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 [20]:
```
data = ImageDataBunch.from_folder(path/'train', ds_tfms=get_transforms(), size=224, bs=bs, valid_pc
```

In [21]:
```
pd.value_counts(data.train_dl.y.items.flatten(), sort=False)
```

Out[21]:
```
0    2672
1     716
dtype: int64
```

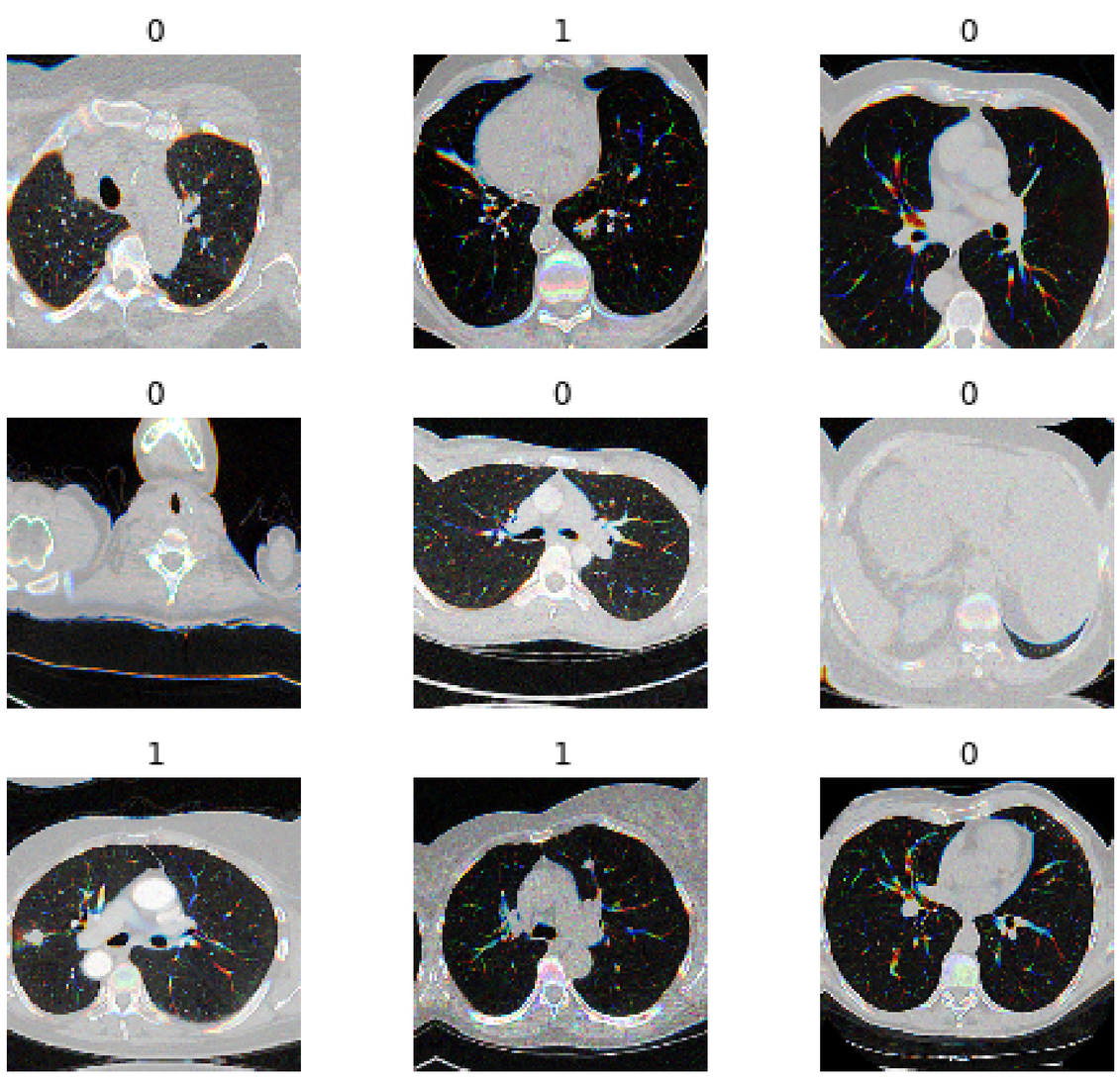
In [22]:
```
pd.value_counts(data.valid_dl.y.items.flatten(), sort=False)
```

Out[22]:
```
0    669
1    177
dtype: int64
```

In [23]:
```
data.classes
```

Out[23]:
```
['0', '1']
```

In [24]:
```
data.show_batch(rows=3, figsize=(7,6))
```

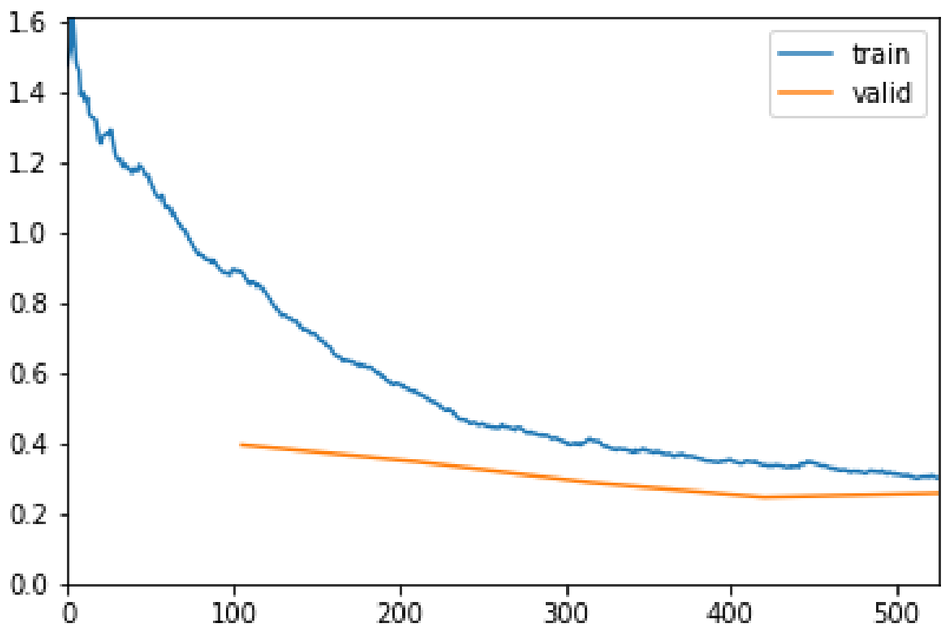


In [26]:
