1. Import Libraries

In [66]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense,Flatten,Conv2D,MaxPooling2D,UpSampling2D
from keras.datasets import mnist
from keras import datasets
```

2. Import Dataset

In [92]:

```
(x_train,_),(x_test,_)=datasets.mnist.load_data()
```

3. Check the shape

In [93]:

```
x_train.shape,x_test.shape
```

Out[93]:

```
((60000, 28, 28), (10000, 28, 28))
```

4. Reshape the data

In [94]:

```
x_train=x_train.astype('float')/255
x_test=x_test.astype('float')/255
```

In [95]:

```
x_train_reshape=np.reshape(x_train,newshape=(60000,28,28,1))
x_test_reshape=np.reshape(x_test,newshape=(10000,28,28,1))
```

In [71]:

```
x_train_reshape.shape
```

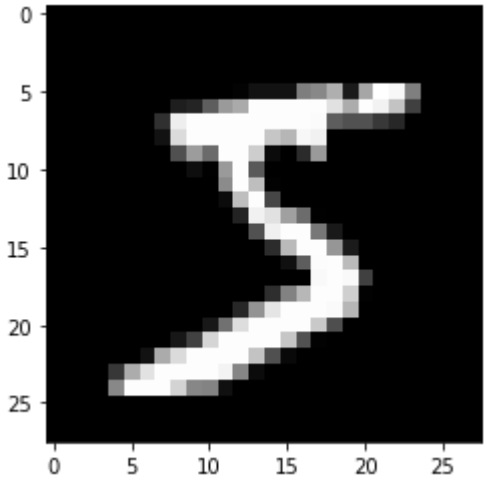
Out[71]:

```
(60000, 28, 28, 1)
```

5. Plot the data

In [96]:

```
plt.imshow(x_train_reshape[0],cmap='gray')
plt.show()
```



6. Import and add some noise

In [35]:

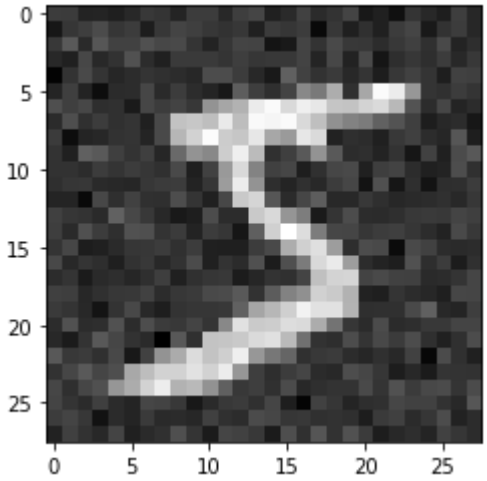
```
# x_train_noise=x_train_reshape+0.02*np.random.normal(loc=0.0, scale=1.0, size=(60000,28,28
# x_test_noise=x_test_reshape+0.02*np.random.normal(loc=0.0, scale=1.0, size=(10000,28,28,1
```

In [99]:

```
x_train_noise=x_train_reshape+0.1*np.random.normal(loc=0.0, scale=1.0, size=(60000,28,28,1)
x_test_noise=x_test_reshape+0.2*np.random.normal(loc=0.0, scale=1.0, size=(10000,28,28,1))
```

In [100]:

```
plt.imshow(x_train_noise[0],cmap='gray')
plt.show()
```



7. Mobile Climping

In [101]:

```
x_train_clip=np.clip(a=x_train_noise,a_min=0, a_max=1)
x_test_clip=np.clip(a=x_test_noise,a_min=0, a_max=1)
x_train_clip.shape
```

Out[101]:

```
(60000, 28, 28, 1)
```

8. encoding

In [102]:

```
model=Sequential()
model.add(Conv2D(input_shape=(28,28,1),filters=32,kernel_size=(3,3),strides=1,padding='same
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=8,kernel_size=(3,3),strides=1,padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(filters=8,kernel_size=(3,3),strides=1,padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),padding='same'))


model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_41 (Conv2D)          (None, 28, 28, 32)        320

 max_pooling2d_23 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 conv2d_42 (Conv2D)          (None, 14, 14, 8)         2312

 max_pooling2d_24 (MaxPoolin  (None, 7, 7, 8)          0
 g2D)

 conv2d_43 (Conv2D)          (None, 7, 7, 8)           584

 max_pooling2d_25 (MaxPoolin  (None, 4, 4, 8)          0
 g2D)

=================================================================
Total params: 3,216
Trainable params: 3,216
Non-trainable params: 0
_________________________________________________________________
```

9.Model Encoding

In [103]:

```
model.add(Conv2D(filters=8,kernel_size=(3,3),padding='same',activation='relu'))
model.add(UpSampling2D(size=(2,2)))
model.add(Conv2D(filters=8,kernel_size=(3,3),padding='same',activation='relu'))
model.add(UpSampling2D(size=(2,2)))
model.add(Conv2D(filters=32,kernel_size=(3,3),activation='relu'))
model.add(UpSampling2D(size=(2,2)))
model.add(Conv2D(filters=1,kernel_size=(3,3),padding='same',activation='relu'))

model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_41 (Conv2D)          (None, 28, 28, 32)        320

 max_pooling2d_23 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 conv2d_42 (Conv2D)          (None, 14, 14, 8)         2312

 max_pooling2d_24 (MaxPoolin  (None, 7, 7, 8)          0
 g2D)

 conv2d_43 (Conv2D)          (None, 7, 7, 8)           584

 max_pooling2d_25 (MaxPoolin  (None, 4, 4, 8)          0
 g2D)

 conv2d_44 (Conv2D)          (None, 4, 4, 8)           584

 up_sampling2d_10 (UpSamplin  (None, 8, 8, 8)          0
 g2D)

 conv2d_45 (Conv2D)          (None, 8, 8, 8)           584

 up_sampling2d_11 (UpSamplin  (None, 16, 16, 8)        0
 g2D)

 conv2d_46 (Conv2D)          (None, 14, 14, 32)        2336

 up_sampling2d_12 (UpSamplin  (None, 28, 28, 32)       0
 g2D)

 conv2d_47 (Conv2D)          (None, 28, 28, 1)         289

=================================================================
Total params: 7,009
Trainable params: 7,009
Non-trainable params: 0
_________________________________________________________________
```

10 Model Compile

In [104]:

```
model.compile(optimizer='rmsprop',loss='mean_squared_error')
```

In [105]:

```
model.fit(x=x_train_clip,y=x_train,batch_size=32,epochs=5,validation_data=(x_test_clip,x_te
```

```
Epoch 1/5
1875/1875 [==============================] - 107s 57ms/step - loss: 0.0310 -
val_loss: 0.0245
Epoch 2/5
1875/1875 [==============================] - 105s 56ms/step - loss: 0.0203 -
val_loss: 0.0216
Epoch 3/5
1875/1875 [==============================] - 111s 59ms/step - loss: 0.0175 -
val_loss: 0.0201
Epoch 4/5
1875/1875 [==============================] - 116s 62ms/step - loss: 0.0160 -
val_loss: 0.0186
Epoch 5/5
1875/1875 [==============================] - 116s 62ms/step - loss: 0.0149 -
val_loss: 0.0161
```

Out[105]:

```
<keras.callbacks.History at 0x21e9a6a2d60>
```

11. Model Evaluate

In [106]:

```
model.evaluate(x_test_clip,x_test)
```

```
313/313 [==============================] - 4s 11ms/step - loss: 0.0161
```

Out[106]:

```
0.016076110303401947
```

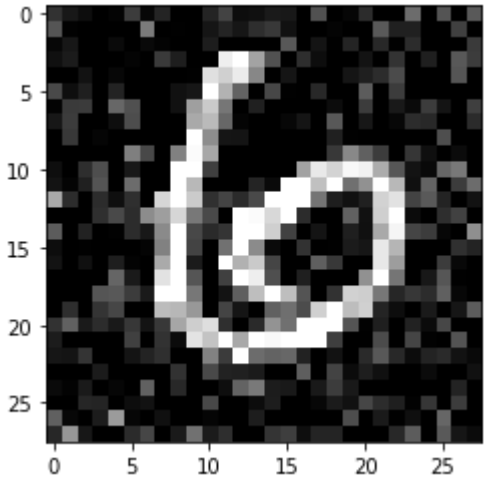
12 .Model Predict

In [108]:

```
y_pred=model.predict(x_testnoise)
```

In [109]:

```
plt.imshow(x_test_clip[11],cmap='gray')
plt.show()
```



In [111]:

```
plt.imshow(y_pred[11],cmap='gray')
plt.show()
```