final (2)

December 30, 2022

1 Load Packages

[34]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[35]:
```
! pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.8/dist-
packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.8/dist-
packages (from opencv-python) (1.21.6)
```

[36]:
```
from keras.layers import Conv2D,ConvLSTM2D,Conv3D,Cropping2D
from keras.layers import Input, Dropout, TimeDistributed,␣
 ↪RepeatVector,Activation
from keras.layers import MaxPooling2D, UpSampling2D,␣
 ↪BatchNormalization,Flatten, Dense, Reshape, LSTM
from keras import losses, Sequential
from keras import layers
from keras.callbacks import EarlyStopping
from keras.models import Model
from keras.optimizers import RMSprop, Adam
import keras

from numpy import reshape
import numpy as np
import matplotlib.pyplot as plt

import cv2
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
from skimage.metrics import structural_similarity
import time
```

2 Prepare the datasets of 48 videos

[4]:
```
import cv2
import numpy as np

dataset_3D = []
dataset = []

for i in range(48):
    # Open the video file
    video = cv2.VideoCapture("/content/drive/MyDrive/UROP Sibo/A Machine␣
↪learning problem/VIDEOS/fire_Chimney_video_{}.mp4".format(i))

    # Read the frames of the video one by one
    while True:
        # Read the next frame
        ret, frame = video.read()

        # If there are no more frames, break out of the loop
        if not ret:
            break

        # Convert the frame to grayscale and threshold it to create a binary␣
↪image
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

        binary_flat = binary.flatten()

        dataset.append(binary_flat)
        dataset_3D.append(gray)

dataset = np.array(dataset)
dataset_3D = np.array(dataset_3D)

np.save("dataset.npy", dataset)
np.save("dataset_3D.npy", dataset_3D)
```

[5]:
```
train_dataset = dataset[:640,:]
test_dataset = dataset[-128:,:]
train_dataset3D = dataset_3D[:640,:]
test_dataset3D = dataset_3D[-128:,:]

print(dataset.shape)
print(train_dataset.shape)
print(dataset_3D.shape)
```

```
np.save("train_dataset.npy", train_dataset)
np.save("test_dataset.npy", test_dataset)
np.save("train_dataset3D.npy", train_dataset3D)
np.save("test_dataset3D.npy", test_dataset3D)
```

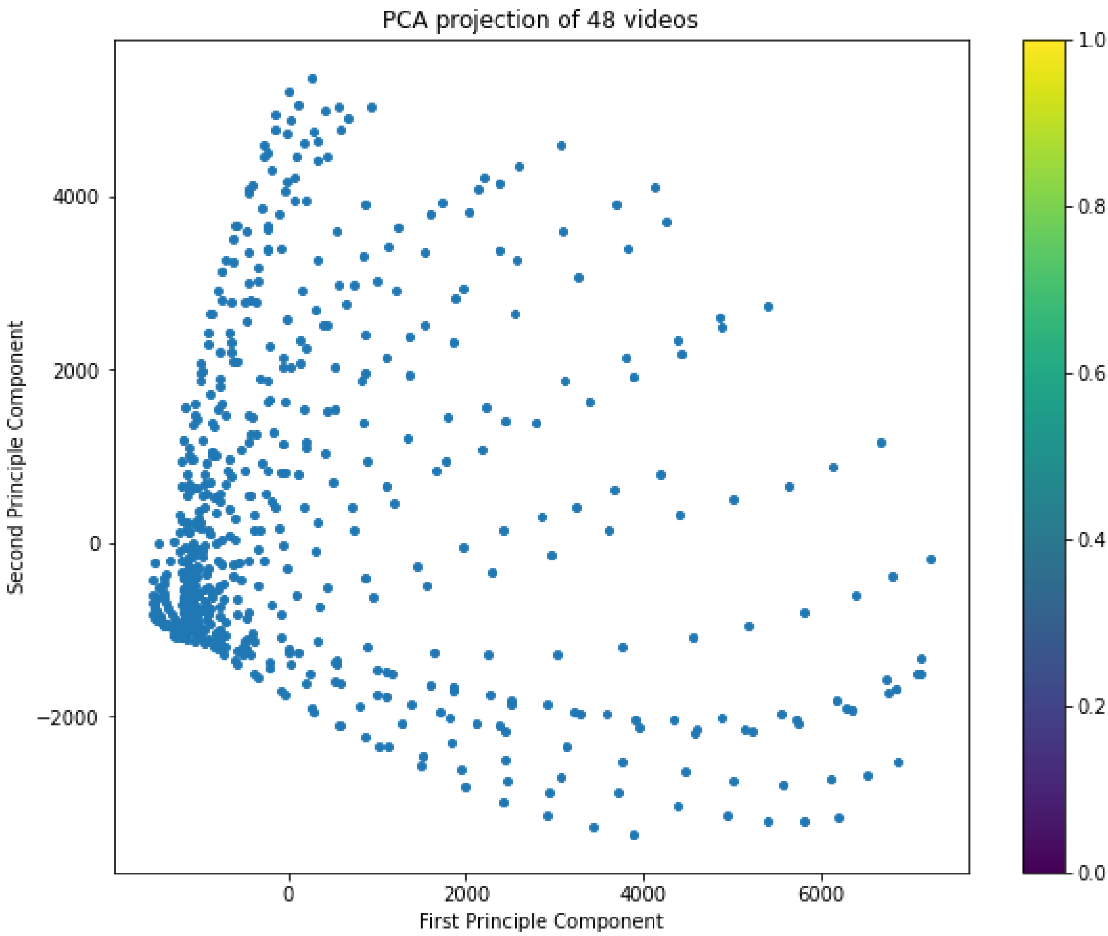
```
(768, 16384)
(640, 16384)
(768, 128, 128)
```

3 1. PCA

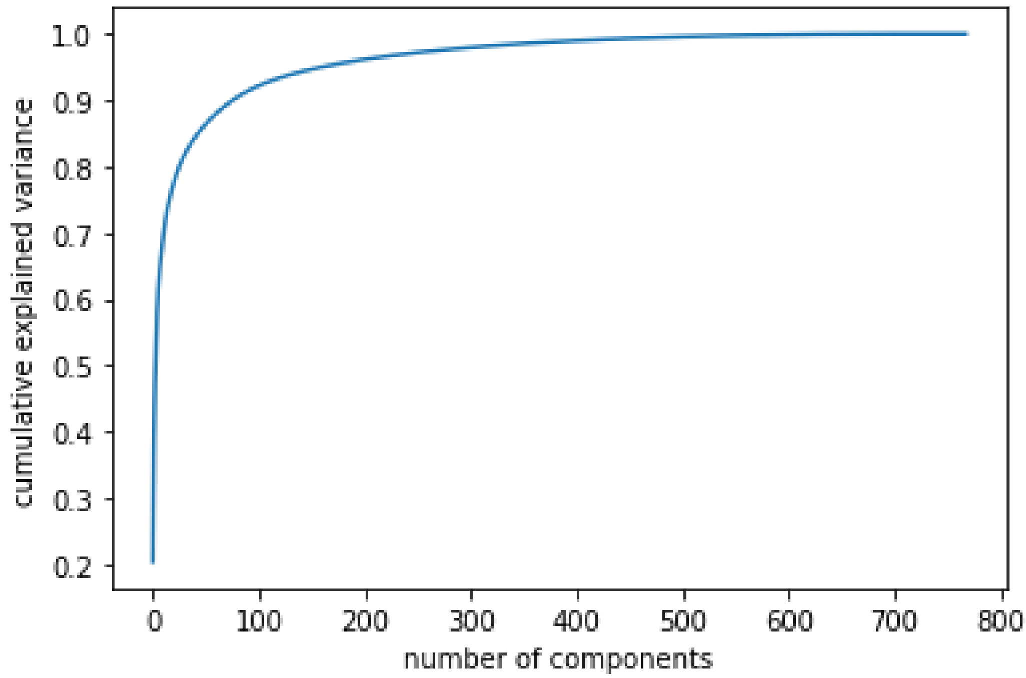
[6]:
```
X_data = np.load("dataset.npy")
pca = PCA(n_components=2)
pca.fit(X_data)
X_pca=pca.transform(X_data)
```

[7]:
```
number_of_videos =  48
index_range = number_of_videos*16
fig = plt.figure(figsize = (10,8))
ax = fig.add_subplot(1,1,1)
scatter = ax.scatter(X_pca[:index_range,0],
                     X_pca[:index_range,1],
                     s = 16,
                     cmap = plt.get_cmap('jet', number_of_videos))

ax.set_xlabel("First Principle Component")
ax.set_ylabel("Second Principle Component")
ax.set_title("PCA projection of {} videos".format(number_of_videos))

fig.colorbar(scatter)
```

[7]:
```
<matplotlib.colorbar.Colorbar at 0x7efe55683b20>
```



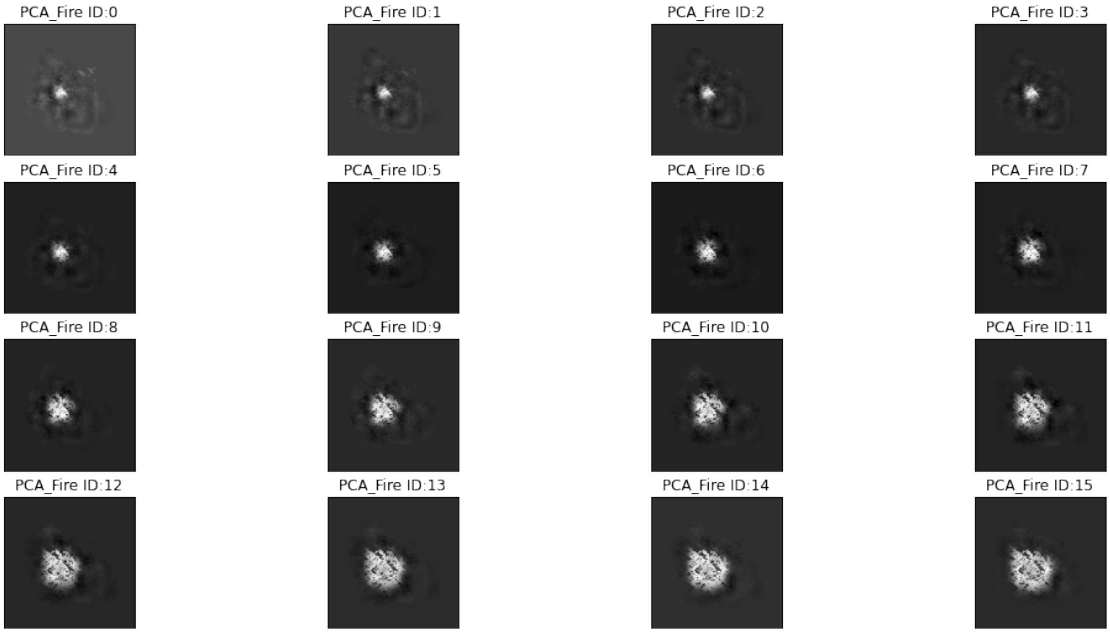
[8]:
```
pca = PCA()
pca.fit(X_data)

plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



We see that these 80 components account for just over 90% of the variance. That would lead us to believe that using these 80 components, we would recover most of the essential characteristics of the data.

3.1 Show 16 snapshots of first video under 32-PCA and original

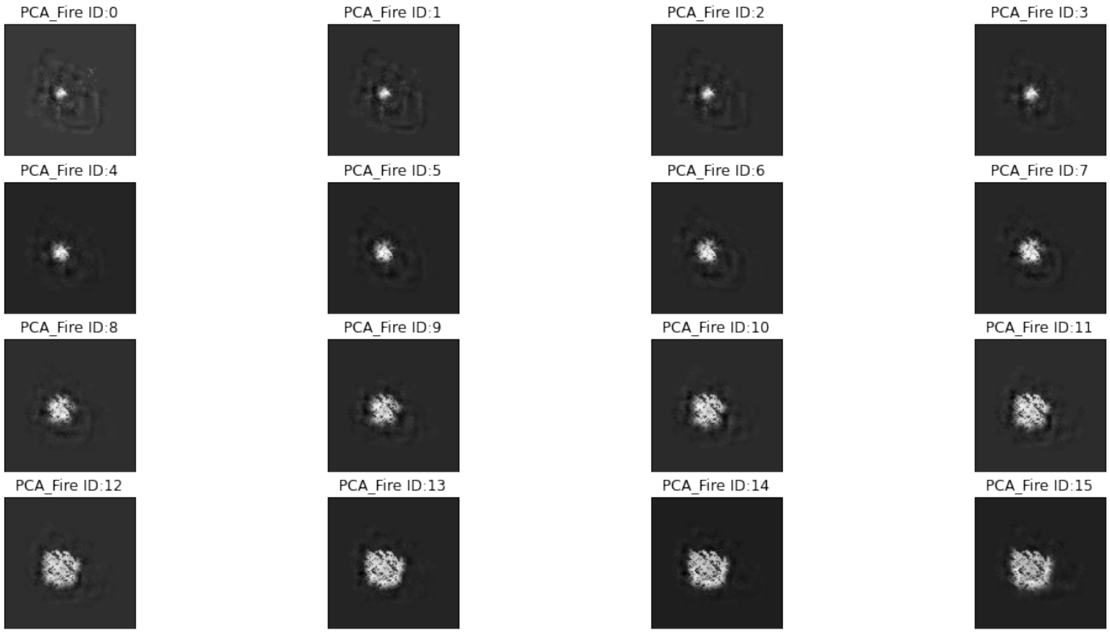
[9]:
```
pca = PCA(n_components = 32, whiten = True)
pca_X = pca.fit_transform(X_data)
X_proj_img = pca.inverse_transform(pca_X).reshape(-1,128,128)
fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(X_proj_img[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("PCA_Fire ID:{}".format(ids))
```



[10]:
```
x = np.zeros((16384,))
for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
pca_err_32 = x.sum()/768/128/128
```

3.2 Show 16 snapshots of first video under 32-PCA

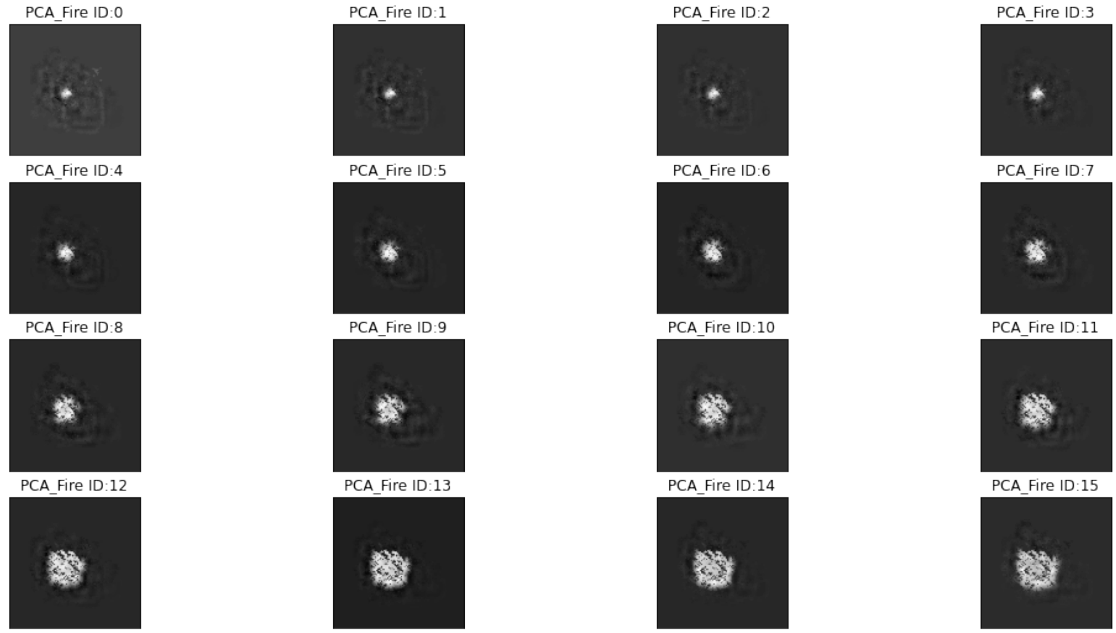
[11]:
```
pca = PCA(n_components = 50, whiten = True)
pca_X = pca.fit_transform(X_data)
X_proj_img = pca.inverse_transform(pca_X).reshape(-1,128,128)
fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(X_proj_img[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("PCA_Fire ID:{}".format(ids))
```



[12]:
```
x = np.zeros((16384,))
for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
pca_err_50 =x.sum()/768/128/128
```

3.3 Show 16 snapshots of first video under 64-PCA

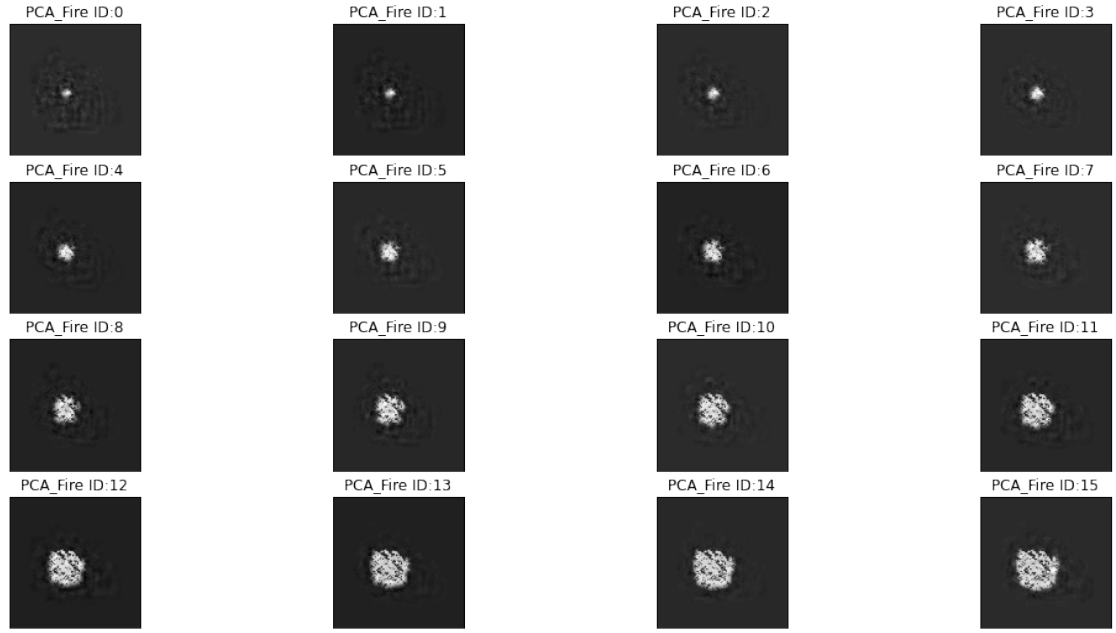
[13]:
```
pca = PCA(n_components = 64, whiten = False)
pca_X = pca.fit_transform(X_data)
X_proj_img = pca.inverse_transform(pca_X).reshape(-1,128,128)
fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(X_proj_img[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("PCA_Fire ID:{}".format(ids))
```



[14]:
```
x = np.zeros((16384,))
for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
pca_err_64 =x.sum()/768/128/128
```

3.4 Show 16 snapshots of first video under 96-PCA

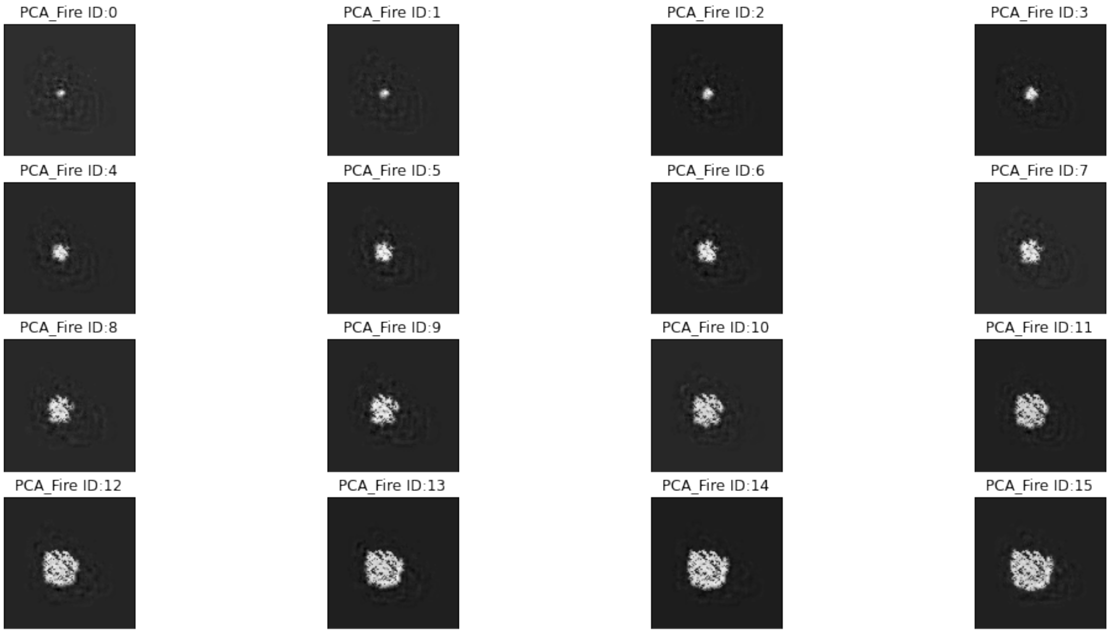
[15]:
```
pca = PCA(n_components = 96, whiten = True)
pca_X = pca.fit_transform(X_data)
reduce = np.save('pca_160.npy', pca_X)
X_proj_img = pca.inverse_transform(pca_X).reshape(-1,128,128)

fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(X_proj_img[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("PCA_Fire ID:{}".format(ids))
```



[16]:
```
x = np.zeros((16384,))
for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
pca_err_96 = x.sum()/768/128/128
```

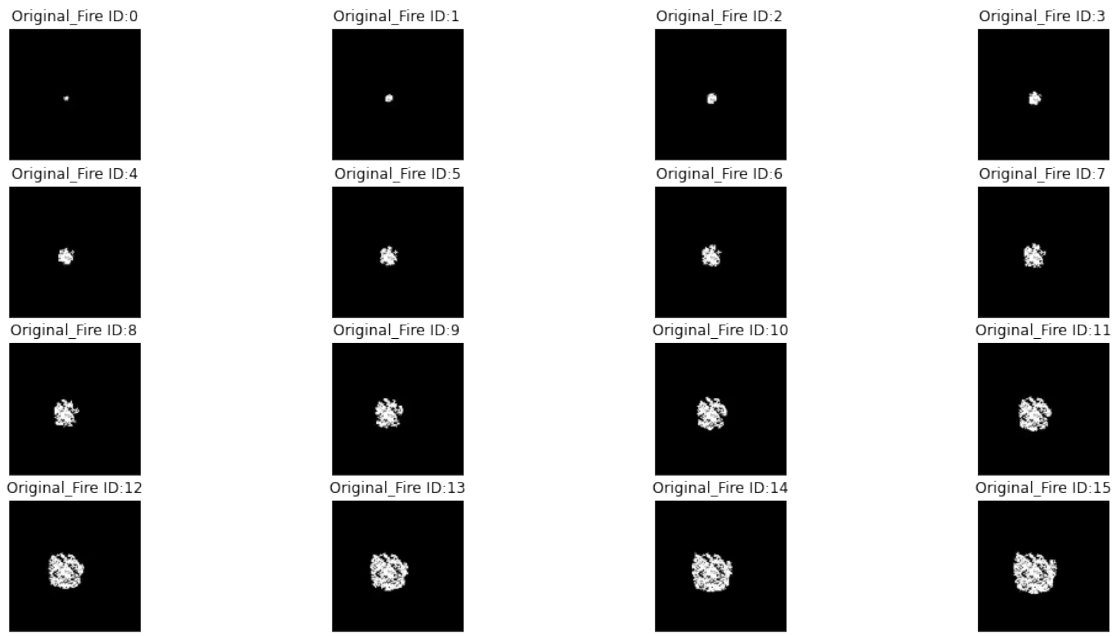
3.5 Show 16 snapshots of first video under 128-PCA

[17]:
```
pca = PCA(n_components = 128, whiten = True)
pca_X = pca.fit_transform(X_data)
reduce = np.save('pca_160.npy', pca_X)
X_proj_img = pca.inverse_transform(pca_X).reshape(-1,128,128)

fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(X_proj_img[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("PCA_Fire ID:{}".format(ids))
```



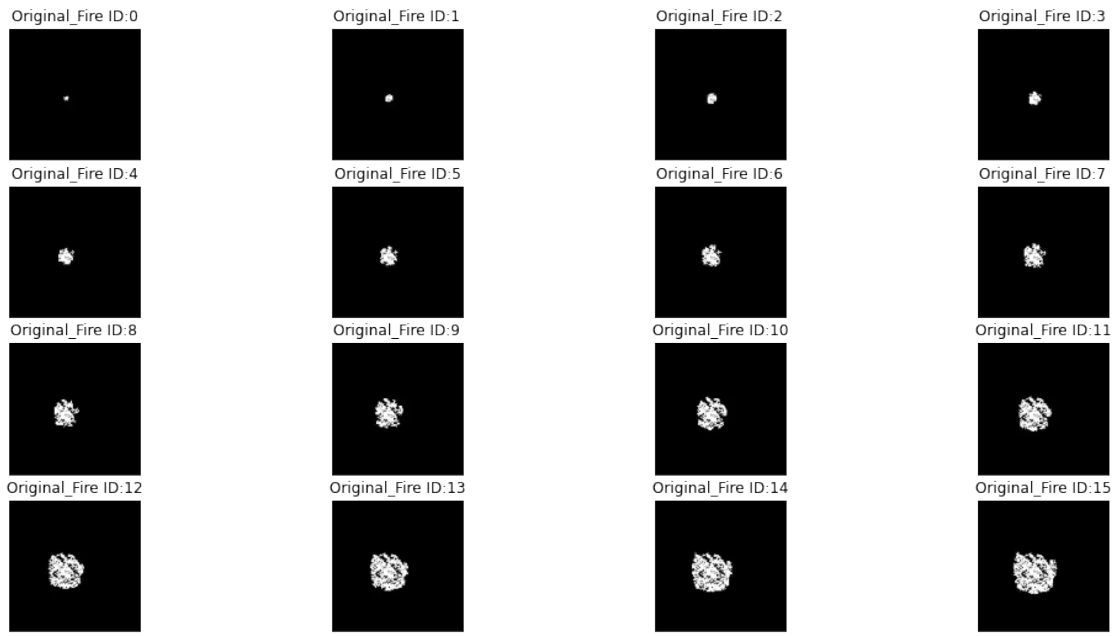
[18]:
```
x = np.zeros((16384,))
for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
pca_err_128 = x.sum()/768/128/128
```

[19]:
```
fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(dataset_3D[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("Original_Fire ID:{}".format(ids))
```



3.6 Show 16 snapshots of first video under original condition

[20]:
```
fig, arr = plt.subplots(nrows = 4, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(16):
    ind = ids
    arr[ids].imshow(dataset_3D[ind], cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("Original_Fire ID:{}".format(ids))
```



4 2. New__version CAE

[246]:
```
dataset = np.load('/content/train_dataset3D.npy')
dataset_test = np.load('/content/test_dataset3D.npy')
X = reshape(dataset_3D, (len(dataset_3D), 128, 128, 1)) .astype('float32')/255
x_train = reshape(dataset, (len(dataset), 128, 128, 1)) .astype('float32')/255

x_test = reshape(dataset_test, (len(dataset_test), 128, 128, 1)).
 ↪astype('float32')/255
```

[247]:
```
def encoder_f(input_img,n):
  #input_img = Input(shape=(128,128, 1))

  x = Conv2D(64, (10, 10), activation='relu', padding='same')(input_img)
  #x = BatchNormalization()(x)
  #x = Conv2D(16, (5, 5), activation='relu', padding='same')(x)
  x = MaxPooling2D((2, 2), padding='same')(x)
  x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
  x = MaxPooling2D((2, 2), padding='same')(x)
  x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
  x = MaxPooling2D((2, 2), padding='same')(x)
```

```
  x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
  x = MaxPooling2D((2, 2), padding='same')(x)
  x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)

  x = Flatten()(x)
  x = Dense(n)(x)

  return Model(input_img, x), x

def decoder_f(decode_img):
  x = Dense(256)(decode_img)
  x = Reshape((8, 8, 4), name='predictions')(x)
  x = Conv2D(4, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
  x = UpSampling2D((2,2))(x)
  x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
  x = UpSampling2D((2,2))(x)
  x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
  x = UpSampling2D((2,2))(x)
  x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
  x = UpSampling2D((2,2))(x)
  # x = Cropping2D(cropping=((16, 0), (16, 0)), data_format=None)(x)

  decoded = Conv2D(1, (5, 5), activation='sigmoid', padding='same')(x)

  return Model(decode_img, decoded), decoded
```

4.1 autuencoder-1: *32*

[248]:
```
auto_input = Input(shape=(128,128,1))
encoded, x = encoder_f(auto_input,32)

decoded, decode_output = decoder_f(x)
autoencoder_1 = Model(auto_input, decode_output)
autoencoder_1.compile(optimizer='rmsprop', loss='mse')
autoencoder_1.summary()
```

```
Model: "model_20"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_18 (InputLayer)       [(None, 128, 128, 1)]     0

 conv2d_35 (Conv2D)          (None, 128, 128, 64)      6464

 max_pooling2d_8 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_36 (Conv2D)          (None, 64, 64, 32)        18464

 batch_normalization_40 (Bat  (None, 64, 64, 32)       128
 chNormalization)

 conv2d_37 (Conv2D)          (None, 64, 64, 32)        9248

 max_pooling2d_9 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 conv2d_38 (Conv2D)          (None, 32, 32, 16)        4624

 batch_normalization_41 (Bat  (None, 32, 32, 16)       64
 chNormalization)

 conv2d_39 (Conv2D)          (None, 32, 32, 16)        2320

 max_pooling2d_10 (MaxPoolin  (None, 16, 16, 16)       0
 g2D)

 conv2d_40 (Conv2D)          (None, 16, 16, 8)         1160

 batch_normalization_42 (Bat  (None, 16, 16, 8)        32
 chNormalization)

 conv2d_41 (Conv2D)          (None, 16, 16, 4)         292

 max_pooling2d_11 (MaxPoolin  (None, 8, 8, 4)          0
 g2D)

 conv2d_42 (Conv2D)          (None, 8, 8, 4)           148

 batch_normalization_43 (Bat  (None, 8, 8, 4)          16
 chNormalization)

 flatten_2 (Flatten)         (None, 256)               0

 dense_8 (Dense)             (None, 32)                8224
```

```
dense_9 (Dense)             (None, 256)              8448

predictions (Reshape)       (None, 8, 8, 4)          0

conv2d_43 (Conv2D)          (None, 8, 8, 4)          148

batch_normalization_44 (Bat  (None, 8, 8, 4)         16
chNormalization)

conv2d_44 (Conv2D)          (None, 8, 8, 8)          296

up_sampling2d_8 (UpSampling  (None, 16, 16, 8)       0
2D)

conv2d_45 (Conv2D)          (None, 16, 16, 16)       1168

batch_normalization_45 (Bat  (None, 16, 16, 16)      64
chNormalization)

conv2d_46 (Conv2D)          (None, 16, 16, 16)       2320

up_sampling2d_9 (UpSampling  (None, 32, 32, 16)      0
2D)

conv2d_47 (Conv2D)          (None, 32, 32, 32)       4640

batch_normalization_46 (Bat  (None, 32, 32, 32)      128
chNormalization)

conv2d_48 (Conv2D)          (None, 32, 32, 32)       9248

up_sampling2d_10 (UpSamplin  (None, 64, 64, 32)      0
g2D)

conv2d_49 (Conv2D)          (None, 64, 64, 32)       9248

batch_normalization_47 (Bat  (None, 64, 64, 32)      128
chNormalization)

conv2d_50 (Conv2D)          (None, 64, 64, 64)       18496

up_sampling2d_11 (UpSamplin  (None, 128, 128, 64)    0
g2D)

conv2d_51 (Conv2D)          (None, 128, 128, 1)      1601

=================================================================
```

```
Total params: 107,133
Trainable params: 106,845
Non-trainable params: 288
_________________________________________________________________
```

[249]:
```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)
history = autoencoder_1.fit(x_train, x_train, epochs=1000,␣
 ↪batch_size=2,shuffle=True, validation_data=(x_test,␣
 ↪x_test),callbacks=[early_stopping, reduce_lr],)
```

