Loading the data from Google drive on Colab

```
In [0]: from google.colab import drive
        drive.mount('/Arch')
```

```
Drive already mounted at /Arch; to attempt to forcibly remount, call drive.mount("/Arch", force_remount=True).
```

```
In [0]: !wget https://raw.githubusercontent.com/anhquan0412/animation-classification/master/gradcam.py
```

```
--2019-12-29 19:09:34--  https://raw.githubusercontent.com/anhquan0412/animation-classification/master/gradcam.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6764 (6.6K) [text/plain]
Saving to: ‘gradcam.py.2’

gradcam.py.2        100%[===================>]   6.61K  --.-KB/s    in 0s

2019-12-29 19:09:35 (99.1 MB/s) - ‘gradcam.py.2’ saved [6764/6764]
```

```
In [0]: %reload_ext autoreload
        %autoreload 2
        %matplotlib inline
```

Importing necessary libraries

```
In [0]: import torchvision
        import torch
        import seaborn as sns
        from fastai.vision import *
        from fastai.metrics import error_rate
        from fastai.callbacks import *
        from gradcam import *
        from PIL import ImageFile
        from collections import OrderedDict
        from sklearn.manifold import TSNE
        from sklearn import manifold, datasets
        from sklearn.metrics.pairwise import pairwise_distances
        from sklearn.metrics import confusion_matrix
        from scipy.spatial.distance import squareform
        from matplotlib.offsetbox import OffsetImage, AnnotationBbox
        from matplotlib.ticker import NullFormatter
        import PIL
        ImageFile.LOAD_TRUNCATED_IMAGES = True
```

Creating path for the data

```
In [0]: path = Path('/Arch/My Drive/Arch')
        path
Out[0]: PosixPath('/Arch/My Drive/Arch')
```

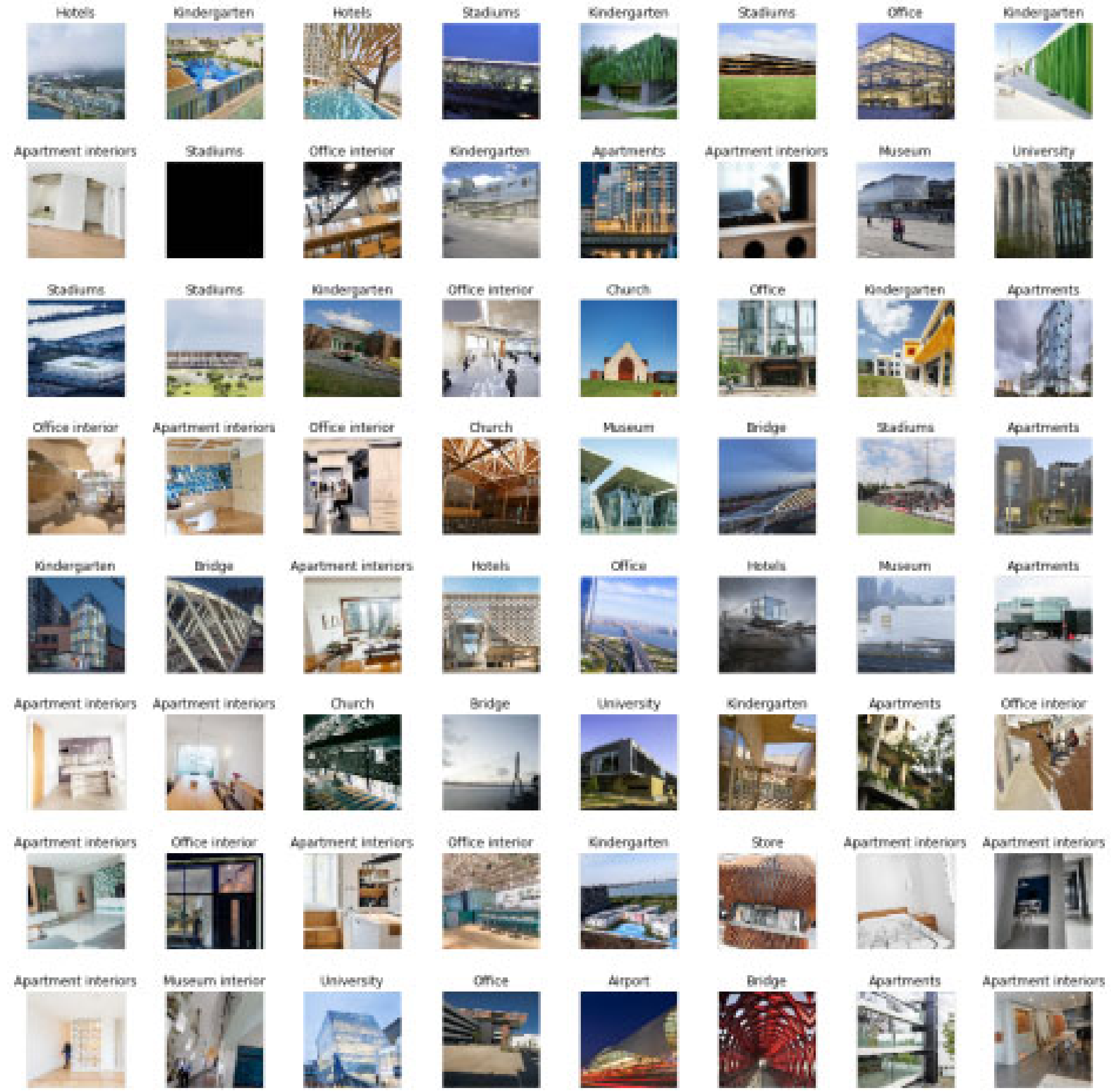
Defining the batch size

```
In [0]: path.ls()
Out[0]: [PosixPath('/Arch/My Drive/Arch/models'),
         PosixPath('/Arch/My Drive/Arch/Office'),
         PosixPath('/Arch/My Drive/Arch/Church'),
         PosixPath('/Arch/My Drive/Arch/Apartment interiors'),
         PosixPath('/Arch/My Drive/Arch/Office interior'),
         PosixPath('/Arch/My Drive/Arch/Bridge'),
         PosixPath('/Arch/My Drive/Arch/Apartments'),
         PosixPath('/Arch/My Drive/Arch/Airport'),
         PosixPath('/Arch/My Drive/Arch/Stadiums'),
         PosixPath('/Arch/My Drive/Arch/Hotels'),
         PosixPath('/Arch/My Drive/Arch/Museum'),
         PosixPath('/Arch/My Drive/Arch/Kindergarten'),
         PosixPath('/Arch/My Drive/Arch/Store'),
         PosixPath('/Arch/My Drive/Arch/University'),
         PosixPath('/Arch/My Drive/Arch/Museum interior')]
```

Creating test-train data using Data bunch

```
In [0]: np.random.seed(42)
        data = ImageDataBunch.from_folder(path, train='Arch', valid_pct=0.2, ds_tfms=get_transforms(do_flip=False), size=224, num_
        workers=16).normalize(imagenet_stats)
```

```
In [0]: data.show_batch(rows=8, figsize=(15,15))
```




```
In [0]: data.classes, data.c, len(data.train_ds), len(data.valid_ds)
Out[0]: (['Airport',
          'Apartment interiors',
          'Apartments',
          'Bridge',
          'Church',
          'Hotels',
          'Kindergarten',
          'Museum',
          'Museum interior',
          'Office',
          'Office interior',
          'Stadiums',
          'Store',
          'University'],
         14,
         4834,
         1208)
```

Using Resnet50 for Transfer Learning

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

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/checkpoints/resnet50-19c8e
357.pth
100%|██████████| 97.8M/97.8M [00:02<00:00, 40.0MB/s]
```

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

epoch	train_loss	valid_loss	accuracy	time
0	1.955981	1.671606	0.474338	04:08
1	1.789256	1.489312	0.500000	03:01
2	1.477718	1.401074	0.522351	02:59
3	1.246057	1.383568	0.526490	02:59

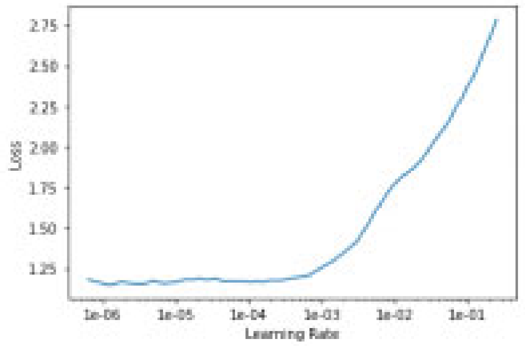
In [0]:
```
learn.save('stage-1')
```

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

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

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

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



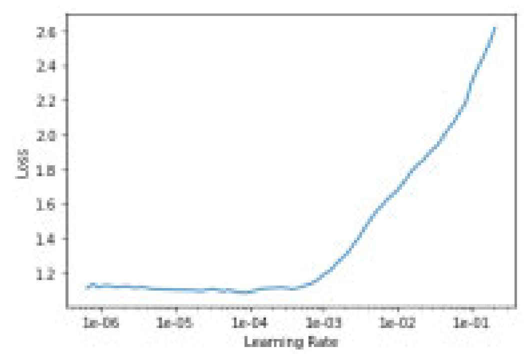
In [0]:
```
learn.fit_one_cycle(2, max_lr=slice(1e-6,9e-6))
```

epoch	train_loss	valid_loss	accuracy	time
0	1.170974	1.380192	0.523179	03:01
1	1.148526	1.382090	0.526490	03:01

Findind the learning rate again

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(4, max_lr=9e-05)
```

epoch	train_loss	valid_loss	accuracy	time
0	1.142794	1.375471	0.533113	03:02
1	1.106165	1.419087	0.520695	03:02
2	0.821772	1.346013	0.543046	03:01
3	0.625261	1.359464	0.556291	03:01

