Ship Classification Analytics Vidhya

August 27, 2019

[2]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪the files in the input directory
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))



# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/sample_submission_ns2btKE.csv
/kaggle/input/test_ApKoW4T.csv
/kaggle/input/train/train.csv
/kaggle/input/train/images/2853965.jpg
/kaggle/input/train/images/2902584.jpg
/kaggle/input/train/images/1188084.jpg
/kaggle/input/train/images/2851832.jpg
/kaggle/input/train/images/2449686.jpg
/kaggle/input/train/images/2804747.jpg
/kaggle/input/train/images/2858880.jpg
/kaggle/input/train/images/2870078.jpg
/kaggle/input/train/images/2798417.jpg
/kaggle/input/train/images/2812927.jpg
/kaggle/input/train/images/2800530.jpg
/kaggle/input/train/images/2863936.jpg
/kaggle/input/train/images/1419049.jpg
/kaggle/input/train/images/2878219.jpg
```

```
/kaggle/input/train/images/2798491.jpg
/kaggle/input/train/images/2819072.jpg
/kaggle/input/train/images/2824360.jpg
/kaggle/input/train/images/2635776.jpg
/kaggle/input/train/images/2854021.jpg
/kaggle/input/train/images/2840647.jpg
/kaggle/input/train/images/2725933.jpg
/kaggle/input/train/images/2847715.jpg
/kaggle/input/train/images/1241847.jpg
/kaggle/input/train/images/2796402.jpg
/kaggle/input/train/images/2888413.jpg
/kaggle/input/train/images/2782247.jpg
/kaggle/input/train/images/986137.jpg
/kaggle/input/train/images/2902574.jpg
/kaggle/input/train/images/1196296.jpg
/kaggle/input/train/images/2841141.jpg
/kaggle/input/train/images/2900682.jpg
/kaggle/input/train/images/2851176.jpg
/kaggle/input/train/images/2878228.jpg
/kaggle/input/train/images/2854310.jpg
/kaggle/input/train/images/1555119.jpg
/kaggle/input/train/images/2175708.jpg
/kaggle/input/train/images/2860258.jpg
/kaggle/input/train/images/2884444.jpg
/kaggle/input/train/images/2868017.jpg
/kaggle/input/train/images/2843615.jpg
/kaggle/input/train/images/1296498.jpg
/kaggle/input/train/images/702094.jpg
/kaggle/input/train/images/2874079.jpg
/kaggle/input/train/images/2841706.jpg
/kaggle/input/train/images/2829243.jpg
/kaggle/input/train/images/2005205.jpg
/kaggle/input/train/images/2893731.jpg
/kaggle/input/train/images/1672063.jpg
/kaggle/input/train/images/2798493.jpg
/kaggle/input/train/images/2806751.jpg
/kaggle/input/train/images/2814819.jpg
/kaggle/input/train/images/2902693.jpg
```

[44]:
```
from fastai.vision import *
import matplotlib.pyplot as plt
tfms = get_transforms(max_rotate=.0, max_zoom=.1,max_lighting=0.05, max_warp=0.)
```

[4]:
```
!cp -r /kaggle/input/train/ /kaggle/TRAIN/
```

[5]:
```
!mkdir /kaggle/TEST/
!cp /kaggle/input/test_ApKoW4T.csv /kaggle/TEST/test.csv
```

[7]:
```
!ls /kaggle/TRAIN/
```

```
images  train.csv
```

[8]:
```
from pathlib import Path
tn_path = Path('/kaggle/TRAIN/')
```

[9]:
```
train_lbl=f'{tn_path}/train.csv'
```

[10]:
```
ts_path = Path('/kaggle/TEST/')
test_lbl = f'{ts_path}/test.csv'
```

[11]:
```
t= pd.read_csv(test_lbl)
tt = pd.read_csv(train_lbl)
```

[12]:
```
paths_to_copy = []
for i in t['image']:
    paths_to_copy.append(str(tn_path)+'/images/'+str(i))
```

[13]:
```
import subprocess
for i in paths_to_copy:
    subprocess.call(['cp {} /kaggle/TEST'.format(i)],shell=True)
```

[14]:
```
!ls /kaggle/TEST | wc -l
```

```
2681
```

[15]:
```
!mkdir /kaggle/TEST/images
!cp /kaggle/TEST/*.jpg /kaggle/TEST/images
```

[16]:
```
!rm /kaggle/TEST/*.jpg
```

[17]:
```
!ls /kaggle/TEST/images | wc -l
```

