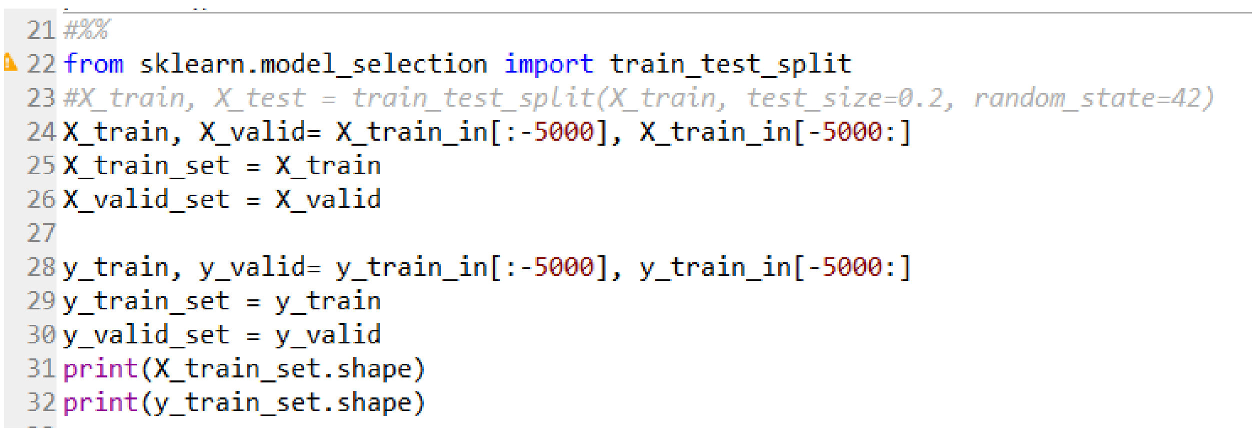
Tam Phi
ECEC 622
Report 4

Introduction

The goal of this project is to clean up noisy image and predict the digits from the noisy datas

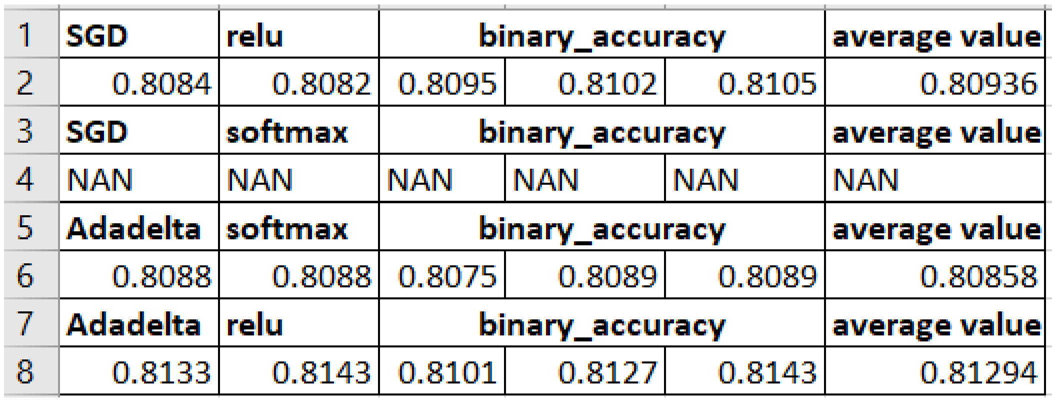
Implementation

Imported and divided datasets into train and valid sets



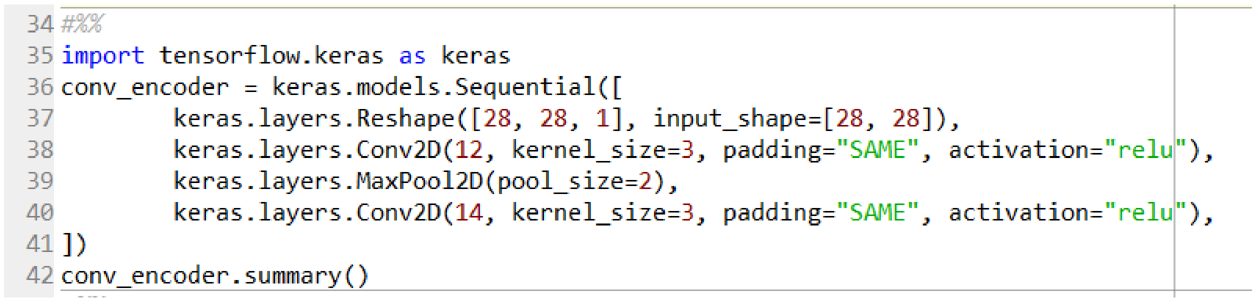
```
#%%
from sklearn.model_selection import train_test_split
#X_train, X_test = train_test_split(X_train, test_size=0.2, random_state=42)
X_train, X_valid= X_train_in[:-5000], X_train_in[-5000:]
X_train_set = X_train
X_valid_set = X_valid

y_train, y_valid= y_train_in[:-5000], y_train_in[-5000:]
y_train_set = y_train
y_valid_set = y_valid
print(X_train_set.shape)
print(y_train_set.shape)
```

A simple for loop was used to compute binary_accuracy value for different hyperparameters and calculate the average accuracy for each combination. The highest value belongs to [Adelta, relu, binary_accuracy]



SGD	**relu**	**binary_accuracy**			**average value**
0.8084	0.8082	0.8095	0.8102	0.8105	0.80936
SGD	**softmax**	**binary_accuracy**			**average value**
NAN	NAN	NAN	NAN	NAN	NAN
Adadelta	**softmax**	**binary_accuracy**			**average value**
0.8088	0.8088	0.8075	0.8089	0.8089	0.80858
Adadelta	**relu**	**binary_accuracy**			**average value**
0.8133	0.8143	0.8101	0.8127	0.8143	0.81294

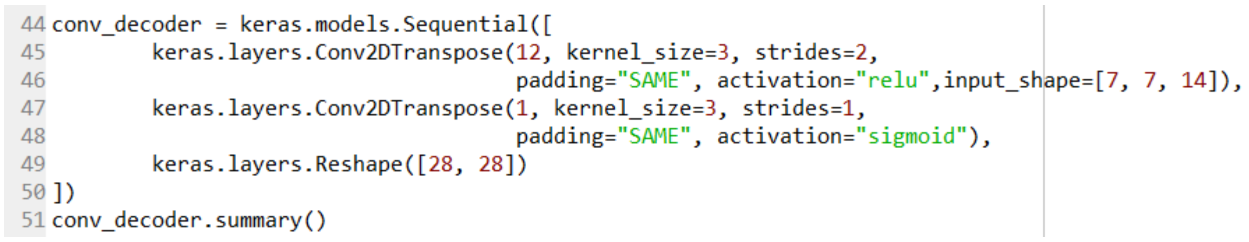
This program uses CNN autoencoder. There are 2 layers with filters of size 12 and 14, and 1 pooling layer of size 2.



```
#%%
import tensorflow.keras as keras
conv_encoder = keras.models.Sequential([
        keras.layers.Reshape([28, 28, 1], input_shape=[28, 28]),
        keras.layers.Conv2D(12, kernel_size=3, padding="SAME", activation="relu"),
        keras.layers.MaxPool2D(pool_size=2),
        keras.layers.Conv2D(14, kernel_size=3, padding="SAME", activation="relu"),
])
conv_encoder.summary()
```

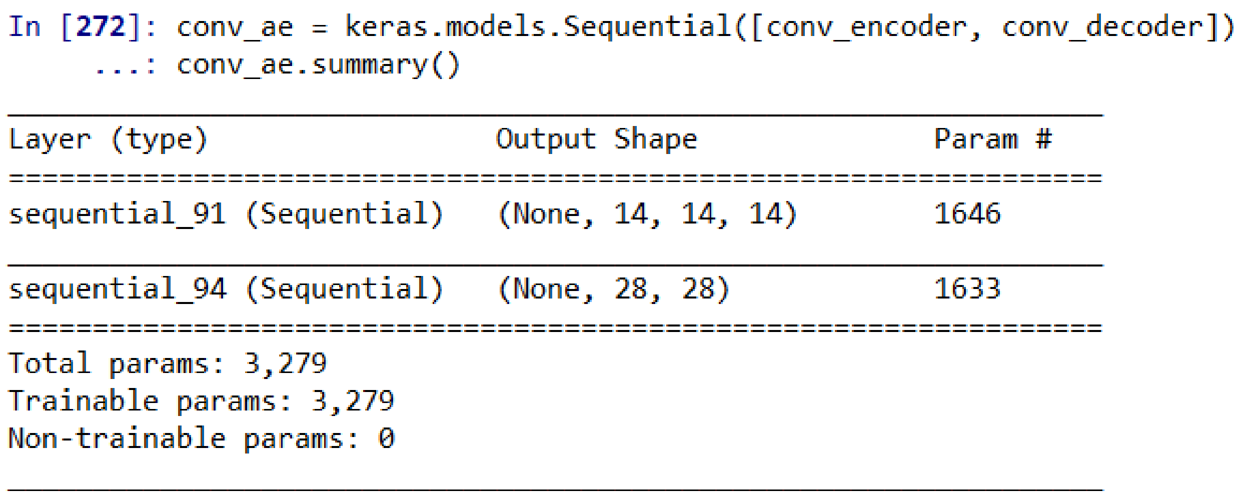
Tam Phi
ECEC 622
Report 4

To restore the compressed representation, a decoder with 2 layer was used.



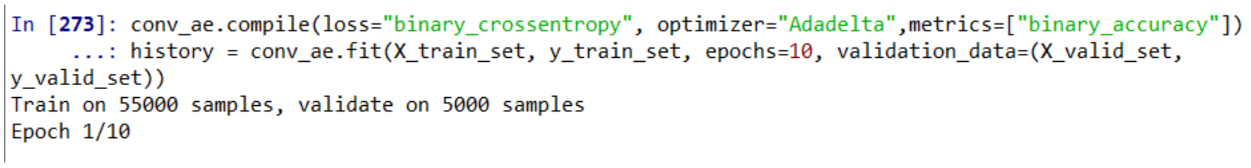
```
conv_decoder = keras.models.Sequential([
        keras.layers.Conv2DTranspose(12, kernel_size=3, strides=2,
                                     padding="SAME", activation="relu",input_shape=[7, 7, 14]),
        keras.layers.Conv2DTranspose(1, kernel_size=3, strides=1,
                                     padding="SAME", activation="sigmoid"),
        keras.layers.Reshape([28, 28])
])
conv_decoder.summary()
```

The final model has a total of 3279 parameters

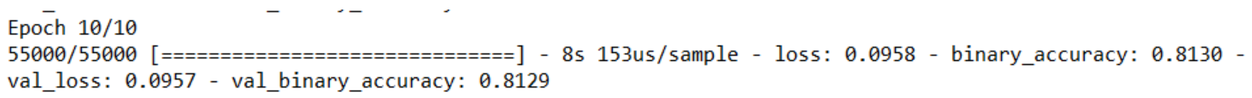


```
In [272]: conv_ae = keras.models.Sequential([conv_encoder, conv_decoder])
     ...: conv_ae.summary()

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_91 (Sequential)   (None, 14, 14, 14)        1646
_________________________________________________________________
sequential_94 (Sequential)   (None, 28, 28)            1633
=================================================================
Total params: 3,279
Trainable params: 3,279
Non-trainable params: 0
_________________________________________________________________
```

The results are reasonably good for a small number of training parameters with the highest accuracy = 0.8130 (full results on last page)



```
In [273]: conv_ae.compile(loss="binary_crossentropy", optimizer="Adadelta",metrics=["binary_accuracy"])
     ...: history = conv_ae.fit(X_train_set, y_train_set, epochs=10, validation_data=(X_valid_set,
y_valid_set))
Train on 55000 samples, validate on 5000 samples
Epoch 1/10
```

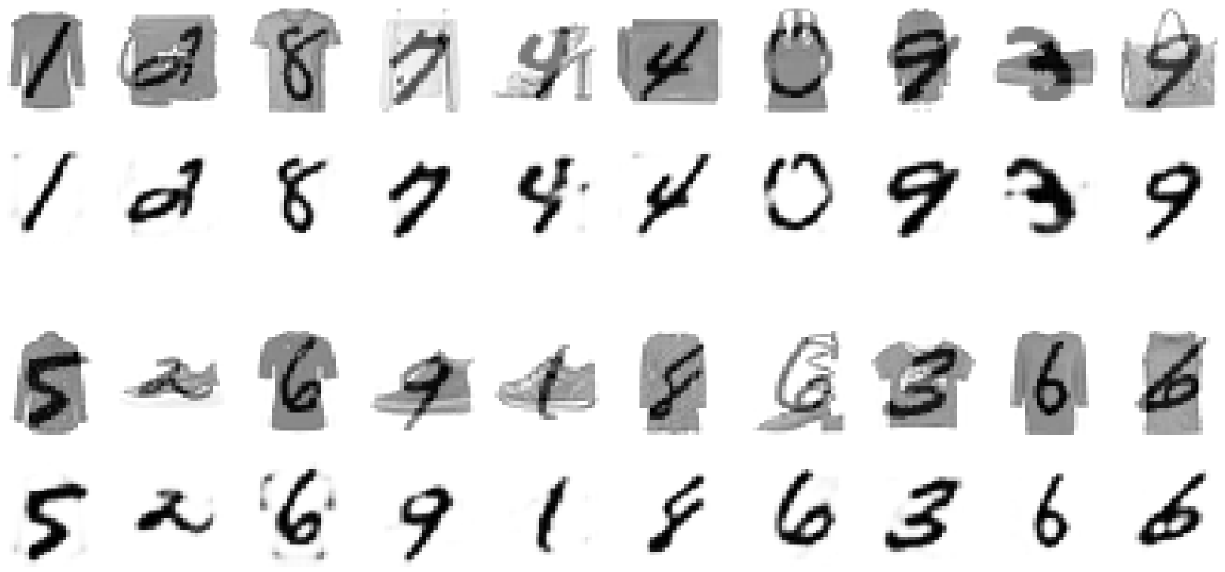


```
Epoch 10/10
55000/55000 [==============================] - 8s 153us/sample - loss: 0.0958 - binary_accuracy: 0.8130 -
val_loss: 0.0957 - val_binary_accuracy: 0.8129
```

Tam Phi
ECEC 622
Report 4

Discussion

The reconstructed images were plotted (textbook functions) to confirm that they can be identified correctly by human eyes. Top row is the original noisy image and the bottom is the denoised ones



The model could perform better with more training parameters and deeper layers in the autocoder. However, within the scope of this project, the model does reasonably well in reconstructing denoised images

Tam Phi
ECEC 622
Report 4

```
Train on 55000 samples, validate on 5000 samples

Epoch 1/10

55000/55000 [==============================] - 10s 188us/sample - loss: 0.1275 -
binary_accuracy: 0.8110 - val_loss: 0.1050 - val_binary_accuracy: 0.8117

Epoch 2/10

55000/55000 [==============================] - 9s 162us/sample - loss: 0.1036 - binary_accuracy:
0.8119 - val_loss: 0.1032 - val_binary_accuracy: 0.8127

Epoch 3/10

55000/55000 [==============================] - 8s 152us/sample - loss: 0.1021 - binary_accuracy:
0.8122 - val_loss: 0.1018 - val_binary_accuracy: 0.8120

Epoch 4/10

55000/55000 [==============================] - 8s 150us/sample - loss: 0.1009 - binary_accuracy:
0.8123 - val_loss: 0.1006 - val_binary_accuracy: 0.8124

Epoch 5/10

55000/55000 [==============================] - 8s 150us/sample - loss: 0.0999 - binary_accuracy:
0.8125 - val_loss: 0.1038 - val_binary_accuracy: 0.8103

Epoch 6/10

55000/55000 [==============================] - 8s 150us/sample - loss: 0.0990 - binary_accuracy:
0.8126 - val_loss: 0.0999 - val_binary_accuracy: 0.8136

Epoch 7/10

55000/55000 [==============================] - 8s 150us/sample - loss: 0.0982 - binary_accuracy:
0.8127 - val_loss: 0.0995 - val_binary_accuracy: 0.8138

Epoch 8/10

55000/55000 [==============================] - 8s 151us/sample - loss: 0.0974 - binary_accuracy:
0.8128 - val_loss: 0.0975 - val_binary_accuracy: 0.8135

Epoch 9/10

55000/55000 [==============================] - 8s 154us/sample - loss: 0.0966 - binary_accuracy:
0.8129 - val_loss: 0.0964 - val_binary_accuracy: 0.8128

Epoch 10/10

55000/55000 [==============================] - 8s 153us/sample - loss: 0.0958 - binary_accuracy:
0.8130 - val_loss: 0.0957 - val_binary_accuracy: 0.8129
```