((28, 28)))
        plt.axis('off')


```

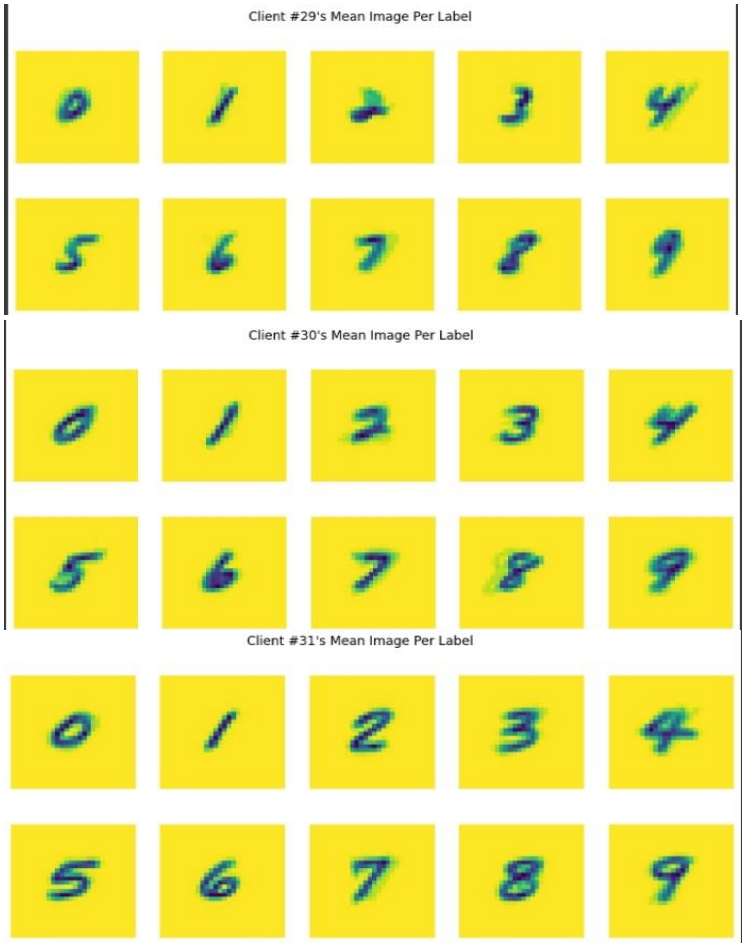
▲圖 4

a. 印出第 X-1, X, X+1 位用戶的資料分布，要求：X 為學號後三碼:



▲圖 5

b. 印出第 X-1,X,X+1 位用戶的 Mean Image:



▲圖 6

c. 根據結果推論本次實驗是執行何種類型的 FL, 並說明原因

根據結果，本次實驗可以推斷是執行的是水平聯邦學習（Horizontal Federated Learning）。

原因如下：
在水平聯邦學習中，用戶之間的特徵重疊較高，而樣本重疊較低。這在結果中得到了體現，因為不同用戶的標籤分佈和平均圖像存在差異，即用戶之間的特徵差異較大。
在水平聯邦學習中，每個用戶擁有相同的特徵空間，但其數據分佈可能不同。由於不同用戶的平均圖像不同，這進一步表明每個用戶的數據分佈存在差異。
綜上所述，根據標籤分佈和平均圖像的差異，以及每個用戶的數據特徵差異，可以推斷本次實驗是執行的水平聯邦學習。

3. Train the Model on Federated Data

要求 3 要回答 a.印出訓練 30 輪的結果(至少顯示出 accurency 以及 loss 的數值）和 b.解釋 client_optimizer_fn 以及 server_optimizer_fn 當中 learning_rate 各自所代表的意義。

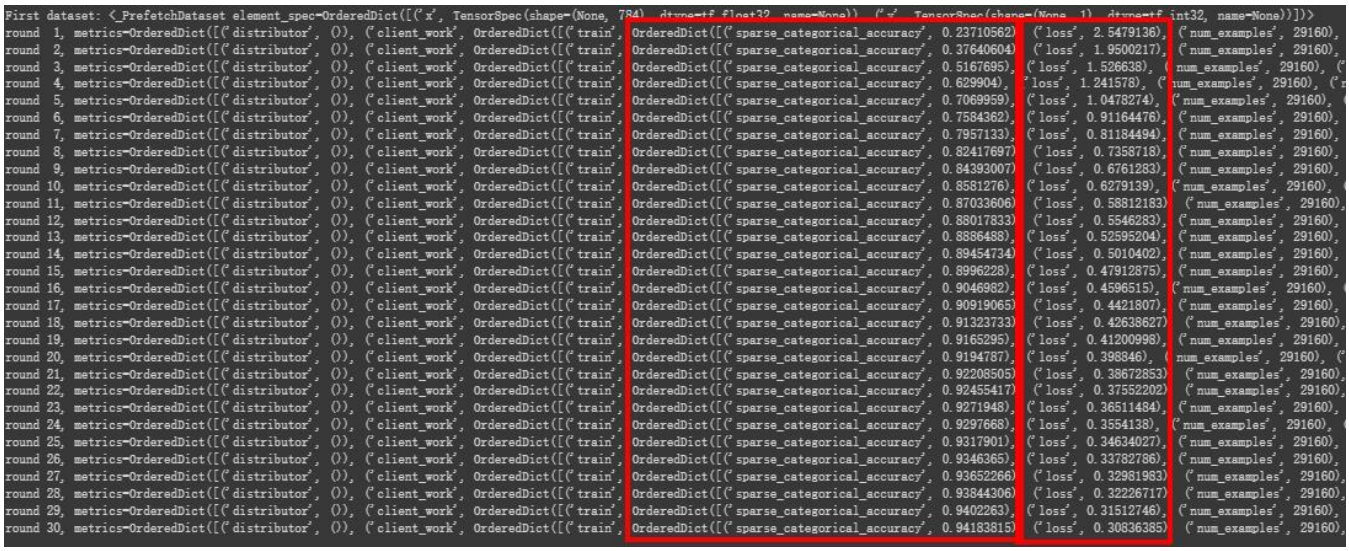
共有這兩個問題，首先先看到下圖 7 的 code。
這段程式碼使用了迴圈來執行 30 輪的訓練，並在每一輪後印出訓練指標的數值。指標包括 accuracy（準確率）和 loss（損失），執行結果如下圖 8。