```
learner = cnn_learner(data, models.vgg16_bn, metrics=[error_rate, AUROC()], callback_fns=[ShowGraph
#learner = cnn_learner(data, models.resnet18, metrics=[error_rate, f1_score(), AUROC()], callback_f
```

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

In [27]:
```
learner.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.889512	0.394543	0.160756	0.862359	00:18
1	0.548829	0.347096	0.138298	0.868076	00:16
2	0.409311	0.287748	0.118203	0.915233	00:15
3	0.337301	0.245484	0.096927	0.940222	00:16
4	0.301509	0.257633	0.105201	0.935928	00:15

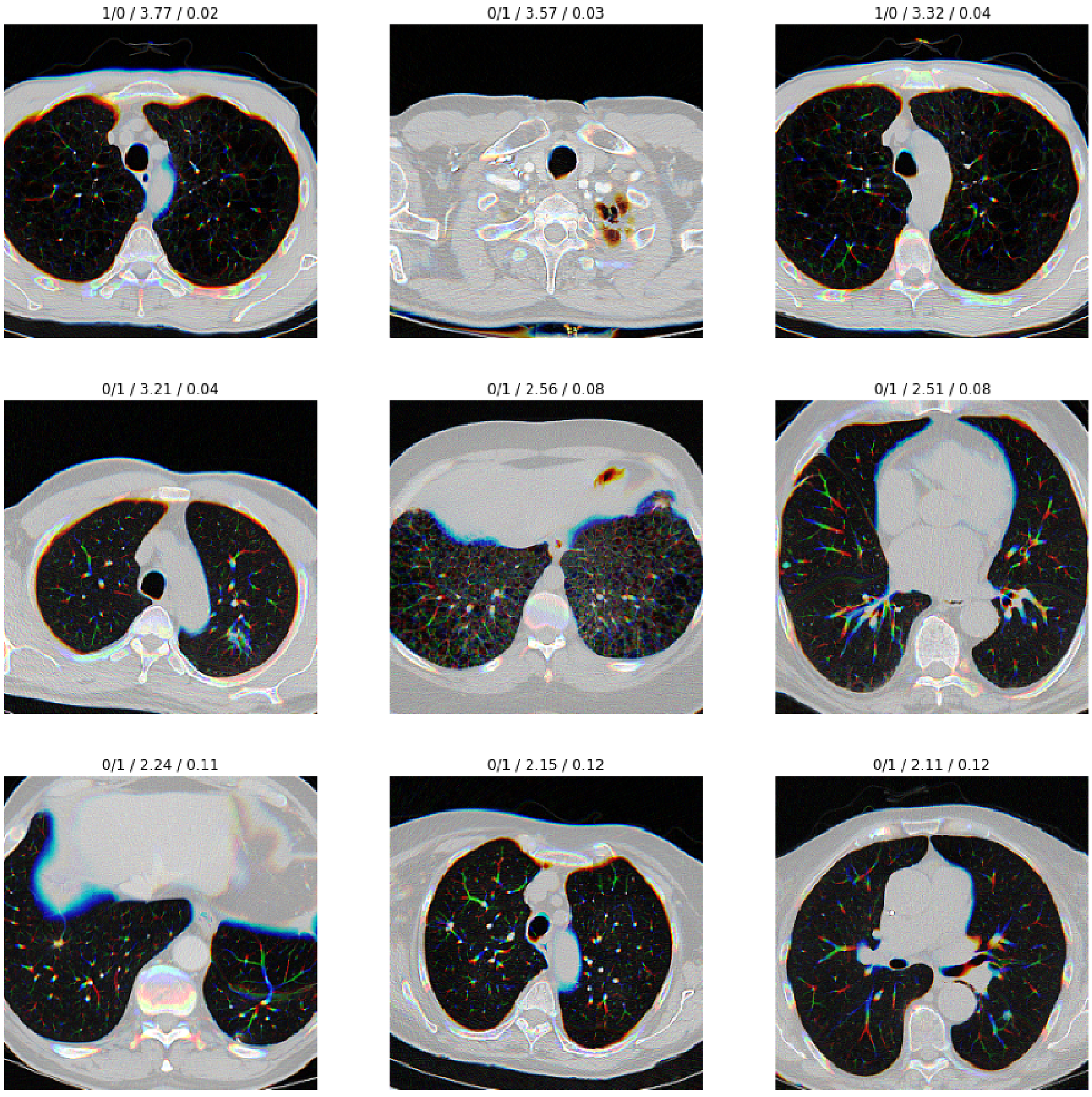


In [28]:
```
learner.save('stage-1')
