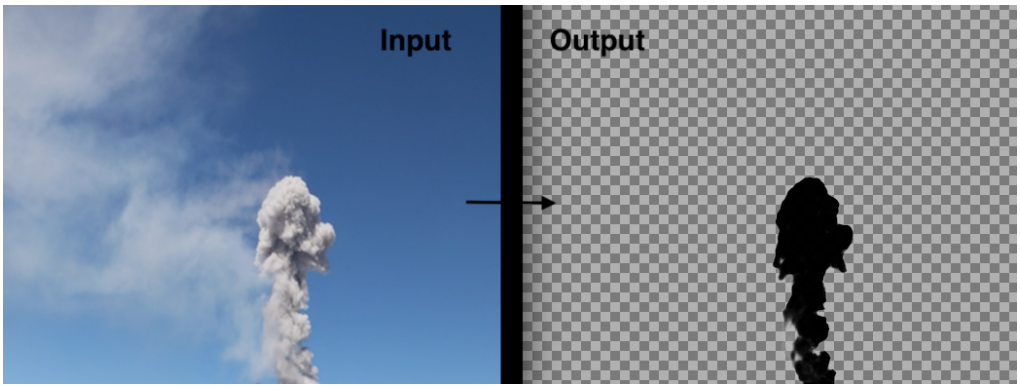
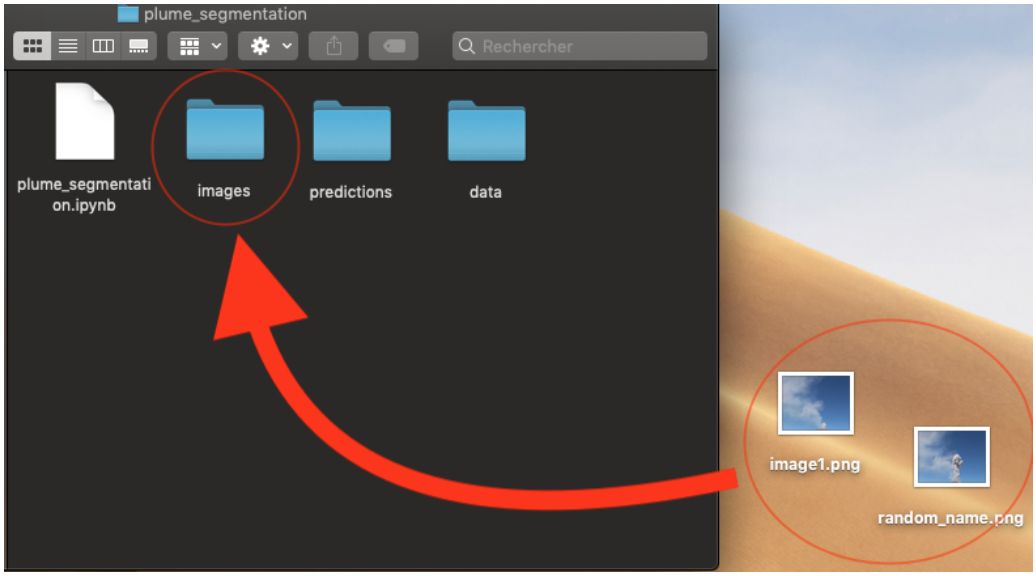
# Plume Segmentation

C.Mergny

This deep learning alogrithm creates masks from RGB images. It was trained on volcanic eruption images to detect pixels that are parts of the volcanic plume.




- To use it simply drag and drop your plumes images to the folder 'images'.
- Then run the notebook in its original folder ( alt + enter to run cell by cell ).
- You'll find the created masks in the folder 'predictions'
- Don't modify the 'data' folder it used to load the neural network weights




## Install and Import Fastai module

The fastai library is required to use this notebook. You can install it by uncommenting and running the following line.

Note: if you are not using **Anaconda-Navigator to run Jupiter Notebook**, there are other commands to install fastai ( see https://docs.fast.ai/install.html (https://docs.fast.ai/install.html) )

In [67]:
```
! conda install -y fastai
```

```
Collecting package metadata: done
Solving environment: \
The environment is inconsistent, please check the package plan carefully
The following packages are causing the inconsistency:

  - defaults/osx-64::numba==0.36.2=np114py36hc2f221f_0
  - defaults/osx-64::blaze==0.11.3=py36h02e7a37_0
  - defaults/osx-64::anaconda==5.1.0=py36_2
done

## Package Plan ##

  environment location: /Applications/anaconda3

  added / updated specs:
    - fastai


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2019.5.15  |                0         133 KB
    openssl-1.1.1c             |       h1de35cc_1         3.4 MB
    ------------------------------------------------------------
                                           Total:         3.6 MB

The following packages will be UPDATED:

  ca-certificates                                2019.1.23-0 --> 2019.5.15-0
  openssl                                    1.1.1b-h1de35cc_1 --> 1.1.1c-h1de35c
c_1



Downloading and Extracting Packages
ca-certificates-2019 | 133 KB    | ##################################### |
100%
openssl-1.1.1c       | 3.4 MB    | ##################################### |
100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Import fastai module:

In [60]:
```
from fastai.vision import *
from fastai import*
%reload_ext autoreload
%autoreload 2
```

## Load Learner

An Image Segmentation Learner has already been trained. Here we are just importing it.

In [68]:
```
# Data used to load the learner. Please don't modify the 'data' folder.
data = (SegmentationItemList.from_folder('data/input')
            .split_none()
            .label_from_func(lambda x: 'data/labels/'+ x.name, classes = ['othe
r','plume'])
            .transform(get_transforms(max_rotate=0, do_flip = False), size =
256, tfm_y = True)
            .databunch(bs =1)
            .normalize(imagenet_stats))

