The notebook shows the classification of photos of real estate interiors and exteriors.

The classes are - bedrooms, bathrooms, front exterior, back yards, kitchen, dining rooms, living room.

The photos for training are downloaded from Google Images. After training, the model can be used to identify the room in photographs.

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

In [20]:
```
#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/'
```

```
Mounted at /content/gdrive
```

Search for images in Google Images using the keywords and save the URLs in a file. Use script in Javscript console

urls = Array.from(document.querySelectorAll('.rg_di
.rg_meta')).map(el=>JSON.parse(el.textContent).ou); window.open('data:text/csv;charset=utf-8,' + escape(urls.join('\n')));

Copy file into Google drive path and name it 'key word+ _urls'

Later we will download the images at these URLs and use them to train the model.

Download pictures and upload into appropriate directories in Google Drive

In [0]:
```
labels = ['bedroom', 'bathroom', 'living', 'dining', 'kitchen', 'front', 'backya
path = Path(base_dir + 'data/realestate')
```

Alternative method for downloading images.

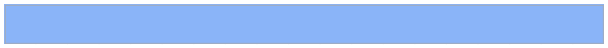
I am using a Windows PC and sometimes the files containing URLs downloaded by running the script do not work, especially if you open them in Windows and save them. Hence the method below is better if you are using Windows.

Use Chrome extension for downloading images. Upload images into Google folder named as the classes and delete images which do not show as thumbnails or are not suitable.

Verify images and delete bad ones

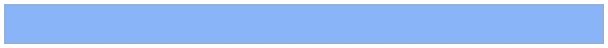
In [22]:
```
classes = ['bedroom', 'bathroom', 'living', 'dining', 'kitchen', 'front', 'backya
for c in classes:
    print(c)
    verify_images(path/c, delete=True, max_size=500)
```

```
bedroom
```



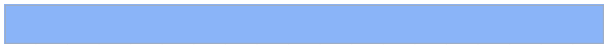
```
100.00% [1359/1359 00:07

bathroom
```



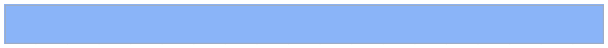
```
100.00% [460/460 00:02

Image /content/gdrive/My Drive/fastai-v3/data/realestate/bathroom/p_3_90236107_
the_controversial_tiny_airplane_bathroom_is_a_big_hit_with_airlines.gif has 1 i
nstead of 3 channels
living
```



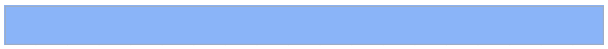
```
100.00% [735/735 00:04

Image /content/gdrive/My Drive/fastai-v3/data/realestate/living/unnamed.gif has
1 instead of 3 channels
Image /content/gdrive/My Drive/fastai-v3/data/realestate/living/02_04_19_hero_h
p.gif has 1 instead of 3 channels
dining
```



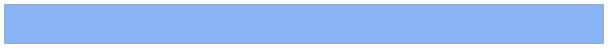
```
100.00% [1207/1207 00:06

kitchen
```



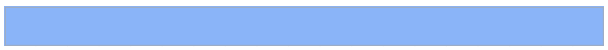
```
100.00% [933/933 00:05

Image /content/gdrive/My Drive/fastai-v3/data/realestate/kitchen/Beautiful_turq
uoise_kitchen_cabinets_artistic_turquoise_mosaic_tiles_backsplash_white_kitchen
_countertop_that_is_combined_with_kitchen_island_some_barstools_three_beautiful
_pendant.gif has 1 instead of 3 channels
Image /content/gdrive/My Drive/fastai-v3/data/realestate/kitchen/kitchen_styles
_Bespoke_and_Fitted_Kitchens_Kitchen_Styles_Installation_hints_kitchen_styles_2
016_36l6l4shxmdcqtgg6r6qrk.gif has 1 instead of 3 channels
front
```



```
100.00% [916/916 00:05

Image /content/gdrive/My Drive/fastai-v3/data/realestate/front/150317_blog_phot
o_African_House_exterior_front.gif has 1 instead of 3 channels
Image /content/gdrive/My Drive/fastai-v3/data/realestate/front/five_plex_town_h
ouse_plan_with_ada_accessible_front_ele_detail_fv_580.gif has 1 instead of 3 ch
annels
backyard
```



```
100.00% [706/706 00:03
```

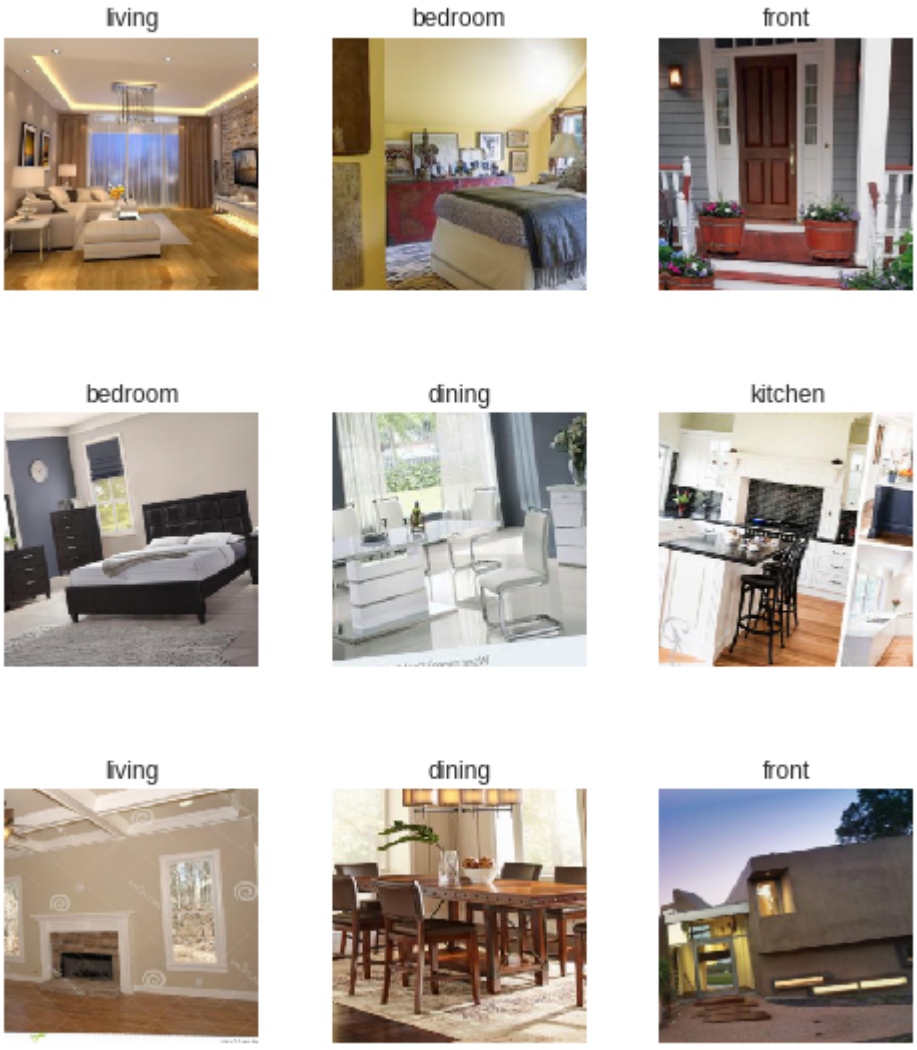
Create data object

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

In [24]:
```
data.classes
```

Out[24]:
```
['backyard', 'bathroom', 'bedroom', 'dining', 'front', 'kitchen', 'living']
```

In [25]:
```
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.resnet34, metrics=error_rate)
```

In [31]:
```
learn.fit_one_cycle(4)
```

Total time: 06:37

epoch	train_loss	valid_loss	error_rate
1	1.178601	0.606814	0.206186
2	0.813155	0.538742	0.180016
3	0.656106	0.499230	0.174465
4	0.575916	0.485419	0.167328

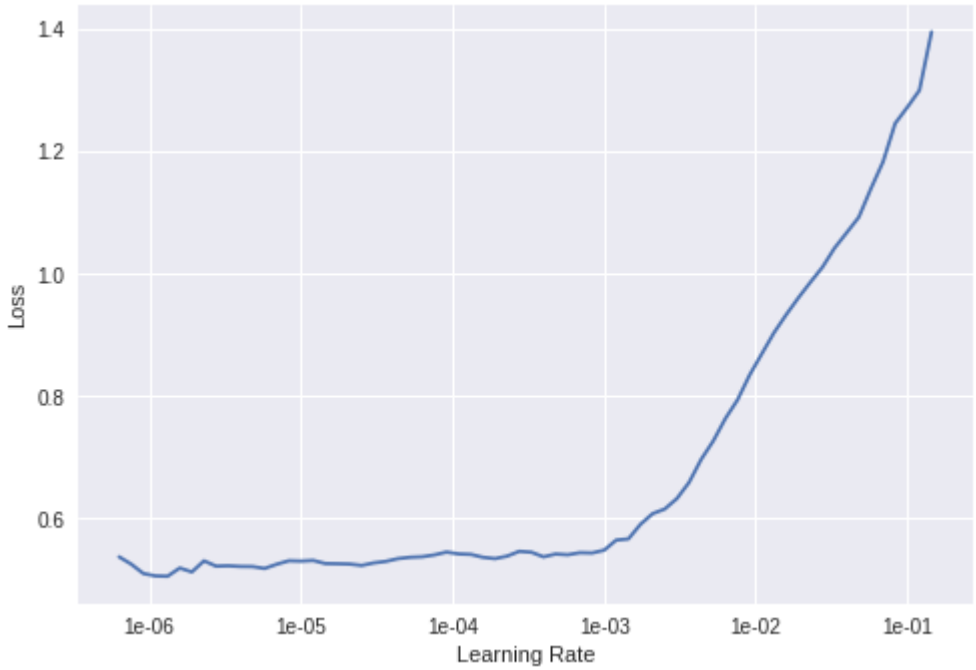
In [28]:
```
learn.unfreeze()
learn.lr_find()
learn.recorder.plot()
```

50.00% [1/2 01:27

epoch	train_loss	valid_loss	error_rate
1	1.394698		

Interrupted

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



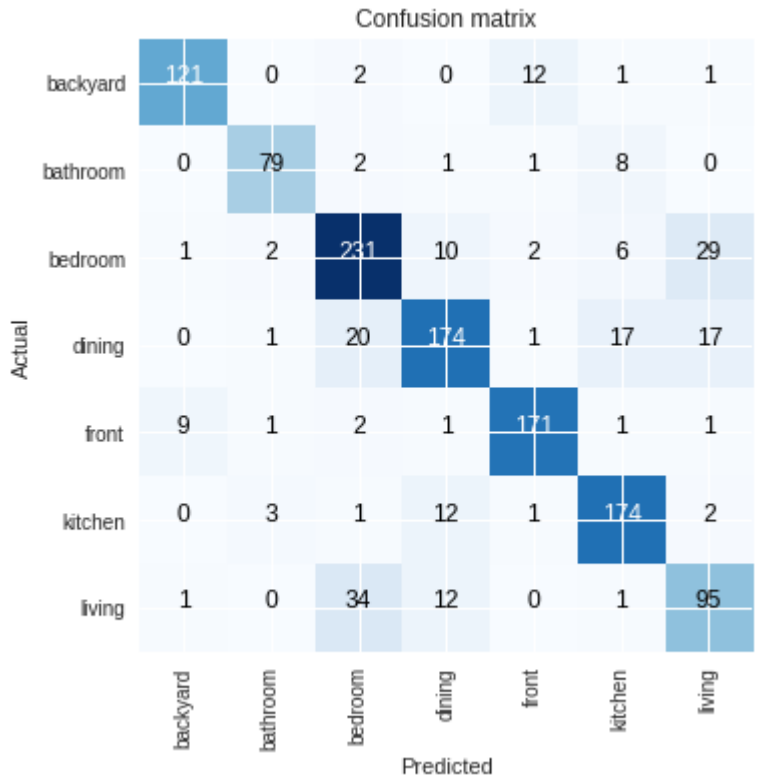
In [32]:
```
learn.fit_one_cycle(4, max_lr=slice(3e-06, 3e-05))
```

Total time: 06:38

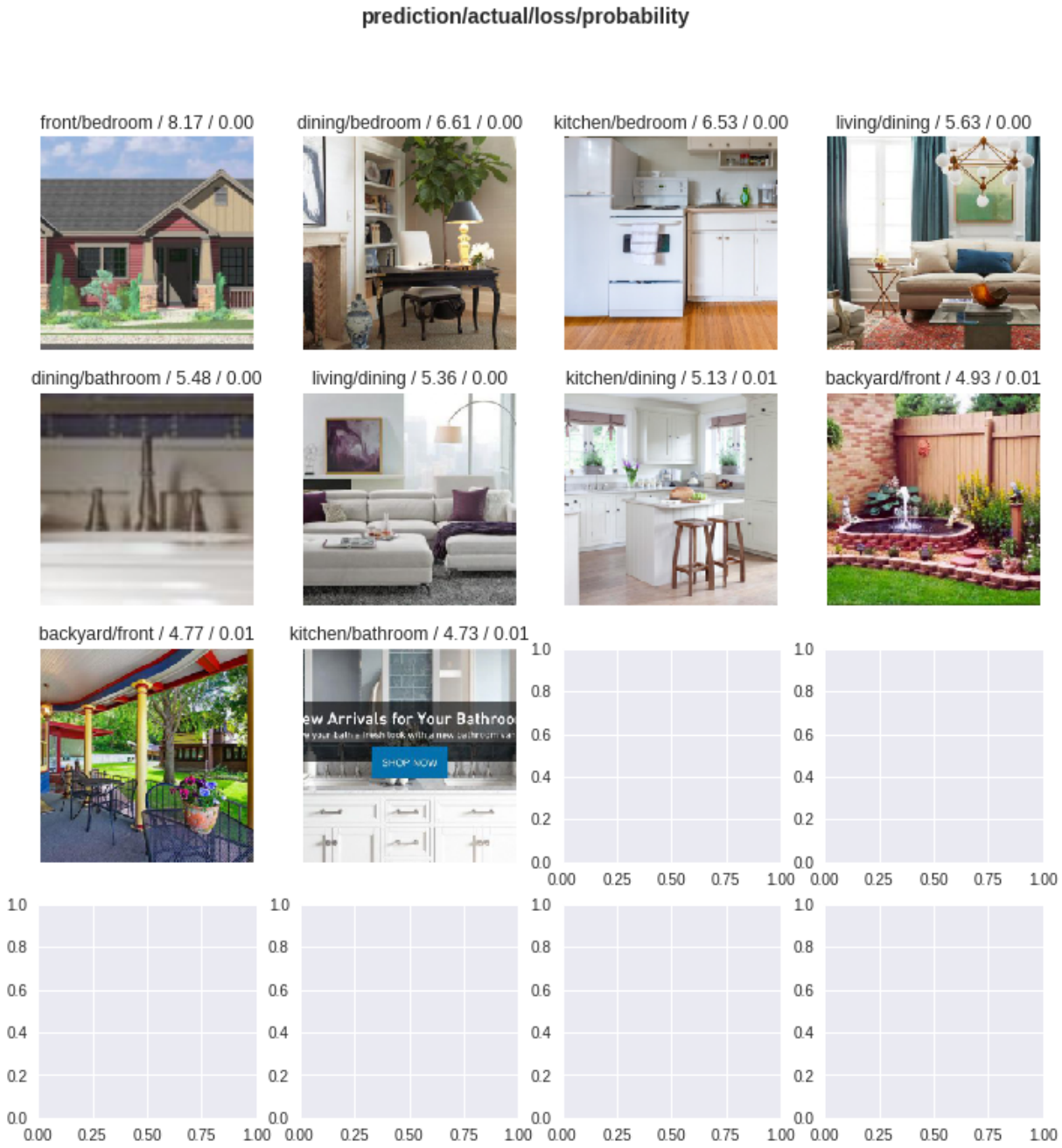
epoch	train_loss	valid_loss	error_rate
1	0.542113	0.487408	0.170500
2	0.548106	0.485540	0.174465
3	0.543804	0.483743	0.168121
4	0.562206	0.483839	0.171293

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

In [34]:
```
interp.plot_confusion_matrix()
```



In [35]:
```
interp.plot_top_losses(10)
```



Export model for portability and do prediction

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

In [38]:
```
# Impoort new image
img = open_image(path/'pic6.jpg')
img
```

Out[38]:

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

In [40]:
```
#Predict class of image
pred_class,pred_idx,outputs = learn.predict(img)
pred_class
```

Out[40]:
```
Category bedroom
```

The error rate for the above model is not great but considering that it was created by random images downloaded from Google, it is pretty good. It can be improved by cleaning up the training images and removing junk.