```
#Process the input data
NUM_CLIENTS = 10   #need to change?
NUM_EPOCHS = 30
BATCH_SIZE = 20
SHUFFLE_BUFFER = 100
PREFETCH_BUFFER = 10

def preprocess(dataset):

    def batch_format_fn(element):
        """Flatten a batch `pixels` and return the features as an `OrderedDict`."""
        return collections.OrderedDict(
            x=tf.reshape(element['pixels'], [-1, 784]),
            y=tf.reshape(element['label'], [-1, 1]))

    return dataset.repeat(NUM_EPOCHS).shuffle(SHUFFLE_BUFFER, seed=1).batch(
        BATCH_SIZE).map(batch_format_fn).prefetch(PREFETCH_BUFFER)

preprocessed_example_dataset = preprocess(example_dataset)

sample_batch = tf.nest.map_structure(lambda x: x.numpy(),
                                     next(iter(preprocessed_example_dataset)))
sample_batch

#Process the input data - Choose Clients
def make_federated_data(client_data, client_ids):
    return [
        preprocess(client_data.create_tf_dataset_for_client(x))
        for x in client_ids
    ]

sample_clients = emnist_train.client_ids[0:NUM_CLIENTS]
federated_train_data = make_federated_data(emnist_train, sample_clients)
print('Number of client datasets: {l}'.format(l=len(federated_train_data)))
print('First dataset: {d}'.format(d=federated_train_data[0]))
```

```
#Create a Model with Keras
def create_keras_model():
    return tf.keras.models.Sequential([
            tf.keras.layers.InputLayer(input_shape=(784,)),
            tf.keras.layers.Dense(10, kernel_initializer='zeros'),
            tf.keras.layers.Softmax(),
    ])

def model_fn():
    # We _must_ create a new model here, and _not_ capture it from an external
    # scope. TFF will call this within different graph contexts.
    keras_model = create_keras_model()
    return tff.learning.models.from_keras_model(
            keras_model,
            input_spec=preprocessed_example_dataset.element_spec,
            loss=tf.keras.losses.SparseCategoricalCrossentropy(),
            metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])

#3.Training the model on federated data
training_process = tff.learning.algorithms.build_weighted_fed_avg(
        model_fn,
        client_optimizer_fn=lambda: tf.keras.optimizers.SGD(learning_rate=0.02),
        server_optimizer_fn=lambda: tf.keras.optimizers.SGD(learning_rate=1.0))
train_state = training_process.initialize()

#  result = training_process.next(train_state, federated_train_data)
#  train_state = result.state
#  train_metrics = result.metrics
#print('round  1, metrics={}'.format(train_metrics))