```
Epoch 1/1000
320/320 [==============================] - 7s 15ms/step - loss: 0.0146 -
val_loss: 0.0072 - lr: 0.0010
Epoch 2/1000
320/320 [==============================] - 6s 19ms/step - loss: 0.0085 -
val_loss: 0.0077 - lr: 0.0010
Epoch 3/1000
320/320 [==============================] - 5s 17ms/step - loss: 0.0073 -
val_loss: 0.0083 - lr: 0.0010
Epoch 4/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0066 -
val_loss: 0.0048 - lr: 0.0010
Epoch 5/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0060 -
val_loss: 0.0045 - lr: 0.0010
Epoch 6/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0056 -
val_loss: 0.0047 - lr: 0.0010
Epoch 7/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0051 -
val_loss: 0.0043 - lr: 0.0010
Epoch 8/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0049 -
val_loss: 0.0039 - lr: 0.0010
Epoch 9/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0047 -
val_loss: 0.0041 - lr: 0.0010
Epoch 10/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0044 -
val_loss: 0.0039 - lr: 0.0010
Epoch 11/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0043 -
val_loss: 0.0037 - lr: 0.0010
Epoch 12/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0041 -
val_loss: 0.0037 - lr: 0.0010
Epoch 13/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0039 -
val_loss: 0.0035 - lr: 0.0010
Epoch 14/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0038 -
val_loss: 0.0039 - lr: 0.0010
Epoch 15/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0036 -
val_loss: 0.0033 - lr: 0.0010
Epoch 16/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0034 -
val_loss: 0.0036 - lr: 0.0010
Epoch 17/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0034 -
val_loss: 0.0038 - lr: 0.0010
Epoch 18/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0033 -
val_loss: 0.0038 - lr: 0.0010
Epoch 19/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0032 -
val_loss: 0.0037 - lr: 0.0010
Epoch 20/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0031 -
val_loss: 0.0033 - lr: 0.0010
Epoch 21/1000
320/320 [==============================] - 5s 15ms/step - loss: 0.0023 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 22/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0021 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 23/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0030 - lr: 1.0000e-04
Epoch 24/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0020 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 25/1000
320/320 [==============================] - 5s 15ms/step - loss: 0.0019 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 26/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0030 - lr: 1.0000e-04
Epoch 27/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0029 - lr: 1.0000e-05
Epoch 28/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-05
Epoch 29/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-05
Epoch 30/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-05
Epoch 31/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-05
Epoch 32/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-06
Epoch 33/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-06
Epoch 34/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-06
Epoch 35/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-06
Epoch 36/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-06
Epoch 37/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0029 - lr: 1.0000e-07
```

[250]:
```python
x = np.zeros((16384,))
X_proj_img = autoencoder_1.predict(dataset_3D)
X_proj_img = np.squeeze(X_proj_img, axis=-1)

for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
err_32 = x.sum()/768/128/128
err_32
```

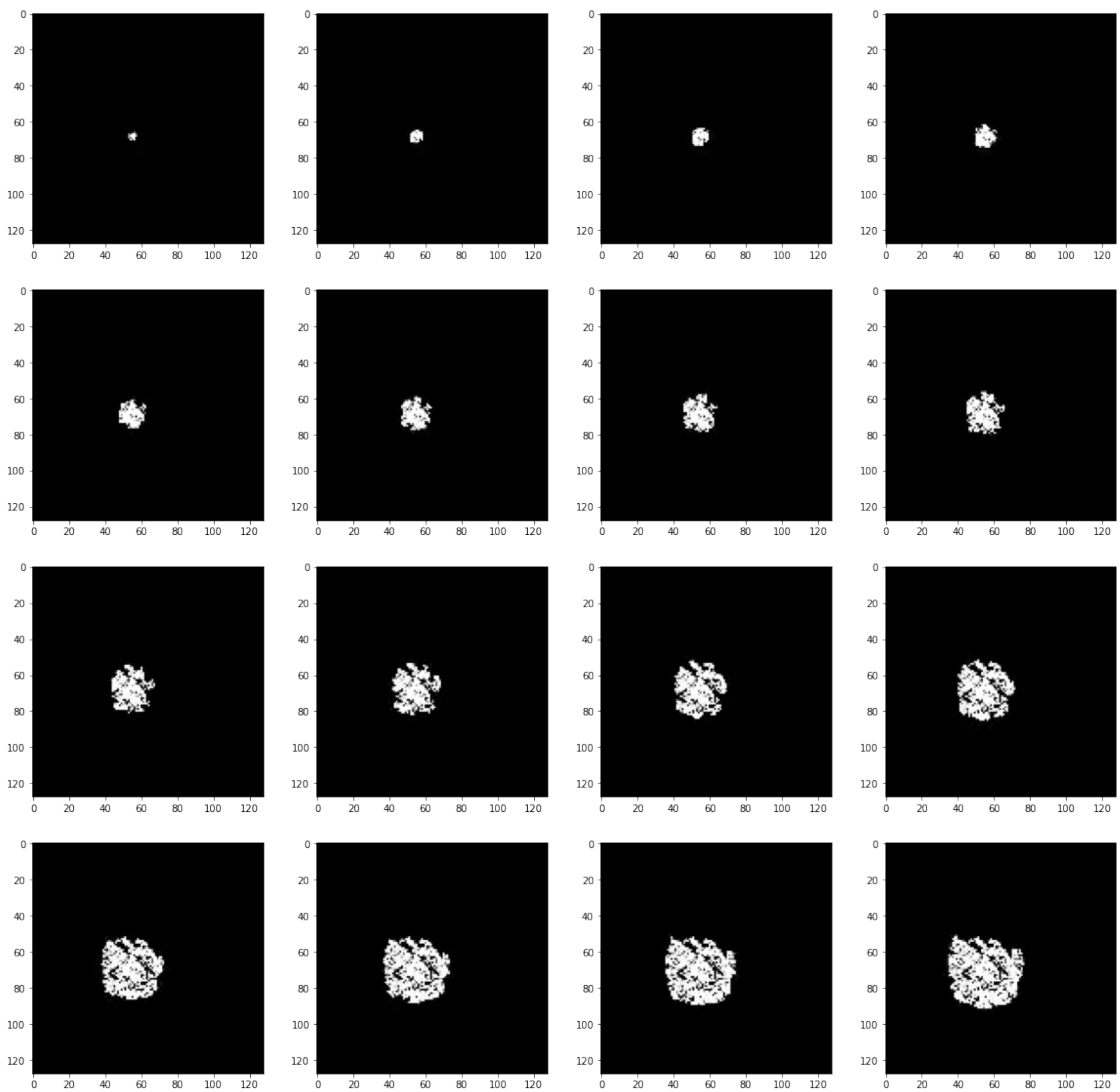
```
24/24 [==============================] - 1s 23ms/step
```

[250]: 4.894783733018802

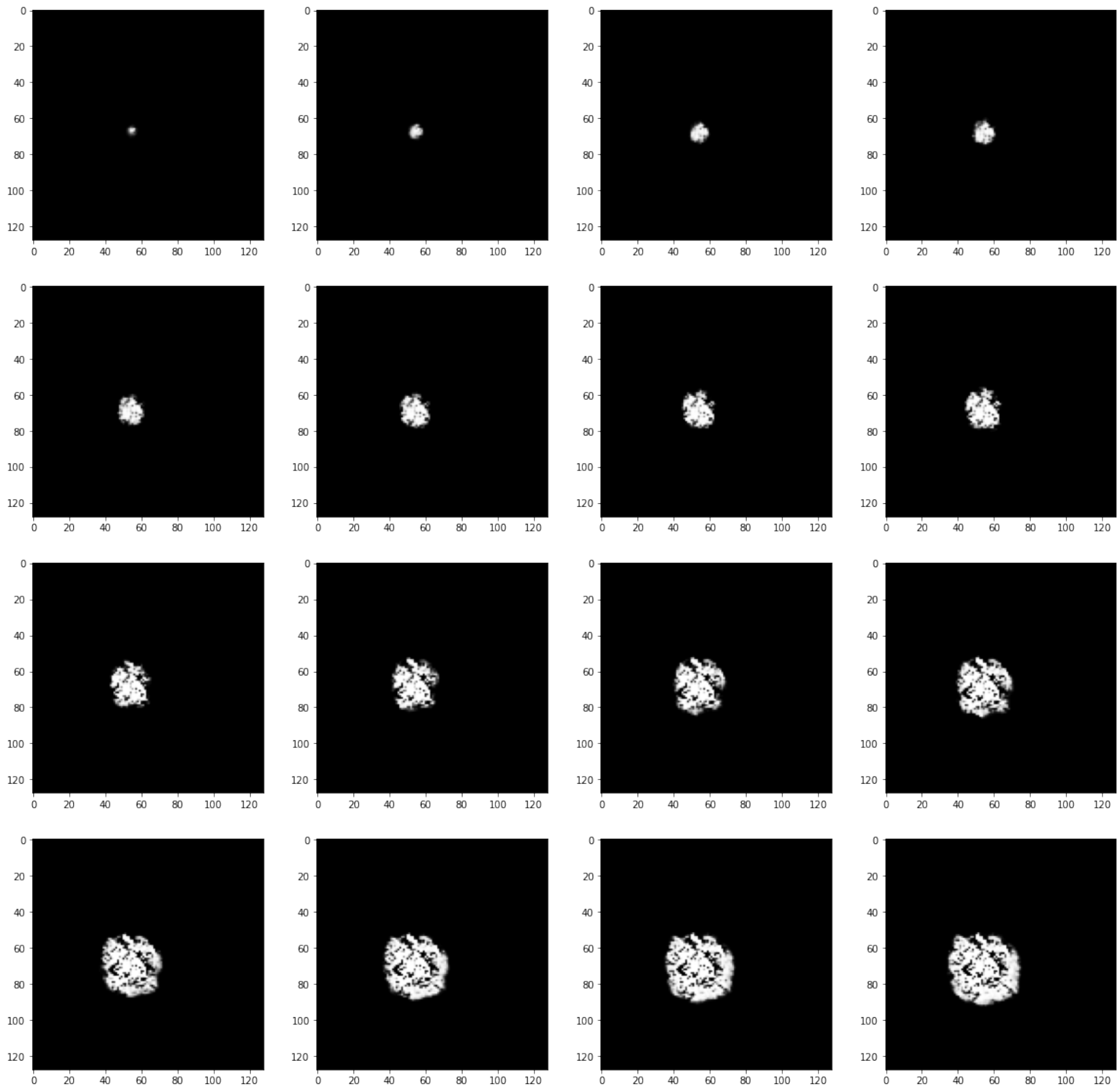
[251]:
```python
pred = autoencoder_1.predict(x_train)
plt.figure(figsize=(20, 20))
print("First Test Video")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(x_train[i, ..., 0], cmap='gray')
plt.show()
```

```python
plt.figure(figsize=(20, 20))
print("Reconstruction of First Test Images")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(pred[i, ..., 0], cmap='gray')
plt.show()
```

```
20/20 [==============================] - 1s 22ms/step
First Test Video
```



```
Reconstruction of First Test Images
```



4.2 autuencoder-1: *50*

[270]:
```
auto_input = Input(shape=(128,128,1))
encoded, x = encoder_f(auto_input,50)

decoded, decode_output = decoder_f(x)
autoencoder_2 = Model(auto_input, decode_output)
autoencoder_2.compile(optimizer='rmsprop', loss='mse')
autoencoder_2.summary()
```

```
Model: "model_35"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
input_28 (InputLayer)       [(None, 128, 128, 1)]     0

conv2d_120 (Conv2D)         (None, 128, 128, 64)      6464

max_pooling2d_28 (MaxPoolin  (None, 64, 64, 64)       0
g2D)

conv2d_121 (Conv2D)         (None, 64, 64, 32)        18464

batch_normalization_80 (Bat  (None, 64, 64, 32)       128
chNormalization)

conv2d_122 (Conv2D)         (None, 64, 64, 32)        9248

max_pooling2d_29 (MaxPoolin  (None, 32, 32, 32)       0
g2D)

conv2d_123 (Conv2D)         (None, 32, 32, 16)        4624

batch_normalization_81 (Bat  (None, 32, 32, 16)       64
chNormalization)

conv2d_124 (Conv2D)         (None, 32, 32, 16)        2320

max_pooling2d_30 (MaxPoolin  (None, 16, 16, 16)       0
g2D)

conv2d_125 (Conv2D)         (None, 16, 16, 8)         1160

batch_normalization_82 (Bat  (None, 16, 16, 8)        32
chNormalization)

conv2d_126 (Conv2D)         (None, 16, 16, 4)         292

max_pooling2d_31 (MaxPoolin  (None, 8, 8, 4)          0
g2D)

conv2d_127 (Conv2D)         (None, 8, 8, 4)           148

batch_normalization_83 (Bat  (None, 8, 8, 4)          16
chNormalization)

flatten_7 (Flatten)         (None, 256)               0

dense_18 (Dense)            (None, 50)                12850

dense_19 (Dense)            (None, 256)               13056
```

```
 predictions (Reshape)       (None, 8, 8, 4)           0

 conv2d_128 (Conv2D)         (None, 8, 8, 4)           148

 batch_normalization_84 (Bat  (None, 8, 8, 4)          16
 chNormalization)

 conv2d_129 (Conv2D)         (None, 8, 8, 8)           296

 up_sampling2d_28 (UpSamplin  (None, 16, 16, 8)        0
 g2D)

 conv2d_130 (Conv2D)         (None, 16, 16, 16)        1168

 batch_normalization_85 (Bat  (None, 16, 16, 16)       64
 chNormalization)

 conv2d_131 (Conv2D)         (None, 16, 16, 16)        2320

 up_sampling2d_29 (UpSamplin  (None, 32, 32, 16)       0
 g2D)

 conv2d_132 (Conv2D)         (None, 32, 32, 32)        4640

 batch_normalization_86 (Bat  (None, 32, 32, 32)       128
 chNormalization)

 conv2d_133 (Conv2D)         (None, 32, 32, 32)        9248

 up_sampling2d_30 (UpSamplin  (None, 64, 64, 32)       0
 g2D)

 conv2d_134 (Conv2D)         (None, 64, 64, 32)        9248

 batch_normalization_87 (Bat  (None, 64, 64, 32)       128
 chNormalization)

 conv2d_135 (Conv2D)         (None, 64, 64, 64)        18496

 up_sampling2d_31 (UpSamplin  (None, 128, 128, 64)     0
 g2D)

 conv2d_136 (Conv2D)         (None, 128, 128, 1)       1601

=================================================================
Total params: 116,367
Trainable params: 116,079
Non-trainable params: 288
```

----

[271]:
```
start_time = time.time()

early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)
history = autoencoder_2.fit(x_train, x_train, epochs=1000,␣
 ↪batch_size=2,shuffle=True, validation_data=(x_test,␣
 ↪x_test),callbacks=[early_stopping, reduce_lr],)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
Epoch 1/1000
320/320 [==============================] - 7s 15ms/step - loss: 0.0137 -
val_loss: 0.0081 - lr: 0.0010
Epoch 2/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0085 -
val_loss: 0.0063 - lr: 0.0010
Epoch 3/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0073 -
val_loss: 0.0052 - lr: 0.0010
Epoch 4/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0066 -
val_loss: 0.0071 - lr: 0.0010
Epoch 5/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0062 -
val_loss: 0.0044 - lr: 0.0010
Epoch 6/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0058 -
val_loss: 0.0045 - lr: 0.0010
Epoch 7/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0055 -
val_loss: 0.0044 - lr: 0.0010
Epoch 8/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0053 -
val_loss: 0.0040 - lr: 0.0010
Epoch 9/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0050 -
val_loss: 0.0043 - lr: 0.0010
Epoch 10/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0049 -
val_loss: 0.0041 - lr: 0.0010
Epoch 11/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0047 -
val_loss: 0.0038 - lr: 0.0010
Epoch 12/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0045 -
val_loss: 0.0040 - lr: 0.0010
Epoch 13/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0044 -
val_loss: 0.0045 - lr: 0.0010
Epoch 14/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0042 -
val_loss: 0.0038 - lr: 0.0010
Epoch 15/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0041 -
val_loss: 0.0035 - lr: 0.0010
Epoch 16/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0039 -
val_loss: 0.0039 - lr: 0.0010
Epoch 17/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0038 -
val_loss: 0.0035 - lr: 0.0010
Epoch 18/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0036 -
val_loss: 0.0035 - lr: 0.0010
Epoch 19/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0035 -
val_loss: 0.0034 - lr: 0.0010
Epoch 20/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0035 -
val_loss: 0.0032 - lr: 0.0010
Epoch 21/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0034 -
val_loss: 0.0034 - lr: 0.0010
Epoch 22/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0033 -
val_loss: 0.0034 - lr: 0.0010
Epoch 23/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0032 -
val_loss: 0.0033 - lr: 0.0010
Epoch 24/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0031 -
val_loss: 0.0034 - lr: 0.0010
Epoch 25/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0030 -
val_loss: 0.0037 - lr: 0.0010
Epoch 26/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0024 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 27/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0022 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 28/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0021 -
val_loss: 0.0028 - lr: 1.0000e-04
Epoch 29/1000
```

```
320/320 [==============================] - 4s 14ms/step - loss: 0.0021 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 30/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0021 -
val_loss: 0.0029 - lr: 1.0000e-04
Epoch 31/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0020 -
val_loss: 0.0028 - lr: 1.0000e-04
Epoch 32/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0028 - lr: 1.0000e-05
Epoch 33/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0028 - lr: 1.0000e-05
Epoch 34/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0029 - lr: 1.0000e-05
Epoch 35/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0028 - lr: 1.0000e-05
Epoch 36/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0028 - lr: 1.0000e-05
Epoch 37/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0028 - lr: 1.0000e-06
Epoch 38/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0028 - lr: 1.0000e-06
--- 165.2624430656433 seconds ---
```

[275]:

```
x = np.zeros((16384,))
X_proj_img = autoencoder_2.predict(dataset_3D)
X_proj_img = np.squeeze(X_proj_img, axis=-1)

for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
err_50 = x.sum()/768/128/128
err_50
```

