Topik : 2.1 Federated Learning for Image Data

Objective : Memahami cara melatih model klasifikasi gambar secara federated

Task : Jalankan Tutorial FL dengan dataset EMNIST

Source : <https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification?hl=id>

Kita menggunakan pelatihan EMNIST klasik untuk memperkenalkan Federasi Learning (FL) lapisan API dari TFF. Tff.learning merupakan satu set antarmuka Tingkat yang lebih tinggi yang dapat digunakan untuk melakukan jenis umum dari tugas – tugas belajar federasi, seperti pelatihan gabungan , terhadap model yang disediakan pengguna yang diimplementasikan di TensorFlow.

Federated Learning API , ditujukan terutama untuk pengguna yang ingin menyambungkan model TensorFlow mereka sendiri ke TFF, memperlakukan yang terakhir Sebagian besar sebagai kotak hitam.

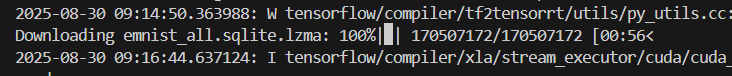
**Preparing the input Data**

Federated learning memerlukan Kumpulan data gabungan yaitu Kumpulan data dari banyak pengguna. Data federasi biasanya non iid, yang menimbulkan serangkaian tantangan yang unik.

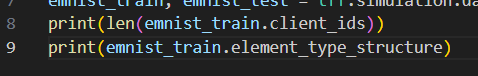
Disini kita menggunakan repositori TFF dengan beberapa dataset, termasuk versi federated dari MINST yang berisi versi dataset asli NIST yang telah di proses ulang menggunakan leaf sehingga data tersebut diindeks berdasarkan penulis angka – angka tersebut.

Berikut cara membuat nya :

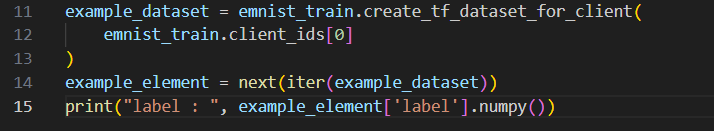


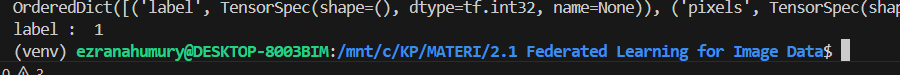


Dataset dikembalikan oleh load\_data() adalah contoh dari tff.simulation.ClienData , sebuah antarmuka yang memungkinkan kita untuk menghitung set pengguna, untuk membangun sebuah tf.data.Dataset yang mewakili data pengguna tertentu, dan untuk query struktur elemen individu. Identitas klien tidak digunakan oleh kerangka pembelajaran federasi yang merupakan satu – satunya tujuan mereka Adalah untuk memungkinkan kita memilih subset data untuk melakukan simulasi.

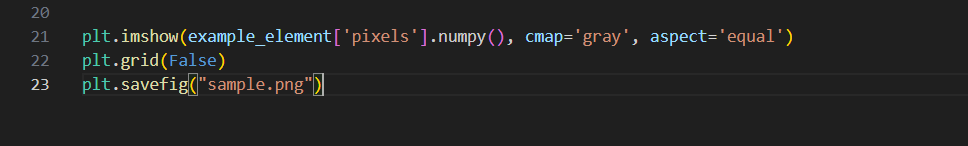


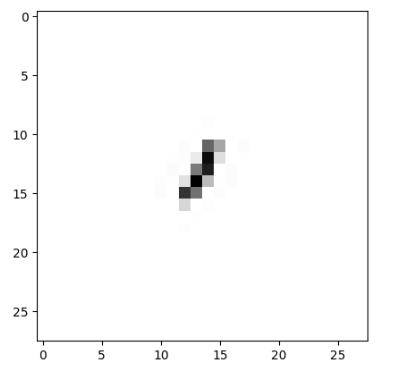






Plt.show nya diubah menjadi plt.savefig. karena kita sedang menjalankan script python di terminal CLI dalam WSL/Linux. bukan di lingkungan yang punya GUI/X-server (misalnya Jupyter Notebook atau VS Code interactive).

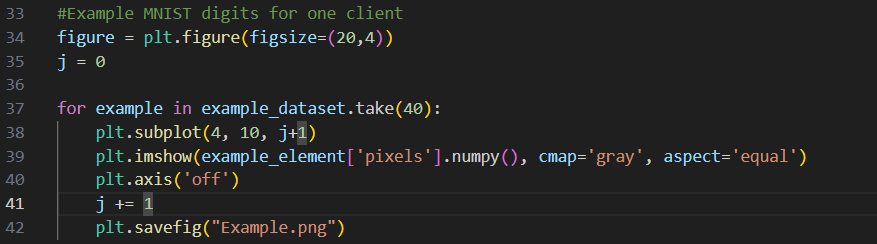




**Exploring heterogeneity in federated data**

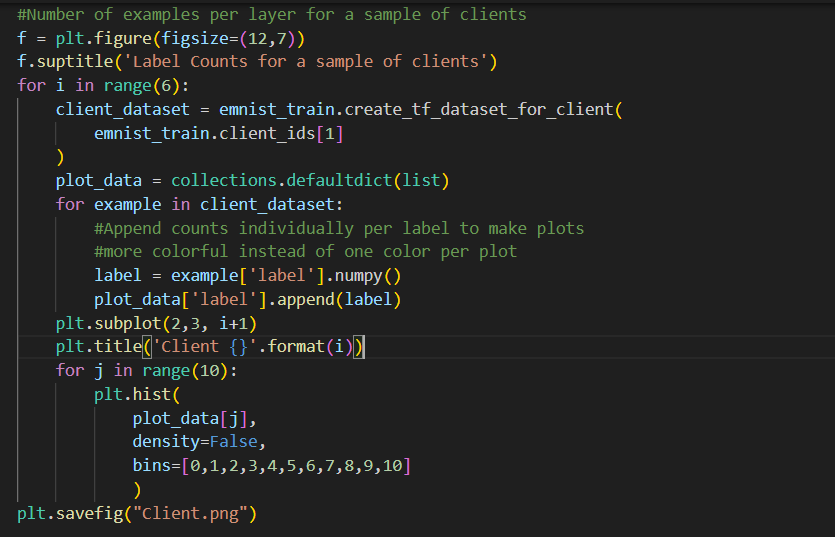
Data federasi biasanya non iid, pengguna biasanya memiliki distribusi data yang berbeda tergantung pada pola penggunaan. Beberapa klien mungkin memilih jumlah contoh pelatihan lebih sedikit dari perangkat, mengalami kekurangan data secara local, sementara klien lain mungkin memilih lebih dari cukup contoh pelatihan. Mari kita jelajahi konsep heterogenitas data yang khas pada sistem federasi menggunakan dataset EMNIST yang tersedia.

