U-Net: Convolutional Networks for Biomedical Image Segmentation

Abstract

The segmentation of biomedical images typically deals with partitioning an image into multiple regions representing anatomical objects of interest. A variety of medical image segmentation problems present significant technical challenges, including heterogeneous pixel intensities, noisy/ill-defined boundaries, and irregular shapes with high variability. This paper uses a convolutional neural network to train the model and perform image segmentation. It presents a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. This strategy outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks.

Introduction

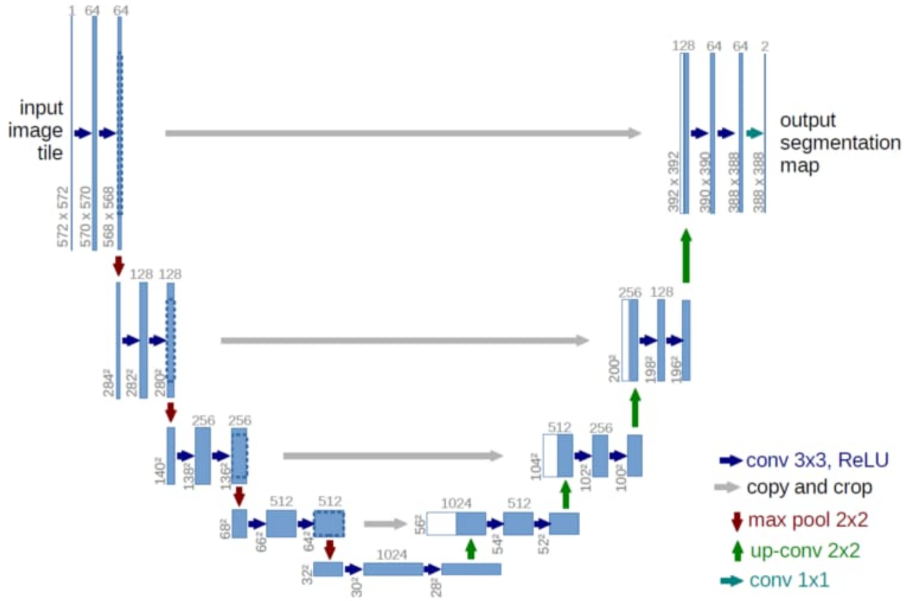


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Fig.1 U-net as proposed in paper

We can see that the network is composed of Convolution Operation, Max Pooling, ReLU Activation, Concatenation and Up Sampling Layers. The above figure shows the network proposed in the actual paper. We have used a similar strategy except for the number of channels and a few other parameters.

Dataset and preprocessing techniques

1. *Dataset used* → kaggle data-science-bowl-2018. Click here to view.
2. *Preprocessing techniques* → The image provided in the above dataset is n *n pixels which is reduced to 128* 128 prior to training model.

Pipeline Used

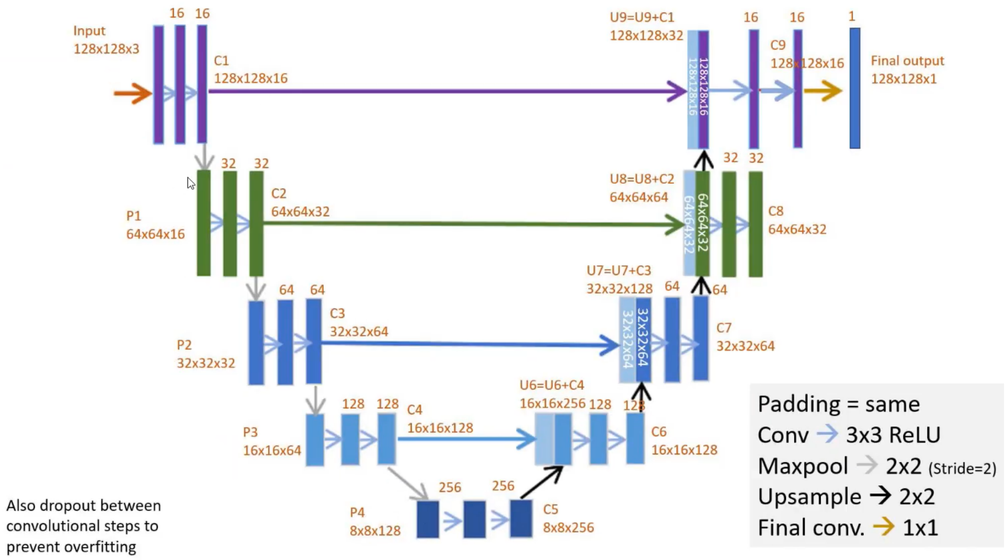


Fig.2 Our Pipeline

> The above figure is the neural network that we have implemented. It uses a slightly different channel width than the initially proposed architecture in the paper.

What else is different?

1. *Used different Optimizer* → As seen below, the original paper used stochastic gradient descent optimizer, we just used an Adam Optimizer.

3 Training

The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent implementation of Caffe [6]. Due to the unpadded convolutions, the output image is smaller than the input

Fig.3

2. *Used different loss function* → As seen below, the original paper have used softmax with cross entropy loss function, we just used an binary_crossentropy.

update in the current optimization step.
The energy function is computed by a pixel-wise soft-max over the final feature map combined with the cross entropy loss function. The soft-max is defined as $p_k(\mathbf{x}) = \exp(a_k(\mathbf{x}))/\left(\sum_{k'=1}^{K}\exp(a_{k'}(\mathbf{x}))\right)$ where $a_k(\mathbf{x})$ denotes the

Fig.4

The operations involved

(i) Convolution operation

We are using a 3*3 kernel for convulution and ReLu for activation.

(ii) Max pooling operation

We are max pooling the convolutional layers on the left side using a 2x2 filter. Here, we select the max value of the 4 entries in the 2x2 region in the input feature map and discard all the others. This reduces the dimension of the image by half in both the directions.

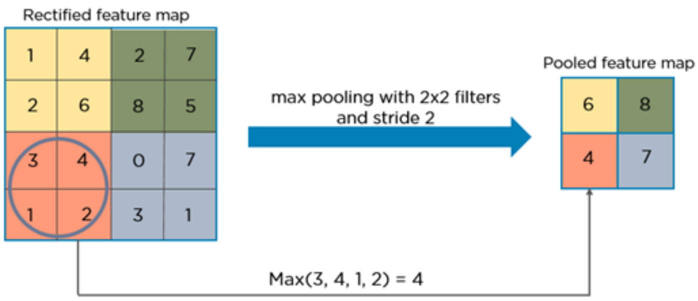


Fig.5 Max pooling.

This is to retain only the important information in the context and to get a better understanding of **"WHAT"** is present in the image rather than **"WHERE"** is it present.

(iii)Up sampling using Transposed convolution

This step is performed in the right side of the U-Net. It is required to convert a low resolution image to a high resolution image to recover the information of the **"WHERE"** of the segments.

We have used Transposed convolultion to perform up sampling. The transposed convolution operation forms the same connectivity as the normal convolution but in the backward direction. We can use it to conduct up-sampling. Moreover, the weights in the transposed convolution are learnable.

It can be noted that we are concatenating feature maps from corresponding downsampling layers for more precise localisation.