```
2680
```

[18]:
```
len(paths_to_copy)
```

[18]:
```
2680
```

[19]:
```
t.describe()
```

[19]:

	image
count	2680
unique	2680
top	2732660.jpg
freq	1

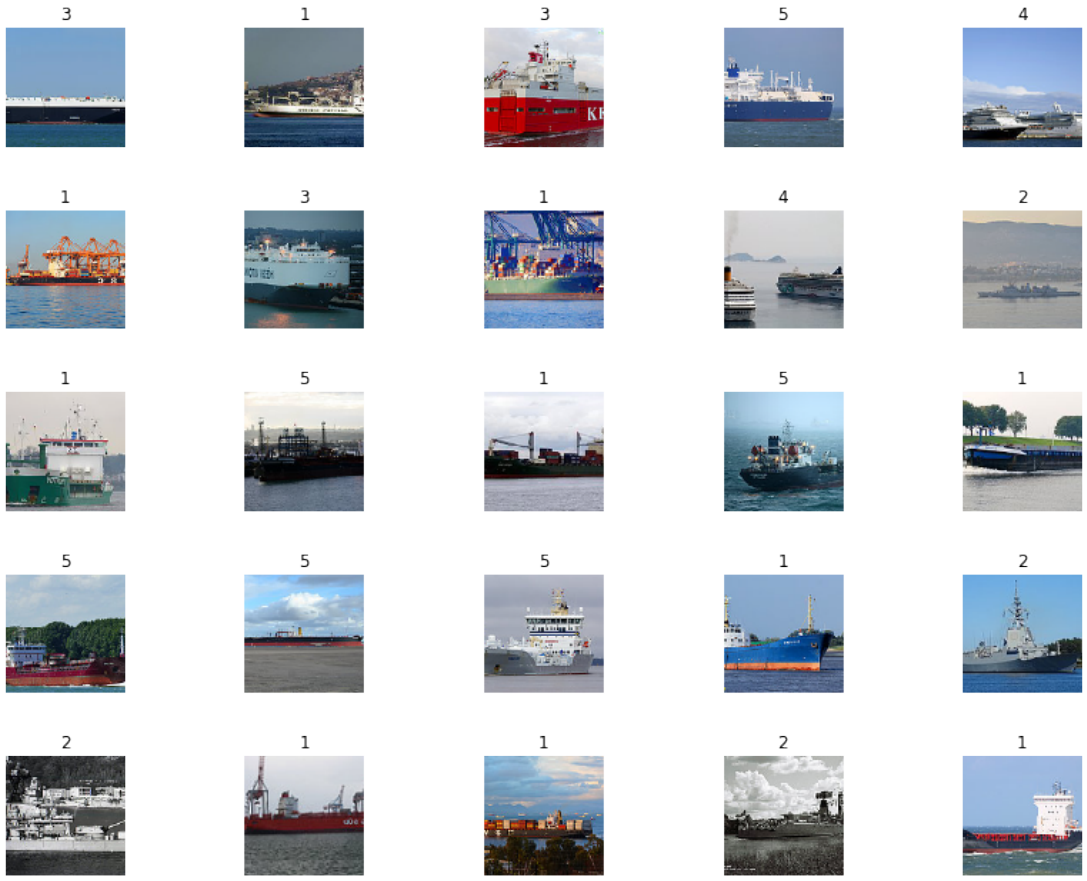
[45]:
```
data = ImageDataBunch.from_csv(tn_path,csv_labels=train_lbl,folder='images',␣
↪ds_tfms=tfms, size=96,bs=64);
stats=data.batch_stats()
data.normalize(stats)
```

[45]:
```
ImageDataBunch;

Train: LabelList (5002 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
1,2,3,2,4
Path: /kaggle/TRAIN;

Valid: LabelList (1250 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
2,1,5,1,4
Path: /kaggle/TRAIN;

Test: None
```

[47]:
```
from torchvision.models import *
arch = resnet34
acc_02 = partial(accuracy_thresh, thresh=0.2)
acc_03 = partial(accuracy_thresh, thresh=0.3)
acc_04 = partial(accuracy_thresh, thresh=0.4)
acc_05 = partial(accuracy_thresh, thresh=0.5)
f_score = partial(fbeta, thresh=0.2)
learn = cnn_learner(data, arch, metrics=[accuracy, FBeta('macro')])
```

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to
/tmp/.cache/torch/checkpoints/resnet34-333f7ec4.pth
100%|      | 83.3M/83.3M [00:02<00:00, 32.0MB/s]
```

