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 [18]:
```
# cell to run the experiments
```

<== End of test section

Train network

In [19]:
```
bs = 32
valid_split = 0.2
```

In [30]:
```
data = ImageDataBunch.from_folder(path/'train', ds_tfms=get_transforms(), size=224, bs=bs, valid_pc
```

In [31]:
```
pd.value_counts(data.train_dl.y.items.flatten(), sort=False)
```

Out[31]:
```
0    2010
1     700
dtype: int64
```

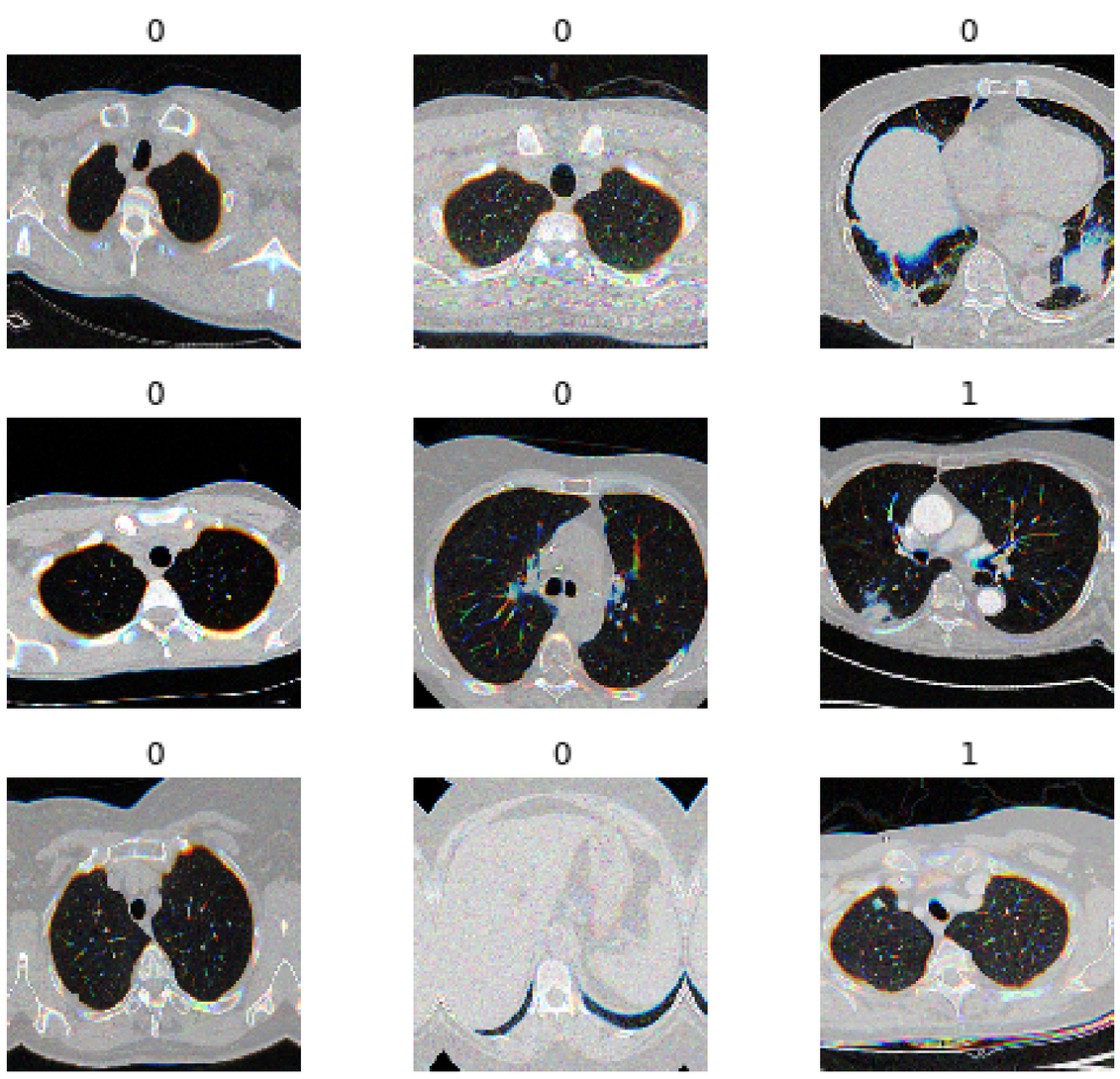
In [32]:
```
pd.value_counts(data.valid_dl.y.items.flatten(), sort=False)
```

