image_denoising_cnn

March 3, 2020

In [2]:
```
import tensorflow as tf
import os
import numpy as np
import pandas as pd
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.utils import plot_model
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D,Flatten,Dense,
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.7)
sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
```

0.0.1 DATASET = http://www.fit.vutbr.cz/~vasicek/imagedb/

In [3]:
```
orig = []
noisy = []
dir_path = "noisyimg"

count = 1
for i in os.listdir(dir_path):
    p1 = os.path.join(dir_path,i)
    for j in (os.listdir(p1)):
        if j == '.ipynb_checkpoints' or i == '.ipynb_checkpoints':
            continue
        if p1 == 'noisyimg/original':
            orig.append(os.path.join(os.path.join(dir_path,i),j))
        else:
            noisy.append(os.path.join(os.path.join(dir_path,i),j))

len(orig)
```

Out[3]: 24

In [19]:
```
noisy = np.sort(noisy)
 orig = np.sort(orig)
```

```
        original = []
        for i in range(15):
            for j in orig:
                original.append(j)
        original = np.array(original)
```

```
In [20]: noisy = noisy[::-1]
         original = original[::-1]
```

```
In [21]: len(noisy)
```

```
Out[21]: 360
```

```
In [23]: count = 0
         for i in range(len(noisy)):
             im = noisy[i].split('/')
             img = original[i].split('/')
             if(im[2] == img[2]):
                 count += 1
         print(count)
```

```
360
```

```
In [9]: import cv2
        data = []

        for i in noisy:
            img = cv2.imread(i,0).reshape(256,384,1)
            img[img == 255] = 0
            data.append(img)
        data = np.array(data)

        label = []
        for i in original:
            img = cv2.imread(i,0).reshape(256,384,1)
            label.append(img)
        label = np.array(label)
```

```
In [10]: ytrain = label[0:300]
         xtrain = data[0:300]
         ytest = label[300:360]
         xtest = data[300:360]
```

```
In [11]: from sklearn.metrics import roc_auc_score, f1_score

         class loss(tf.keras.callbacks.Callback):
             def __init__(self,tr_data,tr_label, decay):
                 self.x = tr_data
```

```
            self.y = tr_label
            self.decay = decay
            self.plr = -1

        def on_epoch_end(self, epoch, logs={}):
            y_pred = self.model.predict(self.x)
            roc = roc_auc_score(self.y, y_pred, average='macro')
            y_pred = convert(y_pred)
            f1 = f1_score(self.y, y_pred, average = 'macro')
            print('roc-score:', str(round(roc,4)),'f1-score:', str(round(f1,4)))

            modelWeights = []
            for layer in self.model.layers:
                layerWeights = []
                for weight in layer.get_weights():

                    print("in weights", weight.shape, layer.name)
                    if np.sum(np.isnan(weight)) != 0:
                        print("stopping train as we have na values in weights")
                        self.model.stop_training = True
                        print("stopping model")
            if np.isnan(logs['loss']):
                print("stopping train as we have na values in loss")
                self.model.stop_training = True
                print("stopping model")
            print('-'*30, "epoch", epoch, "is done", "-"*30)

            old_lr = tf.keras.backend.get_value(self.model.optimizer.lr)
            if logs['val_acc'] < self.plr:
                nlr = old_lr*0.9
                tf.keras.backend.set_value(self.model.optimizer.lr, nlr)
            self.plr = logs['val_acc']

        def on_epoch_begin(self,epoch,logs = {}):

            old_lr = tf.keras.backend.get_value(self.model.optimizer.lr)
            print("learning rate at this epoch: ",old_lr)
            if epoch > 1 and epoch%3 == 0 :
                new_lr = 0.95*old_lr
                tf.keras.backend.set_value(self.model.optimizer.lr, new_lr)
                print("modifying learning rate by a factor of 0.95, in next epoch your lea
            else:
                tf.keras.backend.set_value(self.model.optimizer.lr, old_lr)
```

In [12]:
```
import os
import random as rn
os.environ['PYTHONHASHSEED'] = '0'
```

```
##https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-k
## Have to clear the session. If you are not clearing, Graph will create again and ag
## Varibles will also set to some value from before session
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
#tf.random.set_seed(0)
tf.set_random_seed(0)
rn.seed(0)


input_layer = Input(shape=(256, 384,1))
Conv1 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation='relu',kernel_initializer=tf.keras.initializers.he_normal(se
Conv2 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
batch_norm = tf.keras.layers.BatchNormalization()(Conv2)
relu_act1 = tf.keras.layers.ReLU()(batch_norm)

Conv3 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv3)
relu_act2 = tf.keras.layers.ReLU()(batch_norm)

Conv4 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv4)
relu_act3 = tf.keras.layers.ReLU()(batch_norm)

Conv5 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
batch_norm = tf.keras.layers.BatchNormalization()(Conv5)
relu_act4 = tf.keras.layers.ReLU()(batch_norm)

Conv6 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv6)
relu_act5 = tf.keras.layers.ReLU()(batch_norm)

Conv7 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
               activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv7)
relu_act6 = tf.keras.layers.ReLU()(batch_norm)
```

```
Conv8 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
batch_norm = tf.keras.layers.BatchNormalization()(Conv8)
relu_act7 = tf.keras.layers.ReLU()(batch_norm)

Conv9 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
               kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv9)
relu_act8 = tf.keras.layers.ReLU()(batch_norm)

Conv10 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv10)
relu_act9 = tf.keras.layers.ReLU()(batch_norm)

Conv11 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
batch_norm = tf.keras.layers.BatchNormalization()(Conv11)
relu_act10 = tf.keras.layers.ReLU()(batch_norm)

Conv12 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv12)
relu_act11 = tf.keras.layers.ReLU()(batch_norm)

Conv13 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
batch_norm = tf.keras.layers.BatchNormalization()(Conv13)
relu_act12 = tf.keras.layers.ReLU()(batch_norm)

Conv14 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
batch_norm = tf.keras.layers.BatchNormalization()(Conv14)
relu_act13 = tf.keras.layers.ReLU()(batch_norm)

Conv15 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
batch_norm = tf.keras.layers.BatchNormalization()(Conv15)
relu_act14 = tf.keras.layers.ReLU()(batch_norm)

Conv16 = Conv2D(filters=64,kernel_size=3,strides=1,padding='same',
              activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=
```

```
batch_norm = tf.keras.layers.BatchNormalization()(Conv16)
relu_act15 = tf.keras.layers.ReLU()(batch_norm)

Conv17 = Conv2D(filters=1,kernel_size=3,strides=1,padding='same',
        activation=None,kernel_initializer=tf.keras.initializers.he_normal(seed=3),
                kernel_regularizer= tf.contrib.layers.l2_regularizer(scale=0.01),name=


model6=Model(input_layer,Conv17)
model6.summary()
```

```
WARNING:tensorflow:From /home/18mcmi04/anaconda3/lib/python3.7/site-packages/tensorflow/python/
Instructions for updating:
Colocations handled automatically by placer.

WARNING: The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
If you depend on functionality not listed there, please file an issue.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 256, 384, 1)       0
_________________________________________________________________
Conv1 (Conv2D)               (None, 256, 384, 64)      640
_________________________________________________________________
Conv2 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1 (Batc (None, 256, 384, 64)      256
_________________________________________________________________
re_lu (ReLU)                 (None, 256, 384, 64)      0
_________________________________________________________________
Conv3 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_1 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_1 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv4 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_2 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_2 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv5 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
```

```
batch_normalization_v1_3 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_3 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv6 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_4 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_4 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv7 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_5 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_5 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv8 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_6 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_6 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv9 (Conv2D)               (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_7 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_7 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv10 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_8 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_8 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv11 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_9 (Ba (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_9 (ReLU)               (None, 256, 384, 64)      0
_________________________________________________________________
Conv12 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_10 (B (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_10 (ReLU)              (None, 256, 384, 64)      0
_________________________________________________________________
Conv13 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
```

```
batch_normalization_v1_11 (B (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_11 (ReLU)              (None, 256, 384, 64)      0
_________________________________________________________________
Conv14 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_12 (B (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_12 (ReLU)              (None, 256, 384, 64)      0
_________________________________________________________________
Conv15 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_13 (B (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_13 (ReLU)              (None, 256, 384, 64)      0
_________________________________________________________________
Conv16 (Conv2D)              (None, 256, 384, 64)      36928
_________________________________________________________________
batch_normalization_v1_14 (B (None, 256, 384, 64)      256
_________________________________________________________________
re_lu_14 (ReLU)              (None, 256, 384, 64)      0
_________________________________________________________________
Conv17 (Conv2D)              (None, 256, 384, 1)       577
=================================================================
Total params: 558,977
Trainable params: 557,057
Non-trainable params: 1,920
_________________________________________________________________
```

```
In [26]: checkpoint_path = 'denoise.h5'
         checkp = tf.keras.callbacks.ModelCheckpoint(checkpoint_path, verbose=1, save_best_only

         model6.compile(optimizer=tf.keras.optimizers.Adam(0.001),loss='mean_squared_error')
         model6.fit(xtrain,ytrain,epochs=20,validation_data=(xtest,ytest),batch_size=10,callba
```

```
Train on 300 samples, validate on 60 samples
Epoch 1/20
290/300 [============================>.] - ETA: 1s - loss: 229.8738
Epoch 00001: val_loss improved from inf to 382.60512, saving model to denoise.h5
300/300 [==============================] - 40s 134ms/sample - loss: 229.1760 - val_loss: 382.6
Epoch 2/20
290/300 [============================>.] - ETA: 1s - loss: 237.8895
Epoch 00002: val_loss did not improve from 382.60512
300/300 [==============================] - 35s 116ms/sample - loss: 235.6193 - val_loss: 2939.5
Epoch 3/20
290/300 [============================>.] - ETA: 1s - loss: 196.4048
Epoch 00003: val_loss did not improve from 382.60512
```

```
300/300 [==============================] - 35s 117ms/sample - loss: 195.3289 - val_loss: 961.20
Epoch 4/20
290/300 [============================>.] - ETA: 1s - loss: 194.4583
Epoch 00004: val_loss did not improve from 382.60512
300/300 [==============================] - 35s 116ms/sample - loss: 197.1294 - val_loss: 592.73
Epoch 5/20
290/300 [============================>.] - ETA: 1s - loss: 199.6796
Epoch 00005: val_loss did not improve from 382.60512
300/300 [==============================] - 35s 116ms/sample - loss: 203.9191 - val_loss: 528.47
Epoch 6/20
290/300 [============================>.] - ETA: 1s - loss: 200.5426
Epoch 00006: val_loss did not improve from 382.60512
300/300 [==============================] - 35s 116ms/sample - loss: 197.3376 - val_loss: 461.74
Epoch 7/20
290/300 [============================>.] - ETA: 1s - loss: 182.6041
Epoch 00007: val_loss improved from 382.60512 to 225.27973, saving model to denoise.h5
300/300 [==============================] - 35s 118ms/sample - loss: 180.4551 - val_loss: 225.27
Epoch 8/20
290/300 [============================>.] - ETA: 1s - loss: 194.0120
Epoch 00008: val_loss improved from 225.27973 to 146.94813, saving model to denoise.h5
300/300 [==============================] - 35s 117ms/sample - loss: 194.4750 - val_loss: 146.94
Epoch 9/20
290/300 [============================>.] - ETA: 1s - loss: 237.2906
Epoch 00009: val_loss did not improve from 146.94813
300/300 [==============================] - 35s 116ms/sample - loss: 232.5305 - val_loss: 501.51
Epoch 10/20
290/300 [============================>.] - ETA: 1s - loss: 178.8209
Epoch 00010: val_loss did not improve from 146.94813
300/300 [==============================] - 35s 117ms/sample - loss: 175.8446 - val_loss: 302.88
Epoch 11/20
290/300 [============================>.] - ETA: 1s - loss: 209.2941
Epoch 00011: val_loss did not improve from 146.94813
300/300 [==============================] - 35s 117ms/sample - loss: 208.2327 - val_loss: 260.83
Epoch 12/20
290/300 [============================>.] - ETA: 1s - loss: 184.3874
Epoch 00012: val_loss did not improve from 146.94813
300/300 [==============================] - 35s 117ms/sample - loss: 183.5529 - val_loss: 373.40
Epoch 13/20
290/300 [============================>.] - ETA: 1s - loss: 179.5632
Epoch 00013: val_loss improved from 146.94813 to 86.39509, saving model to denoise.h5
300/300 [==============================] - 35s 117ms/sample - loss: 178.6827 - val_loss: 86.395
Epoch 14/20
290/300 [============================>.] - ETA: 1s - loss: 177.6372
Epoch 00014: val_loss did not improve from 86.39509
300/300 [==============================] - 35s 117ms/sample - loss: 175.4083 - val_loss: 697.56
Epoch 15/20
290/300 [============================>.] - ETA: 1s - loss: 198.9206
Epoch 00015: val_loss did not improve from 86.39509
```

```
300/300 [==============================] - 35s 117ms/sample - loss: 197.1112 - val_loss: 317.98
Epoch 16/20
290/300 [============================>.] - ETA: 1s - loss: 185.0622
Epoch 00016: val_loss did not improve from 86.39509
300/300 [==============================] - 35s 116ms/sample - loss: 184.2561 - val_loss: 146.81
Epoch 17/20
290/300 [============================>.] - ETA: 1s - loss: 209.5901
Epoch 00017: val_loss did not improve from 86.39509
300/300 [==============================] - 35s 116ms/sample - loss: 207.1866 - val_loss: 217.15
Epoch 18/20
290/300 [============================>.] - ETA: 1s - loss: 161.8909
Epoch 00018: val_loss did not improve from 86.39509
300/300 [==============================] - 35s 117ms/sample - loss: 164.7598 - val_loss: 259.95
Epoch 19/20
290/300 [============================>.] - ETA: 1s - loss: 170.2645
Epoch 00019: val_loss did not improve from 86.39509
300/300 [==============================] - 35s 117ms/sample - loss: 171.5929 - val_loss: 142.53
Epoch 20/20
290/300 [============================>.] - ETA: 1s - loss: 160.5561
Epoch 00020: val_loss did not improve from 86.39509
300/300 [==============================] - 35s 116ms/sample - loss: 162.2316 - val_loss: 167.39
```

