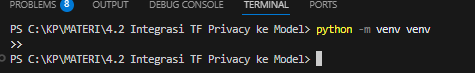
Topik : 4.2. Integrasi TF Privacy ke Model

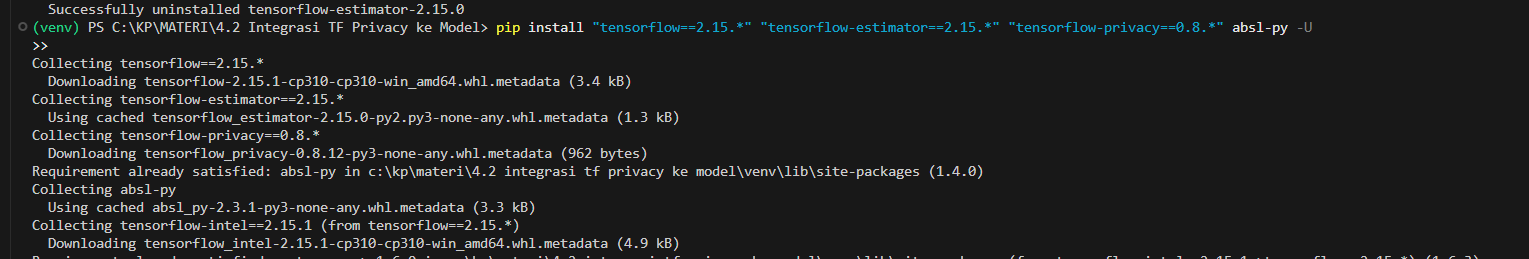
Objective : Implementasikan optimizer dengan DP-SGD

Task : Tambahkan dp\_keras\_optimizer ke model FL

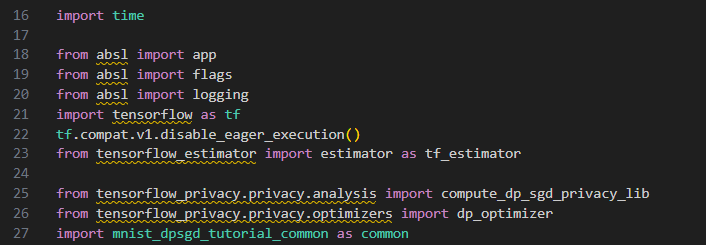
Source : <https://github.com/tensorflow/privacy/blob/master/tutorials/mnist_dpsgd_tutorial.py>





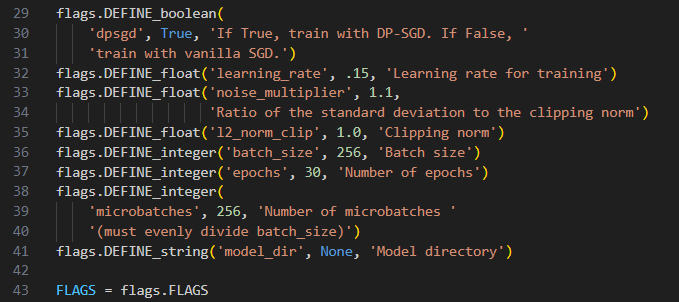


1. Import dan Konfigurasi Awal



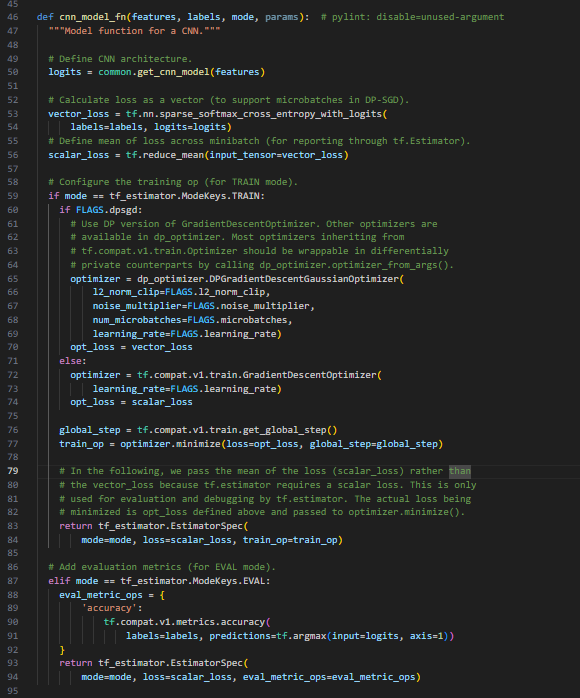
Mengaktifkan graph mode, menyiapkan estimator, DP-SGD, dan modul common yang berisi get\_cnn\_model() & make\_input\_fn()

