ITE4053 Deep Learning course
Practice 5

2016026080 Dajin Han

## # Environments

| | |
|---|---|
| OS | Window 10 edu |
| Language | Python with jupyter notebook |

## # Quick start

run src/practice_5-1_training.ipynb
run src/practice_5-2_compare_performances.ipynb

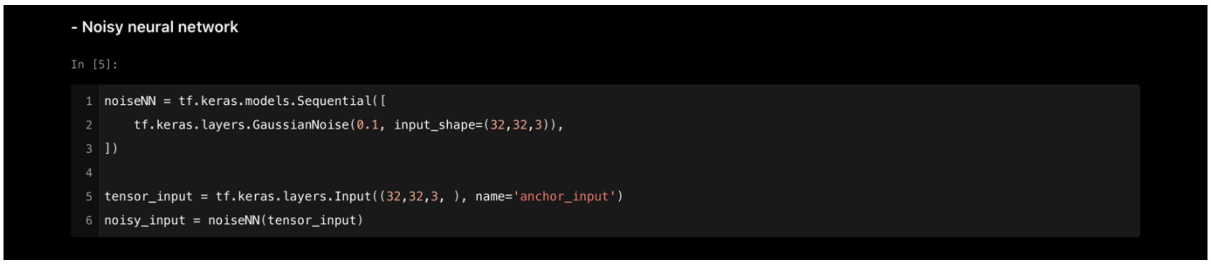
## # Weights of models trained, and the restored images

**Weights** : src/trained_models/[early-stopping option]/[learning_rate]/[# of model].hdf5
**Restored_images :** src/trained_models/[early-stopping option]/[learning_rate]/[# of model].png

## # Code Explanation

### - noisy network

Divide the single model to two separated models, one noisy network and the other the core network for restoring noisy image. Noisy network is used on model1, model2 and model3. Differences are the core part.



- Noisy neural network

In [5]:

```
noiseNN = tf.keras.models.Sequential([
    tf.keras.layers.GaussianNoise(0.1, input_shape=(32,32,3)),
])

tensor_input = tf.keras.layers.Input((32,32,3, ), name='anchor_input')
noisy_input = noiseNN(tensor_input)
```

### - Core model

core model is neural network reducing the noisy of input image. Model 1, model2 and model3 have different core model. This part makes difference on performance of recovering noisy image. Model 1 is an simple cnn model, model 2 add a skip connection on model 1 and model3 add normalization layer on model2.

**- Build model**

In [8]:

```
model_core = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3,3), input_shape=(32,32,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(3, (3,3), padding='same')
])

restored_output = model_core(noisy_input)
model = tf.keras.models.Model(inputs=tensor_input, outputs=restored_output)

model.compile(
    optimizer = tf.keras.optimizers.Adam(lr=learning_rate),
    loss = tf.keras.losses.MeanSquaredError(),
    metrics = [tf.keras.metrics.Accuracy()]
)

model.summary()
```

**- Build model**

In [11]:

```
model_core = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3,3), input_shape=(32,32,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation=tf.nn.relu),
    tf.keras.layers.Conv2D(3, (3,3), padding='same')
])

restored_output = model_core(noisy_input) + noisy_input
model = tf.keras.models.Model(inputs=tensor_input, outputs=restored_output)

model.compile(
    optimizer = tf.keras.optimizers.Adam(lr=learning_rate),
    loss = tf.keras.losses.MeanSquaredError(),
    metrics = [tf.keras.metrics.Accuracy()]
)

model.summary()
```

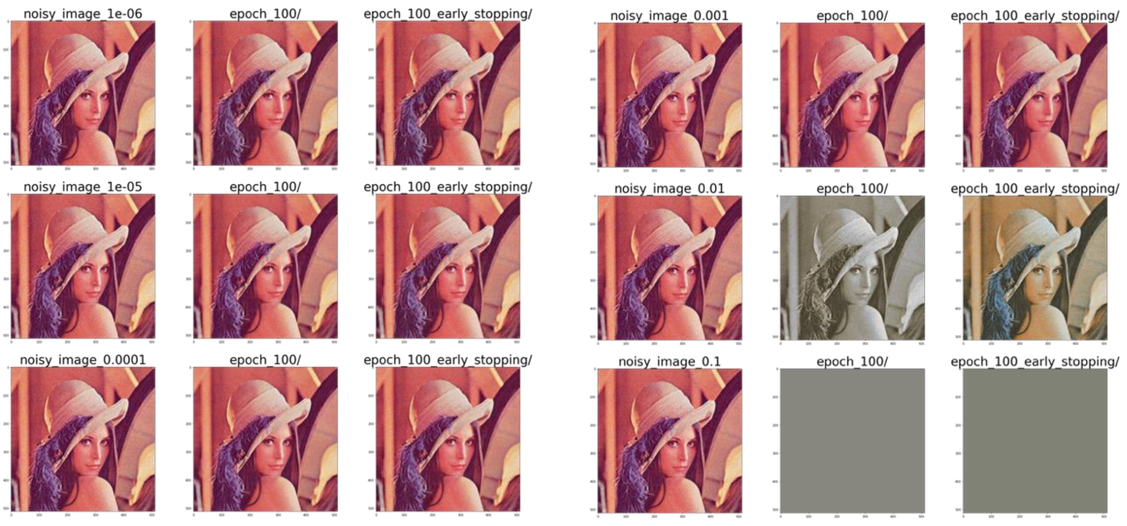
**- Build model**

In [14]:

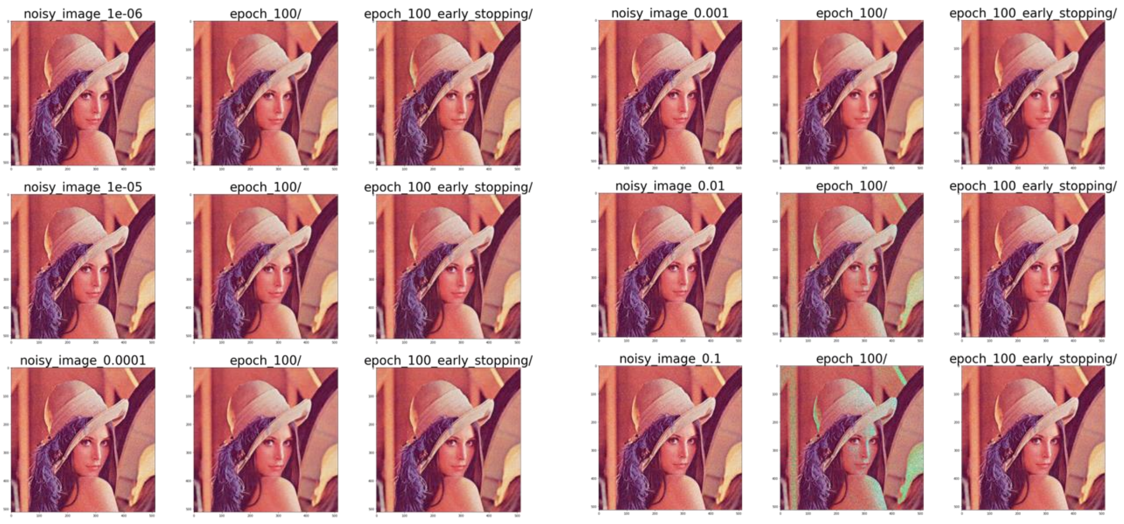
```
model_core = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3,3), input_shape=(32,32,3), padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Conv2D(64, (3,3), padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Conv2D(64, (3,3), padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Conv2D(64, (3,3), padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Activation('relu'),
    tf.keras.layers.Conv2D(3, (3,3), padding='same')
])

restored_output = model_core(noisy_input) + noisy_input
model = tf.keras.models.Model(inputs=tensor_input, outputs=restored_output)

model.compile(
    optimizer = tf.keras.optimizers.Adam(lr=learning_rate),
    loss = tf.keras.losses.MeanSquaredError(),
    metrics = [tf.keras.metrics.Accuracy()]
)

model.summary()
```

# Results

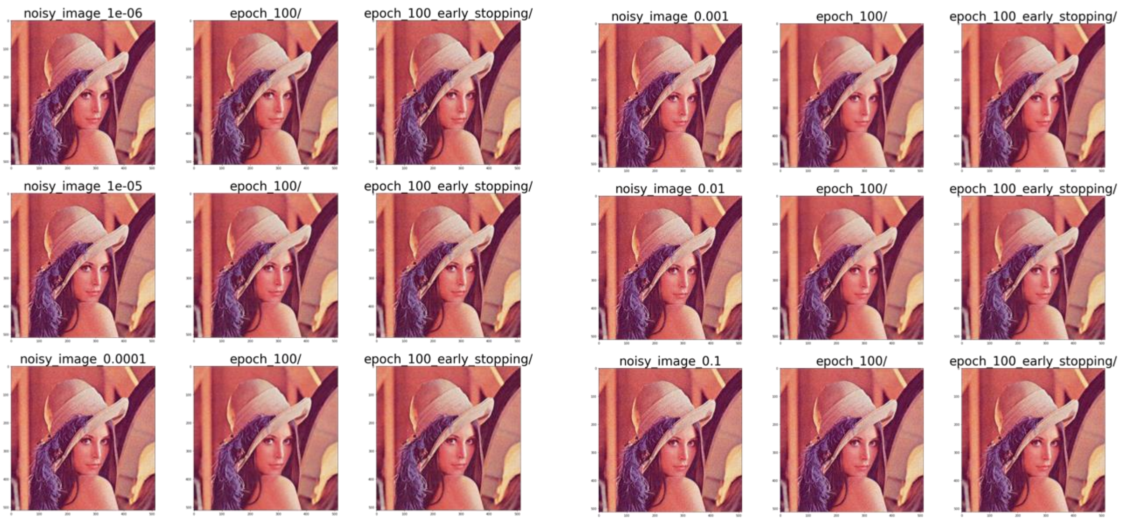
## Results of model #1



## Results of model #2



## Results of model #3



## # Best parameters and options

- **Early-stopping option : DO NOT USE IT**

  Initially, I expected that the performance of photo recovery would be better if the early-stopping option was added. Because early-stopping option prevents model from being trained excessively only about training data, and reduces the differences(error) between restored image and original image.

  But unlike my thoughts, early-stopping does not matter for reducing noisy of image. Trained model with early-stopping option often show worse performance compared to other trained model with no early-stopping option

- **Learning Rate : 1e-4**

  model1, model2 and model 3 showed the best performance in restoring noisy image when the learning rate value was 1e-4

- **Results of learning_rate, 1e-4**