Out[26]: <tensorflow.python.keras.callbacks.History at 0x7f21ab098860>

In [52]:
```
model6.load_weights('denoise.h5')
```

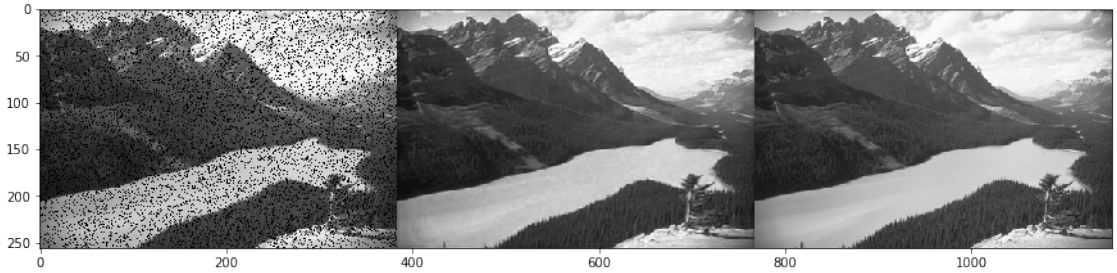
In [53]:
```
from matplotlib import pyplot as plt
import cv2
import math

def psnr(img1, img2):
    mse = np.mean( (img1 - img2) ** 2 )
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))
```

In [55]:
```
for i in range(len(xtest)):
    tmp = np.expand_dims(xtest[i], axis=0)
    pred = model6.predict(tmp).reshape(256,384)
    stacked1 = np.hstack((xtest[i].reshape(256,384),pred,ytest[i].reshape(256,384)))
    print("PSNR: ",psnr(ytest[i].reshape(256,384),pred),"\n")
    plt.figure(figsize=(15,6))
    print('\tnoisy image\t\t\t\tpredicted\t\t\t\toriginal')
    plt.imshow(stacked1,cmap='gray',vmin=0,vmax=255)
    plt.show()
```