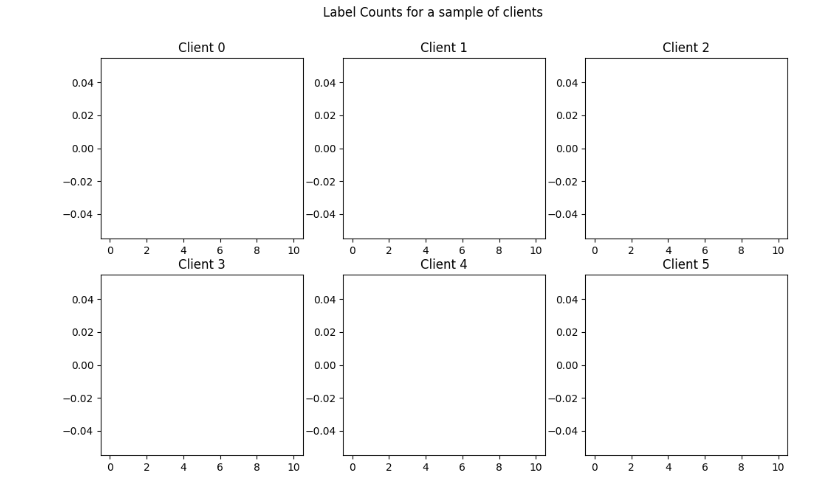
Pertama, mari kita ambil sample data dari satu clien untuk mendapatkan Gambaran tentang contoh – contoh pada satu perangkat simulasi. Karena dataset yang digunakan diindeks berdasarkann penulis unik, data dari satu klien mewakili tulisan tangan satu orang untuk sampel angka 0 – 9, mensimulasikan pola penggunaan unik satu pengguna.



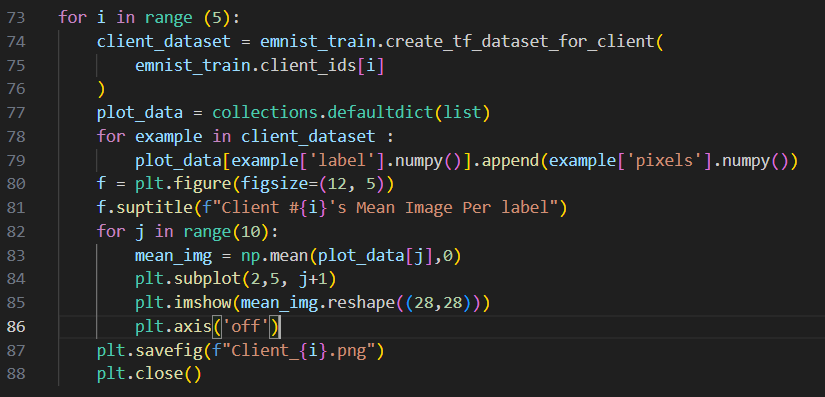


Sekarang, mari kita visualisasikan jumlah contoh pada setiap klien untuk setiap label digit MNIST. Dalam lingkungan federasi, jumlah contoh pada setiap klien dapat sedikit berbeda, tergantung pada perilaku pengguna.



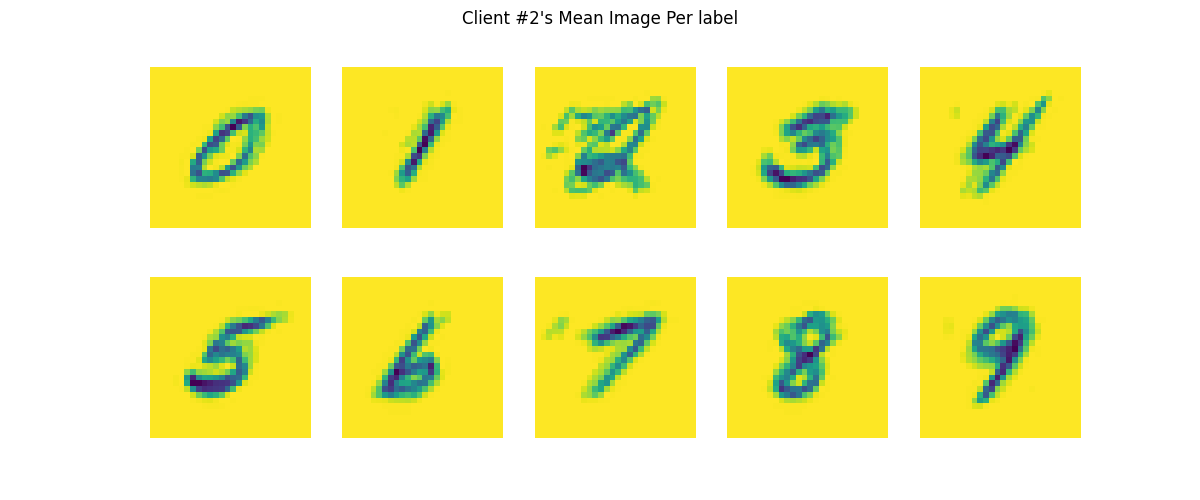


Sekarang mari kita visualisasikan gambar rata – rata per klien untuk setiap label MNIST, kode ini akan menghasilkan rata – rata setiap nilai piksel untuk semua contoh pengguna untuk satu label. Kita akan melihat bahwa gambar rata – rata satu klien untuk satu digit akan terlihat berbeda dari gambar rata – rata klien lain untuk angka yang sama, karena gaya tulisan tangan yang unik dari setiap orang. Kita dapat merenungkan tentang bagaimana setiap putaran pelatihan lokal akan mendorong model ke arah yang berbeda pada setiap klien, karena kita belajar dari data unik pengguna itu sendiri di putaran lokal tersebut