Out[32]:
```
0    484
1    193
dtype: int64
```

In [33]:
```
data.classes
```

Out[33]:
```
['0', '1']
```

In [34]:
```
data.show_batch(rows=3, figsize=(7,6))
```

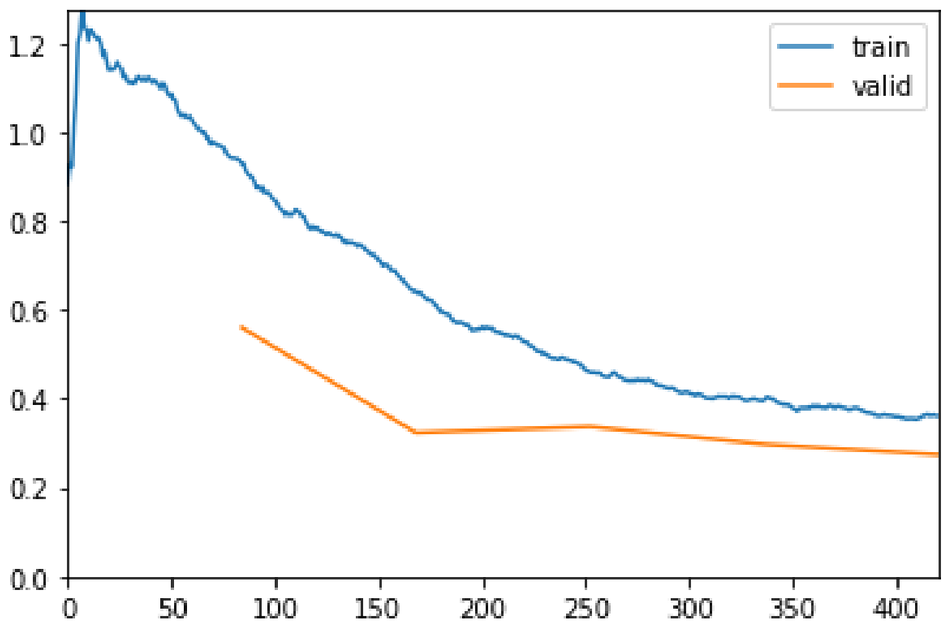


In [35]:
```
learner = cnn_learner(data, models.vgg16_bn, metrics=[error_rate, AUROC()], callback_fns=[ShowGraph
#learner = cnn_learner(data, models.resnet18, metrics=[error_rate, f1_score(), AUROC()], callback_f
```