```
24/24 [==============================] - 1s 23ms/step
```

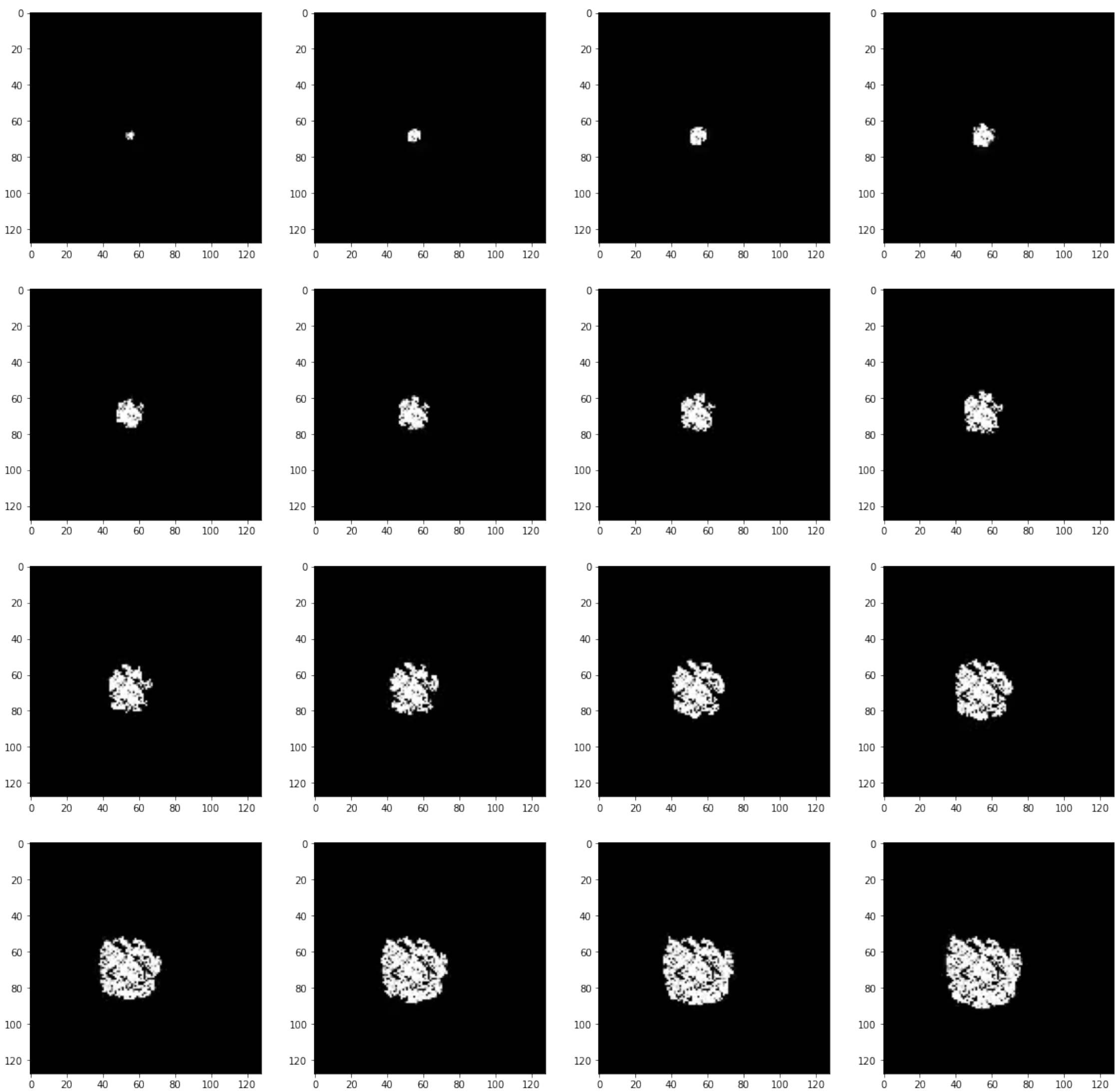
[275]: 4.729812434983852

[276]:

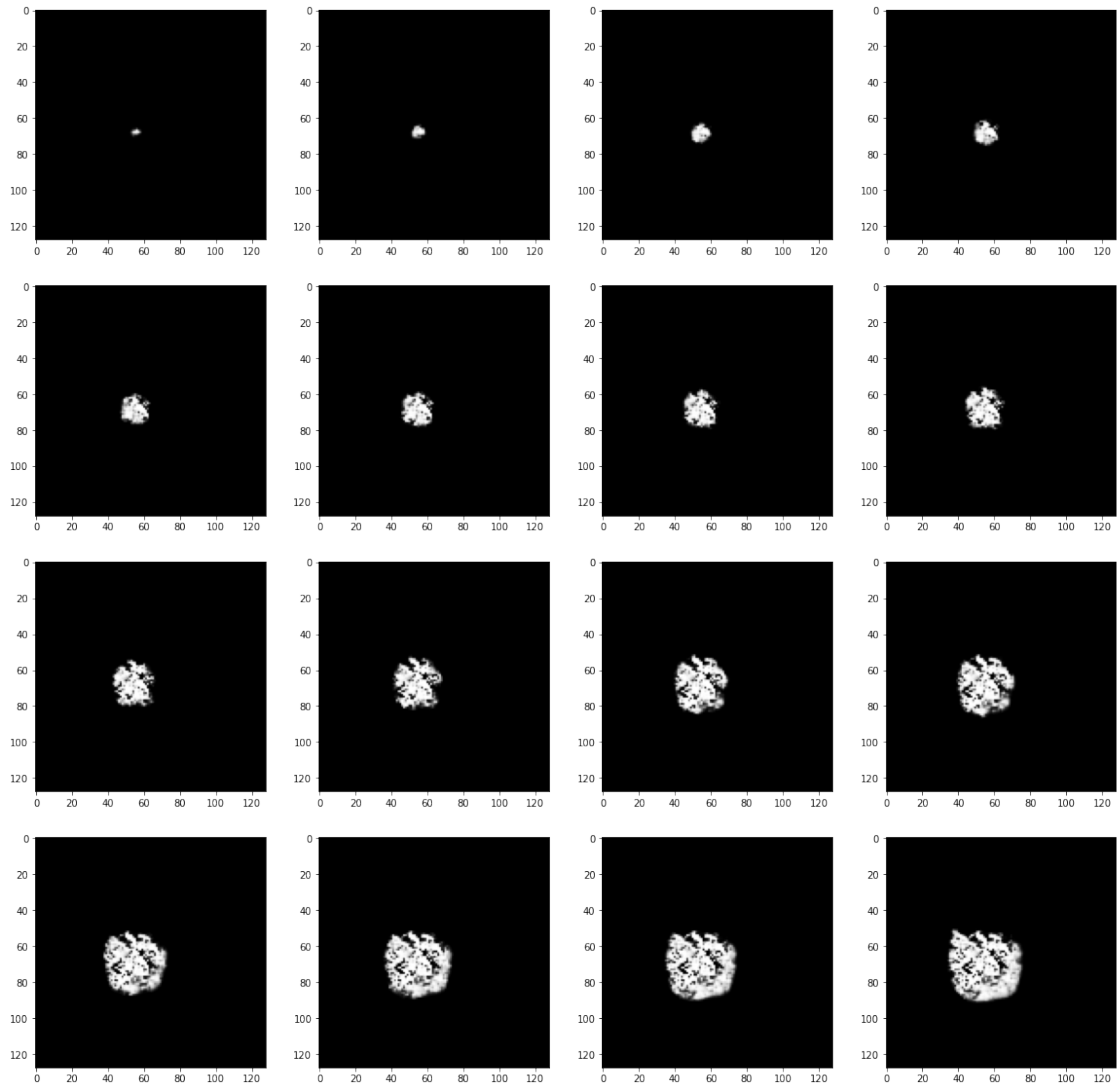
```
ext = encoded.predict(x_train)
pred = decoded.predict(ext)
#pred = autoencoder_3.predict(x_train)
```

```
plt.figure(figsize=(20, 20))
print("First Test Video")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(x_train[i, ..., 0], cmap='gray')
plt.show()
plt.figure(figsize=(20, 20))
print("Reconstruction of First Test Images")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(pred[i, ..., 0], cmap='gray')
plt.show()
```

```
20/20 [==============================] - 0s 13ms/step
20/20 [==============================] - 0s 10ms/step
First Test Video
```



```
Reconstruction of First Test Images
```



[274]:
```
pred.shape
```

[274]:
```
(640, 128, 128, 1)
```

4.3 autuencoder-1: *64*

[257]:
```
auto_input = Input(shape=(128,128,1))
encoded, x = encoder_f(auto_input,64)
```

```python
decoded, decode_output = decoder_f(x)
autoencoder_3 = Model(auto_input, decode_output)
autoencoder_3.compile(optimizer='rmsprop', loss='mse')
autoencoder_3.summary()
```

```
Model: "model_26"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_22 (InputLayer)       [(None, 128, 128, 1)]     0

 conv2d_69 (Conv2D)          (None, 128, 128, 64)      6464

 max_pooling2d_16 (MaxPoolin  (None, 64, 64, 64)       0
 g2D)

 conv2d_70 (Conv2D)          (None, 64, 64, 32)        18464

 batch_normalization_56 (Bat  (None, 64, 64, 32)       128
 chNormalization)

 conv2d_71 (Conv2D)          (None, 64, 64, 32)        9248

 max_pooling2d_17 (MaxPoolin  (None, 32, 32, 32)       0
 g2D)

 conv2d_72 (Conv2D)          (None, 32, 32, 16)        4624

 batch_normalization_57 (Bat  (None, 32, 32, 16)       64
 chNormalization)

 conv2d_73 (Conv2D)          (None, 32, 32, 16)        2320

 max_pooling2d_18 (MaxPoolin  (None, 16, 16, 16)       0
 g2D)

 conv2d_74 (Conv2D)          (None, 16, 16, 8)         1160

 batch_normalization_58 (Bat  (None, 16, 16, 8)        32
 chNormalization)

 conv2d_75 (Conv2D)          (None, 16, 16, 4)         292

 max_pooling2d_19 (MaxPoolin  (None, 8, 8, 4)          0
 g2D)

 conv2d_76 (Conv2D)          (None, 8, 8, 4)           148
```

```
batch_normalization_59 (Bat  (None, 8, 8, 4)          16
chNormalization)

flatten_4 (Flatten)         (None, 256)               0

dense_12 (Dense)            (None, 64)                16448

dense_13 (Dense)            (None, 256)               16640

predictions (Reshape)       (None, 8, 8, 4)           0

conv2d_77 (Conv2D)          (None, 8, 8, 4)           148

batch_normalization_60 (Bat  (None, 8, 8, 4)          16
chNormalization)

conv2d_78 (Conv2D)          (None, 8, 8, 8)           296

up_sampling2d_16 (UpSamplin  (None, 16, 16, 8)        0
g2D)

conv2d_79 (Conv2D)          (None, 16, 16, 16)        1168

batch_normalization_61 (Bat  (None, 16, 16, 16)       64
chNormalization)

conv2d_80 (Conv2D)          (None, 16, 16, 16)        2320

up_sampling2d_17 (UpSamplin  (None, 32, 32, 16)       0
g2D)

conv2d_81 (Conv2D)          (None, 32, 32, 32)        4640

batch_normalization_62 (Bat  (None, 32, 32, 32)       128
chNormalization)

conv2d_82 (Conv2D)          (None, 32, 32, 32)        9248

up_sampling2d_18 (UpSamplin  (None, 64, 64, 32)       0
g2D)

conv2d_83 (Conv2D)          (None, 64, 64, 32)        9248

batch_normalization_63 (Bat  (None, 64, 64, 32)       128
chNormalization)

conv2d_84 (Conv2D)          (None, 64, 64, 64)        18496
```

```
 up_sampling2d_19 (UpSamplin  (None, 128, 128, 64)     0
 g2D)

 conv2d_85 (Conv2D)          (None, 128, 128, 1)       1601

=================================================================
Total params: 123,549
Trainable params: 123,261
Non-trainable params: 288
_________________________________________________________________
```

[258]:
```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)
history = autoencoder_3.fit(x_train, x_train, epochs=1000,␣
 ↪batch_size=2,shuffle=True, validation_data=(x_test,␣
 ↪x_test),callbacks=[early_stopping, reduce_lr],)
```

```
Epoch 1/1000
320/320 [==============================] - 7s 14ms/step - loss: 0.0135 -
val_loss: 0.0061 - lr: 0.0010
Epoch 2/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0083 -
val_loss: 0.0063 - lr: 0.0010
Epoch 3/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0071 -
val_loss: 0.0067 - lr: 0.0010
Epoch 4/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0065 -
val_loss: 0.0049 - lr: 0.0010
Epoch 5/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0059 -
val_loss: 0.0046 - lr: 0.0010
Epoch 6/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0054 -
val_loss: 0.0053 - lr: 0.0010
Epoch 7/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0052 -
val_loss: 0.0043 - lr: 0.0010
Epoch 8/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0049 -
val_loss: 0.0040 - lr: 0.0010
Epoch 9/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0047 -
val_loss: 0.0039 - lr: 0.0010
Epoch 10/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0044 -
val_loss: 0.0040 - lr: 0.0010
Epoch 11/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0042 -
val_loss: 0.0038 - lr: 0.0010
Epoch 12/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0040 -
val_loss: 0.0033 - lr: 0.0010
Epoch 13/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0038 -
val_loss: 0.0040 - lr: 0.0010
Epoch 14/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0037 -
val_loss: 0.0033 - lr: 0.0010
Epoch 15/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0035 -
val_loss: 0.0034 - lr: 0.0010
Epoch 16/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0034 -
val_loss: 0.0033 - lr: 0.0010
Epoch 17/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0033 -
val_loss: 0.0032 - lr: 0.0010
Epoch 18/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0032 -
val_loss: 0.0036 - lr: 0.0010
Epoch 19/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0031 -
val_loss: 0.0030 - lr: 0.0010
Epoch 20/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0030 -
val_loss: 0.0032 - lr: 0.0010
Epoch 21/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0030 -
val_loss: 0.0030 - lr: 0.0010
Epoch 22/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0029 -
val_loss: 0.0032 - lr: 0.0010
Epoch 23/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0028 -
val_loss: 0.0034 - lr: 0.0010
Epoch 24/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0034 - lr: 0.0010
Epoch 25/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0021 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 26/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 27/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 28/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 29/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 30/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 31/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0016 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 32/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0016 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 33/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0016 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 34/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0016 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 35/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0016 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 36/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0016 -
val_loss: 0.0027 - lr: 1.0000e-06
```

[259]:

```
x = np.zeros((16384,))
X_proj_img = autoencoder_3.predict(dataset_3D)
X_proj_img = np.squeeze(X_proj_img, axis=-1)

for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
err_64 = x.sum()/768/128/128
```

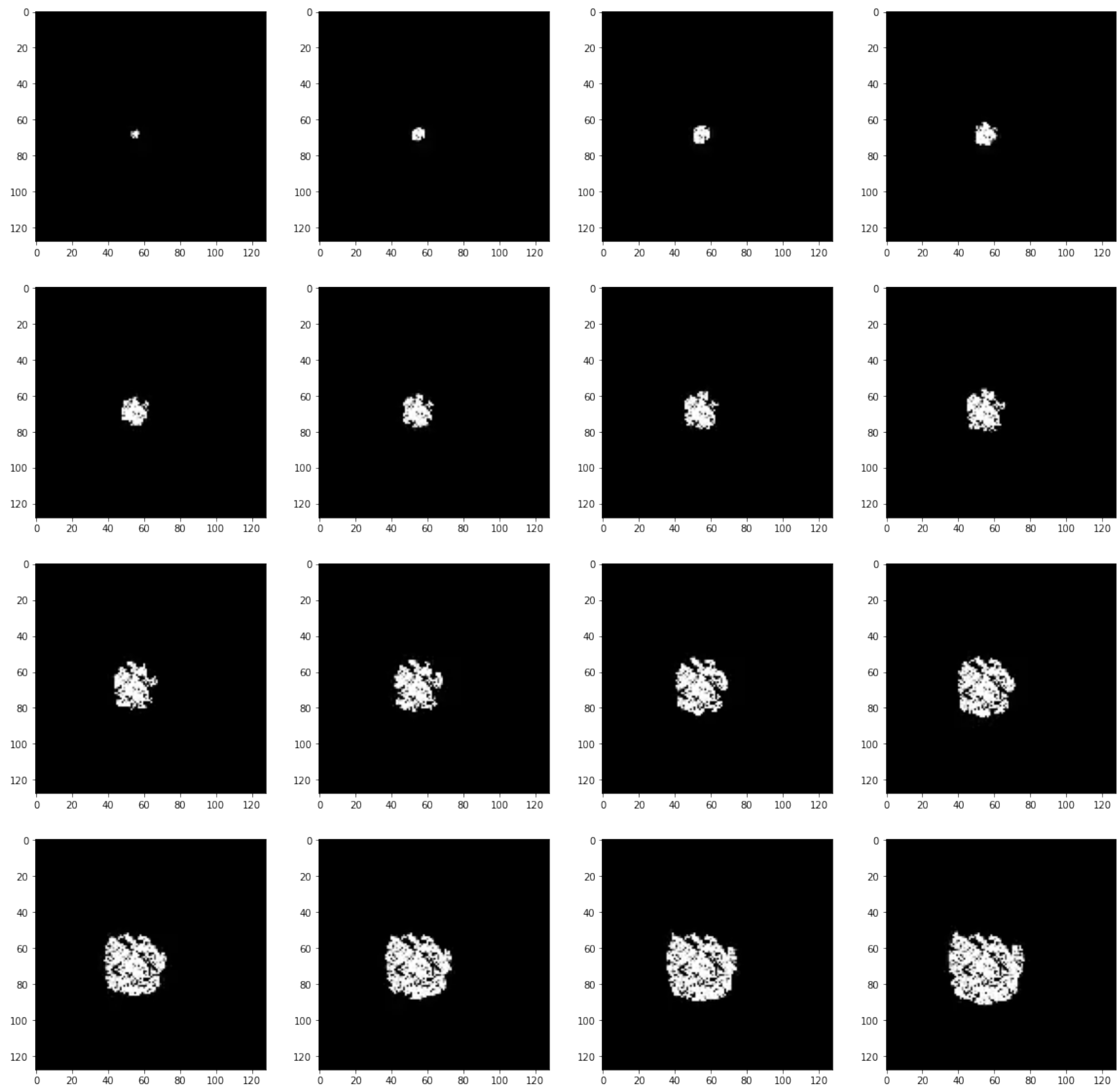
```
24/24 [==============================] - 1s 23ms/step
```

[260]:

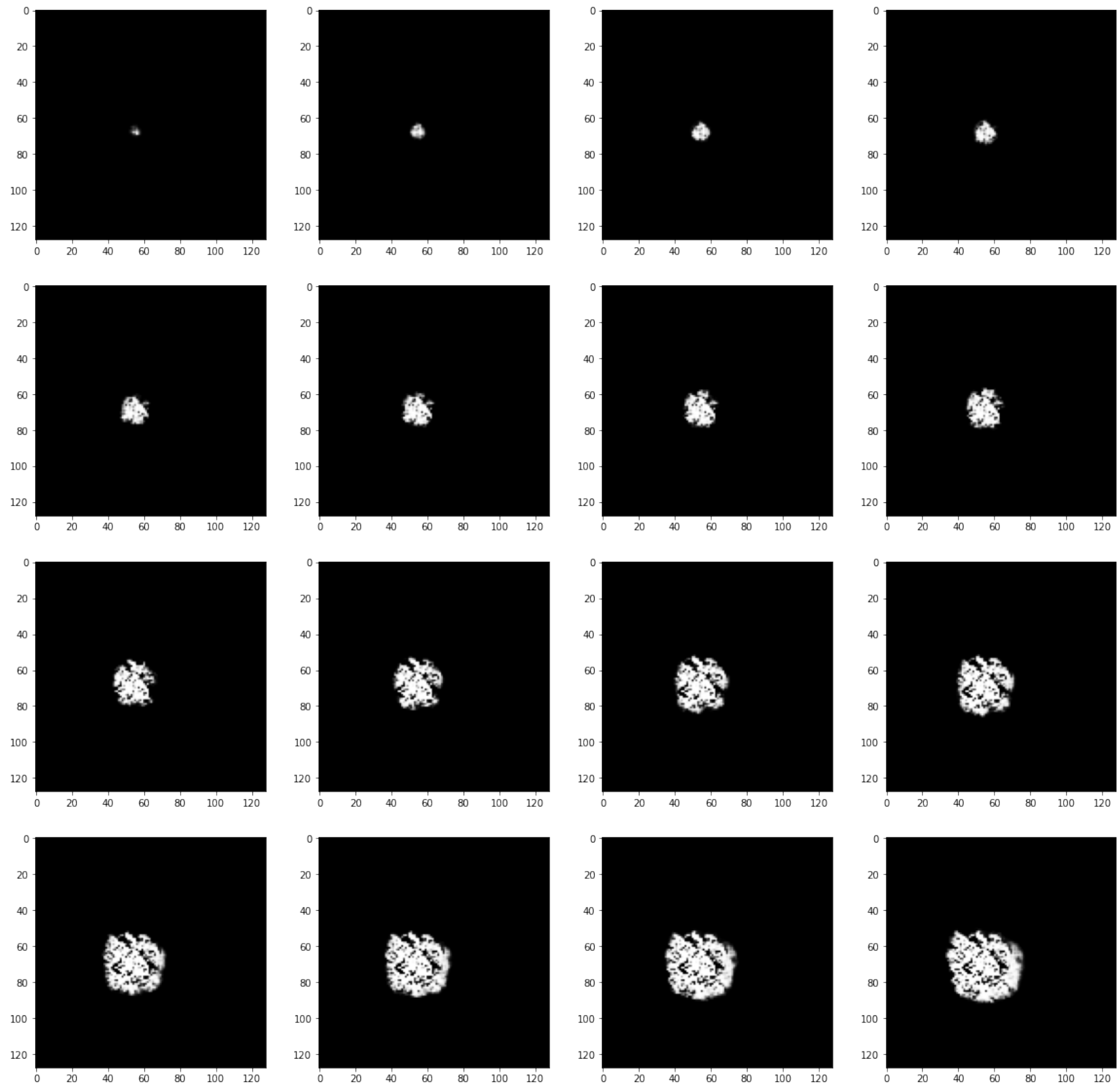
```
pred = autoencoder_3.predict(x_train)
plt.figure(figsize=(20, 20))
print("First Test Video")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(x_train[i, ..., 0], cmap='gray')
plt.show()
```

