TGS Salt Identification Challenge

Segment salt deposits beneath the Earth's surface

Several areas of Earth with large accumulations of oil and gas also have huge deposits of salt below the surface.

But unfortunately, knowing where large salt deposits are precisely is very difficult. Professional seismic imaging still requires expert human interpretation of salt bodies. This leads to very subjective, highly variable renderings. More alarmingly, it leads to potentially dangerous situations for oil and gas company drillers.

To create the most accurate seismic images and 3D renderings, TGS (the world's leading geoscience data company) is hoping Kaggle's machine learning community will be able to build an algorithm that automatically and accurately identifies if a subsurface target is salt or not.

Dataset: https://www.kaggle.com/c/tgs-salt-identification-challenge/data

In [1]:

```
! wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/10151/59042/bundle/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1586444582&Signature=gOljnbjB5OvP4FVSOHsfW5VZ%2F0uYmqqQezFg48aB0eveE64U3zZzzrgAwLp4tVhC9Vf8YJoOywQvBKS9U6lDfuRGZC8uMC%2BusEyHS%2FqJQSwLoJ9MfuqTPFxz%2B%2BxWOonSprVQ7LaFOU0bqWpCsrF%2Fo9Q2EtETvKEfq0aAzzoKBc9N7%2BOg7288FUsJXjJwjZSKwpT3DnACPuUbSqF%2BcuuEMYLvF1WyMH%2FvQZifw2P2YdBYKBXDztDI3TY%2FC4cfOYuBKPpaHClaBxGX1nxazJhtFkqerDvP2doqSxh6rxuWufYF8JY4JD9eazvaffv3%2FM6ZKgk0aA7tB%2F7mgmsIEbZXzg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtgs-salt-identification-challenge.zip" -c -O 'tgs-salt-identification-challenge.zip'
```

```
--2020-04-07 08:01:02--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/10151/59042/bundle/archive.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1586444582&Signature=gOljnbjB5OvP4FVSOHsfW5VZ%2F0uYmqqQezFg48aB0eveE64U3zZzzrgAwLp4tVhC9Vf8YJoOywQvBKS9U6lDfuRGZC8uMC%2BusEyHS%2FqJQSwLoJ9MfuqTPFxz%2B%2BxWOonSprVQ7LaFOU0bqWpCsrF%2Fo9Q2EtETvKEfq0aAzzoKBc9N7%2BOg7288FUsJXjJwjZSKwpT3DnACPuUbSqF%2BcuuEMYLvF1WyMH%2FvQZifw2P2YdBYKBXDztDI3TY%2FC4cfOYuBKPpaHClaBxGX1nxazJhtFkqerDvP2doqSxh6rxuWufYF8JY4JD9eazvaffv3%2FM6ZKgk0aA7tB%2F7mgmsIEbZXzg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtgs-salt-identification-challenge.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.97.128, 2404:6800:4008:c04::80
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.97.128|:443... connected.
HTTP request sent, awaiting response... 416 Requested range not satisfiable

    The file is already fully retrieved; nothing to do.
```

In [0]:

```python
import zipfile
with zipfile.ZipFile("/content/tgs-salt-identification-challenge.zip","r") as f:
  f.extractall()
```

In [0]:

```python
with zipfile.ZipFile('/content/flamingo.zip', 'r') as f:
  f.extractall('/content/flamingo')
```

In [0]:

```python
with zipfile.ZipFile('/content/train.zip') as f:
  f.extractall('/content/train')
```

In [0]:

```python
with zipfile.ZipFile('/content/test.zip') as f:
  f.extractall('/content/test')
```

In [0]:

```python
import os
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use("ggplot")
```

```
from tqdm import notebook, tnrange
from itertools import chain
from skimage.io import imread, imshow, concatenate_images
from skimage.transform import resize
from skimage.morphology import label
from sklearn.model_selection import train_test_split

import tensorflow as tf

from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Conv2D, Conv2DTranspose, BatchNormalization, Activation,\
                                    Dense, Dropout, MaxPooling2D, GlobalMaxPool2D, Lambda, \
                                    RepeatVector, Reshape, concatenate, add
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
```

In [7]:

```
# This has three components ('/content/train/images', [], ['64e79513a3.png',...4000 images])

train_images= next(os.walk('/content/train/images'))
print(len(train_images))

# Images are in 3rd position hence train_images[2] / train_masks[2]
print('No of images in train data:', len(train_images[2]))

train_masks= next(os.walk('/content/train/masks'))
print('No of masks in train data:', len(train_masks[2]))
```

```
3
No of images in train data: 4000
No of masks in train data: 4000
```

In [8]:

```
df= pd.read_csv(r'/content/train.csv')
print(df.shape)
print(df.head())
```

```
(4000, 2)
           id                                           rle_mask
0  575d24d81d                                                NaN
1  a266a2a9df                                          5051 5151
2  75efad62c1  9 93 109 94 210 94 310 95 411 95 511 96 612 96...
3  34e51dba6a  48 54 149 54 251 53 353 52 455 51 557 50 659 4...
4  4875705fb0  1111 1 1212 1 1313 1 1414 1 1514 2 1615 2 1716...
```