In [29]:
```
verify_images(path/'train'/'0')
```

In [36]:
```
learner.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.935358	0.560518	0.202363	0.858605	00:13
1	0.643201	0.323814	0.134417	0.914952	00:13
2	0.463373	0.337405	0.137371	0.915182	00:13
3	0.395136	0.297698	0.110783	0.930651	00:13
4	0.361963	0.274341	0.094535	0.939697	00:13

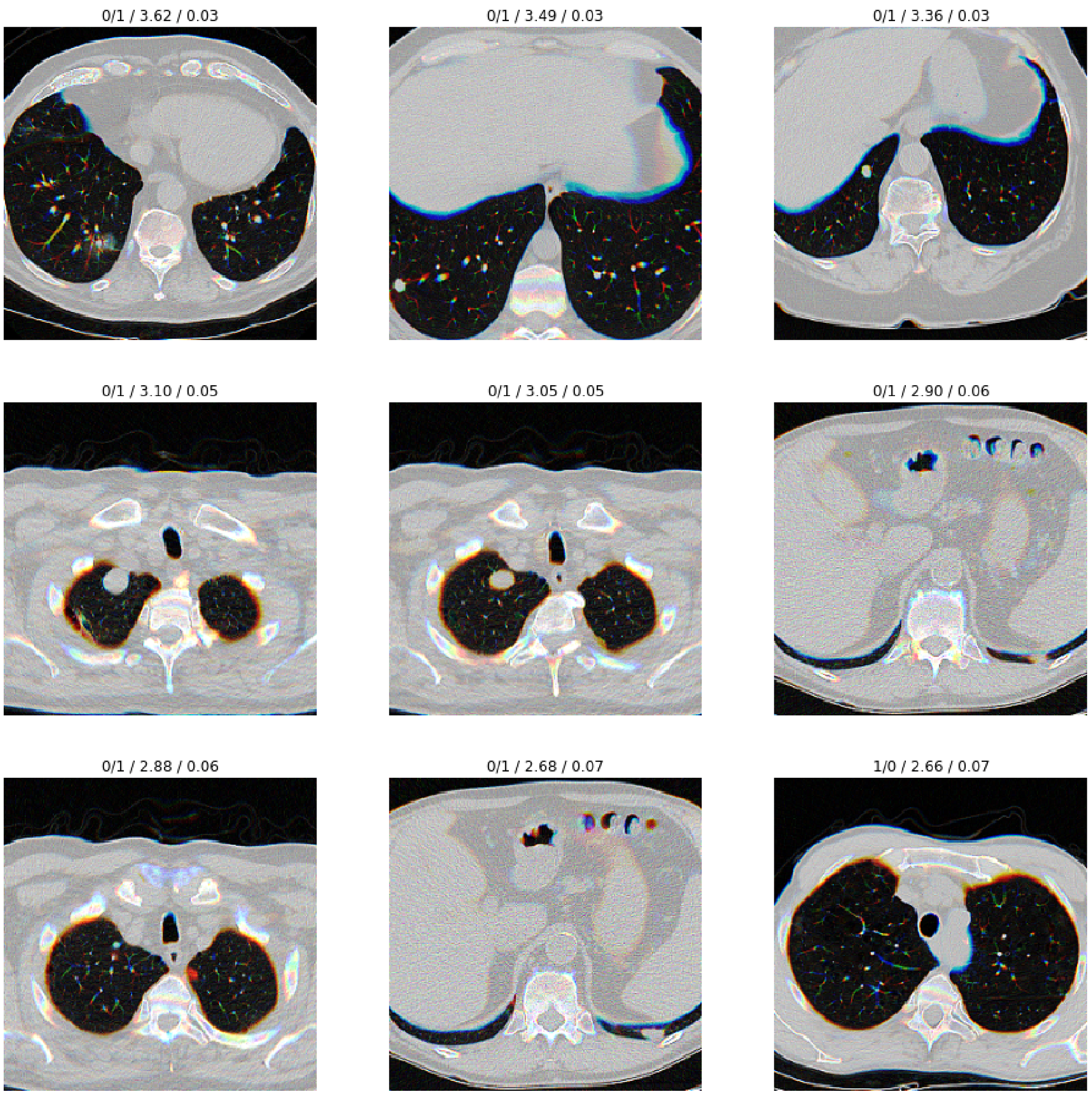


In [37]:
```
learner.save('stage-1')
```

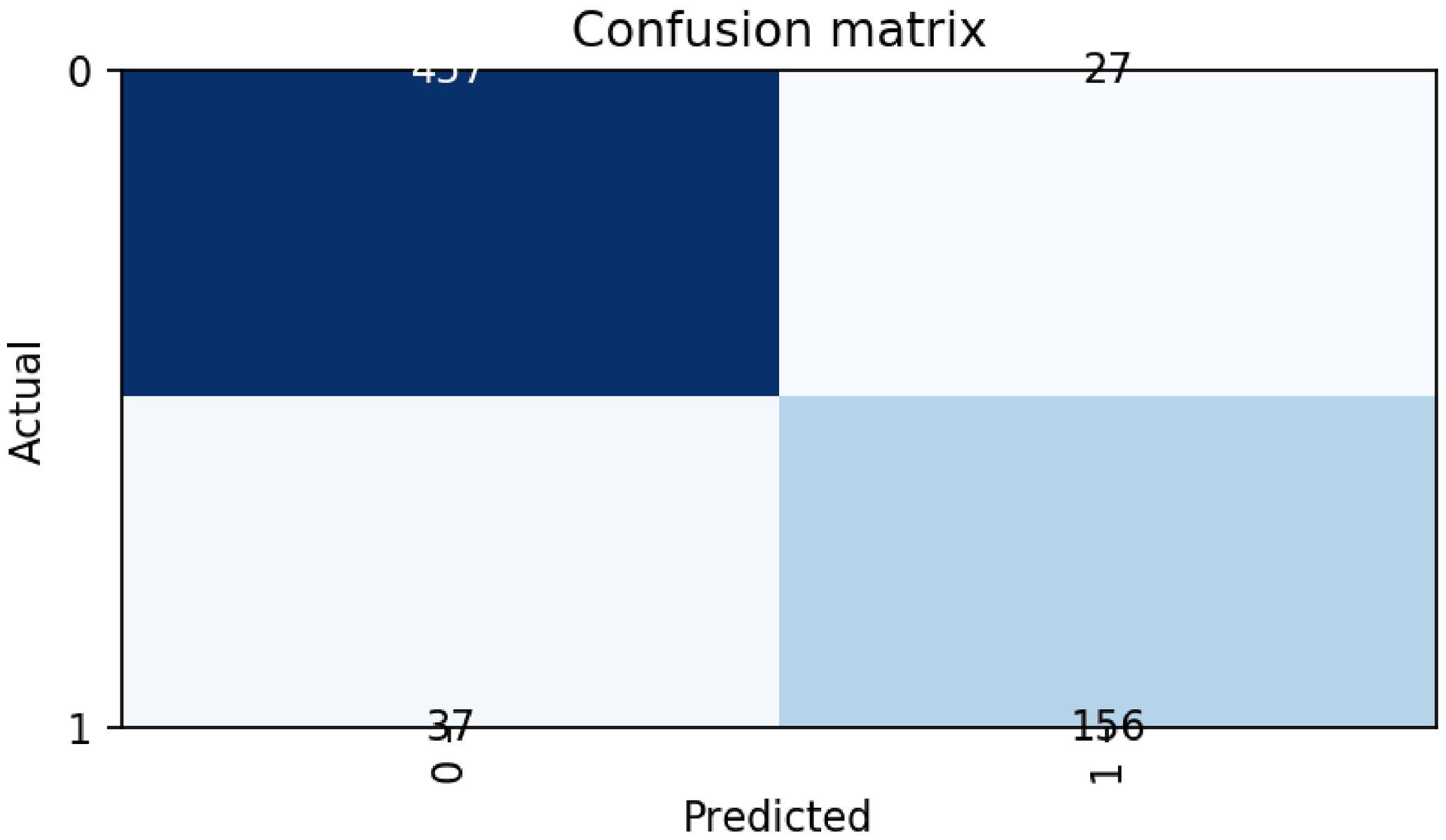
In [38]:
```
interp = ClassificationInterpretation.from_learner(learner)
losses,idxs = interp.top_losses()