NUM_ROUNDS = 30
for round_num in range(1, NUM_ROUNDS+1):
    result = training_process.next(train_state, federated_train_data)
    train_state = result.state
    train_metrics = result.metrics
    print('round {:2d}, metrics={}'.format(round_num, train_metrics))
```

▲圖 7

a. 印出訓練 30 輪的結果(至少顯示出 accurency 以及 loss 的數值):



▲圖 8

b. 解釋 client_optimizer_fn 以及 server_optimizer_fn 當中 learning_rate 各自所代表的意義:

在詳細解釋 client_optimizer_fn 和 server_optimizer_fn 中的 learning_rate 參數之前，我們先了解一下這兩個函數的作用。

client_optimizer_fn: 這個函數用於指定在客戶端進行參數更新時使用的優化器。優化器是一個算法，用於根據模型的損失函數和參數的梯度來調整模型的參數值，以使損失最小化。在每一輪訓練中，客戶端會根據本地數據計算梯度並使用該梯度來更新模型的參數。client_optimizer_fn 函數的目的是返回所需的客戶端優化器。

server_optimizer_fn: 這個函數用於指定在服務器端進行全局模型參數聚合時使用的優化器。在聯邦學習中，服務器端負責聚合來自多個客戶端的參數更新，以獲得全局模型的最新參數。server_optimizer_fn 函數的目的是返回所需的服務器優化器。

現在來理解 learning_rate 參數的意義：
learning_rate: 學習率是優化算法中的一個重要參數，用於控制每次參數更新的步幅大小。它決定了模型在每一次迭代中對於梯度的利用程度，進而影響模型的收斂速度和結果品質。

在客戶端，由於每個客戶端僅擁有本地數據的子集，並且根據本地數據計算梯度並更新參數，因此 client_optimizer_fn 中的 learning_rate 控制著每個客戶端本地模型的參數更新速度。較大的學習率將導致本地模型參數更快地改變，而較小的學習率則會使得本地模型參數改變較為緩慢。

在服務器端，全局模型的參數是由多個客戶端的參數更新聚合而成的。server_optimizer_fn 中的 learning_rate 控制著服務器端對於不同客戶端參數更新的整合速度。較大的學習率將使服務器端更快地融合各個客戶端的參數更新，而較小的學習率將使整合過程更為緩慢。

總而言之，learning_rate 它分別控制著本地模型參數和全局模型參數的更新速度。因此，選擇合適的學習率對於訓練模型的性能和收斂速度是非常重要的。

4. Display Model Metrics in TensorBoard

要求 4 只需回答一個問題 a. 將上個步驟所訓練 30 輪的結果顯示在 TensorBoard 上(請至少貼出 accurency 以及 loss 的結果)。

首先先看到下圖 9 的 code，它用於在 TensorBoard 中顯示模型的指標(metrics)。首先，它定義了一個目錄 logdir，用於存儲 TensorBoard 的日誌文件。然後，它初始化了一個 summary_writer，該寫入器將日誌事件寫入指定的目錄。

在訓練迴圈中，每一個訓練輪次，該代碼調用 training_process.next()方法進行訓練，並獲取訓練指標(train_metrics)。接下來，它使用 tf.summary.scalar()方法將訓練指標的值寫入 TensorBoard 的日誌文件中，並使用當前的迴圈編號(round_num)作為步數。

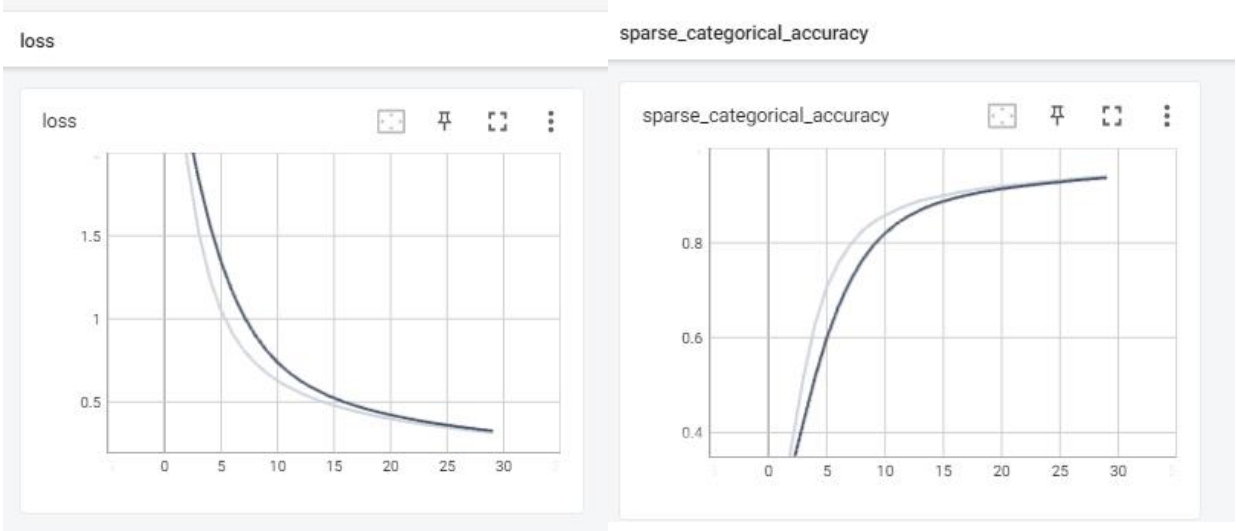
最後，它使用!ls 命令列出日誌目錄的內容，確保日誌文件已經被創建。然後，它使用%tensorboard 命令啟動 TensorBoard，將指定的日誌目錄 logdir 作為參數傳遞給 TensorBoard。--port=0 指定讓 TensorBoard 自動選擇一個可用的端口。

這段程式碼的作用是將訓練過程中獲得的訓練指標(metrics)保存到 TensorBoard 的日誌文件中，然後通過 TensorBoard 在網頁瀏覽器中顯示這些指標的變化情況。你可以通過訪問指定的 TensorBoard 服務器來查看這些指標的視覺化結果(如下圖 10)。

