The notebook shows the classification of photos into the 6 classe as defined by the competition.

Link to competition page https://datahack.analyticsvidhya.com/contest/practice-problem-intel-scene-classification-challe/ (https://datahack.analyticsvidhya.com/contest/practice-problem-intel-scene-classification-challe/)

In [0]:
```
#Import packages
from fastai.vision import *
import numpy as np
import pandas as pd
```

In [0]:
```
#Put at beginning of every notebook to map Google drive to Colab
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
root_dir = "/content/gdrive/My Drive/"
base_dir = root_dir + 'fastai-v3/'
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client
_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&
redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2F
www.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fau
th%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=
code

Enter your authorization code:
··········
Mounted at /content/gdrive
```

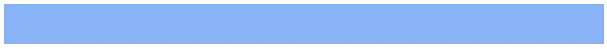
Download pictures and upload into appropriate directories in Google Drive

In [0]:
```
labels = ['0','1','2','3','4','5']
path = Path(base_dir + 'data/intel')
```

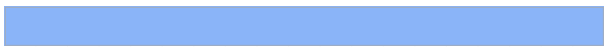
Verify images and delete bad ones

In [0]:
```
classes = labels
for c in classes:
    print(c)
    verify_images(path/c, delete=True, max_size=500)
```

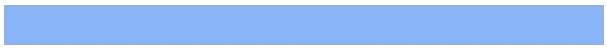
```
0
```



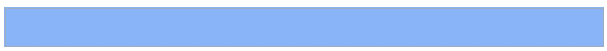
```
100.00% [2623/2623 00:09<00:00]
1
```



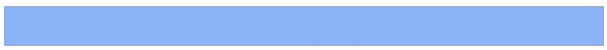
```
100.00% [2745/2745 00:10<00:00]
2
```



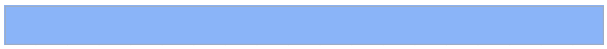
```
100.00% [2957/2957 00:11<00:00]
3
```



```
100.00% [3037/3037 00:11<00:00]
4
```



```
100.00% [2784/2784 00:11<00:00]
5
```



```
100.00% [2883/2883 02:48<00:00]
```

