UC-Merced

September 4, 2019

[1]:
```
# 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␣
 ↪all files under the input directory


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

[23]:
```
"""!cp -r /kaggle/input/uc-merced-land-use/UCMerced_LandUse/ /kaggle/ucm/

!mkdir /kaggle/ucm/valid/
!mkdir /kaggle/ucm/test/
!mv /kaggle/ucm/Images/ /kaggle/ucm/train/


import os
for dirname, _, filenames in os.walk('/kaggle/ucm'):
    for filename in filenames:
        print(os.path.join(dirname, filename))"""
```

[23]:
```
"!cp -r /kaggle/input/uc-merced-land-use/UCMerced_LandUse/
/kaggle/ucm/\n\n!mkdir /kaggle/ucm/valid/\n!mkdir /kaggle/ucm/test/\n!mv
/kaggle/ucm/Images/ /kaggle/ucm/train/\n\n\nimport os\nfor dirname, _, filenames
in os.walk('/kaggle/ucm'):\n    for filename in filenames:\n
print(os.path.join(dirname, filename))"
```

[5]:
```
from pathlib import Path
tn_path = Path('/kaggle/ucm/train/')
val_path = Path('/kaggle/ucm/valid/')
```

```
ts_path = Path('/kaggle/ucm/test/')
```

[6]:
```
classes  = os.listdir(str(tn_path))
print(type(classes))
```

```
<class 'list'>
```

[7]:
```
from fastai.vision import *
from collections import Counter
import matplotlib.pyplot as plt
tfms = get_transforms(do_flip=True, flip_vert=True, max_rotate=.0, max_zoom=.
 ↪1,max_lighting=0.05, max_warp=0.)
```

[8]:
```
#data = ImageDataBunch.from_folder(path=Path('/kaggle/ucm/
 ↪'),train=tn_path,valid = val_path,test=ts_path,valid_pct=0.
 ↪2,classes=Counter(classes), ds_tfms=tfms, size=128);
data = (ImageList.from_folder(Path('/kaggle/ucm/'))
        .split_by_rand_pct()
        .label_from_folder()
        .transform(tfms, size=224)
        .databunch())
stats=data.batch_stats()
data.normalize(stats)
```

[8]:
```
ImageDataBunch;

Train: LabelList (1680 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
denseresidential,denseresidential,denseresidential,denseresidential,denseresiden
tial
Path: /kaggle/ucm;

Valid: LabelList (420 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224,
224),Image (3, 224, 224)
y: CategoryList
mobilehomepark,tenniscourt,mediumresidential,sparseresidential,beach
Path: /kaggle/ucm;

Test: None
```

[10]:
```
data.show_batch(rows=5)
```

[11]:
```
from torchvision.models import *
arch = densenet121
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/densenet121-a639ec97.pth" to
/tmp/.cache/torch/checkpoints/densenet121-a639ec97.pth
100%|      | 30.8M/30.8M [00:00<00:00, 135MB/s]
```

[12]:
```
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()
```

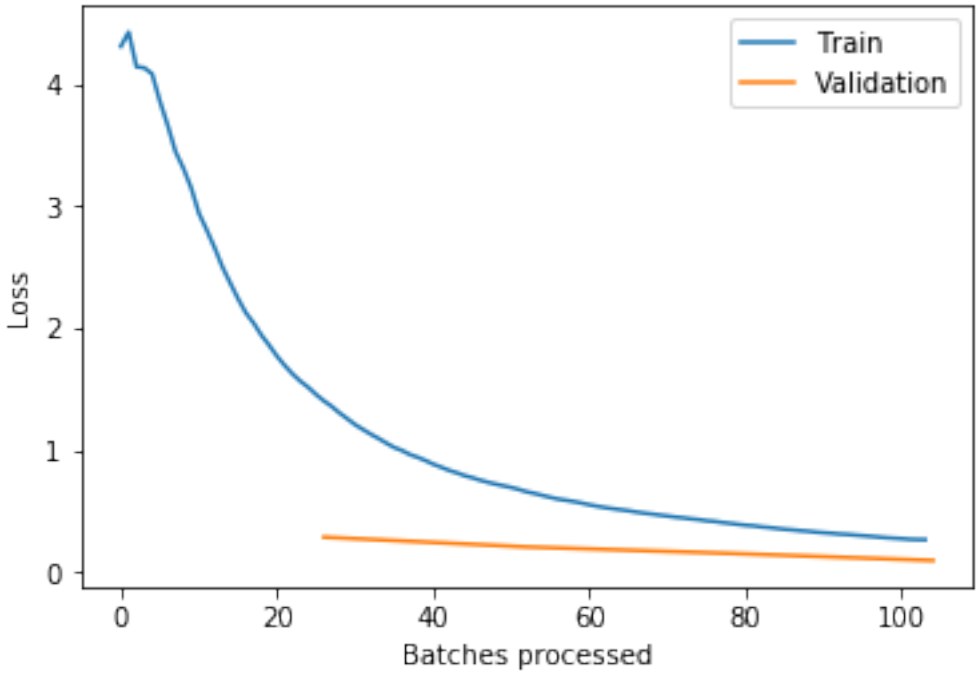
[13]:
```
lr = 7e-3
learn.fit_one_cycle(4, slice(lr))
```

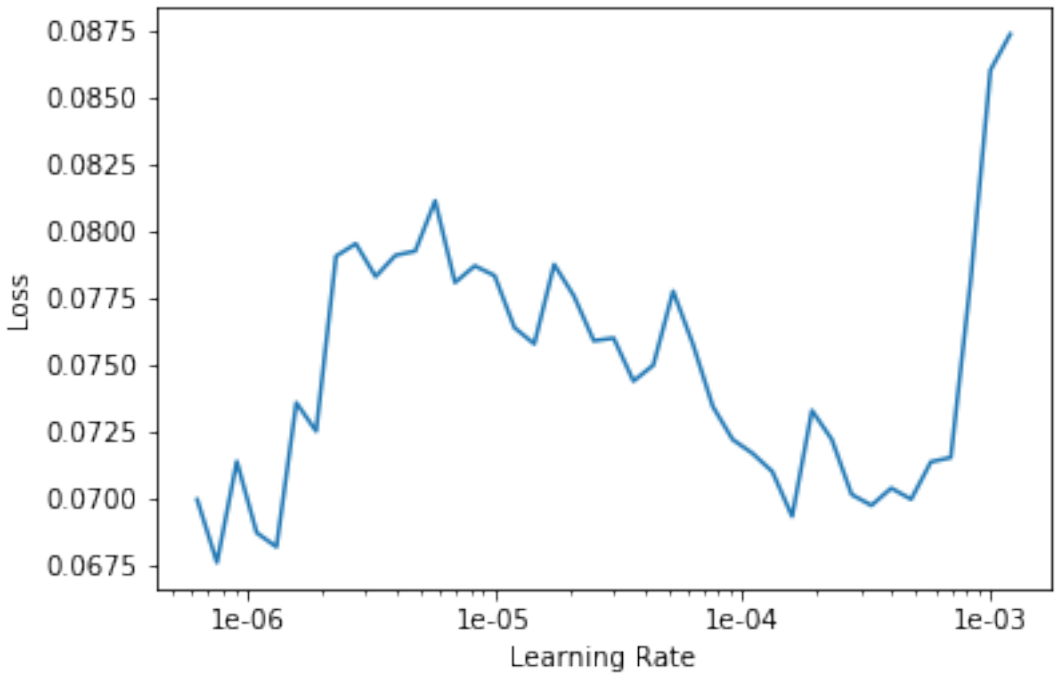
```
<IPython.core.display.HTML object>
```

[14]:
```
learn.save('stage1')
learn.recorder.plot_losses()
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.
```

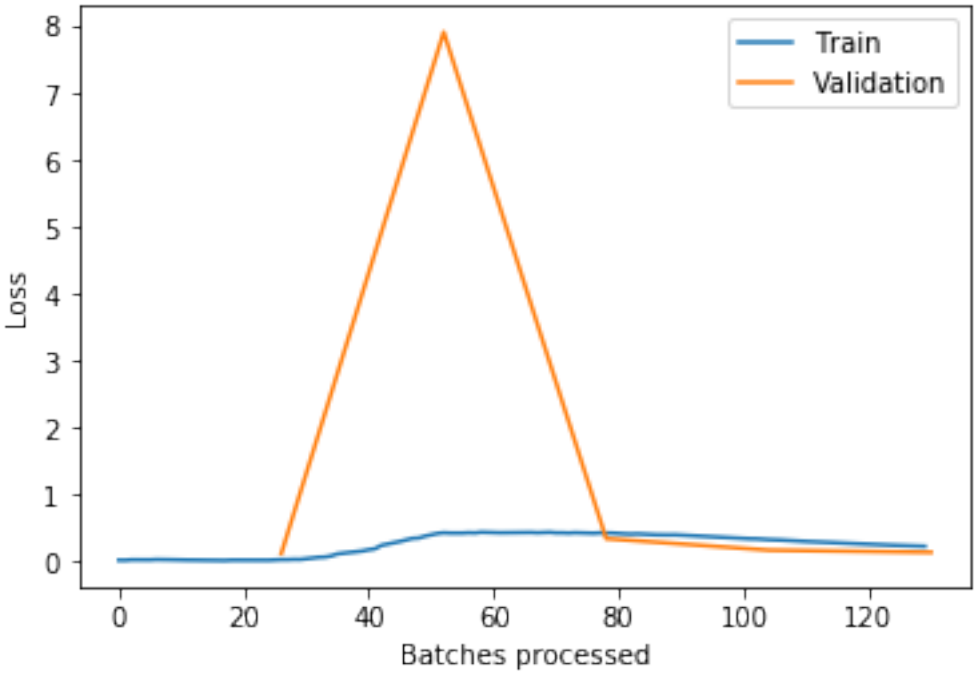




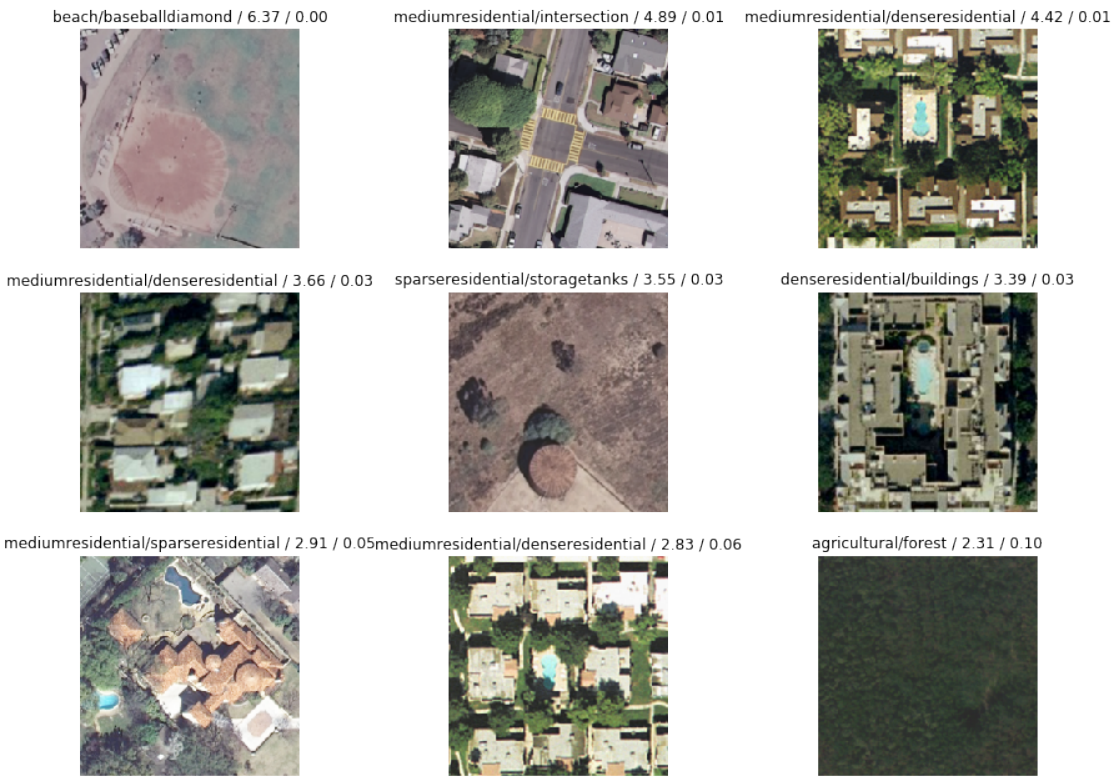
[19]:
```
learn.fit_one_cycle(5, slice(8e-4,lr/5))
```

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

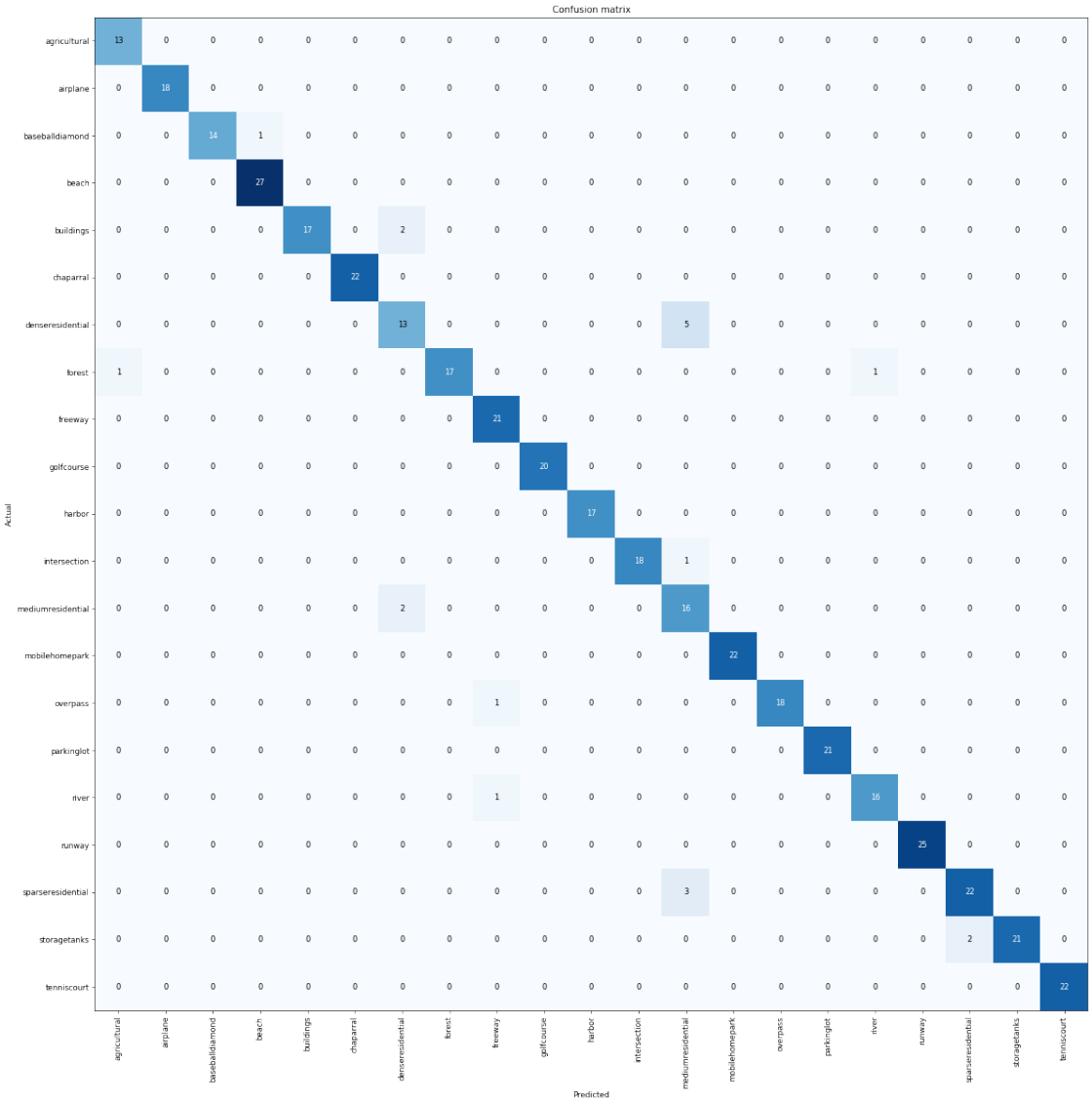
[20]:
```
learn.save('stage2')
learn.recorder.plot_losses()
```



[21]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_top_losses(9, figsize=(15,11))
```



[22]:
```
interp.plot_confusion_matrix(figsize=(20,20), dpi=60)
```


Confusion matrix
Actual
Predicted
agricultural
airplane
baseballdiamond
beach
buildings
chaparral
denseresidential
forest
freeway
golfcourse
harbor
intersection
mediumresidential
mobilehomepark
overpass
parkinglot
river
runway
sparseresidential
storagetanks
tenniscourt