training_20191003

October 3, 2019

[1]:

```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

[2]:

```
import torch
```

[3]:

```
torch.cuda.is_available()
```

[3]:

```
True
```

[4]:

```
import os
from tqdm import tqdm, tnrange, tqdm_notebook
from pathlib import Path
import re
import numpy as np
import matplotlib.pyplot as plt
#import cv2
import sys
import scipy.ndimage
# from mpl_toolkits.mplot3d.art3d import Poly3Dcollection
```

[5]:

```
#import pydicom
#from pydicom.data import get_testdata_files
#from pydicom.filereader import read_dicomdir
#import pydicom.pixel_data_handlers.gdcm_handler as gdcm_handler
# ! gdcm must be installed with conda install (conda install -c conda-forge␣
 ↪gdcm)
# pydicom.config.image_handlers = ['gdcm_handler']
```

[6]:

```
# import nibabel as nib
```

[7]:

```
from fastai.vision import *
from fastai.metrics import error_rate
from fastai.callbacks import *
```

[8]:

```
#from fastai2.data.all import *
#from fastai2.vision.core import *
```

[9]:
```
import pandas as pd
```

0.1 Define paths

[10]:
```
path_str = '/home/ubuntu/sfr-challenge/lungs/dataset'
```

[11]:
```
path = Path(path_str)
```

[12]:
```
path_p = path/'Pathologiques'
```

[13]:
```
path_n = path/'Normaux'
```

[14]:
```
path_train = path_str + '/train'
```

[15]:
```
test_path = path_str + '/Pathologiques/N7Q0jai/N7Q0jai'
```

0.2 Define fixed random seed

[16]:
```
np.random.seed(42)
```

0.3 Test section ==>

[17]:
```
# cell to run the experiments
```

0.4 <== End of test section

0.5 Train network

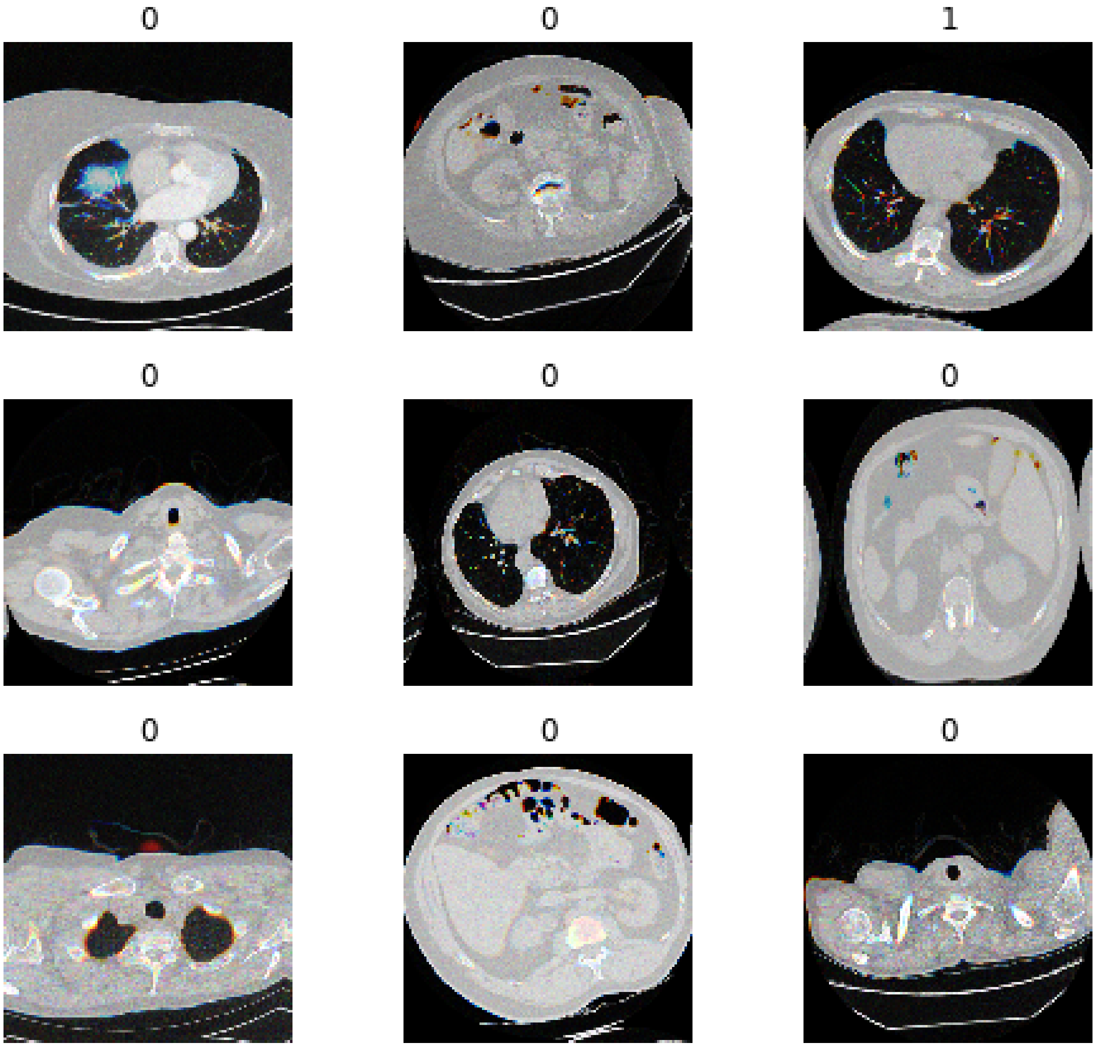
[18]:
```
bs = 32
valid_split = 0.15
```

[19]:
```
data = ImageDataBunch.from_folder(path/'train', ds_tfms=get_transforms(),␣
 ↪size=224, bs=bs, valid_pct=valid_split)
```

[20]:
```
pd.value_counts(data.train_dl.y.items.flatten(), sort=False)
```

[20]:
```
0    642
1    158
dtype: int64
```

[21]:
```
data.show_batch(rows=3, figsize=(7,6))
```



[61]:

```
class fbeta_binary_me(Callback):
    "Computes the f_beta between preds and targets for binary text␣
 ↪classification"

    def __init__(self, beta2 = 1, eps=1e-9,sigmoid = True):
        self.beta2=beta2**2
        self.eps = eps
        self.sigmoid = sigmoid

    def on_epoch_begin(self, **kwargs):
        self.TP = 0
        self.total_y_pred = 0
        self.total_y_true = 0
    def on_batch_end(self, last_output, last_target, **kwargs):
        y_pred = last_output
        y_pred = y_pred.softmax(dim = 1)
        y_pred = y_pred.argmax(dim=1)
        y_true = last_target.float()

        self.TP += ((y_pred==1) * (y_true==1)).float().sum()
        self.total_y_pred += (y_pred==1).float().sum()
        self.total_y_true += (y_true==1).float().sum()

    def on_epoch_end(self, **kwargs):
        prec = self.TP/(self.total_y_pred+self.eps)
        rec = self.TP/(self.total_y_true+self.eps)
        res = (prec*rec)/(prec*self.beta2+rec+self.eps)*(1+self.beta2)
        #self.metric = res.mean()
        self.metric = res
```

[62]:
```
fbeta_binary_me = fbeta_binary_me(1)
```

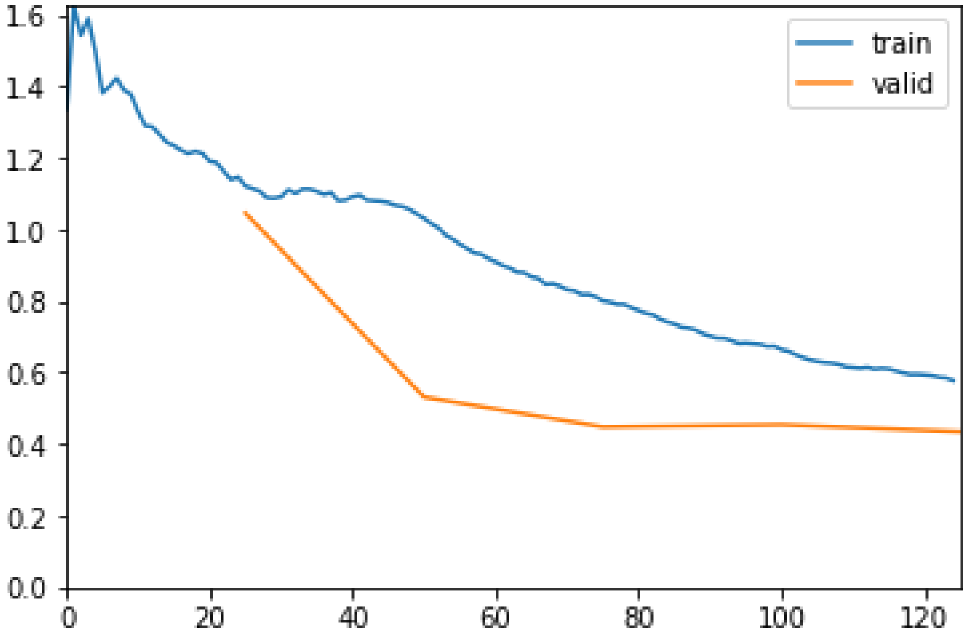
[65]:
```
learner = cnn_learner(data, models.resnet34, metrics=[error_rate, AUROC()],␣
 ↪callback_fns=[ShowGraph], pretrained=True).to_fp16()
```

[67]:
```
learner_vgg = cnn_learner(data, models.vgg16_bn, metrics=[error_rate, AUROC()],␣
 ↪callback_fns=[ShowGraph], pretrained=True).to_fp16()
```

[69]:
```
learner_vgg19 = cnn_learner(data, models.vgg19_bn, metrics=[error_rate,␣
 ↪AUROC()], callback_fns=[ShowGraph], pretrained=True).to_fp16()
```

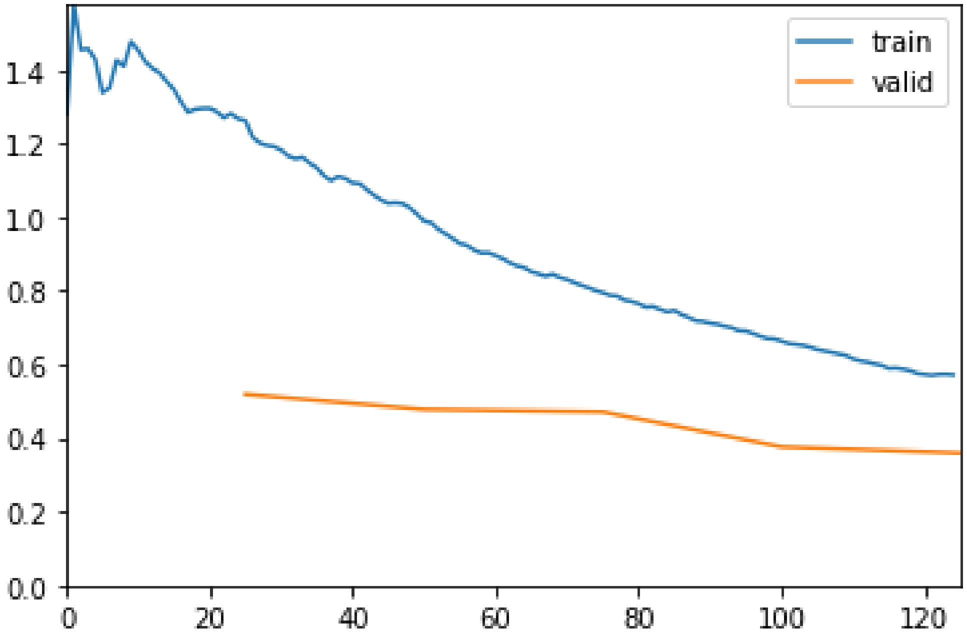
[66]:
```
learner.fit_one_cycle(5)
```

