In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/digit-recognizer/sample_submission.csv
/kaggle/input/digit-recognizer/train.csv
/kaggle/input/digit-recognizer/test.csv
```

In [2]:
```
train_df = pd.read_csv('../input/digit-recognizer/train.csv')
test_df = pd.read_csv('../input/digit-recognizer/test.csv')
```

In [3]:
```
image_train = train_df.drop(columns=['label'])
label_train = train_df['label']
image_test = test_df
```

In [4]:
```
image_train = np.array(image_train)
image_train = image_train.reshape(image_train.shape[0],28,28,1)

image_test = np.array(image_test)
image_test = image_test.reshape(image_test.shape[0],28,28,1)
```

In [5]:
```
image_train = image_train.astype('float32') / 255.
image_test = image_test.astype('float32') / 255.
```

In [6]:
```
from keras.utils.np_utils import to_categorical
n_classes =10
label_train = to_categorical(label_train,n_classes)
```

In [7]:
```
print(image_test.shape)
```

```
(28000, 28, 28, 1)
```

In [8]:
```
image_train = np.reshape(image_train,(42000,784,))
```

In [9]:
```
image_test = np.reshape(image_test,(28000,784,))
```

In [10]:

```
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
,Flatten
from keras.models import Model

# Autoencoder 1
input_img1 = Input(shape=(784,))

############
# Encoding #
############
#x = Flatten()(input_img)
code1 = Dense(512,activation = 'relu')(input_img1)

############
# Decoding #
############

decoded1 = Dense(784,activation='sigmoid')(code1)

# Model
Autoencoder1 = Model(input_img1,decoded1)

Autoencoder1.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense (Dense)                (None, 512)               401920
_________________________________________________________________
dense_1 (Dense)              (None, 784)               402192
=================================================================
Total params: 804,112
Trainable params: 804,112
Non-trainable params: 0
_________________________________________________________________
```

```
2022-04-21 12:36:21.811092: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:21.888800: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:21.889628: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:21.890874: I tensorflow/core/platform/cpu_feature_gua
rd.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural
Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the approp
riate compiler flags.
2022-04-21 12:36:21.891210: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:21.892116: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:21.892950: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:23.405070: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:23.405901: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:23.406597: I tensorflow/stream_executor/cuda/cuda_gpu
_executor.cc:937] successful NUMA node read from SysFS had negative va
```

```
lue (-1), but there must be at least one NUMA node, so returning NUMA
node zero
2022-04-21 12:36:23.407632: I tensorflow/core/common_runtime/gpu/gpu_d
evice.cc:1510] Created device /job:localhost/replica:0/task:0/device:G
PU:0 with 15403 MB memory:  -> device: 0, name: Tesla P100-PCIE-16GB,
pci bus id: 0000:00:04.0, compute capability: 6.0
```

In [11]:

```
Autoencoder1.compile(optimizer='adam', loss='mse')
Autoencoder1.fit(image_train, image_train, epochs=10)
```

```
2022-04-21 12:36:24.426401: I tensorflow/compiler/mlir/mlir_graph_opti
mization_pass.cc:185] None of the MLIR Optimization Passes are enabled
(registered 2)


Epoch 1/10
1313/1313 [==============================] - 4s 2ms/step - loss: 0.012
4
Epoch 2/10
1313/1313 [==============================] - 2s 2ms/step - loss: 0.002
5
Epoch 3/10
1313/1313 [==============================] - 2s 2ms/step - loss: 0.001
7
Epoch 4/10
1313/1313 [==============================] - 2s 2ms/step - loss: 0.001
4
Epoch 5/10
1313/1313 [==============================] - 2s 2ms/step - loss: 0.001
3
Epoch 6/10
1313/1313 [==============================] - 3s 2ms/step - loss: 0.001
2
Epoch 7/10
1313/1313 [==============================] - 2s 2ms/step - loss: 0.001
1
Epoch 8/10
1313/1313 [==============================] - 3s 2ms/step - loss: 0.001
0
Epoch 9/10
1313/1313 [==============================] - 2s 2ms/step - loss: 9.538
3e-04
Epoch 10/10
1313/1313 [==============================] - 3s 2ms/step - loss: 9.077
8e-04
```

Out[11]:

```
<keras.callbacks.History at 0x7fdec1bfee10>
```

In [12]:

```
autoencoder1_hidden_output=Autoencoder1.layers[1].output
trimmed_autoencoder1=Model(inputs=input_img1, outputs=autoencoder1_hidden
_output)
image_train1=trimmed_autoencoder1.predict(image_train)
image_test1=trimmed_autoencoder1.predict(image_test)
```

In [13]:

```
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
,Flatten
from keras.models import Model

# Autoencoder 2
input_img2 = Input(shape=(512,))

############
# Encoding #
############
#x = Flatten()(input_img)
code2 = Dense(256,activation = 'relu')(input_img2)

############
# Decoding #
############

decoded2 = Dense(512,activation='sigmoid')(code2)

# Model
Autoencoder2 = Model(input_img2,decoded2)

Autoencoder2.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 512)]             0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
=================================================================
Total params: 262,912
Trainable params: 262,912
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
Autoencoder2.compile(optimizer='adam', loss='mse')
Autoencoder2.fit(image_train1, image_train1, epochs=50)
```

```
Epoch 1/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.432
4
Epoch 2/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.390
2
Epoch 3/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.382
8
Epoch 4/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.380
4
Epoch 5/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.379
4
Epoch 6/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.378
8
Epoch 7/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.378
4
Epoch 8/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.378
2
Epoch 9/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.378
0
Epoch 10/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.377
8
Epoch 11/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.377
6
Epoch 12/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.377
3
Epoch 13/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.377
1
Epoch 14/50
```

```
1313/1313 [==============================] - 3s 2ms/step - loss: 0.377
0
Epoch 15/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
9
Epoch 16/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
8
Epoch 17/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
7
Epoch 18/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
6
Epoch 19/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
5
Epoch 20/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
5
Epoch 21/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
4
Epoch 22/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
4
Epoch 23/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
4
Epoch 24/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
4
Epoch 25/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
4
Epoch 26/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
4
Epoch 27/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
3
```

```
Epoch 28/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
3
Epoch 29/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
3
Epoch 30/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
3
Epoch 31/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
2
Epoch 32/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
1
Epoch 33/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
1
Epoch 34/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
1
Epoch 35/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 36/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 37/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
0
Epoch 38/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 39/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
0
Epoch 40/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 41/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
```

```
0
Epoch 42/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 43/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 44/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
0
Epoch 45/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.376
0
Epoch 46/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.376
0
Epoch 47/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.375
9
Epoch 48/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.375
9
Epoch 49/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.375
9
Epoch 50/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.375
9
```

Out[14]:

```
<keras.callbacks.History at 0x7fdec01620d0>
```

In [15]:

```
autoencoder2_hidden_output=Autoencoder2.layers[1].output
trimmed_autoencoder2=Model(inputs=input_img2, outputs=autoencoder2_hidden
_output)
image_train2=trimmed_autoencoder2.predict(image_train1)
image_test2=trimmed_autoencoder2.predict(image_test1)
```

In [16]:

```
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
,Flatten
from keras.models import Model


input_img3 = Input(shape=(256,))

output = Dense(10,activation = 'softmax')(input_img3)


# Model
Autoencoder3 = Model(input_img3,output)

Autoencoder3.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 256)]             0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 2,570
Trainable params: 2,570
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
import tensorflow as tf
Autoencoder3.compile(optimizer='adam',loss= tf.keras.losses.CategoricalCr
ossentropy(from_logits=True),
              metrics=['accuracy'])

Autoencoder3.fit(image_train2, label_train, epochs=50)
```

```
Epoch 1/50


/opt/conda/lib/python3.7/site-packages/keras/backend.py:4847: UserWarn
ing: "`categorical_crossentropy` received `from_logits=True`, but the
`output` argument was produced by a sigmoid or softmax activation and
thus does not represent logits. Was this intended?"
  '"`categorical_crossentropy` received `from_logits=True`, but '
```

```
1313/1313 [==============================] - 3s 2ms/step - loss: 0.992
8 - accuracy: 0.7064
Epoch 2/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.387
7 - accuracy: 0.8865
Epoch 3/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.332
0 - accuracy: 0.9031
Epoch 4/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.310
7 - accuracy: 0.9103
Epoch 5/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.302
4 - accuracy: 0.9121
Epoch 6/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.298
0 - accuracy: 0.9132
Epoch 7/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.291
7 - accuracy: 0.9155
Epoch 8/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.289
6 - accuracy: 0.9160
Epoch 9/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.290
8 - accuracy: 0.9160
Epoch 10/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.288
6 - accuracy: 0.9160
Epoch 11/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.286
4 - accuracy: 0.9179
Epoch 12/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.286
6 - accuracy: 0.9177
Epoch 13/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.285
0 - accuracy: 0.9177
Epoch 14/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
```

```
5 - accuracy: 0.9190
Epoch 15/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.283
0 - accuracy: 0.9178
Epoch 16/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.283
3 - accuracy: 0.9192
Epoch 17/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.283
9 - accuracy: 0.9185
Epoch 18/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.285
4 - accuracy: 0.9188
Epoch 19/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.280
6 - accuracy: 0.9201
Epoch 20/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
3 - accuracy: 0.9190
Epoch 21/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.280
3 - accuracy: 0.9201
Epoch 22/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
6 - accuracy: 0.9181
Epoch 23/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.282
7 - accuracy: 0.9188
Epoch 24/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.280
7 - accuracy: 0.9203
Epoch 25/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
7 - accuracy: 0.9196
Epoch 26/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.281
3 - accuracy: 0.9190
Epoch 27/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.281
7 - accuracy: 0.9203
Epoch 28/50
```

```
1313/1313 [==============================] - 2s 2ms/step - loss: 0.280
6 - accuracy: 0.9200
Epoch 29/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.282
6 - accuracy: 0.9193
Epoch 30/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.281
8 - accuracy: 0.9206
Epoch 31/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.281
8 - accuracy: 0.9198
Epoch 32/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.281
7 - accuracy: 0.9194
Epoch 33/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.282
7 - accuracy: 0.9181
Epoch 34/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.283
0 - accuracy: 0.9195
Epoch 35/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
3 - accuracy: 0.9194
Epoch 36/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.278
9 - accuracy: 0.9208
Epoch 37/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
3 - accuracy: 0.9200
Epoch 38/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.280
4 - accuracy: 0.9205
Epoch 39/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.282
9 - accuracy: 0.9197
Epoch 40/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.280
1 - accuracy: 0.9202
Epoch 41/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.280
9 - accuracy: 0.9198
```

```
Epoch 42/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.280
1 - accuracy: 0.9203
Epoch 43/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.279
9 - accuracy: 0.9212
Epoch 44/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.282
4 - accuracy: 0.9200
Epoch 45/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.280
8 - accuracy: 0.9200
Epoch 46/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.281
5 - accuracy: 0.9198
Epoch 47/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.281
9 - accuracy: 0.9203
Epoch 48/50
1313/1313 [==============================] - 2s 2ms/step - loss: 0.282
3 - accuracy: 0.9201
Epoch 49/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.281
9 - accuracy: 0.9197
Epoch 50/50
1313/1313 [==============================] - 3s 2ms/step - loss: 0.279
8 - accuracy: 0.9206
```

Out[17]:

```
<keras.callbacks.History at 0x7fdeb2da6910>
```

In [18]:

```
pred = Autoencoder3.predict(image_test2)
predictions_final=np.argmax(pred, axis=1)
print(predictions_final)
```

```
[2 0 9 ... 3 9 2]
```

In [19]:

```
sub = pd.read_csv('/kaggle/input/digit-recognizer/sample_submission.csv')
sub["Label"] = predictions_final
sub.to_csv('submission.csv',index=False)
```