len(data.valid_ds)==len(losses)==len(idxs)
```

Out[38]: True

In [40]:
```
interp.plot_top_losses(9, figsize=(16,16))
```



In [41]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



In [42]:
```
interp.most_confused(min_val=2)
```

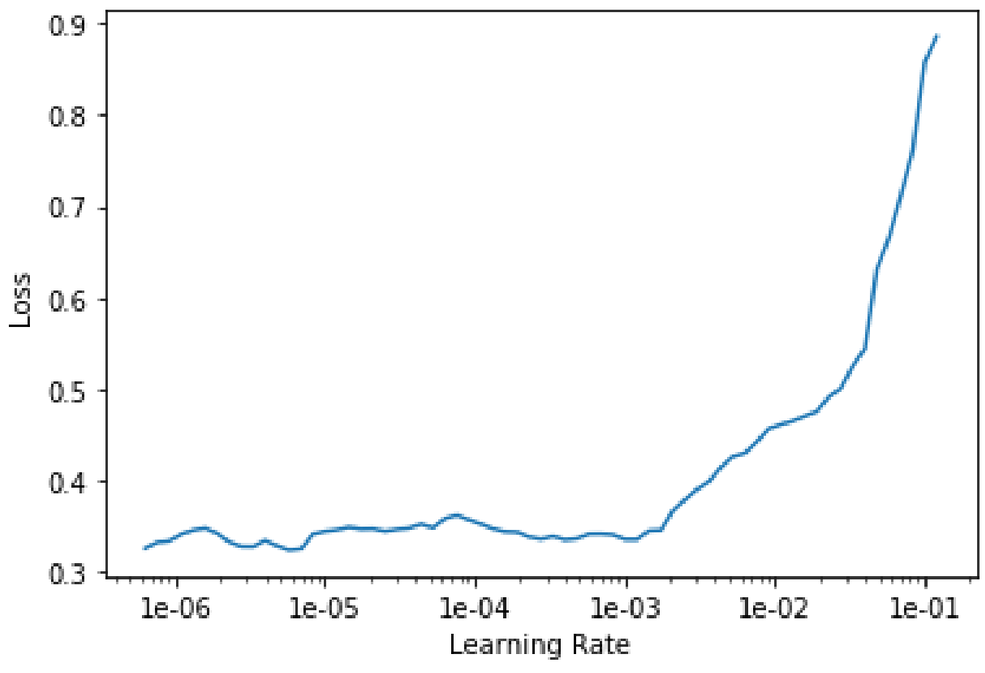
Out[42]:
```
[('1', '0', 37), ('0', '1', 27)]
```

Unfreeze all model layer and tune learning rate

In [43]:
```
learner.unfreeze()
learner.lr_find()
```

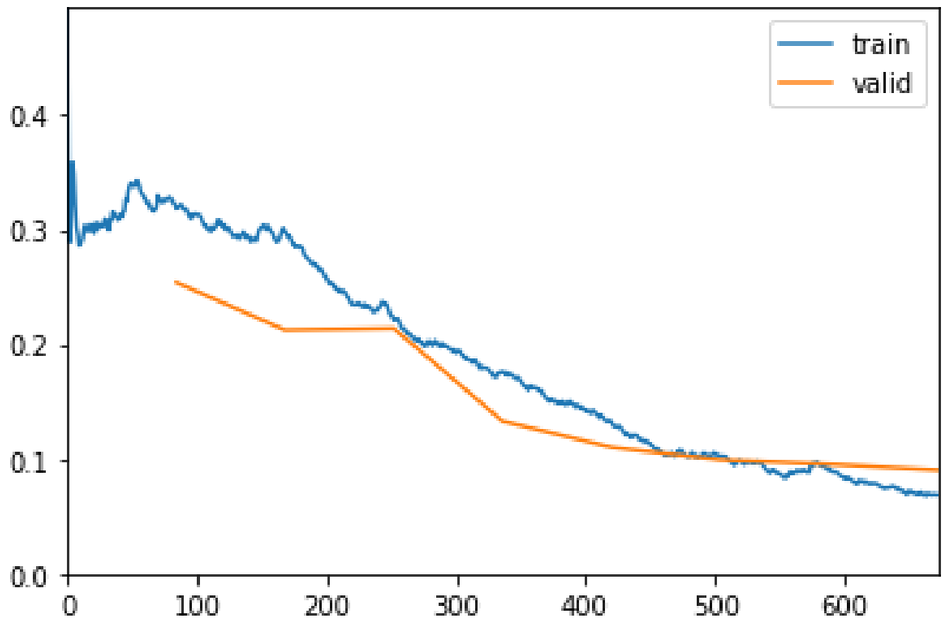
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [44]:
```