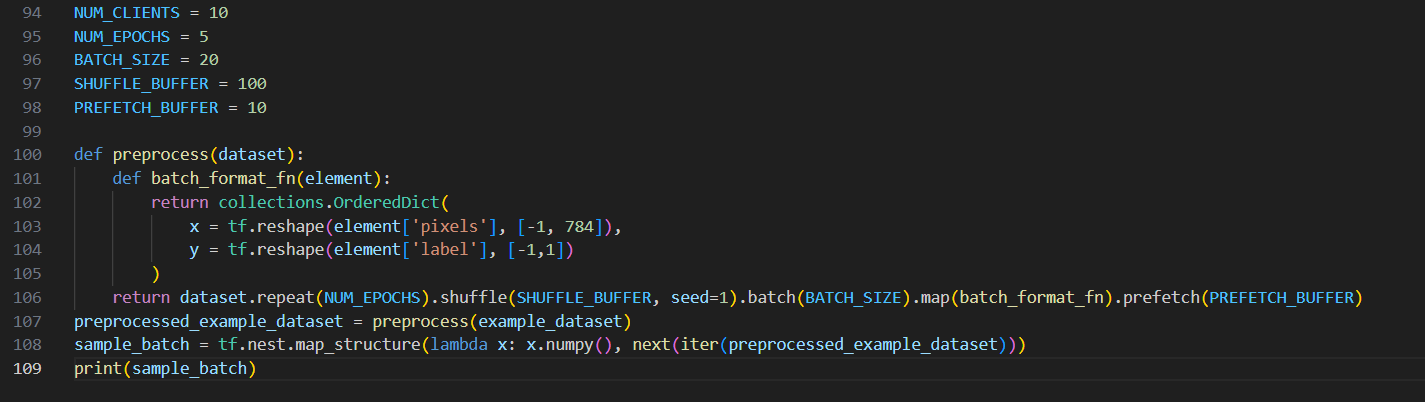


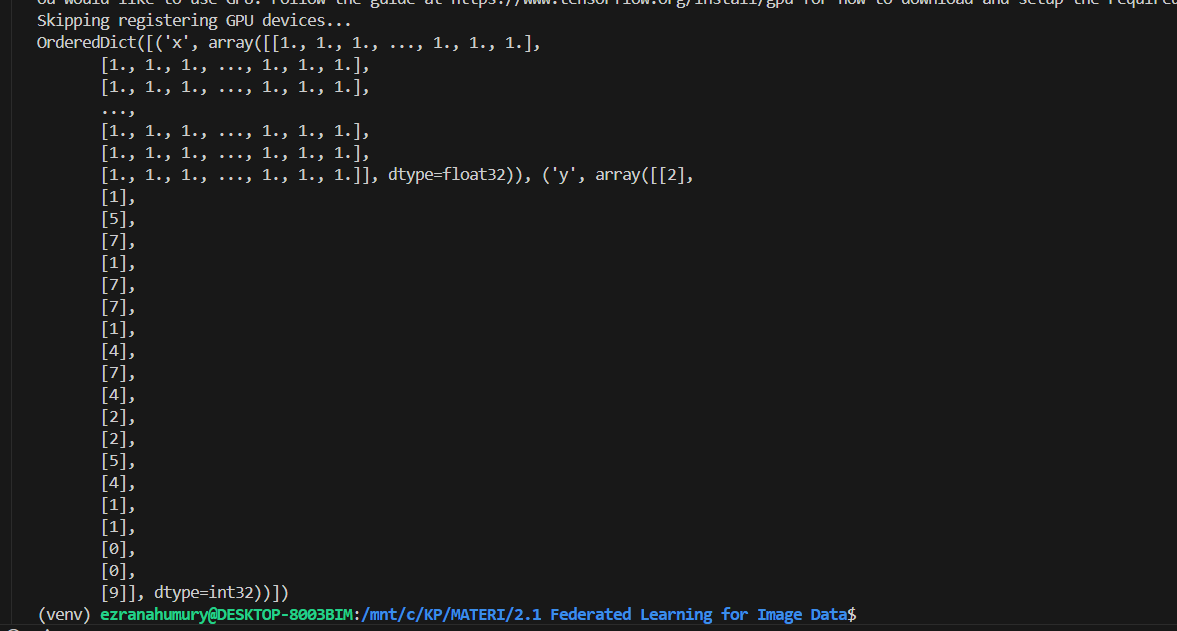


Data pengguna dapat bersifat berisik dan diberi label secara tidak akurat. Misalnya, jika kita melihat data Klien #2 di atas, kita dapat melihat bahwa untuk label 2, kemungkinan ada beberapa contoh yang diberi label salah, yang menyebabkan gambar rata-rata menjadi lebih berisik.

**Prepocessing the input data**

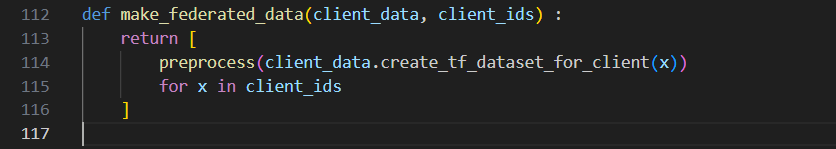
Karena data sudah berupa tf.data.Dataset, PreProcessing dapat dilakukan menggunakan transformasi dataset. Disini, kami meratakan gambar 28x28 menjadi array yang berelemen 784, mengacak contoh – contoh individu, mengorganisirnya ke dalam batch, dan mengganti nama fitur dari pixel dan label ke x dan y untuk digunakan dengan keras. Kami juga menambahkan repeat atas dataset untuk menjalankan beberapa epoch.





Salah satu cara untuk memasukkan data federated ke TFF dalam simulasi Adalah dengan menggunakan daftar python, Dimana setiap elemen daftar tersebut menyimpan data pengguna individu. Baik sebagai daftar maupun sebagai tf.data.Dataset. karena kita sudah memiliki antarmuka yang menyediakan opsi terakhir.

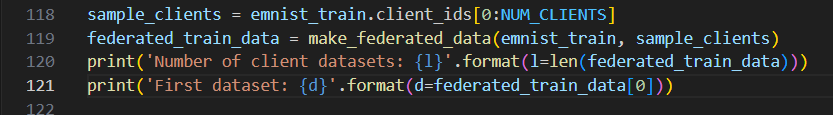
Berikut Adalah fungsi bantu sederhana yang akan membangun daftar dataset dari Kumpulan pengguna yang diberikan sebagai input untuk putaran pelatihan dan evaluasi :

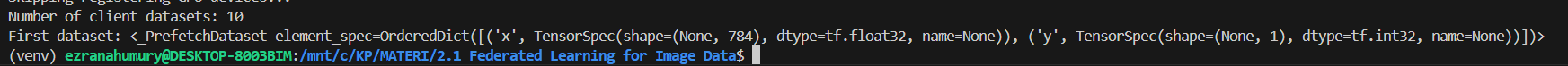


Dalam scenario pelatihan federasi yang tipikal, kita berurusan dengan populasi perangkat pengguna yang sangat besar, Dimana hanya Sebagian kecil dari perangkat tersebut yang mungkin tersedia untuk pelatihan pada suatu waktu tertentu. Hal ini terjadi, misalnya Ketika perangkat klien Adalah ponsel pintar yang hanya berpartisipasi dalam pelatihan saat terhubung ke sumber daya Listrik, terhubung ke jaringan yang diukur, dan dalam keaadaan idle.

Secara umum, saat menjalankan simulasi, kita akan mengambil sample subset acak dari klien yang akan terlibat dalam setiap putaran pelatihan, umumnya berbeda di setiap putaran. Mencapai konvergensi dalam system dengan subset klien yang diambil secara acak di setiap putaran dapat memakan waktu yang cukup lama, dan tidak praktis untuk menjalankan putaran.

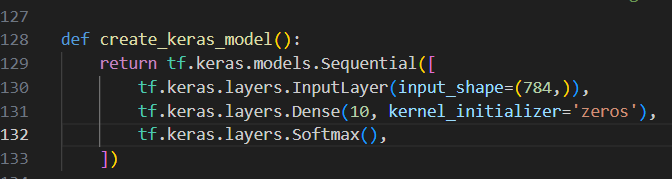
Kita akan melakukan pengambilan sample dari Kumpulan client sekali saja , dan menggunakan Kumpulan yang sama di seluruh putaran untuk mempercepat konvergensi ( secara sengaja melakukan overfitting pada data dari sedikit pengguna).



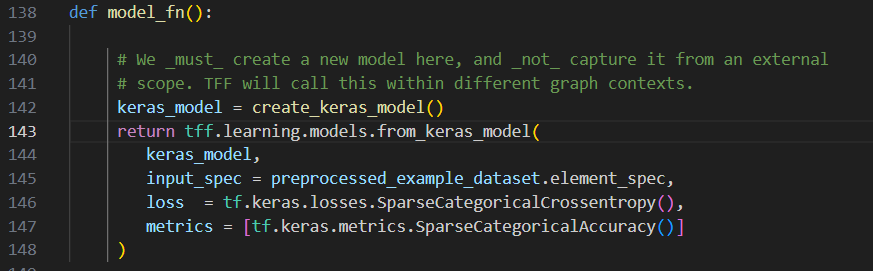


**Creating a model with Keras**

Berikut Adalah contoh sederhana yang membangun model Keras :



Untuk membungkus model apapun dengan TFF, model tersebut harus dibungkus dalam instance antarmuka tff,learning.models.VariableModel, yang menyediakan metode untuk menandai proses forward pass model, property metadata dan sebagainya yang mirip dengan keras. tetapi juga memperkenalkan elemen tambahan, seperti cara mengontrol proses perhitungan metrik federasi.