```python
plt.figure(figsize=(20, 20))
print("Reconstruction of First Test Images")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(pred[i, ..., 0], cmap='gray')
plt.show()
```

```
20/20 [==============================] - 1s 21ms/step
First Test Video
```



```
Reconstruction of First Test Images
```



4.4 autoencoder - 3: 96

[261]:
```
auto_input = Input(shape=(128,128,1))
encoded, x = encoder_f(auto_input,96)

decoded, decode_output = decoder_f(x)
autoencoder_4 = Model(auto_input, decode_output)
autoencoder_4.compile(optimizer='rmsprop', loss='mse')
autoencoder_4.summary()
```

```
Model: "model_29"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
input_24 (InputLayer)       [(None, 128, 128, 1)]     0

conv2d_86 (Conv2D)          (None, 128, 128, 64)      6464

max_pooling2d_20 (MaxPoolin  (None, 64, 64, 64)       0
g2D)

conv2d_87 (Conv2D)          (None, 64, 64, 32)        18464

batch_normalization_64 (Bat  (None, 64, 64, 32)       128
chNormalization)

conv2d_88 (Conv2D)          (None, 64, 64, 32)        9248

max_pooling2d_21 (MaxPoolin  (None, 32, 32, 32)       0
g2D)

conv2d_89 (Conv2D)          (None, 32, 32, 16)        4624

batch_normalization_65 (Bat  (None, 32, 32, 16)       64
chNormalization)

conv2d_90 (Conv2D)          (None, 32, 32, 16)        2320

max_pooling2d_22 (MaxPoolin  (None, 16, 16, 16)       0
g2D)

conv2d_91 (Conv2D)          (None, 16, 16, 8)         1160

batch_normalization_66 (Bat  (None, 16, 16, 8)        32
chNormalization)

conv2d_92 (Conv2D)          (None, 16, 16, 4)         292

max_pooling2d_23 (MaxPoolin  (None, 8, 8, 4)          0
g2D)

conv2d_93 (Conv2D)          (None, 8, 8, 4)           148

batch_normalization_67 (Bat  (None, 8, 8, 4)          16
chNormalization)

flatten_5 (Flatten)         (None, 256)               0

dense_14 (Dense)            (None, 96)                24672

dense_15 (Dense)            (None, 256)               24832
```

```
 predictions (Reshape)       (None, 8, 8, 4)           0

 conv2d_94 (Conv2D)          (None, 8, 8, 4)           148

 batch_normalization_68 (Bat  (None, 8, 8, 4)          16
 chNormalization)

 conv2d_95 (Conv2D)          (None, 8, 8, 8)           296

 up_sampling2d_20 (UpSamplin  (None, 16, 16, 8)        0
 g2D)

 conv2d_96 (Conv2D)          (None, 16, 16, 16)        1168

 batch_normalization_69 (Bat  (None, 16, 16, 16)       64
 chNormalization)

 conv2d_97 (Conv2D)          (None, 16, 16, 16)        2320

 up_sampling2d_21 (UpSamplin  (None, 32, 32, 16)       0
 g2D)

 conv2d_98 (Conv2D)          (None, 32, 32, 32)        4640

 batch_normalization_70 (Bat  (None, 32, 32, 32)       128
 chNormalization)

 conv2d_99 (Conv2D)          (None, 32, 32, 32)        9248

 up_sampling2d_22 (UpSamplin  (None, 64, 64, 32)       0
 g2D)

 conv2d_100 (Conv2D)         (None, 64, 64, 32)        9248

 batch_normalization_71 (Bat  (None, 64, 64, 32)       128
 chNormalization)

 conv2d_101 (Conv2D)         (None, 64, 64, 64)        18496

 up_sampling2d_23 (UpSamplin  (None, 128, 128, 64)     0
 g2D)

 conv2d_102 (Conv2D)         (None, 128, 128, 1)       1601

=================================================================
Total params: 139,965
Trainable params: 139,677
Non-trainable params: 288
```

----

[262]:
```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)
history = autoencoder_4.fit(x_train, x_train, epochs=1000,␣
 ↪batch_size=2,shuffle=True, validation_data=(x_test,␣
 ↪x_test),callbacks=[early_stopping, reduce_lr],)
```

```
Epoch 1/1000
320/320 [==============================] - 7s 14ms/step - loss: 0.0150 -
val_loss: 0.0077 - lr: 0.0010
Epoch 2/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0088 -
val_loss: 0.0058 - lr: 0.0010
Epoch 3/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0073 -
val_loss: 0.0062 - lr: 0.0010
Epoch 4/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0065 -
val_loss: 0.0051 - lr: 0.0010
Epoch 5/1000
320/320 [==============================] - 6s 18ms/step - loss: 0.0061 -
val_loss: 0.0044 - lr: 0.0010
Epoch 6/1000
320/320 [==============================] - 5s 16ms/step - loss: 0.0057 -
val_loss: 0.0047 - lr: 0.0010
Epoch 7/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0054 -
val_loss: 0.0053 - lr: 0.0010
Epoch 8/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0050 -
val_loss: 0.0046 - lr: 0.0010
Epoch 9/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0047 -
val_loss: 0.0043 - lr: 0.0010
Epoch 10/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0045 -
val_loss: 0.0045 - lr: 0.0010
Epoch 11/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0036 -
val_loss: 0.0034 - lr: 1.0000e-04
Epoch 12/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0033 -
val_loss: 0.0034 - lr: 1.0000e-04
Epoch 13/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0032 -
val_loss: 0.0034 - lr: 1.0000e-04
Epoch 14/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0031 -
val_loss: 0.0034 - lr: 1.0000e-04
Epoch 15/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0030 -
val_loss: 0.0033 - lr: 1.0000e-04
Epoch 16/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0030 -
val_loss: 0.0033 - lr: 1.0000e-04
Epoch 17/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0029 -
val_loss: 0.0033 - lr: 1.0000e-04
Epoch 18/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0029 -
val_loss: 0.0032 - lr: 1.0000e-04
Epoch 19/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0028 -
val_loss: 0.0032 - lr: 1.0000e-04
Epoch 20/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0028 -
val_loss: 0.0032 - lr: 1.0000e-04
Epoch 21/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-05
Epoch 22/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-05
Epoch 23/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-05
Epoch 24/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-05
Epoch 25/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-05
Epoch 26/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-06
Epoch 27/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-06
Epoch 28/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-06
Epoch 29/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-06
Epoch 30/1000
```

```
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-06
Epoch 31/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-07
Epoch 32/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-07
Epoch 33/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-07
Epoch 34/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-07
Epoch 35/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-07
Epoch 36/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-08
Epoch 37/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-08
Epoch 38/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-08
Epoch 39/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-08
Epoch 40/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-08
Epoch 41/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-09
Epoch 42/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0027 -
val_loss: 0.0032 - lr: 1.0000e-09
```

[263]:

```
x = np.zeros((16384,))
X_proj_img = autoencoder_4.predict(dataset_3D)
X_proj_img = np.squeeze(X_proj_img, axis=-1)

for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
err_96 = x.sum()/768/128/128
```

```
24/24 [==============================] - 1s 23ms/step
```

4.5 autoencoder - 3: 128

[264]:
```
auto_input = Input(shape=(128,128,1))
encoded, x = encoder_f(auto_input,128)

decoded, decode_output = decoder_f(x)
autoencoder_5 = Model(auto_input, decode_output)
autoencoder_5.compile(optimizer='rmsprop', loss='mse')
autoencoder_5.summary()
```

```
Model: "model_32"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_26 (InputLayer)       [(None, 128, 128, 1)]     0

 conv2d_103 (Conv2D)         (None, 128, 128, 64)      6464

 max_pooling2d_24 (MaxPoolin  (None, 64, 64, 64)       0
 g2D)

 conv2d_104 (Conv2D)         (None, 64, 64, 32)        18464

 batch_normalization_72 (Bat  (None, 64, 64, 32)       128
 chNormalization)

 conv2d_105 (Conv2D)         (None, 64, 64, 32)        9248

 max_pooling2d_25 (MaxPoolin  (None, 32, 32, 32)       0
 g2D)

 conv2d_106 (Conv2D)         (None, 32, 32, 16)        4624

 batch_normalization_73 (Bat  (None, 32, 32, 16)       64
 chNormalization)

 conv2d_107 (Conv2D)         (None, 32, 32, 16)        2320

 max_pooling2d_26 (MaxPoolin  (None, 16, 16, 16)       0
 g2D)

 conv2d_108 (Conv2D)         (None, 16, 16, 8)         1160

 batch_normalization_74 (Bat  (None, 16, 16, 8)        32
 chNormalization)
```

```
conv2d_109 (Conv2D)         (None, 16, 16, 4)         292

max_pooling2d_27 (MaxPoolin  (None, 8, 8, 4)          0
g2D)

conv2d_110 (Conv2D)         (None, 8, 8, 4)           148

batch_normalization_75 (Bat  (None, 8, 8, 4)          16
chNormalization)

flatten_6 (Flatten)         (None, 256)               0

dense_16 (Dense)            (None, 128)               32896

dense_17 (Dense)            (None, 256)               33024

predictions (Reshape)       (None, 8, 8, 4)           0

conv2d_111 (Conv2D)         (None, 8, 8, 4)           148

batch_normalization_76 (Bat  (None, 8, 8, 4)          16
chNormalization)

conv2d_112 (Conv2D)         (None, 8, 8, 8)           296

up_sampling2d_24 (UpSamplin  (None, 16, 16, 8)        0
g2D)

conv2d_113 (Conv2D)         (None, 16, 16, 16)        1168

batch_normalization_77 (Bat  (None, 16, 16, 16)       64
chNormalization)

conv2d_114 (Conv2D)         (None, 16, 16, 16)        2320

up_sampling2d_25 (UpSamplin  (None, 32, 32, 16)       0
g2D)

conv2d_115 (Conv2D)         (None, 32, 32, 32)        4640

batch_normalization_78 (Bat  (None, 32, 32, 32)       128
chNormalization)

conv2d_116 (Conv2D)         (None, 32, 32, 32)        9248

up_sampling2d_26 (UpSamplin  (None, 64, 64, 32)       0
g2D)
```

```
 conv2d_117 (Conv2D)         (None, 64, 64, 32)        9248

 batch_normalization_79 (Bat  (None, 64, 64, 32)       128
 chNormalization)

 conv2d_118 (Conv2D)         (None, 64, 64, 64)        18496

 up_sampling2d_27 (UpSamplin  (None, 128, 128, 64)     0
 g2D)

 conv2d_119 (Conv2D)         (None, 128, 128, 1)       1601

=================================================================
Total params: 156,381
Trainable params: 156,093
Non-trainable params: 288
_________________________________________________________________
```

[265]:
```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)
history = autoencoder_5.fit(x_train, x_train, epochs=1000,␣
 ↪batch_size=2,shuffle=True, validation_data=(x_test,␣
 ↪x_test),callbacks=[early_stopping, reduce_lr],)
```

```
Epoch 1/1000
320/320 [==============================] - 7s 14ms/step - loss: 0.0133 -
val_loss: 0.0071 - lr: 0.0010
Epoch 2/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0082 -
val_loss: 0.0075 - lr: 0.0010
Epoch 3/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0071 -
val_loss: 0.0092 - lr: 0.0010
Epoch 4/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0064 -
val_loss: 0.0054 - lr: 0.0010
Epoch 5/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0059 -
val_loss: 0.0039 - lr: 0.0010
Epoch 6/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0055 -
val_loss: 0.0041 - lr: 0.0010
Epoch 7/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0051 -
val_loss: 0.0042 - lr: 0.0010
Epoch 8/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0049 -
val_loss: 0.0038 - lr: 0.0010
```

```
Epoch 9/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0046 -
val_loss: 0.0037 - lr: 0.0010
Epoch 10/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0043 -
val_loss: 0.0040 - lr: 0.0010
Epoch 11/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0042 -
val_loss: 0.0034 - lr: 0.0010
Epoch 12/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0040 -
val_loss: 0.0038 - lr: 0.0010
Epoch 13/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0038 -
val_loss: 0.0035 - lr: 0.0010
Epoch 14/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0037 -
val_loss: 0.0033 - lr: 0.0010
Epoch 15/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0035 -
val_loss: 0.0035 - lr: 0.0010
Epoch 16/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0034 -
val_loss: 0.0032 - lr: 0.0010
Epoch 17/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0033 -
val_loss: 0.0034 - lr: 0.0010
Epoch 18/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0032 -
val_loss: 0.0033 - lr: 0.0010
Epoch 19/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0031 -
val_loss: 0.0031 - lr: 0.0010
Epoch 20/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0030 -
val_loss: 0.0032 - lr: 0.0010
Epoch 21/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0029 -
val_loss: 0.0031 - lr: 0.0010
Epoch 22/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0022 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 23/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0020 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 24/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0027 - lr: 1.0000e-04
```

```
Epoch 25/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0019 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 26/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0019 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 27/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0018 -
val_loss: 0.0027 - lr: 1.0000e-04
Epoch 28/1000
320/320 [==============================] - 4s 14ms/step - loss: 0.0018 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 29/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0018 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 30/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 31/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 32/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0026 - lr: 1.0000e-05
Epoch 33/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 34/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 35/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 36/1000
320/320 [==============================] - 5s 15ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 37/1000
320/320 [==============================] - 6s 20ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-05
Epoch 38/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-06
Epoch 39/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-06
Epoch 40/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-06
```