learner.recorder.plot()
```



In [45]:
```
learner.fit_one_cycle(8, max_lr=slice(1e-5, 3e-4))
```

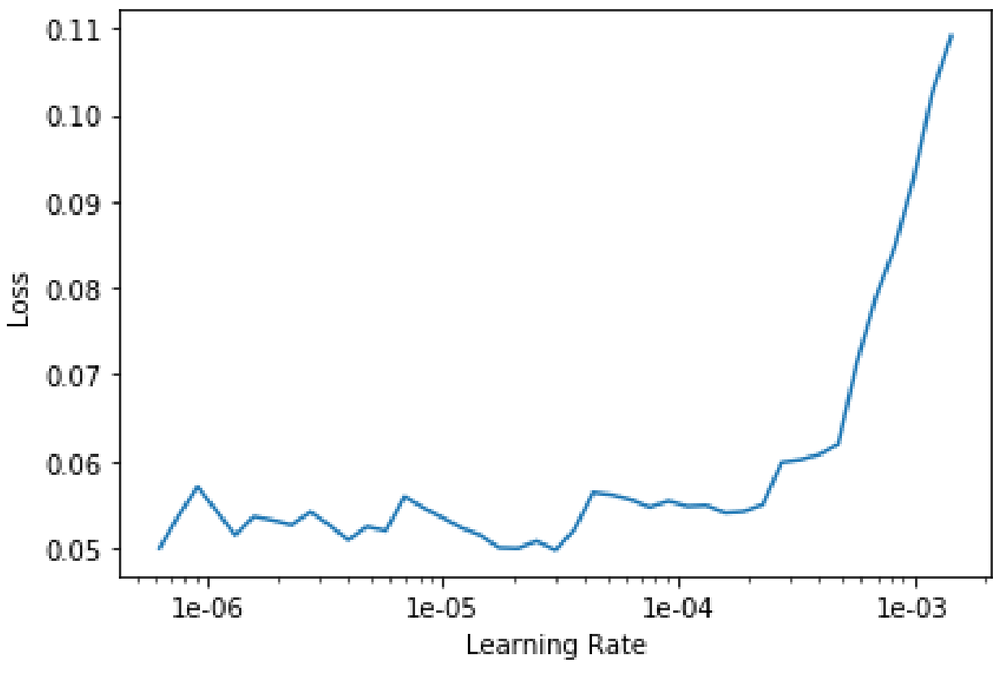
epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.321818	0.254089	0.090103	0.947694	00:14
1	0.301639	0.212658	0.078287	0.967520	00:15
2	0.224414	0.214036	0.081241	0.979285	00:14
3	0.175951	0.133081	0.053176	0.984857	00:14
4	0.133948	0.110364	0.032496	0.990408	00:14
5	0.103333	0.099352	0.032496	0.991050	00:14
6	0.090752	0.095578	0.029542	0.990911	00:14
7	0.068975	0.090498	0.031019	0.992538	00:15



In [46]:
```