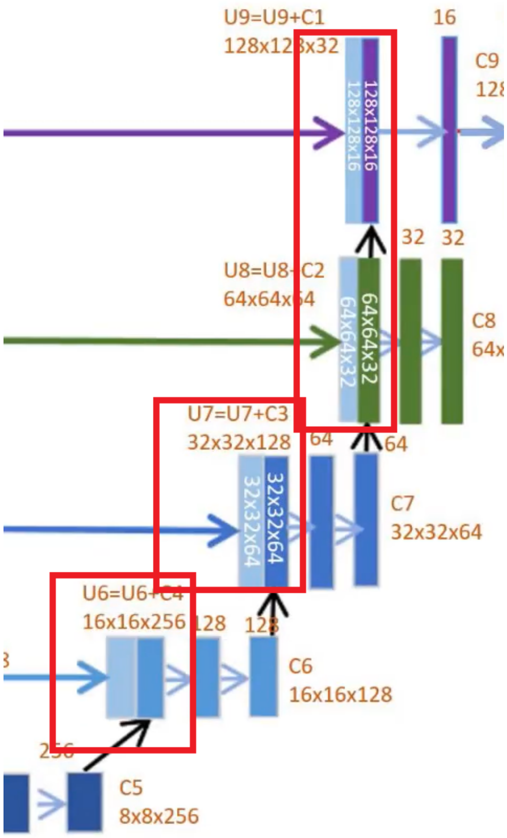


Fig.6 Transposed convolution and concatenation.

Evaluation metrics Used

Pixel by pixel accuracy is not a good measure to evaluate segmentation because it doesn't take into account class imbalance. When our classes are extremely imbalanced, it means that a class or some classes dominate the image, while some other classes make up only a small portion of the image. This means a poorly segmented result might yield a high accuracy.

Here, performace metrics such as IOU and DICE comes into picture.

1. *Intersection over Union(IoU)* → It is an accuracy metric that calculates accuracy as a ratio of area of overlap and area of union.

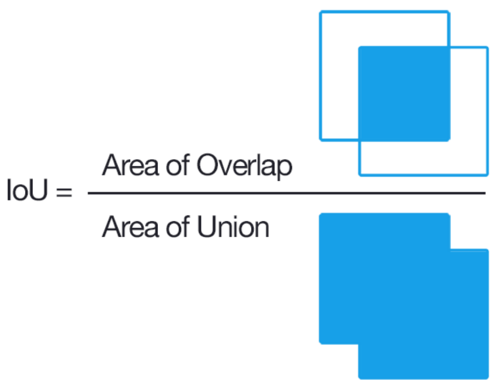


Fig.7 IoU

In confusion matrix terms IoU looks like this:

$$\frac{TP}{TP + FP + FN}$$

We used this metric because of it's ability to reward heavy overlap with actual masks.

Fig.8 An example of computing Intersection over Unions for various cases.

From figure 3 predicted masks that heavily overlap with the actual masks have higher scores than those with less overlap. We aren't concerned with an exact match of (x, y)-coordinates, but we do want to ensure that our predicted masks match as closely as possible — Intersection over Union is able to take this into account

2. *Dice coefficient* → It calculates performance as a ratio of twice the area of intersection and total area.

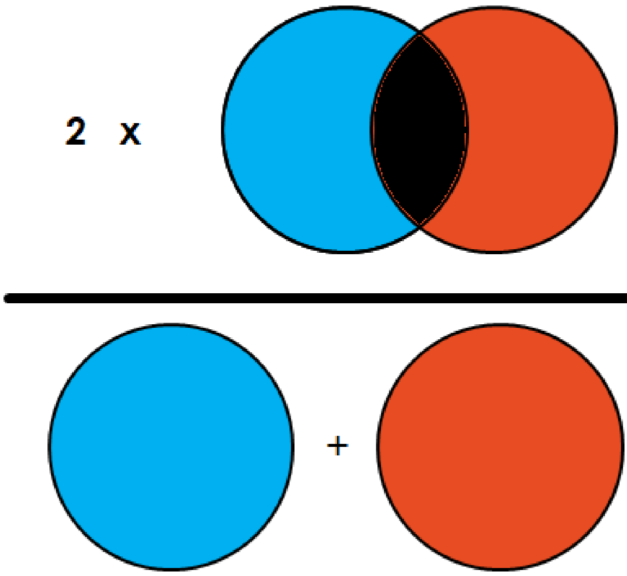


Fig.9 An example of computing Dice coefficient.

In confusion matrix terms dice coefficient looks like this:

$$\frac{2TP}{2TP + FP + FN}$$

Implementation

Installing and importing Required libraries

In [1]:

```
# !pip3 install tqdm
# !pip3 install tensorflow
# !pip3 install scikit-image
import tensorflow as tf
import os
import random
import numpy as np

from tqdm import tqdm
```

```
/home/mukesh/.local/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of typ
e is deprecated; in a future version of numpy, it will be understood as (typ
e, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of typ
e is deprecated; in a future version of numpy, it will be understood as (typ
e, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of typ
e is deprecated; in a future version of numpy, it will be understood as (typ
e, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of typ
e is deprecated; in a future version of numpy, it will be understood as (typ
e, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of typ
e is deprecated; in a future version of numpy, it will be understood as (typ
e, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorflow/python/framework/d
types.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of typ
e is deprecated; in a future version of numpy, it will be understood as (typ
e, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow
_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow
_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow
_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow
_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/mukesh/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow
_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
```

```
/home/mukesh/.local/lib/python3.6/site-packages/tensorboard/compat/tensorflow
_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym
of type is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

Preprocessing the data

- Dataset consists of nuclei images.
- Segmentation is performed on them.
- To create a mask identifying the nulcei, for such images.

In [2]:

```
from skimage.io import imread, imshow
from skimage.transform import resize
import matplotlib.pyplot as plt
%matplotlib inline

seed = 42
np.random.seed = seed

IMG_WIDTH = 128
IMG_HEIGHT = 128
IMG_CHANNELS = 3

TRAIN_PATH = 'stage1_train/'
# TEST_PATH = 'stage1_test/'

train_ids = next(os.walk(TRAIN_PATH))[1]
# test_ids = next(os.walk(TEST_PATH))[1]

X_train = np.zeros((len(train_ids), IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS), dty
Y_train = np.zeros((len(train_ids), IMG_HEIGHT, IMG_WIDTH, 1), dtype=np.bool)

print('Resizing training images and masks')
for n, id_ in tqdm(enumerate(train_ids), total=len(train_ids)):
    path = TRAIN_PATH + id_
    img = imread(path + '/images/' + id_ + '.png')[:,:,:IMG_CHANNELS]
    img = resize(img, (IMG_HEIGHT, IMG_WIDTH), mode='constant', preserve_rang
    X_train[n] = img  #Fill empty X_train with values from img
    mask = np.zeros((IMG_HEIGHT, IMG_WIDTH, 1), dtype=np.bool)
    for mask_file in next(os.walk(path + '/masks/'))[2]:
        mask_ = imread(path + '/masks/' + mask_file)
        mask_ = np.expand_dims(resize(mask_, (IMG_HEIGHT, IMG_WIDTH), mode='c
                                      preserve_range=True), axis=-1)
        mask = np.maximum(mask, mask_)