1. Hyperparameter via absl.flags



semua parameter pelatihan & privasi dikontrol dari sini.

1. Model function untuk melakukan estimator (cnn\_model\_fn)



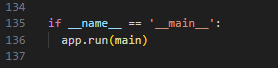
* **DP-SGD**: clipping L2 + Gaussian noise → wajib opt\_loss = vector\_loss.
* **Non-DP**: opt\_loss = scalar\_loss.
* Metrik yang dilaporkan saat evaluasi: **akurasi**.

1. Fungsi main: validasi, Estimator, loop train/eval, dan hitung ε



* **Validasi**: batch\_size harus habis dibagi microbatches.
* **Train/Eval** per-epoch.
* **Epsilon (ε)** dihitung kumulatif per-epoch → trade-off akurasi vs privasi.

1. Entry point



menjalankan main() dan mengurai flag dari command line.

Ringkasan :

* **Estimator** memanggil cnn\_model\_fn untuk **TRAIN** (pilih DP-SGD vs SGD) dan **EVAL** (akurasi).
* **DP-SGD**: pakai **loss per-contoh** (vector\_loss), **clipping L2** (l2\_norm\_clip), **noise Gaussian** (noise\_multiplier), dan **microbatches**.
* Setiap epoch: **latih → uji → cetak akurasi**, dan kalau DP aktif **hitung ε** (semakin kecil = privasi lebih kuat).

Source : <https://www.tensorflow.org/responsible_ai/privacy/tutorials/classification_privacy>

**Implement Differential Privacy with Tensorflow Privacy**

Privacy differensial (DP) Adalah sebuah kerangka kerja untuk mengukur jaminan privasi yang diberikan oleh sebuah algoritma. Melalui lensa privasi differensial , kita dapat merancang algoritma machine learning yang dapat melatih model secara bertanggung jawab pada data pribadi. Pembelajaran dengan privasi diferensial memberikan jaminan privasi yang terukur, membantu mengurangi risiko terbukanya data sensitif dalam pembelajaran mesin. Secara intuitif, model yang dilatih dengan privasi diferensial tidak boleh dipengaruhi oleh satu contoh pelatihan tunggal, atau kumpulan kecil contoh pelatihan, dalam himpunan datanya. Hal ini membantu mengurangi risiko terbukanya data sensitif dalam ML.

Ide dasar dari pendeketan ini, yang disebut *differentially private stochastic gradient descent (DP-SGD)*, Adalah melakukan modifikasi gradien yang akan digunakan dalam *stochastic gradient descent (SGD)*, yang merupakan inti dari hampir semua algoritma deep learning. Model yang dilatih dengan DP-SGD memberikan jaminan privasi diferensial yang terbukti untuk data input mereka.

2 modifikasi yang dilakukan pada algoritma SGD standar :

1. Sensitivitas setiap gradien perlu dibatasi

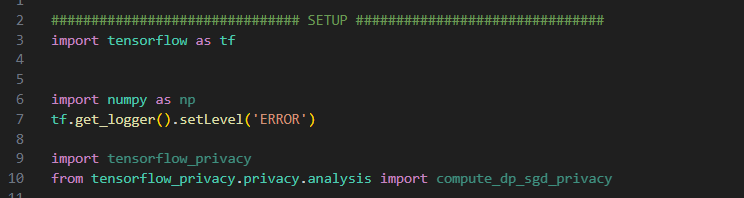
Dengan kata lain, kita perlu membatasi sejauh mana setiap titik data pelatihan yang diambil dalam sebuah minibatch dapat memengaruhi perhitungan gradien dan pembaruan yang diterapkan pada parameter model. Hal ini dilakukan dengan *clipping* (memotong) setiap gradient yang dihitung pada setiap titik pelatihan.