```

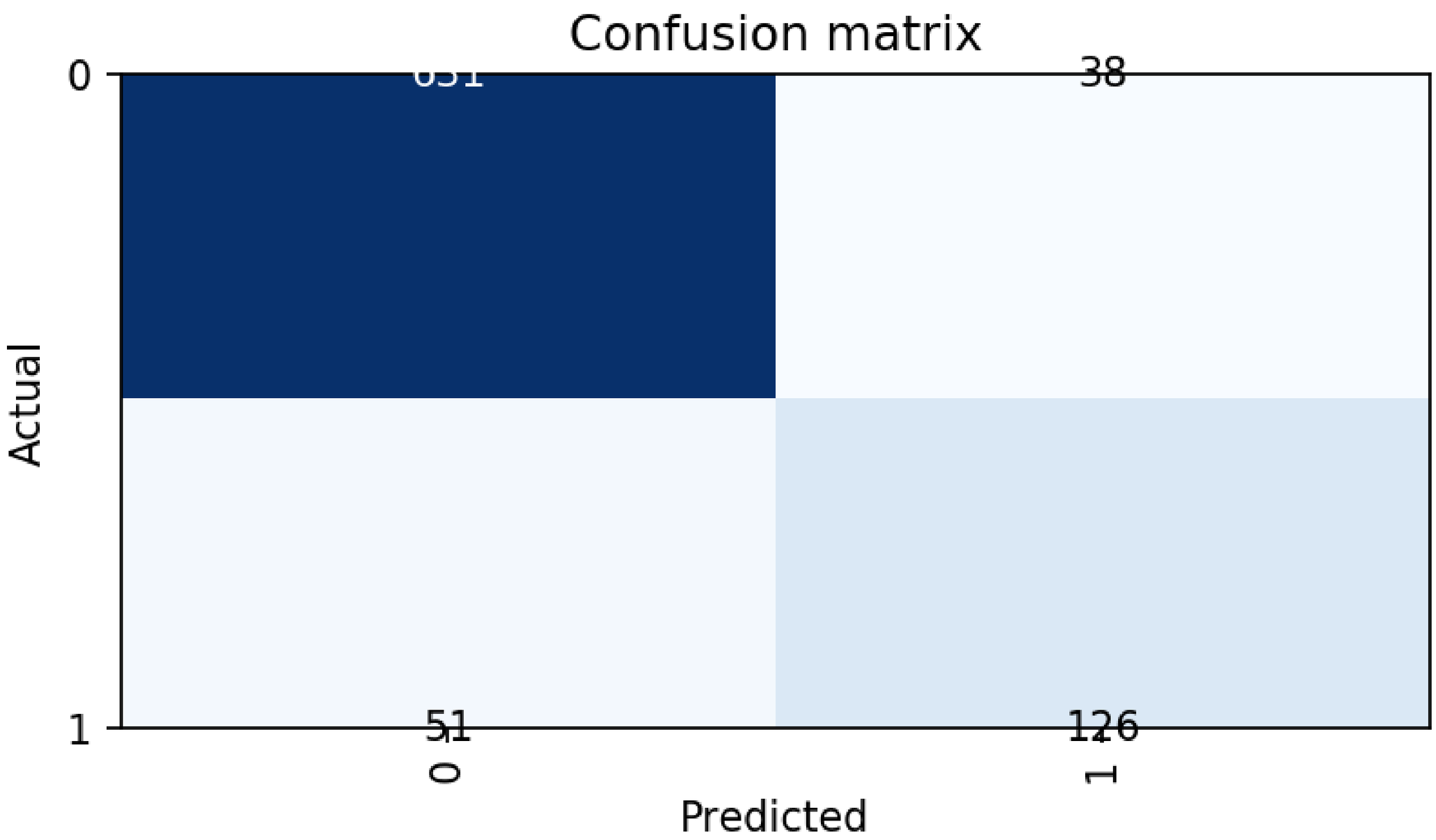
In [29]:
```
interp = ClassificationInterpretation.from_learner(learner)
losses,idxs = interp.top_losses()
len(data.valid_ds)==len(losses)==len(idxs)
```

Out[29]:
```
True
```

In [30]:
```
interp.plot_top_losses(9, figsize=(16,16))
```



In [31]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



In [32]:
```
interp.most_confused(min_val=2)
```

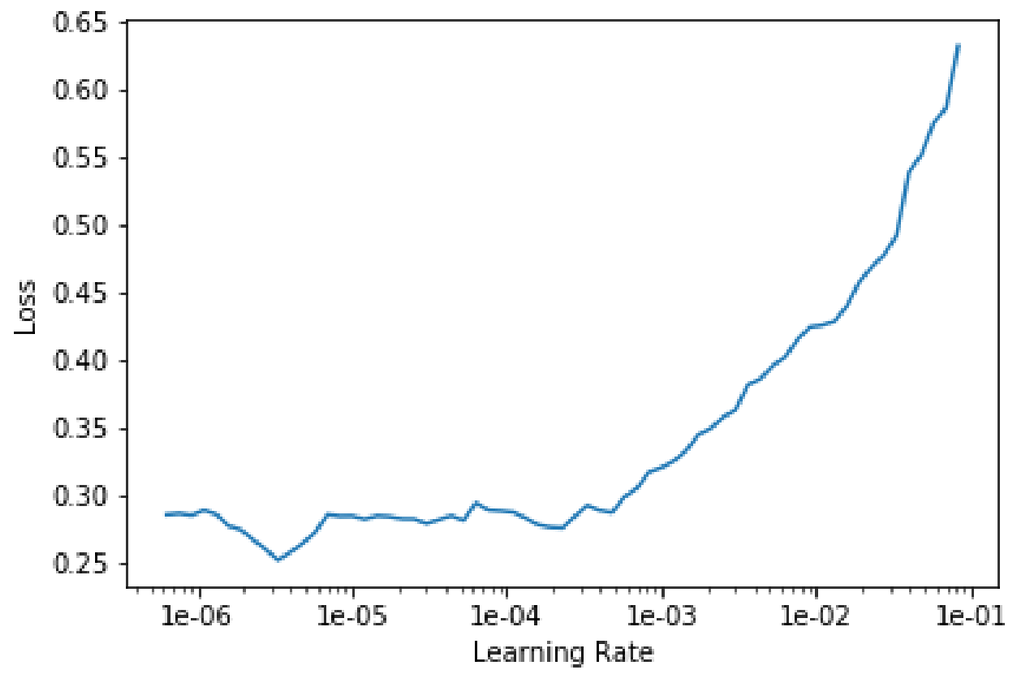
Out[32]:
```
[('1', '0', 51), ('0', '1', 38)]
```

Unfreeze all model layer and tune learning rate

In [33]:
```
learner.unfreeze()
learner.lr_find()
```

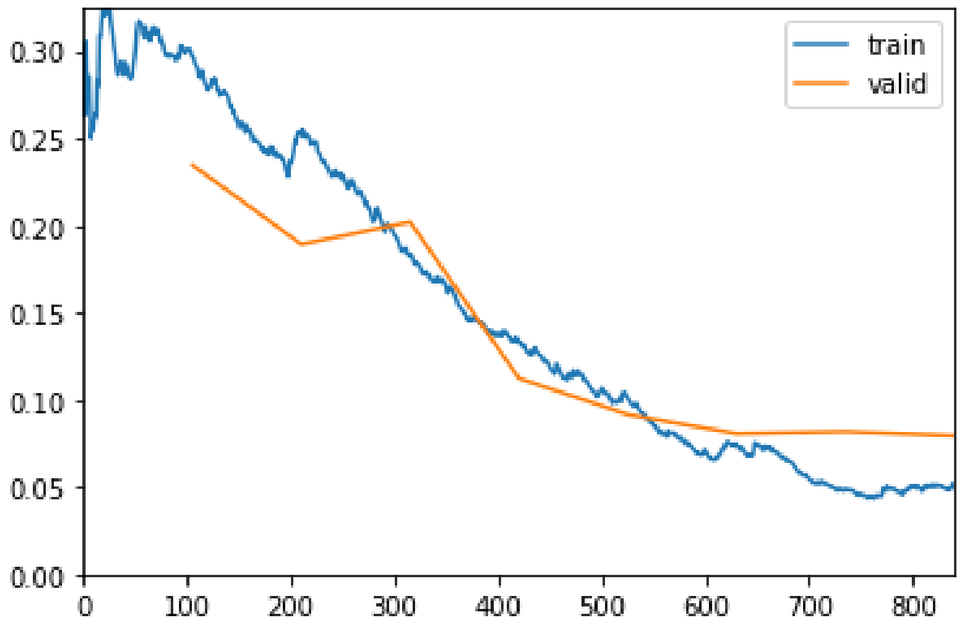
