Satellite-Image-Semantic-Segment-Unet

March 4, 2023

[57]:
```
import os
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import gc
from tqdm.notebook import trange, tqdm
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 glob
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
 ↪array_to_img, img_to_array, load_img
from tensorflow.keras.layers import Conv2D, Input, MaxPooling2D, Dropout,␣
 ↪concatenate, UpSampling2D
from tensorflow.keras.models import load_model, Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,␣
 ↪ReduceLROnPlateau, TensorBoard
from tensorflow.keras import backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (BatchNormalization, Conv2DTranspose,
                                     SeparableConv2D, MaxPooling2D, Activation,␣
 ↪Flatten, Dropout, Dense)
from keras.callbacks import CSVLogger
K.clear_session()
warnings.filterwarnings('ignore')
plt.style.use("ggplot")
%matplotlib inline
```

[32]:
```
#Parameters
w, h = 256,256
border = 5
```

```
ids = next(os.walk("SegTMS/train/"))[1]
print("No. of folder = ", len(ids))
```

```
No. of folder =  2
```

[33]:
```
#Load data
train = sorted(glob.glob("SegTMS/train/Images/*"))
train_mask = sorted(glob.glob("SegTMS/train/Labels/*.png"))#[:64]
print(f'Total Train Images : {len(train)}\nTotal Mask Image : ␣
 ↪{len(train_mask)}')
```

```
Total Train Images : 200
Total Mask Image : 200
```

[34]:
```
#data processing

X = np.zeros((len(train), h, w, 3), dtype=np.float32)
y = np.zeros((len(train_mask), h, w, 1), dtype=np.float32)


for n, (img, mimg) in tqdm(enumerate(zip(train, train_mask))):
    # Load images
    img = load_img(img)
    x_img = img_to_array(img)
    x_img = resize(x_img, (h, w, 3), mode = 'constant', preserve_range = True)
    # # Load masks
    mask = img_to_array(load_img(mimg, color_mode = "grayscale"))
    mask = resize(mask, (h, w, 1), mode = 'constant', preserve_range = True)
    # # Save images
    X[n] = x_img/255.0
    y[n] = mask/255.0
```

```
0it [00:00, ?it/s]
```

[43]:
```
# Save as in Numpy array
np.save('SegTMS/XandY/X.npy', X)
np.save('SegTMS/XandY/y.npy', y)
# print(X.shape, y.shape)

X = np.load('SegTMS/XandY/X.npy')
y = np.load('SegTMS/XandY/y.npy')
print(X.shape, y.shape)
```

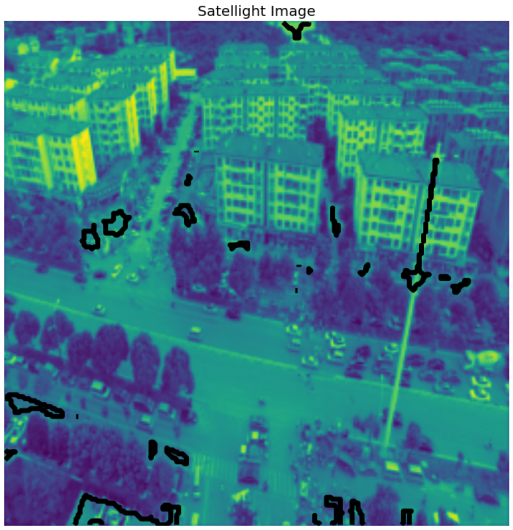
```
(200, 256, 256, 3) (200, 256, 256, 1)
```

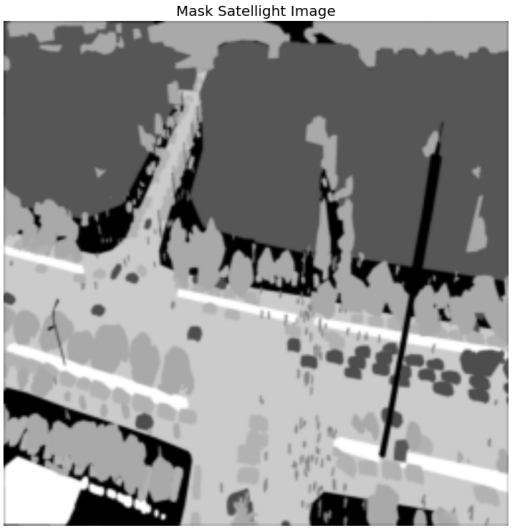
[44]:
```
# Split train and valid
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
↪random_state=42)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
gc.collect()
```

```
(180, 256, 256, 3) (180, 256, 256, 1) (20, 256, 256, 3) (20, 256, 256, 1)
```