```
Epoch 41/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-06
Epoch 42/1000
320/320 [==============================] - 4s 13ms/step - loss: 0.0017 -
val_loss: 0.0027 - lr: 1.0000e-06
```

[266]:

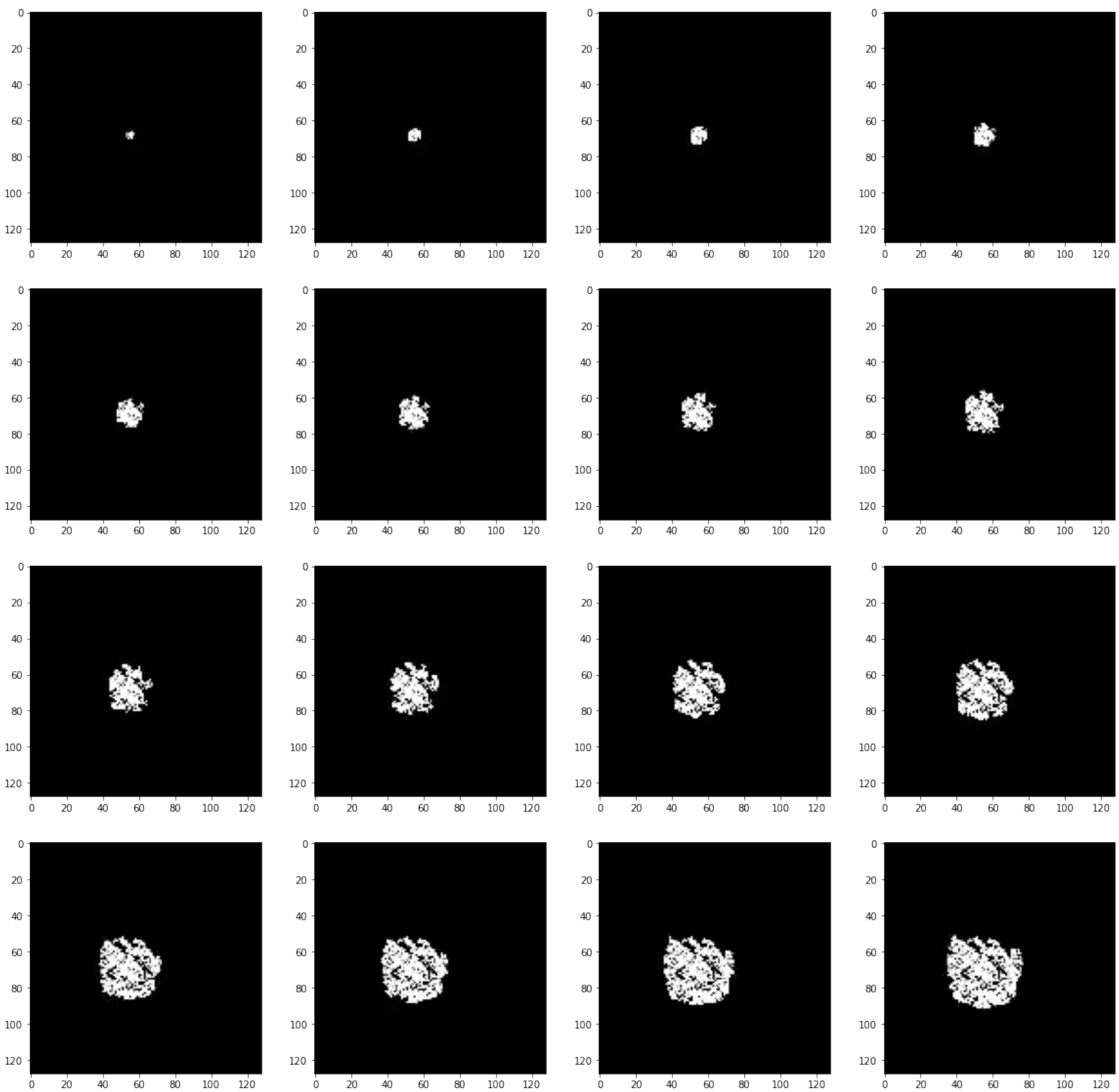
```
x = np.zeros((16384,))
X_proj_img = autoencoder_5.predict(dataset_3D)
X_proj_img = np.squeeze(X_proj_img, axis=-1)

for i in range(768):
  diff = np.abs((X_proj_img[i]-dataset_3D[i])).flatten()
  x+=diff
err_128 = x.sum()/768/128/128
```

```
24/24 [==============================] - 1s 23ms/step
```

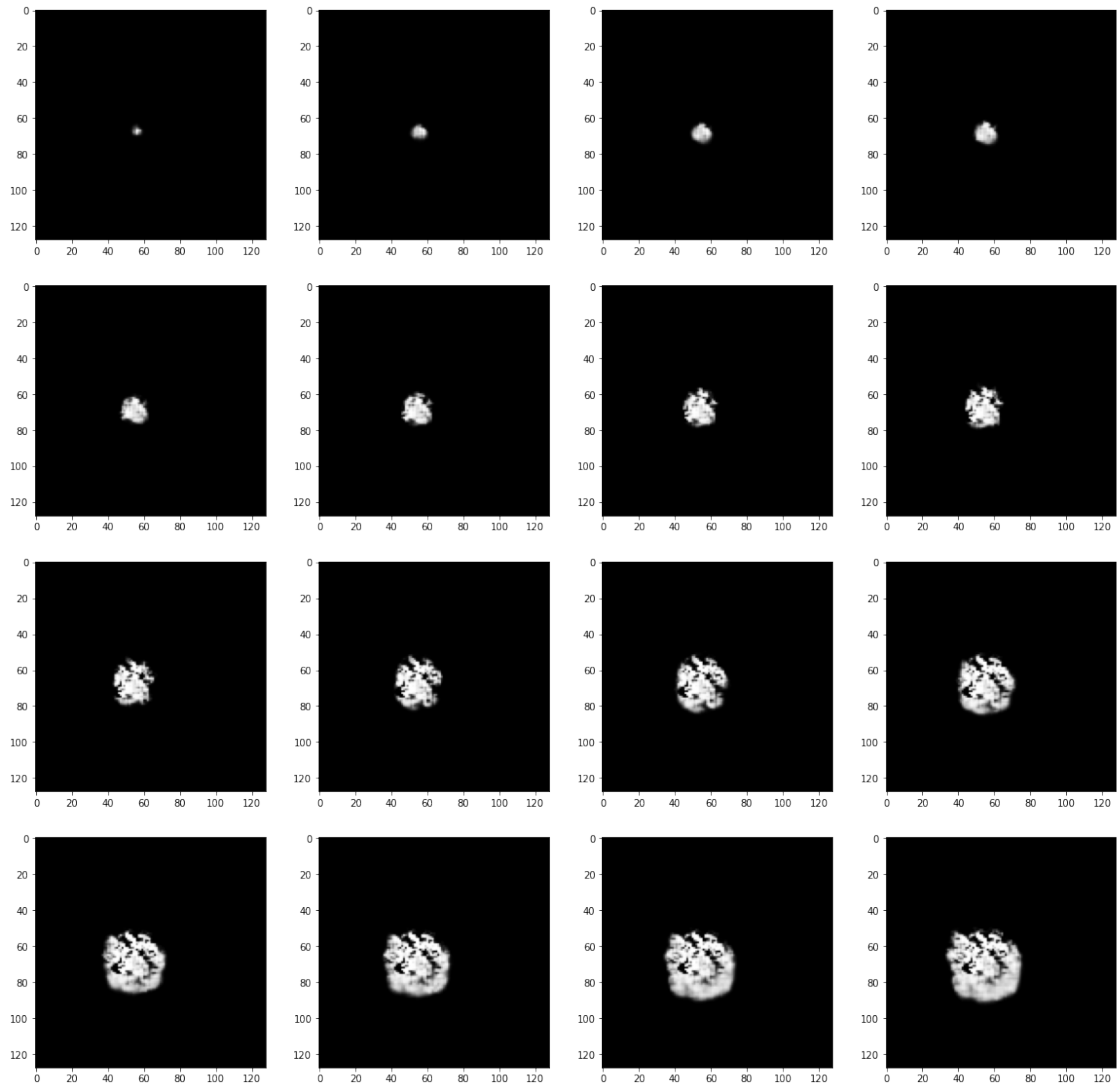
[267]:

```
pred = autoencoder_4.predict(x_train)
plt.figure(figsize=(20, 20))
print("First Test Video")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(x_train[i, ..., 0], cmap='gray')
plt.show()
plt.figure(figsize=(20, 20))
print("Reconstruction of First Test Images")
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.imshow(pred[i, ..., 0], cmap='gray')
plt.show()
```

```
20/20 [==============================] - 1s 22ms/step
First Test Video
```



`Reconstruction of First Test Images`



[277]:
```
print(err_32)
print(err_50)
print(err_64)
print(err_96)
print(err_128)
```

```
4.894783733018802
4.729812434983852
4.885402476595164
4.766555618458457
4.818805018100924
```

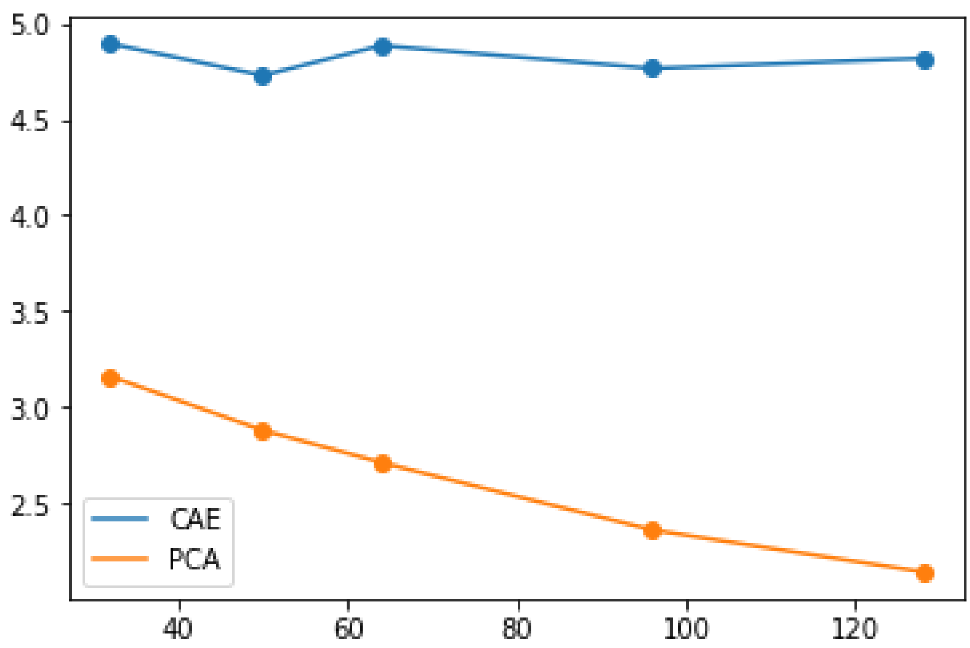
4.6 Figure for Dimensionality Reduction into 32, 50, 64, 96 and 128, respectively on PCA and ConvAutuencoder

The results shows the average error between two sets of data, restruction snapshots and original snapshots, using the L1 norm.

The math formula is

$$\frac{\left\|\mathbf{y}-\mathbf{y}_{\{\mathrm{PCA},\mathrm{CAE}\}}\right\|_1}{\dim(\mathbf{y})}$$

[278]:
```
N = [32, 50, 64, 96, 128]
res_1 = [err_32, err_50, err_64, err_96, err_128]
res_2 = [pca_err_32, pca_err_50,pca_err_64,pca_err_96,pca_err_128]
plt.scatter(N, res_1)
plt.scatter(N, res_2)

plt.plot(N, res_1, label = 'CAE')
plt.plot(N, res_2, label = 'PCA')
plt.legend()
plt.show()
```



5 2. CAE+LSTM

[151]:
```
train = np.array(encoded(x_train))
np.save('train.npy',train)

test = np.array(encoded(x_test))
np.save('test.npy',test)
data_lstm = np.array(encoded(X))
```

[152]:
```
#train = np.load('/content/sample_data/train.npy')
#test = np.load('/content/sample_data/test.npy')

scaler = MinMaxScaler(feature_range=(0, 1))

train_sca = scaler.fit_transform(train).astype('float32')
test_sca = scaler.transform(test).astype('float32')
slices_train = np.split(train_sca, len(train)/4)
train_dataset = np.stack(slices_train, axis=0)
slices_test = np.split(test_sca, len(test)/4)
val_dataset = np.stack(slices_test, axis=0)
```

[153]:
```
def create_shifted_frames(data):
    x = data[0 : data.shape[0] - 1, :, :]
    y = data[1 : data.shape[0], :, :]
    return x, y
x_train, y_train = create_shifted_frames(train_dataset)
x_val, y_val = create_shifted_frames(val_dataset)
x_train.shape
```

[153]:
```
(159, 4, 50)
```

[120]:
```
model = Sequential()

model.add(LSTM(64,input_shape=(4,50)))
model.add(Dropout(0.3))

model.add(RepeatVector(4))

#multi-step
model.add(LSTM(64, activation='relu', return_sequences=True))
#model.add(BatchNormalization())

model.add(Dropout(0.3))
#model.add(BatchNormalization())


#model.add(Dense(50))
```

```
model.add(Dense(50))
#model.add(BatchNormalization())


model.add(TimeDistributed(Dense(50)))

model.add(Activation('relu'))

##################################################################
#training

model.compile(loss='mse', optimizer='rmsprop', metrics=['mae'])
model.summary()
```

```
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

```
Model: "sequential_1"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 64)                29440

 dropout_2 (Dropout)         (None, 64)                0

 repeat_vector_1 (RepeatVect  (None, 4, 64)            0
 or)

 lstm_3 (LSTM)               (None, 4, 64)             33024

 dropout_3 (Dropout)         (None, 4, 64)             0

 dense_6 (Dense)             (None, 4, 50)             3250

 time_distributed_1 (TimeDis  (None, 4, 50)            2550
 tributed)

 activation_1 (Activation)   (None, 4, 50)             0

=================================================================
Total params: 68,264
Trainable params: 68,264
Non-trainable params: 0
_________________________________________________________________
```

[121]:
```
start_time = time.time()
```

```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)
history = model.fit(x_train, y_train,validation_data=(x_val, y_val),␣
 ↪epochs=100,callbacks=[early_stopping, reduce_lr],batch_size=1, verbose=1)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
Epoch 1/100
159/159 [==============================] - 4s 12ms/step - loss: 0.0967 - mae:
0.2480 - val_loss: 0.0582 - val_mae: 0.1883 - lr: 0.0010
Epoch 2/100
159/159 [==============================] - 2s 9ms/step - loss: 0.0634 - mae:
0.1966 - val_loss: 0.0492 - val_mae: 0.1671 - lr: 0.0010
Epoch 3/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0573 - mae:
0.1853 - val_loss: 0.0491 - val_mae: 0.1671 - lr: 0.0010
Epoch 4/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0534 - mae:
0.1767 - val_loss: 0.0478 - val_mae: 0.1635 - lr: 0.0010
Epoch 5/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0494 - mae:
0.1678 - val_loss: 0.0430 - val_mae: 0.1517 - lr: 0.0010
Epoch 6/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0437 - mae:
0.1583 - val_loss: 0.0376 - val_mae: 0.1442 - lr: 0.0010
Epoch 7/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0308 - mae:
0.1382 - val_loss: 0.0225 - val_mae: 0.1175 - lr: 0.0010
Epoch 8/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0268 - mae:
0.1293 - val_loss: 0.0221 - val_mae: 0.1154 - lr: 0.0010
Epoch 9/100
159/159 [==============================] - 2s 15ms/step - loss: 0.0257 - mae:
0.1260 - val_loss: 0.0205 - val_mae: 0.1115 - lr: 0.0010
Epoch 10/100
159/159 [==============================] - 2s 14ms/step - loss: 0.0252 - mae:
0.1250 - val_loss: 0.0225 - val_mae: 0.1165 - lr: 0.0010
Epoch 11/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0240 - mae:
0.1216 - val_loss: 0.0287 - val_mae: 0.1342 - lr: 0.0010
Epoch 12/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0237 - mae:
0.1206 - val_loss: 0.0207 - val_mae: 0.1111 - lr: 0.0010
Epoch 13/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0225 - mae:
0.1178 - val_loss: 0.0189 - val_mae: 0.1067 - lr: 0.0010
Epoch 14/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0219 - mae:
0.1158 - val_loss: 0.0195 - val_mae: 0.1076 - lr: 0.0010
```

```
Epoch 15/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0217 - mae:
0.1154 - val_loss: 0.0182 - val_mae: 0.1017 - lr: 0.0010
Epoch 16/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0214 - mae:
0.1144 - val_loss: 0.0192 - val_mae: 0.1060 - lr: 0.0010
Epoch 17/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0205 - mae:
0.1108 - val_loss: 0.0198 - val_mae: 0.1074 - lr: 0.0010
Epoch 18/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0204 - mae:
0.1107 - val_loss: 0.0192 - val_mae: 0.1059 - lr: 0.0010
Epoch 19/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0209 - mae:
0.1120 - val_loss: 0.0187 - val_mae: 0.1044 - lr: 0.0010
Epoch 20/100
159/159 [==============================] - 2s 9ms/step - loss: 0.0200 - mae:
0.1098 - val_loss: 0.0180 - val_mae: 0.0995 - lr: 0.0010
Epoch 21/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0196 - mae:
0.1085 - val_loss: 0.0184 - val_mae: 0.1024 - lr: 0.0010
Epoch 22/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0192 - mae:
0.1076 - val_loss: 0.0173 - val_mae: 0.0996 - lr: 0.0010
Epoch 23/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0187 - mae:
0.1063 - val_loss: 0.0178 - val_mae: 0.1003 - lr: 0.0010
Epoch 24/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0192 - mae:
0.1074 - val_loss: 0.0184 - val_mae: 0.1025 - lr: 0.0010
Epoch 25/100
159/159 [==============================] - 2s 9ms/step - loss: 0.0189 - mae:
0.1066 - val_loss: 0.0182 - val_mae: 0.1001 - lr: 0.0010
Epoch 26/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0192 - mae:
0.1076 - val_loss: 0.0193 - val_mae: 0.1043 - lr: 0.0010
Epoch 27/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0185 - mae:
0.1056 - val_loss: 0.0178 - val_mae: 0.1004 - lr: 0.0010
Epoch 28/100
159/159 [==============================] - 2s 9ms/step - loss: 0.0168 - mae:
0.1001 - val_loss: 0.0167 - val_mae: 0.0970 - lr: 1.0000e-04
Epoch 29/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0162 - mae:
0.0984 - val_loss: 0.0165 - val_mae: 0.0961 - lr: 1.0000e-04
Epoch 30/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0160 - mae:
0.0978 - val_loss: 0.0165 - val_mae: 0.0958 - lr: 1.0000e-04
```

```
Epoch 31/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0160 - mae:
0.0972 - val_loss: 0.0166 - val_mae: 0.0954 - lr: 1.0000e-04
Epoch 32/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0156 - mae:
0.0963 - val_loss: 0.0169 - val_mae: 0.0962 - lr: 1.0000e-04
Epoch 33/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0154 - mae:
0.0959 - val_loss: 0.0164 - val_mae: 0.0954 - lr: 1.0000e-04
Epoch 34/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0154 - mae:
0.0957 - val_loss: 0.0162 - val_mae: 0.0947 - lr: 1.0000e-04
Epoch 35/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0153 - mae:
0.0957 - val_loss: 0.0161 - val_mae: 0.0943 - lr: 1.0000e-04
Epoch 36/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0153 - mae:
0.0949 - val_loss: 0.0161 - val_mae: 0.0942 - lr: 1.0000e-04
Epoch 37/100
159/159 [==============================] - 2s 9ms/step - loss: 0.0157 - mae:
0.0965 - val_loss: 0.0159 - val_mae: 0.0935 - lr: 1.0000e-04
Epoch 38/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0151 - mae:
0.0946 - val_loss: 0.0163 - val_mae: 0.0944 - lr: 1.0000e-04
Epoch 39/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0155 - mae:
0.0963 - val_loss: 0.0160 - val_mae: 0.0940 - lr: 1.0000e-04
Epoch 40/100
159/159 [==============================] - 2s 10ms/step - loss: 0.0152 - mae:
0.0948 - val_loss: 0.0159 - val_mae: 0.0934 - lr: 1.0000e-04
Epoch 41/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0150 - mae:
0.0946 - val_loss: 0.0161 - val_mae: 0.0938 - lr: 1.0000e-04
Epoch 42/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0153 - mae:
0.0949 - val_loss: 0.0159 - val_mae: 0.0934 - lr: 1.0000e-04
Epoch 43/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0149 - mae:
0.0941 - val_loss: 0.0159 - val_mae: 0.0932 - lr: 1.0000e-05
Epoch 44/100
159/159 [==============================] - 2s 9ms/step - loss: 0.0149 - mae:
0.0941 - val_loss: 0.0159 - val_mae: 0.0932 - lr: 1.0000e-05
Epoch 45/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0151 - mae:
0.0945 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-05
Epoch 46/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0148 - mae:
0.0934 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-05
```

```
Epoch 47/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0150 - mae:
0.0942 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-05
Epoch 48/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0147 - mae:
0.0932 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-06
Epoch 49/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0146 - mae:
0.0928 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-06
Epoch 50/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0148 - mae:
0.0932 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-06
Epoch 51/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0150 - mae:
0.0940 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-06
Epoch 52/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0148 - mae:
0.0935 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-06
Epoch 53/100
159/159 [==============================] - 1s 9ms/step - loss: 0.0146 - mae:
0.0928 - val_loss: 0.0159 - val_mae: 0.0931 - lr: 1.0000e-07
--- 83.14158201217651 seconds ---
```

[154]:
```
x_train.shape
```

[154]:
```
(159, 4, 50)
```

[155]:
```
# make predictions
trainPredict = model.predict(x_train).reshape((636,50))
testPredict = model.predict(x_val).reshape((124,50))
y_train = y_train.reshape((636,50))
y_val = y_val.reshape((124,50))
```

```
5/5 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 18ms/step
```

[156]:
```
trainPredict.shape
```

[156]:
```
(636, 50)
```

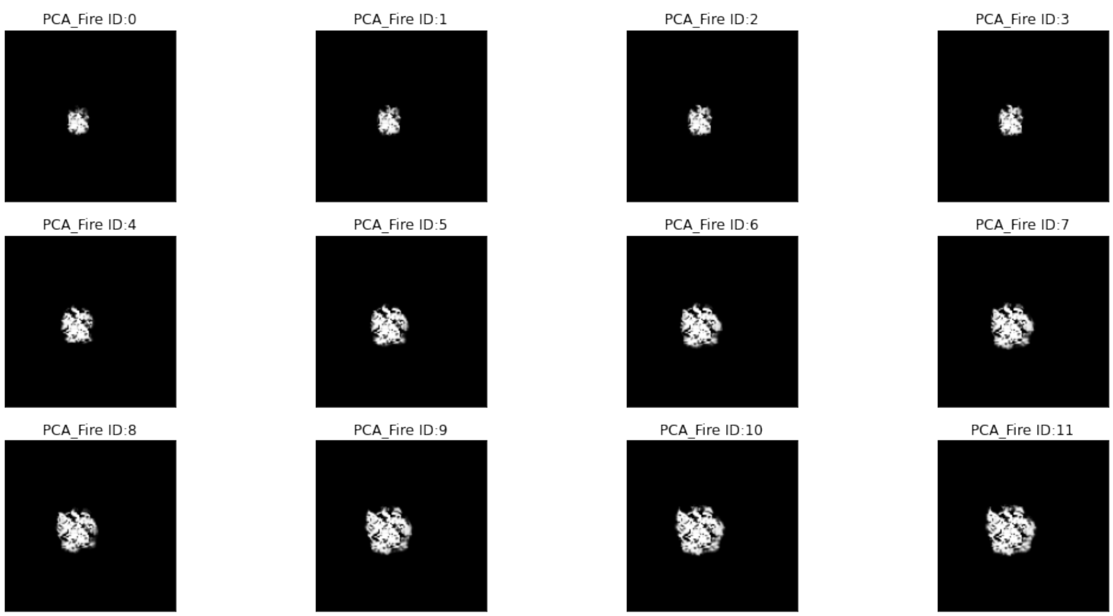
[157]:
```
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform(y_train)
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform(y_val)

x_train = scaler.inverse_transform(x_train.reshape(636,50))
x_train = decoded(x_train)
```

[158]:
```
trainPredict = decoded(trainPredict)
trainY = decoded(trainY)
testPredict = decoded(testPredict)
testY = decoded(testY)
```

[159]:
```
np.max(trainPredict)
```

[159]: 1.0

[160]:
```
trainPredict_show = trainPredict[:12,:,:,:]
trainPredict_show.shape
fig, arr = plt.subplots(nrows = 3, ncols = 4, figsize = (18, 9))
arr = arr.flatten()
for ids in range(12):
    arr[ids].imshow(np.squeeze(trainPredict_show[ids]), cmap = 'gray')
    arr[ids].set_xticks([])
    arr[ids].set_yticks([])
    arr[ids].set_title("PCA_Fire ID:{}".format(ids))
```



[245]:
```
mse = np.mean((testY - testPredict)**2)
rmse = np.sqrt(mse)
rmse
```

[245]: 0.061775286

[162]:
```
testY.shape
```

[162]: TensorShape([124, 128, 128, 1])

[163]:
```
testY = np.array(testY).reshape((124,128,128))
testPredict = np.array(testPredict).reshape((124,128,128))
error = structural_similarity(testY, testPredict,multichannel=False)

print(f'Error: {error:.2f}')
```

Error: 0.96

5.0.1 CAE+LSTM

RMSE : 0.0618

SSIM : 0.96

ALL Based on original images and restruction images

6 3. Conv-LSTM

[232]:
```
import cv2
import numpy as np

dataset_3D = []
dataset = []
data_conv = []
for i in range(48):
    video = cv2.VideoCapture("/content/drive/MyDrive/UROP Sibo/A Machine␣
↪learning problem/VIDEOS/fire_Chimney_video_{}.mp4".format(i))

    data_3 = []
    while True:
        ret, frame = video.read()

        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
        binary_flat = binary.flatten()

        dataset.append(binary_flat)
        dataset_3D.append(gray)
        data_3.append(gray)
    data_conv.append(data_3)

dataset = np.array(dataset)
dataset_3D = np.array(dataset_3D)
```

```
print(dataset.shape)
```

(768, 16384)

[233]:
```
data_conv = np.array(data_conv)
data_conv.shape
dataset = data_conv
```

[234]:
```
dataset = np.expand_dims(dataset, axis=-1)
dataset.shape
```

[234]: (48, 16, 128, 128, 1)

[235]:
```
indexes = np.arange(dataset.shape[0])
np.random.shuffle(indexes)
train_index = indexes[: 40]
val_index = indexes[40 :]
train_dataset = dataset[train_index]
val_dataset = dataset[val_index]
```

[236]:
```
train_dataset = train_dataset / 255
val_dataset = val_dataset / 255
```

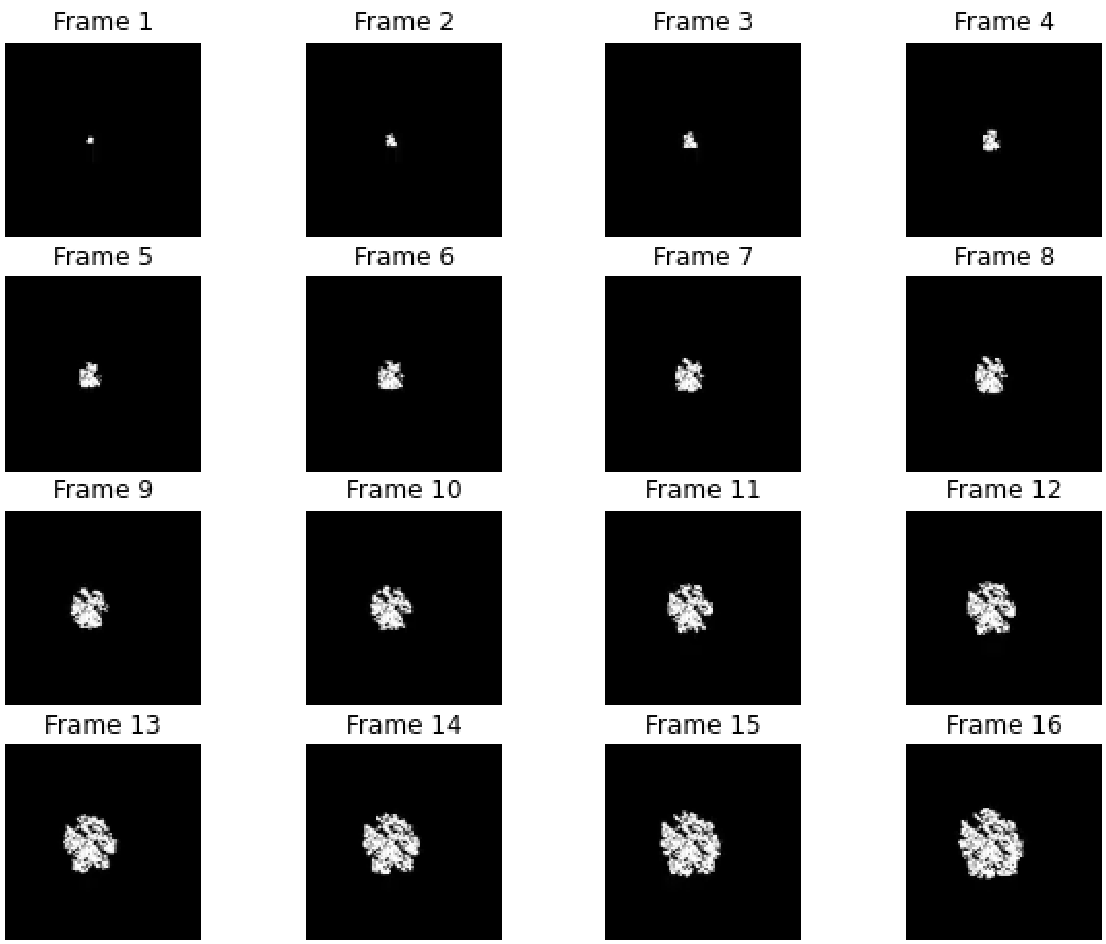
[237]:
```
def create_shifted_frames(data):
    x = data[:, 0 : data.shape[1] - 1, :, :]
    y = data[:, 1 : data.shape[1], :, :]
    return x, y
# Apply the processing function to the datasets.
x_train, y_train = create_shifted_frames(train_dataset)
x_val, y_val = create_shifted_frames(val_dataset)

print("Training Dataset Shapes: " + str(x_train.shape) + ", " + str(y_train.
 ↪shape))
print("Validation Dataset Shapes: " + str(x_val.shape) + ", " + str(y_val.
 ↪shape))
```

```
Training Dataset Shapes: (40, 15, 128, 128, 1), (40, 15, 128, 128, 1)
Validation Dataset Shapes: (8, 15, 128, 128, 1), (8, 15, 128, 128, 1)
```

[238]:
```
fig, axes = plt.subplots(4, 4, figsize=(10, 8))

# Plot each of the sequential images for one random data example.
data_choice = np.random.choice(range(len(train_dataset)), size=1)[0]
for idx, ax in enumerate(axes.flat):
    ax.imshow(np.squeeze(train_dataset[data_choice][idx]), cmap="gray")
    ax.set_title(f"Frame {idx + 1}")
    ax.axis("off")
```

```python
print(f"Displaying frames for example {data_choice}.")
plt.show()
```

```
Displaying frames for example 15.
```



[239]:

```python
inp = layers.Input(shape=(None, *x_train.shape[2:]))

x = layers.ConvLSTM2D(
    filters=16,
    kernel_size=(3, 3),
    padding="same",
    return_sequences=True,
    activation="relu",
)(inp)
x = layers.BatchNormalization()(x)
x = layers.ConvLSTM2D(
    filters=16,
    kernel_size=(3,3),
```

```python
    padding="same",
    return_sequences=True,
    activation="relu",
)(x)
x = layers.BatchNormalization()(x)
x = layers.ConvLSTM2D(
    filters=8,
    kernel_size=(3, 3),
    padding="same",
    return_sequences=True,
    activation="sigmoid",
)(x)

x = layers.Conv3D(
    filters=1, kernel_size=(3, 3, 3), activation="sigmoid", padding="same"
)(x)

model = keras.models.Model(inp, x)
model.compile(
    loss=keras.losses.mse, optimizer=keras.optimizers.Adam(),
)
model.summary()
```

```
Model: "model_17"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_17 (InputLayer)       [(None, None, 128, 128,   0
                             1)]

 conv_lstm2d_33 (ConvLSTM2D)  (None, None, 128, 128, 1  9856
                             6)

 batch_normalization_38 (Bat  (None, None, 128, 128, 1  64
 chNormalization)            6)

 conv_lstm2d_34 (ConvLSTM2D)  (None, None, 128, 128, 1  18496
                             6)

 batch_normalization_39 (Bat  (None, None, 128, 128, 1  64
 chNormalization)            6)

 conv_lstm2d_35 (ConvLSTM2D)  (None, None, 128, 128, 8  6944
                             )

 conv3d_11 (Conv3D)          (None, None, 128, 128, 1  217
                             )
```

```
=================================================================
Total params: 35,641
Trainable params: 35,577
Non-trainable params: 64
_________________________________________________________________
```

[240]:
```
early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=5)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)

epochs = 100
batch_size = 2

start_time = time.time()

model.fit(
    x_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(x_val, y_val),
    callbacks=[early_stopping, reduce_lr],
)

print("--- %s seconds ---" % (time.time() - start_time))
```

```
Epoch 1/100
20/20 [==============================] - 9s 275ms/step - loss: 0.0541 -
val_loss: 0.0278 - lr: 0.0010
Epoch 2/100
20/20 [==============================] - 5s 244ms/step - loss: 0.0080 -
val_loss: 0.0218 - lr: 0.0010
Epoch 3/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0066 -
val_loss: 0.0210 - lr: 0.0010
Epoch 4/100
20/20 [==============================] - 5s 244ms/step - loss: 0.0060 -
val_loss: 0.0206 - lr: 0.0010
Epoch 5/100
20/20 [==============================] - 5s 246ms/step - loss: 0.0056 -
val_loss: 0.0202 - lr: 0.0010
Epoch 6/100
20/20 [==============================] - 5s 249ms/step - loss: 0.0052 -
val_loss: 0.0199 - lr: 0.0010
Epoch 7/100
20/20 [==============================] - 5s 252ms/step - loss: 0.0048 -
val_loss: 0.0195 - lr: 0.0010
Epoch 8/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0045 -
```

```
val_loss: 0.0192 - lr: 0.0010
Epoch 9/100
20/20 [==============================] - 5s 249ms/step - loss: 0.0041 -
val_loss: 0.0189 - lr: 0.0010
Epoch 10/100
20/20 [==============================] - 5s 246ms/step - loss: 0.0039 -
val_loss: 0.0186 - lr: 0.0010
Epoch 11/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0036 -
val_loss: 0.0183 - lr: 0.0010
Epoch 12/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0034 -
val_loss: 0.0180 - lr: 0.0010
Epoch 13/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0032 -
val_loss: 0.0176 - lr: 0.0010
Epoch 14/100
20/20 [==============================] - 5s 244ms/step - loss: 0.0030 -
val_loss: 0.0173 - lr: 0.0010
Epoch 15/100
20/20 [==============================] - 5s 244ms/step - loss: 0.0028 -
val_loss: 0.0169 - lr: 0.0010
Epoch 16/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0027 -
val_loss: 0.0165 - lr: 0.0010
Epoch 17/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0026 -
val_loss: 0.0161 - lr: 0.0010
Epoch 18/100
20/20 [==============================] - 5s 242ms/step - loss: 0.0024 -
val_loss: 0.0160 - lr: 0.0010
Epoch 19/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0023 -
val_loss: 0.0153 - lr: 0.0010
Epoch 20/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0022 -
val_loss: 0.0140 - lr: 0.0010
Epoch 21/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0021 -
val_loss: 0.0139 - lr: 0.0010
Epoch 22/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0020 -
val_loss: 0.0124 - lr: 0.0010
Epoch 23/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0019 -
val_loss: 0.0108 - lr: 0.0010
Epoch 24/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0018 -
```

```
val_loss: 0.0105 - lr: 0.0010
Epoch 25/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0018 -
val_loss: 0.0085 - lr: 0.0010
Epoch 26/100
20/20 [==============================] - 5s 248ms/step - loss: 0.0017 -
val_loss: 0.0079 - lr: 0.0010
Epoch 27/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0016 -
val_loss: 0.0069 - lr: 0.0010
Epoch 28/100
20/20 [==============================] - 6s 292ms/step - loss: 0.0016 -
val_loss: 0.0040 - lr: 0.0010
Epoch 29/100
20/20 [==============================] - 5s 244ms/step - loss: 0.0015 -
val_loss: 0.0041 - lr: 0.0010
Epoch 30/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0015 -
val_loss: 0.0029 - lr: 0.0010
Epoch 31/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0014 -
val_loss: 0.0027 - lr: 0.0010
Epoch 32/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0014 -
val_loss: 0.0023 - lr: 0.0010
Epoch 33/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0013 -
val_loss: 0.0020 - lr: 0.0010
Epoch 34/100
20/20 [==============================] - 5s 247ms/step - loss: 0.0013 -
val_loss: 0.0018 - lr: 0.0010
Epoch 35/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0013 -
val_loss: 0.0016 - lr: 0.0010
Epoch 36/100
20/20 [==============================] - 5s 246ms/step - loss: 0.0012 -
val_loss: 0.0014 - lr: 0.0010
Epoch 37/100
20/20 [==============================] - 5s 244ms/step - loss: 0.0012 -
val_loss: 0.0013 - lr: 0.0010
Epoch 38/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0011 -
val_loss: 0.0012 - lr: 0.0010
Epoch 39/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0011 -
val_loss: 0.0012 - lr: 0.0010
Epoch 40/100
20/20 [==============================] - 5s 242ms/step - loss: 0.0011 -
```

```
val_loss: 0.0012 - lr: 0.0010
Epoch 41/100
20/20 [==============================] - 5s 243ms/step - loss: 0.0010 -
val_loss: 0.0011 - lr: 0.0010
Epoch 42/100
20/20 [==============================] - 5s 245ms/step - loss: 0.0010 -
val_loss: 0.0011 - lr: 0.0010
Epoch 43/100
20/20 [==============================] - 5s 246ms/step - loss: 9.7331e-04 -
val_loss: 0.0010 - lr: 0.0010
Epoch 44/100
20/20 [==============================] - 5s 245ms/step - loss: 9.4514e-04 -
val_loss: 0.0010 - lr: 0.0010
Epoch 45/100
20/20 [==============================] - 5s 249ms/step - loss: 9.2041e-04 -
val_loss: 0.0010 - lr: 0.0010
Epoch 46/100
20/20 [==============================] - 5s 247ms/step - loss: 8.9951e-04 -
val_loss: 9.6570e-04 - lr: 0.0010
Epoch 47/100
20/20 [==============================] - 5s 247ms/step - loss: 8.8652e-04 -
val_loss: 9.4855e-04 - lr: 0.0010
Epoch 48/100
20/20 [==============================] - 5s 246ms/step - loss: 8.5811e-04 -
val_loss: 9.3798e-04 - lr: 0.0010
Epoch 49/100
20/20 [==============================] - 5s 245ms/step - loss: 8.3904e-04 -
val_loss: 8.9979e-04 - lr: 0.0010
Epoch 50/100
20/20 [==============================] - 5s 247ms/step - loss: 8.2492e-04 -
val_loss: 8.8451e-04 - lr: 0.0010
Epoch 51/100
20/20 [==============================] - 5s 249ms/step - loss: 8.0488e-04 -
val_loss: 8.8510e-04 - lr: 0.0010
Epoch 52/100
20/20 [==============================] - 5s 244ms/step - loss: 7.9610e-04 -
val_loss: 8.9067e-04 - lr: 0.0010
Epoch 53/100
20/20 [==============================] - 5s 247ms/step - loss: 7.8535e-04 -
val_loss: 8.7768e-04 - lr: 0.0010
Epoch 54/100
20/20 [==============================] - 5s 244ms/step - loss: 7.5519e-04 -
val_loss: 8.6542e-04 - lr: 0.0010
Epoch 55/100
20/20 [==============================] - 5s 244ms/step - loss: 7.4154e-04 -
val_loss: 8.1743e-04 - lr: 1.0000e-04
Epoch 56/100
20/20 [==============================] - 5s 248ms/step - loss: 7.3670e-04 -
```

```
val_loss: 8.0890e-04 - lr: 1.0000e-04
Epoch 57/100
20/20 [==============================] - 5s 247ms/step - loss: 7.3385e-04 -
val_loss: 8.0332e-04 - lr: 1.0000e-04
Epoch 58/100
20/20 [==============================] - 5s 246ms/step - loss: 7.3304e-04 -
val_loss: 8.0036e-04 - lr: 1.0000e-04
Epoch 59/100
20/20 [==============================] - 5s 245ms/step - loss: 7.3062e-04 -
val_loss: 7.9876e-04 - lr: 1.0000e-04
Epoch 60/100
20/20 [==============================] - 5s 245ms/step - loss: 7.2871e-04 -
val_loss: 7.9651e-04 - lr: 1.0000e-04
Epoch 61/100
20/20 [==============================] - 5s 244ms/step - loss: 7.2704e-04 -
val_loss: 7.9403e-04 - lr: 1.0000e-04
Epoch 62/100
20/20 [==============================] - 5s 245ms/step - loss: 7.2571e-04 -
val_loss: 7.9027e-04 - lr: 1.0000e-04
Epoch 63/100
20/20 [==============================] - 5s 245ms/step - loss: 7.2443e-04 -
val_loss: 7.8976e-04 - lr: 1.0000e-04
Epoch 64/100
20/20 [==============================] - 5s 246ms/step - loss: 7.2197e-04 -
val_loss: 7.8641e-04 - lr: 1.0000e-04
Epoch 65/100
20/20 [==============================] - 5s 244ms/step - loss: 7.2062e-04 -
val_loss: 7.8626e-04 - lr: 1.0000e-05
Epoch 66/100
20/20 [==============================] - 5s 246ms/step - loss: 7.2046e-04 -
val_loss: 7.8535e-04 - lr: 1.0000e-05
Epoch 67/100
20/20 [==============================] - 5s 244ms/step - loss: 7.2193e-04 -
val_loss: 7.8482e-04 - lr: 1.0000e-05
Epoch 68/100
20/20 [==============================] - 5s 247ms/step - loss: 7.2142e-04 -
val_loss: 7.8454e-04 - lr: 1.0000e-05
Epoch 69/100
20/20 [==============================] - 5s 245ms/step - loss: 7.1978e-04 -
val_loss: 7.8403e-04 - lr: 1.0000e-05
Epoch 70/100
20/20 [==============================] - 5s 247ms/step - loss: 7.2008e-04 -
val_loss: 7.8380e-04 - lr: 1.0000e-06
Epoch 71/100
20/20 [==============================] - 5s 247ms/step - loss: 7.1938e-04 -
val_loss: 7.8355e-04 - lr: 1.0000e-06
Epoch 72/100
20/20 [==============================] - 5s 247ms/step - loss: 7.2083e-04 -
```

```
val_loss: 7.8344e-04 - lr: 1.0000e-06
Epoch 73/100
20/20 [==============================] - 5s 247ms/step - loss: 7.1906e-04 -
val_loss: 7.8326e-04 - lr: 1.0000e-06
Epoch 74/100
20/20 [==============================] - 5s 246ms/step - loss: 7.2037e-04 -
val_loss: 7.8320e-04 - lr: 1.0000e-06
Epoch 75/100
20/20 [==============================] - 5s 250ms/step - loss: 7.2007e-04 -
val_loss: 7.8311e-04 - lr: 1.0000e-07
Epoch 76/100
20/20 [==============================] - 5s 246ms/step - loss: 7.2009e-04 -
val_loss: 7.8311e-04 - lr: 1.0000e-07
Epoch 77/100
20/20 [==============================] - 5s 249ms/step - loss: 7.2028e-04 -
val_loss: 7.8312e-04 - lr: 1.0000e-07
Epoch 78/100
20/20 [==============================] - 5s 245ms/step - loss: 7.1989e-04 -
val_loss: 7.8304e-04 - lr: 1.0000e-07
Epoch 79/100
20/20 [==============================] - 5s 247ms/step - loss: 7.1970e-04 -
val_loss: 7.8298e-04 - lr: 1.0000e-07
Epoch 80/100
20/20 [==============================] - 5s 243ms/step - loss: 7.2104e-04 -
val_loss: 7.8309e-04 - lr: 1.0000e-08
Epoch 81/100
20/20 [==============================] - 5s 243ms/step - loss: 7.1990e-04 -
val_loss: 7.8308e-04 - lr: 1.0000e-08
Epoch 82/100
20/20 [==============================] - 5s 247ms/step - loss: 7.1965e-04 -
val_loss: 7.8302e-04 - lr: 1.0000e-08
Epoch 83/100
20/20 [==============================] - 5s 244ms/step - loss: 7.1938e-04 -
val_loss: 7.8290e-04 - lr: 1.0000e-08
Epoch 84/100
20/20 [==============================] - 5s 245ms/step - loss: 7.1997e-04 -
val_loss: 7.8291e-04 - lr: 1.0000e-08
Epoch 85/100
20/20 [==============================] - 6s 287ms/step - loss: 7.1954e-04 -
val_loss: 7.8289e-04 - lr: 1.0000e-09
Epoch 86/100
20/20 [==============================] - 5s 255ms/step - loss: 7.2015e-04 -
val_loss: 7.8289e-04 - lr: 1.0000e-09
Epoch 87/100
20/20 [==============================] - 5s 247ms/step - loss: 7.2015e-04 -
val_loss: 7.8294e-04 - lr: 1.0000e-09
Epoch 88/100
20/20 [==============================] - 5s 246ms/step - loss: 7.1956e-04 -
```

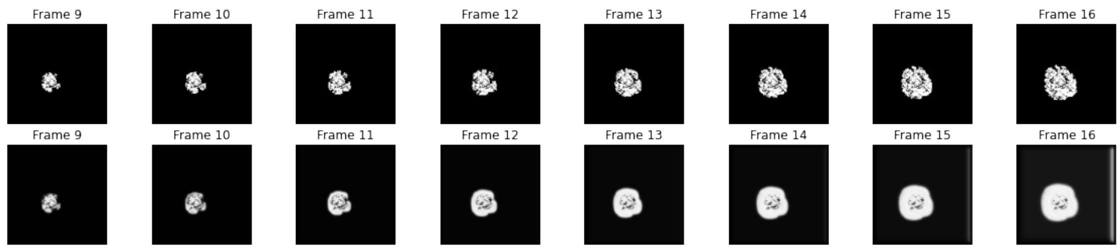
```
val_loss: 7.8290e-04 - lr: 1.0000e-09
Epoch 89/100
20/20 [==============================] - 5s 248ms/step - loss: 7.1992e-04 -
val_loss: 7.8294e-04 - lr: 1.0000e-09
Epoch 90/100
20/20 [==============================] - 5s 245ms/step - loss: 7.2018e-04 -
val_loss: 7.8298e-04 - lr: 1.0000e-10
--- 447.8971948623657 seconds ---
```

[244]:

```
# Select a random example from the validation dataset.
example = val_dataset[np.random.choice(range(len(val_dataset)), size=1)[0]]

# Pick the first/last ten frames from the example.
frames = example[:8, ...]
original_frames = example[8:, ...]

for _ in range(8):
    new_prediction = model.predict(np.expand_dims(frames, axis=0))
    new_prediction = np.squeeze(new_prediction, axis=0)
    predicted_frame = np.expand_dims(new_prediction[-1, ...], axis=0)

    # Extend the set of prediction frames.
    frames = np.concatenate((frames, predicted_frame), axis=0)

# Construct a figure for the original and new frames.
fig, axes = plt.subplots(2, 8, figsize=(20, 4))

# Plot the original frames.
for idx, ax in enumerate(axes[0]):
    ax.imshow(np.squeeze(original_frames[idx]), cmap="gray")
    ax.set_title(f"Frame {idx + 9}")
    ax.axis("off")

new_frames = frames[8:, ...]
for idx, ax in enumerate(axes[1]):
    ax.imshow(np.squeeze(new_frames[idx]), cmap="gray")
    ax.set_title(f"Frame {idx + 9}")
    ax.axis("off")

plt.show()
```

```
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 87ms/step
1/1 [==============================] - 0s 103ms/step
```

```
1/1 [==============================] - 0s 76ms/step
```



[242]:
```python
y_val = y_val.reshape((120,128,128))
y_pred = model.predict(x_val).reshape((120,128,128))
error = structural_similarity(y_val, y_pred,multichannel=False)

mse = np.mean((y_val - y_pred)**2)
rmse = np.sqrt(mse)
```

```
1/1 [==============================] - 0s 171ms/step
```

```
<ipython-input-242-ead3bd754bd4>:3: UserWarning: Inputs have mismatched dtype.
Setting data_range based on im1.dtype.
  error = structural_similarity(y_val, y_pred,multichannel=False)
```

[243]:
```python
print(error)
print(mse)
print(rmse)
```

```
0.968146969793889
0.0007829778851874285
0.02798174199701349
```

7 Table for evalution results on Test datasets and online computional time on train datasets

Name of Models/Evalution Metrics	MSE	RMSE	SSIM	Train Time (s)
CAE + LSTM	0.0038	0.0617	0.96	332
ConvLSTM	0.0007	0.0279	0.97	447.90

Based on the results (MSE, RMSE, SSIM), it appears that the ConvLSTM model is performing much better than the CAE-LSTM model on the test dataset. This could be due to a number of factors, including the specific characteristics of the dataset, the design of the models, and the hyperparameters used. It's also possible that the ConvLSTM model is simply a better fit for the task. From the aspect of online training computional time, CAE+LSTM shows a better performance than ConvLSTM.

Some potential explanations include: 1. The quality of the features: The ConvLSTM model is able to extract more relevant and useful features from the input data than the CAE-LSTM model, leading to better performance.

2. The complexity of the models: The ConvLSTM model is able to capture more complex patterns in the data than the CAE-LSTM model, leading to better performance. (Not more complex, more better. A overfitting question should be considered in the design of the model.)

Name of Models/Number	trainable params
CAE + LSTM	68,264
ConvLSTM	35,577

3. The hyperparameters of the models: The specific values of the hyperparameters (such as the learning rate, the number of layersm, loss function, optimizer) used for the ConvLSTM and CAE-LSTM models could be affecting their performance. You know, the hyperparameters'choice of DL models are unexplainable and unknown. Any params lead to any results.

4. The characteristics of the dataset: The ConvLSTM model may simply be a better fit for the specific characteristics of the wildfire prediction dataset.

A common problem is that performance of models is limited by the size of the dataset. In the future, probabilistic DL methods can be considered for the prediction of wildfires, such as probabilistic cGAN, GP and MDN.

8 4. Some Strategy

1. Data preprocessing: If the wildfire field can only be increasing, it means that any decrease in the field area is most likely due to errors in the data collection or processing. Therefore, it may be helpful to identify and remove such erroneous data points before training the model.

2. Model design: We can incorporate this information into the design of the model by using an architecture that only allows for increasing predictions. For example, a monotonic activation function such as the sigmoid function or the softplus function can be used to ensure that the model only produces increasing outputs.

3. Loss function: We can also modify the loss function to encourage the model to make only increasing predictions. One way to do this is to add a penalty term to the loss function for decreasing predictions. The loss function can be motified used to train the model to encourage monotonic behavior. One way to do this is to add a penalty term to the loss function for decreasing predictions. For example, the following loss function:

$$loss = (1 - k) * MSE + k * abs(y_p - y_t)$$

   where k is a hyperparameter that controls the weight of the penalty term.

4. Training strategy: We can also use a training strategy that explicitly encourages the model to make only increasing predictions. For example, a curriculum learning approach is avaliable, where the model is first trained on easy examples where the wildfire field is increasing, and then gradually exposed to more challenging examples. The training data is presented to the model in a sequence of increasingly difficult tasks. This can help the model learn to make only increasing predictions more effectively.

5. Data augmentation: Use data augmentation techniques to generate additional training data that is consistent with the monotonicity constraint. For example, we could generate synthetic examples where the wildfire field is increasing by applying transformations to the existing data that preserve the monotonic relationship. Throughout the implementation of autoencoder and LSTM, the overfitting caused by the small dataset was evident, but the network structure was not well generalised if simplified, so a larger dataset was necessary.