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [34]:
```
learner.recorder.plot()
```



In [35]:
```
learner.fit_one_cycle(8, max_lr=slice(1e-5, 3e-4))
```

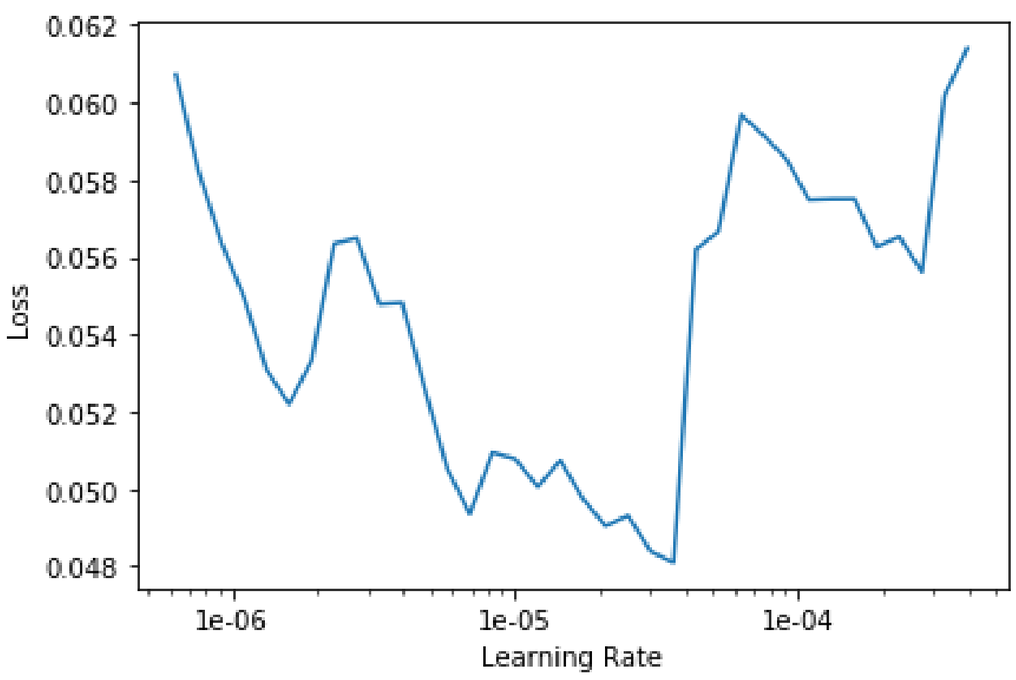
epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.297720	0.234594	0.100473	0.953130	00:17
1	0.252415	0.189253	0.072104	0.972098	00:17
2	0.183706	0.202034	0.081560	0.979229	00:17
3	0.133181	0.112315	0.034279	0.984596	00:17
4	0.101831	0.091629	0.030733	0.988164	00:18
5	0.075065	0.080705	0.021277	0.989727	00:18
6	0.048595	0.081809	0.027187	0.988836	00:17
7	0.051993	0.079529	0.026005	0.989752	00:17