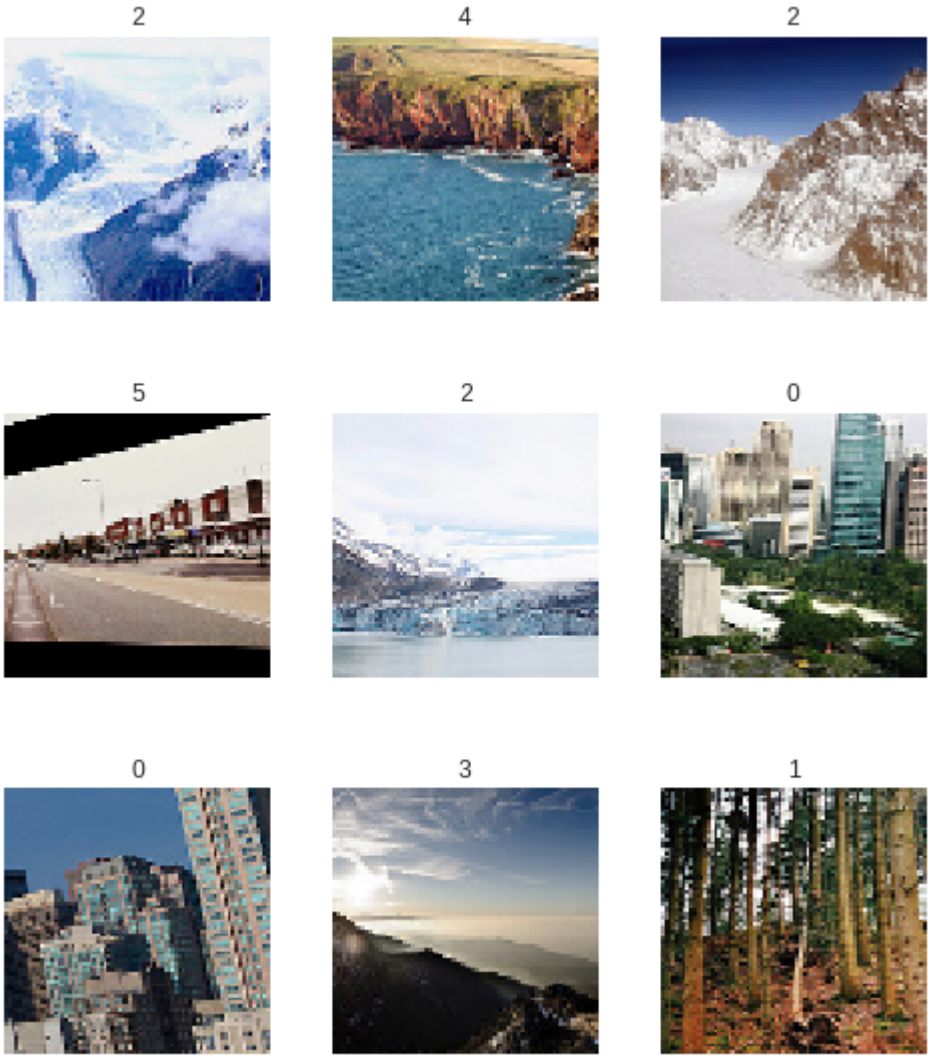
Create data object

In [0]:
```
np.random.seed(123)
#data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get
_transforms(do_flip=True, flip_vert=False, max_rotate=0.0, max_zoom=0.0, max_l
ighting=0.0,max_warp=0.0), size=150, num_workers=4).normalize(imagenet_stats)
data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_
transforms(), size=150, num_workers=4).normalize(imagenet_stats)
```

In [0]:
```
data.classes
```

Out[0]:
```
['0', '1', '2', '3', '4', '5']
```

In [0]:
```
data.show_batch(rows=3, figsize=(7,8))
```



Train the model using pre-trained model resnet34 and fit model

In [0]:
```
learn = create_cnn(data, models.resnet50, metrics=error_rate)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /
root/.torch/models/resnet50-19c8e357.pth
100%|██████████| 102502400/102502400 [00:01<00:00, 97631646.24it/s]
```

In [0]:
```
#Run twice. First in the first round and then after the setting the first LR.
 Saved as Stage-1
learn.fit_one_cycle(4)
```

Total time: 09:30

epoch	train_loss	valid_loss	error_rate
1	0.372148	0.303716	0.105140
2	0.287590	0.229756	0.080470
3	0.234536	0.205784	0.074302
4	0.207465	0.191289	0.067841

In [0]:
```
#Saved after second round
learn.save('stage-1_2_057669')
```

In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

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

In [0]:
```
#Run twice.  First run using LRs 3e-6 and 7e-6 and epoch 4.  Second set below.
learn.fit_one_cycle(10, max_lr=slice(3e-04))
```

