**Step-by-Step: Install TensorFlow in a Virtual Environment on Raspberry Pi 4**

**📦 Step 1: Update system & install dependencies**

sudo apt update && sudo apt upgrade -y

sudo apt install -y python3-venv python3-pip libatlas-base-dev

**🧪 Step 2: Create and activate a virtual environment**

python3 -m venv tf-env

source tf-env/bin/activate

You should now see (tf-env) in your terminal prompt.

**🚀 Step 3A: Install TensorFlow Lite Runtime (for inference)**

pip install --upgrade pip

pip install tflite-runtime

Check it's working:

python -c "import tflite\_runtime.interpreter as tflite; print(tflite.\_\_version\_\_)"

## ✅ Step-by-Step: Federated Learning on Raspberry Pi in a Virtual Environment

### 🔧 1. ****Install**** venv ****if you don’t already have it****

sudo apt update

sudo apt install python3-venv

### 🧪 2. ****Create and activate the virtual environment****

python3 -m venv fl-env

source fl-env/bin/activate

Now your shell should show something like this:

(fl-env) pi@raspberrypi:~ $

### 📦 3. ****Upgrade pip and install required packages****

Inside the virtual environment:

pip install --upgrade pip setuptools wheel

Now install TensorFlow (Lite or full version depending on your Pi’s memory):

# For TensorFlow Lite (preferred on Raspberry Pi for performance)

pip install tflite-runtime

# OR (for full TensorFlow, slow but possible on Pi 4 with enough RAM)

**pip install tensorflow==2.13**

Optionally, install other helpful libraries:

pip install numpy flask requests

❗ **Note:** Do not install tensorflow\_federated directly on the Pi. It's not ARM-compatible. Instead, run your **TFF server** on a PC or cloud, and just use the Pi as a **federated client**.

### 🛠️ 4. ****Write your local client training script****

Inside fl-env, create a script, e.g., client\_train.py:

python

import tensorflow as tf

import numpy as np

import requests

import json

# Dummy data

X = np.random.rand(100, 5)

y = np.random.randint(0, 2, 100)

# Simple model

model = tf.keras.Sequential([

tf.keras.layers.Dense(8, activation='relu', input\_shape=(5,)),

tf.keras.layers.Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy')

# Train locally

model.fit(X, y, epochs=3)

# Serialize weights

weights = [w.tolist() for w in model.get\_weights()]

# Send to central server

res = requests.post("http://192.168.137.177:5000/upload\_weights", json={'weights': weights})

print("Server response:", res.text)

### 🌐 5. ****Create a Flask server on your PC or cloud (server-side)****

python

# server.py

from flask import Flask, request

import numpy as np

app = Flask(\_\_name\_\_)

client\_weights = []

@app.route("/upload\_weights", methods=["POST"])

def upload\_weights():

data = request.get\_json()

weights = data["weights"]

client\_weights.append(weights)

print(f"Received weights from client. Total clients: {len(client\_weights)}")

return "Weights received", 200

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", port=5000)

### 🚀 6. ****Run everything****

* On Raspberry Pi:

source fl-env/bin/activate

python client\_train.py

* On your PC/server:

python server.py

Now the Pi will train the model locally and send weights to the server! 🎉

**Install flask if not installed**

pip install flask

### 🔄 Optional: Send updated weights back to the Pi

You can extend the Flask server to respond with averaged model weights after multiple clients have sent theirs.

Would you like me to:

* Help you with the server-side aggregation logic?
* Add code to send updated weights back to the Pi?
* Set up multiple Pis/clients?

Let me know how deep you want to go!

4o

##### You said:

how to check tensorflow has been installed or not

###### ChatGPT said:

Great question! To check whether **TensorFlow** has been installed in your **virtual environment** (or system-wide), just follow these steps:

### ✅ Inside your virtual environment

Make sure you're **in the virtual environment** first:

source fl-env/bin/activate

### 🔍 Then run this command:

python -c "import tensorflow as tf; print(tf.\_\_version\_\_)"

If TensorFlow is installed, it will print the version like:

2.13.0

If it's **not installed**, you’ll get an error like:

ModuleNotFoundError: No module named 'tensorflow'

### 🧪 Bonus: Check pip list

You can also check all installed packages with:

pip list

Look for tensorflow or tflite-runtime in the output.

Let me know what it shows—if it’s not installed, I can help you troubleshoot or install the right version.