[48]:
```
ts_path = Path('/kaggle/TEST/images')
test_imgs = ts_path.ls()
test_imgs.sort(key=lambda x: x.stem)
data.add_test(test_imgs)
learn.data = data
preds = learn.get_preds(ds_type=DatasetType.Test)
data.show_batch(rows=5, figsize=(12,9))
```

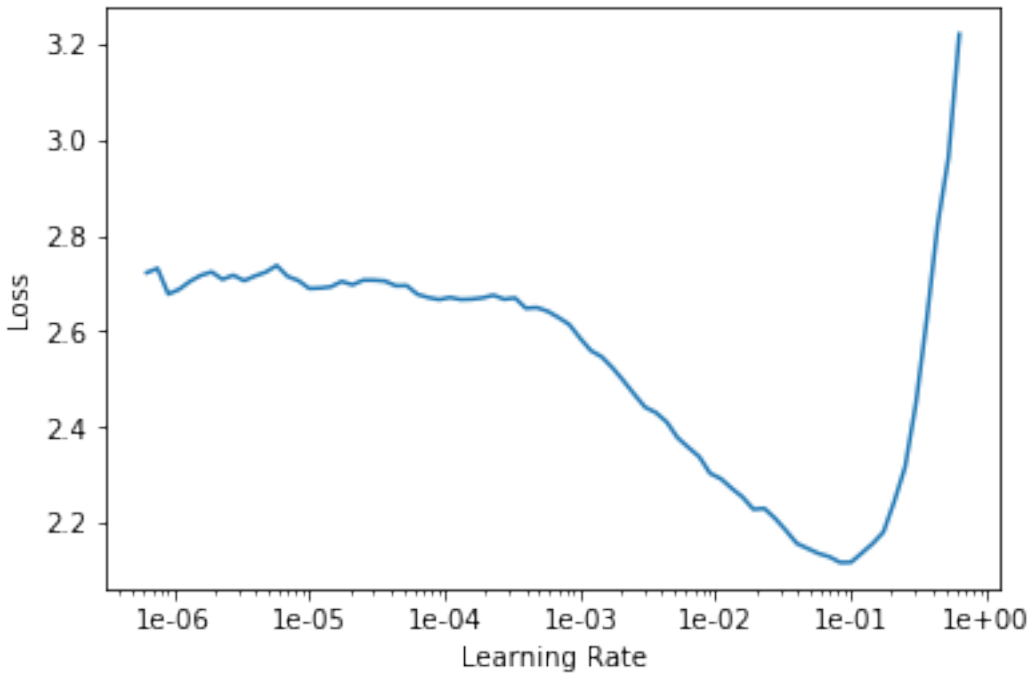


[49]:
```
learn.lr_find()
```

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

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

[50]:
```
learn.recorder.plot()
```



[51]:
```
lr = 1e-4
```

[52]:
```
learn.fit_one_cycle(4, slice(lr))
```

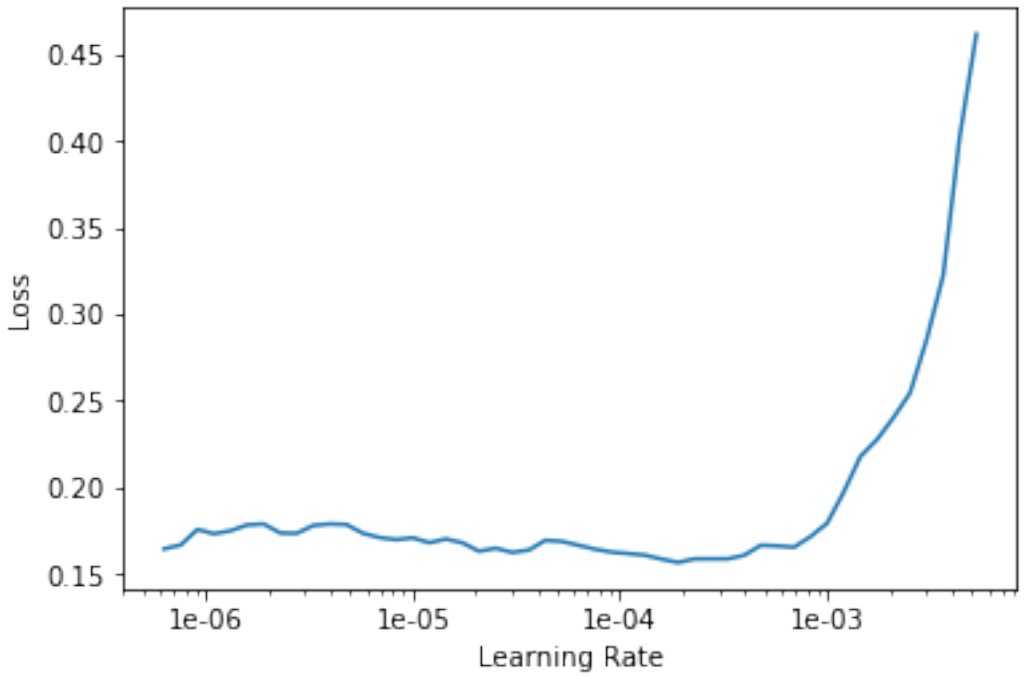
```
<IPython.core.display.HTML object>
```

