In [2]:
```
from keras.layers import Dense,Conv2D,MaxPooling2D,UpSampling2D
from keras import Input, Model
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
```

In [16]:
```
encoding_dim = 1500
input_img = Input(shape=(196608,))
# encoded representation of input
encoded = Dense(encoding_dim, activation='relu')(input_img)
# decoded representation of code
decoded = Dense(196608, activation='sigmoid')(encoded)
# Model which take input image and shows decoded images
autoencoder = Model(input_img, decoded)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-16-b4bcac93aefd> in <module>()
      2 input_img = Input(shape=(196608,))
      3 # encoded representation of input
----> 4 encoded = Dense(encoding_dim, activation='relu')(input_img)
      5 # decoded representation of code
      6 decoded = Dense(196608, activation='sigmoid')(encoded)

~\anaconda3\lib\site-packages\keras\utils\tracebackutils.py in error_handler(*args, **kwargs)
     63         try:
     64             return fn(*args, **kwargs)
---> 65         except Exception as e:
     66             filtered_tb = process_traceback_frames(e.__traceback__)

~\anaconda3\lib\site-packages\keras\engine\base_layer.py in __call__(self, *args, **kwargs)
   1009                 self, inputs, args, kwargs, input_list
   1010             ):
-> 1011                 return self._functional_construction_call(
   1012                     inputs, args, kwargs, input_list
   1013                 )

~\anaconda3\lib\site-packages\keras\engine\base_layer.py in _functional_construction_call(self, inputs, args, kwargs, input_list)
   2496             # Check input assumptions set after layer building, e.g. input
   2497             # shape.
-> 2498             outputs = self._keras_tensor_symbolic_call(
   2499                 inputs, input_masks, args, kwargs
   2500             )

~\anaconda3\lib\site-packages\keras\engine\base_layer.py in _keras_tensor_symbolic_call(self, inputs, input_masks, args, kwargs)
   2343             )
   2344         else:
-> 2345             return self._infer_output_signature(
   2346                 inputs, args, kwargs, input_masks
   2347             )

~\anaconda3\lib\site-packages\keras\engine\base_layer.py in _infer_output_signature(self, inputs, args, kwargs, input_masks)
   2407                     # TODO(kaftan): do we maybe_build here, or have we already
   2408                     # done it?
-> 2409                     self._maybe_build(inputs)
   2410                     inputs = self._maybe_cast_inputs(inputs)
   2411                     outputs = call_fn(inputs, *args, **kwargs)

~\anaconda3\lib\site-packages\keras\engine\base_layer.py in _maybe_build(self, inputs)
   2941                 # operations.
   2942                 with tf_utils.maybe_init_scope(self):
-> 2943                     self.build(input_shapes)
   2944             # We must set also ensure that the layer is marked as built, and the
   2945             # build shape is stored since user defined build functions may not

~\anaconda3\lib\site-packages\keras\layers\core\dense.py in build(self, input_shape)
    152         self.input_spec = InputSpec(min_ndim=2, axes={-1: last_dim})
--> 154         self.kernel = self.add_weight(
    155             "kernel",
    156             shape=[last_dim, self.units],

~\anaconda3\lib\site-packages\keras\engine\base_layer.py in add_weight(self, name, shape, dtype, initializer, regularizer, trainable, constraint, use_resource, synchronization, aggregation, **kwargs)
    732         getter = functools.partial(getter, layout=layout)
    733
--> 734         variable = self._add_variable_with_custom_getter(
    735             name=name,
    736             shape=shape,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\trackable\base.py in _add_variable_with_custom_getter(self, name, shape, dtype, initializer, getter, overwrite, **kwargs_for_getter)
    827     # Restore-on-create is only supported when an initializer is passed.
    828     initializer = checkpoint_initializer
--> 829     new_variable = getter(
    830         name=name,
    831         shape=shape,

~\anaconda3\lib\site-packages\keras\engine\base_layer_utils.py in make_variable(name, shape, dtype, initializer, trainable, caching_device, validate_shape, constraint, use_resource, collections, synchronization, aggregation, partitioner, layout)
    132     # However, this breaks legacy (Estimator) checkpoints because
    133     # it changes variable names. Remove this when V1 is fully deprecated.
--> 134     return tf1.Variable(
    135         initial_value=init_val,
    136         name=name,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\tracebackutils.py in error_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = process_traceback_frames(e.__traceback__)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\variables.py in __call__(cls, *args, **kwargs)
    262   def __call__(cls, *args, **kwargs):
    263     if cls is VariableV1:
--> 264       return cls._variable_v1_call(*args, **kwargs)
    265     elif cls is Variable:
    266       return cls._variable_v2_call(*args, **kwargs)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\variables.py in _variable_v1_call(cls, initial_value, trainable, collections, validate_shape, caching_device, name, variable_def, dtype, expected_shape, import_scope, constraint, use_resource, synchronization, aggregation, shape)
    207     if aggregation is None:
    208       aggregation = VariableAggregation.NONE
--> 209     return previous_getter(
    210         initial_value=initial_value,
    211         trainable=trainable,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\variables.py in <lambda>(**kwargs)
    200                          shape=None):
    201     """Call on Variable class. Useful to force the signature."""
--> 202     previous_getter = lambda **kwargs: default_variable_creator(None, **kwargs)
    203     for _, getter in ops.get_default_graph()._variable_creator_stack:  # pylint: disable=protected-access
    204       previous_getter = _make_getter(getter, previous_getter)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\variable_scope.py in default_variable_creator(next_creator, **kwargs)
   2644     if use_resource:
   2645       distribute_strategy = kwargs.get("distribute_strategy", None)
-> 2646       return resource_variable_ops.ResourceVariable(
   2647           initial_value=initial_value,
   2648           trainable=trainable,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\tracebackutils.py in error_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = process_traceback_frames(e.__traceback__)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\variables.py in __call__(cls, *args, **kwargs)
    266       return cls._variable_v2_call(*args, **kwargs)
    267     else:
--> 268       return super(VariableMetaclass, cls).__call__(*args, **kwargs)
    269
    270

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\resource_variable_ops.py in __init__(self, initial_value, trainable, collections, validate_shape, caching_device, name, dtype, variable_def, import_scope, constraint, distribute_strategy, synchronization, aggregation, shape)
   1657                 validate_shape=validate_shape)
   1658     else:
-> 1659       self._init_from_args(
   1660           initial_value=initial_value,
   1661           trainable=trainable,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\resource_variable_ops.py in _init_from_args(self, initial_value, trainable, collections, caching_device, name, dtype, constraint, synchronization, aggregation, distribute_strategy, shape, validate_shape)
   1810           with ops.name_scope("Initializer"), device_context_manager(None):
   1811             if init_from_fn:
-> 1812               initial_value = initial_value()
   1813             if isinstance(initial_value, trackable.CheckpointInitialValue):
   1814               self._maybe_initialize_trackable()

~\anaconda3\lib\site-packages\keras\initializers\initializers_v2.py in __call__(self, shape, dtype, **kwargs)
    635             nonce=nonce,
    636         )
--> 637       return self._generate_init_val(shape=shape, dtype=dtype, nonce=nonce)
    638
    639     def _generate_init_val(self, shape, dtype, nonce):

~\anaconda3\lib\site-packages\keras\initializers\initializers_v2.py in _generate_init_val(self, shape, dtype, nonce)
    660         else:
    661             limit = math.sqrt(3.0 * scale)
--> 662             return self._random_generator.random_uniform(
    663                 shape, -limit, limit, dtype, nonce
    664             )

~\anaconda3\lib\site-packages\keras\backend.py in random_uniform(self, shape, minval, maxval, dtype, nonce)
   2298     if nonce:
   2299         seed = tf.random.experimental.stateless_fold_in(seed, nonce)
-> 2300     return tf.random.stateless_uniform(
   2301         shape=shape,
   2302         minval=minval,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\tracebackutils.py in error_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = process_traceback_frames(e.__traceback__)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.py in op_dispatch_handler(*args, **kwargs)
   1174       # Fallback dispatch system (dispatch v1):
   1175       try:
-> 1176         return dispatch_target(*args, **kwargs)
   1177       except (TypeError, ValueError):
   1178         # Note: convert_to_eager_tensor currently raises a ValueError, not a

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\stateless_random_ops.py in stateless_random_uniform(shape, seed, minval, maxval, dtype, name, alg)
    486     rnd = gen_stateless_random_ops_v2.stateless_random_uniform_v2(
    487         shape, key=key, counter=counter, dtype=dtype, alg=alg)
--> 488     result = math_ops.add(rnd * (maxval - minval), minval, name=name)
    489     tensor_util.maybe_set_static_shape(result, shape)
    490     return result

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\tracebackutils.py in error_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = process_traceback_frames(e.__traceback__)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.py in op_dispatch_handler(*args, **kwargs)
   1174       # Fallback dispatch system (dispatch v1):
   1175       try:
-> 1176         return dispatch_target(*args, **kwargs)
   1177       except (TypeError, ValueError):
   1178         # Note: convert_to_eager_tensor currently raises a ValueError, not a

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\math_ops.py in multiply(x, y, name)
    521   """
    522
--> 523   return gen_math_ops.mul(x, y, name)
    524
    525

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\gen_math_ops.py in mul(x, y, name)
   6573   if tld.is_eager:
   6574     try:
-> 6575       result = pywrap_tfe.TFE_Py_FastPathExecute(
   6576         _ctx, "Mul", name, x, y)
   6577       return _result

KeyboardInterrupt:
```

In [4]:
```
# This model shows encoded images
encoder = Model(input_img, encoded)
# Creating a decoder model
encoded_input = Input(shape=(encoding_dim,))
# last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# decoder model
decoder = Model(encoded_input, decoder_layer(encoded_input))
```

In [5]:
```
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

In [6]:
```
import cv2

im = cv2.imread('lena_color_256.tif')
print(im.shape)
```

```
(256, 256, 3)
```

In [7]:
```
autoencoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 196608)]          0

 dense (Dense)               (None, 1500)              294913500

 dense_1 (Dense)             (None, 196608)            295108608

=================================================================
Total params: 590,022,108
Trainable params: 590,022,108
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
im=im.reshape(1,196608))
```

In [9]:
```
encoded_img = encoder.predict(im)
```

```
1/1 [==============================] - 0s 305ms/step
```

In [10]:
```
encoded_img.shape
```

Out[10]:
```
(1, 1500)
```

In [11]:
```
decoded_img = decoder.predict(encoded_img)
```

```
1/1 [==============================] - 0s 229ms/step
```

In [12]:
```
decoded_img.shape
```

Out[12]:
```
(1, 196608)
```

In [13]:
```
import matplotlib.pyplot as plt
```

In [15]:
```
plt.imshow(decoded_img.reshape(256,256,3))
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x2537d0c6a0>
```


In [ ]:
```
cv2.imshow('win',decoded_img)
```

In [ ]:
```
cv2.namedWindow(winname='my')
while True:
    cv2.imshow('my',decoded_img.reshape(256,256,3))
    if cv2.waitKey(1) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```

In [ ]:
```

```