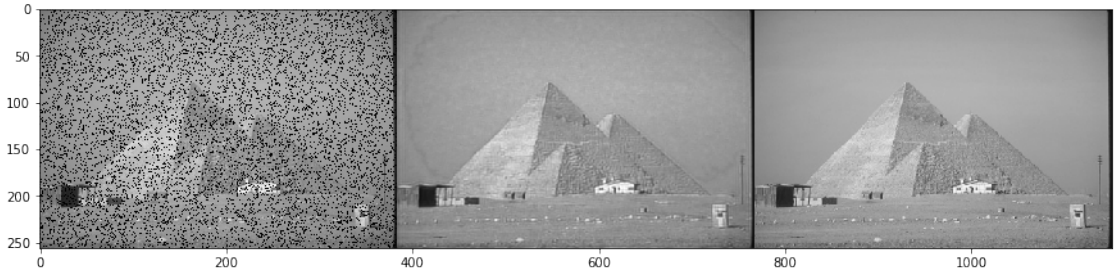
PSNR: 29.57968681574071

noisy image predicted ori



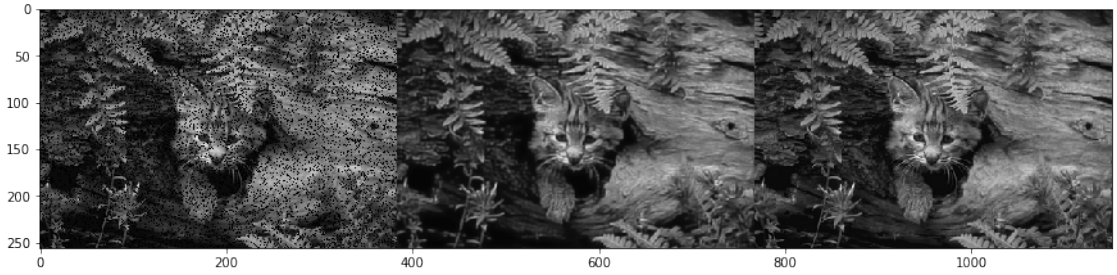
PSNR: 30.901232569873237

noisy image predicted ori



PSNR: 28.598410931055334

noisy image predicted ori

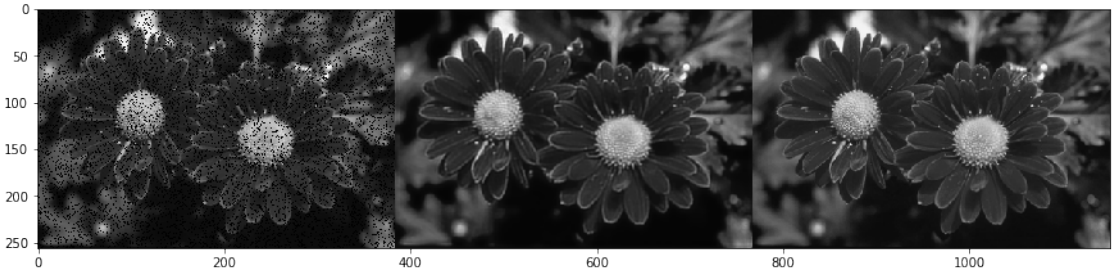


PSNR: 27.651684501289758

noisy image predicted ori

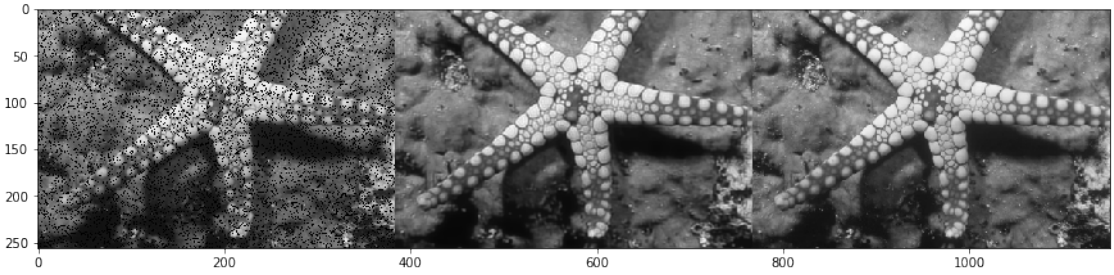
PSNR: 29.607581269391293

noisy image predicted ori



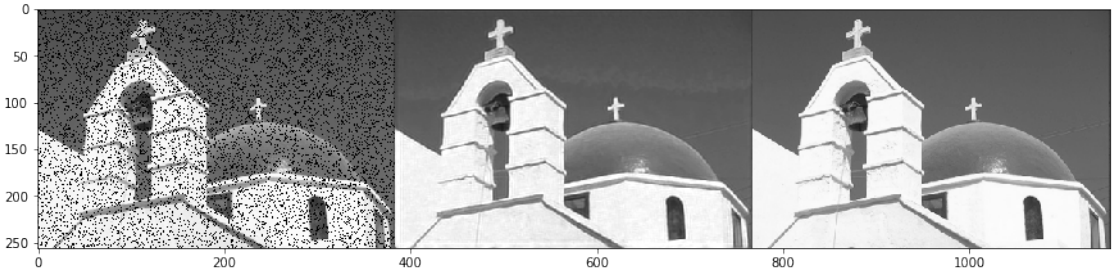
PSNR: 29.869868650366836

noisy image predicted ori



PSNR: 30.841802728761365

noisy image predicted ori

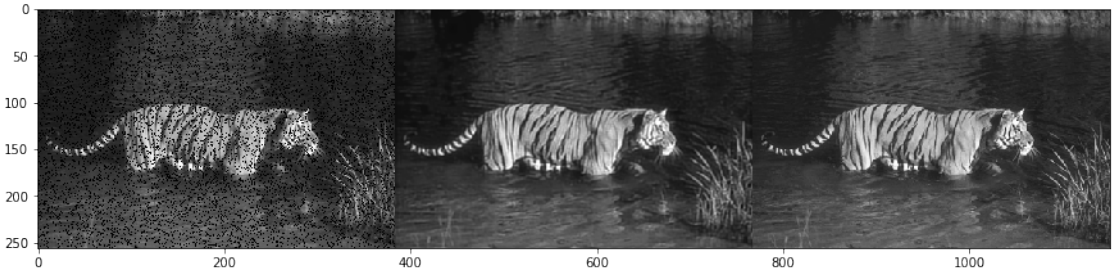


PSNR: 28.07802467470688

noisy image predicted ori

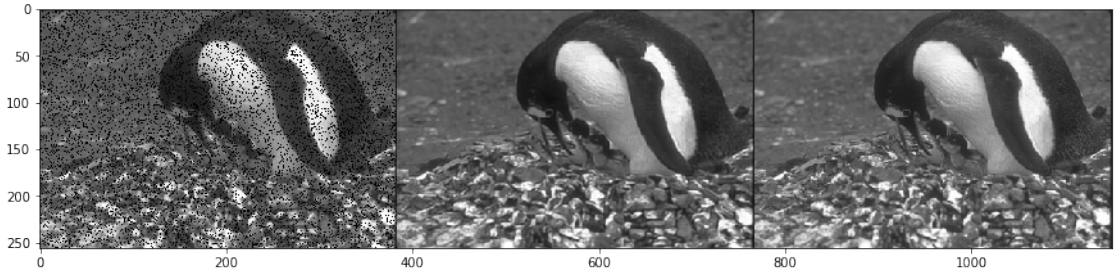
PSNR: 28.091999592224166

noisy image predicted ori



PSNR: 30.210846165303625

noisy image predicted ori

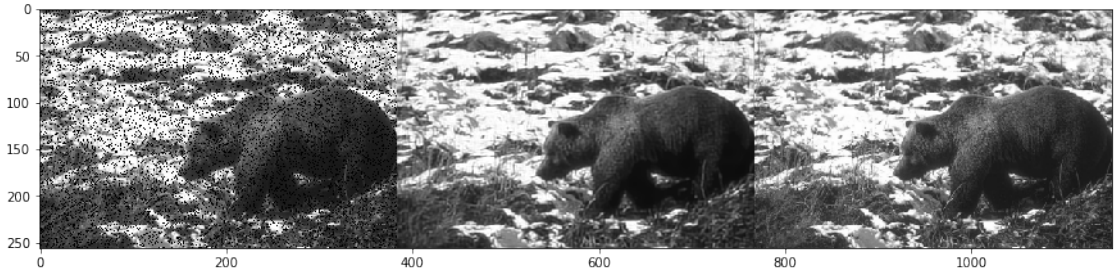


PSNR: 32.95923319101156

noisy image predicted ori

PSNR: 26.824950919465493

noisy image predicted ori



PSNR: 29.33642569408815

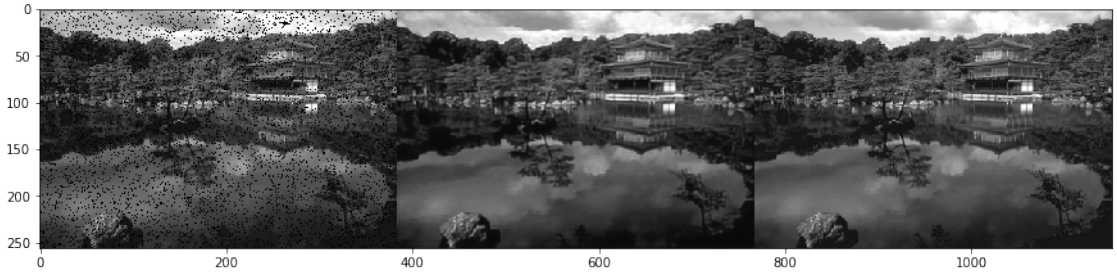
noisy image predicted ori

PSNR: 28.51145540502026

noisy image predicted ori

PSNR: 28.15759609495028

noisy image predicted ori



PSNR: 27.867297179430103

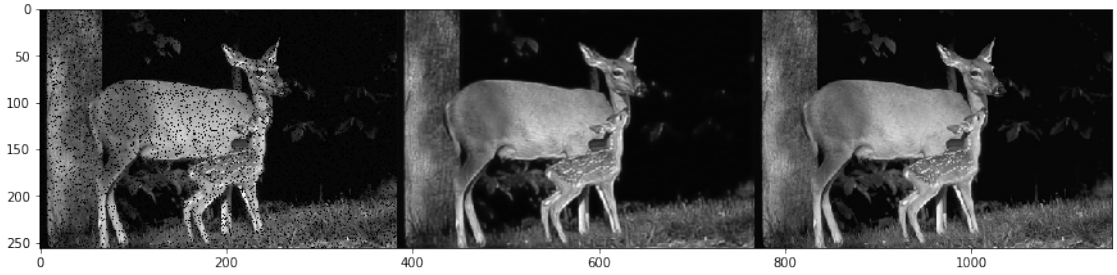