    Y_train[n] = mask

# # test images
# X_test = np.zeros((len(test_ids), IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS), dty
# sizes_test = []
# print('Resizing test images')
# for n, id_ in tqdm(enumerate(test_ids), total=len(test_ids)):
#     path = TEST_PATH + id_
#     img = imread(path + '/images/' + id_ + '.png')[:,:,:IMG_CHANNELS]
#     sizes_test.append([img.shape[0], img.shape[1]])
#     img = resize(img, (IMG_HEIGHT, IMG_WIDTH), mode='constant', preserve_ra
#     X_test[n] = img

print('Done!')
```

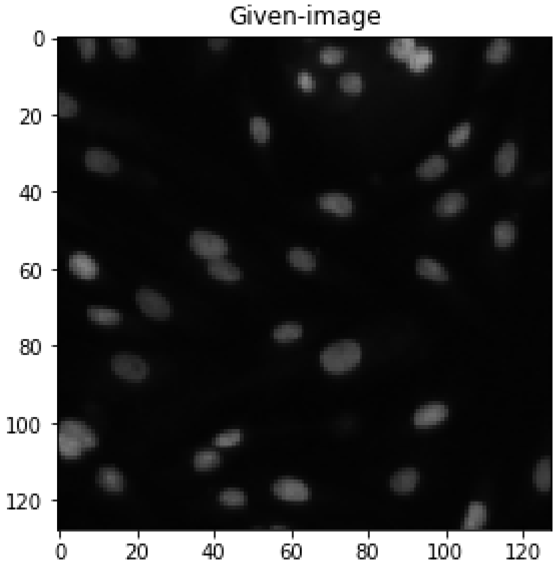
```
  0%|          | 1/670 [00:00<01:50,  6.05it/s]
Resizing training images and masks
```

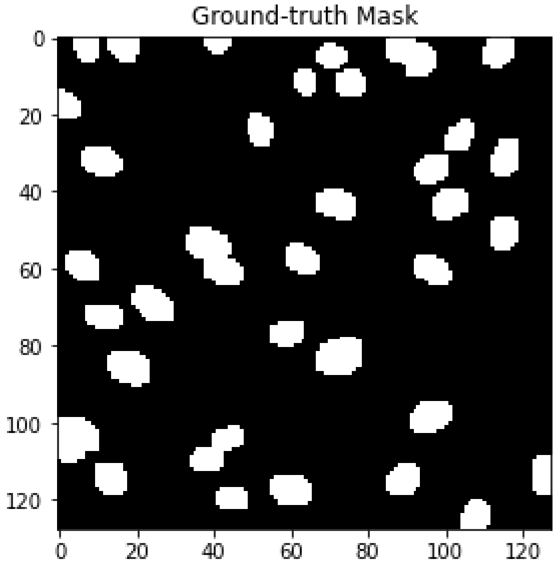
```
100%|██████████| 670/670 [06:23<00:00,  1.75it/s]
Done!
```

In [4]:

```
print("An Example of Processed Training Data")
image_x = random.randint(0, len(train_ids))
imshow(X_train[image_x])
plt.title("Given-image")
plt.show()

imshow(np.squeeze(Y_train[image_x]))
plt.title("Ground-truth Mask")
plt.show()
```

```
An Example of Processed Training Data
```





Defining the model Pipeline

In [5]:

```
#Building the model
inputs = tf.keras.layers.Input((IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS))
## Conv2D expects pixels in float values, also normalizing all the pixel valu
s = tf.keras.layers.Lambda(lambda x: x / 255)(inputs)

#Contraction path(Encoder)
c1 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer
c1 = tf.keras.layers.Dropout(0.1)(c1)
```

```
c1 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer
p1 = tf.keras.layers.MaxPooling2D((2, 2))(c1)

c2 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer
c2 = tf.keras.layers.Dropout(0.1)(c2)
c2 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer
p2 = tf.keras.layers.MaxPooling2D((2, 2))(c2)

c3 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer
c3 = tf.keras.layers.Dropout(0.2)(c3)
c3 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer
p3 = tf.keras.layers.MaxPooling2D((2, 2))(c3)

c4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initialize
c4 = tf.keras.layers.Dropout(0.2)(c4)
c4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initialize
p4 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(c4)

c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', kernel_initialize
c5 = tf.keras.layers.Dropout(0.3)(c5)
c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', kernel_initialize

#Expansive path(Decoder)
u6 = tf.keras.layers.Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='sa
u6 = tf.keras.layers.concatenate([u6, c4])
c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initialize
c6 = tf.keras.layers.Dropout(0.2)(c6)
c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initialize

u7 = tf.keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='sam
u7 = tf.keras.layers.concatenate([u7, c3])
c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer
c7 = tf.keras.layers.Dropout(0.2)(c7)
c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer

u8 = tf.keras.layers.Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='sam
u8 = tf.keras.layers.concatenate([u8, c2])
c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer
c8 = tf.keras.layers.Dropout(0.1)(c8)
c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer

u9 = tf.keras.layers.Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='sam
u9 = tf.keras.layers.concatenate([u9, c1], axis=3)
c9 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer
c9 = tf.keras.layers.Dropout(0.1)(c9)
c9 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer

outputs = tf.keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c9)

model = tf.keras.Model(inputs=[inputs], outputs=[outputs])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accurac
model.summary()

################################
#Modelcheckpoint-------to save the model state in a file, if for some reason
checkpointer = tf.keras.callbacks.ModelCheckpoint('model_for_nuclei.h5', verb

# Using EarlyStooping to avoid overfitting
# Also using TensorBoard to visualize final training(or/and validation) vs ep
callbacks = [
        tf.keras.callbacks.EarlyStopping(patience=2, monitor='val_loss'),
        tf.keras.callbacks.TensorBoard(log_dir='logs'),
        checkpointer]
```

```
WARNING:tensorflow:From /home/mukesh/.local/lib/python3.6/site-packages/tenso
```

```
rflow/python/ops/init_ops.py:1251: calling VarianceScaling.__init__ (from ten
sorflow.python.ops.init_ops) with dtype is deprecated and will be removed in
a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to th
e constructor
WARNING:tensorflow:From /home/mukesh/.local/lib/python3.6/site-packages/tenso
rflow/python/ops/nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "model"
_____________________________________________________________________________
_____________________
Layer (type)                    Output Shape         Param #     Connected to
=============================================================================
=====================
input_1 (InputLayer)            [(None, 128, 128, 3) 0
_____________________________________________________________________________
_____________________
lambda (Lambda)                 (None, 128, 128, 3)  0           input_1[0]
[0]
_____________________________________________________________________________
_____________________
conv2d (Conv2D)                 (None, 128, 128, 16) 448         lambda[0][0]
_____________________________________________________________________________
_____________________
dropout (Dropout)               (None, 128, 128, 16) 0           conv2d[0][0]
_____________________________________________________________________________
_____________________
conv2d_1 (Conv2D)               (None, 128, 128, 16) 2320        dropout[0]
[0]
_____________________________________________________________________________
_____________________
max_pooling2d (MaxPooling2D)    (None, 64, 64, 16)   0           conv2d_1[0]
[0]
_____________________________________________________________________________
_____________________
conv2d_2 (Conv2D)               (None, 64, 64, 32)   4640        max_pooling2
d[0][0]
_____________________________________________________________________________
_____________________
dropout_1 (Dropout)             (None, 64, 64, 32)   0           conv2d_2[0]
[0]
_____________________________________________________________________________
_____________________
conv2d_3 (Conv2D)               (None, 64, 64, 32)   9248        dropout_1[0]
[0]
_____________________________________________________________________________
_____________________
max_pooling2d_1 (MaxPooling2D)  (None, 32, 32, 32)   0           conv2d_3[0]
[0]
_____________________________________________________________________________
_____________________
conv2d_4 (Conv2D)               (None, 32, 32, 64)   18496       max_pooling2
d_1[0][0]
_____________________________________________________________________________
_____________________
dropout_2 (Dropout)             (None, 32, 32, 64)   0           conv2d_4[0]
[0]
_____________________________________________________________________________
_____________________
conv2d_5 (Conv2D)               (None, 32, 32, 64)   36928       dropout_2[0]
[0]
_____________________________________________________________________________
_____________________
max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 64)   0           conv2d_5[0]
[0]
```