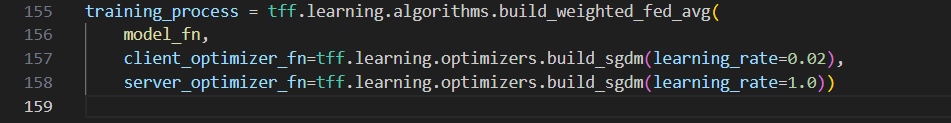


**Training the model on federated data**

Catatan penting tentang algoritma Federated Averaging yaitu terdapat 2 optimizer :

1. \_clientoptimizer ( hanya digunakan untuk menghitung pembaruan model local di setiap klien)
2. \_serveroptimizer (menerapkan pembaruan rata – rata ke model global di server )

Secara khusus, ini berarti pilihan optimizer dan laju pembelajaran yang digunakan mungkin perlu berbeeda dari yang kita gunakan untuk melatih model pada dataset i.i.d standar.

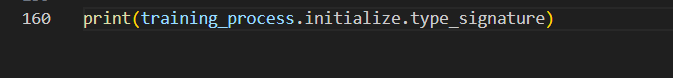


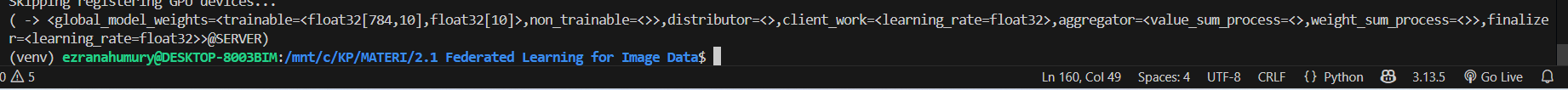
TFF telah membangun sepasang *perhitungan federasi* dan dikemas ke dalam sebuah tff.templates.IterativeProcess Dimana perhitungan ini tersedia sebagai sepasang sifat initialize dan next.

Singkatnya, *perhitungan federasi* adalah program di bahasa internal TFF yang dapat mengekspresikan berbagai algoritma federasi. Dalam hal ini, dua perhitungan yang dihasilkan dan dikemas ke dalam iterative\_process menerapkan Federasi Averaging.

Tujuan TFF untuk mendefinisikan perhitungan dengan cara yang mereka dapat dieksekusi dalam pengaturan pembelajaran federasi nyata, tetapi saat ini hanya runtime simulasi eksekusi lokal yang diterapkan. Untuk menjalankan komputasi dalam simulator, Anda cukup memanggilnya seperti fungsi Python. Lingkungan interpretasi default ini tidak dirancang untuk kinerja tinggi, tetapi cukup untuk tutorial ini; kami berharap dapat menyediakan runtime simulasi berkinerja lebih tinggi untuk memfasilitasi penelitian skala besar di rilis mendatang.

Mari kita mulai dengan initialize perhitungan. Seperti halnya untuk semua komputasi gabungan, Anda dapat menganggapnya sebagai fungsi. Komputasi tidak memerlukan argumen, dan mengembalikan satu hasil - representasi status proses Rata-Rata Federasi di server. Meskipun kami tidak ingin menyelami detail TFF, mungkin bermanfaat untuk melihat seperti apa keadaan ini. Anda dapat memvisualisasikannya sebagai berikut.





Tipe tanda tangan di atas mungkin pada awalnya tampak samar sedikit, Anda dapat mengenali bahwa server negara terdiri dari model (model parameter awal untuk MNIST yang akan didistribusikan ke semua perangkat), dan optimizer\_state (informasi tambahan dikelola oleh server, seperti jumlah putaran yang digunakan untuk jadwal hyperparameter, dll.).

Mari kita memanggil initialize perhitungan untuk membangun server negara.



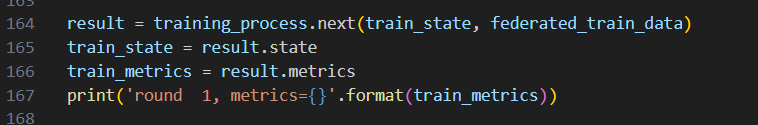
Kedua dari pasangan perhitungan federasi, next , merupakan satu putaran Federasi Averaging, yang terdiri dari mendorong negara Server (termasuk parameter model) kepada klien, pada perangkat pelatihan data lokal mereka, mengumpulkan dan update Model averaging , dan menghasilkan model baru yang diperbarui di server.

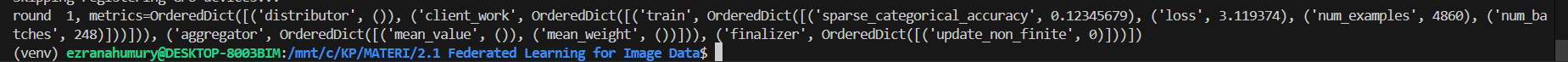
Secara konseptual, Anda bisa memikirkan next sebagai memiliki tanda tangan jenis fungsional yang terlihat sebagai berikut.



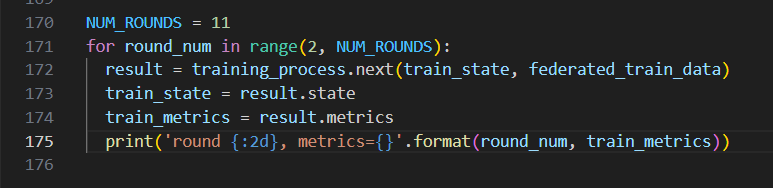
Salah satu harus berpikir tentang next() tidak sebagai fungsi yang berjalan di server, melainkan menjadi representasi fungsional deklaratif dari seluruh perhitungan desentralisasi - beberapa input yang disediakan oleh server ( SERVER\_STATE ), tetapi masing-masing peserta perangkat menyumbangkan set data lokalnya sendiri.