[44]:
```
6868
```

[45]:
```
# Visualize any randome image along with the mask
ix = random.randint(0, len(X_train))
has_mask = y_train[ix].max() > 0 # salt indicator

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 15))

ax1.imshow(X_train[ix, ..., 0], interpolation = 'bilinear')

if has_mask:
    ax1.contour(y_train[ix].squeeze(), colors = 'k', linewidths = 5, levels =␣
↪[0.5])
ax1.set_title('Satellight Image')
ax1.set_axis_off()
ax2.imshow(y_train[ix].squeeze(), cmap = 'gray', interpolation = 'bilinear')
ax2.set_title('Mask Satellight Image')
ax2.set_axis_off()
```





[51]:
```
#valid tensor gen
nbatch=8
```

```
dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train)).batch(nbatch)
valset = tf.data.Dataset.from_tensor_slices((X_test, y_test)).batch(nbatch)
```

[52]:
```
#unet layes
def conv2d_block(input_tensor, n_filters, kernel_size = 3, batchnorm = True):
    # first layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size,␣
 ↪kernel_size),kernel_initializer = 'he_normal', padding =␣
 ↪'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # second layer
    x = Conv2D(filters = n_filters, kernel_size = (kernel_size,␣
 ↪kernel_size),kernel_initializer = 'he_normal', padding =␣
 ↪'same')(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)

    return x
```

[53]:
```
# UNET Model

def Unet(input_img, n_filters = 16, dropout = 0.1, batchnorm = True):
    # Contracting Path
    c1 = conv2d_block(input_img, n_filters * 1, kernel_size = 3, batchnorm =␣
 ↪batchnorm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)

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

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

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

    c5 = conv2d_block(p4, n_filters = n_filters * 16, kernel_size = 3,␣
 ↪batchnorm = batchnorm)
```

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

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

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

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

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

[54]:
```
#model set

input_img = Input((h, w, 3), name='img')
model = Unet(input_img, n_filters=16, dropout=0.05, batchnorm=True)
metrics = ["accuracy",
           tf.keras.metrics.AUC(),
           tf.keras.metrics.SensitivityAtSpecificity(0.5),
           tf.keras.metrics.SpecificityAtSensitivity(0.5)]
model.compile(optimizer=Adam(), loss="binary_crossentropy", metrics=metrics)
gc.collect()
```

[54]:
```
8796
```

[55]:
```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________
```

```
------------------
 Layer (type)                Output Shape              Param #     Connected to
==================================================================================
==================
 img (InputLayer)            [(None, 256, 256, 3  0           []
                             )]

 conv2d_1 (Conv2D)           (None, 256, 256, 16  448         ['img[0][0]']
                             )

 batch_normalization_1 (BatchNo  (None, 256, 256, 16  64
['conv2d_1[0][0]']
 rmalization)                )

 activation_1 (Activation)   (None, 256, 256, 16  0
['batch_normalization_1[0][0]']
                             )

 max_pooling2d (MaxPooling2D)   (None, 128, 128, 16  0
['activation_1[0][0]']
                             )

 dropout (Dropout)           (None, 128, 128, 16  0
['max_pooling2d[0][0]']
                             )

 conv2d_3 (Conv2D)           (None, 128, 128, 32  4640
['dropout[0][0]']
                             )

 batch_normalization_3 (BatchNo  (None, 128, 128, 32  128
['conv2d_3[0][0]']
 rmalization)                )

 activation_3 (Activation)   (None, 128, 128, 32  0
['batch_normalization_3[0][0]']
                             )

 max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 32)  0
['activation_3[0][0]']

 dropout_1 (Dropout)         (None, 64, 64, 32)   0
['max_pooling2d_1[0][0]']

 conv2d_5 (Conv2D)           (None, 64, 64, 64)   18496
['dropout_1[0][0]']

 batch_normalization_5 (BatchNo  (None, 64, 64, 64)  256
```

```
['conv2d_5[0][0]']
 rmalization)

 activation_5 (Activation)      (None, 64, 64, 64)   0
['batch_normalization_5[0][0]']

 max_pooling2d_2 (MaxPooling2D)  (None, 32, 32, 64)  0
['activation_5[0][0]']

 dropout_2 (Dropout)            (None, 32, 32, 64)   0
['max_pooling2d_2[0][0]']

 conv2d_7 (Conv2D)              (None, 32, 32, 128)  73856
['dropout_2[0][0]']

 batch_normalization_7 (BatchNo  (None, 32, 32, 128)  512
['conv2d_7[0][0]']
 rmalization)

 activation_7 (Activation)      (None, 32, 32, 128)  0
['batch_normalization_7[0][0]']

 max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 128)  0
['activation_7[0][0]']

 dropout_3 (Dropout)            (None, 16, 16, 128)  0
['max_pooling2d_3[0][0]']

 conv2d_9 (Conv2D)              (None, 16, 16, 256)  295168
['dropout_3[0][0]']

 batch_normalization_9 (BatchNo  (None, 16, 16, 256)  1024
['conv2d_9[0][0]']
 rmalization)

 activation_9 (Activation)      (None, 16, 16, 256)  0
['batch_normalization_9[0][0]']

 conv2d_transpose (Conv2DTransp  (None, 32, 32, 128)  295040
['activation_9[0][0]']
 ose)

 concatenate (Concatenate)      (None, 32, 32, 256)  0
['conv2d_transpose[0][0]',
'activation_7[0][0]']

 dropout_4 (Dropout)            (None, 32, 32, 256)  0
['concatenate[0][0]']
```

```
 conv2d_11 (Conv2D)             (None, 32, 32, 128)  295040
['dropout_4[0][0]']

 batch_normalization_11 (BatchN  (None, 32, 32, 128)  512
['conv2d_11[0][0]']
 ormalization)

 activation_11 (Activation)     (None, 32, 32, 128)  0
['batch_normalization_11[0][0]']

 conv2d_transpose_1 (Conv2DTran  (None, 64, 64, 64)  73792
['activation_11[0][0]']
 spose)

 concatenate_1 (Concatenate)    (None, 64, 64, 128)  0
['conv2d_transpose_1[0][0]',
'activation_5[0][0]']

 dropout_5 (Dropout)            (None, 64, 64, 128)  0
['concatenate_1[0][0]']

 conv2d_13 (Conv2D)             (None, 64, 64, 64)   73792
['dropout_5[0][0]']

 batch_normalization_13 (BatchN  (None, 64, 64, 64)  256
['conv2d_13[0][0]']
 ormalization)

 activation_13 (Activation)     (None, 64, 64, 64)   0
['batch_normalization_13[0][0]']

 conv2d_transpose_2 (Conv2DTran  (None, 128, 128, 32  18464
['activation_13[0][0]']
 spose)                         )

 concatenate_2 (Concatenate)    (None, 128, 128, 64  0
['conv2d_transpose_2[0][0]',
                                )
'activation_3[0][0]']

 dropout_6 (Dropout)            (None, 128, 128, 64  0
['concatenate_2[0][0]']
                                )

 conv2d_15 (Conv2D)             (None, 128, 128, 32  18464
['dropout_6[0][0]']
                                )
```

```
 batch_normalization_15 (BatchN  (None, 128, 128, 32  128
['conv2d_15[0][0]']
 ormalization)                  )

 activation_15 (Activation)     (None, 128, 128, 32  0
['batch_normalization_15[0][0]']
                                )

 conv2d_transpose_3 (Conv2DTran  (None, 256, 256, 16  4624
['activation_15[0][0]']
 spose)                         )

 concatenate_3 (Concatenate)    (None, 256, 256, 32  0
['conv2d_transpose_3[0][0]',
                                )
'activation_1[0][0]']

 dropout_7 (Dropout)            (None, 256, 256, 32  0
['concatenate_3[0][0]']
                                )

 conv2d_17 (Conv2D)             (None, 256, 256, 16  4624
['dropout_7[0][0]']
                                )

 batch_normalization_17 (BatchN  (None, 256, 256, 16  64
['conv2d_17[0][0]']
 ormalization)                  )

 activation_17 (Activation)     (None, 256, 256, 16  0
['batch_normalization_17[0][0]']
                                )

 conv2d_18 (Conv2D)             (None, 256, 256, 1)  17
['activation_17[0][0]']

==================================================================================
==================
Total params: 1,179,409
Trainable params: 1,177,937
Non-trainable params: 1,472
----------------------------------------------------------------------------------
------------------
```

[61]:
```
callbacks = [
    EarlyStopping(patience=100, verbose=1),
```

```
    ReduceLROnPlateau(factor=0.1, patience=25, min_lr=0.00001, verbose=1),
    ModelCheckpoint('modelUnet.h5', verbose=1, save_best_only=True,␣
 ↪save_weights_only=True),
    CSVLogger("dataResUnet.csv")]
```

[62]:
```
nepochs=1
results = model.fit(X_train, y_train, batch_size=nbatch, epochs=nepochs,␣
 ↪callbacks=callbacks,
                    validation_data=(X_test, y_test), use_multiprocessing=True)
```

```
23/23 [==============================] - ETA: 0s - loss: 0.5906 - accuracy:
0.0931 - auc: 0.5601 - sensitivity_at_specificity: 0.5875 -
specificity_at_sensitivity: 0.5907
Epoch 1: val_loss improved from inf to 6.20001, saving model to modelUnet.h5
23/23 [==============================] - 62s 2s/step - loss: 0.5906 - accuracy:
0.0931 - auc: 0.5601 - sensitivity_at_specificity: 0.5875 -
specificity_at_sensitivity: 0.5907 - val_loss: 6.2000 - val_accuracy: 0.0272 -
val_auc: 0.5205 - val_sensitivity_at_specificity: 0.5263 -
val_specificity_at_sensitivity: 0.5185 - lr: 0.0010
```

[63]:
```
df_result = pd.DataFrame(results.history)
df_result.sort_values('val_loss', ascending=True, inplace = True)
df_result
```

[63]:

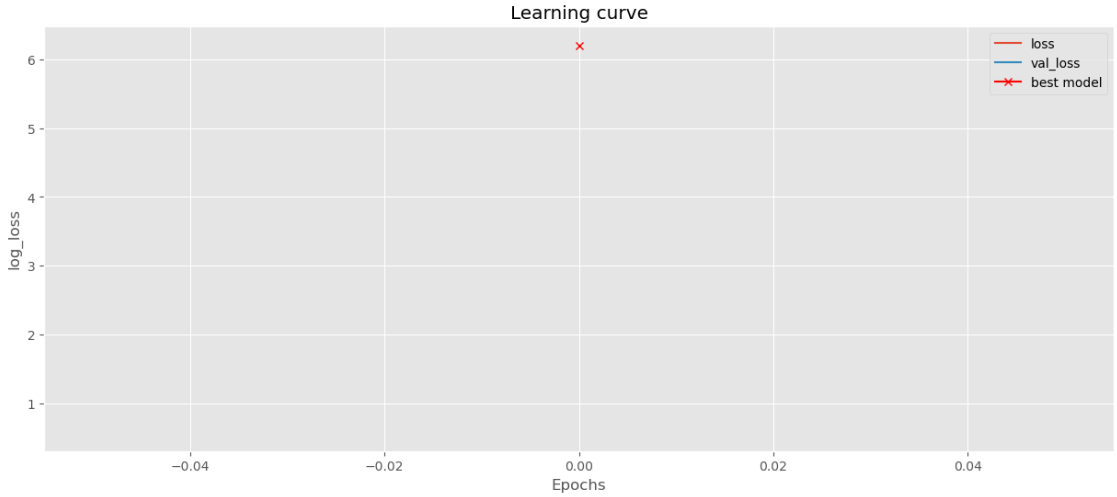
	loss	accuracy	auc	sensitivity_at_specificity	\
0	0.590563	0.093097	0.560148	0.587481	

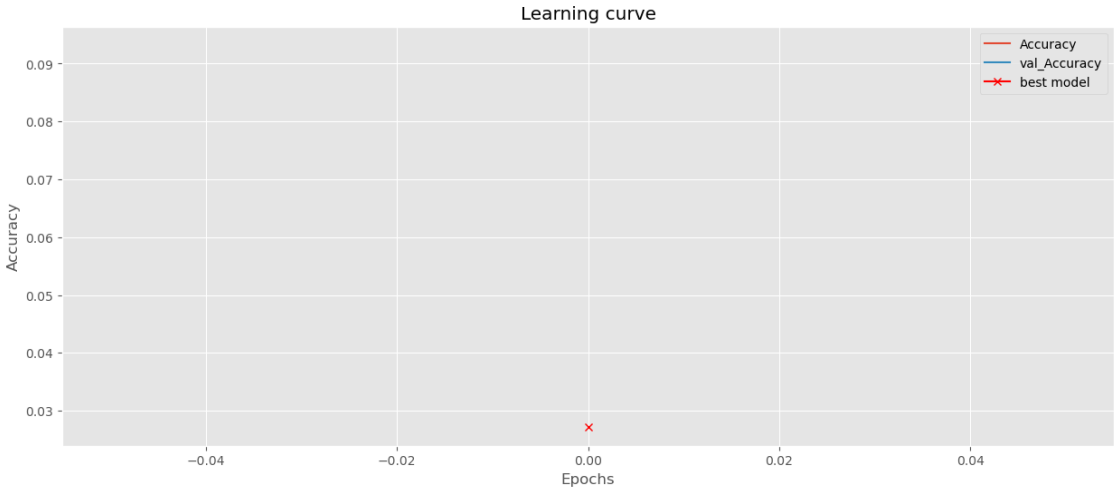
	specificity_at_sensitivity	val_loss	val_accuracy	val_auc	\
0	0.590679	6.20001	0.027247	0.52049	

	val_sensitivity_at_specificity	val_specificity_at_sensitivity	lr
0	0.526263	0.518489	0.001

[64]:
```
#plot performances
plt.figure(figsize = (15,6))
plt.title("Learning curve")
plt.plot(results.history["loss"], label="loss")
plt.plot(results.history["val_loss"], label="val_loss")
plt.plot(np.argmin(results.history["val_loss"]), np.min(results.
 ↪history["val_loss"]), marker="x", color="r", label="best model")
plt.xlabel("Epochs")
plt.ylabel("log_loss")
plt.legend();
#-------------
plt.figure(figsize = (15,6))
plt.title("Learning curve")
plt.plot(results.history["accuracy"], label="Accuracy")
```

```
plt.plot(results.history["val_accuracy"], label="val_Accuracy")
plt.plot(np.argmax(results.history["val_accuracy"]), np.max(results.
 ↪history["val_accuracy"]), marker="x", color="r", label="best model")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend();
```





[65]:
```
model.load_weights('modelUnet.h5')

model.evaluate(X_test, y_test, verbose=1)
```

```
1/1 [==============================] - 1s 1s/step - loss: 6.2000 - accuracy:
0.0272 - auc: 0.5205 - sensitivity_at_specificity: 0.5263 -
```

```
specificity_at_sensitivity: 0.5185
```

[65]:
```
[6.200010299682617,
 0.027246857061982155,
 0.5204902291297913,
 0.526262640953064,
 0.5184887051582336]
```

[66]:
```
# Predict on train, val and test
preds_train = model.predict(X_train, verbose=1)
preds_val = model.predict(X_test, verbose=1)
```

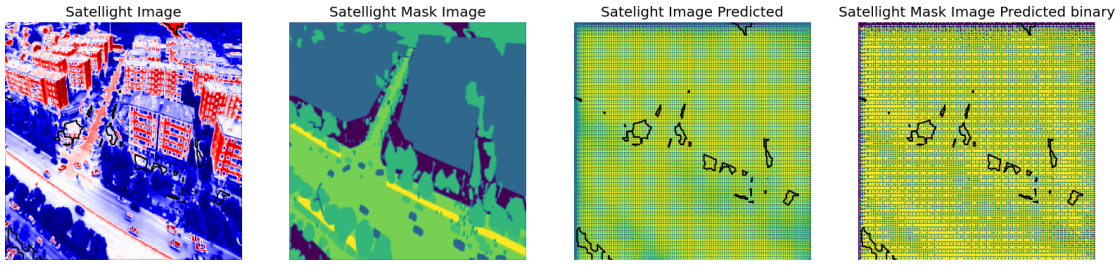
```
6/6 [==============================] - 8s 1s/step
1/1 [==============================] - 1s 854ms/step
```

[67]:
```
# Threshold predictions
preds_train_t = (preds_train > 0.5).astype(np.uint8)
preds_val_t = (preds_val > 0.5).astype(np.uint8)
```

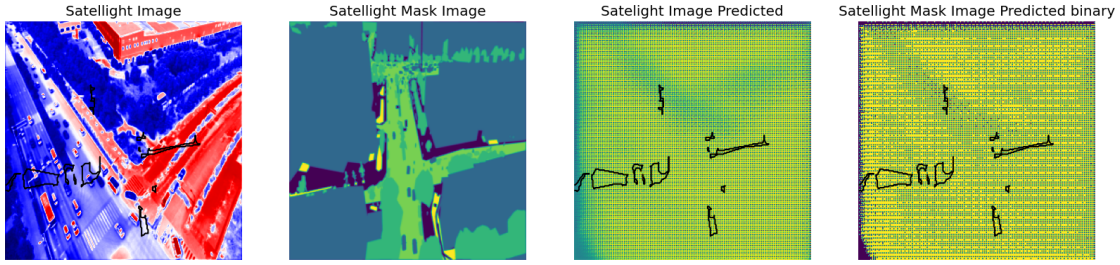
[68]:
```
# plot results

def plot_sample(X, y, preds, binary_preds, ix=None):

    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('Satellight Image')
    ax[0].set_axis_off()

    ax[1].imshow(y[ix].squeeze())
    ax[1].set_title('Satellight Mask Image')
    ax[1].set_axis_off()

    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('Satelight Image Predicted')
    ax[2].set_axis_off()

    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('Satellight Mask Image Predicted binary');
    ax[3].set_axis_off()
    plt.show()
```

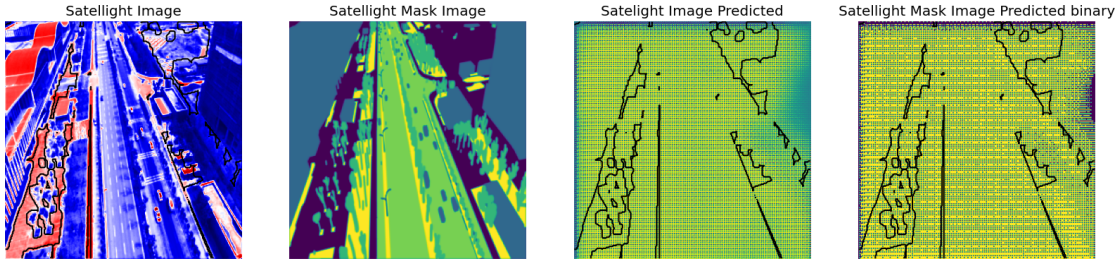
[72]:
```
#Predictions on training set
plot_sample(X_train, y_train, preds_train, preds_train_t, ix=5)
```



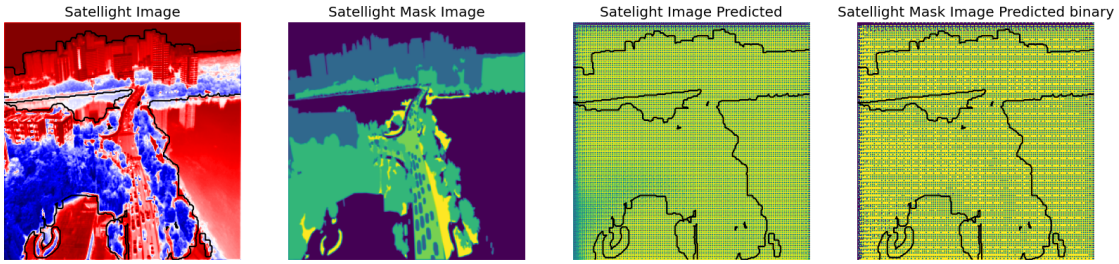
[70]:
```
#Predictions on training set
plot_sample(X_train, y_train, preds_train, preds_train_t, ix=10)
```



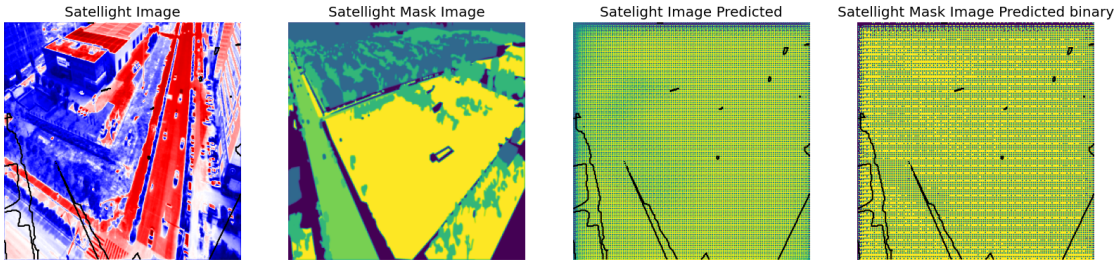
[71]:
```
#Predictions on training set
plot_sample(X_train, y_train, preds_train, preds_train_t, ix=15)
```



[69]:
```
#Predictions on training set
plot_sample(X_train, y_train, preds_train, preds_train_t, ix=20)
```



[73]:
```
#Predictions on training set
plot_sample(X_train, y_train, preds_train, preds_train_t, ix=25)
```



[ ]:
```

```