```
________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 16, 16, 128)  73856       max_pooling2
d_2[0][0]
________________________________________________________________________________
dropout_3 (Dropout)             (None, 16, 16, 128)  0           conv2d_6[0]
[0]
________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 16, 16, 128)  147584      dropout_3[0]
[0]
________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 8, 8, 128)    0           conv2d_7[0]
[0]
________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 8, 8, 256)    295168      max_pooling2
d_3[0][0]
________________________________________________________________________________
dropout_4 (Dropout)             (None, 8, 8, 256)    0           conv2d_8[0]
[0]
________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 8, 8, 256)    590080      dropout_4[0]
[0]
________________________________________________________________________________
conv2d_transpose (Conv2DTranspo (None, 16, 16, 128)  131200      conv2d_9[0]
[0]
________________________________________________________________________________
concatenate (Concatenate)       (None, 16, 16, 256)  0           conv2d_trans
pose[0][0]
                                                                 conv2d_7[0]
[0]
________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 16, 16, 128)  295040      concatenate
[0][0]
________________________________________________________________________________
dropout_5 (Dropout)             (None, 16, 16, 128)  0           conv2d_10[0]
[0]
________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 16, 16, 128)  147584      dropout_5[0]
[0]
________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 32, 32, 64)   32832       conv2d_11[0]
[0]
________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 32, 32, 128)  0           conv2d_trans
pose_1[0][0]
                                                                 conv2d_5[0]
[0]
________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 32, 32, 64)   73792       concatenate_
1[0][0]
________________________________________________________________________________
dropout_6 (Dropout)             (None, 32, 32, 64)   0           conv2d_12[0]
[0]
________________________________________________________________________________
```

```
____________________
conv2d_13 (Conv2D)              (None, 32, 32, 64)   36928       dropout_6[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_transpose_2 (Conv2DTrans (None, 64, 64, 32)   8224        conv2d_13[0]
[0]
__________________________________________________________________________________________
____________________
concatenate_2 (Concatenate)     (None, 64, 64, 64)   0           conv2d_trans
pose_2[0][0]
                                                                 conv2d_3[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_14 (Conv2D)              (None, 64, 64, 32)   18464       concatenate_
2[0][0]
__________________________________________________________________________________________
____________________
dropout_7 (Dropout)             (None, 64, 64, 32)   0           conv2d_14[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_15 (Conv2D)              (None, 64, 64, 32)   9248        dropout_7[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_transpose_3 (Conv2DTrans (None, 128, 128, 16) 2064        conv2d_15[0]
[0]
__________________________________________________________________________________________
____________________
concatenate_3 (Concatenate)     (None, 128, 128, 32) 0           conv2d_trans
pose_3[0][0]
                                                                 conv2d_1[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_16 (Conv2D)              (None, 128, 128, 16) 4624        concatenate_
3[0][0]
__________________________________________________________________________________________
____________________
dropout_8 (Dropout)             (None, 128, 128, 16) 0           conv2d_16[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_17 (Conv2D)              (None, 128, 128, 16) 2320        dropout_8[0]
[0]
__________________________________________________________________________________________
____________________
conv2d_18 (Conv2D)              (None, 128, 128, 1)  17          conv2d_17[0]
[0]
==========================================================================================
====================
Total params: 1,941,105
Trainable params: 1,941,105
Non-trainable params: 0
__________________________________________________________________________________________
____________________
```

Training Phase

In [6]:

```
X_train1 = X_train[:X_train.shape[0]-2]
Y_train1 = Y_train[:Y_train.shape[0]-2]

# Splitting 2 images for test-set
# Not for accuracy measuring puporse, but we will use them just for visualiza
X_test = X_train[X_train.shape[0]-2:]
```

```
Y_test = Y_train[Y_train.shape[0]-2:]

X_train = X_train1
Y_train = Y_train1

# Also 10% of taining set is then taken for validation set.
results = model.fit(X_train, Y_train, validation_split=0.1, batch_size=16, ep

####################################


preds_train = model.predict(X_train[:int(X_train.shape[0]*0.9)], verbose=1)
preds_val = model.predict(X_train[int(X_train.shape[0]*0.9):], verbose=1)
# preds_test = model.predict(X_test, verbose=1)

# Every pixel value belonged b/w 0 and 1. (we normalized the dataset before t
# Threshold is taken as 0.5,
# every pixel grater than 0.5 is treated as 1 and
# every pixel lower than 0.5 is treated as 0.
preds_train_t = (preds_train > 0.5).astype(np.uint8)
preds_val_t = (preds_val > 0.5).astype(np.uint8)
# preds_test_t = (preds_test > 0.5).astype(np.uint8)
```

```
Train on 601 samples, validate on 67 samples
Epoch 1/40
592/601 [============================>.] - ETA: 0s - loss: 0.6326 - acc: 0.77
82
Epoch 00001: val_loss improved from inf to 0.51059, saving model to model_for
_nuclei.h5
601/601 [==============================] - 70s 116ms/sample - loss: 0.6297 -
acc: 0.7794 - val_loss: 0.5106 - val_acc: 0.8016
Epoch 2/40
592/601 [============================>.] - ETA: 1s - loss: 0.3485 - acc: 0.83
16
Epoch 00002: val_loss improved from 0.51059 to 0.22818, saving model to model
_for_nuclei.h5
601/601 [==============================] - 74s 123ms/sample - loss: 0.3468 -
acc: 0.8326 - val_loss: 0.2282 - val_acc: 0.8927
Epoch 3/40
592/601 [============================>.] - ETA: 1s - loss: 0.2037 - acc: 0.91
72
Epoch 00003: val_loss improved from 0.22818 to 0.15790, saving model to model
_for_nuclei.h5
601/601 [==============================] - 71s 118ms/sample - loss: 0.2030 -
acc: 0.9175 - val_loss: 0.1579 - val_acc: 0.9344
Epoch 4/40
592/601 [============================>.] - ETA: 1s - loss: 0.1493 - acc: 0.94
04
Epoch 00004: val_loss improved from 0.15790 to 0.12793, saving model to model
_for_nuclei.h5
601/601 [==============================] - 75s 125ms/sample - loss: 0.1484 -
acc: 0.9408 - val_loss: 0.1279 - val_acc: 0.9495
Epoch 5/40
592/601 [============================>.] - ETA: 1s - loss: 0.1309 - acc: 0.94
85
Epoch 00005: val_loss improved from 0.12793 to 0.11476, saving model to model
_for_nuclei.h5
601/601 [==============================] - 74s 123ms/sample - loss: 0.1332 -
acc: 0.9477 - val_loss: 0.1148 - val_acc: 0.9574
Epoch 6/40
592/601 [============================>.] - ETA: 0s - loss: 0.1192 - acc: 0.95
39
Epoch 00006: val_loss improved from 0.11476 to 0.11232, saving model to model
_for_nuclei.h5
601/601 [==============================] - 58s 96ms/sample - loss: 0.1188 - a
cc: 0.9540 - val_loss: 0.1123 - val_acc: 0.9572
Epoch 7/40
592/601 [============================>.] - ETA: 0s - loss: 0.1112 - acc: 0.95
```