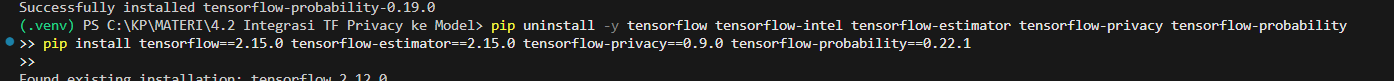
1. Noise acak

Diambil sampelnya dan ditambahkan pada gradient yang sudah dipotong, untuk membuatnya secara statistic mustahil mengetahui apakah suatu titik data tertentu termasuk atau tidak dalam dataset pelatihan, dengan cara membandingkan pembaruan yang dilakukan oleh SGD Ketika beroperasi dengan atau tanpa titik data tersebut dalam dataset pelatihan.

Tutorial ini menggunakan tf.keras untuk melatih *convolutional neural network (CNN)* dalam mengenali digit tulisan tangan dengan optimizer DP-SGD yang disediakan oleh Pustaka Tensorflow Privacy. Tensorflow Privacy menyediakan kode untuk membungkus optimizer Tensorflow yang sudah ada untuk membuat varian yang mengimplementasikan DP-SGD.

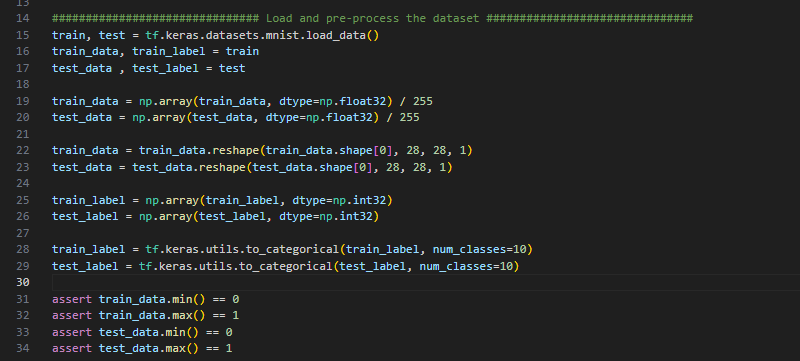
1. SetUp

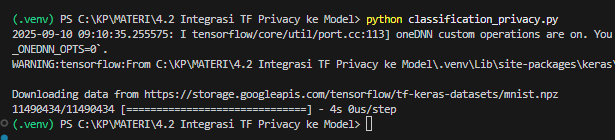




1. Load and pre-process the dataset

Muat dataset MNIST dan siapkan data untuk pelatihan.





1. Define the hyperparameters

Tetapkan nilai hiperparameter model pembelajaran.



DP-SGD memiliki tiga hyperparameter khusus privasi dan satu hyperparameter yang sudah ada yang harus anda atur :

1. l2\_norm\_clip (float)

Nilai maksimum norma Euclidean (L2) dari setiap gradient yang diterapkan untuk memperbarui parameter model. Hyperparameter ini digunakan untuk membatasi sensitivitas optimizer terhadap titik data pelatihan individu.