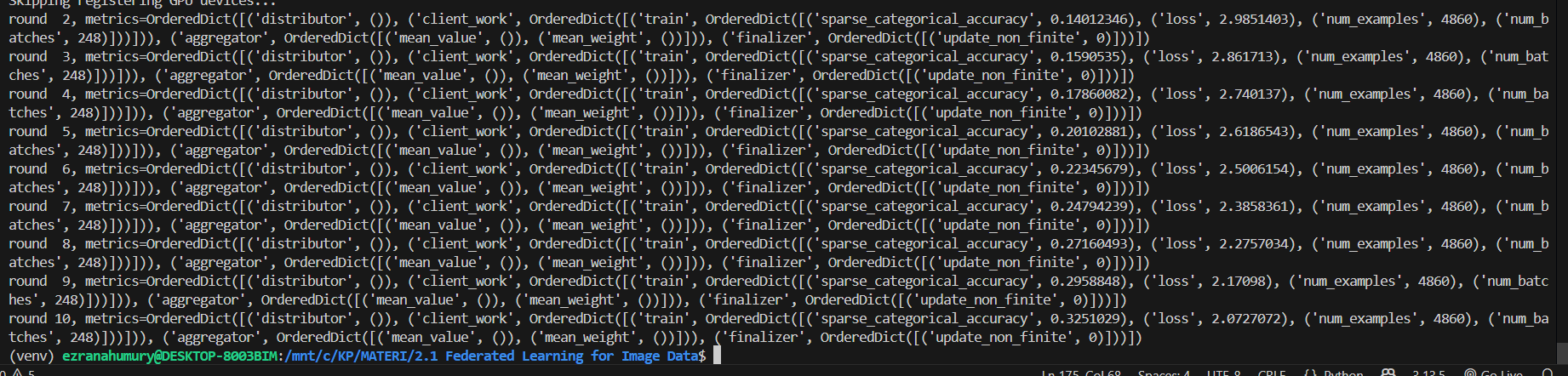
Mari kita jalankan satu putaran pelatihan dan visualisasikan hasilnya. Kami dapat menggunakan data gabungan yang telah kami buat di atas untuk sampel pengguna.





Jalankan beberapa putaran lagi. Seperti disebutkan sebelumnya, biasanya pada titik ini anda akan memilih subset data simulasi anda dari sample pengguna baru yang dipilih secara acak untuk setiap putaran untuk mensimulasikan penerapan realistis Dimana pengguna terus datang dan pergi.

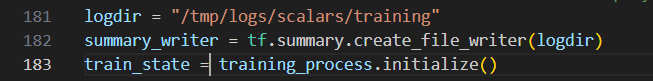




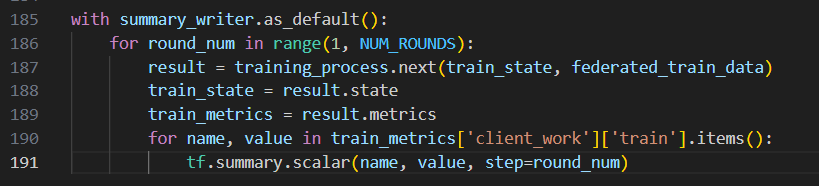
Kehilangan pelatihan menurun setelah setiap putaran pelatihan gabungan, menunjukkan model konvergen. Ada beberapa keberatan penting dengan metrik pelatihan ini, bagaimanapun, lihat bagian *Evaluasi* nanti dalam tutorial ini.

**Displaying model metrics in TensorBoard**

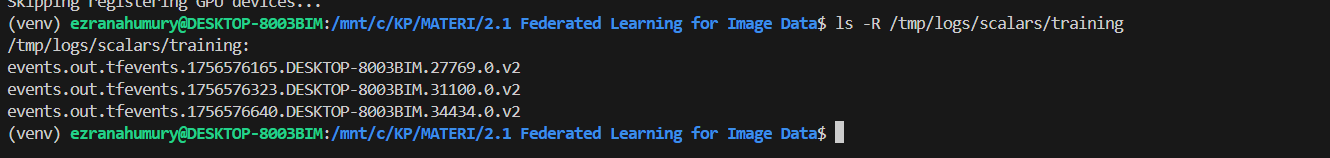
Visualisasikan metrik dari komputasi gabungan menggunakan TensorBoard, dimulai dengan membuat direktori dan penulis ringkasan yang sesuai untuk menulis metrics :