```
66
Epoch 00007: val_loss improved from 0.11232 to 0.10689, saving model to model
_for_nuclei.h5
601/601 [==============================] - 63s 104ms/sample - loss: 0.1112 -
acc: 0.9566 - val_loss: 0.1069 - val_acc: 0.9598
Epoch 8/40
592/601 [============================>.] - ETA: 0s - loss: 0.1053 - acc: 0.95
92
Epoch 00008: val_loss improved from 0.10689 to 0.10624, saving model to model
_for_nuclei.h5
601/601 [==============================] - 67s 112ms/sample - loss: 0.1054 -
acc: 0.9592 - val_loss: 0.1062 - val_acc: 0.9609
Epoch 9/40
592/601 [============================>.] - ETA: 1s - loss: 0.1045 - acc: 0.95
94
Epoch 00009: val_loss improved from 0.10624 to 0.09331, saving model to model
_for_nuclei.h5
601/601 [==============================] - 74s 123ms/sample - loss: 0.1042 -
acc: 0.9595 - val_loss: 0.0933 - val_acc: 0.9654
Epoch 10/40
592/601 [============================>.] - ETA: 0s - loss: 0.0999 - acc: 0.96
15
Epoch 00010: val_loss improved from 0.09331 to 0.09112, saving model to model
_for_nuclei.h5
601/601 [==============================] - 66s 110ms/sample - loss: 0.0999 -
acc: 0.9614 - val_loss: 0.0911 - val_acc: 0.9667
Epoch 11/40
592/601 [============================>.] - ETA: 0s - loss: 0.0992 - acc: 0.96
19
Epoch 00011: val_loss did not improve from 0.09112
601/601 [==============================] - 63s 106ms/sample - loss: 0.0987 -
acc: 0.9621 - val_loss: 0.0939 - val_acc: 0.9661
Epoch 12/40
592/601 [============================>.] - ETA: 1s - loss: 0.0948 - acc: 0.96
35
Epoch 00012: val_loss improved from 0.09112 to 0.08895, saving model to model
_for_nuclei.h5
601/601 [==============================] - 71s 118ms/sample - loss: 0.0955 -
acc: 0.9632 - val_loss: 0.0889 - val_acc: 0.9674
Epoch 13/40
592/601 [============================>.] - ETA: 0s - loss: 0.0918 - acc: 0.96
42
Epoch 00013: val_loss improved from 0.08895 to 0.08864, saving model to model
_for_nuclei.h5
601/601 [==============================] - 67s 111ms/sample - loss: 0.0915 -
acc: 0.9643 - val_loss: 0.0886 - val_acc: 0.9679
Epoch 14/40
592/601 [============================>.] - ETA: 0s - loss: 0.0926 - acc: 0.96
38
Epoch 00014: val_loss did not improve from 0.08864
601/601 [==============================] - 66s 110ms/sample - loss: 0.0932 -
acc: 0.9636 - val_loss: 0.0892 - val_acc: 0.9659
Epoch 15/40
592/601 [============================>.] - ETA: 1s - loss: 0.0896 - acc: 0.96
50
Epoch 00015: val_loss improved from 0.08864 to 0.08625, saving model to model
_for_nuclei.h5
601/601 [==============================] - 88s 147ms/sample - loss: 0.0897 -
acc: 0.9650 - val_loss: 0.0862 - val_acc: 0.9685
Epoch 16/40
592/601 [============================>.] - ETA: 1s - loss: 0.0904 - acc: 0.96
47
Epoch 00016: val_loss did not improve from 0.08625
601/601 [==============================] - 75s 125ms/sample - loss: 0.0908 -
acc: 0.9645 - val_loss: 0.1005 - val_acc: 0.9605
Epoch 17/40
592/601 [============================>.] - ETA: 1s - loss: 0.0894 - acc: 0.96
50
Epoch 00017: val_loss improved from 0.08625 to 0.08512, saving model to model
```

```
_for_nuclei.h5
601/601 [==============================] - 79s 131ms/sample - loss: 0.0888 -
acc: 0.9653 - val_loss: 0.0851 - val_acc: 0.9689
Epoch 18/40
592/601 [============================>.] - ETA: 1s - loss: 0.0869 - acc: 0.96
58
Epoch 00018: val_loss improved from 0.08512 to 0.08289, saving model to model
_for_nuclei.h5
601/601 [==============================] - 76s 127ms/sample - loss: 0.0863 -
acc: 0.9660 - val_loss: 0.0829 - val_acc: 0.9692
Epoch 19/40
592/601 [============================>.] - ETA: 0s - loss: 0.0880 - acc: 0.96
56
Epoch 00019: val_loss did not improve from 0.08289
601/601 [==============================] - 51s 84ms/sample - loss: 0.0877 - a
cc: 0.9657 - val_loss: 0.0948 - val_acc: 0.9638
Epoch 20/40
592/601 [============================>.] - ETA: 0s - loss: 0.0883 - acc: 0.96
56
Epoch 00020: val_loss did not improve from 0.08289
601/601 [==============================] - 51s 84ms/sample - loss: 0.0884 - a
cc: 0.9656 - val_loss: 0.0876 - val_acc: 0.9663
601/601 [==============================] - 7s 12ms/sample
67/67 [==============================] - 1s 14ms/sample
```

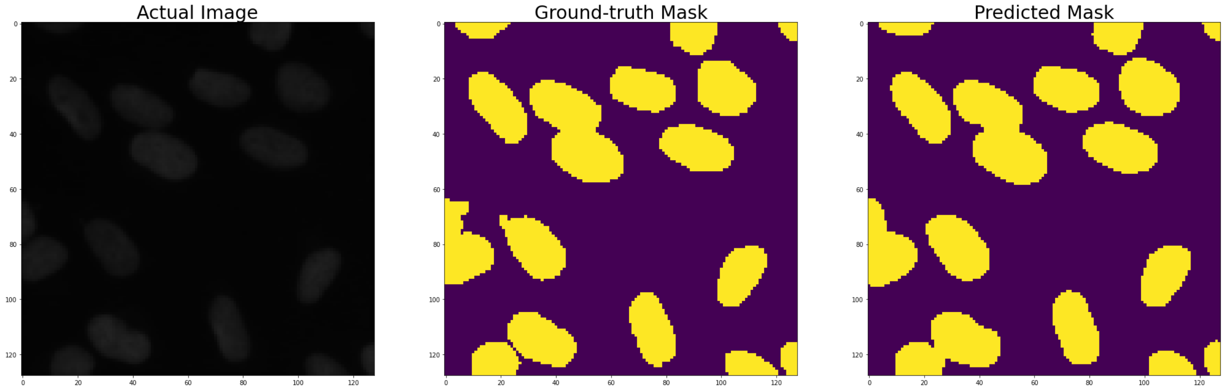
In [7]:

```python
import warnings
warnings.filterwarnings('ignore')

# Checking  some random training samples
print("Checking Out Random Image from Train Set")
ix = random.randint(0, len(preds_train_t))

fig = plt.figure(figsize=(35,35))
fig.subplots_adjust(hspace=0.2 , wspace=0.2)

# from matplotlib.pyplot import figure
# figure(num=None, figsize=(18, 12), dpi=200, facecolor='w', edgecolor='k')
ax = fig.add_subplot(1,3,1)
ax.set_title('Actual Image',fontsize=30)
ax.imshow(X_train[ix])

ax = fig.add_subplot(1,3,2)
ax.set_title('Ground-truth Mask',fontsize=30)
ax.imshow(np.squeeze(Y_train[ix]))

ax = fig.add_subplot(1,3,3)
ax.set_title('Predicted Mask',fontsize=30)
ax.imshow(np.squeeze(preds_train_t[ix]))
```