Total time: 29:56

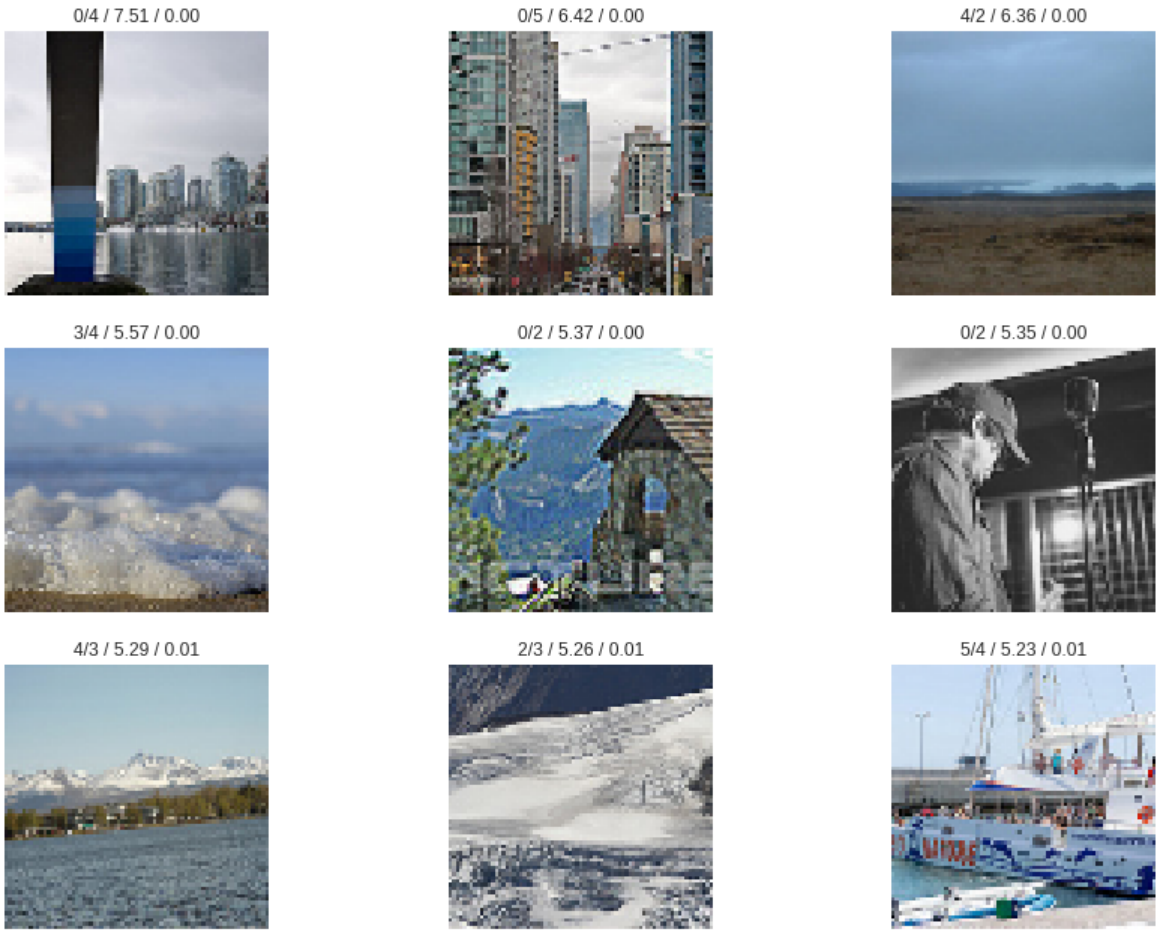
epoch	train_loss	valid_loss	error_rate
1	0.195388	0.188559	0.070485
2	0.187868	0.180619	0.067254
3	0.167311	0.175483	0.063436
4	0.130478	0.172025	0.060206
5	0.099417	0.184098	0.060206
6	0.080284	0.192827	0.062849
7	0.057502	0.194102	0.061380
8	0.041636	0.201459	0.062555
9	0.039781	0.198782	0.059618
10	0.031534	0.195404	0.059325