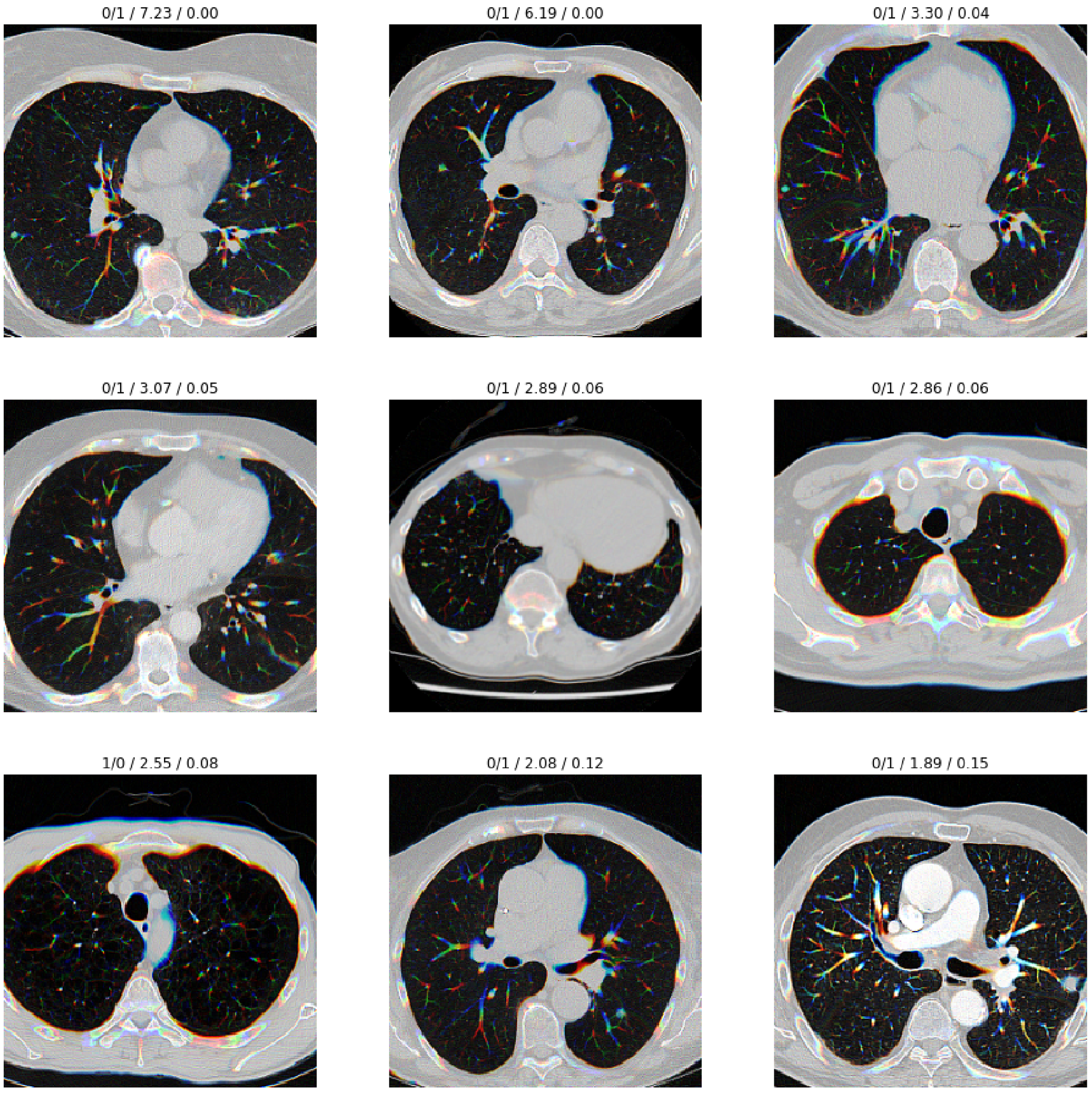
In [36]:
```
learner.save('stage-2-8epc')
```

In [37]:
```
learner.lr_find()
learner.recorder.plot()
```

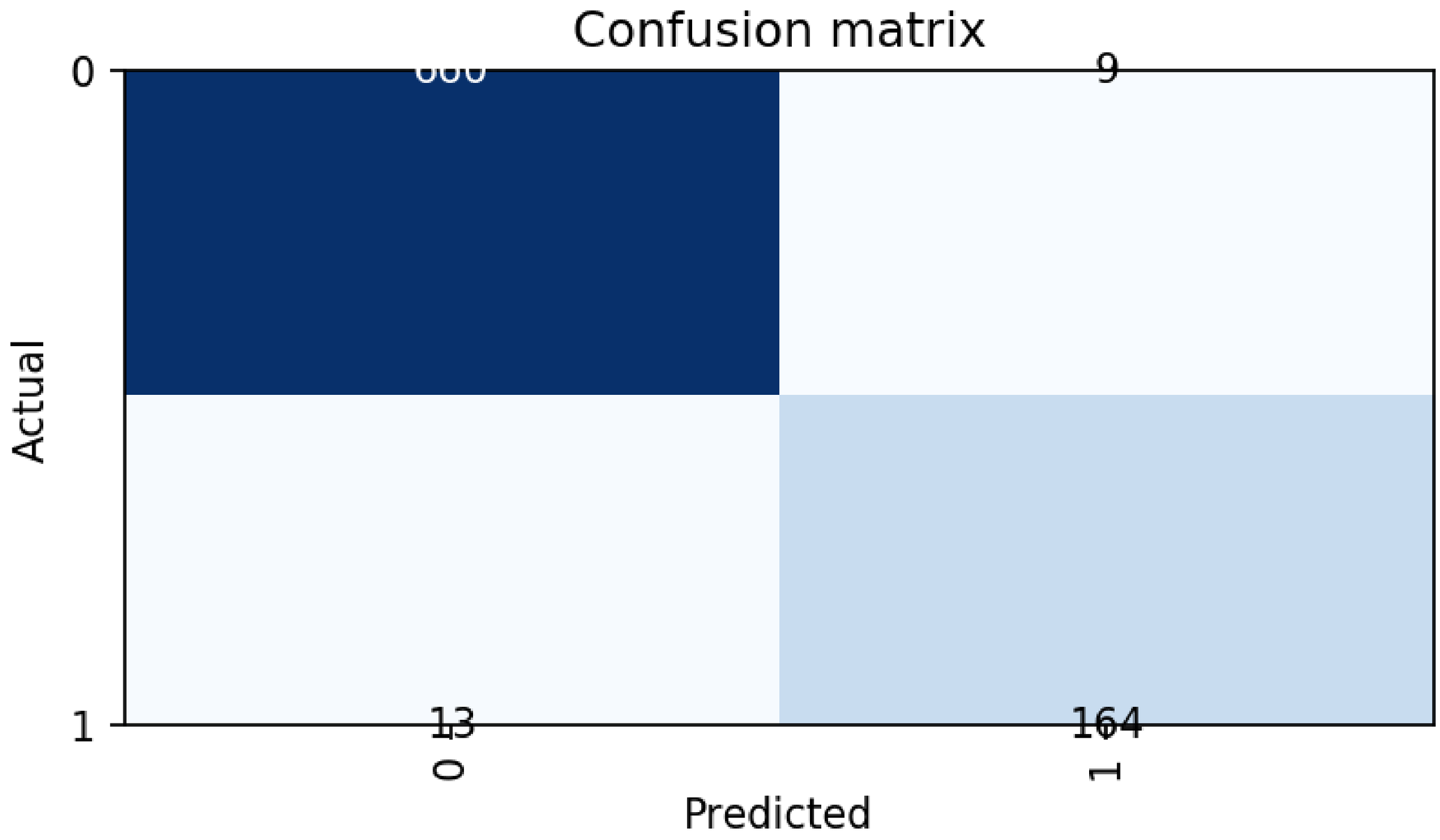
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



In [38]:
```
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))
```



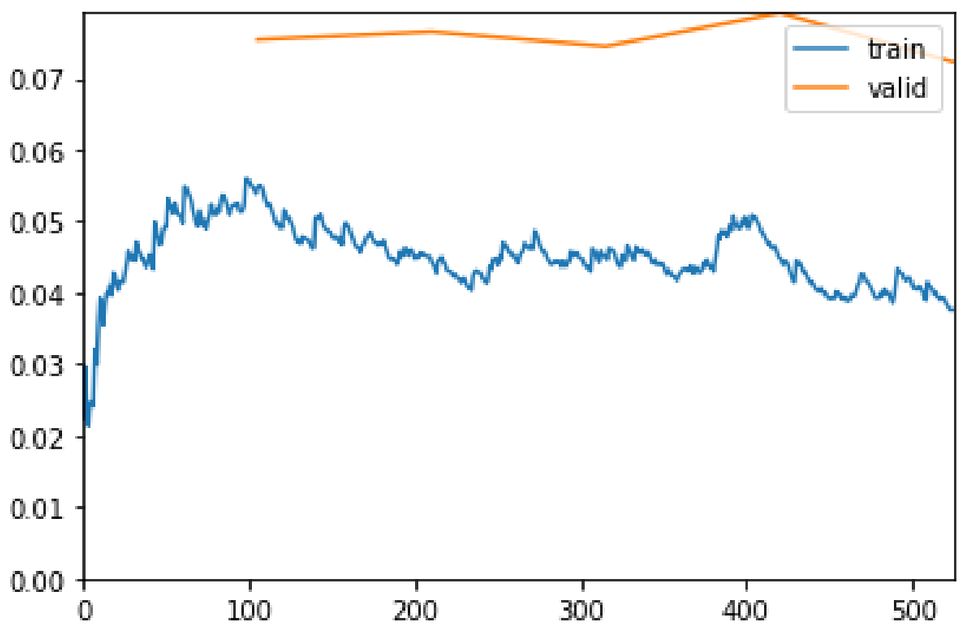
In [39]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



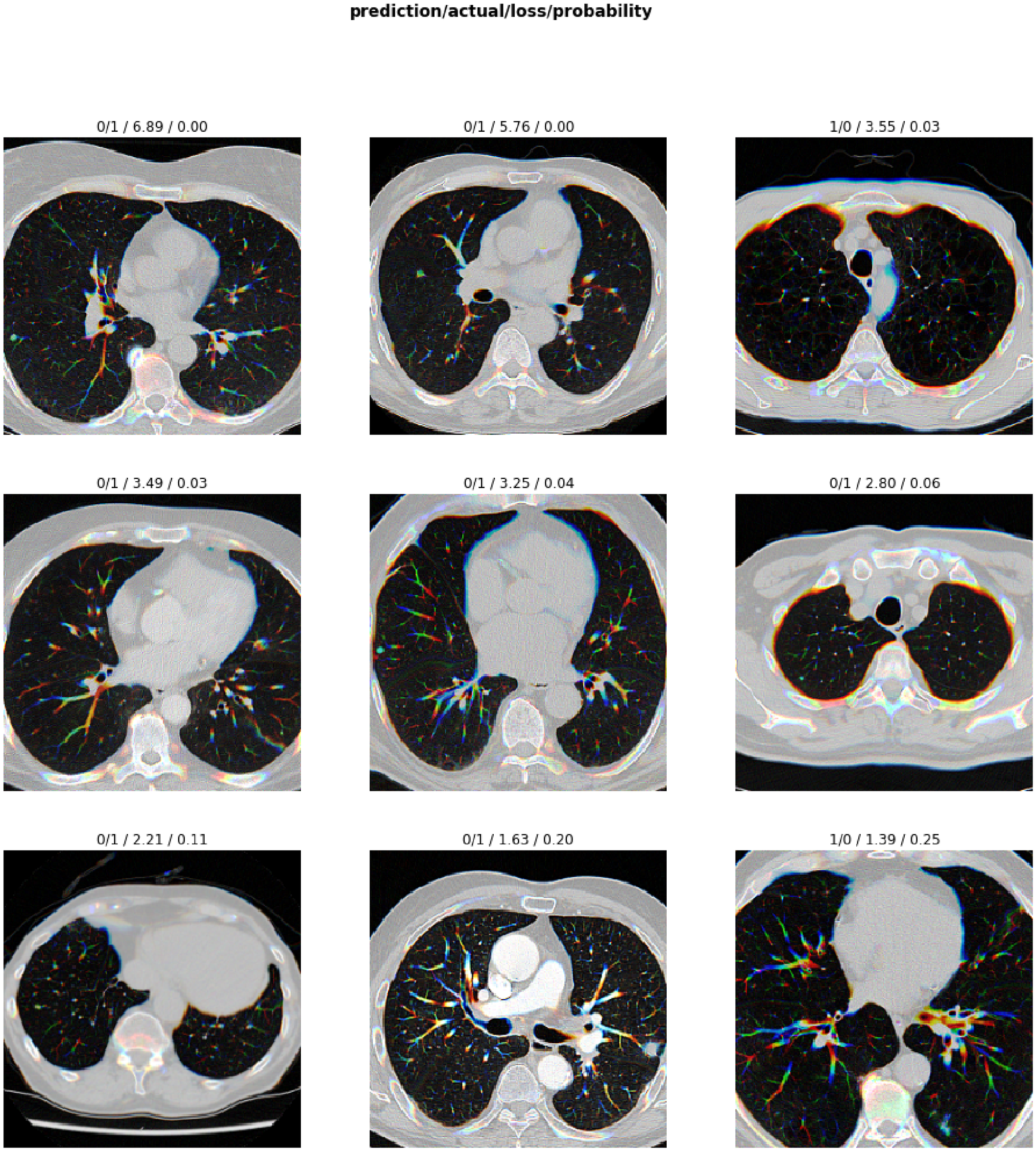
Add x4 epochs using lower learning rate

In [40]:
```
learner.unfreeze()
learner.fit_one_cycle(5, max_lr=1e-5)
```

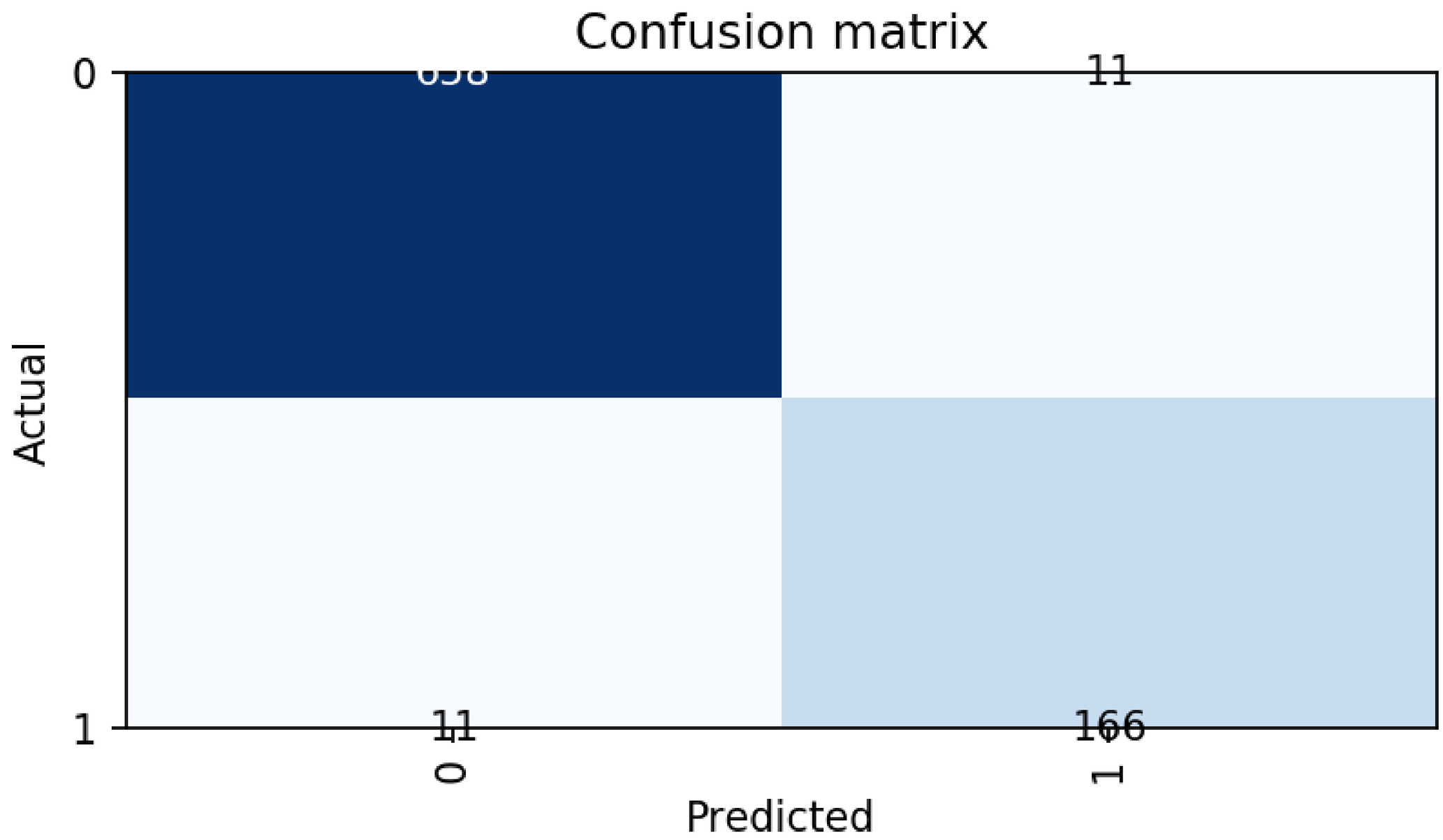
epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.053889	0.075421	0.018913	0.989140	00:17
1	0.045037	0.076536	0.026005	0.989292	00:18
2	0.045115	0.074487	0.024823	0.989659	00:18
3	0.045150	0.079163	0.026005	0.989731	00:17
4	0.037624	0.072283	0.026005	0.990390	00:18



In [41]:
```
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))
```

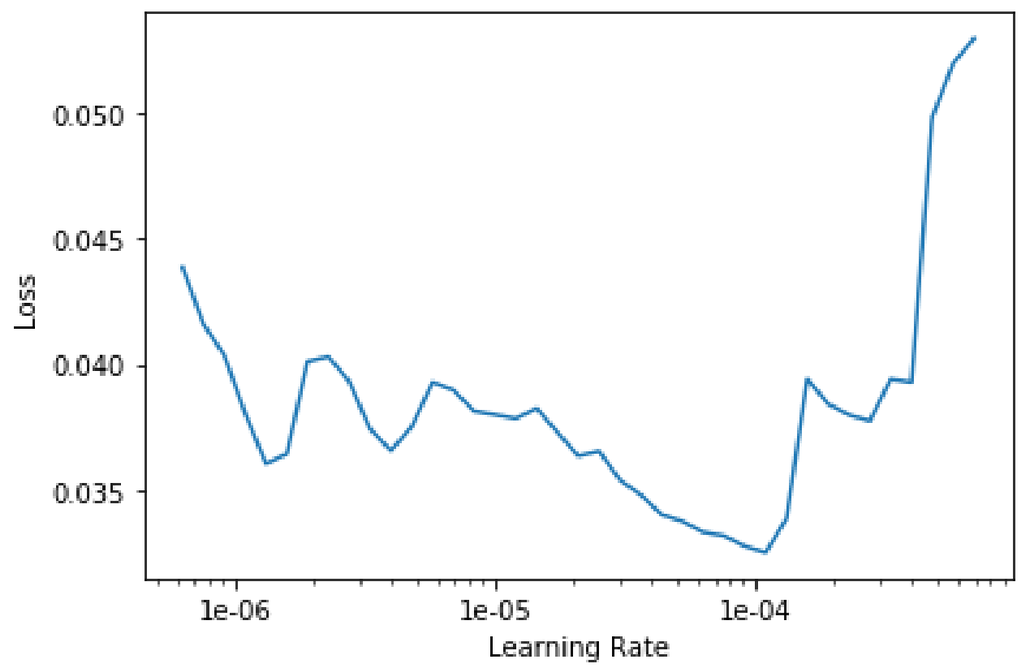


In [42]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



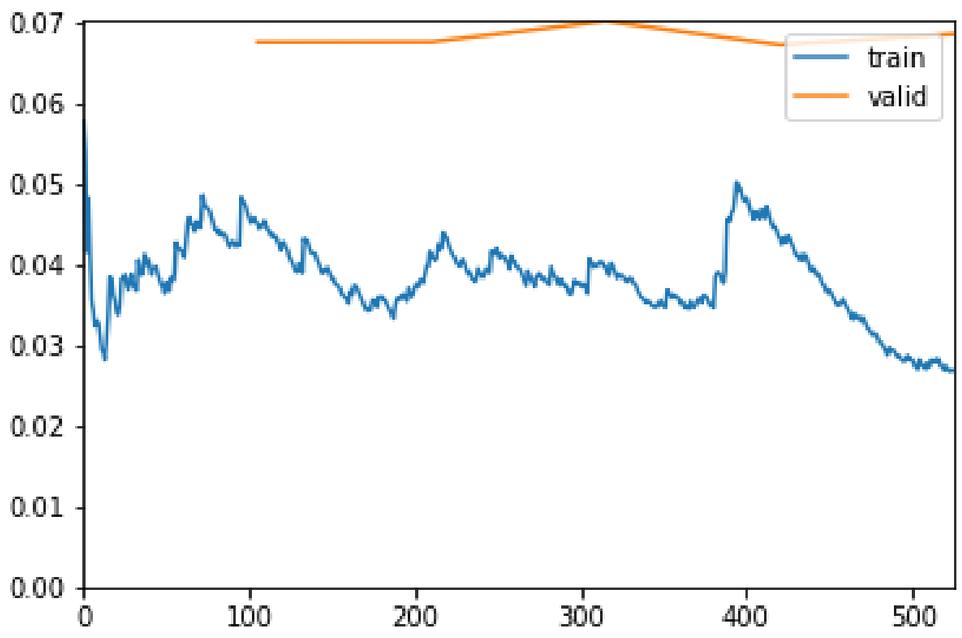
In [43]:
```
learner.lr_find()
learner.recorder.plot()
```

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

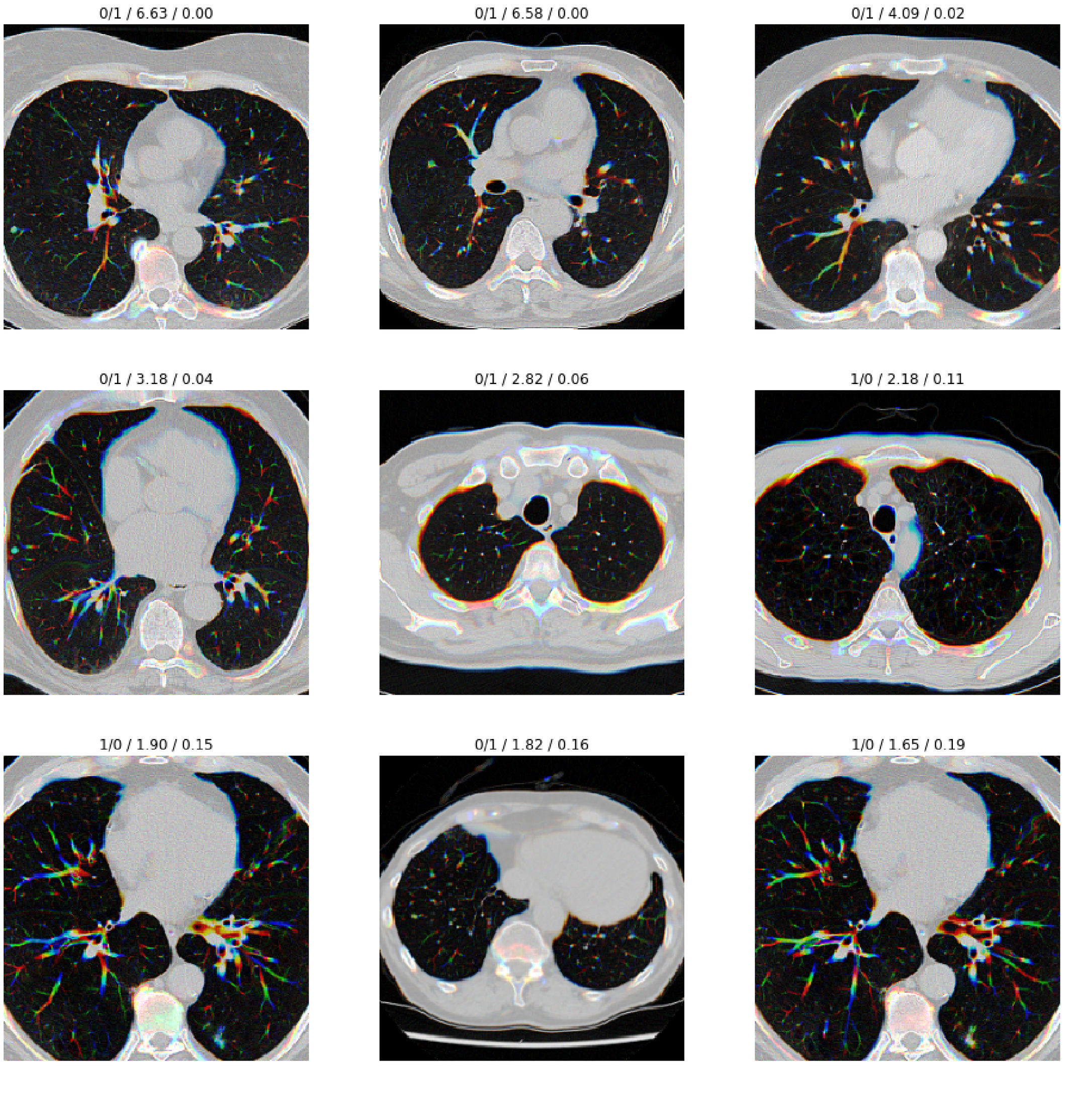


In [44]:
```
learner.unfreeze()
learner.fit_one_cycle(5, max_lr=1e-5)
```

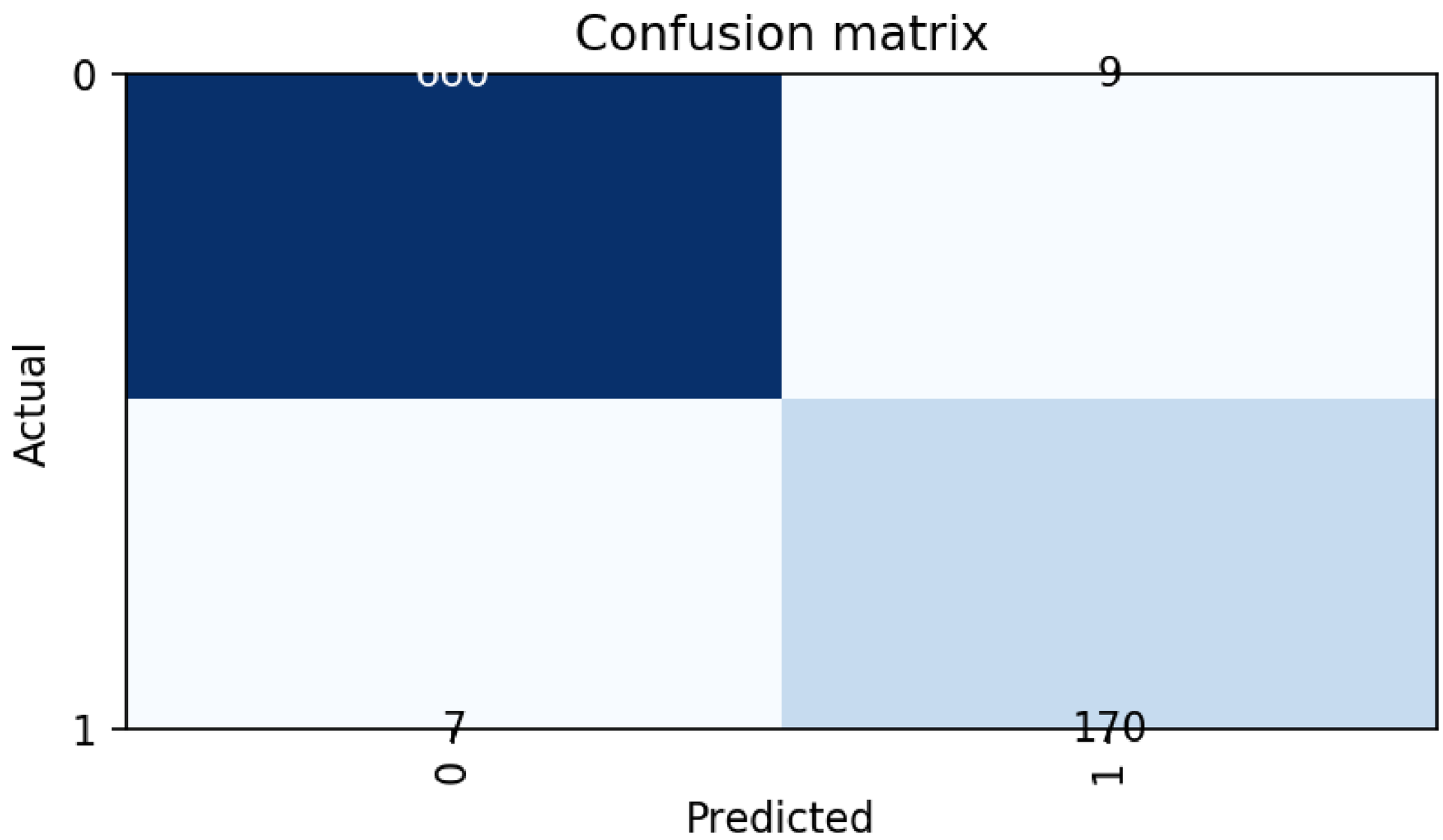
epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.045264	0.067641	0.018913	0.991656	00:17
1	0.041613	0.067668	0.023641	0.991711	00:18
2	0.039806	0.070195	0.018913	0.990972	00:18
3	0.043492	0.067287	0.020095	0.989942	00:18
4	0.026908	0.068629	0.020095	0.990178	00:18



In [51]:
```
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))
```

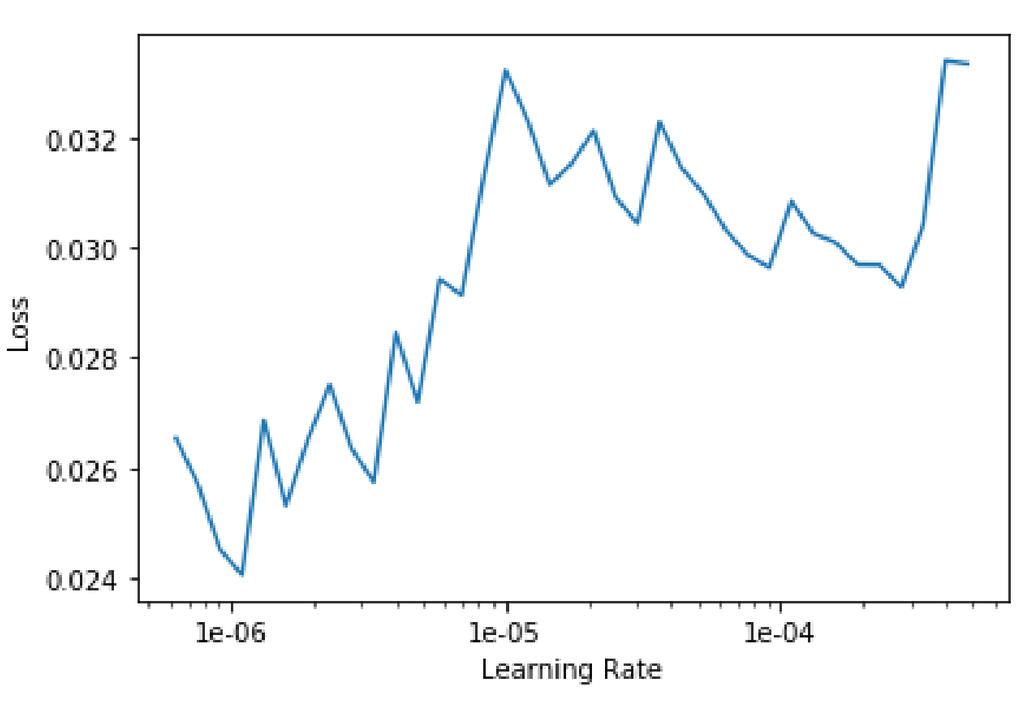


In [52]:
```
interp.plot_confusion_matrix(figsize=(5,5), dpi=160)
```



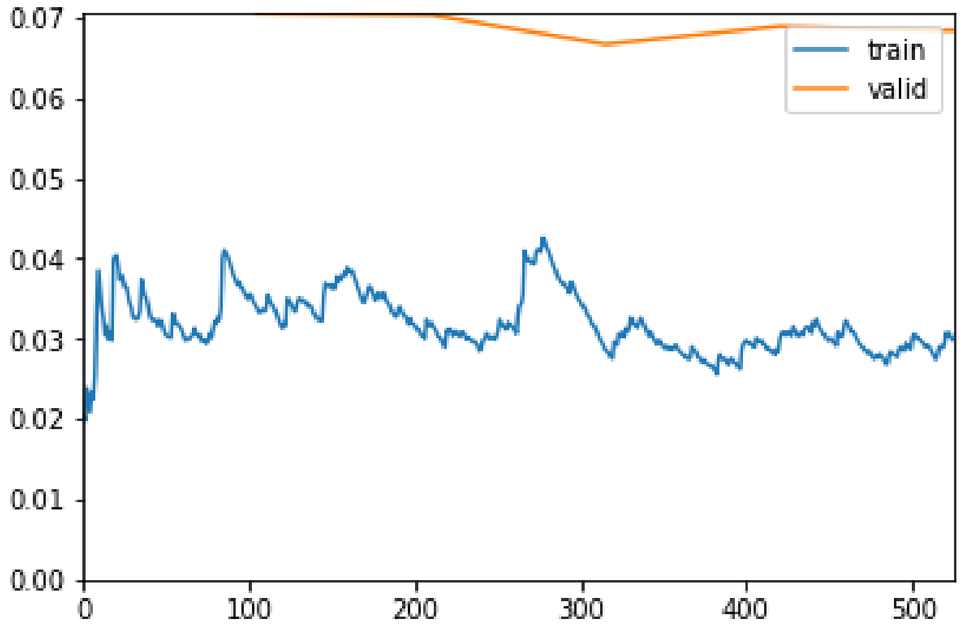
In [49]:
```
learner.lr_find()
learner.recorder.plot()
```

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



In [50]:
```
learner.unfreeze()
learner.fit_one_cycle(5, max_lr=1e-6)
```

epoch	train_loss	valid_loss	error_rate	auroc	time
0	0.034001	0.070497	0.023641	0.990153	00:18
1	0.031622	0.070329	0.020095	0.990288	00:18
2	0.029037	0.066694	0.021277	0.990111	00:18
3	0.028435	0.068945	0.024823	0.990626	00:18
4	0.029930	0.068407	0.018913	0.990639	00:17



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

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

In [53]:
```
learner.export('vgg16-25epc-rescale-crop256-unbalanced')
```