We got an accuracy of 55.6% and we will save this model now

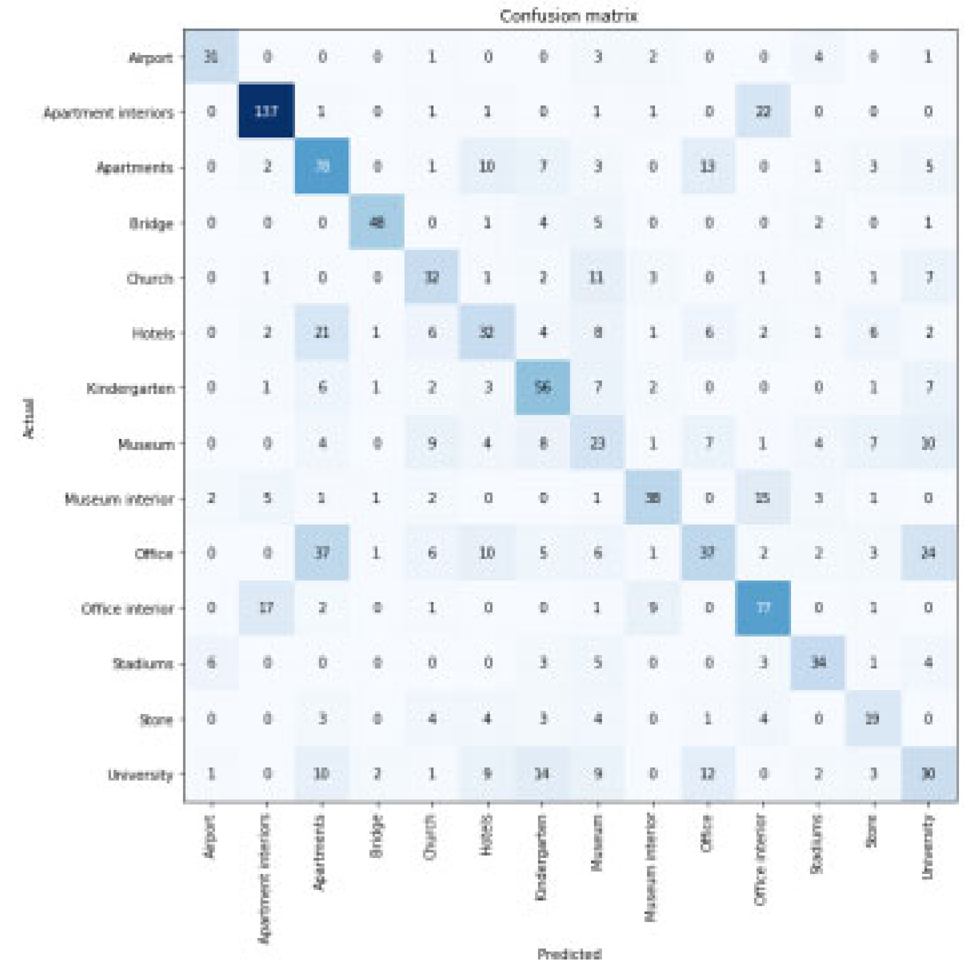
In [0]:
```
learn.save('stage-2');
```

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

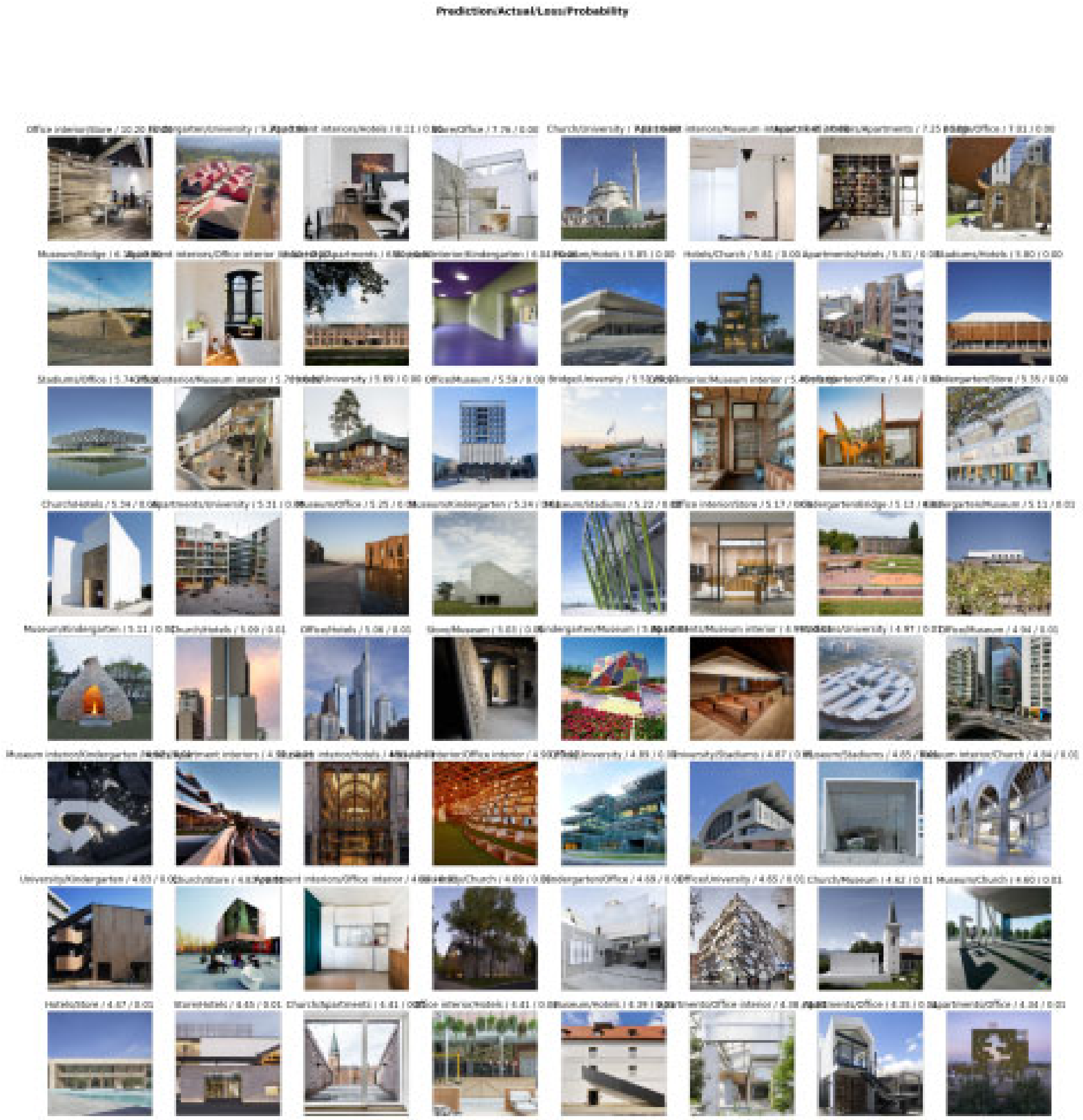
In [0]:
```
interp = ClassificationInterpretation.from_learner(learn)
```

We will check with confusion matrix that which architectural typology is conceived as something else

In [0]:
```
interp.plot_confusion_matrix(figsize=(10,10))
```



In [0]:
```
interp.plot_top_losses(64, figsize=(25,25))
```

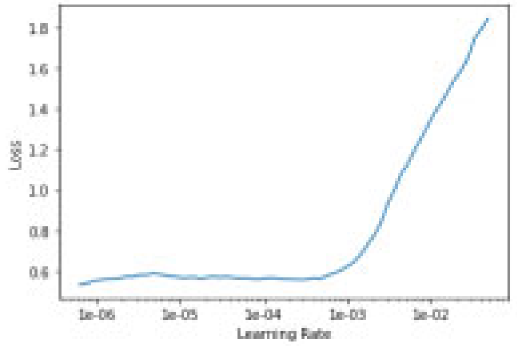


In [0]:
```
interp.most_confused(min_val=2)
```

Out[0]:
```
[('Office', 'Apartments', 37),
 ('Office', 'University', 24),
 ('Apartment interiors', 'Office interior', 22),
 ('Hotels', 'Apartments', 21),
 ('Office interior', 'Apartment interiors', 17),
 ('Museum interior', 'Office interior', 15),
 ('University', 'Kindergarten', 14),
 ('Apartments', 'Office', 13),
 ('University', 'Office', 12),
 ('Church', 'Museum', 11),
 ('Apartments', 'Hotels', 10),
 ('Museum', 'University', 10),
 ('Office', 'Hotels', 10),
 ('University', 'Apartments', 10),
 ('Museum', 'Church', 9),
 ('Office interior', 'Museum interior', 9),
 ('University', 'Hotels', 9),
 ('University', 'Museum', 9),
 ('Hotels', 'Museum', 8),
 ('Museum', 'Kindergarten', 8),
 ('Apartments', 'Kindergarten', 7),
 ('Church', 'University', 7),
 ('Kindergarten', 'Museum', 7),
 ('Kindergarten', 'University', 7),
 ('Museum', 'Office', 7),
 ('Museum', 'Store', 7),
 ('Hotels', 'Church', 6),
 ('Hotels', 'Office', 6),
 ('Hotels', 'Store', 6),
 ('Kindergarten', 'Apartments', 6),
 ('Office', 'Church', 6),
 ('Office', 'Museum', 6),
 ('Stadiums', 'Airport', 6),
 ('Apartments', 'University', 5),
 ('Bridge', 'Museum', 5),
 ('Museum interior', 'Apartment interiors', 5),
 ('Office', 'Kindergarten', 5),
 ('Stadiums', 'Museum', 5),
 ('Airport', 'Stadiums', 4),
 ('Bridge', 'Kindergarten', 4),
 ('Hotels', 'Kindergarten', 4),
 ('Museum', 'Apartments', 4),
 ('Museum', 'Hotels', 4),
 ('Museum', 'Stadiums', 4),
 ('Stadiums', 'University', 4),
 ('Store', 'Church', 4),
 ('Store', 'Hotels', 4),
 ('Store', 'Museum', 4),
 ('Store', 'Office interior', 4),
 ('Airport', 'Museum', 3),
 ('Apartments', 'Museum', 3),
 ('Apartments', 'Store', 3),
 ('Church', 'Museum interior', 3),
 ('Kindergarten', 'Hotels', 3),
 ('Museum interior', 'Stadiums', 3),
 ('Office', 'Store', 3),
 ('Stadiums', 'Kindergarten', 3),
 ('Stadiums', 'Office interior', 3),
 ('Store', 'Apartments', 3),
 ('Store', 'Kindergarten', 3),
 ('University', 'Store', 3),
 ('Airport', 'Museum interior', 2),
 ('Apartments', 'Apartment interiors', 2),
 ('Bridge', 'Stadiums', 2),
 ('Church', 'Kindergarten', 2),
 ('Hotels', 'Apartment interiors', 2),
 ('Hotels', 'Office interior', 2),
 ('Hotels', 'University', 2),
 ('Kindergarten', 'Church', 2),
 ('Kindergarten', 'Museum interior', 2),
 ('Museum interior', 'Airport', 2),
 ('Museum interior', 'Church', 2),
 ('Office', 'Office interior', 2),
 ('Office', 'Stadiums', 2),
 ('Office interior', 'Apartments', 2),
 ('University', 'Bridge', 2),
 ('University', 'Stadiums', 2)]
```

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(10, max_lr=1e-4)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.570935	1.360548	0.560430	03:15
1	0.538401	1.445140	0.548841	03:13
2	0.583791	1.615703	0.535596	03:16
3	0.504413	1.624838	0.543874	03:16
4	0.396519	1.718363	0.533113	03:17
5	0.263588	1.673504	0.554636	03:18
6	0.180753	1.662497	0.552980	03:16
7	0.129170	1.713078	0.556291	03:16
8	0.096595	1.696554	0.571192	03:17
9	0.079907	1.690977	0.558775	03:15