learner.save('stage-2-8epc')
```

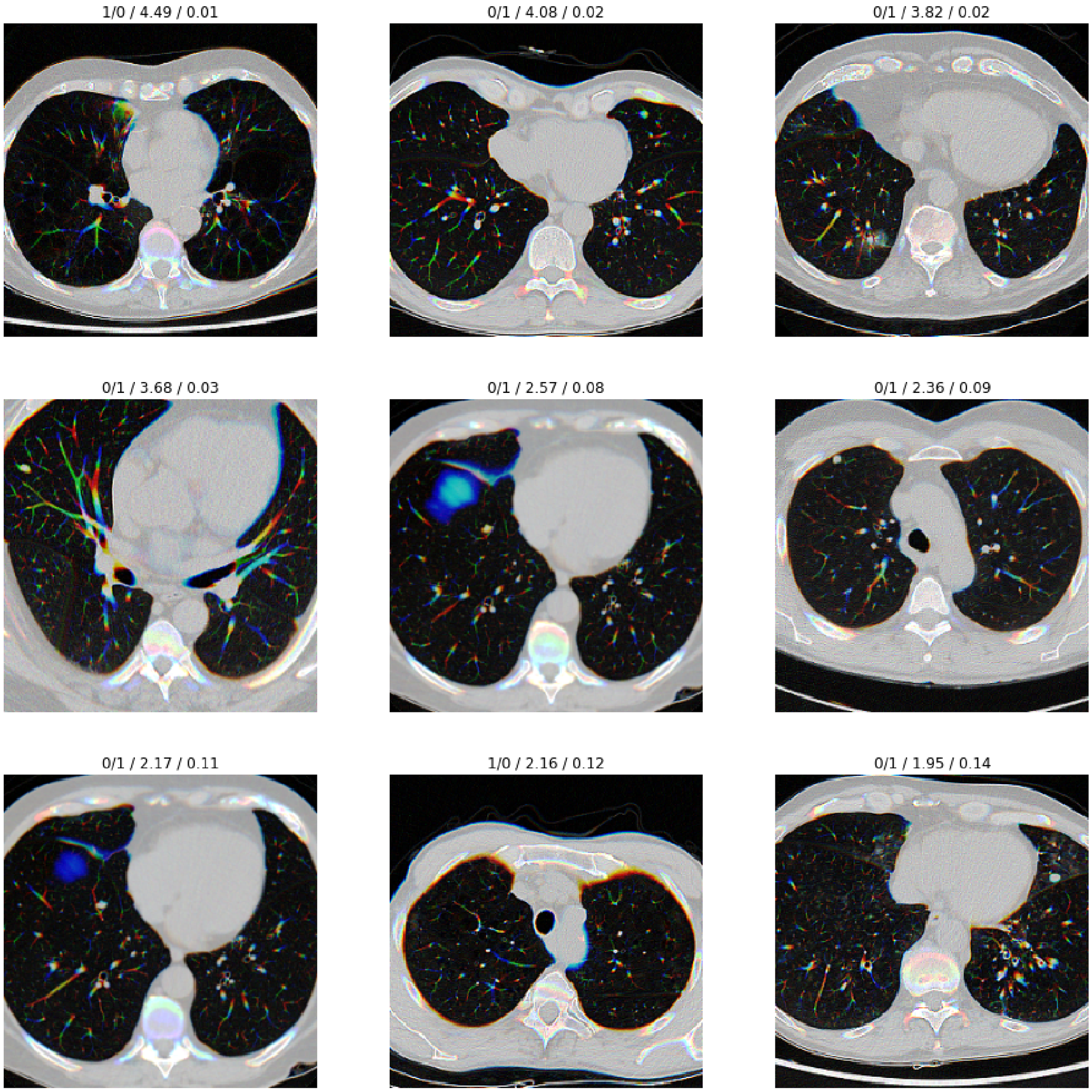
In [52]:
```
learner.lr_find()
learner.recorder.plot()
```

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

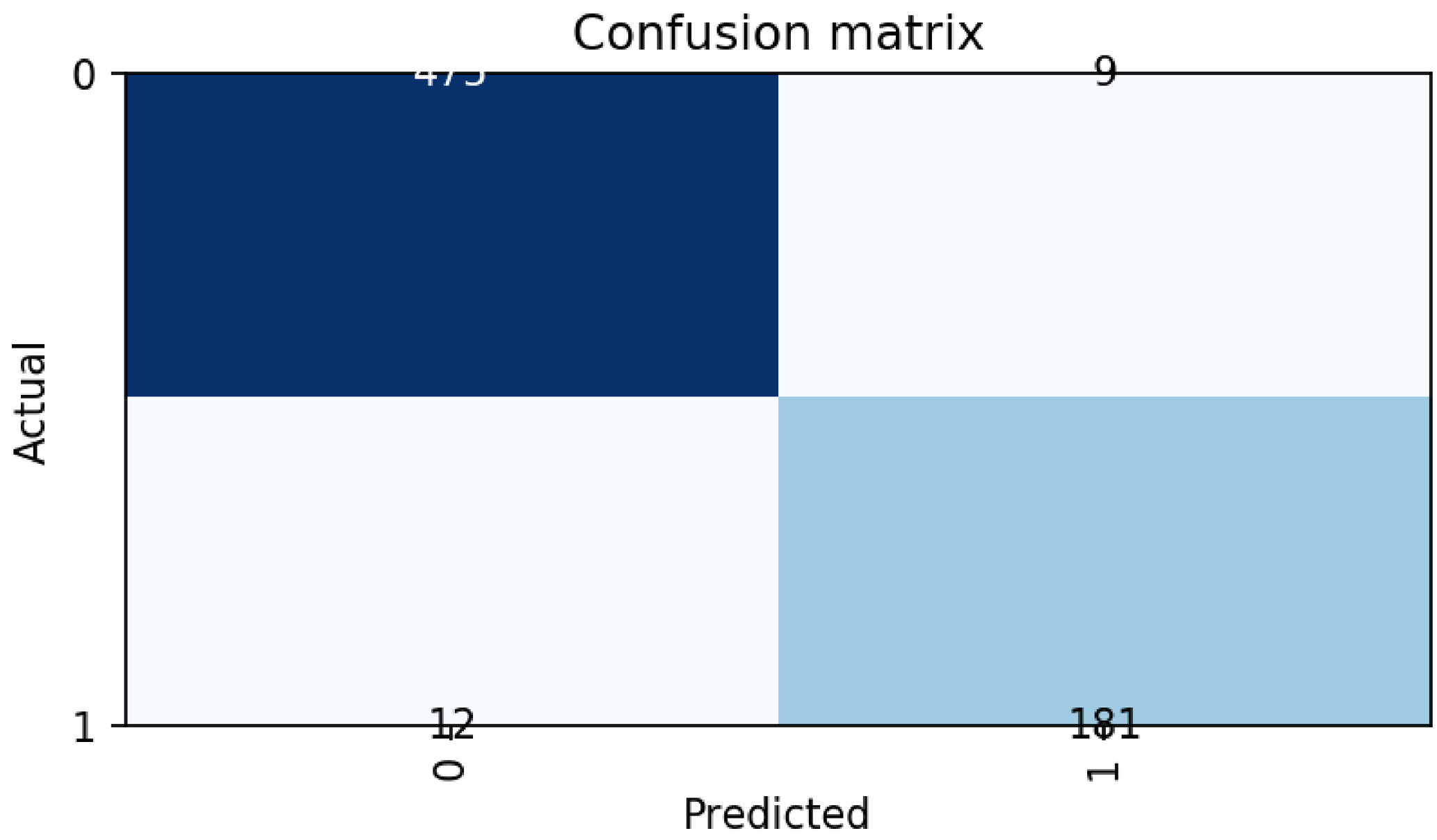


In [49]:
```
interp = ClassificationInterpretation.from_learner(learner)
losses,idxs = interp.top_losses()
len(data.valid_ds)==len(losses)==len(idxs)
# plot
interp.plot_top_losses(9, figsize=(16,16))
```

prediction/actual/loss/probability



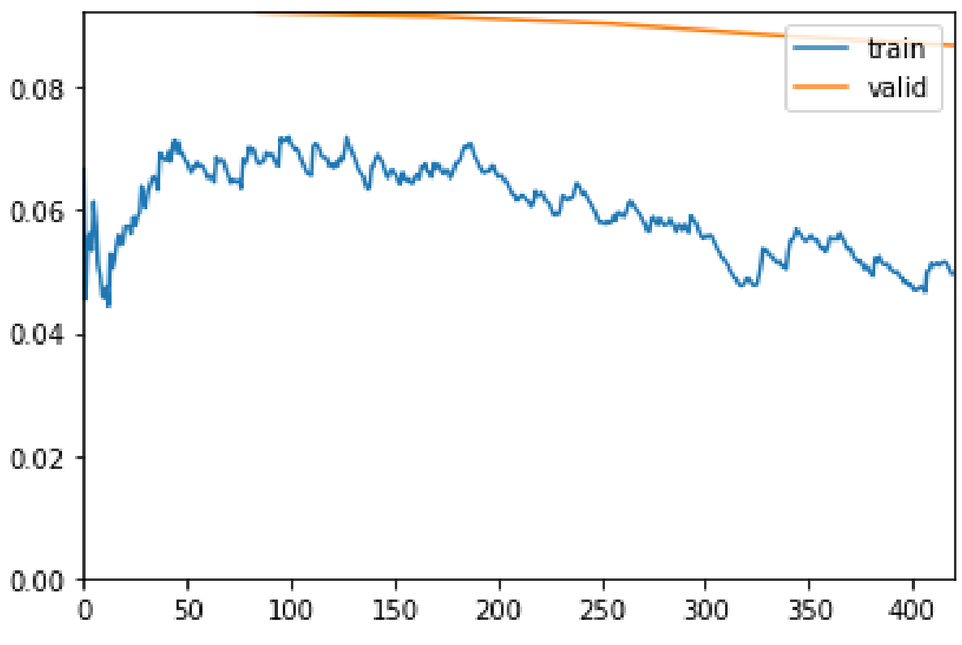
In [50]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



