In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import tqdm

from tensorflow.keras import utils
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
!unzip '/content/drive/MyDrive/Pushkar-MIT-DL/dataset/Archive.zip' -d '/content'
```

In [5]:
```
SIZE = 32
```

In [6]:
```
train_clean_data = []
path = r'/content/train_clean_data'
files = os.listdir(path)

for file in tqdm.tqdm(files):
    try:
        img = cv2.imread(path+"/"+file,0) # 0 for BW image, 1 for color
        #print('Original Dimensions : ',img.shape)
        img = cv2.resize(img,(SIZE,SIZE))
        train_clean_data.append(utils.img_to_array(img))
    except:
        pass
```

```
100%|██████████| 30000/30000 [00:03<00:00, 7519.43it/s]
```

In [7]:
```
test_clean_data = []
path = r'/content/test_clean_data'
files = os.listdir(path)

for file in tqdm.tqdm(files):
    try:
        img = cv2.imread(path+"/"+file,0) # 0 for BW image, 1 for color
        img = cv2.resize(img,(SIZE, SIZE))
        test_clean_data.append(utils.img_to_array(img))
    except:
        pass
```

```
100%|██████████| 20000/20000 [00:02<00:00, 7425.42it/s]
```

In [8]:
```
train_noisy_data = []
path = r'/content/test_noisy_data'
files = os.listdir(path)

for file in tqdm.tqdm(files):
    try:
        img = cv2.imread(path+"/"+file,0) # 0 for BW image, 1 for color
```

```
        #print('Original Dimensions : ',img.shape)
        img = cv2.resize(img,(SIZE, SIZE))
        train_noisy_data.append(utils.img_to_array(img))
    except:
        pass
```

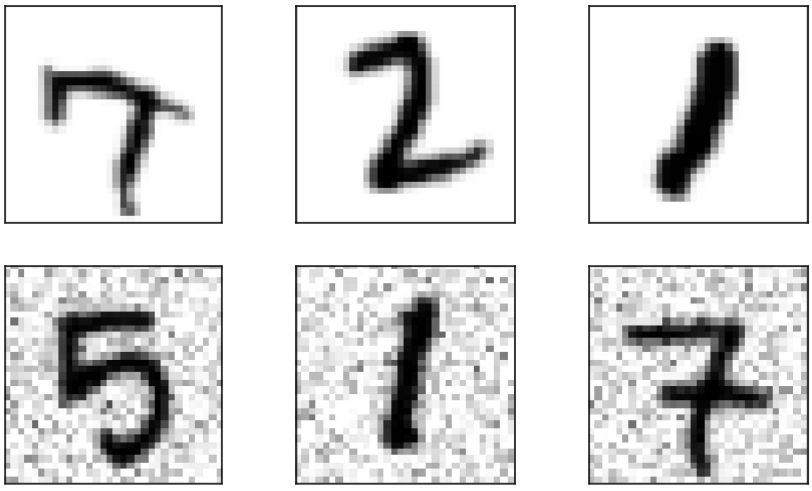
```
100%|██████████| 20000/20000 [00:03<00:00, 5956.82it/s]
```

In [9]:
```
test_noisy_data = []
path = r'/content/test_noisy_data'
files = os.listdir(path)

for file in tqdm.tqdm(files):
    try:
        img = cv2.imread(path+"/"+file,0) # 0 for BW image, 1 for color
        img = cv2.resize(img,(SIZE, SIZE))
        test_noisy_data.append(utils.img_to_array(img))
    except:
        pass
```

```
100%|██████████| 20000/20000 [00:03<00:00, 6223.97it/s]
```

In [10]:
```
train_clean = np.reshape(train_clean_data, (len(train_clean_data), SIZE, SIZE,1))
train_clean = train_clean.astype("float32")/255.0

train_noisy = np.reshape(train_noisy_data, (len(train_noisy_data), SIZE, SIZE,1))
train_noisy = train_noisy.astype("float32")/255.0

test_clean = np.reshape(test_clean_data, (len(test_clean_data), SIZE, SIZE,1))
test_clean = test_clean.astype("float32")/255.0

test_noisy = np.reshape(test_noisy_data, (len(test_noisy_data), SIZE, SIZE,1))
test_noisy = test_noisy.astype("float32")/255.0
```

In [11]:
```
plt.figure(figsize=(10, 2))
for i in range(1,4):
    ax = plt.subplot(1, 4, i)
    plt.imshow(train_clean[i].reshape(SIZE, SIZE), cmap="binary")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    #plt.title("Image without Noise")
plt.show()


plt.figure(figsize=(10, 2))
for i in range(1,4):
    ax = plt.subplot(1, 4, i)
    plt.imshow(train_noisy[i].reshape(SIZE, SIZE), cmap="binary")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    #plt.title("Image with Noise")
plt.show()
```



In [13]:
```
input_img = Input(shape=(SIZE, SIZE, 1))

x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(24, (3, 3), activation='relu', padding='same')(x)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(x)

encoded = layers.MaxPooling2D((2, 2), padding='same')(x) # Bottolneck layer # The repr
encoder = Model(input_img, encoded) # for checking latent vectors

x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(encoded)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(24, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)

decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = models.Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']

autoencoder.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 1)]       0

 conv2d_3 (Conv2D)           (None, 32, 32, 32)        320

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 24)        6936

 max_pooling2d_4 (MaxPooling  (None, 8, 8, 24)         0
 2D)

 conv2d_5 (Conv2D)           (None, 8, 8, 16)          3472

 max_pooling2d_5 (MaxPooling  (None, 4, 4, 16)         0
 2D)

 conv2d_6 (Conv2D)           (None, 4, 4, 16)          2320

 up_sampling2d (UpSampling2D  (None, 8, 8, 16)         0
 )

 conv2d_7 (Conv2D)           (None, 8, 8, 24)          3480

 up_sampling2d_1 (UpSampling  (None, 16, 16, 24)       0
 2D)

 conv2d_8 (Conv2D)           (None, 16, 16, 32)        6944

 up_sampling2d_2 (UpSampling  (None, 32, 32, 32)       0
 2D)

 conv2d_9 (Conv2D)           (None, 32, 32, 1)         289

=================================================================
Total params: 23,761
Trainable params: 23,761
Non-trainable params: 0
_________________________________________________________________
```

In [15]:
```
autoencoder.fit(train_noisy, train_clean, epochs=15, batch_size=32, shuffle=True, verb
```

```
Epoch 1/15
500/500 [==============================] - 5s 8ms/step - loss: 0.5739 - accuracy: 7.4
707e-04 - val_loss: 0.5743 - val_accuracy: 7.3242e-04
Epoch 2/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.4
176e-04 - val_loss: 0.5742 - val_accuracy: 7.8052e-04
Epoch 3/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.3
834e-04 - val_loss: 0.5742 - val_accuracy: 7.2021e-04
Epoch 4/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.3
834e-04 - val_loss: 0.5742 - val_accuracy: 7.7930e-04
Epoch 5/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.4
042e-04 - val_loss: 0.5742 - val_accuracy: 7.8052e-04
Epoch 6/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.4
054e-04 - val_loss: 0.5742 - val_accuracy: 7.8149e-04
Epoch 7/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.3
248e-04 - val_loss: 0.5742 - val_accuracy: 7.8052e-04
Epoch 8/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.4
078e-04 - val_loss: 0.5742 - val_accuracy: 7.7148e-04
Epoch 9/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.3
755e-04 - val_loss: 0.5742 - val_accuracy: 7.7222e-04
Epoch 10/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.3
254e-04 - val_loss: 0.5742 - val_accuracy: 7.8149e-04
Epoch 11/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.3
364e-04 - val_loss: 0.5742 - val_accuracy: 7.4414e-04
Epoch 12/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.3
273e-04 - val_loss: 0.5742 - val_accuracy: 7.8052e-04
Epoch 13/15
500/500 [==============================] - 3s 6ms/step - loss: 0.5739 - accuracy: 7.3
187e-04 - val_loss: 0.5742 - val_accuracy: 7.7979e-04
Epoch 14/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.4
567e-04 - val_loss: 0.5742 - val_accuracy: 7.8052e-04
Epoch 15/15
500/500 [==============================] - 3s 7ms/step - loss: 0.5739 - accuracy: 7.3
560e-04 - val_loss: 0.5742 - val_accuracy: 7.8052e-04
```

Out[15]:
```
<keras.callbacks.History at 0x7f13b007e810>
```

In [16]:
```
encoded_img = encoder.predict(test_noisy)
no_noise_img = autoencoder.predict(test_noisy)
```

```
625/625 [==============================] - 1s 2ms/step
625/625 [==============================] - 1s 2ms/step
```

In [17]:
```
#visulization
# Noisy Image
plt.figure(figsize=(20, 5))
for i in range(1,11):
    ax = plt.subplot(1, 11, i)
    plt.imshow(test_noisy[i].reshape(SIZE, SIZE), cmap="gray")
```

```
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.title("Noisy Image")
plt.show()

# Latent Vector
plt.figure(figsize=(20, 5))
for i in range(1,11):
    ax = plt.subplot(1, 11, i)
    img = encoded_img[i].reshape(16,16)
    plt.imshow(cv2.resize(img,(8,8)), cmap="gray")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.title("Encoded Image")
plt.show()

# Denoised Image
plt.figure(figsize=(20, 5))
for i in range(1,11):
    ax = plt.subplot(1, 11, i)
    plt.imshow(no_noise_img[i].reshape(SIZE, SIZE), cmap="gray")
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.title("Denoised Image")
plt.show()
```