1. noise\_multiplier (float)

Jumlah noise yang diambil sampelnya dan ditambahkan ke gradien selama pelatihan. Secara umum, semakin banyak noise maka semakin menghasilkan privasi yang lebih baik (meskipun tidak selalu dengan mengorbankan utilitas yang lebih rendah)

1. microbatches (int)

Setiap batch data dibagi ke dalam unit – unit lebih kecil yang disebut microbatch. Secara default, setiap microbatch sebaiknya hanya berisi satu contoh pelatihan. Hal ini memungkinan kita untuk melakukan clipping gradien pada basis per-contoh, bukan setelah rata – rata di seluruh minibatch. Dengan demikian, efek (nefatif) clipping pada sinyal yang ditemukan di gradien berkurang dan biasanya memaksimalkan utilitas. Namun, overhead komputasi dapat dikurangi dengan memperbesar ukuran microbatch sehingga mencakup lebih dari satu contoh pelatihan. Rata – rata gradien di seluruh contoh pelatihan ini kemudian di-clipping.

Jumlah total contoh yang digunakan dalam satu batch (satu Langkah *gradient descent)* tetap sama. Jumlah microbatch harus membagi ukuran batch secara merata.

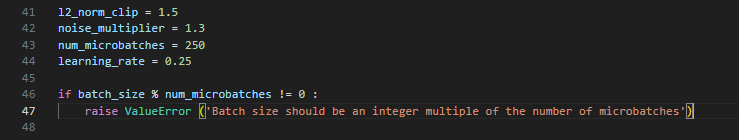
1. learning\_rate (float)

Hyperparameter ini sudah ada dalam vanilla SGD. Semakin tinggi nilai learning rate, semakin besar pengaruh setiap pembaruan. Jika pembaruan sangat bising (misalnya saat noise aditif besar dibandingkan ambang clipping), learning rate yang rendah dapat membantu prosedur pelatihan untuk konvergen.

Rangkuman :

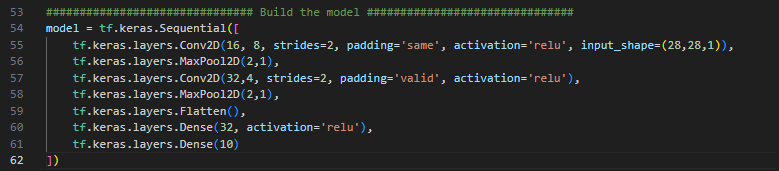
* l2\_norm\_clip → membatasi gradien biar tidak ada satu data mendominasi.
* noise\_multiplier → mengatur seberapa besar noise ditambahkan demi privasi.
* microbatches → membagi batch jadi unit kecil supaya clipping lebih akurat.
* learning\_rate → mengatur kecepatan update (harus seimbang dengan noise & clipping).

Gunakan nilai hiperparameter di bawah ini untuk mendapatkan model yang cukup akurat (akurasi pengujian 95%):

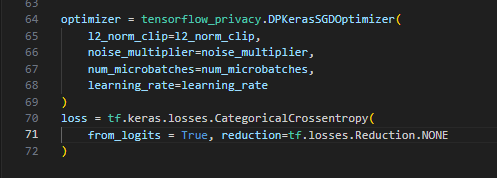


4. Build the model

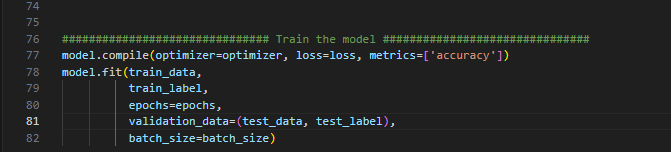
Tentukan jaringan saraf konvolusional sebagai model pembelajaran.

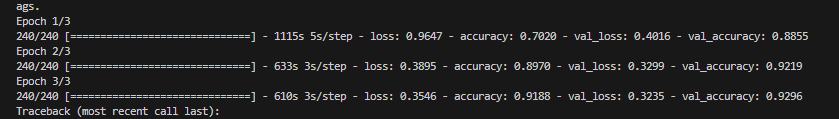


Tentukan pengoptimal dan fungsi kerugian untuk learning model. Hitung kerugian sebagai vector kerugian per-contoh, alih – alih sebagai rata – rata pada minibatch untuk mendukung manipulasi gradien pada setiap titik pelatihan.



5. Train the model





6. Measure the differential privacy guarantee

Melakukan analisis privasi untuk mengukur jaminan DP yang dicapai oleh algoritma pelatihan. Mengetahui Tingkat DP yang dicapai memungkinkan perbandingan objektif dari dua proses pelatihan untuk menentukan mana yang lebih menjaga privasi.

Analisis privasi mengukur seberapa jauh seorang lawan potensial dapat meningkatkan tebakan mereka tentang property titik data individu dengan mengamati hasil dari prosedur pelatihan (misalnya, pembaruan model dan parameter).

Jaminan ini kadang disebut sebagai anggaran privasi (privacy budget). Anggaran privasi yang lebih rendah membatasi lebih ketat kemampuan lawan untuk meningkatkan tebakan mereka. Hal ini memastikan jaminan privasi yang lebih kuat. Secara intuitif, ini karena lebih sulit bagi satu titik data pelatihan untuk memengaruhi hasil pembelajaran : misalnya, informasi dalam titik data pelatihan tidak bisa dihafal oleh algoritma ML, dan privasi individu yang menyumbangkan titik data tersebut ke dataset tetap terjaga.

Dalam tutorial ini, analisis privasi dilakukan dengan kerangka kerja *Renyi Differential Privacy (*RDP), yang merupakan relaksasi dari pure DP yang sangat cocok untuk DP-SGD.

Dua metrics yang digunakan untuk mengekspresikan jaminan DP dari sebuah algoritma ML :

1. **Delta (δ)**

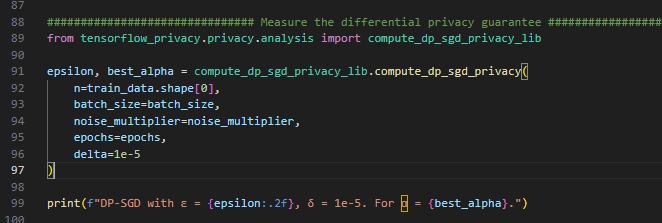
Membatasi probabilitas jaminan privasi tidak berlaku. Aturannya Adalah nilainya lebih kecil dari kebalikan ukuran dataset pelatihan. Dalam tutorial ini, δ diset ke 10^-5 karena dataset MNIST memiliki 60.000 titik data pelatihan.

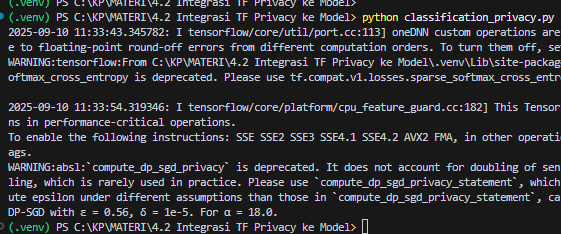
1. **Epsilon (ε)**

Ini Adalah privacy budget. Mengukur kekuatan jaminan privasi dengan membatasi seberapa besar probabilitas keluaran model tertentu dapat bervariasi dengan menyertakan (atau menghapus) satu titik data pelatihan. Nilai ε yang lebih kecil berarti jaminan privasi lebih baik. Namun, nilai ε hanyalah batas atas; nilai yang besar masih bisa berarti privasi yang cukup baik dalam praktik.

Tensorflow privacy menyediakan sebuah alat, **compute\_dp\_sgd\_privacy**, untuk menghitung nilai ε dengan nilai δ yang tetap dan hyperparameter berikut dari proses pelatihan:

1. Jumlah total titik data dalam data pelatihan, **n**
2. **Batch\_size**
3. **noise\_multiplier**
4. jumlah **epochs** pelatihan

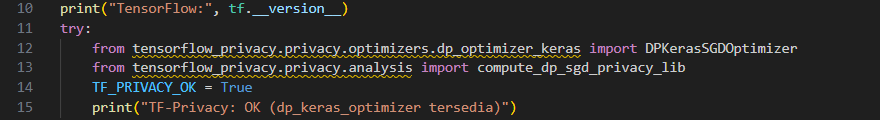




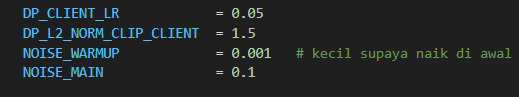
Alat tersebut melaporkan bahwa untuk hiperparameter yang dipilih di atas, model yang dilatih memiliki nilai sebesar 1,18.