In [0]:
```
learn.save('stage-3');
```

Trying a new method

In [0]:
```
def get_data(sz, bs):
  data = ImageDataBunch.from_folder(path, train='Arch', valid_pct=0.2, ds_tfms=get_transforms(), size=sz, bs=bs).normalize
(imagenet_stats)
  return data
```

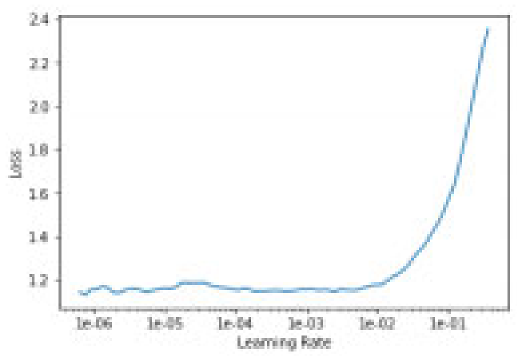
In [0]:
```
learn2 = cnn_learner(get_data(32, 512), models.resnet50, metrics=[error_rate, accuracy])
```

In [0]:
```
learn2.fit_one_cycle(5)
```

In [0]:
```
learn2.lr_find()
```

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

In [0]:
```
learn2.recorder.plot()
```



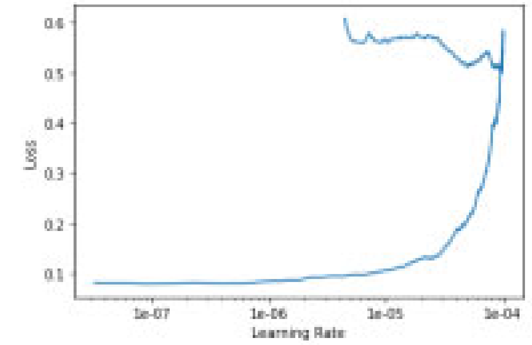
In [0]:
```
learn2.fit_one_cycle(3, max_lr=1e-06)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.129086	1.398519	0.468543	0.531457	03:16
1	1.138634	1.399333	0.463576	0.536424	03:17
2	1.107931	1.396177	0.470199	0.529801	03:17

In [0]:
```
learn2.lr_find()
```

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

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



In [0]:
```
learn2.fit_one_cycle(3)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.334627	1.568510	0.498344	0.501656	03:18
1	1.402888	1.441917	0.480960	0.519040	03:17
2	1.165841	1.380477	0.474338	0.525662	03:19

In [0]:
```
learn2.save('phase1')
```

In [0]:
```
learn2 = cnn_learner(get_data(48, 512), models.resnet50, metrics=[error_rate, accuracy]).load('phase1')
```

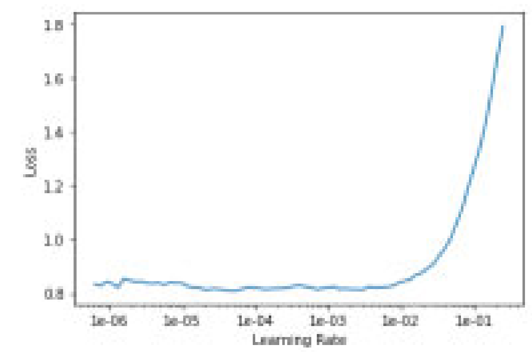
In [0]:
```
learn2.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.070465	1.412984	0.468543	0.531457	03:19
1	1.213626	1.519876	0.494205	0.505795	03:19
2	1.160599	1.445225	0.480132	0.519868	03:20
3	1.022192	1.424774	0.478477	0.521523	03:20
4	0.876660	1.402982	0.468543	0.531457	03:18

In [0]:
```
learn2.lr_find()
```

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

In [0]:
```
learn2.recorder.plot()
```



In [0]:
```
learn2.fit_one_cycle(3, max_lr=7e-05)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.822961	1.406894	0.471854	0.528146	03:19
1	0.801683	1.390496	0.470199	0.529801	03:19
2	0.821771	1.389226	0.464404	0.535596	03:19

In [0]:
```
learn2.save('phase2')
```

In [0]:
```
learn2 = cnn_learner(get_data(64, 512), models.resnet50, metrics=[error_rate, accuracy]).load('phase2')
```

In [0]:
```
learn2.fit_one_cycle(5)
```

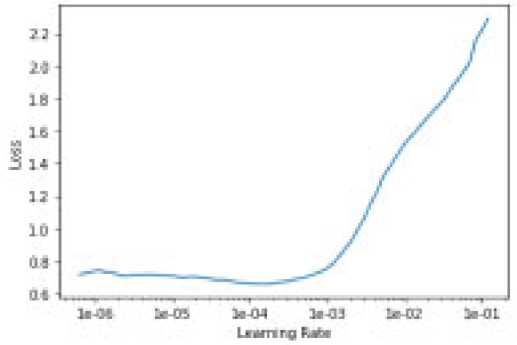
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.813695	1.427225	0.475993	0.524007	03:20
1	0.992092	1.601808	0.510762	0.489238	03:19
2	0.954474	1.512853	0.482616	0.517384	03:19
3	0.844257	1.497500	0.479305	0.520695	03:19
4	0.725966	1.485151	0.476821	0.523179	03:17

In [0]:
```
learn2.unfreeze()
```

In [0]:
```
learn2.lr_find()
```

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

In [0]:
```
learn2.recorder.plot()
```



In [0]:
```
learn2.fit_one_cycle(8, max_lr=3e-04)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.750782	1.519779	0.476821	0.523179	03:18
1	1.061973	2.141714	0.536424	0.463576	03:18
2	1.251276	1.670148	0.519868	0.480132	03:21
3	1.070525	1.812504	0.526490	0.473510	03:18
4	0.802944	1.643080	0.508278	0.491722	03:19
5	0.526182	1.457213	0.434603	0.565397	03:18
6	0.308313	1.465662	0.422185	0.577815	03:19
7	0.217918	1.464630	0.424669	0.575331	03:19

In [0]:
```
learn2.save('phase3')
```

In [0]:
```
learn2 = cnn_learner(get_data(128, 128), models.resnet50, metrics=[error_rate, accuracy]).load('phase3')
```

In [0]:
```
learn2.fit_one_cycle(5)
```

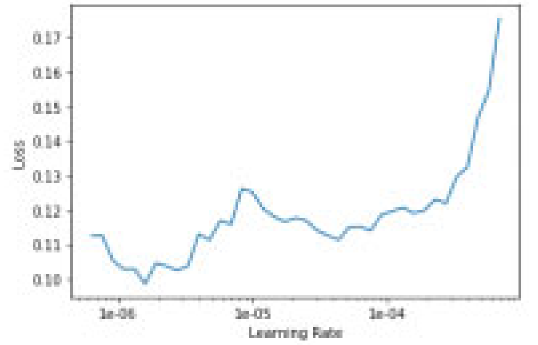
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.167189	1.699540	0.420530	0.579470	03:19
1	0.221488	2.015457	0.450331	0.549669	03:18
2	0.188798	2.075188	0.427980	0.572020	03:18
3	0.154109	2.098802	0.421358	0.578642	03:19
4	0.131698	2.085149	0.421358	0.578642	03:19

In [0]:
```
learn2.unfreeze()
```

In [0]:
```
learn2.lr_find()
```

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

In [0]:
```
learn2.recorder.plot()
```



In [0]:
```
learn2.fit_one_cycle(8, max_lr=2e-06)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.122472	2.098169	0.420530	0.579470	03:20
1	0.119493	2.099597	0.423841	0.576159	03:19
2	0.110110	2.114947	0.419702	0.580298	03:22
3	0.106753	2.069649	0.425497	0.574503	03:21
4	0.096137	2.093524	0.418046	0.581954	03:19
5	0.099020	2.090341	0.423841	0.576159	03:20
6	0.104677	2.104749	0.419702	0.580298	03:20
7	0.101605	2.072259	0.420530	0.579470	03:20

In [0]:
```
learn2.save('phase4')
```

In [0]:
```
learn2 = cnn_learner(get_data(224, 64), models.resnet50, metrics=[error_rate, accuracy]).load('phase4')
```

In [0]:
```
learn2.fit_one_cycle(5)
```

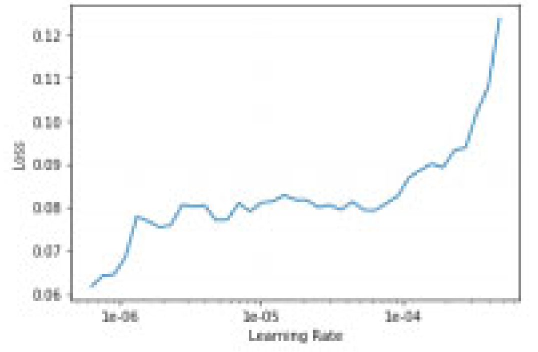
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.099750	2.214071	0.418046	0.581954	03:21
1	0.150246	2.356459	0.444536	0.555464	03:20
2	0.162536	2.400943	0.441225	0.558775	03:17
3	0.118002	2.358040	0.425497	0.574503	03:18
4	0.113363	2.359541	0.431291	0.568709	03:20

In [0]:
```
learn2.unfreeze()
```

In [0]:
```
learn2.lr_find()
```

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

In [0]:
```
learn2.recorder.plot()
```



In [0]:
```
learn2.fit_one_cycle(3)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.901894	3.564818	0.560430	0.439570	03:20
1	1.092290	1.548566	0.454470	0.545530	03:22
2	0.601046	1.568590	0.447848	0.552152	03:19

In [0]:
```
learn2.save('phase5')
```

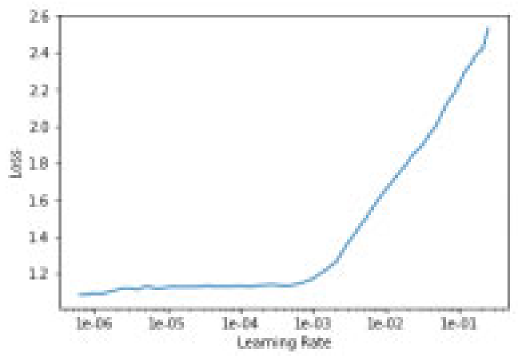
In [0]:
```
learn2 = cnn_learner(get_data(224, 64), models.resnet50, metrics=[error_rate, accuracy]).load('phase5').mixup()
```

In [0]:
```
learn2.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.442502	1.429762	0.428808	0.571192	03:22
1	1.298910	1.455719	0.448675	0.551324	03:20
2	1.242569	1.433064	0.431291	0.568709	03:20
3	1.161303	1.416264	0.427152	0.572848	03:18
4	1.129809	1.410552	0.423841	0.576159	03:17

In [0]:
```
learn2.unfreeze()
learn2.lr_find()
learn2.recorder.plot()
```

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



In [0]:
```
learn2.save('phase6_mixup_0')
```

In [0]:
```
learn2 = cnn_learner(get_data(224, 64), models.resnet50, metrics=[error_rate, accuracy]).load('phase6_mixup_0').mixup()
```