```
Checking Out Random Image from Train Set
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x7f5d700cfdd8>
```



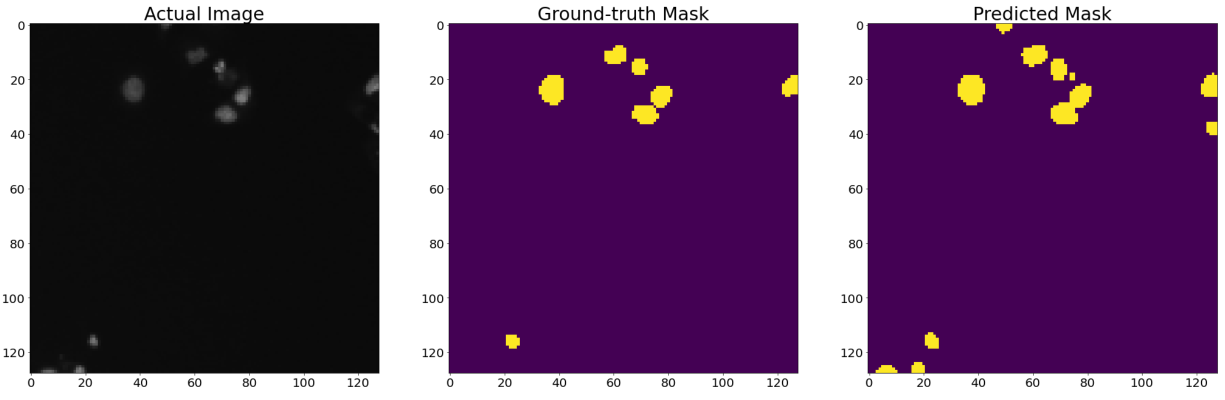
In [9]:

```
#-----------------------------------------------------------------------------
# Checking some random validation samples
import matplotlib
matplotlib.rc('xtick', labelsize=20)
matplotlib.rc('ytick', labelsize=20)

ix = random.randint(0, len(preds_val_t))
print("Checking Out Random Image from Validation Set")

fig = plt.figure(figsize=(35,35))
fig.subplots_adjust(hspace=0.2 , wspace=0.2)

# from matplotlib.pyplot import figure
# figure(num=None, figsize=(18, 12), dpi=200, facecolor='w', edgecolor='k')
ax = fig.add_subplot(1,3,1)
ax.set_title('Actual Image',fontsize=30)
ax.imshow(X_train[int(X_train.shape[0]*0.9):][ix])


ax = fig.add_subplot(1,3,2)
ax.set_title('Ground-truth Mask',fontsize=30)
ax.imshow(np.squeeze(Y_train[int(Y_train.shape[0]*0.9):][ix]))

ax = fig.add_subplot(1,3,3)
ax.set_title('Predicted Mask',fontsize=30)
ax.imshow(np.squeeze(preds_val_t[ix]))
```

```
Checking Out Random Image from Validation Set
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x7f5d606aeeb8>
```



In [10]:

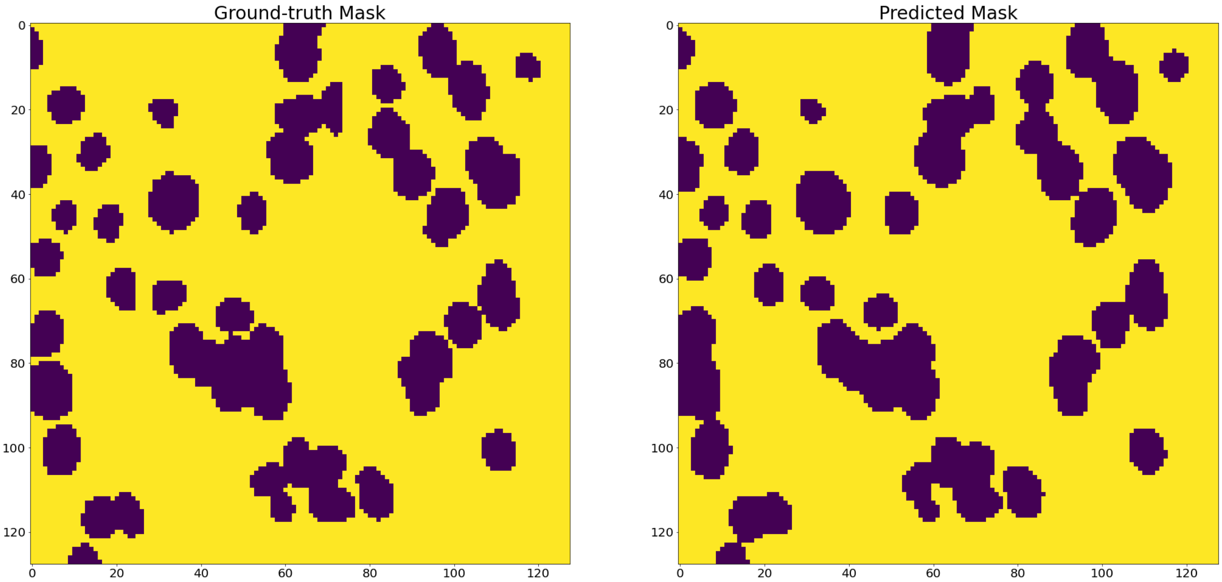
```
# Defining y_true and y_pred -------to calculate performance metric---- IOU a
y_true = Y_train[int(Y_train.shape[0]*0.9):].astype('float32')
y_pred = preds_val_t.astype('float32')

# reversing the images of validation set
# Later, we will use this complement image to calculate IOU_background and Di
y_true_black = 1-Y_train[int(Y_train.shape[0]*0.9):].astype('float32')
y_pred_black = 1- preds_val_t.astype('float32')

ix = random.randint(0, len(preds_val_t))
matplotlib.rc('xtick', labelsize=20)
matplotlib.rc('ytick', labelsize=20)

print("Example-----Reversing the pixels of Validation Set images")

fig = plt.figure(figsize=(35,35))
fig.subplots_adjust(hspace=0.2 , wspace=0.2)


ax = fig.add_subplot(1,2,1)
ax.set_title('Ground-truth Mask',fontsize=30)
```

```
ax.imshow(y_true_black[ix])

ax = fig.add_subplot(1,2,2)
ax.set_title('Predicted Mask',fontsize=30)
ax.imshow(y_pred_black[ix])
```

```
Example-----Reversing the pixels of Validation Set images
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x7f5d6058e6a0>
```



Performace Metrics ----IOU & DICE

In [11]:

```
def give_iou_dice(y_pred,y_true):

    axes = (1,2) # W,H axes of each image
    intersection = np.sum(np.logical_and(y_pred, y_true), axis=axes)
    # intersection = np.sum(np.abs(y_pred * y_true), axis=axes)
    union = np.sum(np.logical_or(y_pred, y_true), axis=axes)
    mask_sum = np.sum(np.abs(y_true), axis=axes) + np.sum(np.abs(y_pred), axi
    # union = mask_sum  - intersection

    smooth = .001 # to prevent-----( 0/0= nan  )------such cases
    iou = (intersection + smooth) / (union + smooth)
    dice = 2 * (intersection + smooth)/(mask_sum + smooth)

#     print(iou.shape)

#     print("pre: ",iou)

    iou = np.mean(iou)
    dice = np.mean(dice)

    print("iou: {}".format(iou))
    print("dice: {}".format(dice))

    return iou,dice

print("iou and dice calculated for White pixels----\n")
iouW, diceW = give_iou_dice(y_pred,y_true)
print("\n")
print("iou and dice calculated for Black pixels------\n")
iouB, diceB = give_iou_dice(y_pred_black,y_true_black)

iou_T = (iouW+iouB)/2
dice_T = (diceW+diceB)/2
```

```
print("\nThe final IOU and Dice metrics calculated for the model:")
print("iou: {}".format(iou_T),"\ndice: {}".format(dice_T))
print("\nThese iou and dice values are evaluated for validation set, as test-
```