learn = unet_learner(data, models.resnet34, wd =1e-2)
learn.load('sgm_learner_RGB');
#Instead of RGB you could also load the learner trained for thermal data, also
I'm not sure this one works.
#learn.load('sgm_learner_thermal');
```

# Predicting

Create a folder called 'predictions' where you'll find created masks from your input images

- By default the learner is trained to output 256x256 images.
- So you may want to uncomment the .transform line and comment the current one.
- If not for bigger images it will take more time and may not be as accurate, although I haven't seen any major differences.
- To process many images (frames from a movie for example) I strongly urge you to specify size = 256.

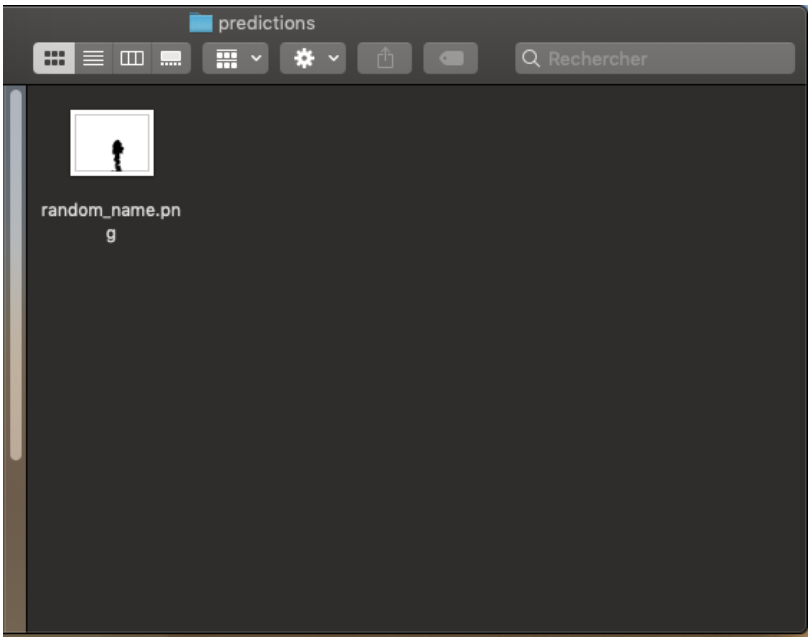
In [63]:
```
#Data used to create masks.
data_test = (SegmentationItemList.from_folder('images',ignore_empty=True)
            .split_none()
            .label_empty()
            #.transform(get_transforms(max_rotate=0, do_flip = False))
            .transform(get_transforms(max_rotate=0, do_flip = False), size =
256)
            .databunch(bs =1)
            .normalize(imagenet_stats))
learn.data = data_test
```

In [64]:
```
# Create a directory called predictions for outputs
path_pred = 'predictions/'
#!rm -r $path_pred # Uncomment to remove the previous files in predictions
!mkdir $path_pred
```