noisy image predicted ori

PSNR: 27.857574789294514

noisy image predicted ori

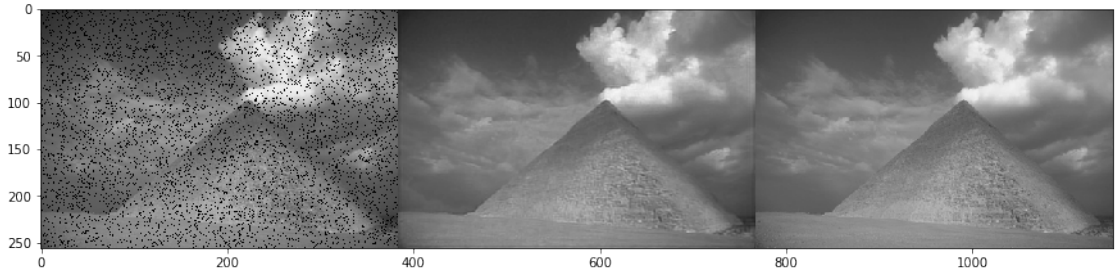
PSNR: 29.94187696063571

noisy image predicted ori



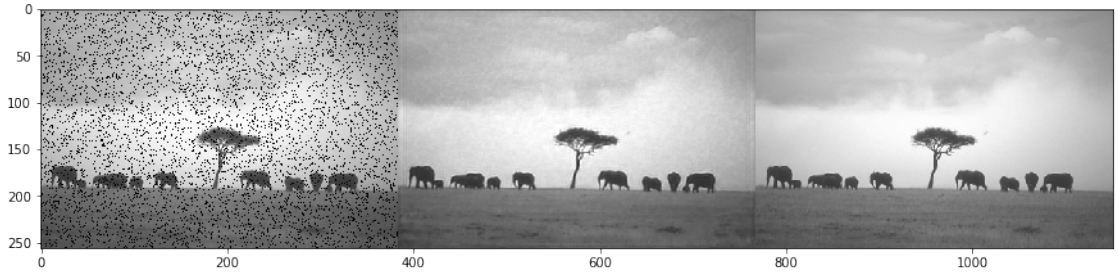
PSNR: 33.331701592340536

noisy image predicted ori



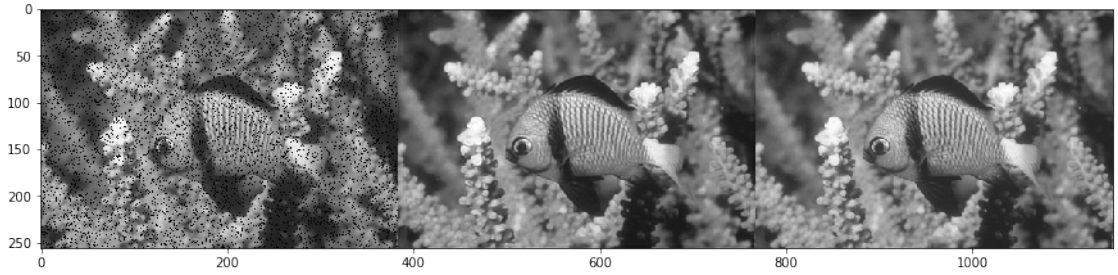
PSNR: 33.6266404596378

noisy image predicted ori



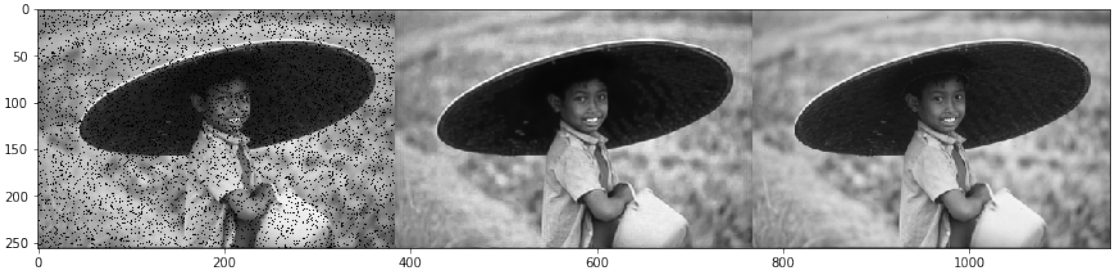
PSNR: 30.97912415672715

noisy image predicted ori



PSNR: 30.373673177759027

noisy image predicted ori

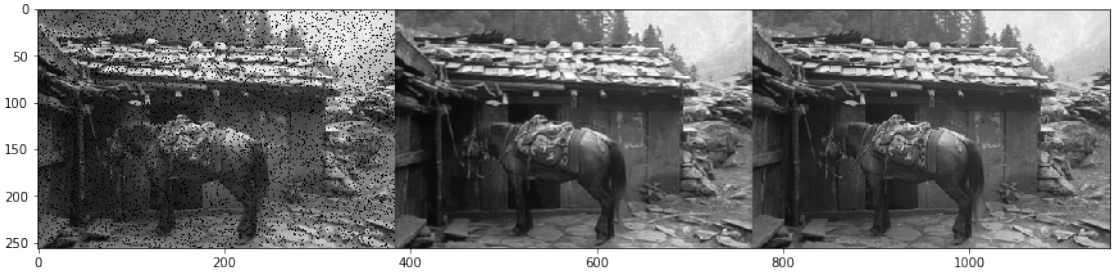


PSNR: 27.666764393404236

noisy image predicted ori

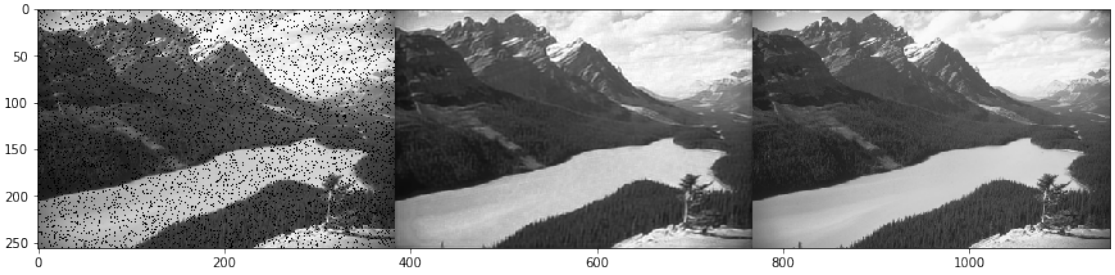
PSNR: 27.90265648127597

noisy image predicted ori



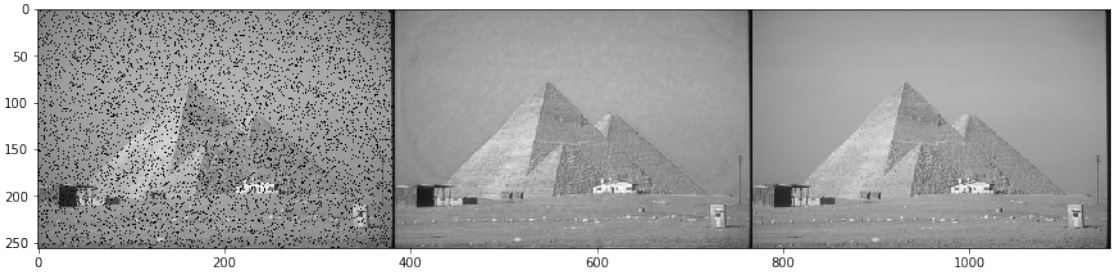
```
PSNR:  29.6077335019389
```

```
       noisy image                                    predicted                                    ori
```



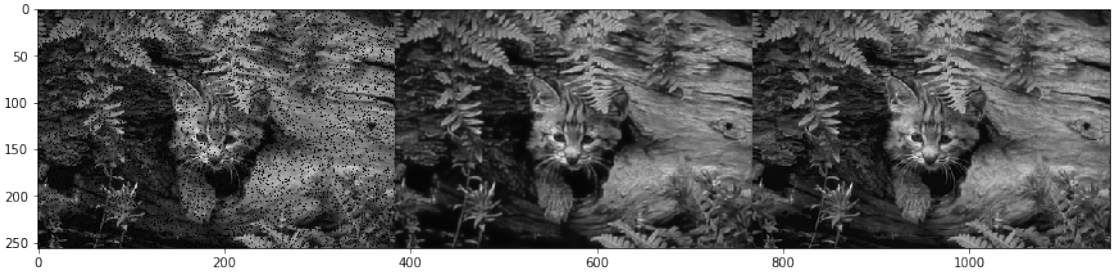

```
PSNR:  31.598640182871076
```

```
       noisy image                                    predicted                                    ori
```




```
PSNR:  28.597017044674892
```

```
       noisy image                                    predicted                                    ori
```



PSNR: 27.633916462401512

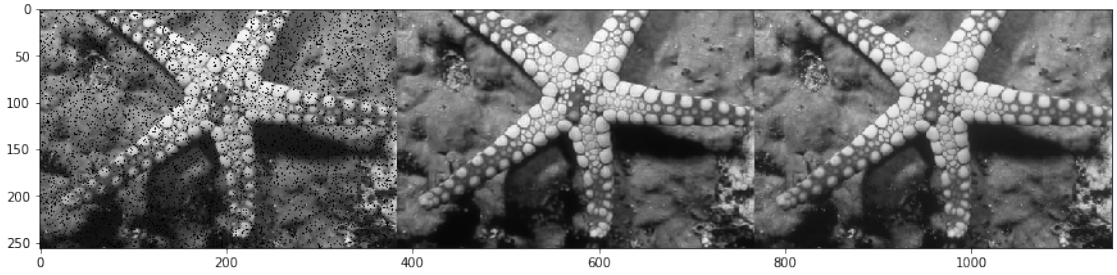
noisy image predicted ori

PSNR: 29.24656121238661

noisy image predicted ori

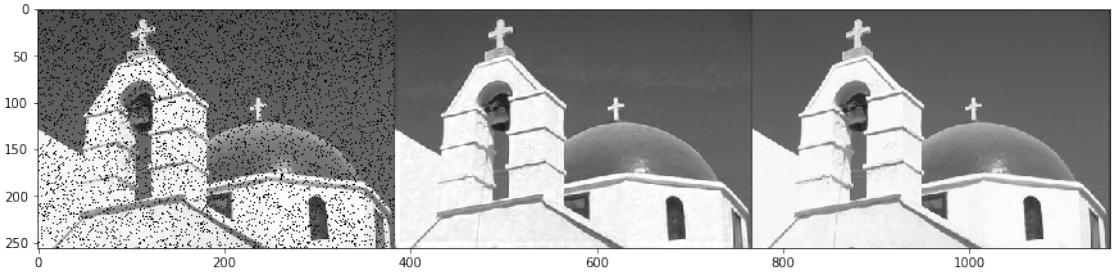
PSNR: 29.757644296322002

noisy image predicted ori



PSNR: 30.962733369275686

noisy image predicted ori

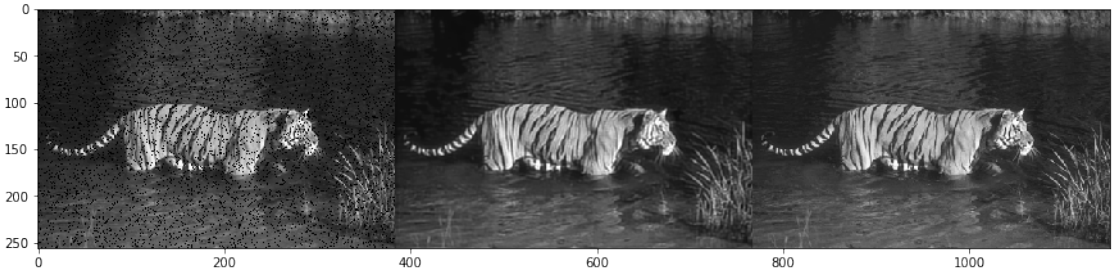


PSNR: 28.111410214862424

noisy image predicted ori

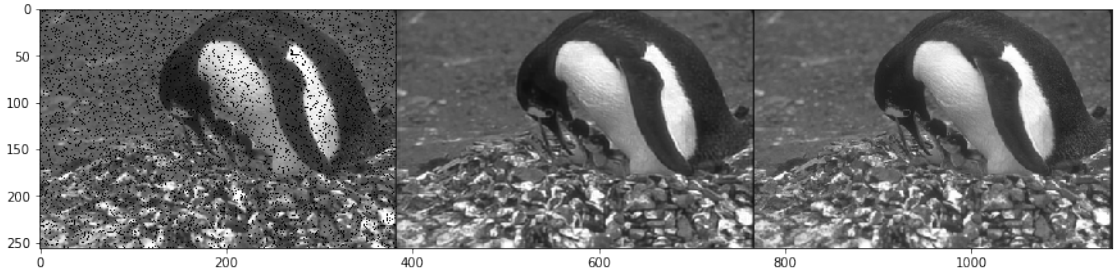
PSNR: 27.827911723049155

noisy image predicted ori



PSNR: 30.195001191936637

noisy image predicted ori

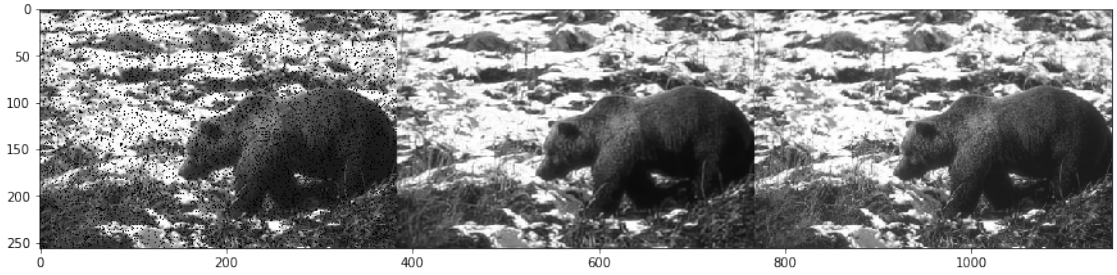


PSNR: 33.304886224728534

noisy image predicted ori

PSNR: 26.777060560280844

noisy image predicted ori



PSNR: 29.439524884673176

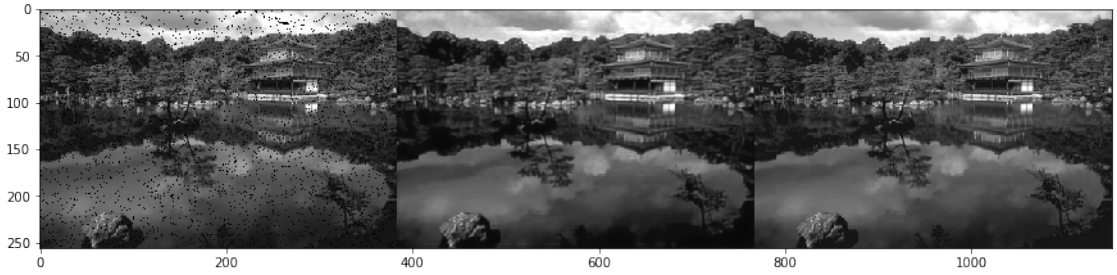
noisy image predicted ori

PSNR: 28.758142687054487

noisy image predicted ori

PSNR: 27.953535448441038

noisy image predicted ori



PSNR: 27.883117242192235

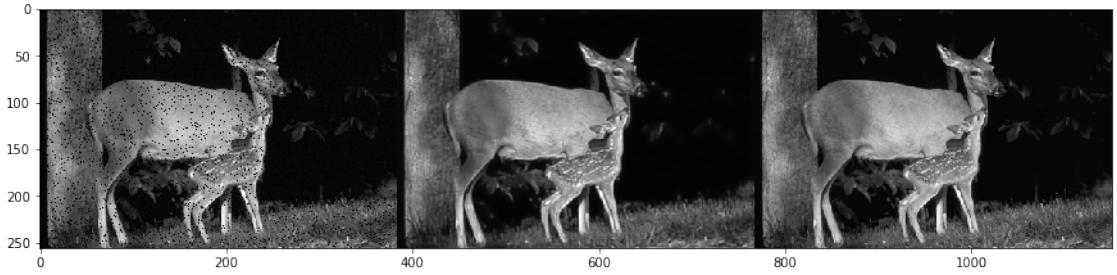
noisy image predicted ori

PSNR: 27.84097969186041

noisy image predicted ori

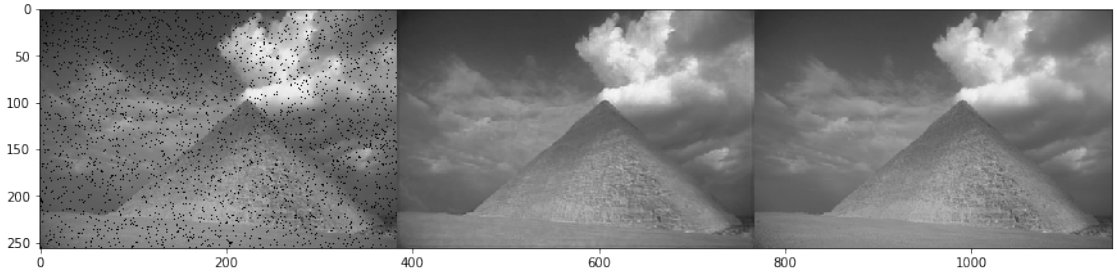
PSNR: 30.022616328285036

noisy image predicted ori



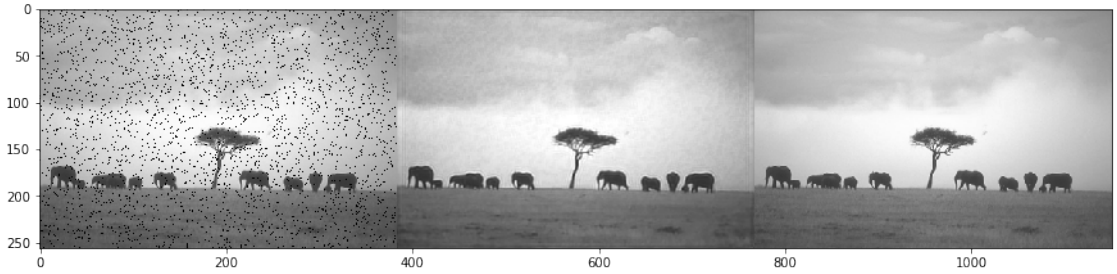
PSNR: 33.907731000241455

noisy image predicted ori



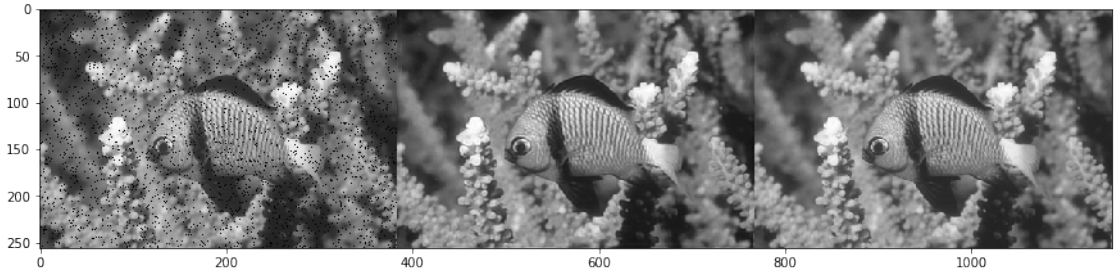
PSNR: 34.122420359037655

noisy image predicted ori



PSNR: 31.050853997970673

noisy image predicted ori



PSNR: 30.04925281399033

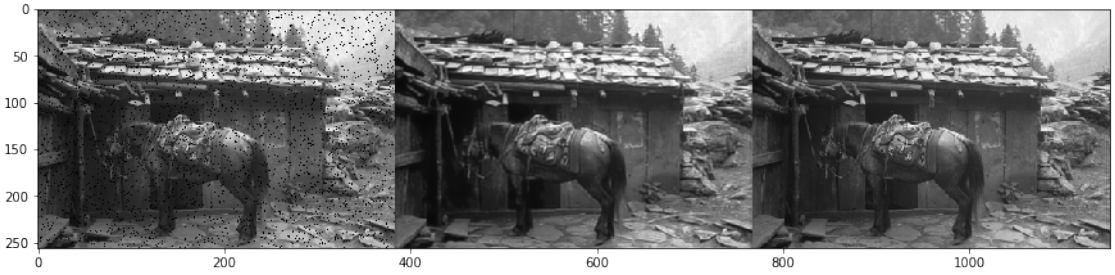
noisy image predicted ori

PSNR: 27.581622472201296

noisy image predicted ori

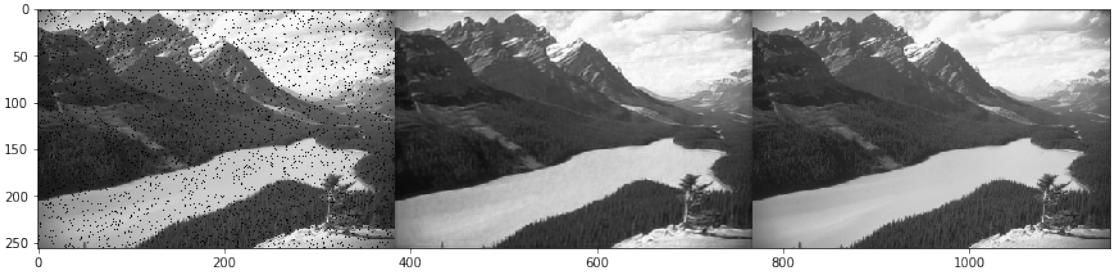
PSNR: 27.61100158822595

noisy image predicted ori



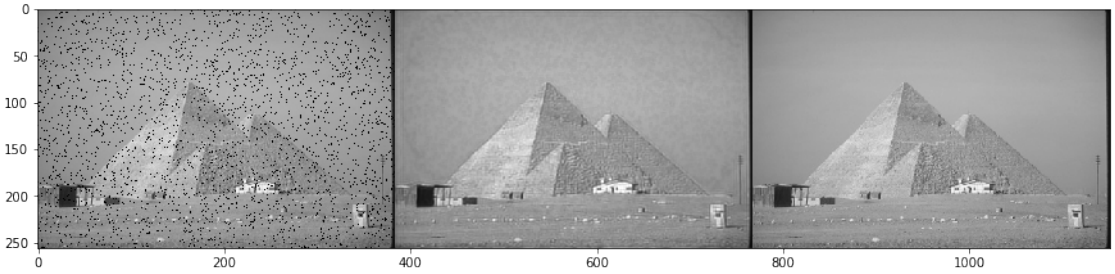
PSNR: 29.55767791336962

noisy image predicted ori



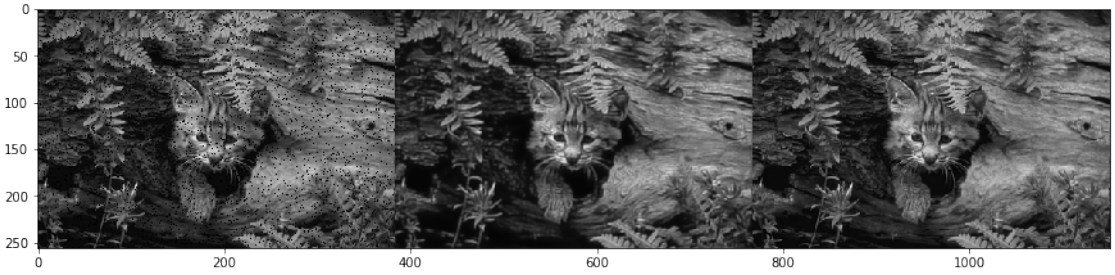
PSNR: 32.31776700427186

noisy image predicted ori



