Autoencoder

A neural network architecture that attempts to find the compressed representation of the given input data by uncovering the hidden structure in the input distribution. Autoencoder will learn the representation code automatically from the data alone without labeled data. Can be consider as self-supervised learning algorithm.

Denoise Autoencoder (DAE)
Input a noisy picture (mnist dataset) and output a denoised picture.

Original Picture:

In [ ]:
```
image_show = x_test[0]
image_show = image_show.reshape((28, 28))
plt.axis('off')
plt.imshow(image_show, interpolation='none', cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f909c46f898>

Noisy Picture:

In [ ]:
```
image_show = x_test_noisy[0]
image_show = image_show.reshape((28, 28))
plt.axis('off')
plt.imshow(image_show, interpolation='none', cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f909c46f0f0>

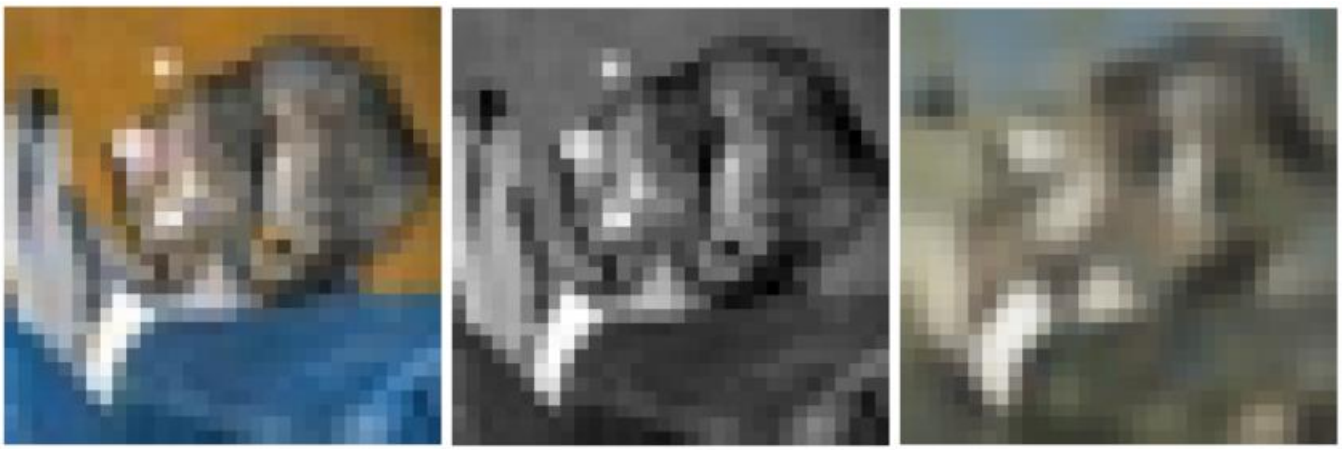
Denoised Picture:

```
In [ ]:
image_show = x_decoded[0]
image_show = image_show.reshape((28, 28))
plt.axis('off')
plt.imshow(image_show, interpolation='none', cmap='gray')
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x7f908413dd68>
```

Colorization with Autoencoder

Input a gray scaled picture and output a colored picture. This process is similar to add a good noise to the picture (reverse of the DAE)



Troubleshoot Tensorflow when training

When training with the ciffar10 dataset, the usage of the system memory (RAM) keeps increasing. Unusual behavior.
Reference forum section
https://github.com/tensorflow/tensorflow/issues/31312

According to this forum, it might be memory leak that course this problem which is the implementation defect exist in the implementation of tensorflow framwork

Solution:

Create a garbage collector callback, reset the session and set the training to be eager.

```
import gc
from tensorflow.keras import backend as k
from tensorflow.keras.callbacks import Callback


class ClearMemory(Callback):
    def on_epoch_end(self, epoch, logs=None):
        gc.collect()
        k.clear_session()

...

model.compile(
    ...,
    run_eagerly=True
)

model.fit(
    ...,
    callbacks=ClearMemory()
)
```

However, it will sacrifice the performance since the graph optimization is removed.