```
#4.Displaying model metrics in TensorBoard

logdir = "/tmp/logs/scalars/training/"
try:
    tf.io.gfile.rmtree(logdir)    # delete any previous results
except tf.errors.NotFoundError as e:
    pass # Ignore if the directory didn't previously exist.
summary_writer = tf.summary.create_file_writer(logdir)
train_state = training_process.initialize()

with summary_writer.as_default():
    for round_num in range(1, NUM_ROUNDS):
        result = training_process.next(train_state, federated_train_data)
        train_state = result.state
        train_metrics = result.metrics
        for name, value in train_metrics['client_work']['train'].items():
            tf.summary.scalar(name, value, step=round_num)

!ls {logdir}
%tensorboard --logdir {logdir} --port=0

```

▲圖 9

a. 將上個步驟所訓練 30 輪的結果顯示在 TensorBoard 上(請至少貼出 accurency 以及 loss 的結果)



▲圖 10

5. 問題與討論

Horizontal Federated Learning、Vertical Federated Learning、Federated Transfer Learning之間的差異:

Horizontal Federated Learning (HFL):

HFL是指在FL中參與訓練的不同參與者（客戶端）擁有相同的特徵空間，但擁有不同的樣本數據。換句話說，不同參與者的數據屬性相同，但樣本分布可能有所不同。在HFL中，客戶端彼此共享模型的架構，但使用自己的本地數據進行訓練。這種方法通常用於數據分散且不易共享的情況，例如不同設備上的用戶數據。

Vertical Federated Learning (VFL):

VFL是指在FL中參與訓練的不同參與者（客戶端）具有不同的特徵空間，但共享相同的樣本數據。換句話說，不同參與者的數據屬性不同，可能包含不同的特徵，但樣本來自相同的數據集。在VFL中，參與者需要將本地特徵進行加密或處理，以保護數據隱私，然後共享加密後的特徵以進行聯合訓練。這種方法通常用於涉及數據所有者之間共享的情況，例如不同機構之間的合作訓練。

Federated Transfer Learning (FTL):

FTL是指在FL中參與訓練的參與者（客戶端）擁有不同的特徵空間和樣本數據。與HFL和VFL不同，FTL的目標是實現模型的知識轉移。在FTL中，模型在一個參與者上訓練後，可以將該模型的部分或全部權重傳遞給其他參與者，以作為他們本地訓練的起點。這種方法通常用於解決數據稀缺的問題，其中某些參與者具有豐富的數據，而其他參與者數據有限。

6. 心得與感想

這次的實驗讓我深入了解了聯邦學習的異質性以及相關概念和技術。我明白到在聯邦學習中，參與者的數據分佈和特徵可能存在差異，這需要我們針對異質性進行處理和調整。同時，數據隱私和安全性在聯邦學習中扮演著關鍵的角色，我需要確保數據保持在本地並進行本地模型訓練，只將參數的更新聚合到中央服務器上。在訓練過程中，客戶端和服務器端的優化器的選擇和設定對於模型的性能和收斂速度具有重要影響。最後，通過使用 TensorBoard 進行可視化，我能夠更好地了解模型在每輪訓練中的表現和趨勢，並進行相應的調整和優化。總體而言，這次實驗為我提供了寶貴的學習機會，使我對聯邦學習的工作流程和挑戰有了更深入的理解。

7. 參考文獻

https://medium.com/sherry-ai/%E8%81%AF%E7%9B%9F%E5%BC%8F%E5%AD%B8%E7%BF%92-federated-learning-b4cc5af7a9c0

https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification

https://zh.wikipedia.org/zh-tw/Softmax%E5%87%BD%E6%95%B0
https://medium.com/%E6%89%8B%E5%AF%AB%E7%AD%86%E8%A8%98/%E4%BD%BF%E7%94%A8-tensorflow-%E5%AD%B8%E7%BF%92-softmax-%E5%9B%9E%E6%AD%B8-softmax-regression-41a12b619f04