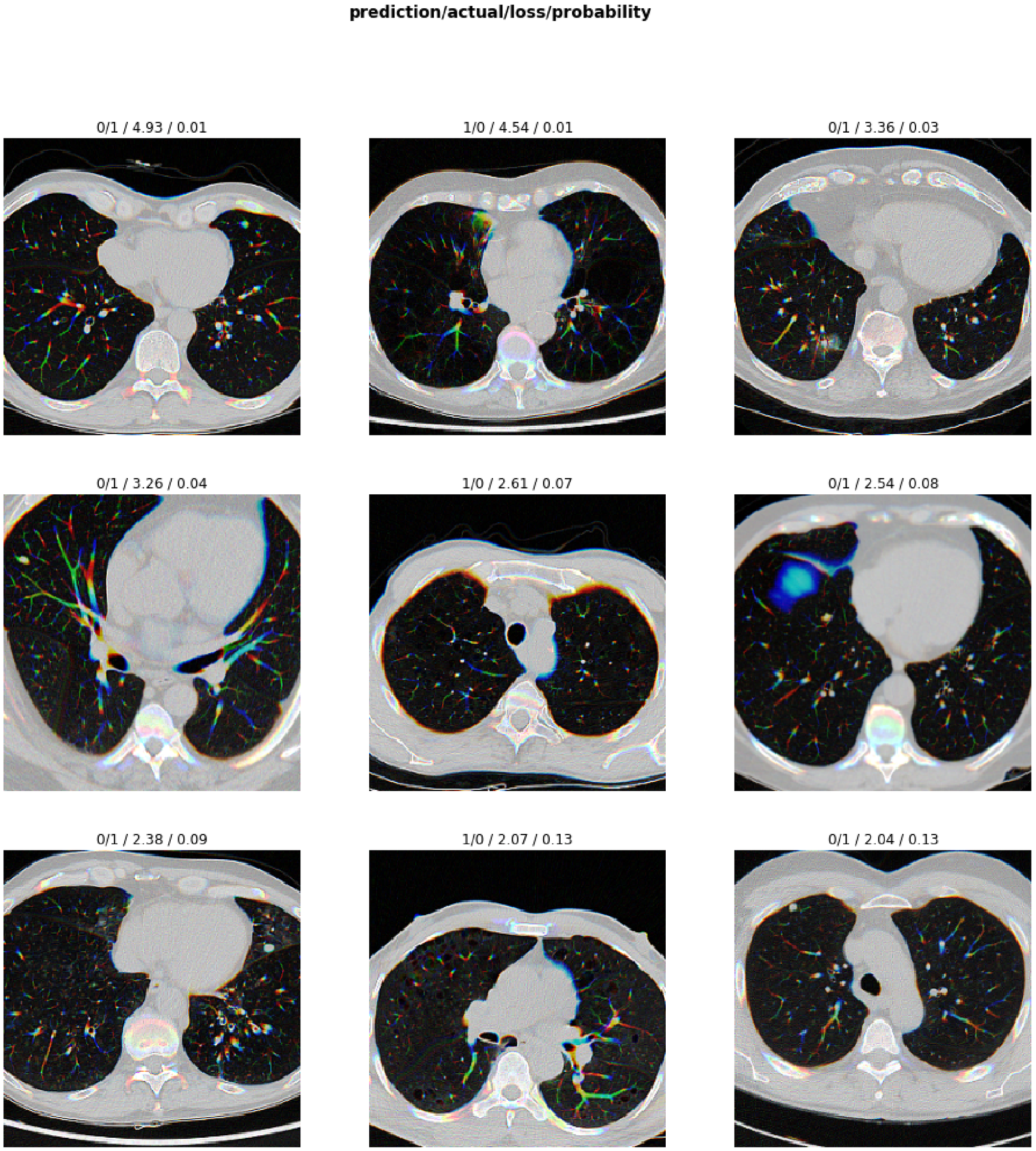
Add x4 epochs using lower learning rate

In [53]:
```
learner.unfreeze()
learner.fit_one_cycle(5, max_lr=1e-5)
```

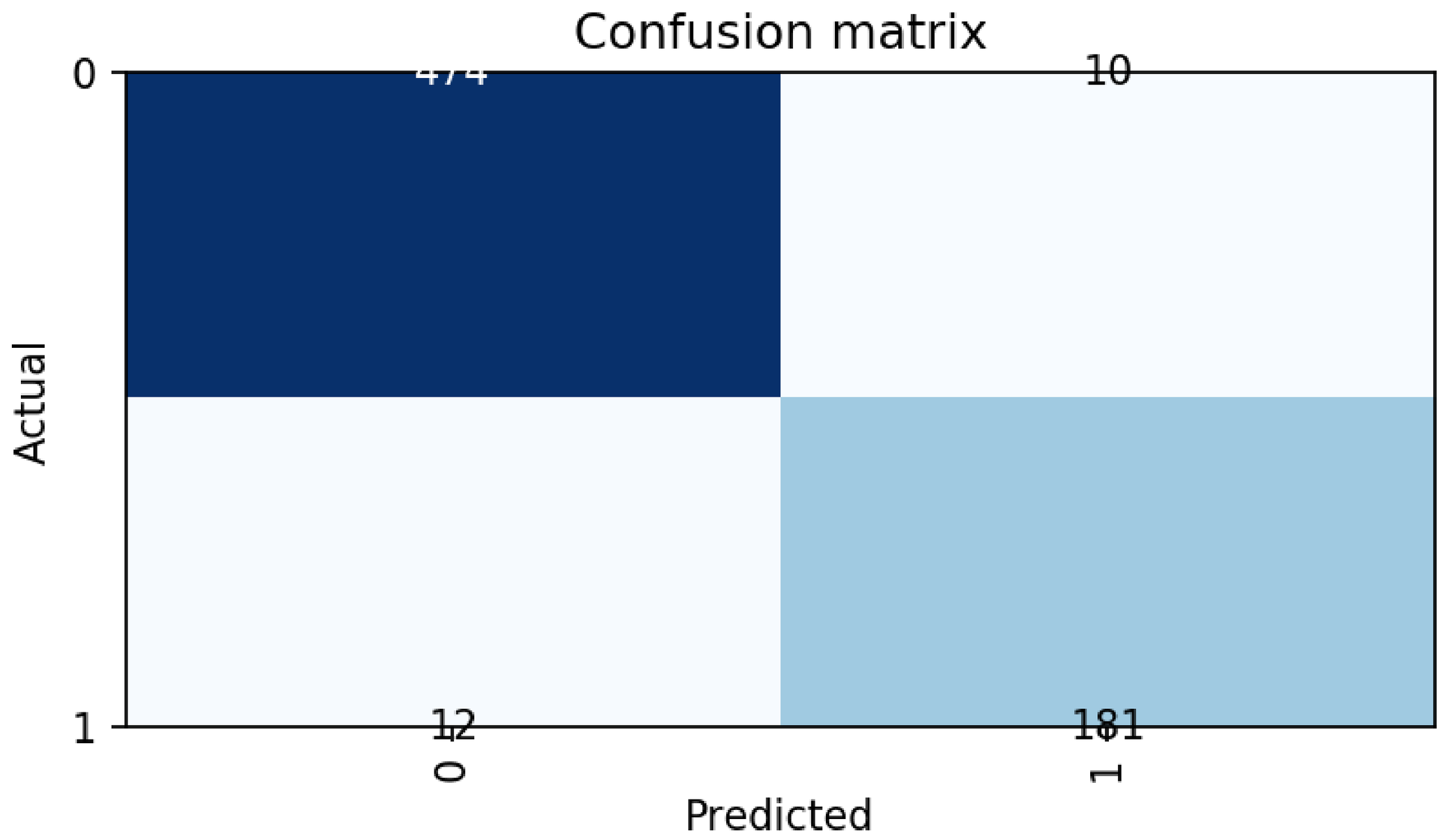
epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.068937	0.092331	0.032496	0.992271	00:14
1	0.066127	0.091716	0.036928	0.992876	00:14
2	0.057957	0.090538	0.032496	0.993031	00:15
3	0.051425	0.088523	0.029542	0.993534	00:14
4	0.049669	0.086949	0.032496	0.993497	00:14



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))
```