In [9]:

```
# We observe zero nulls in 'id' column but 1562 NaNs in 'rle_mask' column
df.isnull().sum()
```

Out[9]:

```
id             0
rle_mask    1562
dtype: int64
```

In [10]:

```
# we find the extension is '.png'
print(train_images[2][0])
print(train_masks[2][0])
```

```
f409d55fee.png
f409d55fee.png
```

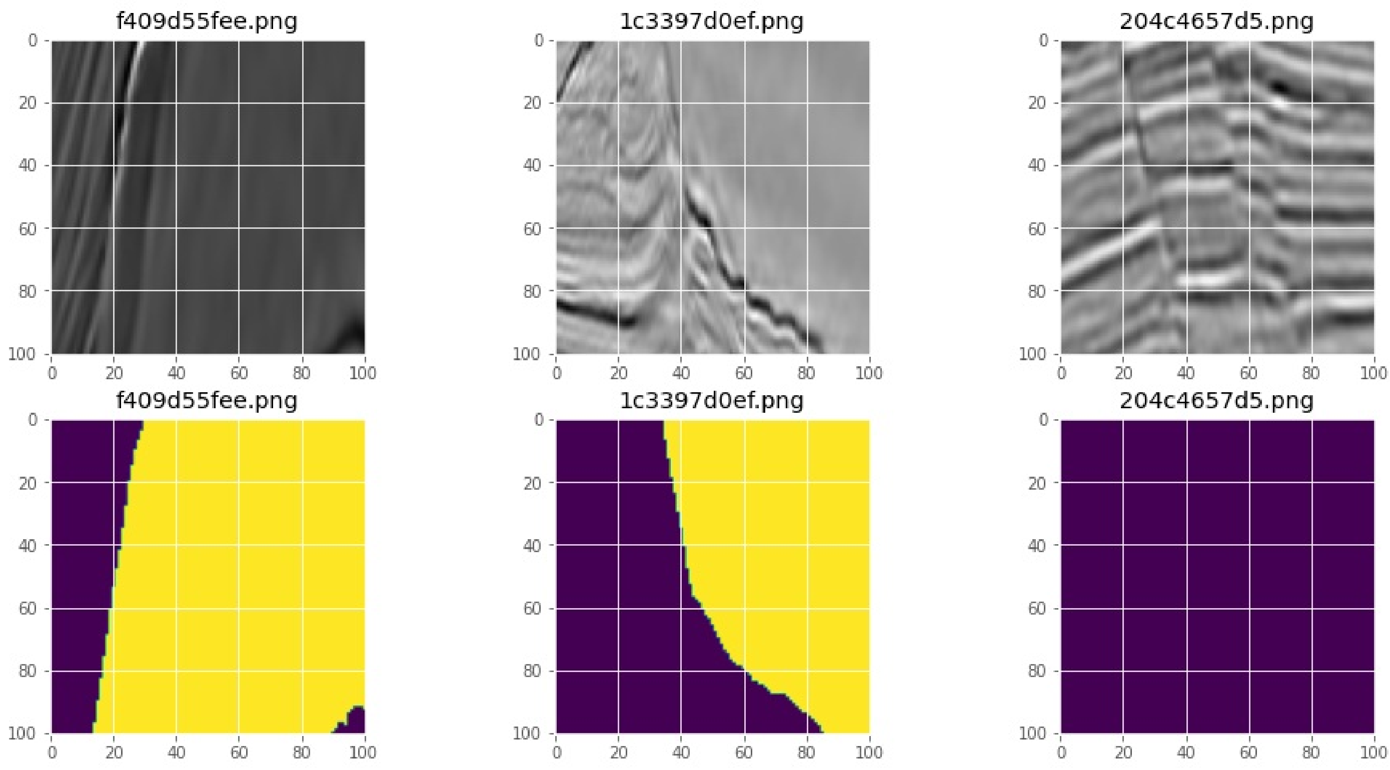
Below code can be used to visualize the images and corresponding masks

In [11]:

```
# We now check sample images
```

```
path= '/content/train/images/'     # the path is provided
key= train_images[2][:3]

path_mask= '/content/train/masks/' # the path is provided
key_mask= train_masks[2][:3]

image= []
title= []

for i in key[:3]:
  image.append(path+i)
  title.append(i)

for i in key_mask[:3]:
  image.append(path_mask+i)
  title.append(i)

plt.figure(figsize= (16, 8))
for i in range(6):
  plt.subplot(2, 3, i+1)
  j= plt.imread(image[i])
  print(j.shape)
  plt.imshow(j)
  plt.title(title[i])

plt.show()
```

```
(101, 101, 3)
(101, 101, 3)
(101, 101, 3)
(101, 101)
(101, 101)
(101, 101)
```



In [12]:

```
# check the min and max sizes of images

import time
start= time.time()

# We check the max and min shape of images
width=[]
height=[]
for i in train_images[2][:]:
  j= plt.imread(path+i)
  w, h, channel= j.shape
  width.append(w)
  height.append(h)
```

```
print('Max Width: ', max(width))
print('Min Width: ', min(width))
print('Max Height: ', max(height))
print('Min Height: ', min(height))
print("Time Taken is: " + str(time.time() - start))
```

```
Max Width:  101
Min Width:  101
Max Height:  101
Min Height:  101
Time Taken is: 3.57474946975708
```