```
mkdir: predictions/: File exists
```

In [65]:
```
# Create masks from images in the 'images' folder
def save_preds(dl):
    i=0
    names = dl.dataset.items
    for batch in dl:
        preds = learn.pred_batch(batch=batch, reconstruct=True)
        for o in preds:
            o.save(path_pred+names[i].name)
            i += 1
save_preds(data_test.fix_dl);
```

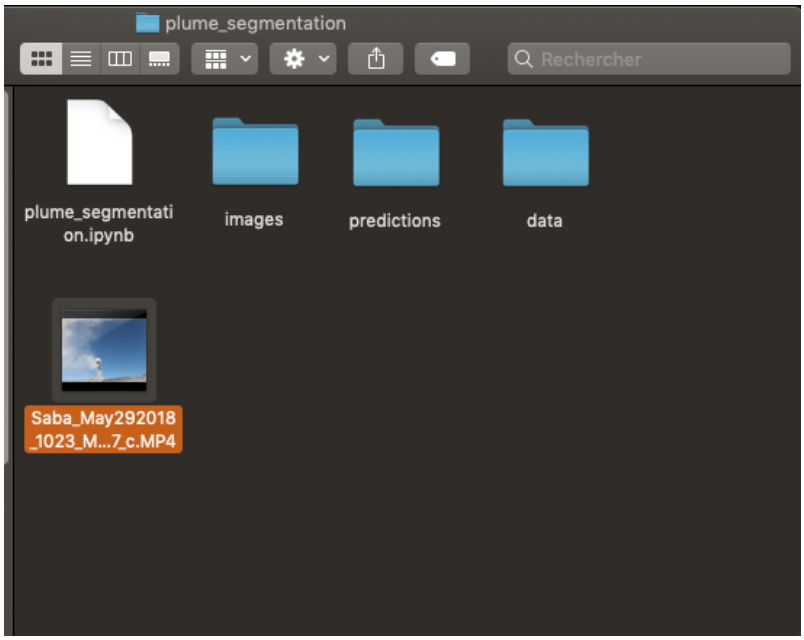
Once that last cell is finished, you should find your mask(s) in the 'predictions' folder.




# Extra: Extract frames of a movie

Here are a few bash lines that extracts frames from a movie. Images can then be used to create masks. Run it in this notebook (thanks to the '!' prefix) or on a terminal if you prefer (without the '!').

First you'll need to add a movie to the 'plume_segmentation' folder:




If you don't have **brew** installed on your terminal uncomment this line. (It's a missing package manager for mac)

In [ ]:
```
#! /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/in
stall/master/install)"
```

If you don't already have ffmpeg install on your terminal run the following line:

In [ ]:
```
! brew install ffmpeg
```

In the following cell you can:

- Enter the name of the video.
- Change the name of the output frames.
- Choose the quality of the images output.
- Choose the number of frames per second to extract.

You can create high quality images with high fps for your personal purposes. However be aware that for plume segmentation HD images take a lot of time to process.

In [54]:
```
video_name = 'Saba_May292018_1023_MVI_5587_c.MP4'
output_name = 'images/frame_%4d.png' # the %4d means that each frame is numbere
d with 4 digits e.g 'frame_0001.png'
quality = '480x360'
fps = 1/10
```

Run the next line to create frames in the 'images' folder.

In [58]:
```
! ffmpeg -i $video_name -vf fps=$fps -s $quality $output_name
```

```
ffmpeg version 4.1.3 Copyright (c) 2000-2019 the FFmpeg developers
  built with Apple LLVM version 10.0.0 (clang-1000.11.45.5)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/4.1.3_1 --enable-shared --e
nable-pthreads --enable-version3 --enable-hardcoded-tables --enable-avresample
--cc=clang --host-cflags='-I/Library/Java/JavaVirtualMachines/adoptopenjdk
-11.0.2.jdk/Contents/Home/include -I/Library/Java/JavaVirtualMachines/adoptope
njdk-11.0.2.jdk/Contents/Home/include/darwin' --host-ldflags= --enable-ffplay
--enable-gnutls --enable-gpl --enable-libaom --enable-libbluray --enable-libmp
3lame --enable-libopus --enable-librubberband --enable-libsnappy --enable-libt
esseract --enable-libtheora --enable-libvorbis --enable-libvpx --enable-libx
264 --enable-libx265 --enable-libxvid --enable-lzma --enable-libfontconfig --e
nable-libfreetype --enable-frei0r --enable-libass --enable-libopencore-amrnb
--enable-libopencore-amrwb --enable-libopenjpeg --enable-librtmp --enable-libs
peex --enable-videotoolbox --disable-libjack --disable-indev=jack --enable-lib
aom --enable-libsoxr
  libavutil      56. 22.100 / 56. 22.100
  libavcodec     58. 35.100 / 58. 35.100
  libavformat    58. 20.100 / 58. 20.100
  libavdevice    58.  5.100 / 58.  5.100
  libavfilter     7. 40.101 /  7. 40.101
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  3.100 /  5.  3.100
  libswresample   3.  3.100 /  3.  3.100
  libpostproc    55.  3.100 / 55.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'Saba_May292018_1023_MVI_5587_c.MP4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.20.100
  Duration: 00:10:50.00, start: 0.000000, bitrate: 46 kb/s
    Stream #0:0(eng): Video: h264 (High) (avc1 / 0x31637661), yuvj420p(pc),
480x360, 46 kb/s, 1 fps, 1 tbr, 16384 tbn, 2 tbc (default)
    Metadata:
      handler_name    : VideoHandler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> png (native))
Press [q] to stop, [?] for help
[swscaler @ 0x7fc36d11a600] deprecated pixel format used, make sure you did se
t range correctly
Output #0, image2, to 'images/frame_%4d.png':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2avc1mp41
    encoder         : Lavf58.20.100
    Stream #0:0(eng): Video: png, rgb24, 480x360, q=2-31, 200 kb/s, 0.04 fps,
0.04 tbn, 0.04 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc58.35.100 png
frame=   27 fps=0.0 q=-0.0 Lsize=N/A time=00:10:48.00 bitrate=N/A speed= 749x
video:5262kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxin
g overhead: unknown
```

NB: always be careful when running a 'rm' command, it could erase all the data in your computer.

If you want to clean the the 'images' folder, uncomment and run the following line:

In [57]:
```
#!rm -r 'images'
#!mkdir 'images'
```

# END