PSNR: 28.52670288555683

noisy image predicted ori



PSNR: 27.492655337568475

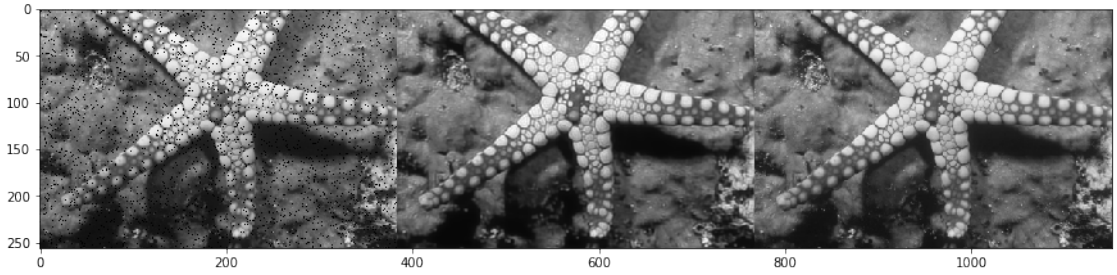
noisy image predicted ori

PSNR: 28.68822945636329

noisy image predicted ori

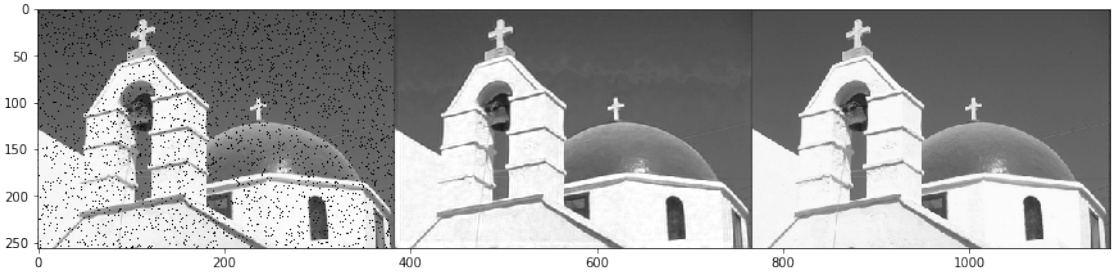
PSNR: 29.54457280283386

noisy image predicted ori



PSNR: 30.94898828054912

noisy image predicted ori

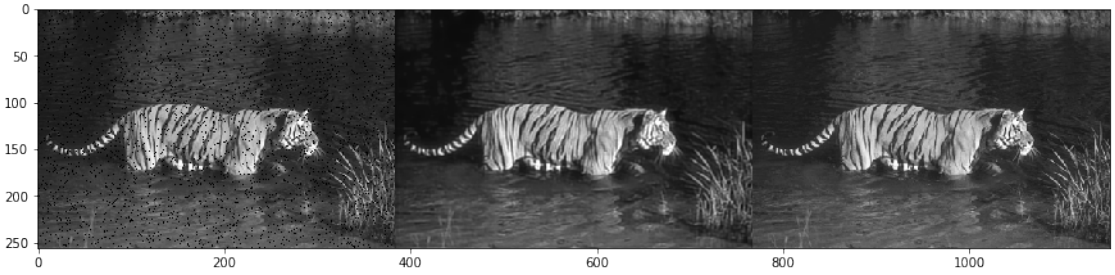


PSNR: 28.094179355564833

noisy image predicted ori

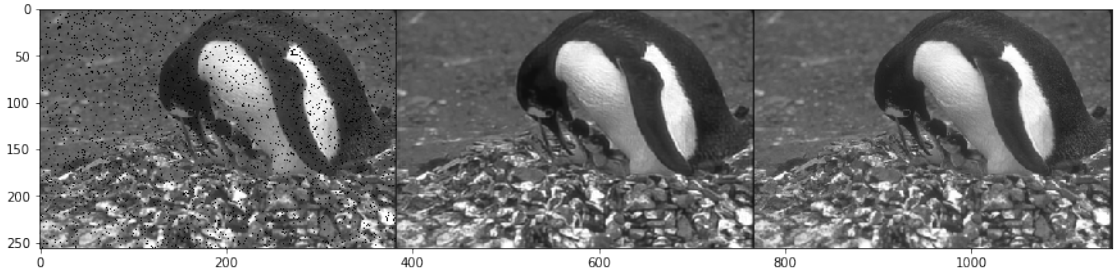
PSNR: 27.45941663420069

noisy image predicted ori



PSNR: 30.15168632228001

noisy image predicted ori

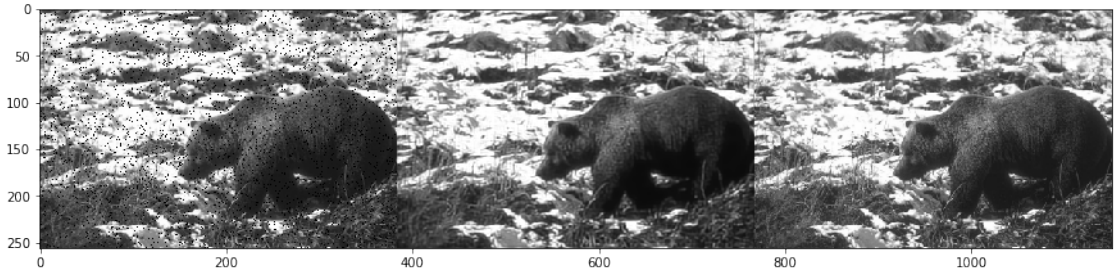


PSNR: 33.82648662684312

noisy image predicted ori

PSNR: 26.549803052146913

noisy image predicted ori



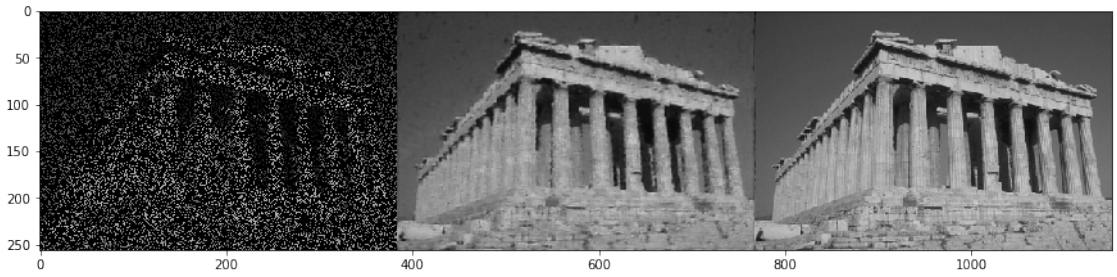
```
In [56]: for i in range(len(xtrain[:30])):
             tmp = np.expand_dims(xtrain[i], axis=0)
             pred = model6.predict(tmp).reshape(256,384)
             stacked1 = np.hstack((xtrain[i].reshape(256,384),pred,ytrain[i].reshape(256,384)))
             print("PSNR: ",psnr(ytrain[i].reshape(256,384),pred),"\n")
             plt.figure(figsize=(15,6))
             print('\tnoisy image\t\t\t\tpredicted\t\t\t\toriginal')
             plt.imshow(stacked1,cmap='gray',vmin=0,vmax=255)
             plt.show()
```

```
PSNR:  25.157604805261606

        noisy image                                predicted                                ori
```

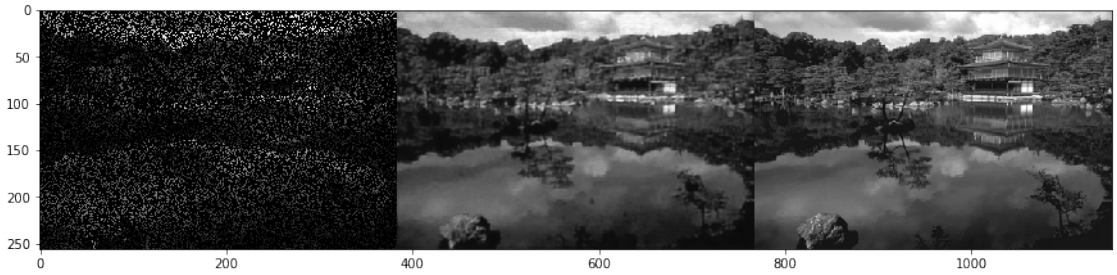

```
PSNR:  25.013580187007133

        noisy image                                predicted                                ori
```



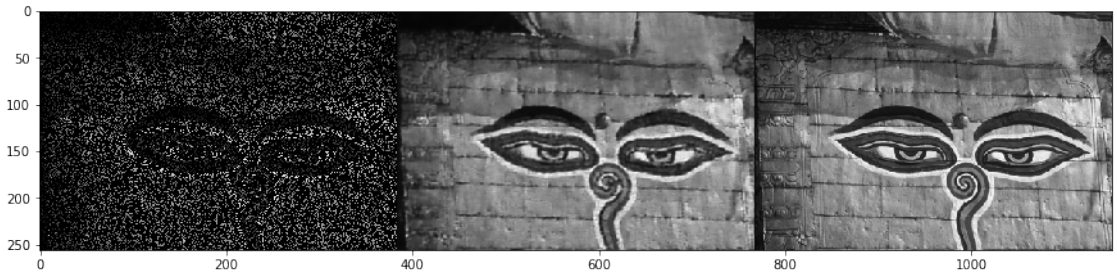

```
PSNR:  25.339223641905235

        noisy image                                predicted                                ori
```