Load the images and masks into arrays

In [0]:

```
# Reshape the image from 101, 101, 3 to 128, 128, 1
im_height= 128
im_width= 128

X = np.zeros((4000, im_height, im_width, 1), dtype=np.float32)
y = np.zeros((4000, im_height, im_width, 1), dtype=np.float32)
```

In [0]:

```
for i, j in enumerate(train_images[2][:]):
    # Load images
    img = load_img(path+j, color_mode= 'grayscale')
    img = img_to_array(img)
    img = resize(img, (128, 128, 1), mode = 'constant', preserve_range = True)
    # Save images
    X[i] = img / 255.0


for i, j in enumerate(train_masks[2][:]):
    # Load masks
    mask = img_to_array(load_img(path_mask+j, color_mode= 'grayscale'))
    mask = resize(mask, (128, 128, 1), mode = 'constant', preserve_range = True)
    # Save images
    y[i] = mask / 255.0
```

In [16]:

```
print('Shape of X:', X.shape)
print('Shape of Y:', y.shape)
```

```
Shape of X: (4000, 128, 128, 1)
Shape of Y: (4000, 128, 128, 1)
```

In [17]:

```
# Split train and valid
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.1, random_state=42)

print('Shape of X_train:', X_train.shape)
print('Shape of y_train:', y_train.shape)
```

```
Shape of X_train: (3600, 128, 128, 1)
Shape of y_train: (3600, 128, 128, 1)
```

In [18]:

```
print('Shape of X:', X_valid.shape)
print('Shape of Y:', y_valid.shape)
```

```
Shape of X: (400, 128, 128, 1)
Shape of Y: (400, 128, 128, 1)
```

In [0]:

```
# To define a conv2D block (2 times conv2D)

def conv2d_block(input_tensor, filters, kernel_size = 3, batchnorm = True):
```

```
    """Function to add 2 convolutional layers with the parameters passed to it"""
    # first layer
    x = Conv2D(filters = filters, kernel_size = (kernel_size, kernel_size), activation= 'relu', \
              padding = 'same' )(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)

    # second layer
    x = Conv2D(filters = filters, kernel_size = (kernel_size, kernel_size), activation= 'relu', \
              padding = 'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)

    return x
```

U-NET Architechture

- Encoder part => (16, 16), (32, 32), (64, 64), (128, 128), 256
- Decoder part => [128, (256, 256), 128], [64, (128, 128), 64], [32, (64, 64), 32], [16, (32, 32), 16], 1

In [0]:

```
def get_unet(input_img, filters = 16, dropout = 0.1, batchnorm = True):
    """Function to define the UNET Model"""
    # Contracting Path
    c1 = conv2d_block(input_img, filters * 1, kernel_size = 3, batchnorm = batchnorm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)

    c2 = conv2d_block(p1, filters * 2, kernel_size = 3, batchnorm = batchnorm)
    p2 = MaxPooling2D((2, 2))(c2)
    p2 = Dropout(dropout)(p2)

    c3 = conv2d_block(p2, filters * 4, kernel_size = 3, batchnorm = batchnorm)
    p3 = MaxPooling2D((2, 2))(c3)
    p3 = Dropout(dropout)(p3)

    c4 = conv2d_block(p3, filters * 8, kernel_size = 3, batchnorm = batchnorm)
    p4 = MaxPooling2D((2, 2))(c4)
    p4 = Dropout(dropout)(p4)

    c5 = conv2d_block(p4, filters * 16, kernel_size = 3, batchnorm = batchnorm)

    # Expansive Path
    u6 = Conv2DTranspose(filters * 8, (3, 3), strides = (2, 2), padding = 'same')(c5)
    u6 = concatenate([u6, c4])
    u6 = Dropout(dropout)(u6)
    c6 = conv2d_block(u6, filters * 8, kernel_size = 3, batchnorm = batchnorm)

    u7 = Conv2DTranspose(filters * 4, (3, 3), strides = (2, 2), padding = 'same')(c6)
    u7 = concatenate([u7, c3])
    u7 = Dropout(dropout)(u7)
    c7 = conv2d_block(u7, filters * 4, kernel_size = 3, batchnorm = batchnorm)

    u8 = Conv2DTranspose(filters * 2, (3, 3), strides = (2, 2), padding = 'same')(c7)
    u8 = concatenate([u8, c2])
    u8 = Dropout(dropout)(u8)
    c8 = conv2d_block(u8, filters * 2, kernel_size = 3, batchnorm = batchnorm)

    u9 = Conv2DTranspose(filters * 1, (3, 3), strides = (2, 2), padding = 'same')(c8)
    u9 = concatenate([u9, c1])
    u9 = Dropout(dropout)(u9)
    c9 = conv2d_block(u9, filters * 1, kernel_size = 3, batchnorm = batchnorm)

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)
    model = Model(inputs=[input_img], outputs=[outputs])
    return model
```

In [33]:

```
input_img = Input((im_height, im_width, 1), name='img')
model = get_unet(input_img, filters=16, dropout=0.05, batchnorm=True)
model.compile(optimizer=Adam(), loss="binary_crossentropy", metrics=["accuracy"])
model.summary()
```

Model: "model_1"

____

Layer (type)   Output Shape   Param #   Connected to

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
img (InputLayer)                [(None, 128, 128, 1) 0
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, 128, 128, 16) 160         img[0][0]
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, 128, 128, 16) 64          conv2d_20[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 64, 64, 16)   0           batch_normalization_19[0][0]
__________________________________________________________________________________________________
dropout_8 (Dropout)             (None, 64, 64, 16)   0           max_pooling2d_4[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 64, 64, 32)   4640        dropout_8[0][0]
__________________________________________________________________________________________________
batch_normalization_21 (BatchNo (None, 64, 64, 32)   128         conv2d_22[0][0]
__________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 32, 32, 32)   0           batch_normalization_21[0][0]
__________________________________________________________________________________________________
dropout_9 (Dropout)             (None, 32, 32, 32)   0           max_pooling2d_5[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, 32, 32, 64)   18496       dropout_9[0][0]
__________________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, 32, 32, 64)   256         conv2d_24[0][0]
__________________________________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)  (None, 16, 16, 64)   0           batch_normalization_23[0][0]
__________________________________________________________________________________________________
dropout_10 (Dropout)            (None, 16, 16, 64)   0           max_pooling2d_6[0][0]
__________________________________________________________________________________________________
conv2d_26 (Conv2D)              (None, 16, 16, 128)  73856       dropout_10[0][0]
__________________________________________________________________________________________________
batch_normalization_25 (BatchNo (None, 16, 16, 128)  512         conv2d_26[0][0]
__________________________________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)  (None, 8, 8, 128)    0           batch_normalization_25[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 8, 8, 128)    0           max_pooling2d_7[0][0]
__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, 8, 8, 256)    295168      dropout_11[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, 8, 8, 256)    1024        conv2d_28[0][0]
__________________________________________________________________________________________________
conv2d_transpose_4 (Conv2DTrans (None, 16, 16, 128)  295040      batch_normalization_27[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 16, 16, 256)  0           conv2d_transpose_4[0][0]
                                                                 batch_normalization_25[0][0]
__________________________________________________________________________________________________
dropout_12 (Dropout)            (None, 16, 16, 256)  0           concatenate_4[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 16, 16, 128)  295040      dropout_12[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 16, 16, 128)  512         conv2d_30[0][0]
__________________________________________________________________________________________________
conv2d_transpose_5 (Conv2DTrans (None, 32, 32, 64)   73792       batch_normalization_29[0][0]
__________________________________________________________________________________________________
concatenate_5 (Concatenate)     (None, 32, 32, 128)  0           conv2d_transpose_5[0][0]
                                                                 batch_normalization_23[0][0]
__________________________________________________________________________________________________
dropout_13 (Dropout)            (None, 32, 32, 128)  0           concatenate_5[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 32, 32, 64)   73792       dropout_13[0][0]
__________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 32, 32, 64)   256         conv2d_32[0][0]
__________________________________________________________________________________________________
conv2d_transpose_6 (Conv2DTrans (None, 64, 64, 32)   18464       batch_normalization_31[0][0]
__________________________________________________________________________________________________
concatenate_6 (Concatenate)     (None, 64, 64, 64)   0           conv2d_transpose_6[0][0]
                                                                 batch_normalization_21[0][0]
__________________________________________________________________________________________________
dropout_14 (Dropout)            (None, 64, 64, 64)   0           concatenate_6[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 64, 64, 32)   18464       dropout_14[0][0]
__________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 64, 64, 32)   128         conv2d_34[0][0]
__________________________________________________________________________________________________
conv2d_transpose_7 (Conv2DTrans (None, 128, 128, 16) 4624        batch_normalization_33[0][0]
__________________________________________________________________________________________________
concatenate_7 (Concatenate)     (None, 128, 128, 32) 0           conv2d_transpose_7[0][0]
                                                                 batch_normalization_19[0][0]
__________________________________________________________________________________________________
dropout_15 (Dropout)            (None, 128, 128, 32) 0           concatenate_7[0][0]
__________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 128, 128, 16) 4624        dropout_15[0][0]
__________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, 128, 128, 16) 64          conv2d_36[0][0]
```

```
conv2d_37 (Conv2D)            (None, 128, 128, 1)  17         batch_normalization_35[0][0]
==================================================================================================
Total params: 1,179,121
Trainable params: 1,177,649
Non-trainable params: 1,472
__________________________________________________________________________________________________
```

In [0]:

```
callbacks= [ EarlyStopping(monitor= 'val_loss', patience= 10, verbose= 1),
          ReduceLROnPlateau(monitor= 'val_loss', patience= 5, min_lr= 0.001, verbose= 1),
          ModelCheckpoint(filepath= '/content/model-tgs-salt.h5', monitor= 'val_loss', verbose= 1, save_best_only= True,
                    save_weights_only= True )
          ]
```

In [34]:

```
history= model.fit(x= X_train, y= y_train, batch_size= 32, epochs= 50, verbose= 1, callbacks= callbacks,
              validation_data= (X_valid, y_valid))
```

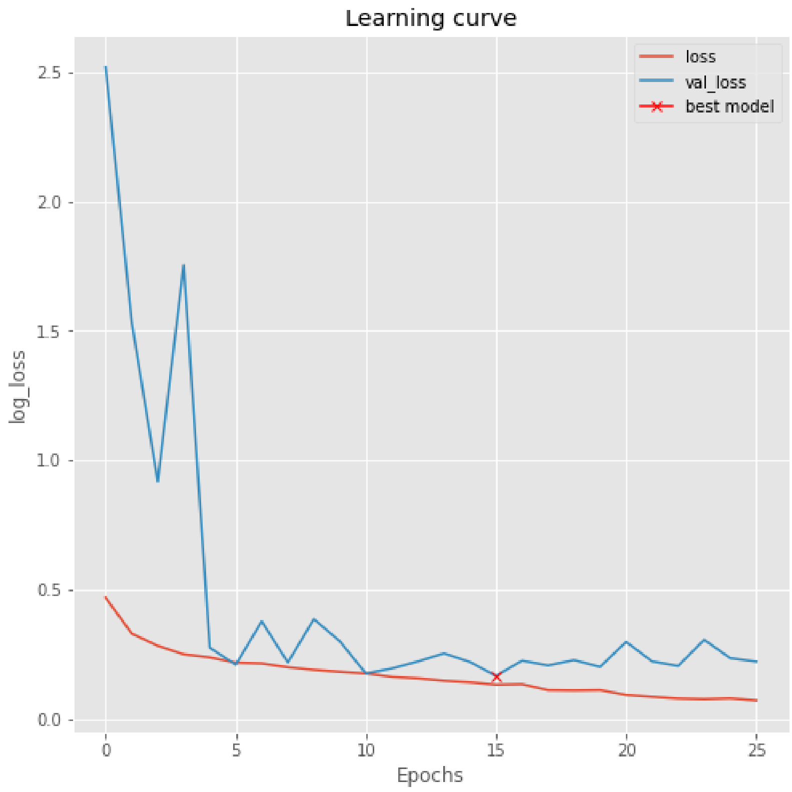
```
Epoch 1/50
113/113 [==============================] - ETA: 0s - loss: 0.4697 - accuracy: 0.8218
Epoch 00001: val_loss did not improve from 0.24889
113/113 [==============================] - 5s 46ms/step - loss: 0.4697 - accuracy: 0.8218 - val_loss: 2.5187 - val_accuracy: 0.2845 - lr: 0.0
010
Epoch 2/50
113/113 [==============================] - ETA: 0s - loss: 0.3309 - accuracy: 0.8750
Epoch 00002: val_loss did not improve from 0.24889
113/113 [==============================] - 5s 41ms/step - loss: 0.3309 - accuracy: 0.8750 - val_loss: 1.5341 - val_accuracy: 0.2952 - lr: 0.0
010
Epoch 3/50
113/113 [==============================] - ETA: 0s - loss: 0.2837 - accuracy: 0.8876
Epoch 00003: val_loss did not improve from 0.24889
113/113 [==============================] - 5s 41ms/step - loss: 0.2837 - accuracy: 0.8876 - val_loss: 0.9180 - val_accuracy: 0.5301 - lr: 0.0
010
Epoch 4/50
113/113 [==============================] - ETA: 0s - loss: 0.2506 - accuracy: 0.8967
Epoch 00004: val_loss did not improve from 0.24889
113/113 [==============================] - 5s 41ms/step - loss: 0.2506 - accuracy: 0.8967 - val_loss: 1.7537 - val_accuracy: 0.5471 - lr: 0.0
010
Epoch 5/50
113/113 [==============================] - ETA: 0s - loss: 0.2389 - accuracy: 0.8991
Epoch 00005: val_loss did not improve from 0.24889
113/113 [==============================] - 5s 41ms/step - loss: 0.2389 - accuracy: 0.8991 - val_loss: 0.2762 - val_accuracy: 0.8803 - lr: 0.0
010
Epoch 6/50
113/113 [==============================] - ETA: 0s - loss: 0.2185 - accuracy: 0.9057
Epoch 00006: val_loss improved from 0.24889 to 0.21064, saving model to /content/model-tgs-salt.h5
113/113 [==============================] - 5s 42ms/step - loss: 0.2185 - accuracy: 0.9057 - val_loss: 0.2106 - val_accuracy: 0.9105 - lr: 0.0
010
Epoch 7/50
113/113 [==============================] - ETA: 0s - loss: 0.2154 - accuracy: 0.9079
Epoch 00007: val_loss did not improve from 0.21064
113/113 [==============================] - 5s 41ms/step - loss: 0.2154 - accuracy: 0.9079 - val_loss: 0.3783 - val_accuracy: 0.8383 - lr: 0.0
010
Epoch 8/50
113/113 [==============================] - ETA: 0s - loss: 0.2010 - accuracy: 0.9134
Epoch 00008: val_loss did not improve from 0.21064
113/113 [==============================] - 5s 41ms/step - loss: 0.2010 - accuracy: 0.9134 - val_loss: 0.2190 - val_accuracy: 0.9085 - lr: 0.0
010
Epoch 9/50
113/113 [==============================] - ETA: 0s - loss: 0.1907 - accuracy: 0.9185
Epoch 00009: val_loss did not improve from 0.21064
113/113 [==============================] - 5s 41ms/step - loss: 0.1907 - accuracy: 0.9185 - val_loss: 0.3868 - val_accuracy: 0.8302 - lr: 0.0
010
Epoch 10/50
113/113 [==============================] - ETA: 0s - loss: 0.1831 - accuracy: 0.9223
Epoch 00010: val_loss did not improve from 0.21064
113/113 [==============================] - 5s 41ms/step - loss: 0.1831 - accuracy: 0.9223 - val_loss: 0.3012 - val_accuracy: 0.8931 - lr: 0.0
010
Epoch 11/50
113/113 [==============================] - ETA: 0s - loss: 0.1767 - accuracy: 0.9233
Epoch 00011: val_loss improved from 0.21064 to 0.17648, saving model to /content/model-tgs-salt.h5
113/113 [==============================] - 5s 41ms/step - loss: 0.1767 - accuracy: 0.9233 - val_loss: 0.1765 - val_accuracy: 0.9286 - lr: 0.0
010
Epoch 12/50
113/113 [==============================] - ETA: 0s - loss: 0.1635 - accuracy: 0.9272
Epoch 00012: val_loss did not improve from 0.17648
113/113 [==============================] - 5s 41ms/step - loss: 0.1635 - accuracy: 0.9272 - val_loss: 0.1963 - val_accuracy: 0.9227 - lr: 0.0
```

```
010
Epoch 13/50
113/113 [==============================] - ETA: 0s - loss: 0.1579 - accuracy: 0.9315
Epoch 00013: val_loss did not improve from 0.17648
113/113 [==============================] - 5s 41ms/step - loss: 0.1579 - accuracy: 0.9315 - val_loss: 0.2231 - val_accuracy: 0.9120 - lr: 0.0
010
Epoch 14/50
113/113 [==============================] - ETA: 0s - loss: 0.1487 - accuracy: 0.9340
Epoch 00014: val_loss did not improve from 0.17648
113/113 [==============================] - 5s 41ms/step - loss: 0.1487 - accuracy: 0.9340 - val_loss: 0.2542 - val_accuracy: 0.8917 - lr: 0.0
010
Epoch 15/50
113/113 [==============================] - ETA: 0s - loss: 0.1429 - accuracy: 0.9363
Epoch 00015: val_loss did not improve from 0.17648
113/113 [==============================] - 5s 41ms/step - loss: 0.1429 - accuracy: 0.9363 - val_loss: 0.2211 - val_accuracy: 0.9008 - lr: 0.0
010
Epoch 16/50
113/113 [==============================] - ETA: 0s - loss: 0.1336 - accuracy: 0.9396
Epoch 00016: val_loss improved from 0.17648 to 0.16743, saving model to /content/model-tgs-salt.h5
113/113 [==============================] - 5s 41ms/step - loss: 0.1336 - accuracy: 0.9396 - val_loss: 0.1674 - val_accuracy: 0.9270 - lr: 0.0
010
Epoch 17/50
113/113 [==============================] - ETA: 0s - loss: 0.1354 - accuracy: 0.9379
Epoch 00017: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.1354 - accuracy: 0.9379 - val_loss: 0.2261 - val_accuracy: 0.8886 - lr: 0.0
010
Epoch 18/50
113/113 [==============================] - ETA: 0s - loss: 0.1127 - accuracy: 0.9475
Epoch 00018: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 40ms/step - loss: 0.1127 - accuracy: 0.9475 - val_loss: 0.2078 - val_accuracy: 0.9306 - lr: 0.0
010
Epoch 19/50
113/113 [==============================] - ETA: 0s - loss: 0.1116 - accuracy: 0.9481
Epoch 00019: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.1116 - accuracy: 0.9481 - val_loss: 0.2283 - val_accuracy: 0.9211 - lr: 0.0
010
Epoch 20/50
113/113 [==============================] - ETA: 0s - loss: 0.1129 - accuracy: 0.9485
Epoch 00020: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.1129 - accuracy: 0.9485 - val_loss: 0.2023 - val_accuracy: 0.9258 - lr: 0.0
010
Epoch 21/50
113/113 [==============================] - ETA: 0s - loss: 0.0941 - accuracy: 0.9554
Epoch 00021: ReduceLROnPlateau reducing learning rate to 0.001.

Epoch 00021: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.0941 - accuracy: 0.9554 - val_loss: 0.2989 - val_accuracy: 0.8798 - lr: 0.0
010
Epoch 22/50
113/113 [==============================] - ETA: 0s - loss: 0.0870 - accuracy: 0.9582
Epoch 00022: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.0870 - accuracy: 0.9582 - val_loss: 0.2233 - val_accuracy: 0.9175 - lr: 0.0
010
Epoch 23/50
113/113 [==============================] - ETA: 0s - loss: 0.0803 - accuracy: 0.9604
Epoch 00023: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.0803 - accuracy: 0.9604 - val_loss: 0.2059 - val_accuracy: 0.9257 - lr: 0.0
010
Epoch 24/50
113/113 [==============================] - ETA: 0s - loss: 0.0780 - accuracy: 0.9617
Epoch 00024: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.0780 - accuracy: 0.9617 - val_loss: 0.3070 - val_accuracy: 0.8993 - lr: 0.0
010
Epoch 25/50
113/113 [==============================] - ETA: 0s - loss: 0.0809 - accuracy: 0.9602
Epoch 00025: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.0809 - accuracy: 0.9602 - val_loss: 0.2360 - val_accuracy: 0.9257 - lr: 0.0
010
Epoch 26/50
113/113 [==============================] - ETA: 0s - loss: 0.0731 - accuracy: 0.9638
Epoch 00026: ReduceLROnPlateau reducing learning rate to 0.001.

Epoch 00026: val_loss did not improve from 0.16743
113/113 [==============================] - 5s 41ms/step - loss: 0.0731 - accuracy: 0.9638 - val_loss: 0.2228 - val_accuracy: 0.9264 - lr: 0.0
010
Epoch 00026: early stopping
```

In [37]:

```
plt.figure(figsize=(8, 8))
plt.title("Learning curve")
plt.plot(history.history["loss"], label="loss")
```

```
plt.plot(history.history["loss"], label="loss")
plt.plot(history.history["val_loss"], label="val_loss")
plt.plot( np.argmin(history.history["val_loss"]), np.min(history.history["val_loss"]), marker="x", color="r", label="best model")
plt.xlabel("Epochs")
plt.ylabel("log_loss")
plt.legend()
plt.show()
```



Inference

In [0]:

```
# load the best model
model.load_weights('/content/model-tgs-salt.h5')
```

In [40]:

```
score = model.evaluate(x = X_valid, y= y_valid)
print('Test Loss: ', score[0])
print()
print('Test Accuracy: ', score[1])
```

```
13/13 [==============================] - 0s 12ms/step - loss: 0.1674 - accuracy: 0.9270
Test Loss:  0.16743120551109314

Test Accuracy:  0.9269952178001404
```

In [41]:

```
train_pred= model.predict(x= X_train, verbose= 1)
val_pred= model.predict(x= X_valid)
```

```
113/113 [==============================] - 1s 13ms/step
```

In [0]:

```
# Threshold_predictions
train_pred_t= (train_pred > 0.5).astype(np.uint8)

val_pred_t= (val_pred > 0.5).astype(np.uint8)
```

In [0]:

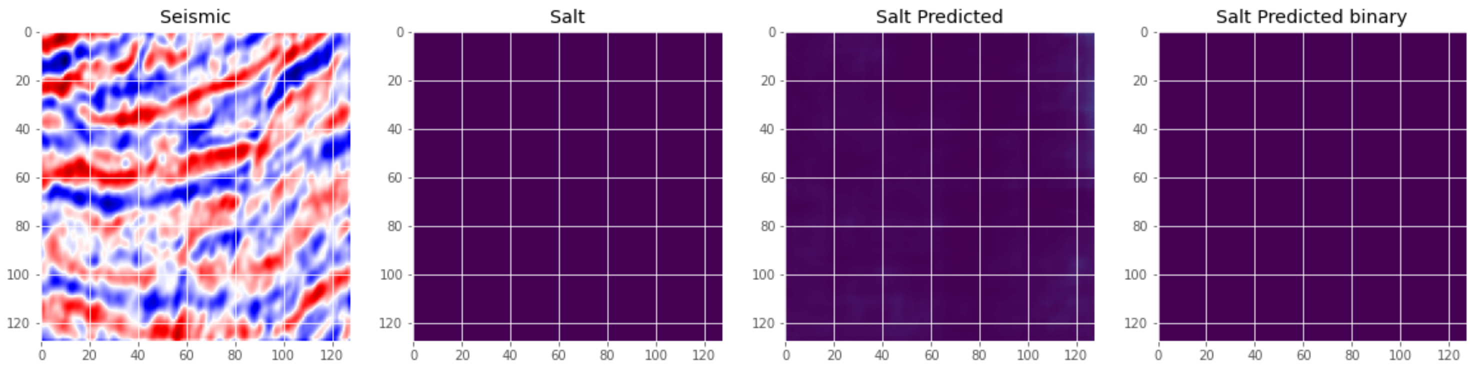
```
def plot_sample(X, y, preds, binary_preds, ix=None):
    """Function to plot the results"""
```

```python
    # Function to plot the results
    if ix is None:
        ix = random.randint(0, len(X))

    has_mask = y[ix].max() > 0

    fig, ax = plt.subplots(1, 4, figsize=(20, 10))
    ax[0].imshow(X[ix, ..., 0], cmap='seismic')
    if has_mask:
        ax[0].contour(y[ix].squeeze(), colors='k', levels=[0.5])
    ax[0].set_title('Seismic')

    ax[1].imshow(y[ix].squeeze())
    ax[1].set_title('Salt')

    ax[2].imshow(preds[ix].squeeze(), vmin=0, vmax=1)
    if has_mask:
        ax[2].contour(y[ix].squeeze(), colors='k', levels=[0.5])
    ax[2].set_title('Salt Predicted')

    ax[3].imshow(binary_preds[ix].squeeze(), vmin=0, vmax=1)
    if has_mask:
        ax[3].contour(y[ix].squeeze(), colors='k', levels=[0.5])
    ax[3].set_title('Salt Predicted binary');
```

Predit on Training set and Val set

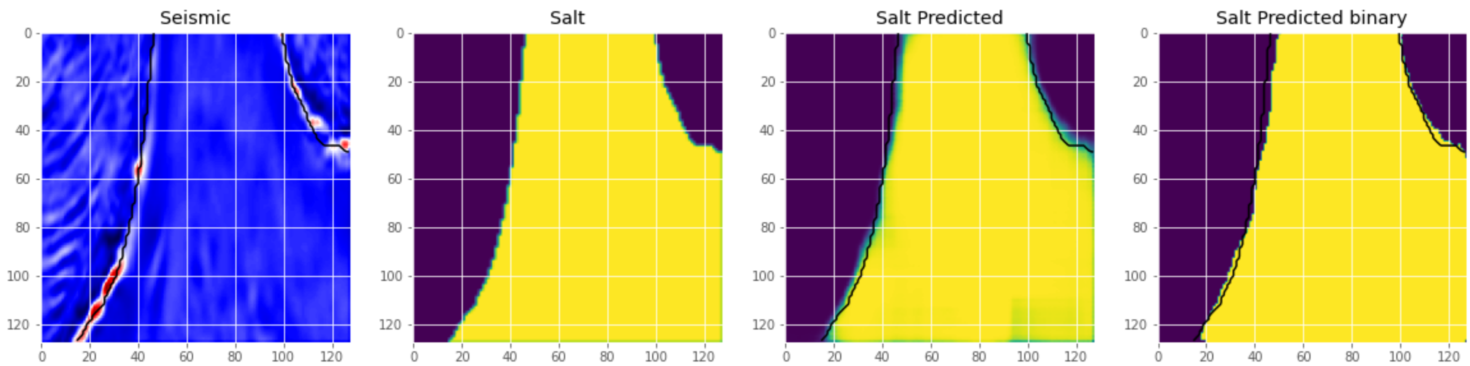
In [48]:

```python
# Check if training data looks all right
# ix is any number between (1 - 3600)
plot_sample(X_train, y_train, train_pred, train_pred_t, ix=14)
```



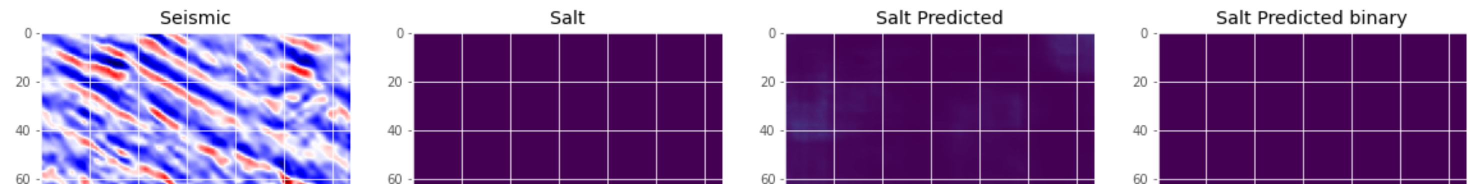
In [54]:

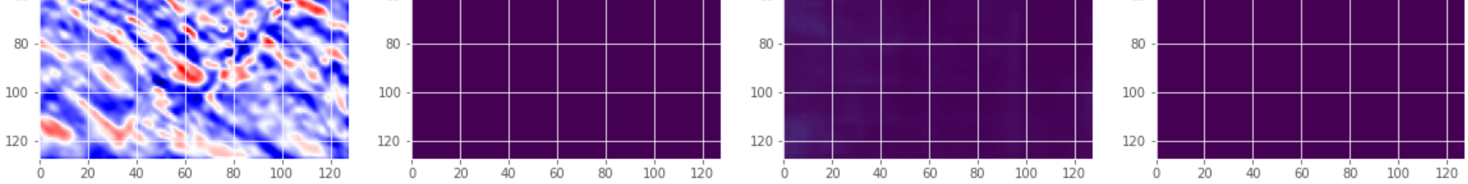
```python
plot_sample(X_train, y_train, train_pred, train_pred_t, ix= 114)
```



In [57]:

```python
plot_sample(X_valid, y_valid, val_pred, val_pred_t, ix= 394)
```





In [53]:

```
# ix is any number between (1 - 400)
plot_sample(X_valid, y_valid, val_pred, val_pred_t, ix= 30)
```