```
<IPython.core.display.HTML object>
```



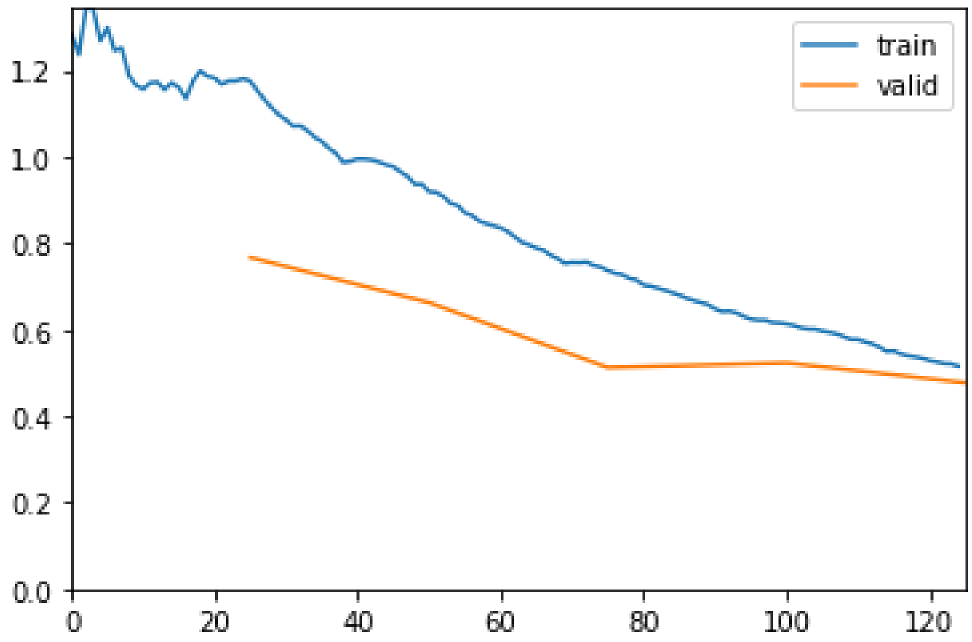
[68]:
```
learner_vgg.fit_one_cycle(5)
```

```
<IPython.core.display.HTML object>
```



```
[71]: learner_vgg19.fit_one_cycle(5)
```

```
<IPython.core.display.HTML object>
```


train
valid