PSNR: 23.646501907088346

noisy image predicted ori

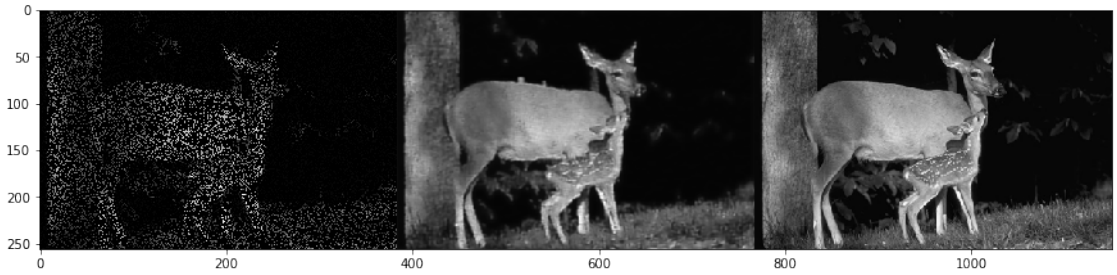


PSNR: 24.369304023583346

noisy image predicted ori

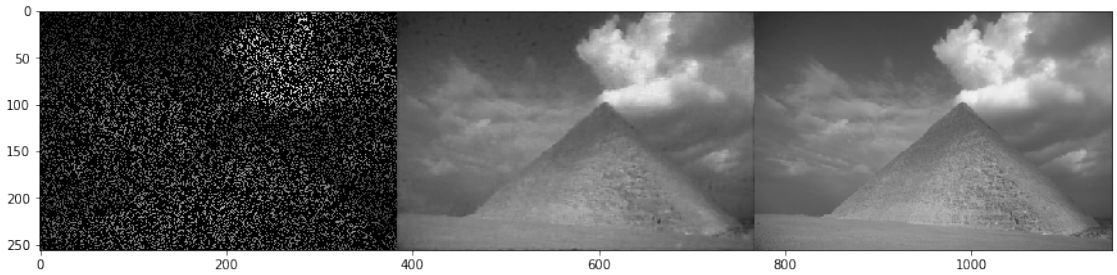
PSNR: 26.172697785131316

noisy image predicted ori



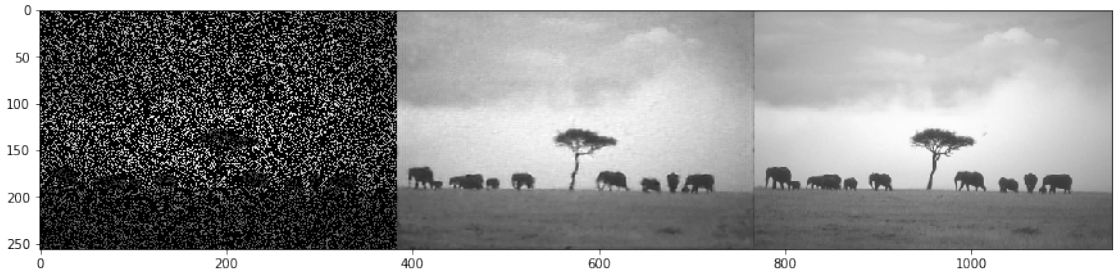
PSNR: 29.91232702041378

noisy image predicted ori



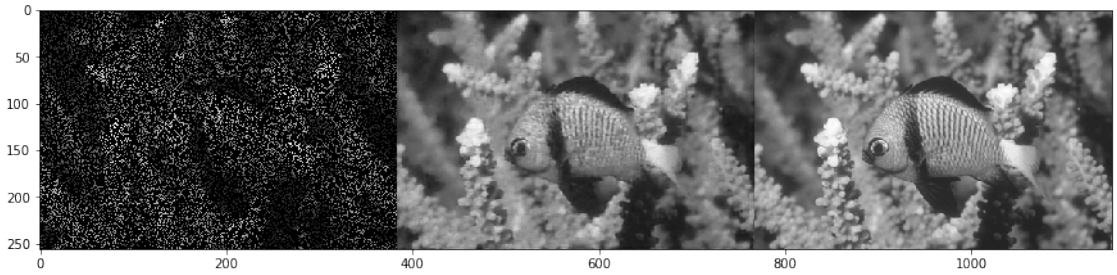
PSNR: 28.27663597531302

noisy image predicted ori



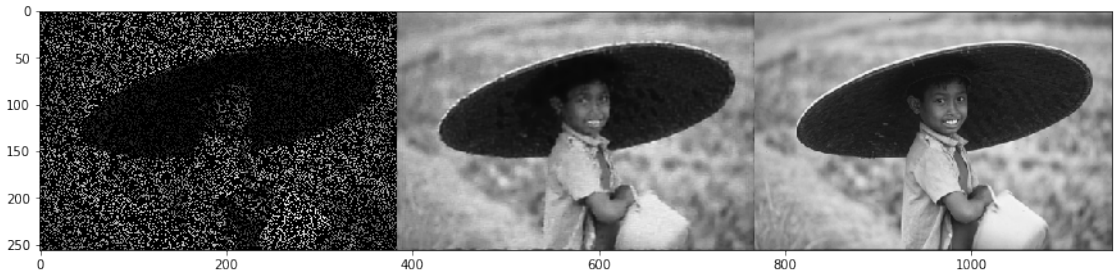
PSNR: 26.930399140088156

noisy image predicted ori



PSNR: 27.399419776592573

noisy image predicted ori

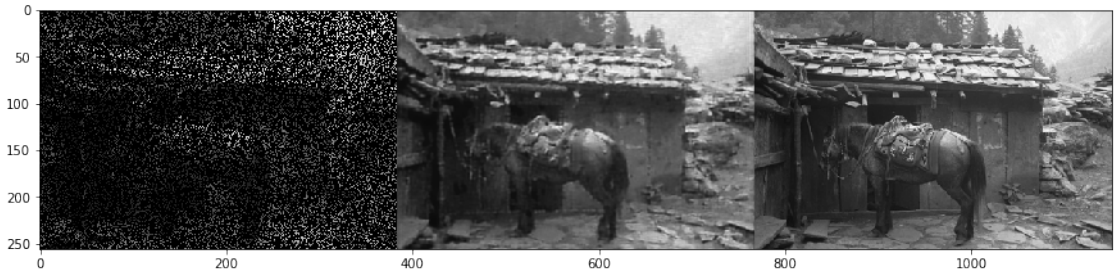


PSNR: 23.73115760822975

noisy image predicted ori

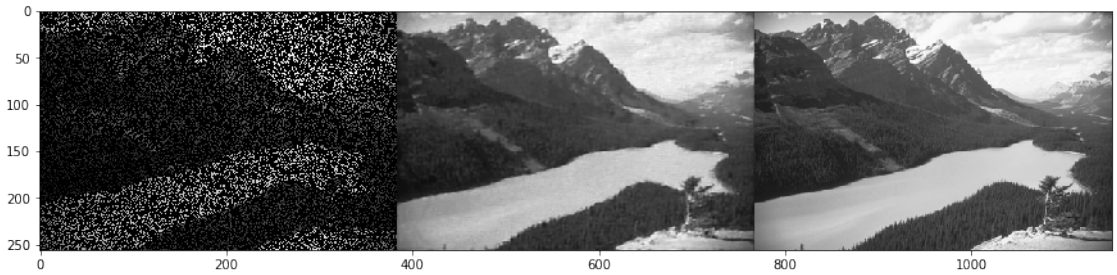
PSNR: 24.324995833724266

noisy image predicted ori



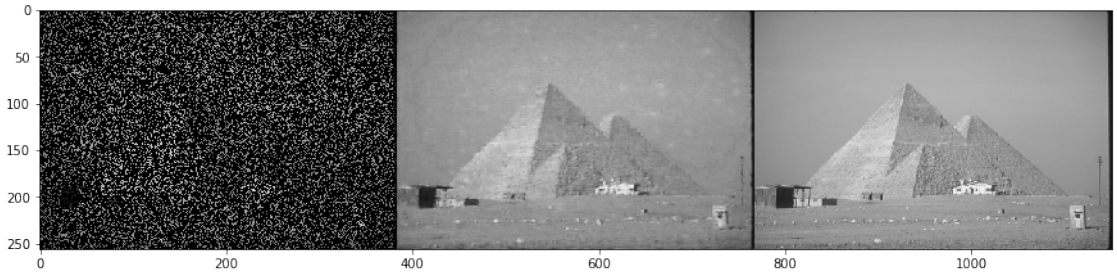
PSNR: 26.435224182195146

noisy image predicted ori



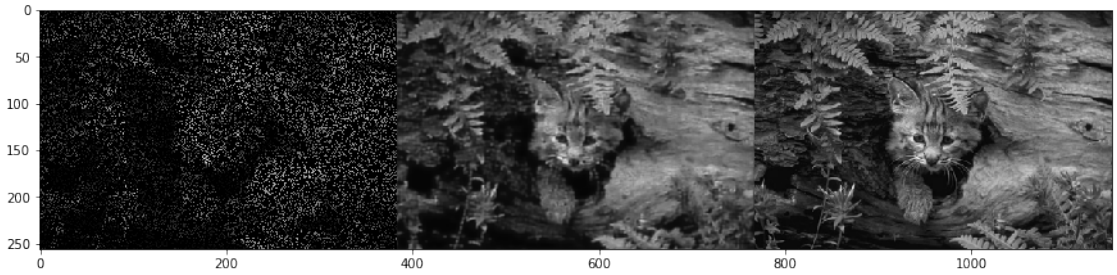
PSNR: 27.295313339912852

noisy image predicted ori



PSNR: 24.517224294043206

noisy image predicted ori

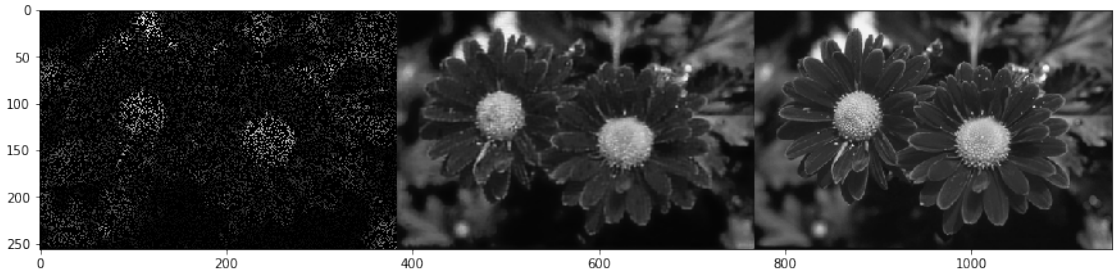


PSNR: 23.062915213338538

noisy image predicted ori

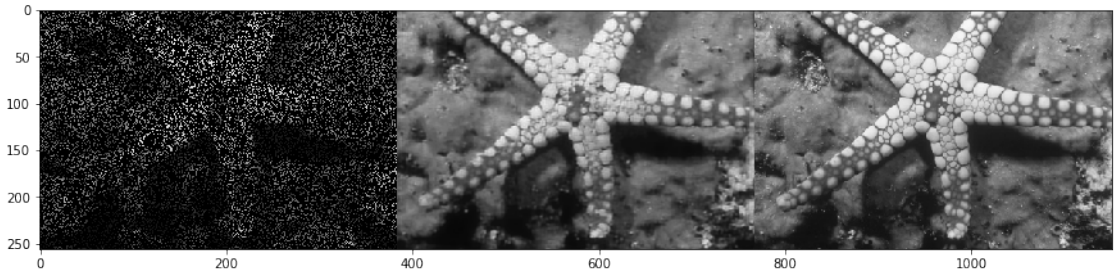
PSNR: 27.41767761685275

noisy image predicted ori



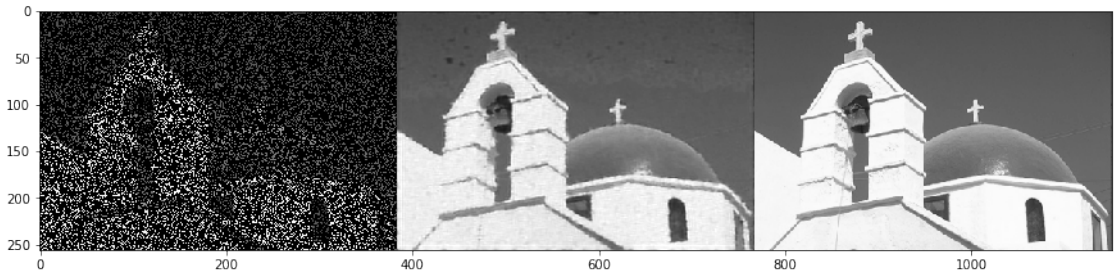
PSNR: 25.824494742861468

noisy image predicted ori



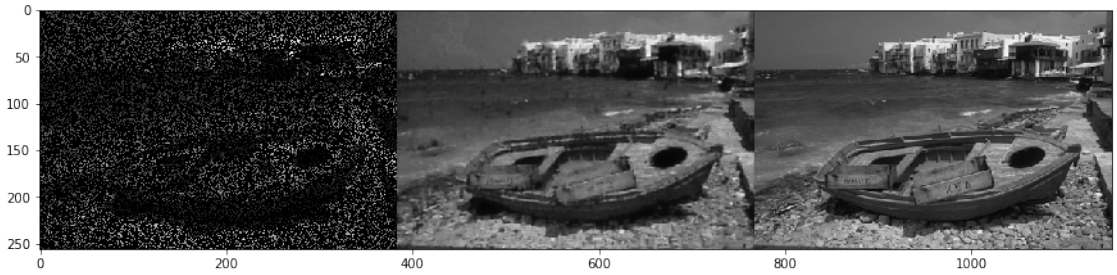
PSNR: 26.670612078243682

noisy image predicted ori



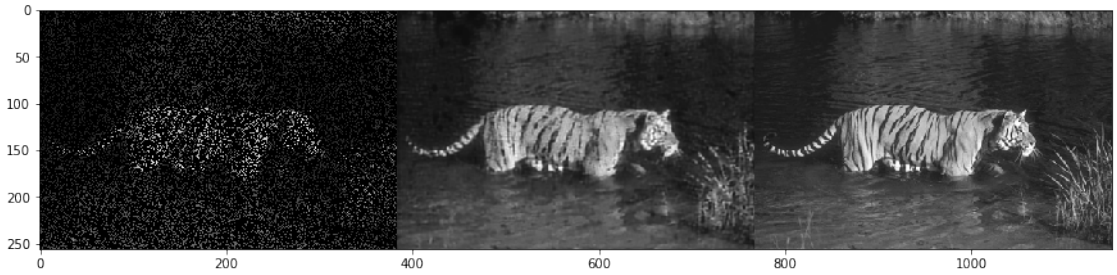
PSNR: 24.324575409676882

noisy image predicted ori



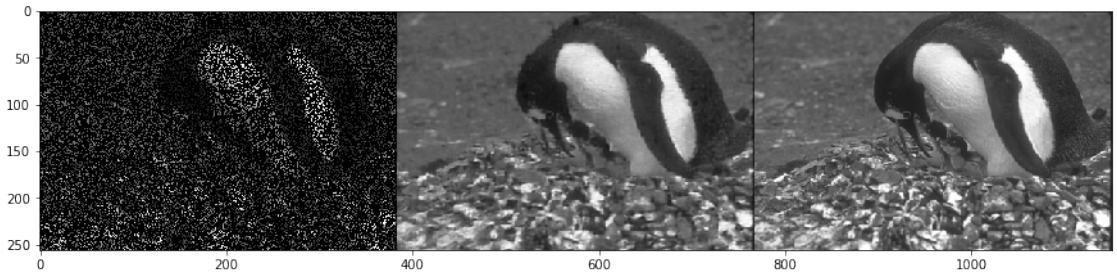
PSNR: 24.915776362244944

noisy image predicted ori



PSNR: 25.644083975716285

noisy image predicted ori

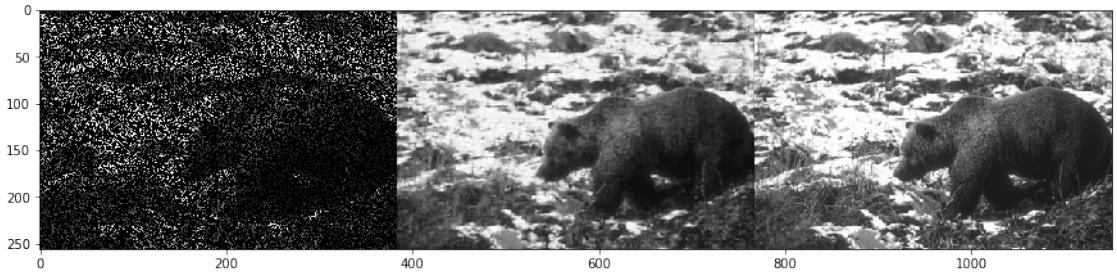


PSNR: 29.97504172987448

noisy image predicted ori

PSNR: 22.330123580697748

noisy image predicted ori



PSNR: 25.75575196448309

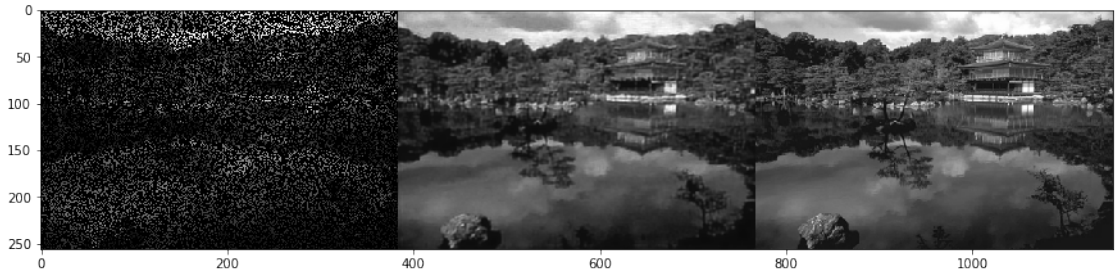
noisy image predicted ori

PSNR: 25.314014119624417

noisy image predicted ori

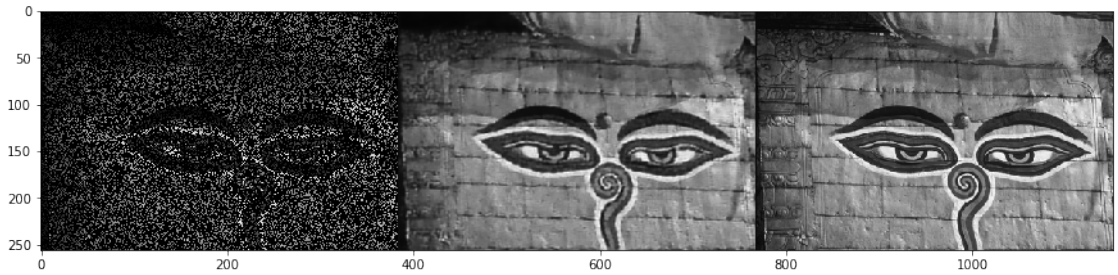
PSNR: 25.968656767359683

noisy image predicted ori



PSNR: 24.128417217662438

noisy image predicted ori

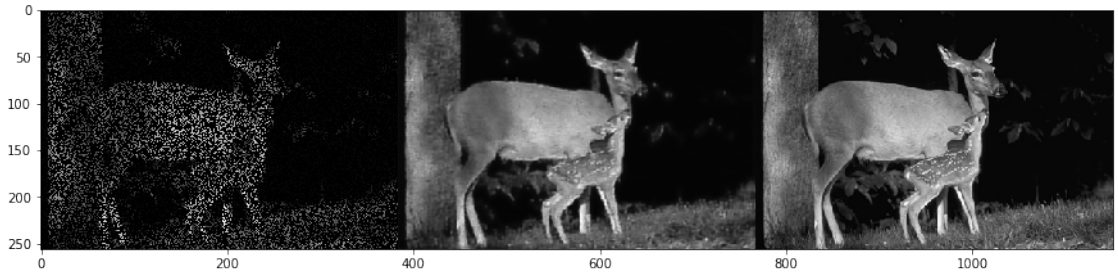


PSNR: 24.75814096819091

noisy image predicted ori

PSNR: 26.917982902820885

noisy image predicted ori



In [ ]: