In [1]:

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

In [2]:

```
import torch
```

In [3]:

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

Out[3]:

```
True
```

In [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
```

In [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']
```

In [6]:

```
# import nibabel as nib
```

In [7]:

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

In [8]:

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

In [9]:

```
import pandas as pd
```

Define paths

In [10]:

```
path_str = '/home/ubuntu/sfr-challenge/lungs/dataset'
#path_str = '/Users/igorgarbuz/SoftDev/sfr-challenge/dataset'
```

In [11]:

```
path = Path(path_str)
```

In [12]:

```
path_seg = path/'seg_3d'
```

In [13]:

```
path_p = path/'Pathologiques'
```

In [14]:

```
path_n = path/'Normaux'
```

In [15]:

```
path_train = path_str + '/train'
```

In [16]:

```
test_path = path_str + '/Pathologiques/N7Q0jai/N7Q0jai'
```

Define fixed random seed

In [17]:

```
np.random.seed(42)
```

Test section ==>

In [ ]:

```
# cell to run the experiments
```

<== End of test section

Train network

In [18]:

```
bs = 32
valid_split = 0.2
```

In [19]:

```
data = ImageDataBunch.from_folder(path/'train', ds_tfms=get_transforms(), size=224, bs=bs, valid_pc
```

In [20]:

```
pd.value_counts(data.train_dl.y.items.flatten(), sort=False)
```

Out[20]:

```
0    1340
1     692
dtype: int64
```

In [23]:

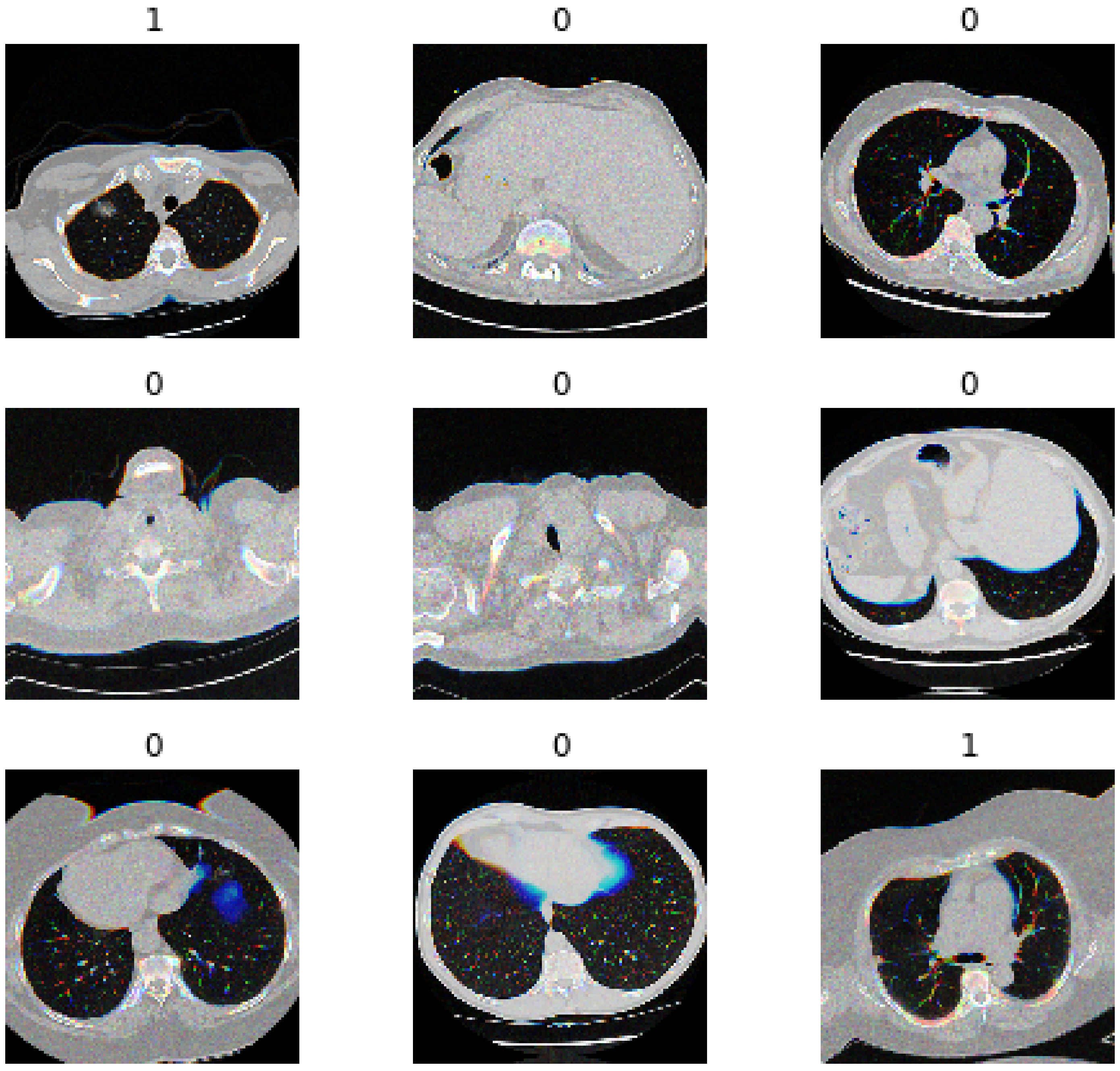
```
data.classes
```

Out[23]:

```
['0', '1']
```

In [21]:

```
data.show_batch(rows=3, figsize=(7,6))
```



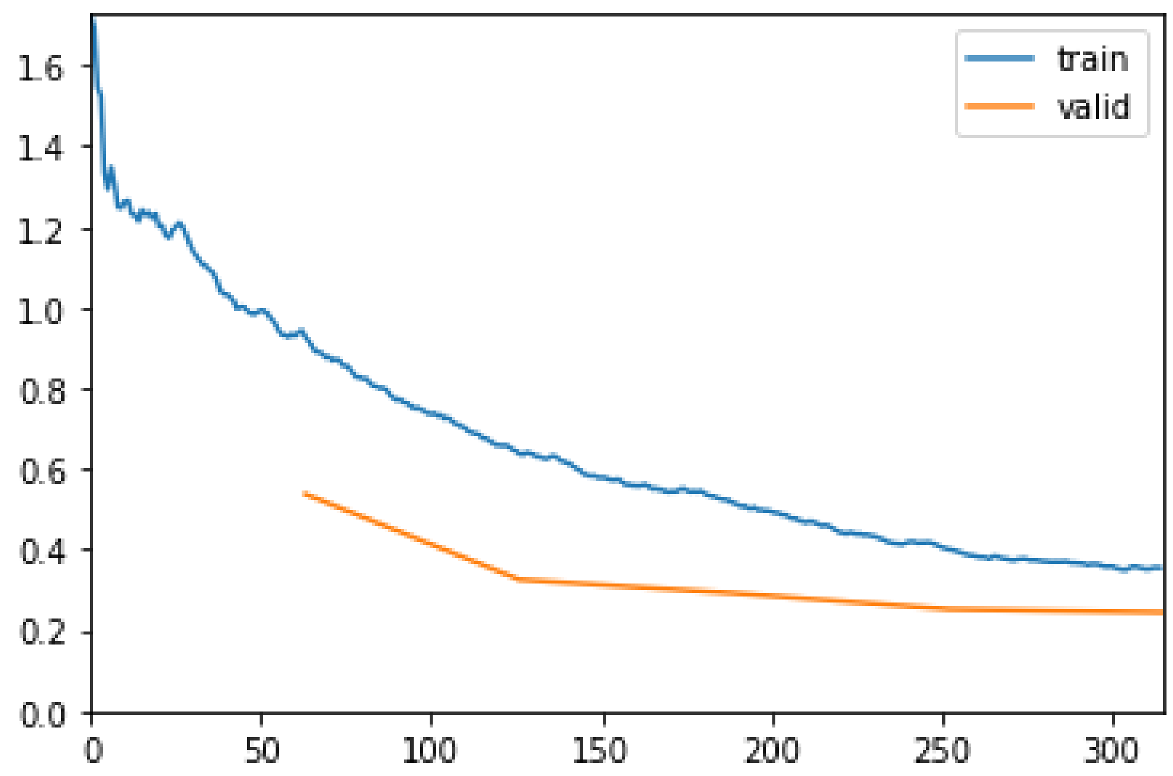
In [24]:

```
learner = cnn_learner(data, models.vgg16_bn, metrics=[error_rate, AUROC()], callback_fns=[ShowGraph
#learner = cnn_learner(data, models.resnet18, metrics=[error_rate, f1_score(), AUROC()], callback_f
```

In [25]:

```
learner.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.942929	0.537809	0.211045	0.877893	00:22
1	0.644295	0.325365	0.142012	0.925125	00:20
2	0.518053	0.291829	0.114398	0.942676	00:20
3	0.403076	0.252297	0.100592	0.960460	00:21
4	0.354266	0.244355	0.088757	0.958702	00:19



In [26]:

```
learner.save('stage-1')
```

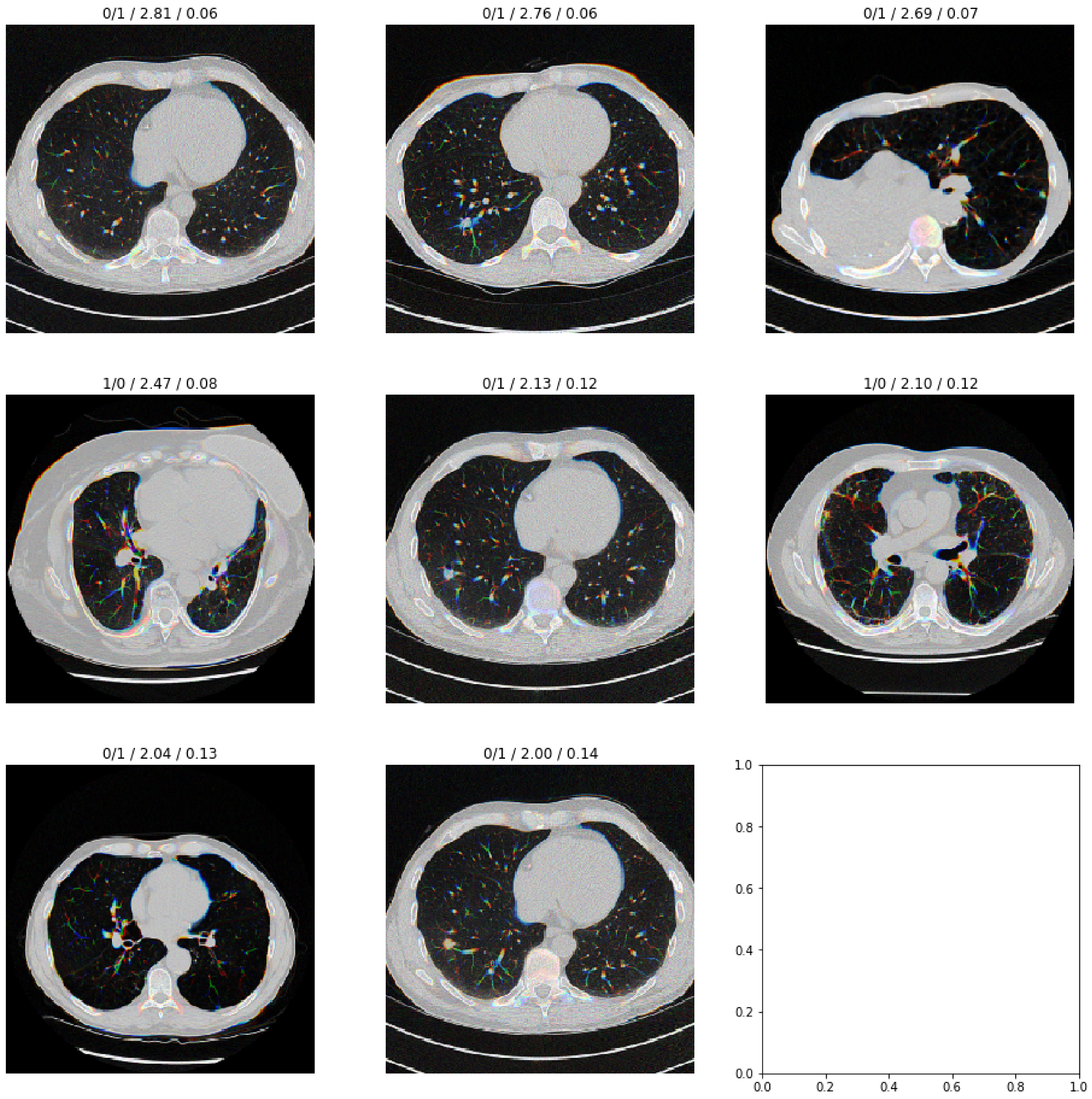
In [27]:

```
interp = ClassificationInterpretation.from_learner(learner)
losses,idxs = interp.top_losses()
len(data.valid_ds)==len(losses)==len(idxs)
```

Out[27]:

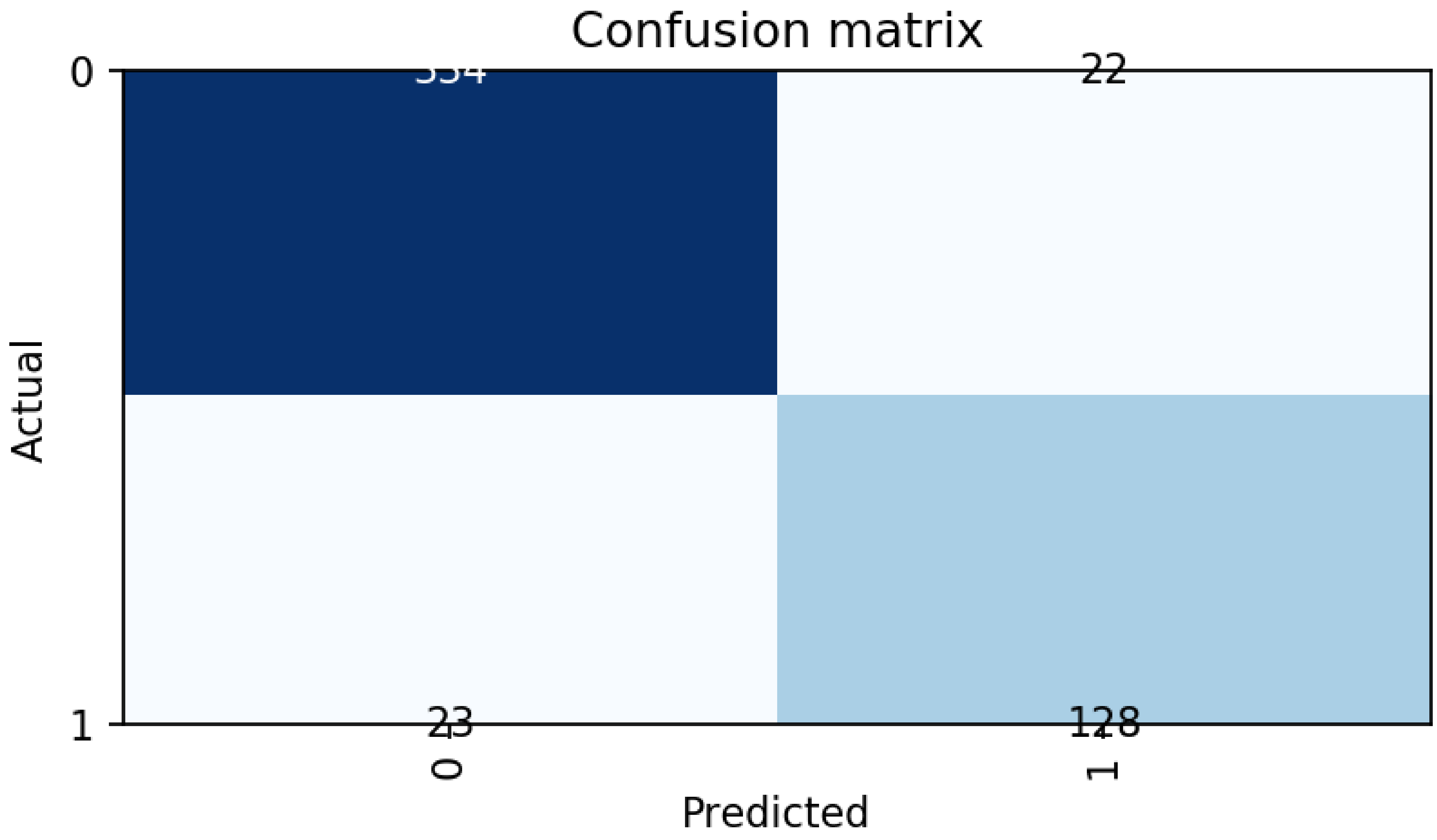
```
True
```

```
In [34]: interp.plot_top_losses(8, figsize=(16,16))
```



prediction/actual/loss/probability

```
In [42]: interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```

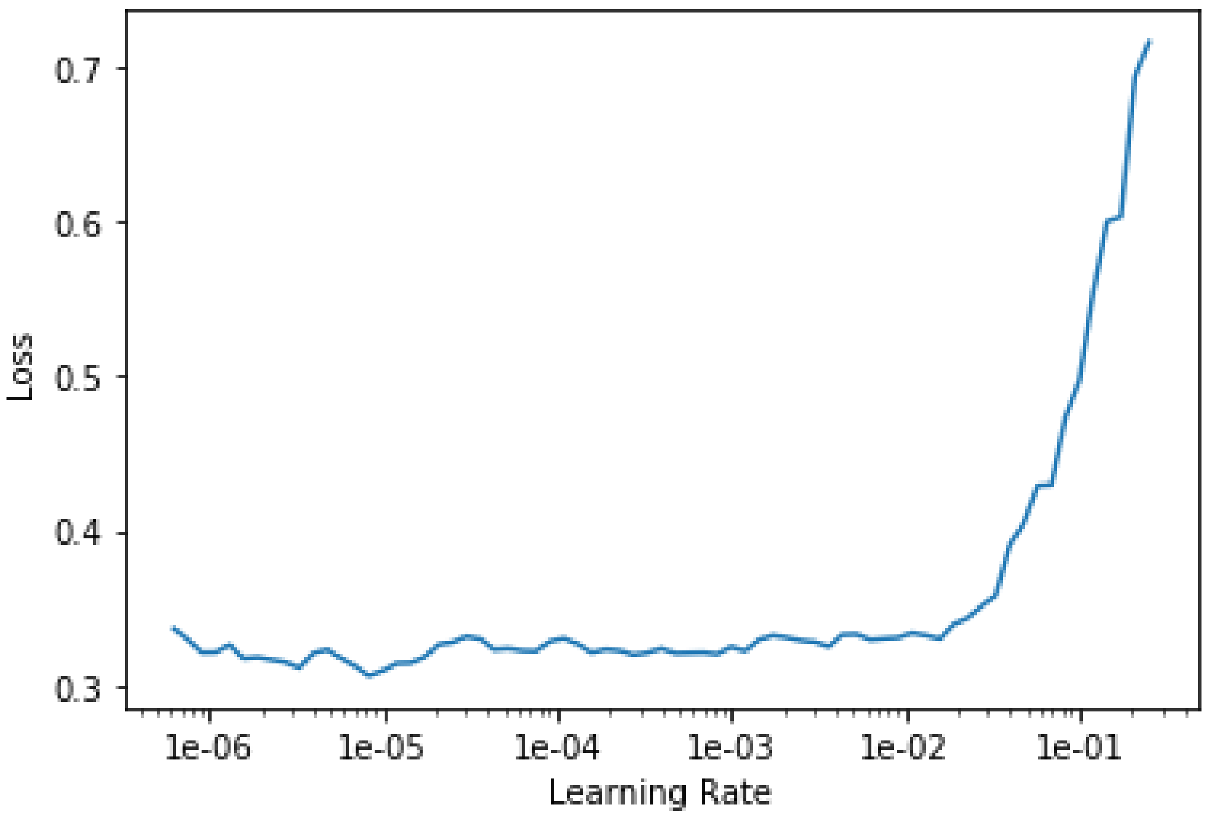


```
In [43]: interp.most_confused(min_val=2)
Out[43]: [('1', '0', 23), ('0', '1', 22)]
```

Unfreeze all model layer and tune learning rate

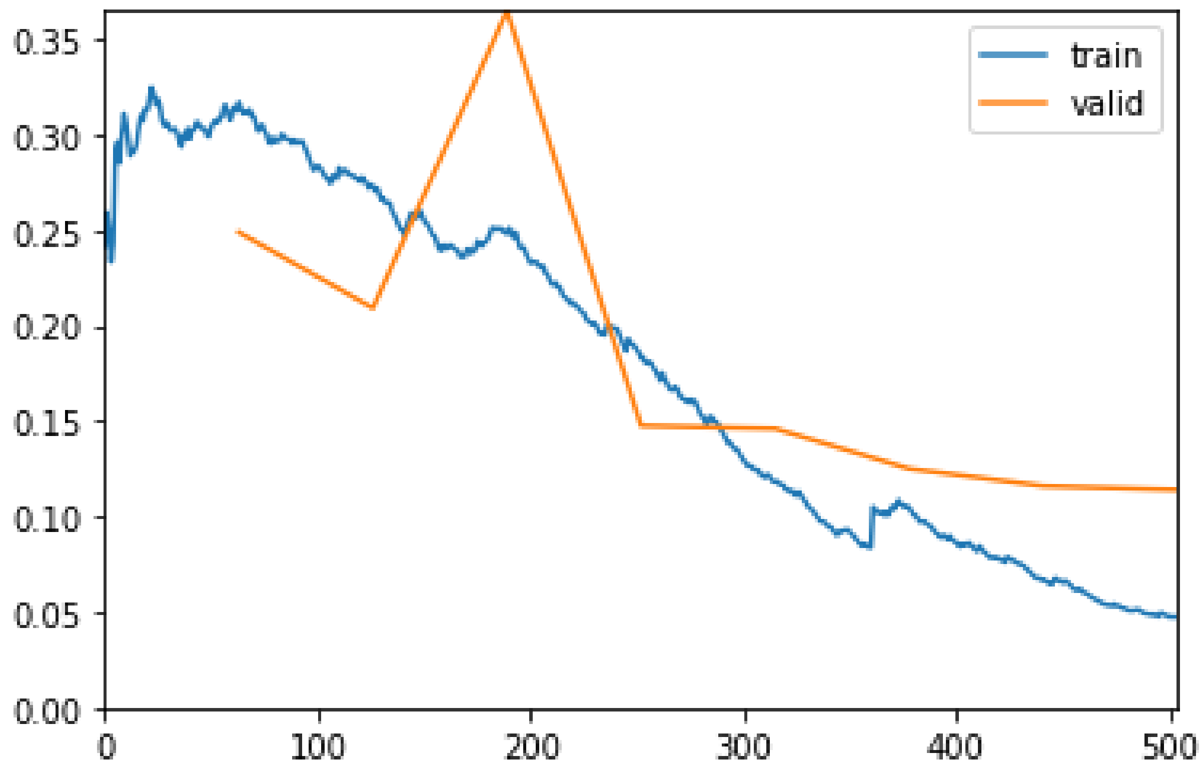
```
In [48]: learner.lr_find()

         LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [49]:
```
learner.recorder.plot()
```



In [50]:
```
learner.unfreeze()
learner.fit_one_cycle(8, max_lr=slice(1e-5, 1e-3))
```

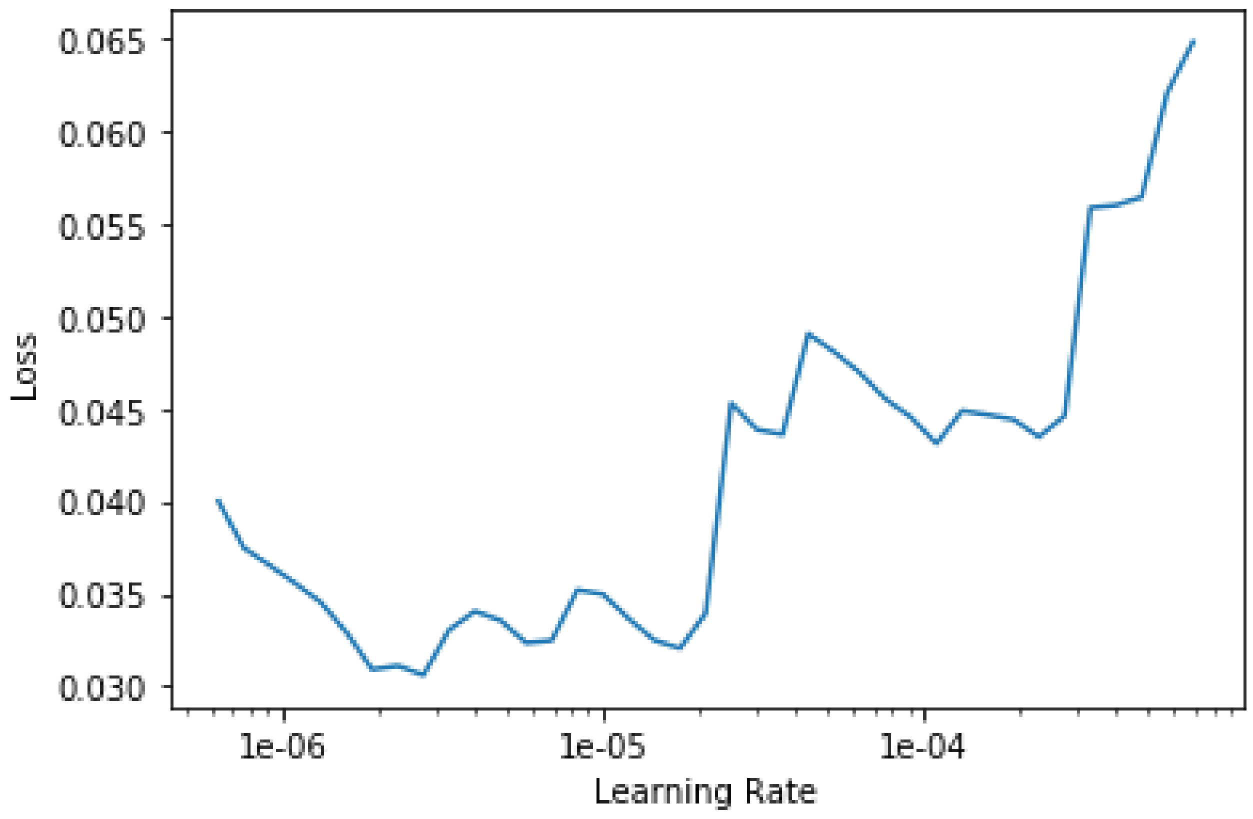
epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.312120	0.248750	0.092702	0.953410	00:21
1	0.273778	0.209047	0.074951	0.966115	00:21
2	0.249115	0.364573	0.118343	0.941467	00:21
3	0.185665	0.147487	0.057199	0.985285	00:21
4	0.118345	0.145971	0.043393	0.983667	00:21
5	0.105349	0.124930	0.035503	0.984876	00:21
6	0.066887	0.115927	0.031558	0.987257	00:21
7	0.047418	0.113995	0.031558	0.987332	00:21



In [51]:
```
learner.save('stage-2-8epc')
```

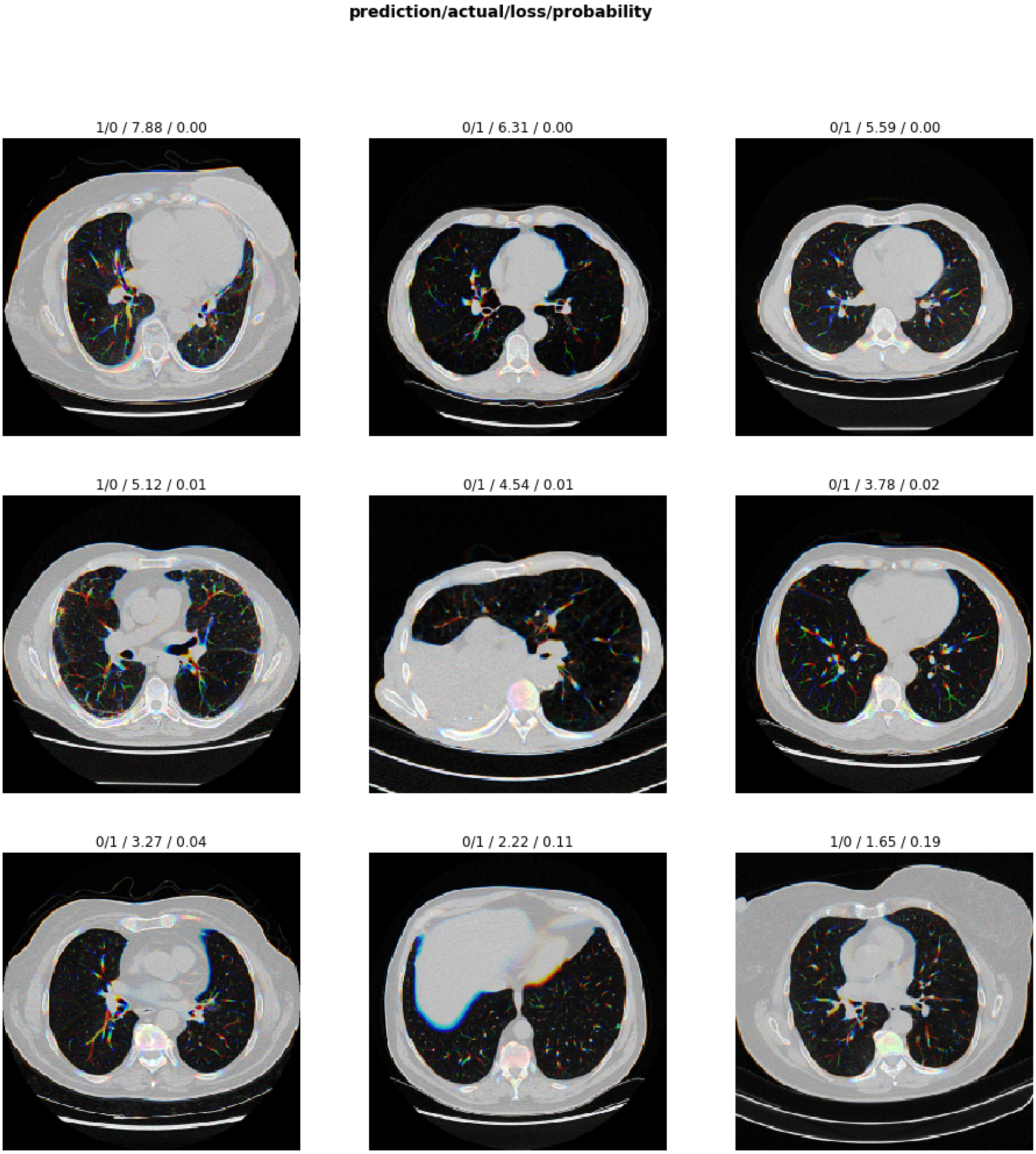
In [52]:
```
learner.lr_find()
learner.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

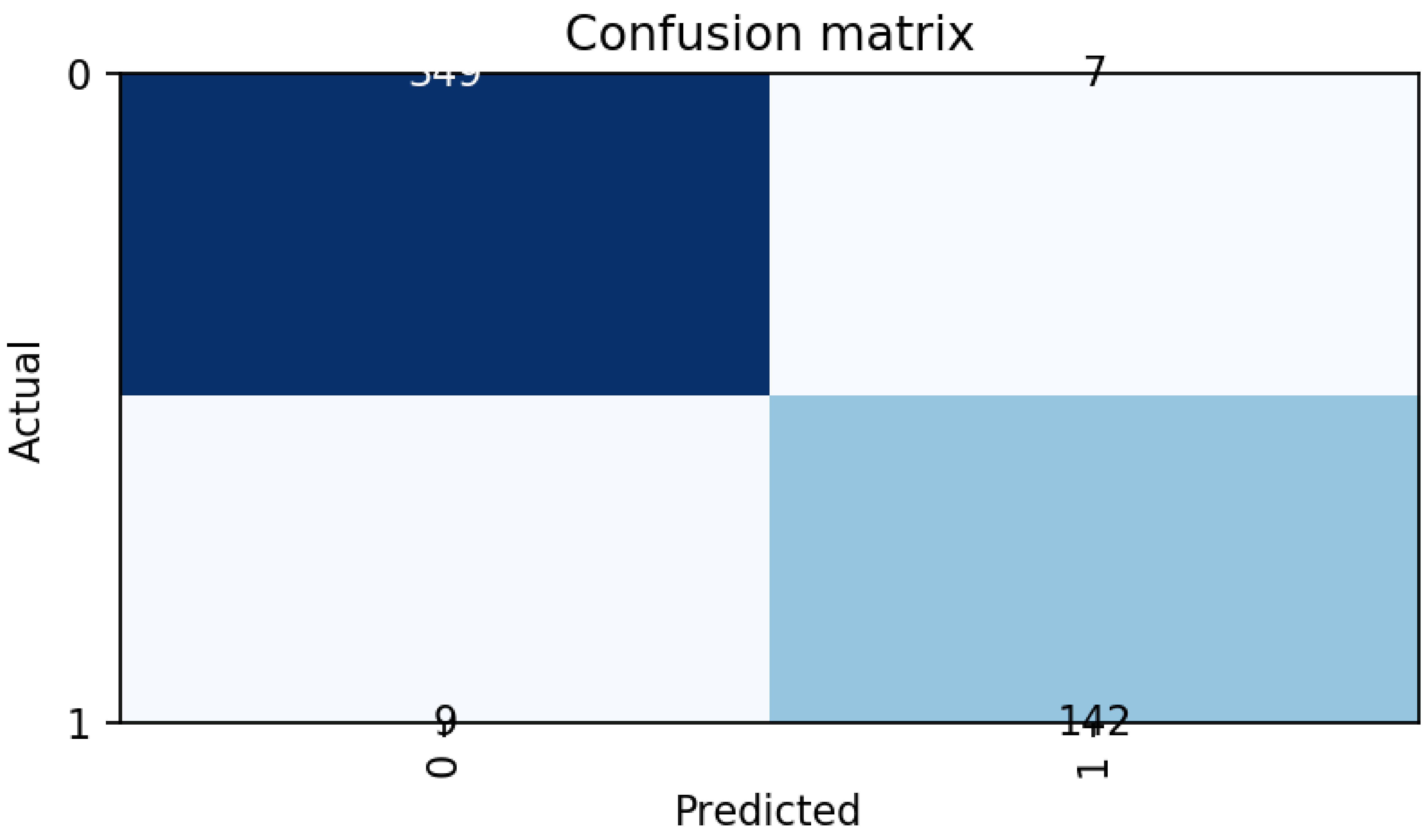


In [54]:

```
interp = ClassificationInterpretation.from_learner(learner)
losses,idxs = interp.top_losses()
len(data.valid_ds)==len(losses)==len(idxs)
# plot
interp.plot_top_losses(9, figsize=(16,16))
```



```
In [55]: interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```

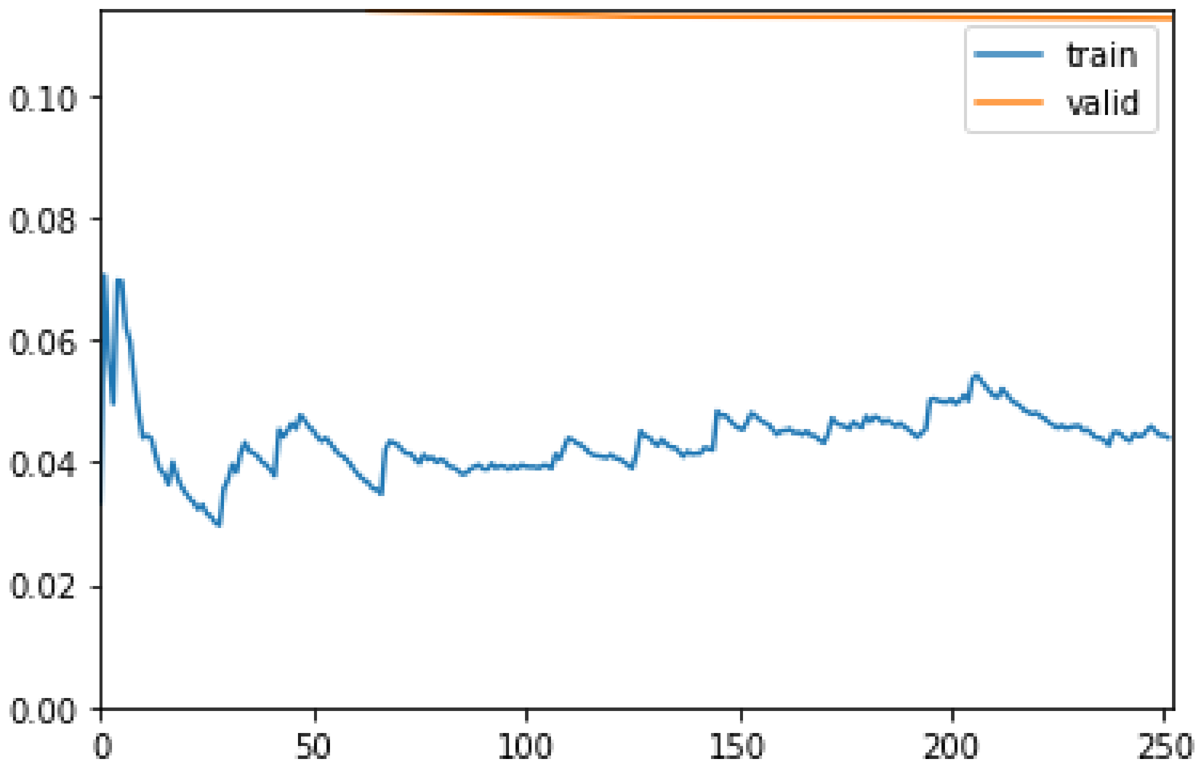


Add x4 epochs using lower learning rate

In [56]:

```
learner.unfreeze()
learner.fit_one_cycle(4, max_lr=1e-6)
```

```
epoch  train_loss  valid_loss  error_rate     auroc   time
    0    0.037134    0.113960    0.027613  0.987350  00:21
    1    0.039050    0.112980    0.027613  0.987127  00:21
    2    0.046436    0.112893    0.029586  0.987555  00:20
    3    0.044054    0.112644    0.025641  0.987090  00:21
```



In [57]:

```
learner.save('stage-2-12epc')
```

In [58]:

```
learner.export('vgg16-16epc-no-rescale')
```