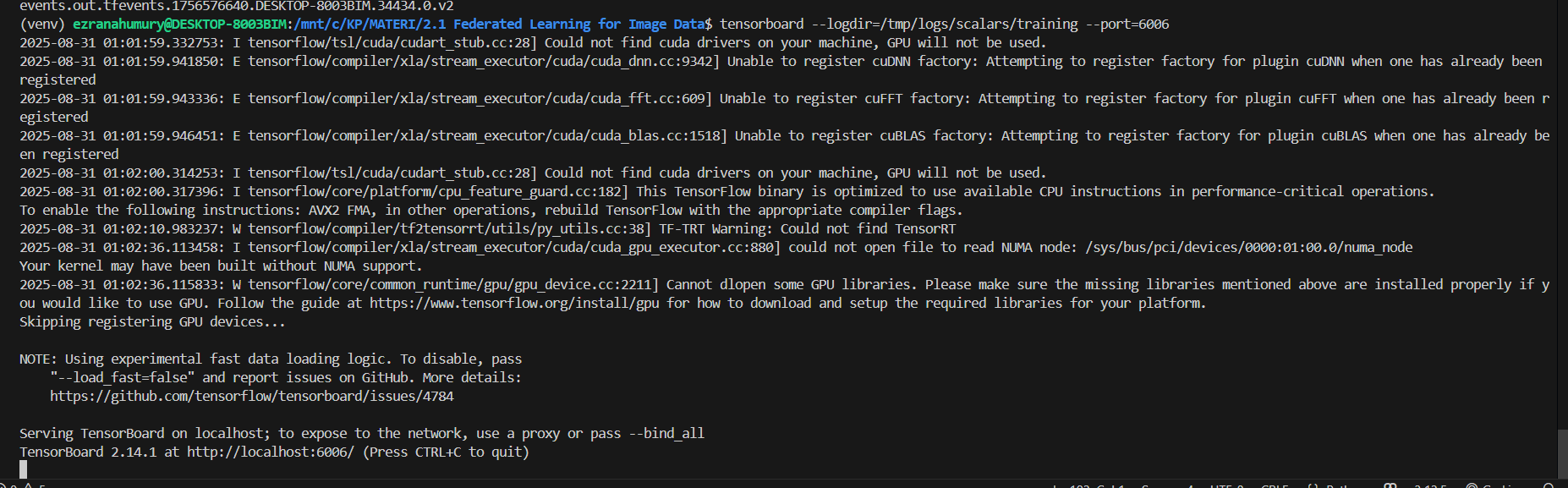


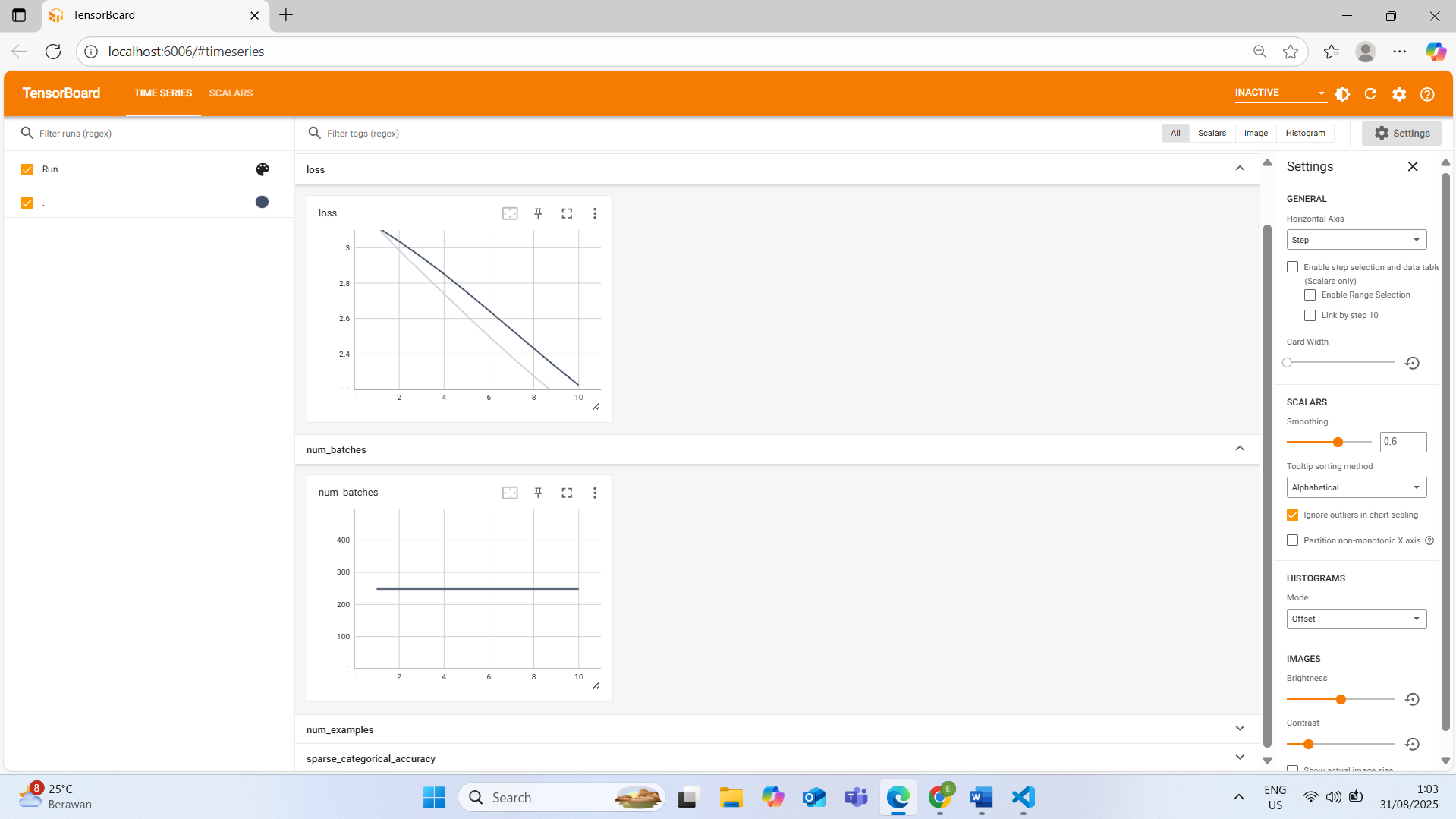
Plot metrics scalar yang relevan dengan penulis ringkasan yang sama :



Mulai TensorBoard dengan direktori root log yang ditentukan di atas. Diperlukan beberapa detik untuk memuat data.







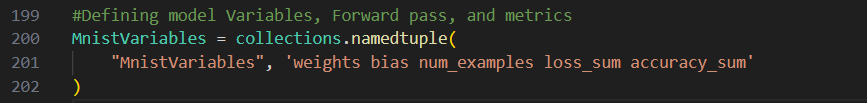
**Customizing the model implementation**

Keras Adalah antarmuka model Tingkat tinggi yang direkomendasikan untuk TensorFlow, dan kami menyarankan untuk menggunakan model Keras (melalui tff.learning.models.from\_keras\_model) dalam TFF sebanyak mungkin.

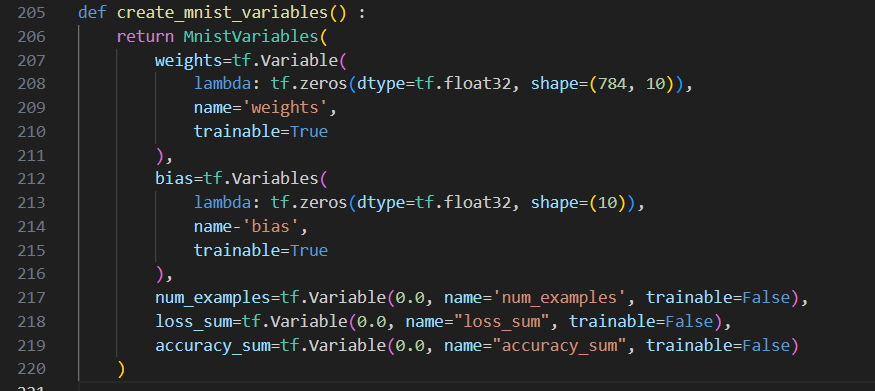
Namun, tff.learning menyediakan antarmuka model Tingkat rendah, tff.learning.models.VariableModel, yang mengekspos fungsionalitas minimal yang diperlukan untuk menggunakan model dalam pembelajaran federasi. Mengimplementasikan antarmuka ini secara langsung (mungkin masih menggunakan blok bangunan seperti tf.keras.layers) memungkinkan kustomisasi maksimal tanpa mengubah internal algoritma pembelajaran federasi.

1. Defining model Variables, Forward pass, and metrics

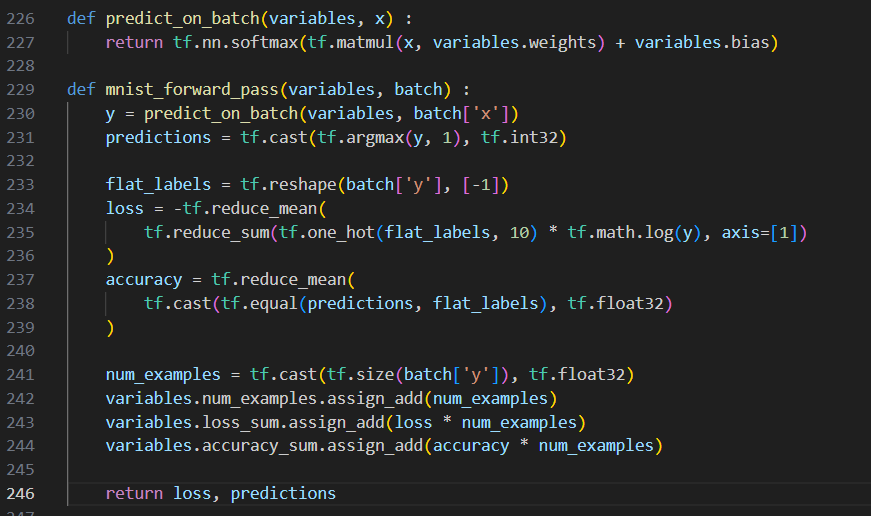
Langkah pertama Adalah mengidentifikasi variable Tensorflow yang akan kita gunakan. Untuk membuat kode berikut lebih mudah dibaca, kita definisikan struktur data untuk mewakili seluruh set. Struktur ini akan mencakup variable seperti bobot dan bias yang akan kita latih, serta variable yang akan menyimpan berbagai statistic kumpulatif dan penghitung yang akan kita perbarui selama pelatihan, seperti loss\_sum, accuracy\_sum, dan num\_examples.



Berikut Adalah metode yang membuat variable. Demi kesederhanaan, kami mewakili semua statistic sebagai tf.float32, seperti yang akan menghilangkan kebutuhan untuk konversi tipe pada tahap berikutnya. Pembungkus initializer variable sebagai lambdas Adalah persyaratan yang diberlakukan oleh variable sumber daya.