```
iou and dice calculated for White pixels----

iou: 0.8389312916826821
dice: 0.9078408673497741


iou and dice calculated for Black pixels------

iou: 0.9540827196545468
dice: 0.9758305754095381

The final IOU and Dice metrics calculated for the model:
iou: 0.8965070056686144
dice: 0.9418357213796561

These iou and dice values are evaluated for validation set, as test-set did
n't had any given true masks
```

Calculation of Confusion Matrix ------For Validation Set Images

In [12]:

```
def confusion_matrix(preds, labels, conf_m):
#     preds = normalize(preds,0.9) # returns [0,1] tensor
    preds = preds.flatten()
    labels = labels.flatten()
    for i in range(len(preds)):
        if preds[i]==1 and labels[i]==1:
            conf_m[0][0] += 1/(len(preds)) # TP
        elif preds[i]==1 and labels[i]==0:
            conf_m[0][1] += 1/(len(preds)) # FP
        elif preds[i]==0 and labels[i]==0:
            conf_m[1][1] += 1/(len(preds)) # TN
        elif preds[i]==0 and labels[i]==1:
            conf_m[1][0] += 1/(len(preds)) # FN
    return conf_m
confm=[[0,0],[0,0]]

# y_true = np.array([[0,0,0],[0,1,0],[0,1,0]])
# y_pred = np.array([[1,1,1],[1,1,1],[1,1,1]])
print(len(y_pred.flatten()))
confm = confusion_matrix(y_pred,y_true,confm)
print("CONFUSION MATRIX-------")
confm
```

```
1097728
CONFUSION MATRIX-------
```

Out[12]:

```
[[0.18947954320131039, 0.024759321070416642],
 [0.008965791161381489, 0.7767953445623779]]
```

In [13]:

```
def get_confusion_matrix_intersection_mats(groundtruth, predicted):
    """ Returns dict of 4 boolean numpy arrays with True at TP, FP, FN, TN
    """

    confusion_matrix_arrs = {}

    groundtruth_inverse = np.logical_not(groundtruth)
    predicted_inverse = np.logical_not(predicted)

    confusion_matrix_arrs['tp'] = np.logical_and(groundtruth, predicted)
    confusion_matrix_arrs['tn'] = np.logical_and(groundtruth_inverse, predict
    confusion_matrix_arrs['fp'] = np.logical_and(groundtruth_inverse, predict
```

```
    confusion_matrix_arrs['fn'] = np.logical_and(groundtruth, predicted_inver

    return confusion_matrix_arrs
```

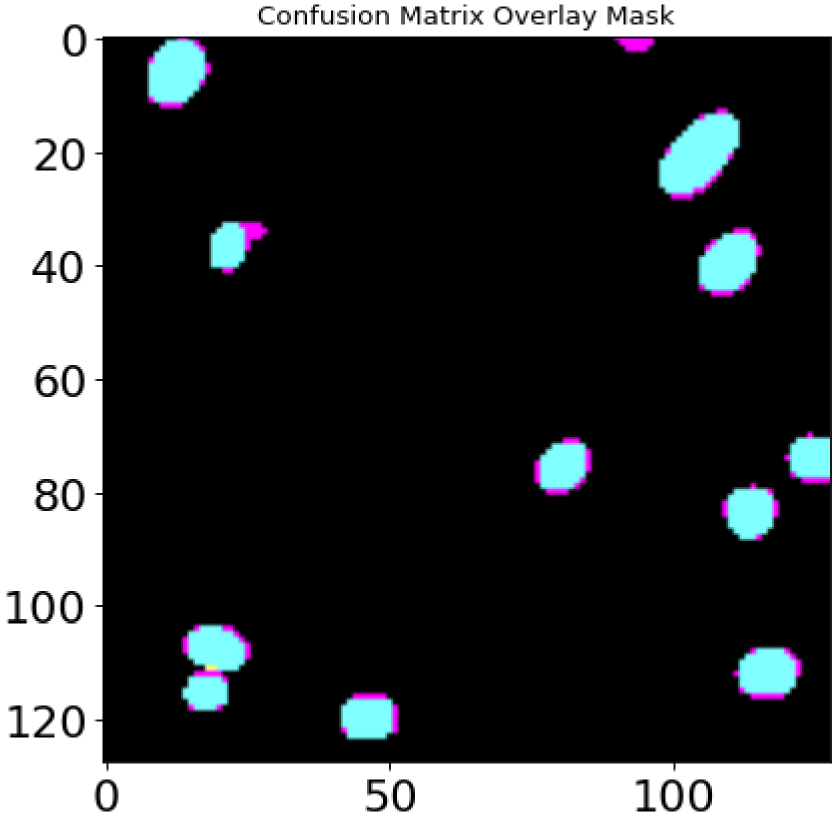
Confusion Matrix Overlay Mask

In [14]:

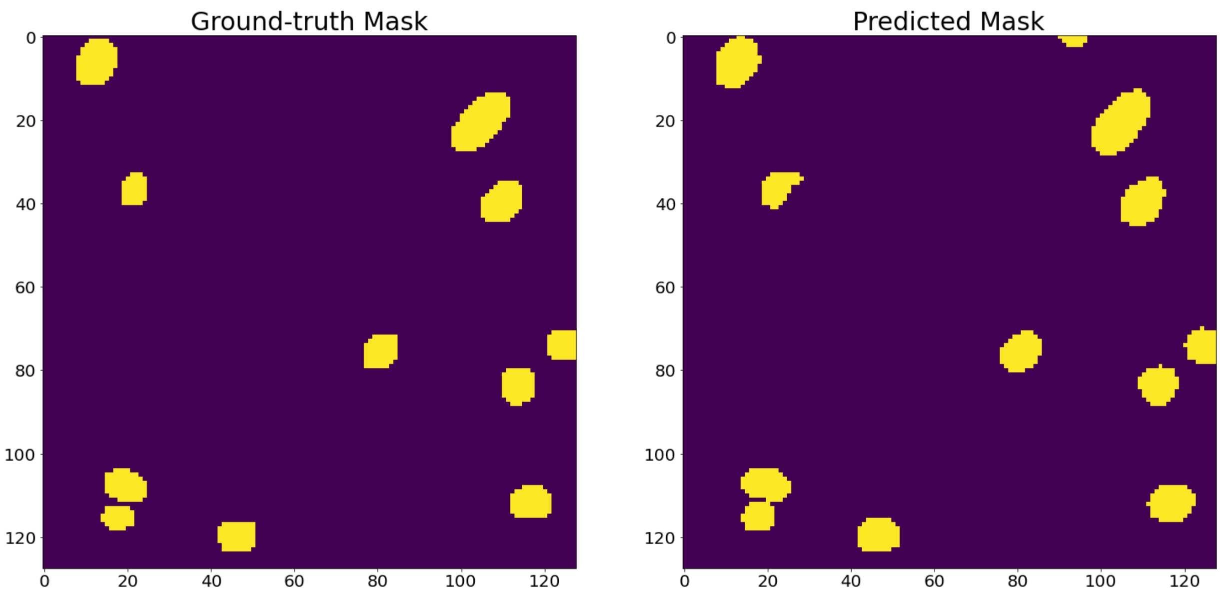
```
import cv2
def get_confusion_matrix_overlaid_mask(image, groundtruth, predicted, alpha,
    """
    Returns overlay the 'image' with a color mask where TP, FP, FN, TN are
    each a color given by the 'colors' dictionary
    """
    image = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)

    masks = get_confusion_matrix_intersection_mats(groundtruth, predicted)
    color_mask = np.zeros_like(image)
    for label, mask in masks.items():
        color = colors[label]
        mask_rgb = np.zeros_like(image)
        mask_rgb[mask != 0] = color
        color_mask += mask_rgb
    return cv2.addWeighted(image, alpha, color_mask, 1 - alpha, 0)

alpha = 0.5
confusion_matrix_colors = {
   'tp': (0, 255, 255),  #cyan
   'fp': (255, 0, 255),  #magenta
   'fn': (255, 255, 0),  #yellow
   'tn': (0, 0, 0)     #black
   }