**Task :** Tambahkan dp\_keras\_optimizer ke model FL

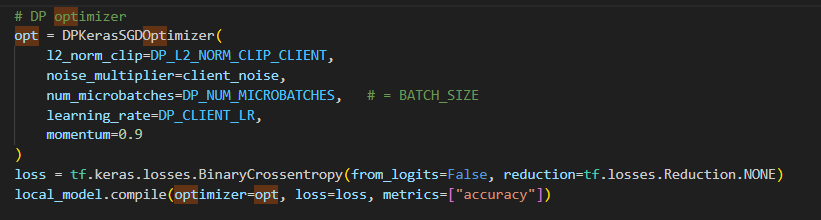
1. **Import & hyperparameter DP klien**







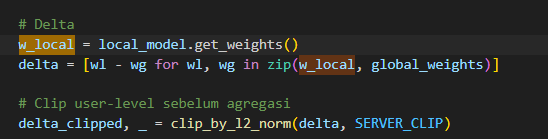
1. **Dipasang ke model klien saat compile**

****

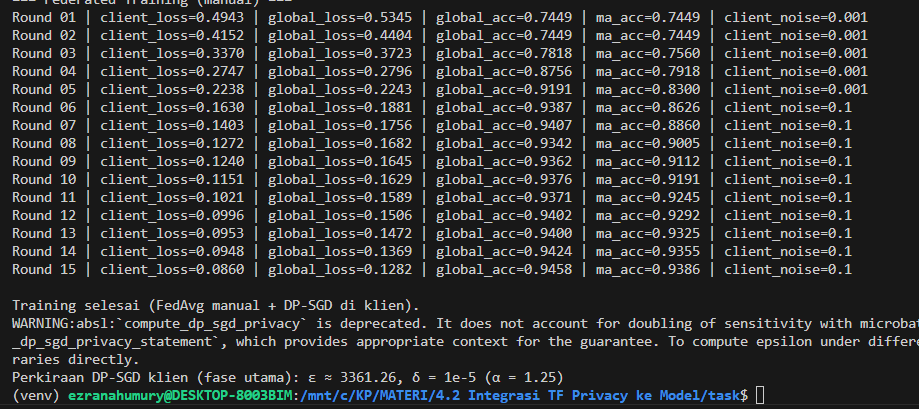
1. **Dataset klien disiapkan agar kompatibel dengan DP**

****

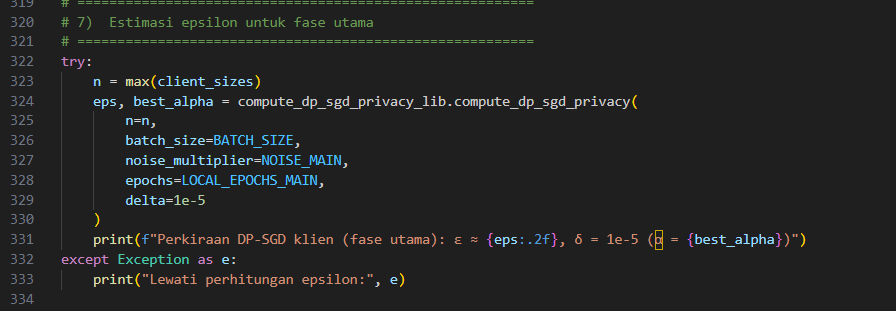
1. **Setelah training lokal selesai, delta diklip & di-agregasi → inilah bagian FL-nya:**

****

Hasil Training :



Measure the differential privacy





Estimasi ε, δ dengan compute\_dp\_sgd\_privacy\_lib.compute\_dp\_sgd\_privacy(...) per klien.