[53]:
```
learn.save('stage1')
```

[40]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

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

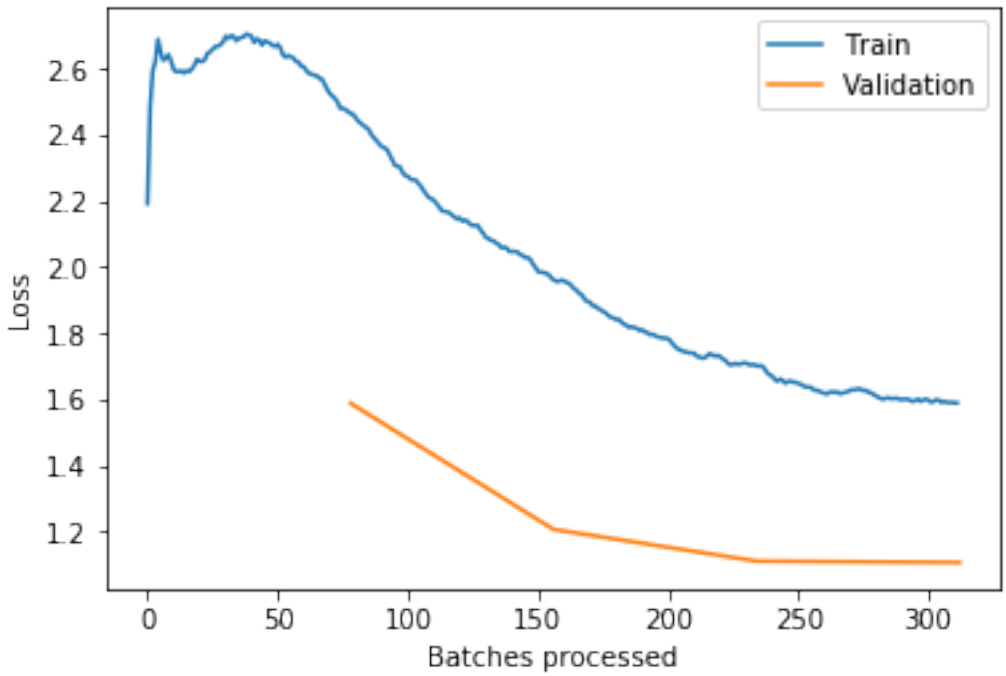
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



[41]:
```
learn.fit_one_cycle(5, slice(1e-4, lr/5))
```

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

[54]:
```
learn.save('stage2')
learn.recorder.plot_losses()
```



[55]:
```
pred_test,y_test = learn.get_preds(DatasetType.Test)
pred_score = accuracy(pred_test,y_test)
pred_test_tta,y_test_tta = learn.TTA(ds_type=DatasetType.Test)
pred_score_tta = accuracy(pred_test_tta,y_test_tta)
pred_test_tta.shape
```

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

[55]:
```
torch.Size([2680, 5])
```

[56]:
```
y_test_tta.shape
```

[56]:
```
torch.Size([2680])
```

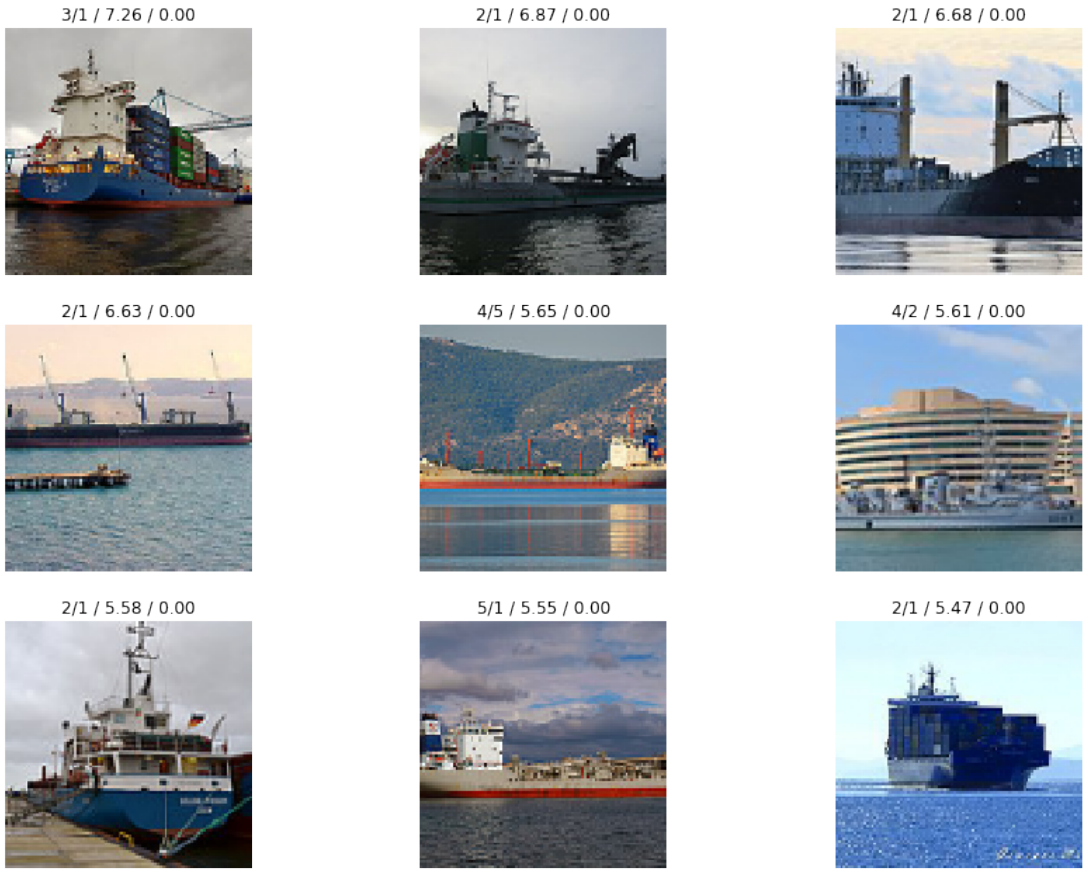
[57]:
```
clean_fname=np.vectorize(lambda fname: str(fname).split('/')[-1])
fname_cleaned=clean_fname(data.test_ds.items)
fname_cleaned=fname_cleaned.astype(str)
```

[58]:
```
data
```

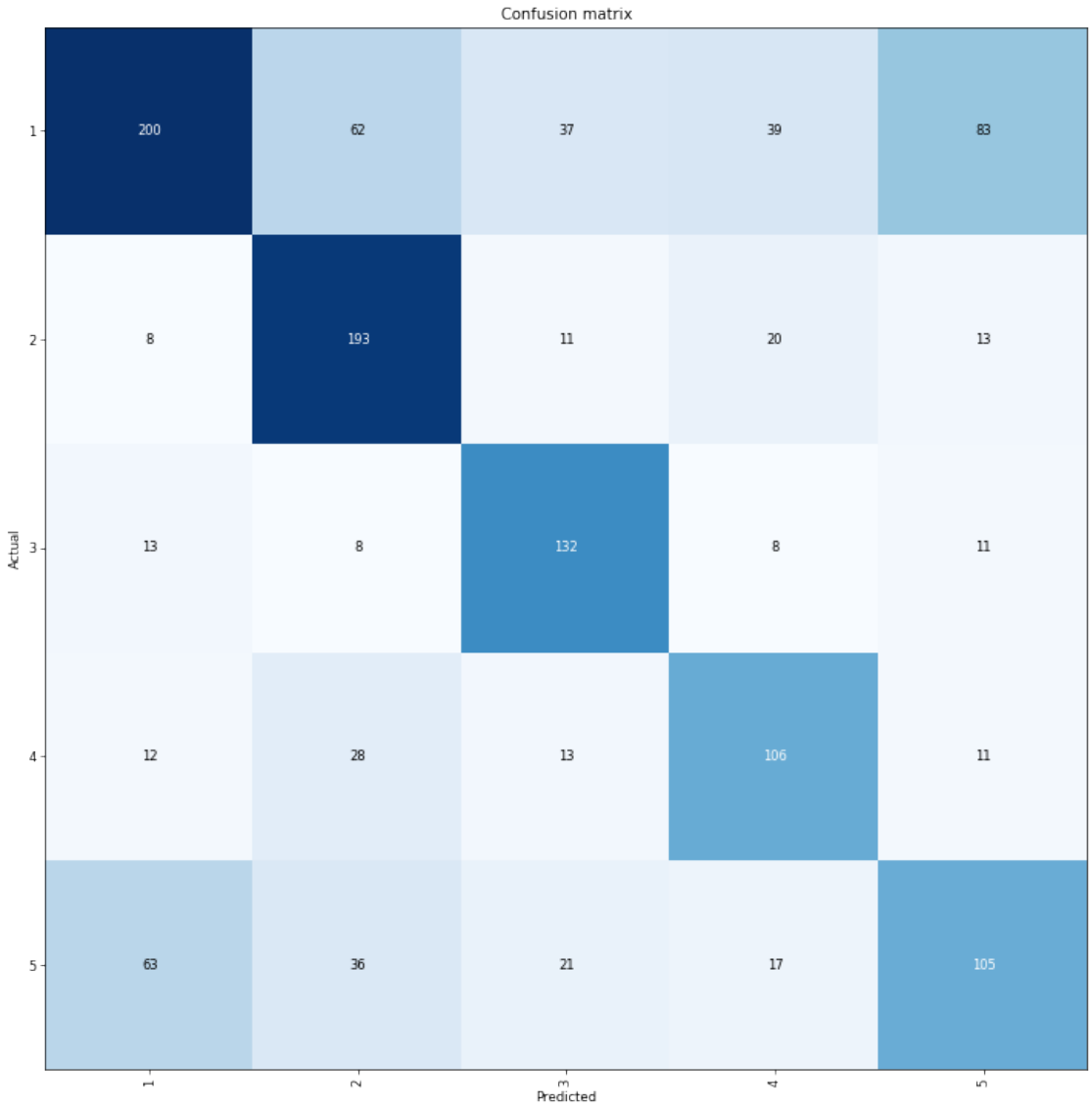
[58]:
```
ImageDataBunch;

Train: LabelList (5002 items)
x: ImageList
```

```
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
1,2,3,2,4
Path: /kaggle/TRAIN;

Valid: LabelList (1250 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: CategoryList
2,1,5,1,4
Path: /kaggle/TRAIN;

Test: LabelList (2680 items)
x: ImageList
Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image (3, 96, 96),Image
(3, 96, 96)
y: EmptyLabelList
,,,,
Path: /kaggle/TRAIN
```

[59]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_top_losses(9, figsize=(15,11))
```



[60]:
```
interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```



[61]:
```
interp.most_confused(min_val=2)
```

[61]:
```
[(1, 5, 83),
 (5, 1, 63),
 (1, 2, 62),
 (1, 4, 39),
 (1, 3, 37),
 (5, 2, 36),
 (4, 2, 28),
 (5, 3, 21),
 (2, 4, 20),
 (5, 4, 17),
```

```
 (2, 5, 13),
 (3, 1, 13),
 (4, 3, 13),
 (4, 1, 12),
 (2, 3, 11),
 (3, 5, 11),
 (4, 5, 11),
 (2, 1, 8),
 (3, 2, 8),
 (3, 4, 8)]
```

[62]:
```
thresh = 0.2
labelled_preds = [' '.join([str(learn.data.classes[i]) for i,p in␣
 ↪enumerate(pred) if p > thresh]) for pred in pred_test]
fnames = [f.name[:-4] for f in learn.data.test_ds.items]
```

[66]:
```
sub=pd.read_csv('/kaggle/input/sample_submission_ns2btKE.csv').
 ↪set_index('image')
sub.head()
```

[66]:

image	category
1007700.jpg	1
1011369.jpg	1
1051155.jpg	1
1062001.jpg	1
1069397.jpg	1

[67]:
```
sub.loc[fname_cleaned,'label']=to_np(pred_test[:,1])
sub.to_csv(f'submission_{pred_score}.csv')
sub.loc[fname_cleaned,'label']=to_np(pred_test_tta[:,1])
sub.to_csv(f'submission_{pred_score_tta}.csv')
```

```
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
```