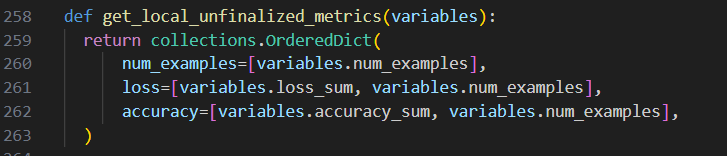


Dengan variable untuk parameter model dan statistic kumulatif, sekarang kita dapat mendefinisikan metode forward pass yang menghitung kerugian, memancarkan prediksi dan memperbarui statistic kumulatif untuk satu Kumpulan data input sebagai berikut :

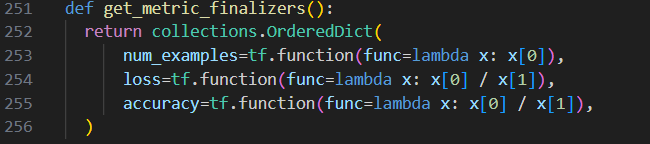


Selanjutnya, kami mendefinisikan fungsi yang mengembalikan sekumpulan metrics local, sekali lagi menggunakan Tensorflow. Ini Adalah nilai (selain pembaruan model, yang ditangani secara otomatis) yang memenuhi syarat untuk digabungkan ke server dalam proses pembelajaran atau evaluasi gabungan.

Kita hanya mengembalikan rata – rata loss dan accuracy, serta num\_examples, yang kita harus benar berat kontribusi dari pengguna yang berbeda Ketika menghitung agregat federasi.



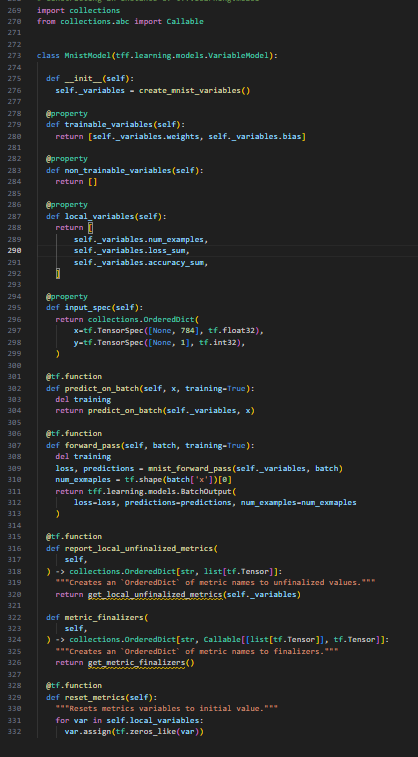
Kita perlu menentukan bagaimana agregat metrics local yang dipancarkan oleh masing – masing perangkat melalui perangkat get\_local\_mnist\_metrics. Ini Adalah satu – satunya bagian dari kode yang tidak ditulis dalam Tensorflow itu Adalah perhitungan federasi yang dinyatakan dalam TFF.



Input metrics argumen berkorespondensi dengan OrderedDict dikembalikan oleh get\_local\_mnist\_metrics di atas, tetapi kritis nilai-nilai tidak lagi tf.Tensors - mereka adalah "kotak" sebagai tff.Value s, untuk membuatnya jelas Anda tidak lagi dapat memanipulasi mereka menggunakan TensorFlow, tetapi hanya menggunakan operator federasi TFF seperti tff.federated\_mean dan tff.federated\_sum .

1. Constructing an instance of tff.learning.model

Membuat representasi model untuk digunakan dengan TFF serupa dengan yang dibuat dengan membiarkan TFF menyerap dengan model keras.



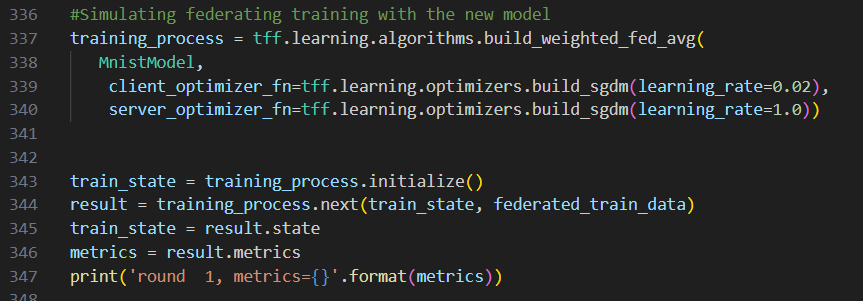
Metode abstrak dan properti yang didefinisikan oleh tff.learning.Model berkorespondensi dengan potongan kode di bagian sebelumnya yang memperkenalkan variabel dan mendefinisikan kerugian dan statistik.

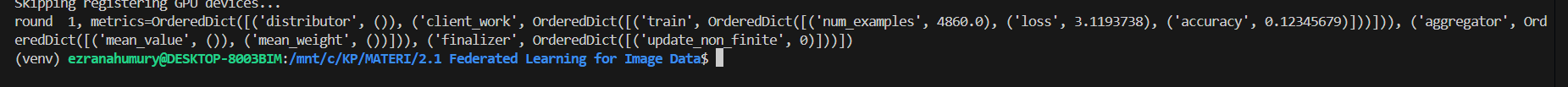
Berikut beberapa poin yang perlu ditekankan:

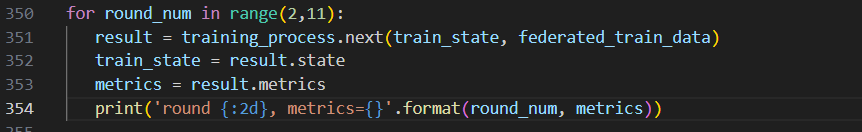
* Semua variable yang akan digunakan oleh model harus didefinisikan sebagai variable tensorflow, karena TFF tidak menggunakan Python saat Runtime.
* Model harus mendeskripsikan jenis data yang dapat diterimanya (input\_spec), karena secara umum TFF Adalah lingkungan yang kuat tipenya dan ingin menentukan tanda tipe untuk semua komponen.
* Membungkus semua logika Tensorflow (forward pass, perhitungan metrics, dll) sebagai tf.function, karena ini membantu memastikan tensorflow dapat diserialisasi dan menghilangkan kebutuhan akan dependensi control eksplisit.

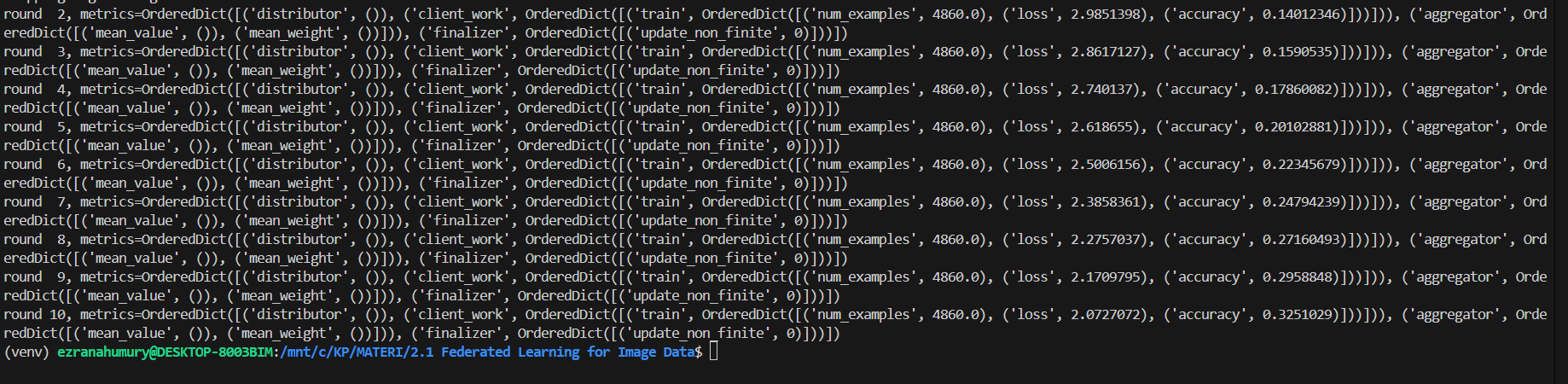
1. Simulating federating training with the new model

Mengganti konstruktur kelas model baru, dan gunakan dua perhitungan federasi dalam proses iterative yang telah dibuat untuk mengulang putaran pelatihan









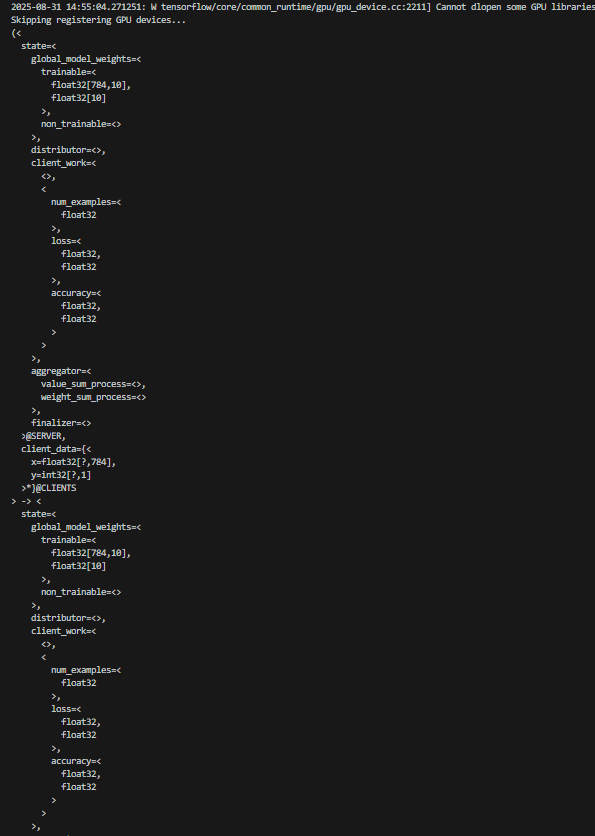
**Evaluation**

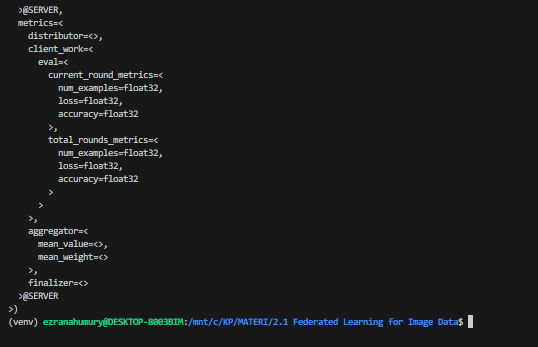
Untuk melakukan evaluasi data federasi, kita akan membuat perhitungan federasi lain yang dirancang untuk tujuan ini menggunakan tff.learning.build\_federated\_evaluation fungsi, dan lewat dalam model konstruktor sebagai argument. Evaluasi tidak melakukan penurunan gradien, dan tidak perlu membuat pengoptimal.



Memeriksa tipe signature abstrak dari fungsi evaluasi :





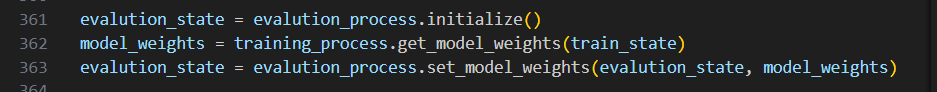


Mirip dengan tff.templates.IterativeProcess.next tetapi dengan dua perbandingan penting :

1. Kami tidak mengembalikan status server, karena evaluasi tidak mengubah model atau aspek status lainnya.
2. Evaluasi hanya membutuhkan model, dan tidak memerlukan bagian lain dari status server yang mungkin terkait dengan pelatihan, seperti variable pengoptimalan.

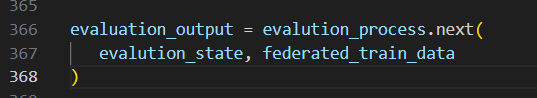


Mari kita lakukan evaluasi pada status terakhir yang kita capai selama pelatihan. Dalam rangka untuk mengekstrak model dilatih terbaru dari negara server, Anda cukup mengakses .model anggota, sebagai berikut.



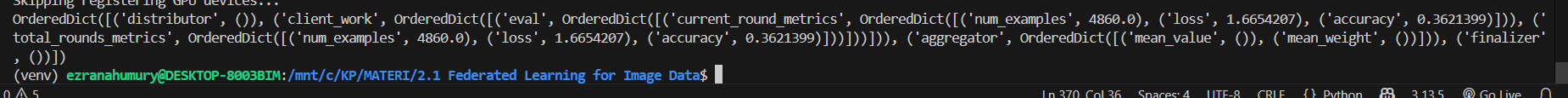
Dengan keadaan evaluasi yang berisi bobot model yang akan dievaluasi, kita dapat menghitung metrics evaluasi menggunakan dataset evaluasi dengan memanggil metode sama seperti saat pelatihan.

Ini akan mengembalikan instance tff.learning.templates.LearningProcessOutput

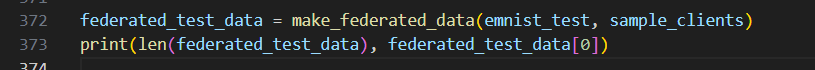


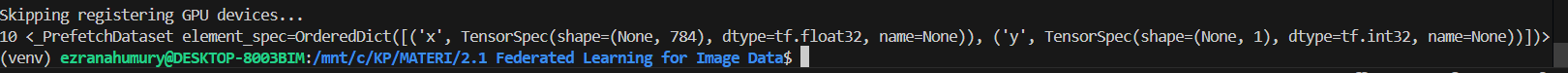
Secara konvensional, metrics pelatihan yang dilaporkan oleh proses pelatihan iterative umumnya mencerminkan kinerja model pada awal putaran pelatihan, sehingga metrics evaluasi selalu satu Langkah di depan.

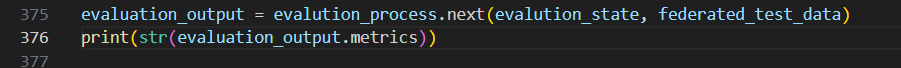




Mari kita kompilasi sampel uji data federasi dan jalankan kembali evaluasi pada data uji. Data tersebut akan berasal dari sampel pengguna nyata yang sama, tetapi dari kumpulan data yang dipisahkan secara terpisah.





’