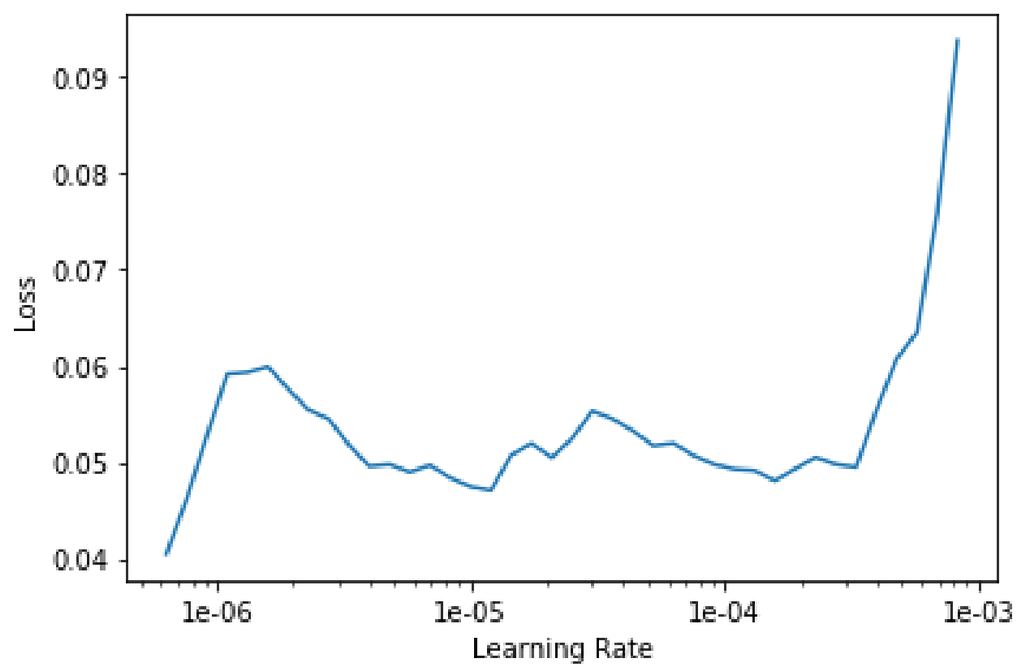


In [55]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



In [60]:
```
learner.lr_find()
learner.recorder.plot()
```

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



In [58]:
```
# # get most confused filenames
# losses,idxs = interp.top_losses(10)
# for p in data.valid_ds.x.items[idxs]:
#     print(p)
```

In [61]:
```
learner.save('stage-2-13epc')
```

In [63]:
```
learner.export('vgg16-18epc-rescale-crop256')
```