# X_val_true = np.squeeze(X_train[int(X_train.shape[0]*0.9):].astype('float32
validation_mask = get_confusion_matrix_overlaid_mask(np.squeeze(y_true[2]), r
print('Cyan - TP')
print('Magenta - FP')
print('Yellow - FN')
print('Black - TN')

from matplotlib.pyplot import figure
figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

plt.imshow(validation_mask)
# plt.axis('off')
plt.title('Confusion Matrix Overlay Mask')
plt.show()



fig = plt.figure(figsize=(25,25))
fig.subplots_adjust(hspace=0.2 , wspace=0.2)

ax = fig.add_subplot(1,2,1)
ax.set_title('Ground-truth Mask',fontsize=30)
ax.imshow(y_true[2])

ax = fig.add_subplot(1,2,2)
ax.set_title('Predicted Mask',fontsize=30)
ax.imshow(y_pred[2])
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for f
loats or [0..255] for integers).
Cyan - TP
```

```
Magenta - FP
Yellow - FN
Black - TN
```



Out[14]: <matplotlib.image.AxesImage at 0x7f5d58104e48>



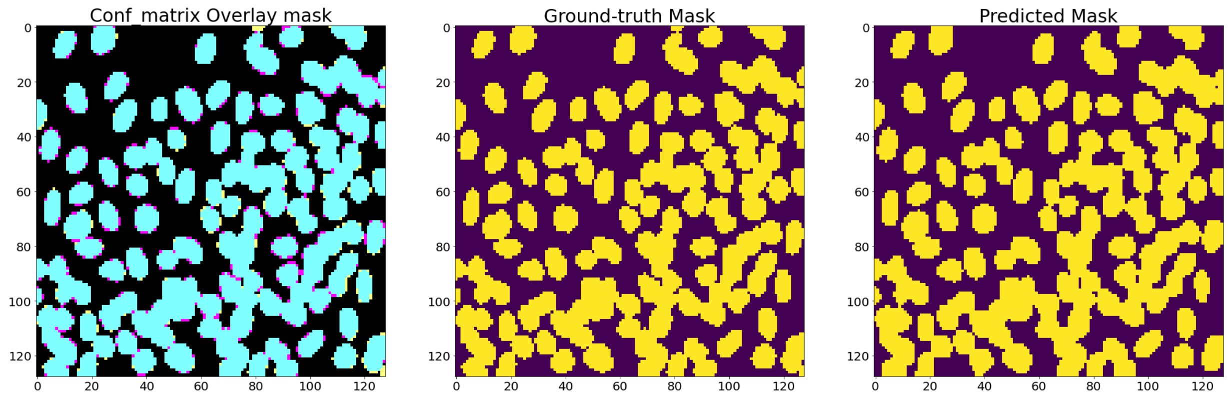
In [20]:

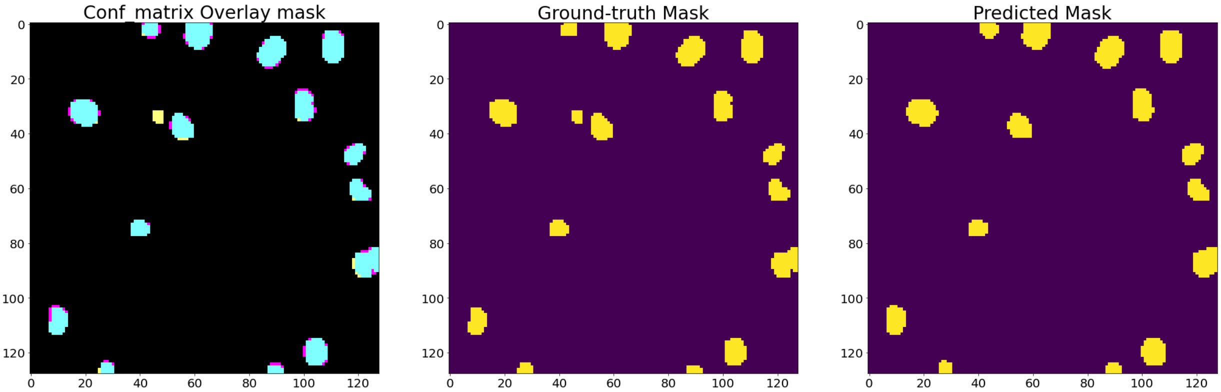
```
preds_test = model.predict(X_test, verbose=1)
preds_test_t = (preds_test > 0.5).astype(np.uint8)

y_pred_test = preds_test_t.astype('float32')
y_true_test = Y_test.astype('float32')

test1_confusion_mask = get_confusion_matrix_overlaid_mask(np.squeeze(y_true_t
test2_confusion_mask = get_confusion_matrix_overlaid_mask(np.squeeze(y_true_t


#---------------------------------------------------------------------------
# Plotting Confusion-matrix overlay mask for 2 test images
print("Plotting Confusion-matrix overlay mask for 2 test images")
```

```
print('Cyan - TP')
print('Magenta - FP')
print('Yellow - FN')
print('Black - TN')
matplotlib.rc('xtick', labelsize=20)
matplotlib.rc('ytick', labelsize=20)

fig = plt.figure(figsize=(35,35))
fig.subplots_adjust(hspace=0.2 , wspace=0.2)

ax = fig.add_subplot(1,3,1)
ax.set_title('Conf_matrix Overlay mask',fontsize=30)
ax.imshow(test1_confusion_mask)

ax = fig.add_subplot(1,3,2)
ax.set_title('Ground-truth Mask',fontsize=30)
ax.imshow(Y_test[0])

ax = fig.add_subplot(1,3,3)
ax.set_title('Predicted Mask',fontsize=30)
ax.imshow(y_pred_test[0])

#---------------------------------

fig = plt.figure(figsize=(35,35))
fig.subplots_adjust(hspace=0.2 , wspace=0.2)

ax = fig.add_subplot(1,3,1)
ax.set_title('Conf_matrix Overlay mask',fontsize=30)
ax.imshow(test2_confusion_mask)

ax = fig.add_subplot(1,3,2)
ax.set_title('Ground-truth Mask',fontsize=30)
ax.imshow(Y_test[1])

ax = fig.add_subplot(1,3,3)
ax.set_title('Predicted Mask',fontsize=30)
ax.imshow(y_pred_test[1])
```

```
2/2 [==============================] - 0s 27ms/sample
Clipping input data to the valid range for imshow with RGB data ([0..1] for f
loats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for f
loats or [0..255] for integers).
Plotting Confusion-matrix overlay mask for 2 test images
Cyan - TP
Magenta - FP
Yellow - FN
Black - TN
```

Out[20]: `<matplotlib.image.AxesImage at 0x7f5ce80bdcf8>`





Conclusion

In [25]:

```
print("The performance (measured using IOU and DICE) in validation set-----")
print(" 1. IOU: {}".format(iou_T))
print(" 2. DICE: {}".format(dice_T))
print("which is in coherence with the performace claimed result in the publis
```

```
The performance (measured using IOU and DICE) in validation set-----
 1. IOU: 0.8965070056686144
 2. DICE: 0.9418357213796561
which is in coherence with the performace claimed result in the published pap
er, i.e., 0.92 IOU.
```

In [ ]: