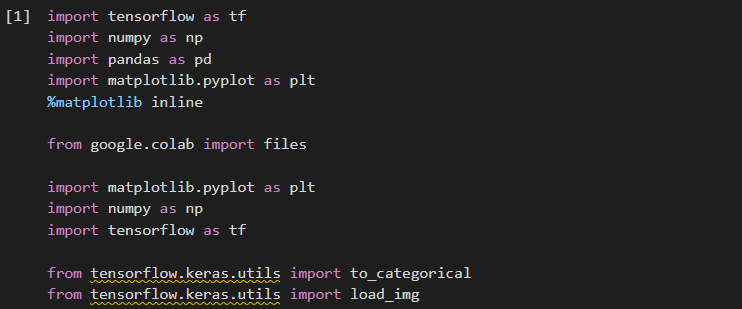
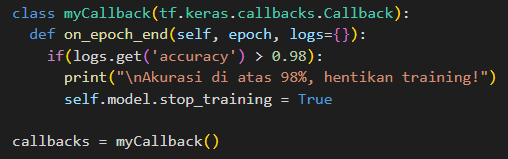
**La Ode Muhammad Sampaga**

**50421741**

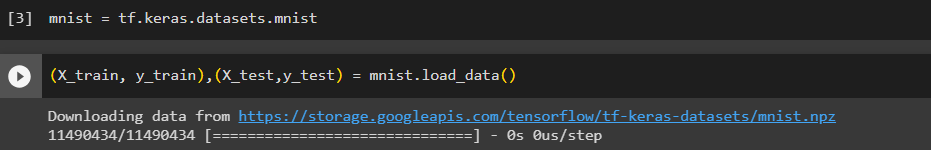


Import semua Library yang diperlukan, seperti TensorFlow,Numpy,Pandas dan Matplotlib



Membuat kelas *myCallback* menggunakan *tf.keras.callbacks.Callback.* berisi method *on\_epoch\_end* yang akan dipanggil disetiap akhir epoch saat pelatihan model nantinya.

Jika Akurasi pelatihan model > 98% maka pelatihan model akan di stop dengan cara mengaktifkan *self.model.stop\_training* menjadi **True**



Mengimport MNIST Dataset menggunakan modul *tf.keras.datasets.mnist*.

Dataset MNIST adalah kumpulan data gambar angka tulisan tangan berukuran 28x28 piksel

Membagi data menjadi empat bagian *X\_train, y\_train, X\_test, y\_test*.

**X\_train** berisi gambar-gambar angka 0 – 9, yang akan digunakan untuk pelatihan,

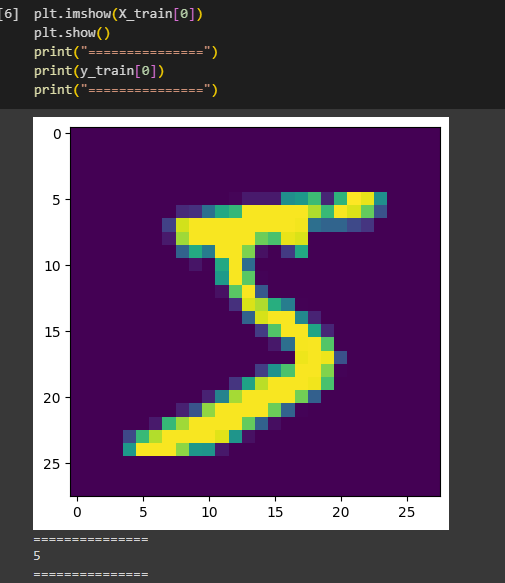
**y\_train** berisi label-label sesuai gambar angka-angka pada *X\_train.*

**X\_test** berisi gambar – gambar angka 0 – 9 yang akan digunakan untuk pengujian model.

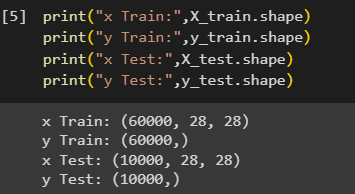
**y\_test** berisi label-label sesuai gambar angka-angka pada *X\_test.*

*Berikut gambaran sederhana bagaimana salah satu isi dataset X\_train dan y\_train.*

*Yaitu, X\_train[0] dan y\_train[0]:*



*X\_train[0]* berisi gambar angka 5 dan *y\_train[0]* berisi label angka 5



**X\_train** : (60000, 28, 28) artinya memiliki 60.000 gambar pelatihan dengan ukuran 28 x 28 piksel yang direpresentasikan dengan Matriks 2D

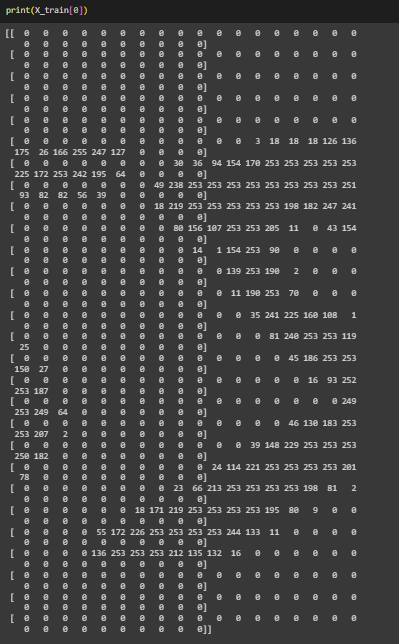
**Y\_train** : (60000,)

Array 1D yang berisi 60000 label untuk *X\_train*

**X\_test** : (10000, 28, 28) artinya memiliki 10.000 gambar uji dengan ukuran 28 x 28 piksel yang direpresentasikan dengan Matriks 2D

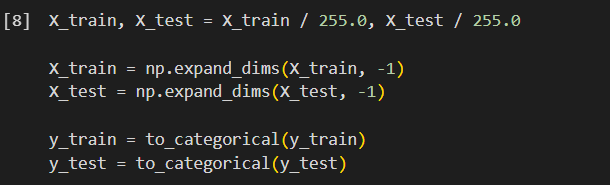
**Y\_test** : (10000,)

Array 1D yang berisi 10000 label untuk *X\_test*



Representasi Matriks 2D *X\_train[0]* yang dapat dilihat seperti angka 5.

Nilai-nilai matriks diatas berisi antara 0 sampai 255, dimana 0 adalah warna hitam dan 255 adalah warna putih.



***X\_train, X\_test = X\_train / 255.0, X\_test / 255.0***

Menormalisasi nilai piksel pada *X\_train* dan *X\_test* dengan membagi nya dengan 255.

Normalisasi digunakan agar model CNN kita nantinya berkerja lebih baik.

***X\_train = np.expand\_dims(X\_train, -1)***

***X\_test = np.expand\_dims(X\_test, -1)***

Menambah dimensi baru pada *X\_train* dan *X\_test* pada akhir untuk memenuhi kebutuhan format input yang diterima oleh Model CNN.

Sebelum: (28, 28) ---- > tidak dapat diterima model CNN

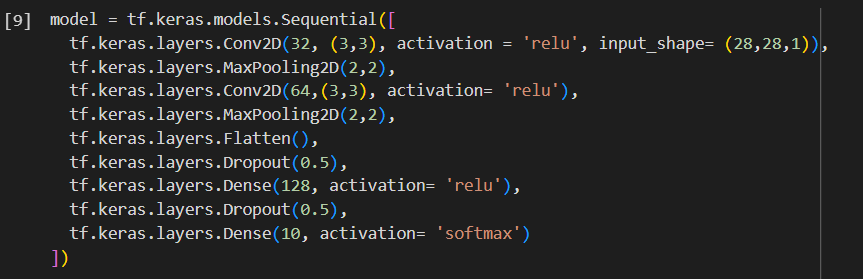
Sesudah: (28, 28, 1)

***y\_train = to\_categorical(y\_train)***

***y\_test = to\_categorical(y\_test)***

One-hot Encoding terhadap label dalam dataset. One-hot Encoding mengubah bentuk biner sesuai dengan jumlah kelas.

*Pada kasus kali ini sebenarnya one-hot Encoding dapat tidak dilakukan, tapi dengan catatan nanti kita mengubah fungsi loss dibawah dengan “sparse\_categorical\_crossentropy”*



*Tf.keras.models.sequential* berarti kita akan menggunakan Multi-Layer-Perceptron (MLP)

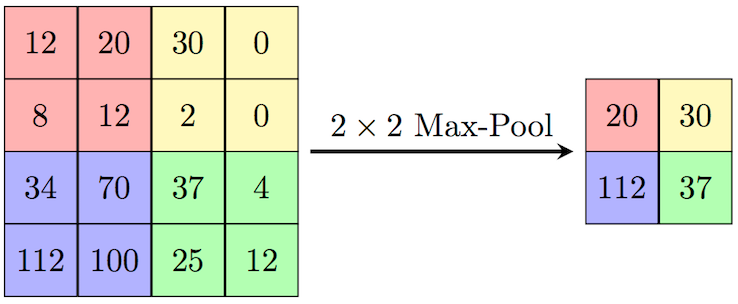
**Conv2D Layer (32, (3,3), activation=’relu’, input\_shape=(28,28,1))**

Layer Input pada model kali ini memiliki 32 Filter dengan ukuran (3,3) dan Fungsi Aktivasi yang digunakan adalah ReLU. Layer ini menggunakan input shape (28, 28, 1) karena diatas tadi kita telah menambahkan dimensi pada gambar MNIST kita.

**MaxPooling2D (2,2)**

layer max pooling dengan ukuran filter 2x2, mengurangi ukura gambar menjadi setengah.

Pooling adalah proses untuk mengurangi resolusi gambar dengan tetap mempertahankan informasi pada gambar.



Proses max pooling dipakai karena, jumlah filter yang digunakan pada proses konvolusi berjumlah banyak. Ketika kita menggunakan 64 filter pada konvolusi maka akan menghasilkan 64 gambar baru. Max pooling membantu mengurangi ukuran dari setiap gambar dari proses konvolusi.

**Flatten()**

Mentransformasikan matriks 2D menjadi vector 1D. digunakan agar dapat dipakai untuk Fully Connected Layers nantinya.

**Dropout(0.5)**

Menghapus secara acak neuron selama pelatihan yang akan membantu untuk mencegah overfitting.

*0.5* artinya setengah dari neuron akan di drop selama pelatihan.

**Dense(128, activation= 'relu'),**

Fully Connected Layer dengan 128 neuron dengan mengguanakn fungsi aktivasi ReLU

**Dense(10, activation= 'softmax')**

Layer output yang memiliki 10 neuron sesuai dengan jumlah kelas pada dataset MNIST, yaitu 0 – 9. Untuk dataset yang memiliki 3 kelas atau lebih, gunakan fungsi aktivasi Softmax.

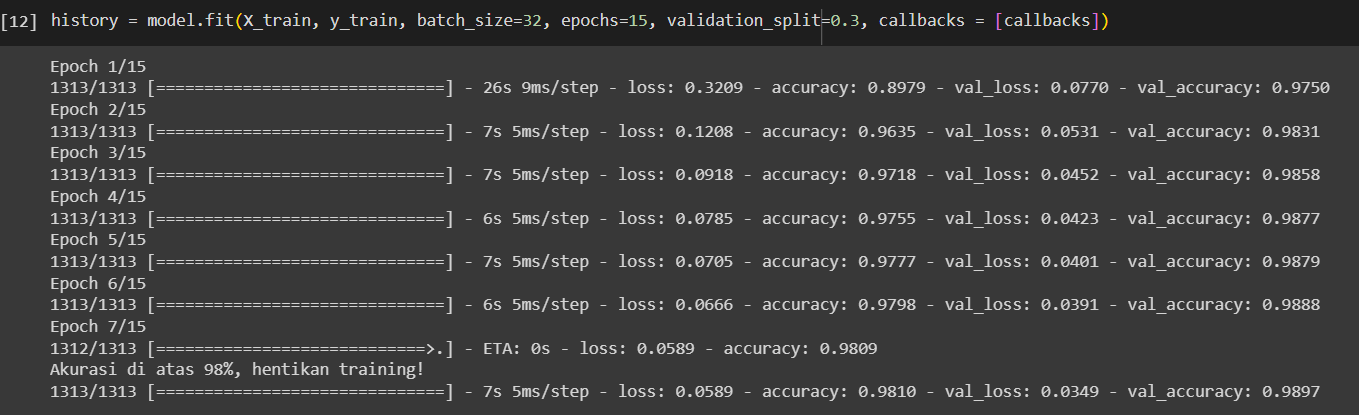
Fungsi aktivasi Softmax akan menghasilkan probabilitas dari kelas-kelas yang ada.



Pada model.compile kali saya menggunakan **Loss Function** nya yaitu “Categorical\_crossentropy” karena kita telah melakukan one-hot-encoding diatas.

**Optimizer** yang digunakan adalah “adam” yang digunakan untuk mengoptimalkan bobot jaringan selama pelatihan dengan mengurangi nilai loss nya. “adam” adalah optimizer yang paling umum digunakan.

**Matriks evaluasi** yang akan mengukur seberapa akurat model dalam memprediksi kelas-kelas yang ada



**X\_train dan y\_train** (label) akan digunakan untuk data pelatihan. **Batch\_size** menggunakan ukuran sebesar 32, semakin besar batch\_size maka akan semakin cepat proses modelling-nya.

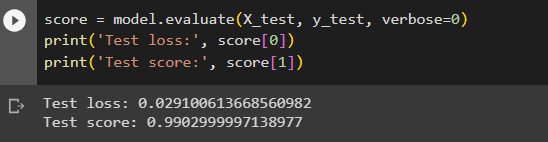
**Epoch** jumlah iterasi model yang kita tentukan sebanyak 15 kali.

**Validation\_split** yang akan digunakan untuk mengukur peforma model pada setiap epoch. Kali ini saya menggunakan 30% dari data sebagai data validation.

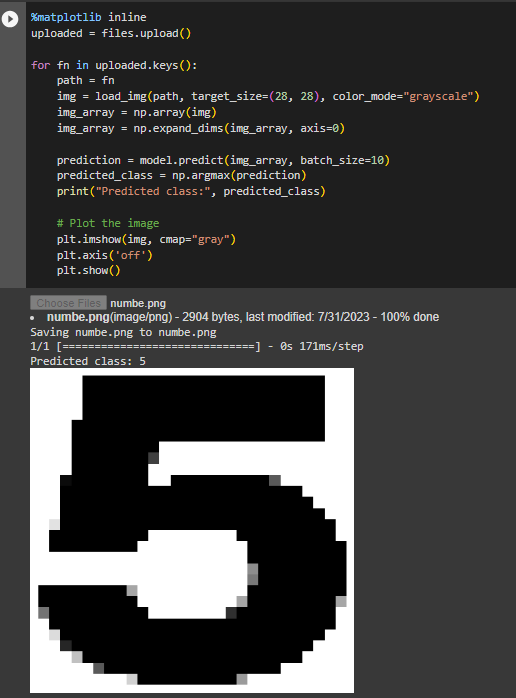
**Callback** yang telah kita buat diatas, yaitu akan men-stop pemodelan jika akurasi telah diatas 98% digunakan agar proses modelling tidak memakan banyak waktu.



Memvisualisasikan accuracy dari training dan validation, dapat dilihat terjadi kenaikan drastis dari iterasi-0 ke iterasi-1 dan terjadi peningkatan stabil seterusnya.



Mengetest model kita pada *X\_test dan y\_test* dan mendapatkan accuracy score **99%** dan loss score sebanyak **0.02%**





Mengetest model kita dengan gambar yang kita punya. Dengan memanfaatkan google colab untuk mengunggah gambar menggunakan modul *google.colab import files*