In [0]:
```
learn.save('stage-2')
```

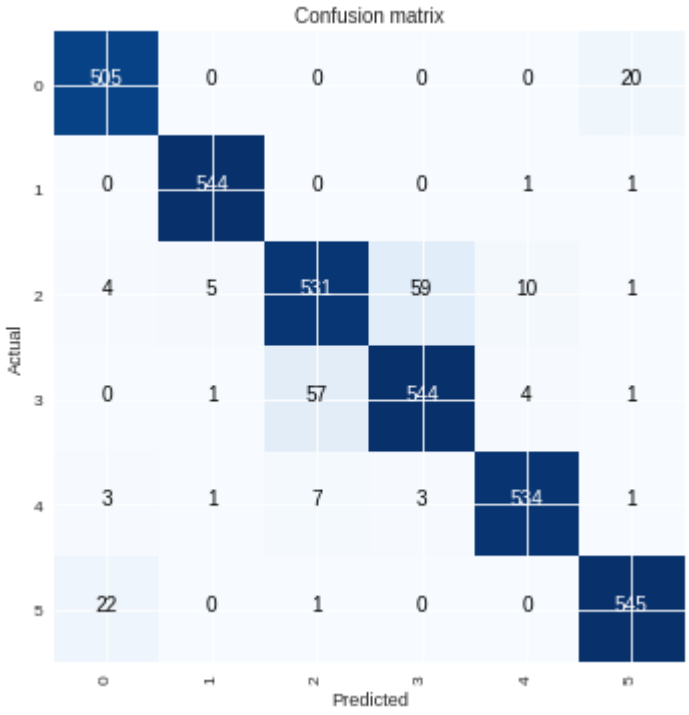
In [0]:
```
learn.load('stage-2')
```

In [0]:
```
interp = ClassificationInterpretation.from_learner(learn)
```

In [0]:
```
interp.plot_top_losses(9, figsize=(15,11))
```



In [0]:
```
interp.plot_confusion_matrix(figsize=(6,6), dpi=60)
```



In [0]:
```
{'buildings' -> 0,

'forest' -> 1,

'glacier' -> 2,

'mountain' -> 3,

'sea' -> 4,

'street' -> 5 }
```

In [0]:
```
#Second run
```

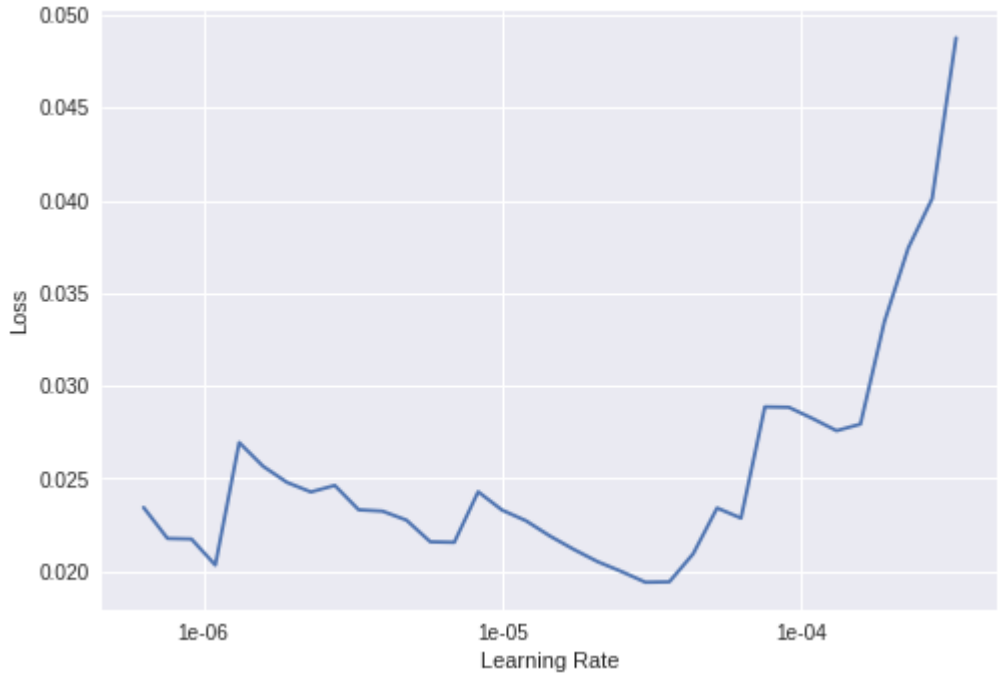
In [0]:
```
learn.fit_one_cycle(5)
```

Total time: 30:51

epoch	train_loss	valid_loss	error_rate
1	0.037090	0.266523	0.067548
2	0.064087	0.279576	0.064611
3	0.049562	0.279806	0.063730
4	0.031625	0.272229	0.062849
5	0.026172	0.275753	0.061087

In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

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



In [0]:
```
learn.fit_one_cycle(5, max_lr=slice(5e-05))
```

Total time: 14:22

epoch	train_loss	valid_loss	error_rate
1	0.030320	0.279645	0.061968
2	0.026739	0.280954	0.061674
3	0.028342	0.291730	0.061380
4	0.023287	0.279847	0.062555
5	0.025070	0.284250	0.059618

In [0]:
```
learn.export()
```

Do predictions from saved model

In [0]:
```
#Import exported model
learn = load_learner(path)
```

```
/usr/local/lib/python3.6/dist-packages/torch/serialization.py:435: SourceChan
geWarning: source code of class 'fastai.layers.Flatten' has changed. you can
retrieve the original source code by accessing the object's source attribute
or set `torch.nn.Module.dump_patches = True` and use the patch tool to revert
the changes.
  warnings.warn(msg, SourceChangeWarning)
```

In [0]:
```
import pandas as pd
test = pd.read_csv(path/'test_WyRytb0.csv')
```

In [0]:
```
images = []
prediction = []
probability = []
for i in test['image_name']:
  images.append(i)
  link = str(path) + '/test/' + i
  img = open_image(link)
  pred_class,pred_idx,outputs = learn.predict(img)
  prediction.append(pred_class.obj)
  probability.append(outputs.abs().max().item())

answer = pd.DataFrame({'image_name':images, 'label':prediction, 'probability':
probability})
```

In [0]:
```
answer.head()
```

Out[0]:

	image_name	label	probability
0	3.jpg	5	1.000000
1	5.jpg	0	0.999999
2	6.jpg	4	1.000000
3	11.jpg	2	0.999135
4	14.jpg	5	0.999989

In [0]:
```
answer.to_csv(path/'submission original.csv')
```

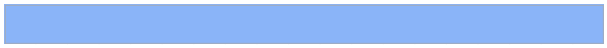
In [0]:
```

```

Using Cleaned data

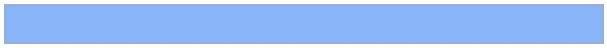
In [0]:
```
labels = ['0','1','2','3','4','5']
path = Path(base_dir + 'data/intel')
```

In [0]:
```
for c in labels:
    print(c)
    verify_images(path/c, delete=True, max_size=500)
```

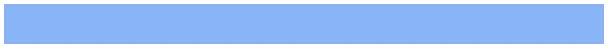
```
0
```



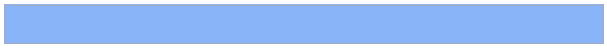
```
100.00% [2559/2559 00:10<00:00]
1
```



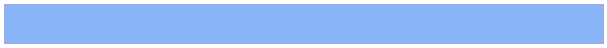
```
100.00% [2721/2721 00:11<00:00]
2
```



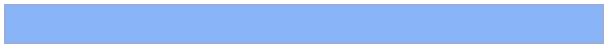
```
100.00% [2844/2844 00:11<00:00]
3
```



```
100.00% [2993/2993 00:12<00:00]
4
```



```
100.00% [2681/2681 00:10<00:00]
5
```



```
100.00% [2774/2774 05:26<00:00]
```

In [0]:
```
np.random.seed(21)
#version 1 with all default parameters, full size of image
data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_
transforms(), size=150, num_workers=4).normalize(imagenet_stats)
#version 2
#data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get
_transforms(do_flip=True, flip_vert=False, max_rotate=0.0, max_zoom=0.0, max_l
ighting=0.0,max_warp=0.0), size=150, num_workers=4).normalize(imagenet_stats)
```

In [0]:
```
data.classes
```

Out[0]:
```
['0', '1', '2', '3', '4', '5']
```

In [0]:
```
#Version1
#learn = create_cnn(data, models.resnet34, metrics=error_rate)
#version 2
learn = create_cnn(data, models.resnet50, metrics=error_rate)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /
root/.torch/models/resnet50-19c8e357.pth
100%|██████████| 102502400/102502400 [00:01<00:00, 81751156.66it/s]
```

In [0]:
```
#Basic first run
learn.fit_one_cycle(5)
```

Total time: 26:59

epoch	train_loss	valid_loss	error_rate
1	0.373825	0.272387	0.095655
2	0.288052	0.224510	0.076645
3	0.208732	0.178297	0.063368
4	0.182422	0.171626	0.059143
5	0.161896	0.170516	0.060350

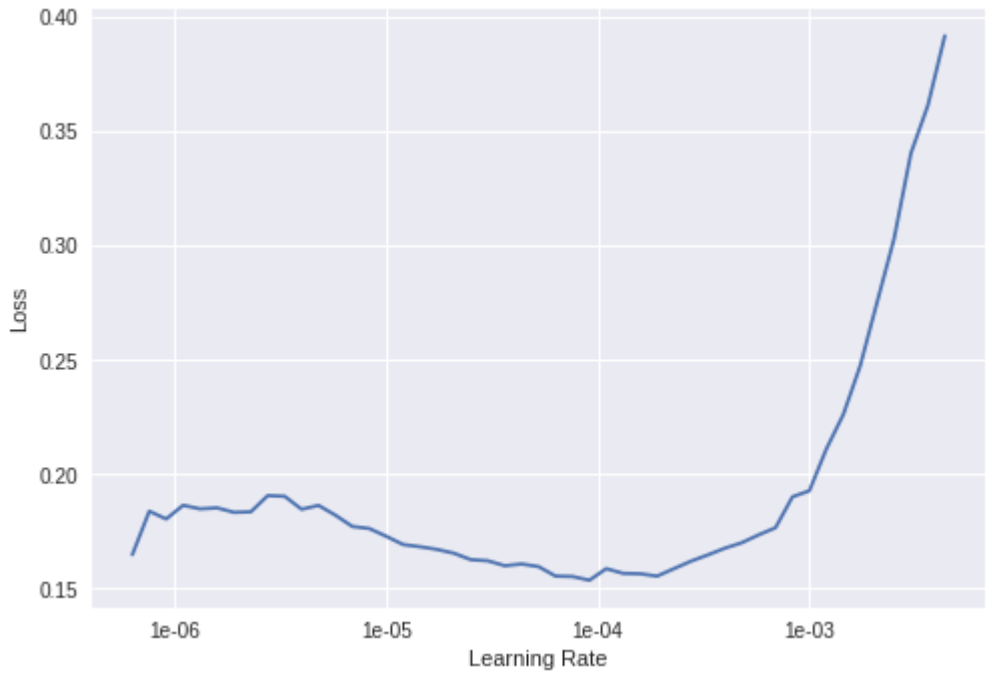
In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

0.00% [0/1 00:00<00:00]

epoch	train_loss	valid_loss	error_rate

Interrupted

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



In [0]:
```
learn.fit_one_cycle(10, max_lr=slice(7e-05))
```

Total time: 30:26

epoch	train_loss	valid_loss	error_rate
1	0.153341	0.168476	0.059445
2	0.150859	0.164064	0.059143
3	0.141444	0.164761	0.056427
4	0.131698	0.159697	0.058238
5	0.120742	0.156737	0.055522
6	0.097470	0.153611	0.052505
7	0.097339	0.158279	0.056126
8	0.093523	0.157430	0.054617
9	0.089451	0.155200	0.052806
10	0.087648	0.155845	0.054013

In [0]:
```

```

In [0]:
```
learn.save('stage-1')
```

In [0]:
```
#Second run
learn.fit_one_cycle(4)
```

Total time: 12:06

epoch	train_loss	valid_loss	error_rate
1	0.272003	0.281966	0.093241
2	0.234487	0.201482	0.071515
3	0.180427	0.175729	0.058238
4	0.119573	0.151686	0.049487

In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

0.00% [0/1 00:00<00:00]

epoch	train_loss	valid_loss	error_rate

Interrupted

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

In [0]:
```
learn.load('stage-2')
```

In [0]:
```
learn.fit_one_cycle(4, max_lr=slice(8e-06))
```

Total time: 12:07

epoch	train_loss	valid_loss	error_rate
1	0.100559	0.150867	0.049789
2	0.094074	0.148615	0.048280
3	0.096702	0.148911	0.047375
4	0.096529	0.149316	0.048280

In [0]:
```
learn.save('stage-2')
```

In [0]:
```
learn.export()
```

In [0]:
```
path
```

Out[0]:
```
PosixPath('/content/gdrive/My Drive/fastai-v3/data/intel')
```

Do predictions

In [0]:
```
#Import exported model
learn = load_learner(path)
```

```
/usr/local/lib/python3.6/dist-packages/torch/serialization.py:435: SourceChan
geWarning: source code of class 'fastai.layers.Flatten' has changed. you can
retrieve the original source code by accessing the object's source attribute
or set `torch.nn.Module.dump_patches = True` and use the patch tool to revert
the changes.
  warnings.warn(msg, SourceChangeWarning)
```

In [0]:
```
import pandas as pd
```

In [0]:
```
test = pd.read_csv(path/'test_WyRytb0.csv')
```

In [0]:
```
images = []
prediction = []
count = 1
probability = []
for i in test['image_name']:
  images.append(i)
  link = str(path) + '/test/' + i
  img = open_image(link)
  pred_class,pred_idx,outputs = learn.predict(img)
  prediction.append(pred_class.obj)
  probability.append(outputs.abs().max().item())
  print(count)
  count = count + 1

answer = pd.DataFrame({'image_name':images, 'label':prediction, 'probability':
probability})
```

In [0]:
```
answer.head()
```

Out[0]:

	image_name	label	probability
0	3.jpg	5	0.999787
1	5.jpg	0	0.999895
2	6.jpg	4	0.999799
3	11.jpg	2	0.934836
4	14.jpg	5	0.997912

In [0]:
```
answer.to_csv(path/'submission clean.csv')
```

Compare the clean and original submissions

In [0]:
```
clean = pd.read_csv(path/'submission clean.csv')
clean.columns = ['clean_index', 'clean_image_name', 'clean_label', 'clean_prob
ability']
```

In [0]:
```
original = pd.read_csv(path/'submission original.csv')
original.columns = ['original_index', 'original_image_name', 'original_label',
 'original_probability']
```

In [0]:
```
clean.shape
```

Out[0]:
```
(7301, 4)
```

In [0]:
```
original.shape
```

Out[0]:
```
(7301, 4)
```

In [0]:
```
clean_original = pd.concat([clean, original], axis=1)
```

In [0]:
```
clean_original.columns
```

Out[0]:
```
Index(['clean_index', 'clean_image_name', 'clean_label', 'clean_probability',
       'original_index', 'original_image_name', 'original_label',
       'original_probability'],
      dtype='object')
```

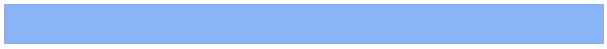
In [0]:
```
final_label = []
for index, row in clean_original.iterrows():
    if row['clean_probability'] > row['original_probability']:
        final_label.append(row['clean_label'])
    else:
        final_label.append(row['original_label'])
clean_original['final_label'] = final_label
```

In [0]:
```
clean_original.to_csv(path/'final submission.csv')
```

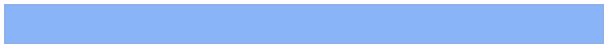
Use the entire dataset - train + test

In [0]:
```
classes = labels
for c in classes:
    print(c)
    verify_images(path/c, delete=True, max_size=500)
```

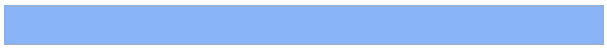
```
0
```



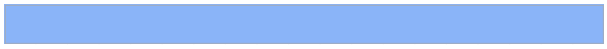
```
100.00% [3767/3767 00:16<00:00]
1
```



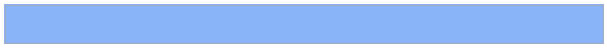
```
100.00% [3924/3924 00:17<00:00]
2
```



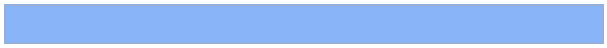
```
100.00% [4161/4161 00:18<00:00]
3
```



```
100.00% [4382/4382 00:19<00:00]
4
```



```
100.00% [3976/3976 00:17<00:00]
5
```



```
100.00% [4124/4124 03:57<00:00]
```

In [0]:
```
np.random.seed(21)
data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_
transforms(), size=150, num_workers=4).normalize(imagenet_stats)
data.classes
```

Out[0]:
```
['1', '2', '3', '4', '5']
```

In [0]:
```
learn = create_cnn(data, models.resnet34, metrics=error_rate)
```

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /
root/.torch/models/resnet34-333f7ec4.pth
87306240it [00:00, 92303889.04it/s]
```

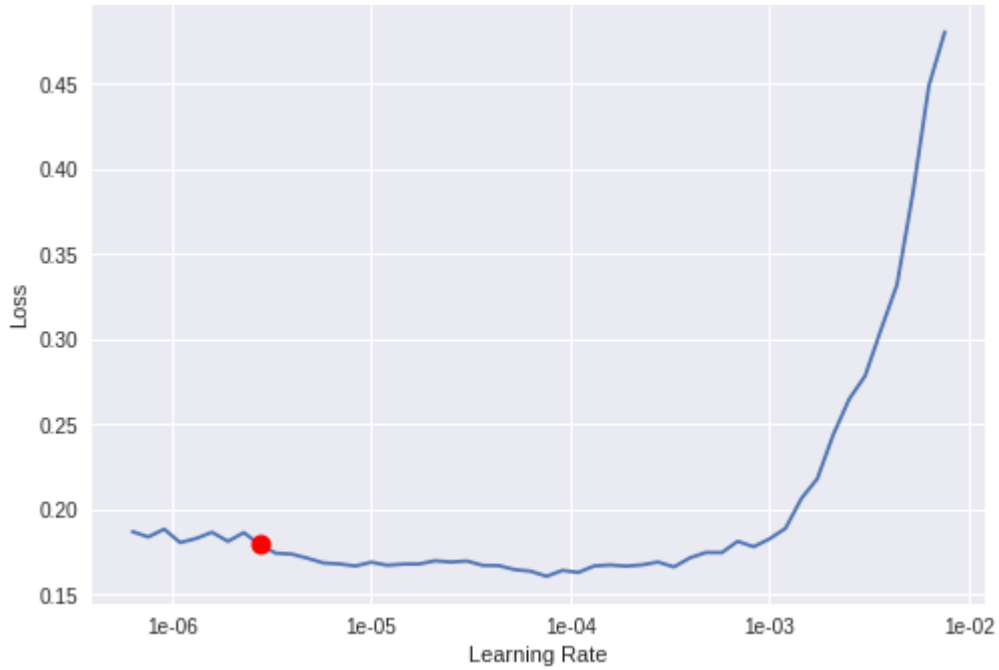
In [0]:
```
learn.fit_one_cycle(5)
```

Total time: 50:41

epoch	train_loss	valid_loss	error_rate
1	0.370209	0.253084	0.093606
2	0.253444	0.192867	0.066375
3	0.225514	0.175306	0.062242
4	0.170809	0.167288	0.058838
5	0.162577	0.164064	0.057622

In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
Min numerical gradient: 2.75E-06
```



In [0]:
```
learn.fit_one_cycle(5, max_lr=slice(2.75E-06))
```

Total time: 12:29

epoch	train_loss	valid_loss	error_rate
1	0.173309	0.164875	0.057622
2	0.169492	0.162990	0.056407
3	0.167039	0.166728	0.057622
4	0.156043	0.163976	0.057379
5	0.162284	0.161446	0.056650

In [0]:
```
learn.fit_one_cycle(5)
```

Total time: 12:30

epoch	train_loss	valid_loss	error_rate
1	0.212022	0.211561	0.070994
2	0.215170	0.204195	0.076100
3	0.185952	0.165888	0.057622
4	0.123204	0.138167	0.048626
5	0.077650	0.132117	0.044007

In [0]:
```
learn.save('all_data_044007')
```

In [0]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
Min numerical gradient: 1.32E-06
```

In [0]:
```
learn.fit_one_cycle(4, max_lr=slice(1.32E-06))
```

Total time: 10:02

epoch	train_loss	valid_loss	error_rate
1	0.070186	0.131861	0.045222
2	0.074217	0.130532	0.043277
3	0.067320	0.130728	0.043034
4	0.069246	0.129879	0.043521

In [0]:
```
learn.export()
```

In [0]:
```
#Import exported model
learn = load_learner(path)
```

In [0]:
```
test = pd.read_csv(path/'test_WyRytb0.csv')
```

In [0]:
```
images = []
prediction = []
probability = []
for i in test['image_name']:
  images.append(i)
  link = str(path) + '/test/' + i
  img = open_image(link)
  pred_class,pred_idx,outputs = learn.predict(img)
  prediction.append(pred_class.obj)
  probability.append(outputs.abs().max().item())

answer = pd.DataFrame({'image_name':images, 'label':prediction, 'probability':
probability})
```

In [0]:
```
answer.head()
```

Out[0]:

	image_name	label	probability
0	3.jpg	5	0.999980
1	5.jpg	4	0.928470
2	6.jpg	4	0.999883
3	11.jpg	2	0.979406
4	14.jpg	5	0.999978

In [0]:
```
answer.to_csv(path/'submission full resnet34.csv')
```

In [0]:
```

```