# intel_scene_classification

March 4, 2019

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/

```
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-

Enter your authorization code:
ůůůůůůůůůů
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


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

```
1
```

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

```
2
```

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

```
3
```

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

```
4
```

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

```
5
```

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

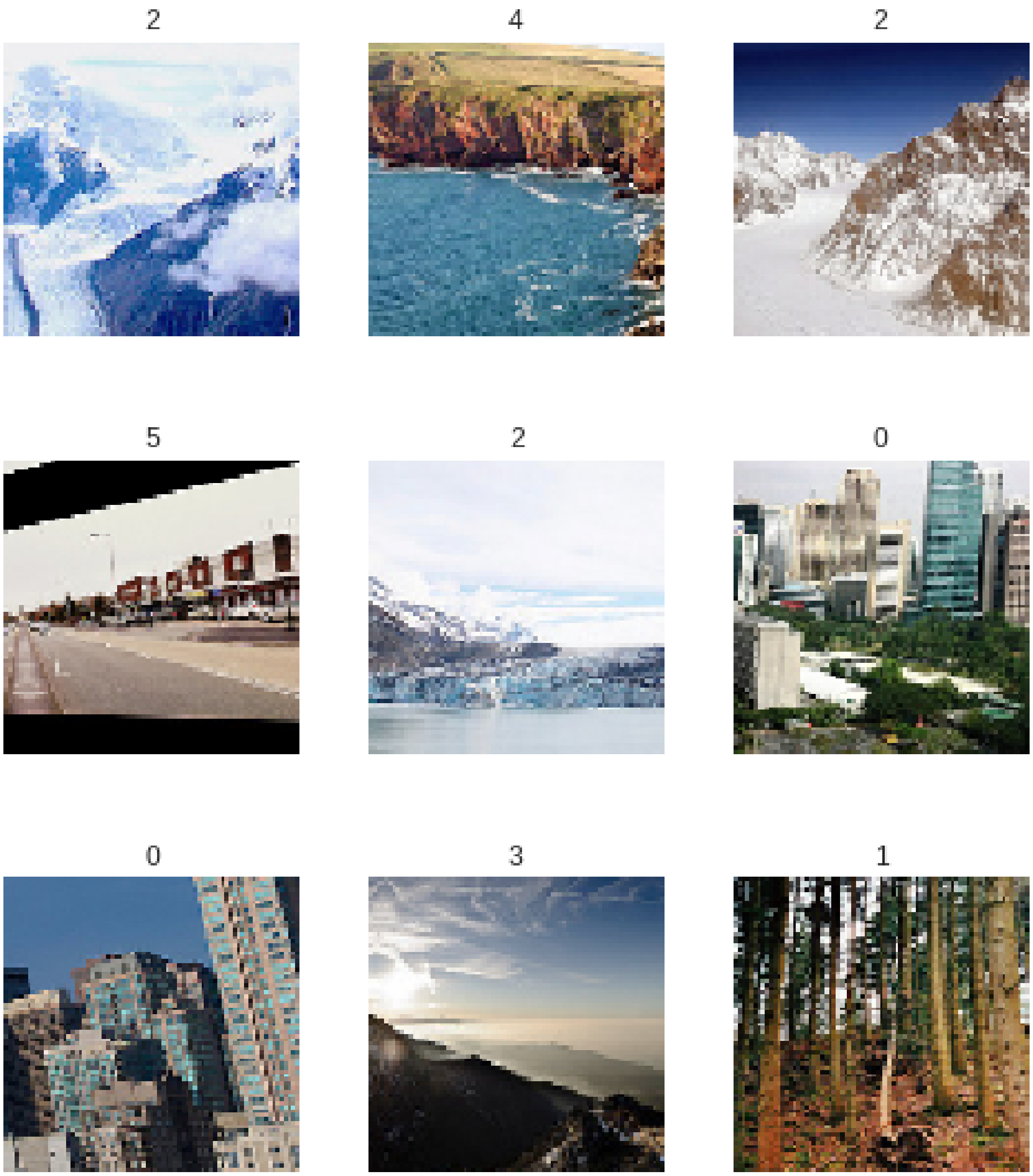
Create data object

```
In [0]: np.random.seed(123)
        #data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_transfo
        data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_transfor
```

```
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
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
        learn.fit_one_cycle(4)
```

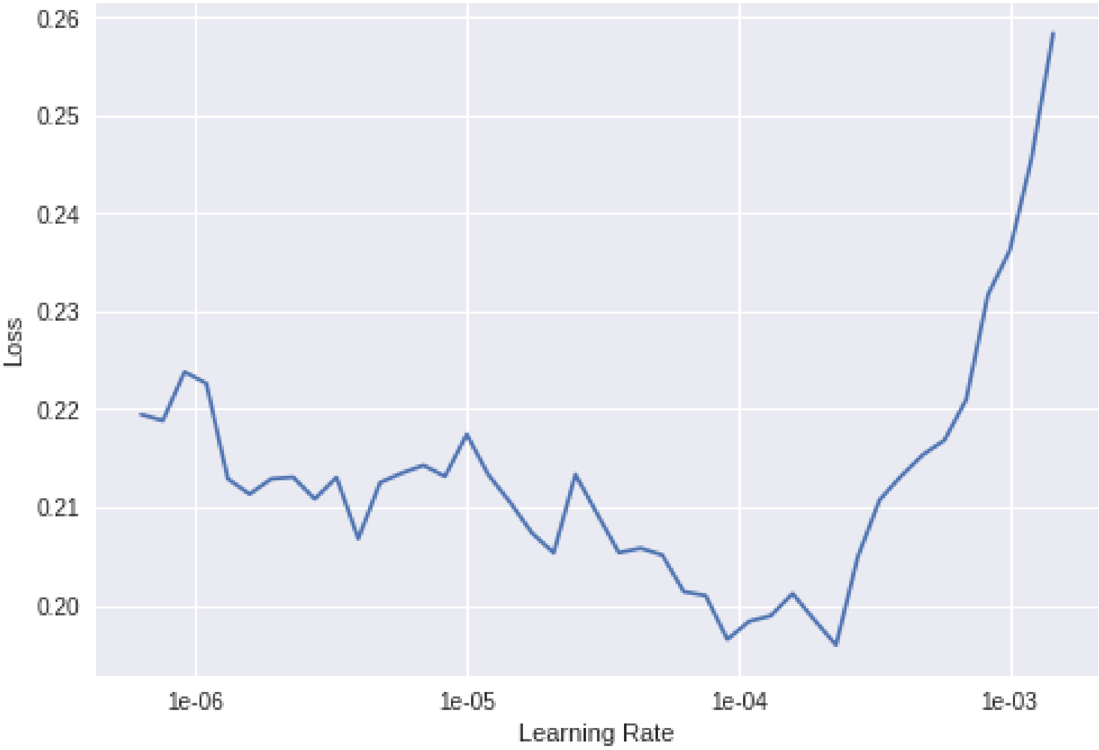
```
<IPython.core.display.HTML object>
```

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

```
In [0]: 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.
```



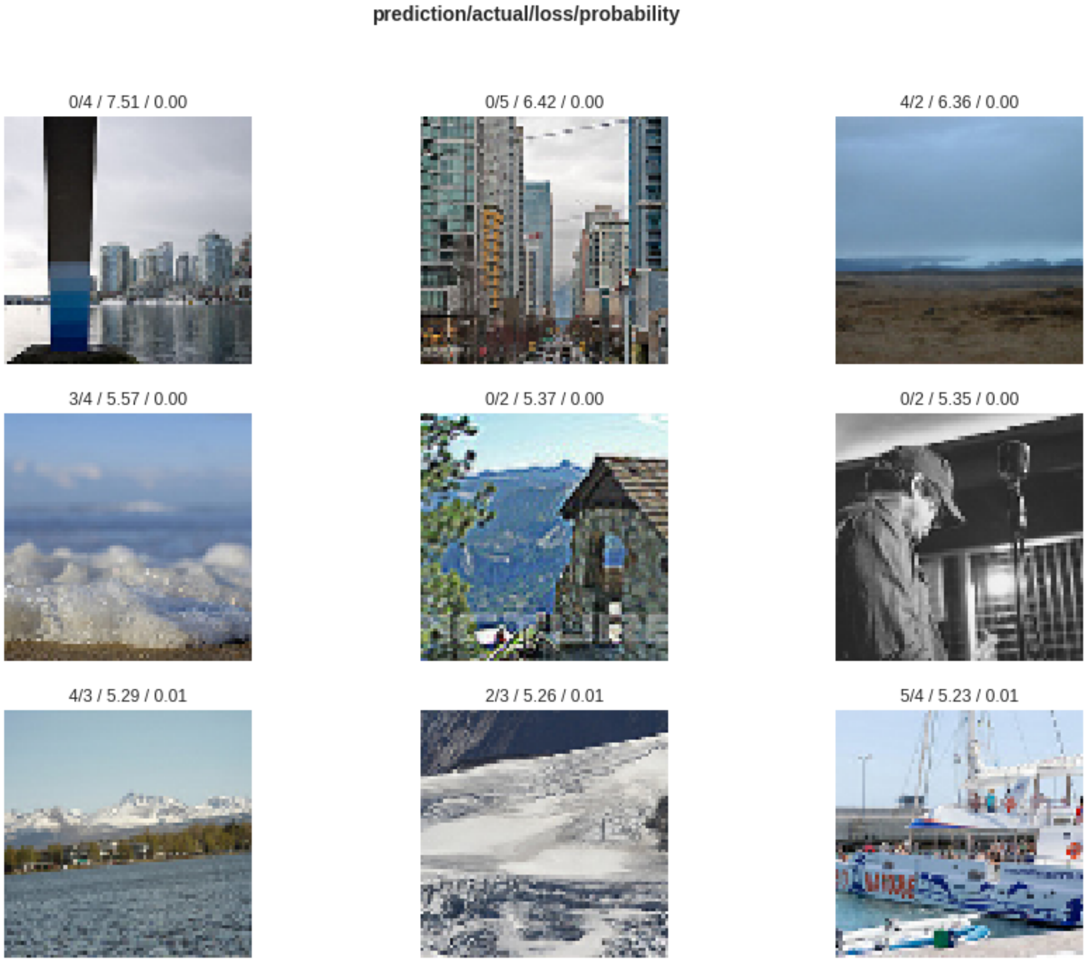
```
In [0]: #Run twice.  First run using LRs 3e-6 and 7e-6 and epoch 4.  Second set below.
        learn.fit_one_cycle(10, max_lr=slice(3e-04))
```

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

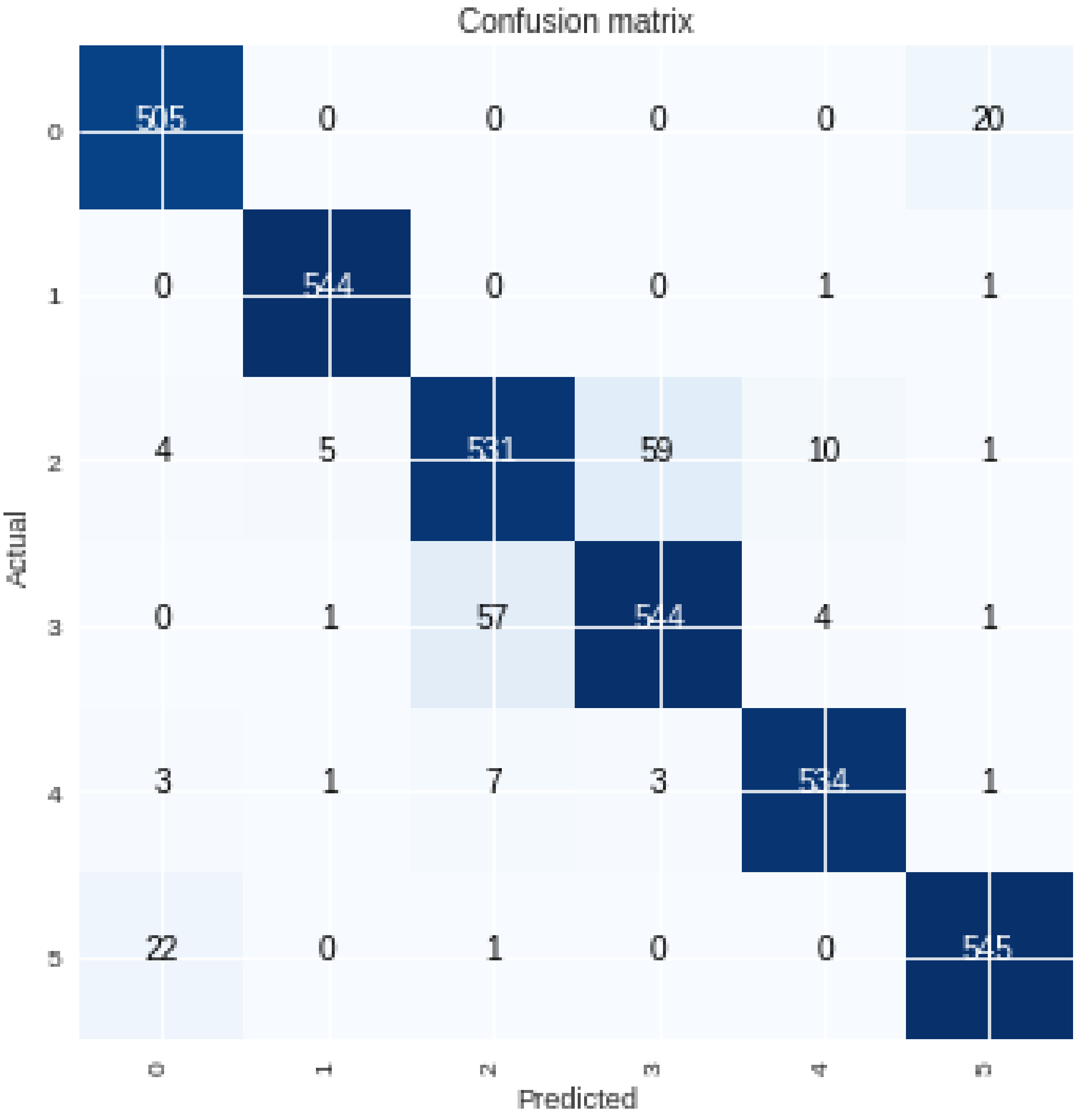
```
In [0]: learn.save('stage-2')
In [0]: learn.load('stage-2')
```

```
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 }
```

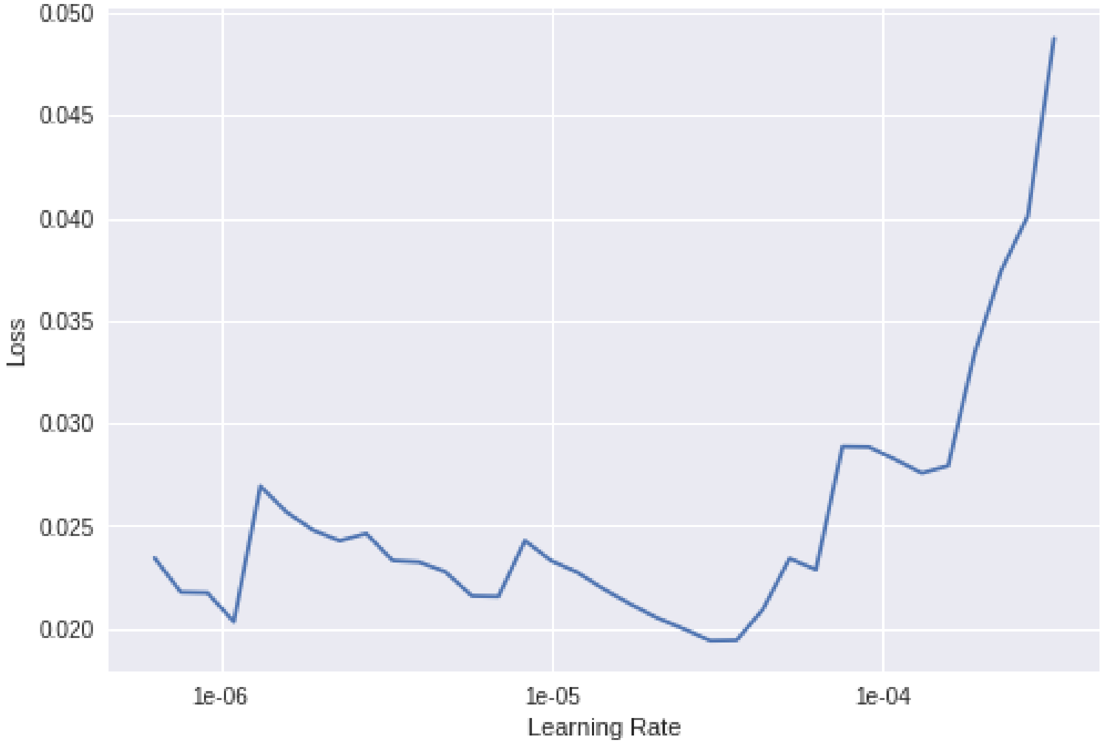
```
In [0]: #Second run
In [0]: learn.fit_one_cycle(5)
```

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

```
In [0]: 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.
```



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

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

```
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: SourceChangeWarning: source
  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':probabil
```

```
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


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

```
1
```

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

```
2
```

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

```
3
```

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

```
4
```

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

```
5
```

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

```
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_transforr
        #version 2
        #data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_transfo
```

```
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
100%|| 102502400/102502400 [00:01<00:00, 81751156.66it/s]
```

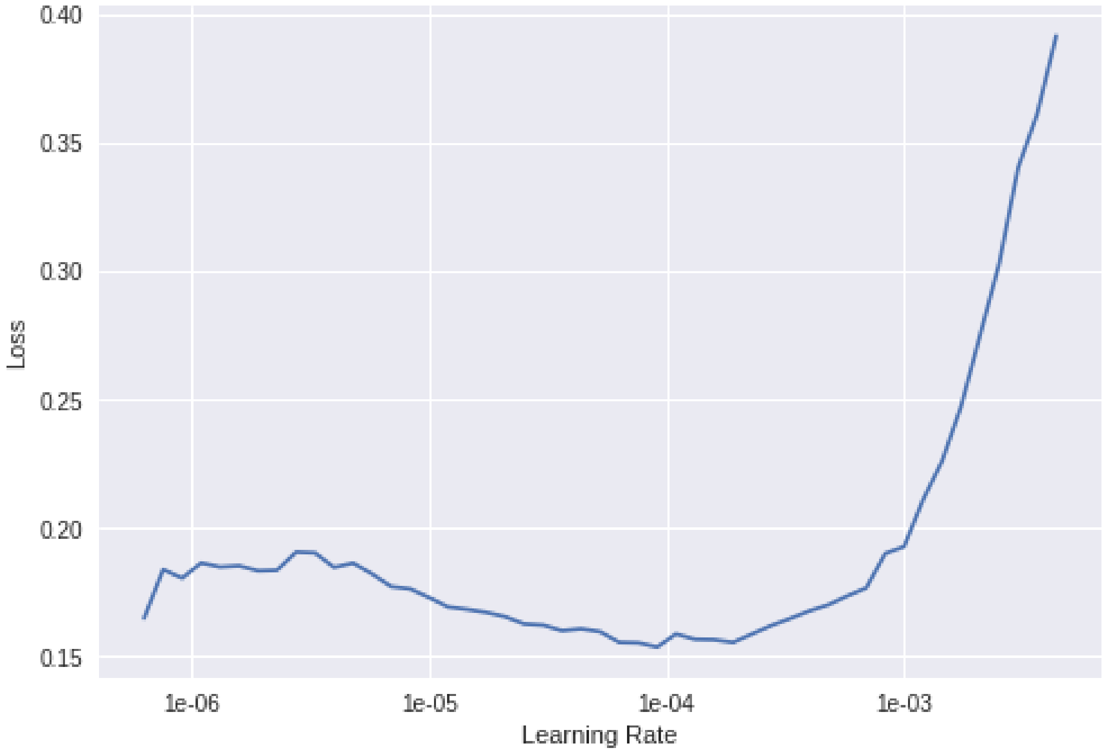
```
In [0]: #Basic first run
        learn.fit_one_cycle(5)
```

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

```
In [0]: 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.
```



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

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

```
In [0]:
```

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

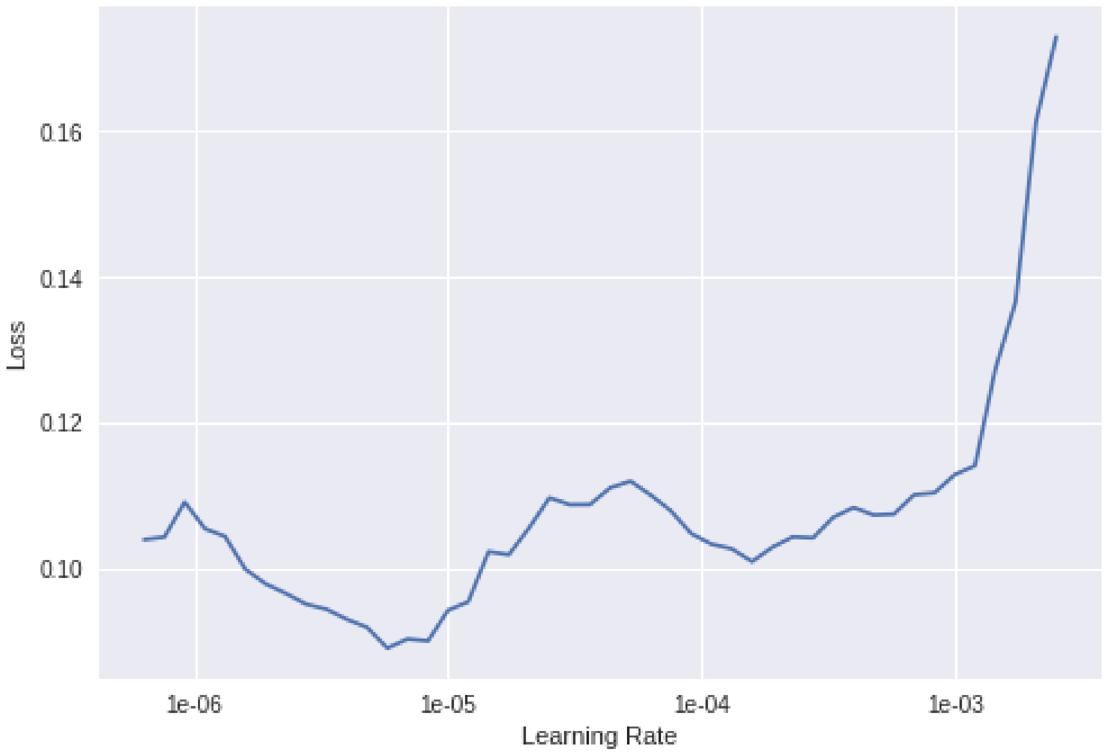
```
In [0]: #Second run
        learn.fit_one_cycle(4)
```

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

```
In [0]: 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.
```



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

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

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

```
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: SourceChangeWarning: source
  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':probabili
```

```
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_probability']
```

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

```
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


<IPython.core.display.HTML object>


1


<IPython.core.display.HTML object>


2


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

```
3
```

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

```
4
```

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

```
5
```

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

```
In [0]: np.random.seed(21)
        data = ImageDataBunch.from_folder(path, train=".", valid_pct=0.2, ds_tfms=get_transforn
        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
87306240it [00:00, 92303889.04it/s]
```

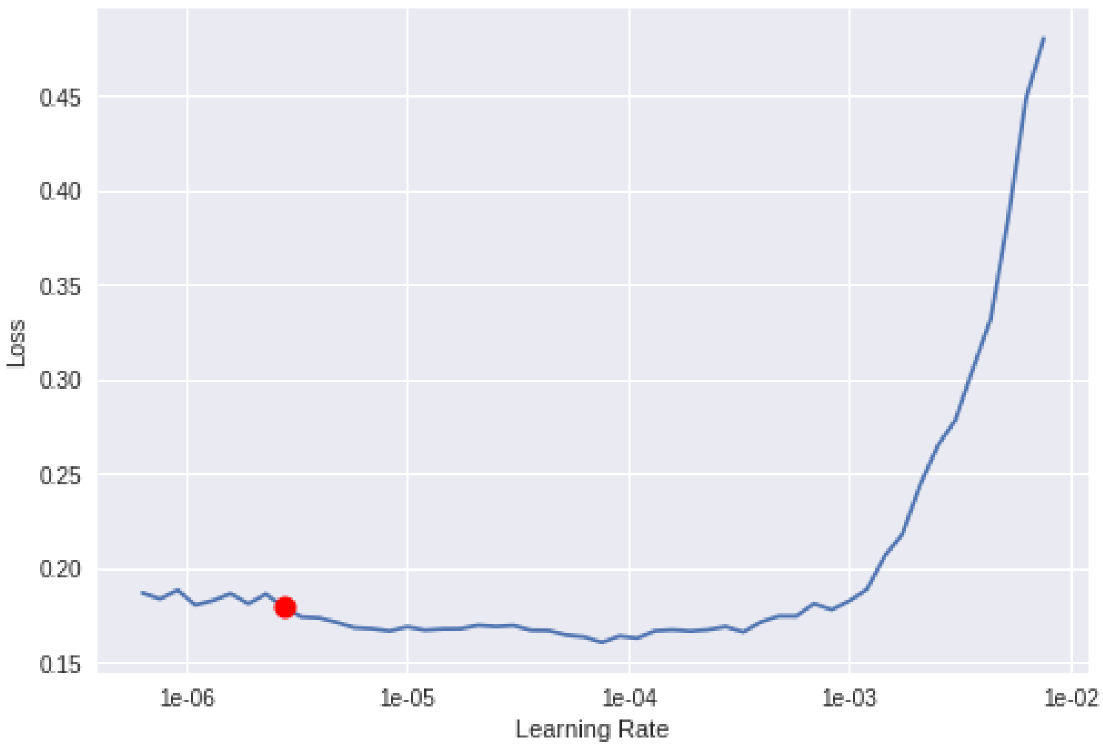
```
In [0]: learn.fit_one_cycle(5)
```

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

```
In [0]: 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.
Min numerical gradient: 2.75E-06
```



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

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

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

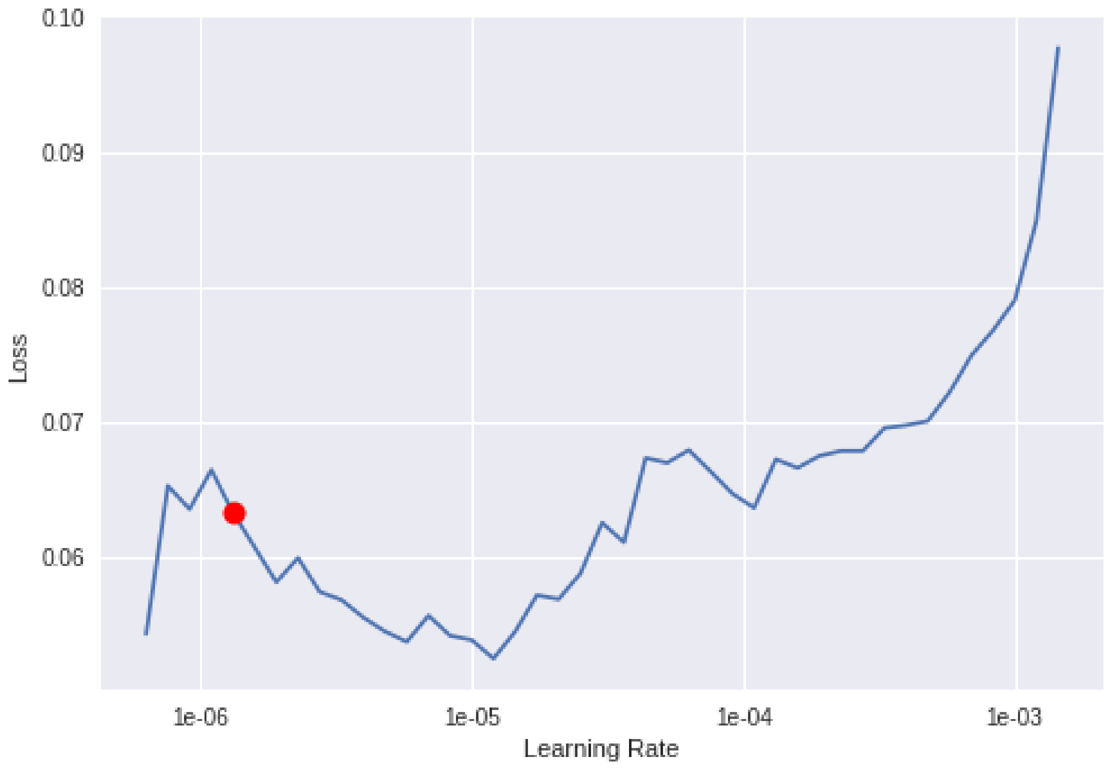
```
<IPython.core.display.HTML object>
```

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

```
In [0]: 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.
Min numerical gradient: 1.32E-06
```



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

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

```
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':probabil
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]:
```