In [0]:
```
learn2.fit_one_cycle(8, max_lr=1e-06)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.110395	1.409555	0.425497	0.574503	03:20
1	1.101688	1.404688	0.424669	0.575331	03:21
2	1.116517	1.423025	0.427980	0.572020	03:19
3	1.115806	1.408130	0.419702	0.580298	03:20
4	1.125047	1.412692	0.420530	0.579470	03:19
5	1.118622	1.403645	0.421358	0.578642	03:18
6	1.112767	1.409595	0.418874	0.581126	03:19
7	1.118993	1.409714	0.422185	0.577815	03:19

In [0]:
```
learn2.save('phase6_mixup_1')
```

Resnet152 with mixup(regularization)

In [0]:
```
learn3 = cnn_learner(get_data(224, 64), models.resnet152, metrics=accuracy).mixup()
```

```
Downloading: "https://download.pytorch.org/models/resnet152-b121ed2d.pth" to /root/.cache/torch/checkpoints/resnet152-b12
1ed2d.pth
100%|██████████| 230M/230M [00:04<00:00, 57.3MB/s]
```

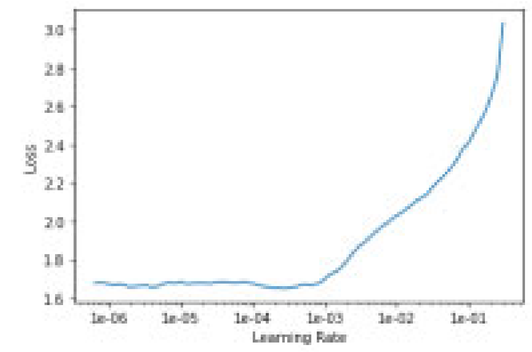
Ran the below code twice

In [0]:
```
learn3.fit_one_cycle(4)
```

epoch	train_loss	valid_loss	accuracy	time
0	1.928610	1.465155	0.501656	03:00
1	1.940260	1.460222	0.511589	02:59
2	1.824344	1.368909	0.538079	03:00
3	1.694844	1.347014	0.539735	03:00

In [0]:
```
learn3.unfreeze()
learn3.lr_find()
learn3.recorder.plot()
```

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



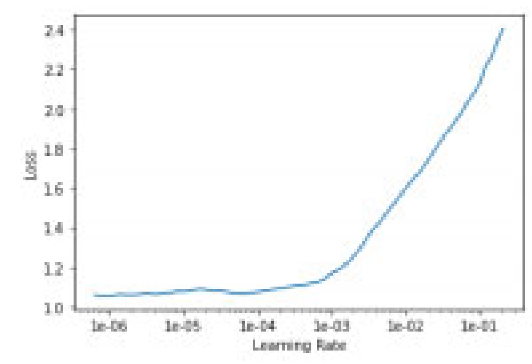
In [0]:
```
learn3.fit_one_cycle(8, max_lr=3e-4)
```

epoch	train_loss	valid_loss	accuracy	time
0	1.679416	1.335181	0.554636	03:04
1	1.774198	1.764813	0.432119	03:05
2	1.785291	1.735680	0.457781	03:04
3	1.712331	1.400715	0.544702	03:06
4	1.581151	1.355430	0.545530	03:05
5	1.417123	1.318366	0.570364	03:06
6	1.225719	1.273213	0.600166	03:07
7	1.126336	1.273245	0.608444	03:05

In [0]:
```
learn3.save('resnet152_phase1')
```

In [0]:
```
learn3.unfreeze()
learn3.lr_find()
learn3.recorder.plot()
```

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



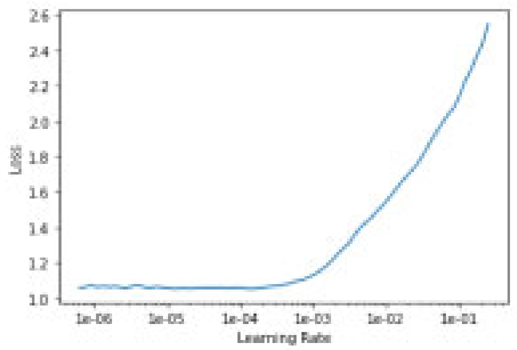
In [0]:
```
learn3.fit_one_cycle(8, max_lr=1e-06)
```

epoch	train_loss	valid_loss	accuracy	time
0	1.089854	1.271377	0.604305	03:06
1	1.074826	1.279301	0.606788	03:05
2	1.098520	1.267761	0.608444	03:06
3	1.070563	1.268897	0.597682	03:07
4	1.079534	1.269859	0.603477	03:05
5	1.093139	1.268313	0.609272	03:04
6	1.083919	1.274353	0.605960	03:06
7	1.070323	1.264602	0.605960	03:06

In [0]:
```
learn3.save('resnet152_phase2')
```

In [0]:
```
learn3.unfreeze()
learn3.lr_find()
learn3.recorder.plot()
```

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



In [0]:
```
learn3.fit_one_cycle(8, max_lr=1e-04)
```

epoch	train_loss	valid_loss	accuracy	time
0	1.094313	1.298552	0.596026	03:07
1	1.108809	1.453950	0.568709	03:06
2	1.106317	1.530727	0.533113	03:06
3	1.068021	1.471262	0.567053	03:07
4	1.005413	1.420844	0.591887	03:08
5	0.947496	1.408078	0.584437	03:10
6	0.926861	1.385136	0.586093	03:06
7	0.898268	1.369511	0.595199	03:09

Accuracy reached almost 60%

Using pretrained weights of Places365

In [0]:
```
!!wget http://places2.csail.mit.edu/models_places365/resnet50_places365.pth.tar -P data/models/
```

Out[0]:
```
['--2019-12-29 19:10:14--  http://places2.csail.mit.edu/models_places365/resnet50_places365.pth.tar',
 'Resolving places2.csail.mit.edu (places2.csail.mit.edu)... 128.30.100.255',
 'Connecting to places2.csail.mit.edu (places2.csail.mit.edu)|128.30.100.255|:80... connected.',
 'HTTP request sent, awaiting response... 200 OK',
 'Length: 97270159 (93M) [application/x-tar]',
 'Saving to: ‘data/models/resnet50_places365.pth.tar.1’',
 '',
 '',
 '          resnet50_   0%[                    ]       0  --.-KB/s               ',
 '         resnet50_p   0%[                    ]  21.84K   109KB/s               ',
 '        resnet50_pl   0%[                    ]  50.12K   125KB/s               ',
 '       resnet50_pla   0%[                    ]  94.87K   157KB/s               ',
 '      resnet50_plac   0%[                    ] 185.37K   230KB/s               ',
 '     resnet50_place   0%[                    ] 365.16K   363KB/s               ',
 '    resnet50_places   0%[                    ] 696.05K   576KB/s               ',
 '   resnet50_places3   1%[                    ]   1.22M   887KB/s               ',
 '  resnet50_places36   2%[                    ]   1.91M  1.19MB/s               ',
 ' resnet50_places365   2%[                    ]   2.76M  1.52MB/s               ',
 'resnet50_places365.   4%[                    ]   3.86M  1.92MB/s               ',
 'esnet50_places365.p   5%[>                   ]   5.20M  2.35MB/s               ',
 'snet50_places365.pt   7%[>                   ]   6.88M  2.85MB/s               ',
 'net50_places365.pth   9%[>                   ]   9.21M  3.52MB/s               ',
 'et50_places365.pth.  12%[=>                  ]  11.86M  4.21MB/s               ',
 't50_places365.pth.t  15%[==>                 ]  14.71M  4.87MB/s    eta 16s    ',
 '50_places365.pth.ta  19%[==>                 ]  17.71M  5.50MB/s    eta 16s    ',
 '0_places365.pth.tar  21%[===>                ]  20.29M  5.91MB/s    eta 16s    ',
 '_places365.pth.tar.  24%[===>                ]  22.79M  6.26MB/s    eta 16s    ',
 'places365.pth.tar.1  27%[====>               ]  25.32M  6.60MB/s    eta 16s    ',
 'laces365.pth.tar.1   29%[====>               ]  27.82M  6.88MB/s    eta 9s     ',
 'aces365.pth.tar.1    32%[=====>              ]  30.32M  7.50MB/s    eta 9s     ',
 'ces365.pth.tar.1     35%[======>             ]  32.83M  8.11MB/s    eta 9s     ',
 'es365.pth.tar.1      38%[======>             ]  35.33M  8.72MB/s    eta 9s     ',
 's365.pth.tar.1       40%[=======>            ]  37.84M  9.32MB/s    eta 9s     ',
 '365.pth.tar.1        43%[=======>            ]  40.36M  9.89MB/s    eta 7s     ',
 '65.pth.tar.1         46%[========>           ]  43.05M  10.5MB/s    eta 7s     ',
 '5.pth.tar.1          49%[========>           ]  45.55M  11.0MB/s    eta 7s     ',
 '.pth.tar.1           51%[=========>          ]  48.08M  11.4MB/s    eta 7s     ',
 'pth.tar.1            54%[=========>          ]  50.58M  11.8MB/s    eta 7s     ',
 'th.tar.1             57%[==========>         ]  53.09M  12.2MB/s    eta 5s     ',
 'h.tar.1              59%[==========>         ]  55.59M  12.4MB/s    eta 5s     ',
 '.tar.1               62%[===========>        ]  58.11M  12.7MB/s    eta 5s     ',
 'tar.1                65%[============>       ]  60.62M  12.7MB/s    eta 5s     ',
 'ar.1                 68%[============>       ]  63.12M  12.7MB/s    eta 5s     ',
 'r.1                  70%[=============>      ]  65.61M  12.6MB/s    eta 3s     ',
 '.1                   73%[=============>      ]  68.13M  12.5MB/s    eta 3s     ',
 '1                    76%[==============>     ]  70.64M  12.5MB/s    eta 3s     ',
 '                     78%[==============>     ]  73.13M  12.5MB/s    eta 3s     ',
 '                  r  81%[===============>    ]  75.64M  12.5MB/s    eta 3s     ',
 '                 re  84%[===============>    ]  78.16M  12.5MB/s    eta 2s     ',
 '                res  86%[================>   ]  80.67M  12.5MB/s    eta 2s     ',
 '               resn  89%[================>   ]  83.18M  12.5MB/s    eta 2s     ',
 '              resne  92%[=================>  ]  85.70M  12.5MB/s    eta 2s     ',
 '             resnet  95%[==================> ]  88.21M  12.5MB/s    eta 2s     ',
 '            resnet5  97%[==================> ]  90.72M  12.5MB/s    eta 0s     ',
 'resnet50_places365. 100%[===================>]  92.76M  12.9MB/s    in 9.1s    ',
 '',
 '2019-12-29 19:10:24 (10.2 MB/s) - ‘data/models/resnet50_places365.pth.tar.1’ saved [97270159/97270159]',
 '']
```

In [0]:
```
places_res50 = torch.load('/content/data/models/resnet50_places365.pth.tar',  map_location=lambda storage, loc: storage)
```

In [0]:
```
default_res50 = models.resnet50()
state_dict = places_res50['state_dict']
new_state_dict = OrderedDict()
```

In [0]:
```
for key in state_dict.keys():
  new_state_dict[key[7:]]= state_dict[key]
```

In [0]:
```
default_res50.fc = torch.nn.Linear(2048, 365) # Matching with default res50 dense layer
default_res50.load_state_dict(new_state_dict)
```

Out[0]: <All keys matched successfully>

In [0]:
```
def new_resnet(prtn):
  return default_res50
```

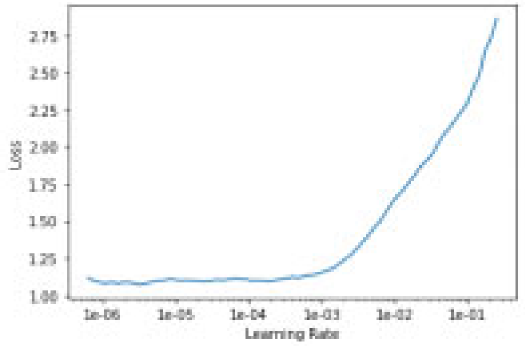
In [0]:
```
learn233 = cnn_learner(get_data(224, 64), new_resnet, metrics=[error_rate, accuracy])
```

In [0]:
```
learn233.fit_one_cycle(4)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	2.283662	1.887077	0.525662	0.474338	03:26
1	1.723866	1.460125	0.447848	0.552152	03:23
2	1.412615	1.358377	0.437914	0.562086	03:23
3	1.197710	1.341950	0.425497	0.574503	03:24

In [0]:
```
learn233.unfreeze()
learn233.lr_find()
learn233.recorder.plot()
```

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



In [0]:
```
learn233.fit_one_cycle(8, max_lr=slice(1e-06,1e-04))
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.127274	1.323371	0.418046	0.581954	03:26
1	1.126416	1.326906	0.422185	0.577815	03:25
2	1.103190	1.317386	0.422185	0.577815	03:25
3	1.065696	1.318328	0.415563	0.584437	03:26
4	1.027151	1.321286	0.421358	0.578642	03:27
5	1.000289	1.308829	0.422185	0.577815	03:27
6	0.997371	1.318654	0.423841	0.576159	03:27
7	0.985281	1.313385	0.424669	0.575331	03:28

In [0]:
```
learn233.save('ne_resnet_1')
```

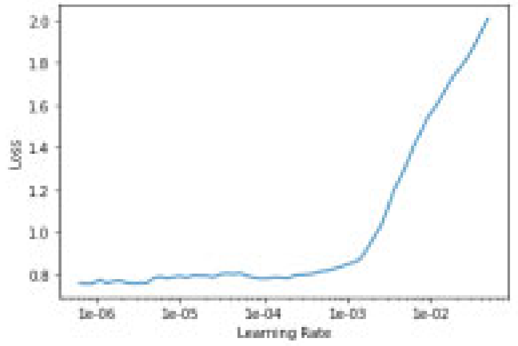
In [0]:
```
learn233 = cnn_learner(get_data(224, 32), new_resnet, metrics=[error_rate, accuracy]).load('ne_resnet_1')
```

In [0]:
```
learn233.fit_one_cycle(10)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.258776	0.971517	0.325331	0.674669	03:28
1	1.381434	1.195777	0.411424	0.588576	03:26
2	1.368047	1.335075	0.441225	0.558775	03:26
3	1.333285	1.334576	0.452815	0.547185	03:26
4	1.254198	1.267678	0.436258	0.563742	03:24
5	1.116826	1.285337	0.424669	0.575331	03:25
6	1.015423	1.247776	0.416391	0.583609	03:25
7	0.954135	1.227312	0.418874	0.581126	03:26
8	0.836304	1.232664	0.417219	0.582781	03:27
9	0.820164	1.228195	0.413907	0.586093	03:26

In [0]:
```
learn233.unfreeze()
learn233.lr_find()
learn233.recorder.plot()
```

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



In [0]:
```
learn233.fit_one_cycle(3, max_lr=1e-04)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.977339	1.345697	0.424669	0.575331	03:27
1	0.859146	1.269117	0.415563	0.584437	03:28
2	0.588134	1.254941	0.399834	0.600166	03:28

In [0]:
```
learn233.save('ne_resnet_2')
```

A sudden increase in the accuracy because of reducing the batch size to 16

In [0]:
```
learn233 = cnn_learner(get_data(224, 16), new_resnet, metrics=[error_rate, accuracy]).load('ne_resnet_2')
```

In [0]:
```
learn233.fit_one_cycle(2)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.168207	0.642344	0.199503	0.800497	03:34
1	0.958618	0.556656	0.172185	0.827815	03:35

In [0]:
```
learn233.save('ne_resnet_3')
```

In [0]:
```
learn233 = cnn_learner(get_data(224, 16), new_resnet, metrics=[error_rate, accuracy]).load('ne_resnet_3').mixup()
```

In [0]:
```
learn233.fit_one_cycle(4)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.694418	0.656723	0.215232	0.784768	03:34
1	1.657834	0.672429	0.210265	0.789735	03:33
2	1.572796	0.669547	0.221026	0.778974	03:33
3	1.456823	0.642834	0.202815	0.797185	03:33

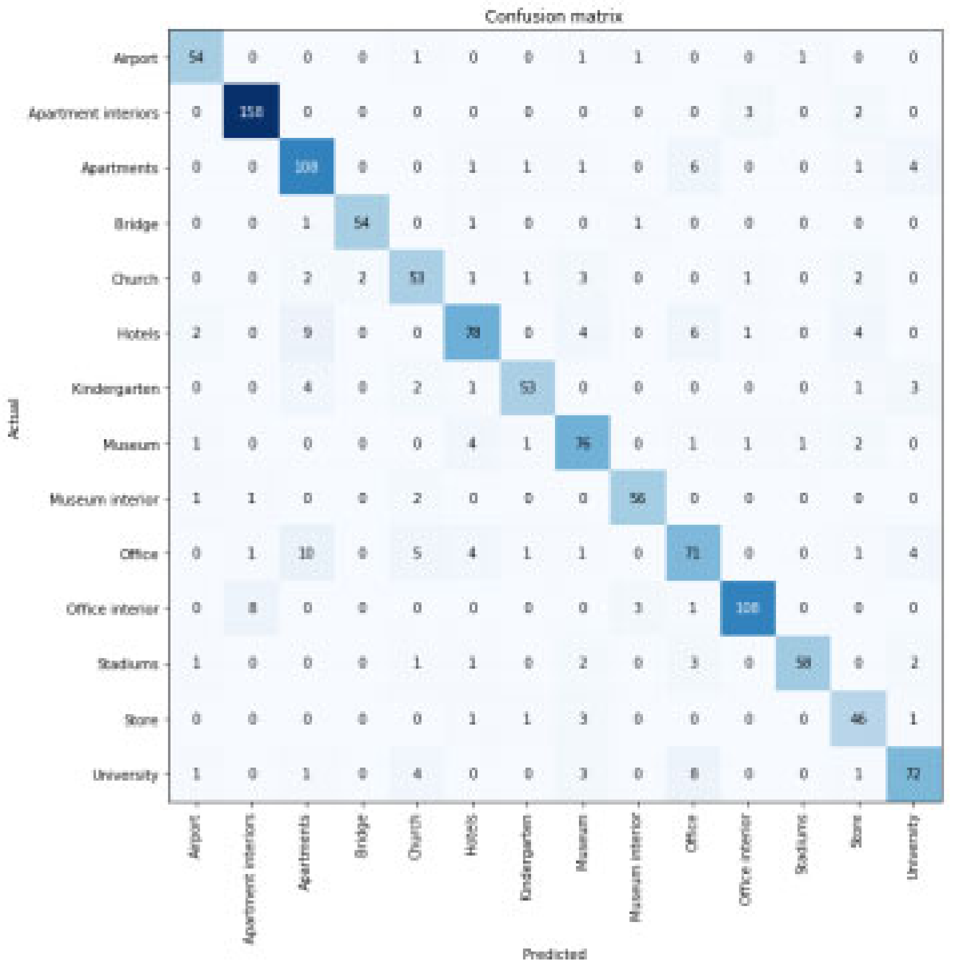
In [0]:
```
learn233.save('ne_resnet_4')
```

Using 82.7% accuracy for further analysis

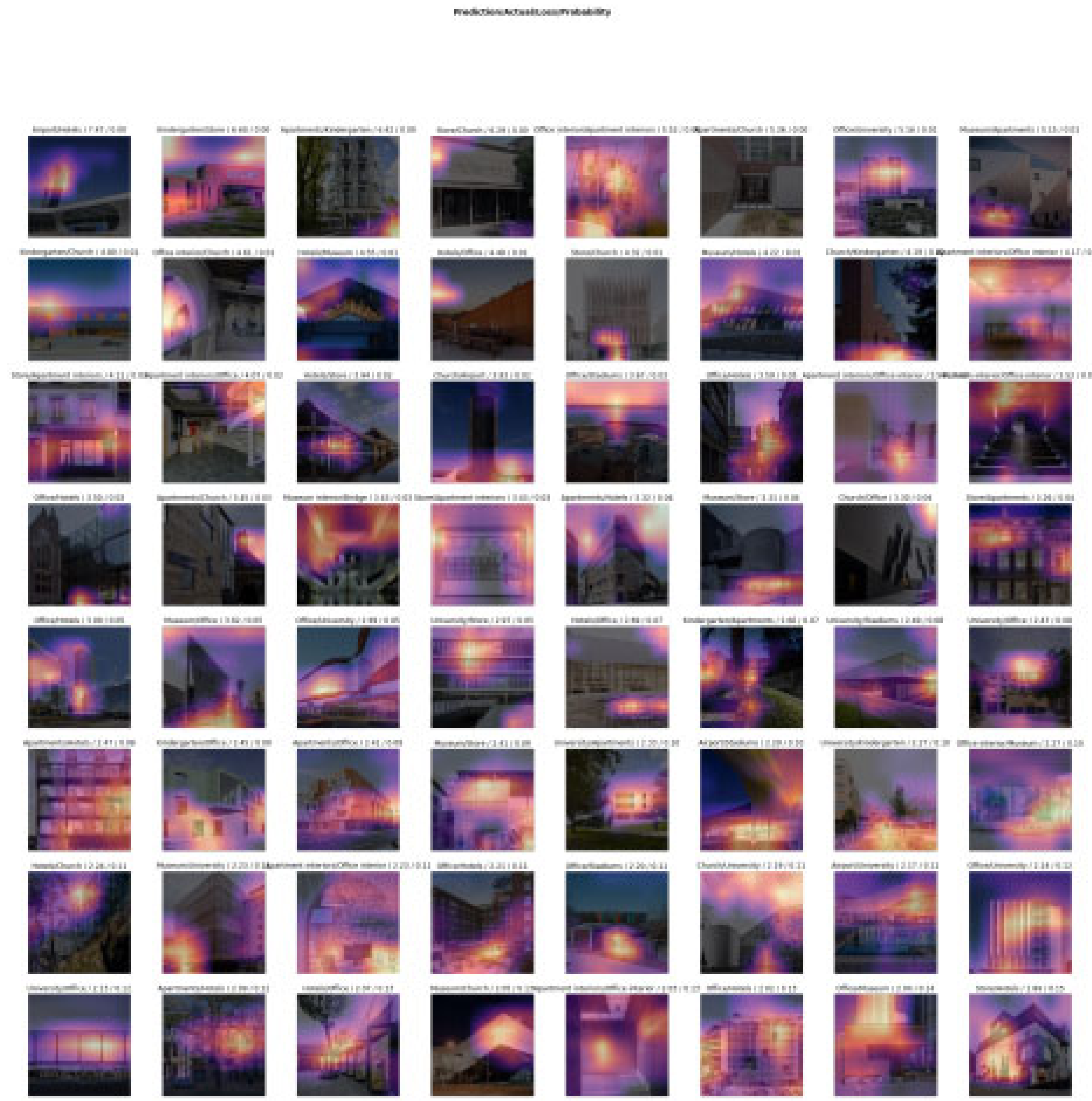
In [0]:
```
learn233.load('ne_resnet_3');
```

In [0]:
```
interp = ClassificationInterpretation.from_learner(learn233)
```

In [0]:
```
interp.plot_confusion_matrix(figsize=(10,10))
```



In [0]:
```
interp.plot_top_losses(64, figsize=(32,30), largest=True, heatmap=True, alpha=0.6)
```

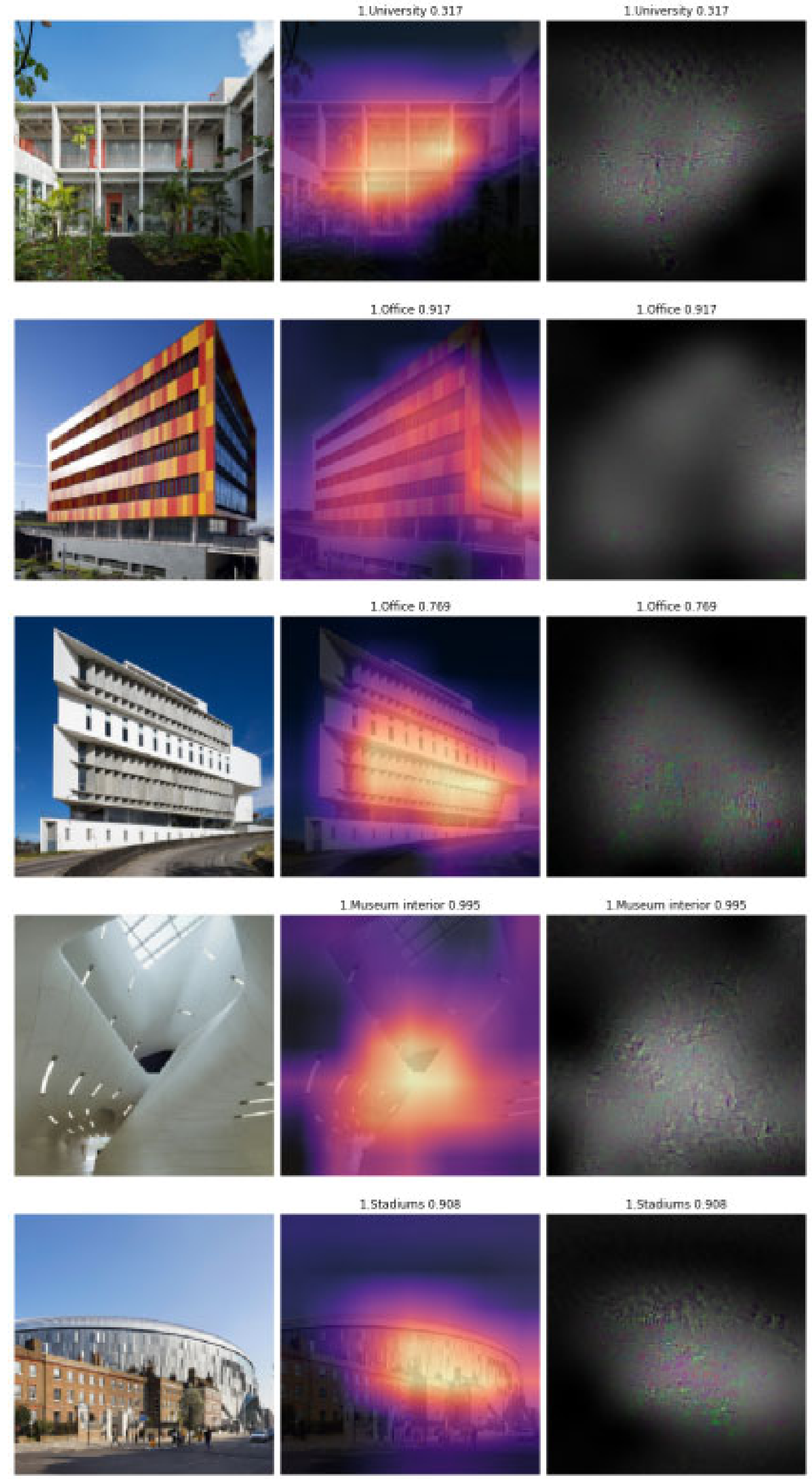


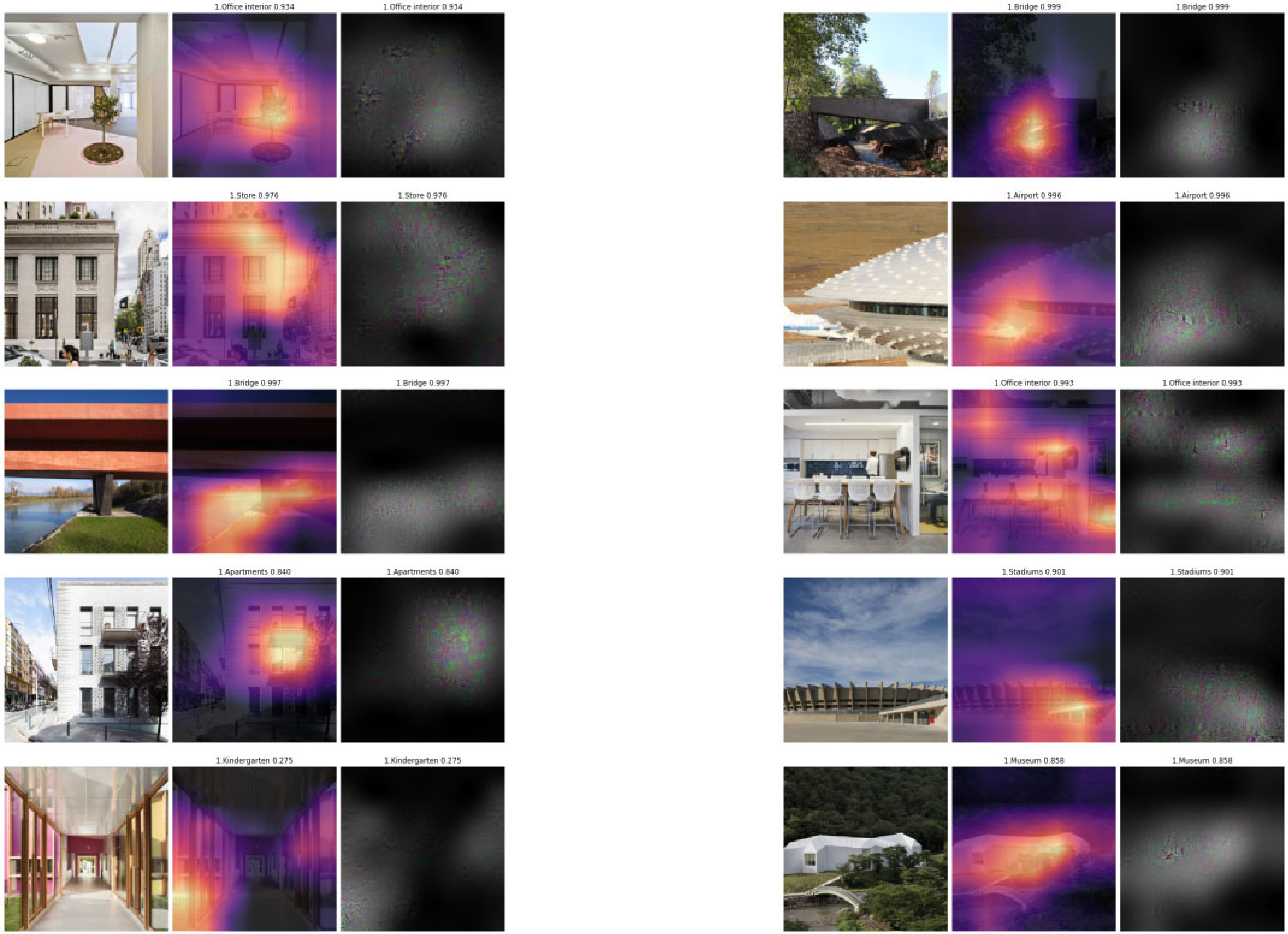
In [0]:
```
interp.most_confused(min_val=2)
```

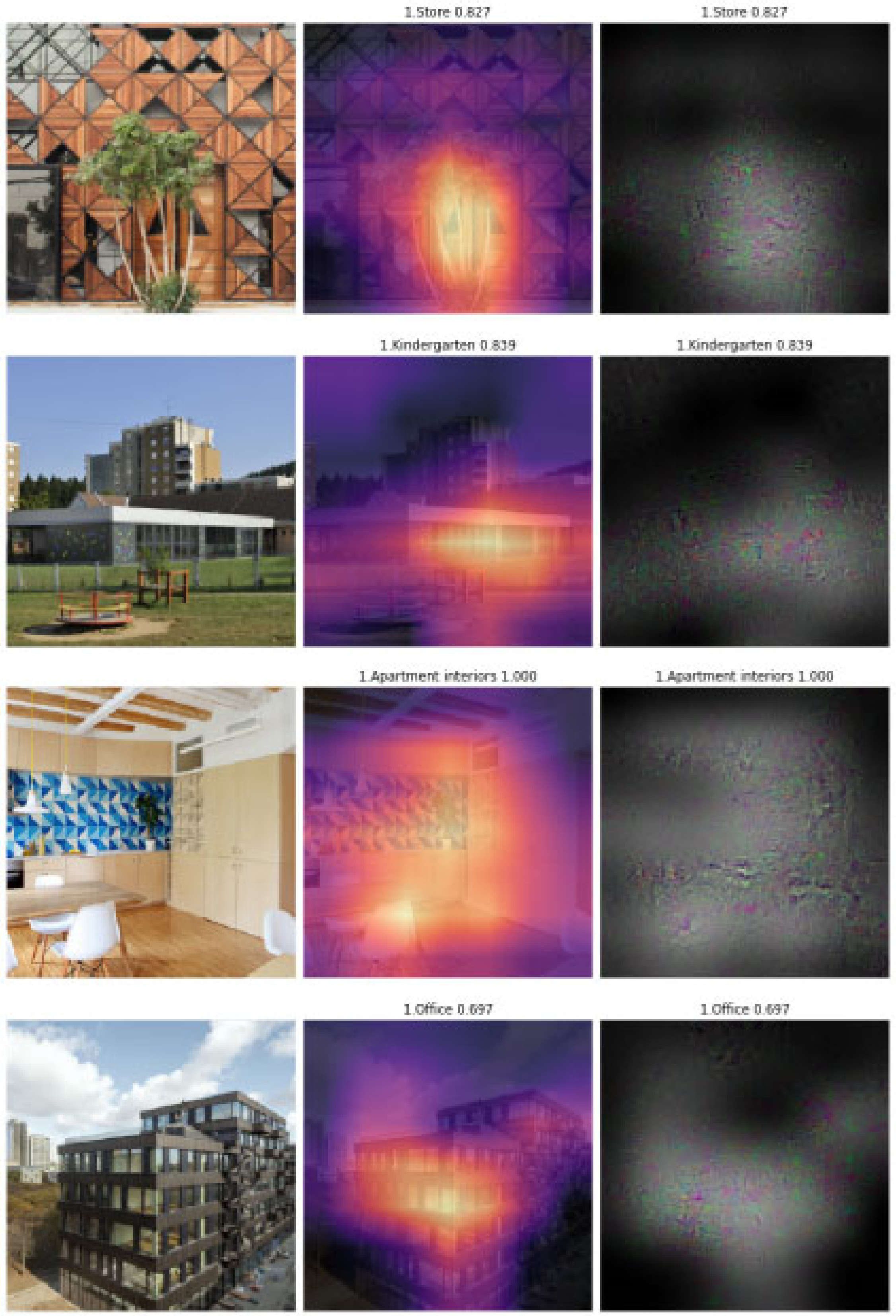
Out[0]:
```
[('Office', 'Apartments', 10),
 ('Hotels', 'Apartments', 9),
 ('Office interior', 'Apartment interiors', 8),
 ('University', 'Office', 8),
 ('Apartments', 'Office', 6),
 ('Hotels', 'Office', 6),
 ('Office', 'Church', 5),
 ('Apartments', 'University', 4),
 ('Hotels', 'Museum', 4),
 ('Hotels', 'Store', 4),
 ('Kindergarten', 'Apartments', 4),
 ('Museum', 'Hotels', 4),
 ('Office', 'Hotels', 4),
 ('Office', 'University', 4),
 ('University', 'Church', 4),
 ('Apartment interiors', 'Office interior', 3),
 ('Church', 'Museum', 3),
 ('Kindergarten', 'University', 3),
 ('Office interior', 'Museum interior', 3),
 ('Stadiums', 'Office', 3),
 ('Store', 'Museum', 3),
 ('University', 'Museum', 3),
 ('Apartment interiors', 'Store', 2),
 ('Church', 'Apartments', 2),
 ('Church', 'Bridge', 2),
 ('Church', 'Store', 2),
 ('Hotels', 'Airport', 2),
 ('Kindergarten', 'Church', 2),
 ('Museum', 'Store', 2),
 ('Museum interior', 'Church', 2),
 ('Stadiums', 'Museum', 2),
 ('Stadiums', 'University', 2)]
```

Using Grad Cam for identifying discriminative regions used by a particular class

In [0]:
```
for i in range(1,20):
  gcam = GradCam.from_interp(learn233,interp,i, include_label=True)
  gcam.plot()
```




1.Office interior 0.934
1.Office interior 0.934
1.Bridge 0.999
1.Bridge 0.999
1.Store 0.976
1.Store 0.976
1.Airport 0.996
1.Airport 0.996
1.Bridge 0.997
1.Bridge 0.997
1.Office interior 0.993
1.Office interior 0.993
1.Apartments 0.840
1.Apartments 0.840
1.Stadiums 0.901
1.Stadiums 0.901
1.Kindergarten 0.275
1.Kindergarten 0.275
1.Museum 0.858
1.Museum 0.858



Visualizing the layers

Looking through the progression of layers, we can see how the model breaks the bridge in the image apart from the background and convolves the image down

In [0]:
```
m = learn233.model.eval()
```

In [0]:
```
class SaveFeatures_ng():
    features=None
    def __init__(self, m): self.hook = m.register_forward_hook(self.hook_fn)
    def hook_fn(self, module, input, output): self.features = output.detach()
    def remove(self): self.hook.remove()
```

In [0]:
```
sfs = [SaveFeatures_ng(children(m)[0][i]) for i in range(len(children(m)[0]))]
```

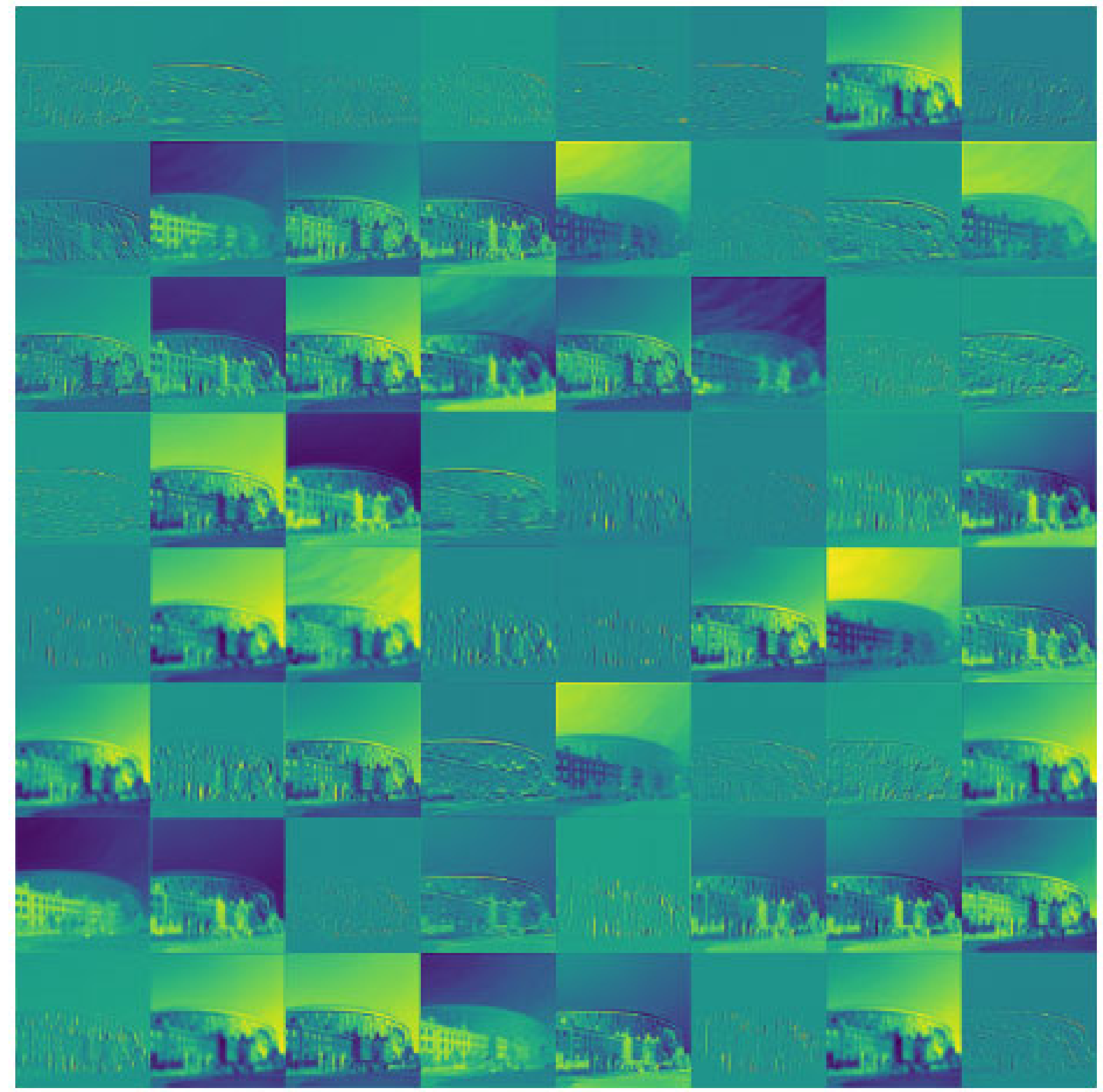
In [0]:
```
x, y = next(iter(learn233.data.valid_dl))
```

In [0]:
```
p = m(x[5][None])
```

In [0]:
```
[i.features.shape for i in sfs]
```

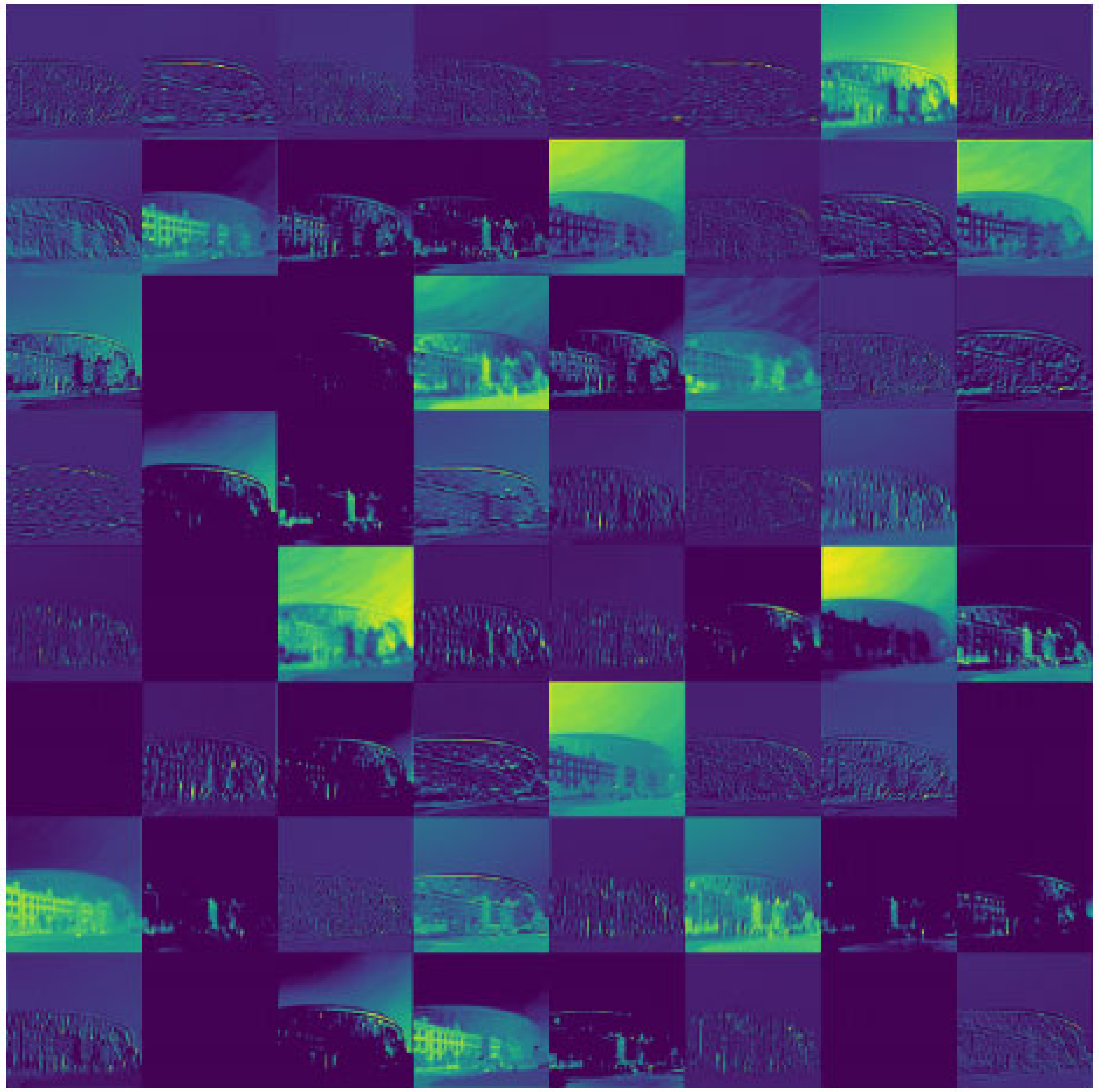
Out[0]:
```
[torch.Size([1, 64, 112, 112]),
 torch.Size([1, 64, 112, 112]),
 torch.Size([1, 64, 112, 112]),
 torch.Size([1, 64, 56, 56]),
 torch.Size([1, 256, 56, 56]),
 torch.Size([1, 512, 28, 28]),
 torch.Size([1, 1024, 14, 14]),
 torch.Size([1, 2048, 7, 7])]
```

In [0]:
```
activs = sfs[0].features.detach().cpu().numpy()[0]
fig, axes = plt.subplots(8,8, figsize=(15,15))
fig.subplots_adjust(hspace=0.0, wspace=0, left=0, right=1, top=1, bottom=0)
for i, ax in enumerate(axes.flat):
    ax.imshow(activs[i])
    ax.set_axis_off()
```



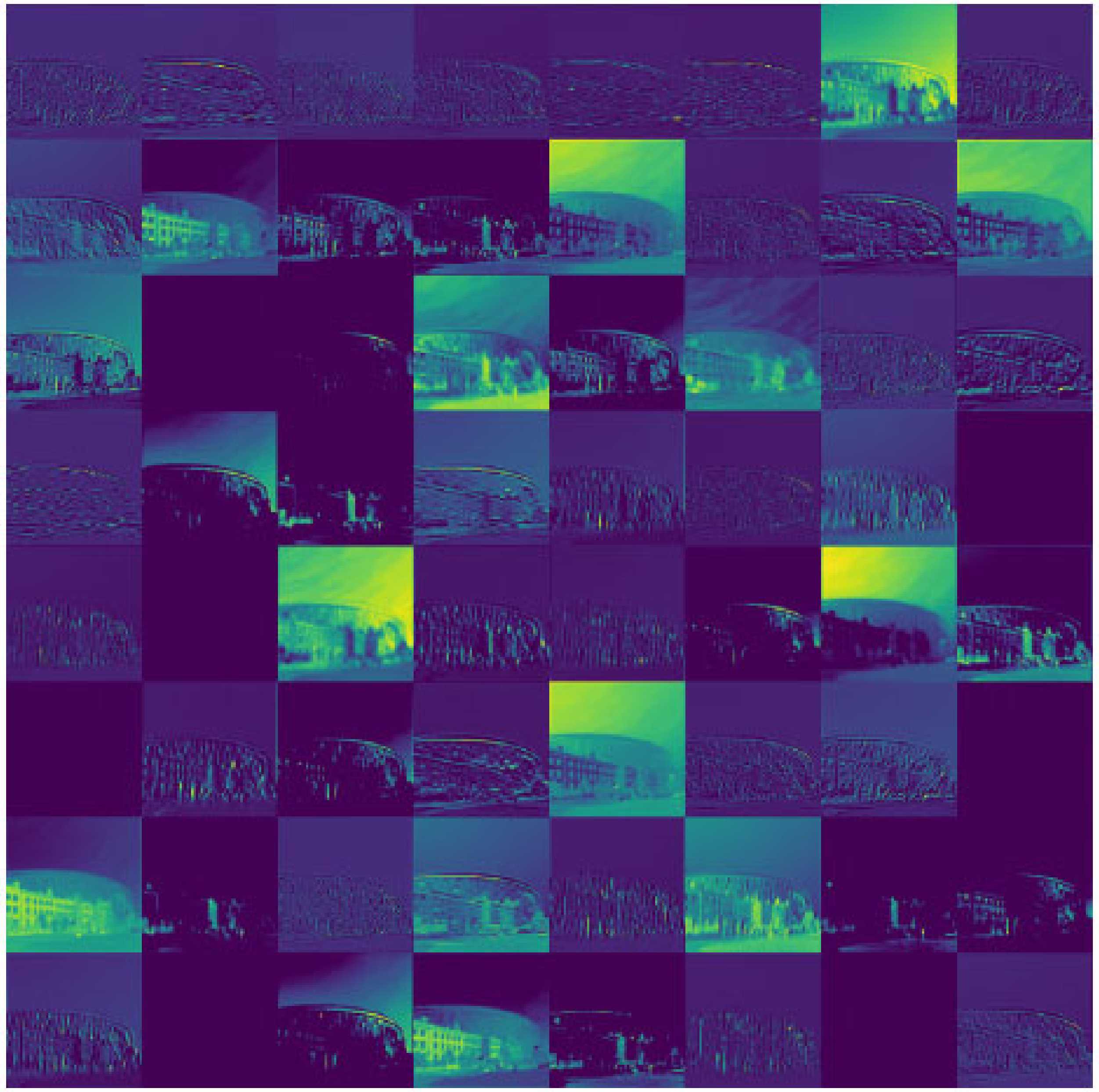
In [0]:

```
activs = sfs[1].features.detach().cpu().numpy()[0]
fig, axes = plt.subplots(8,8, figsize=(15,15))
fig.subplots_adjust(hspace=0.0, wspace=0, left=0, right=1, top=1, bottom=0)
for i, ax in enumerate(axes.flat):
    ax.imshow(activs[i])
    ax.set_axis_off()
```

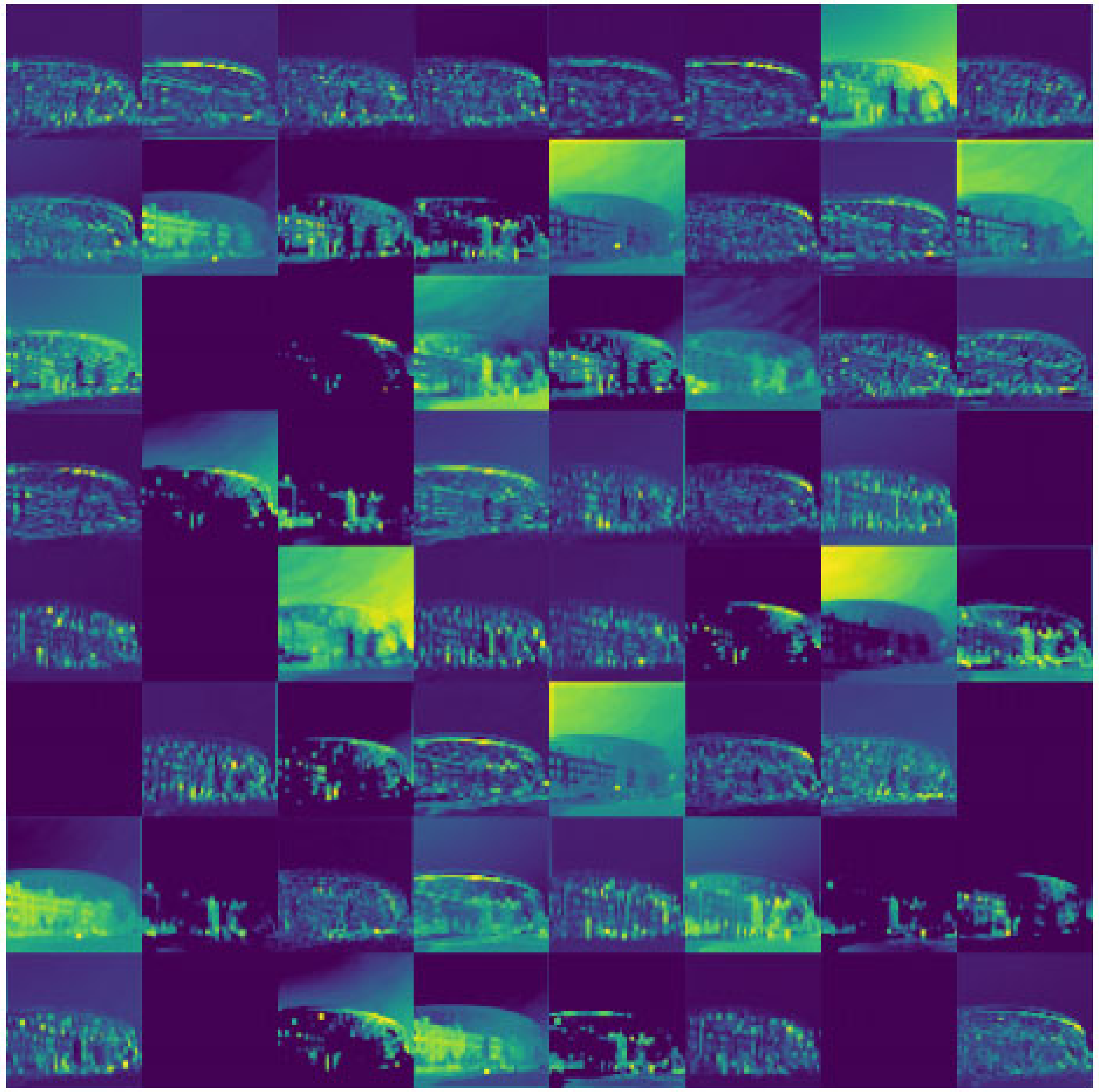


In [0]:

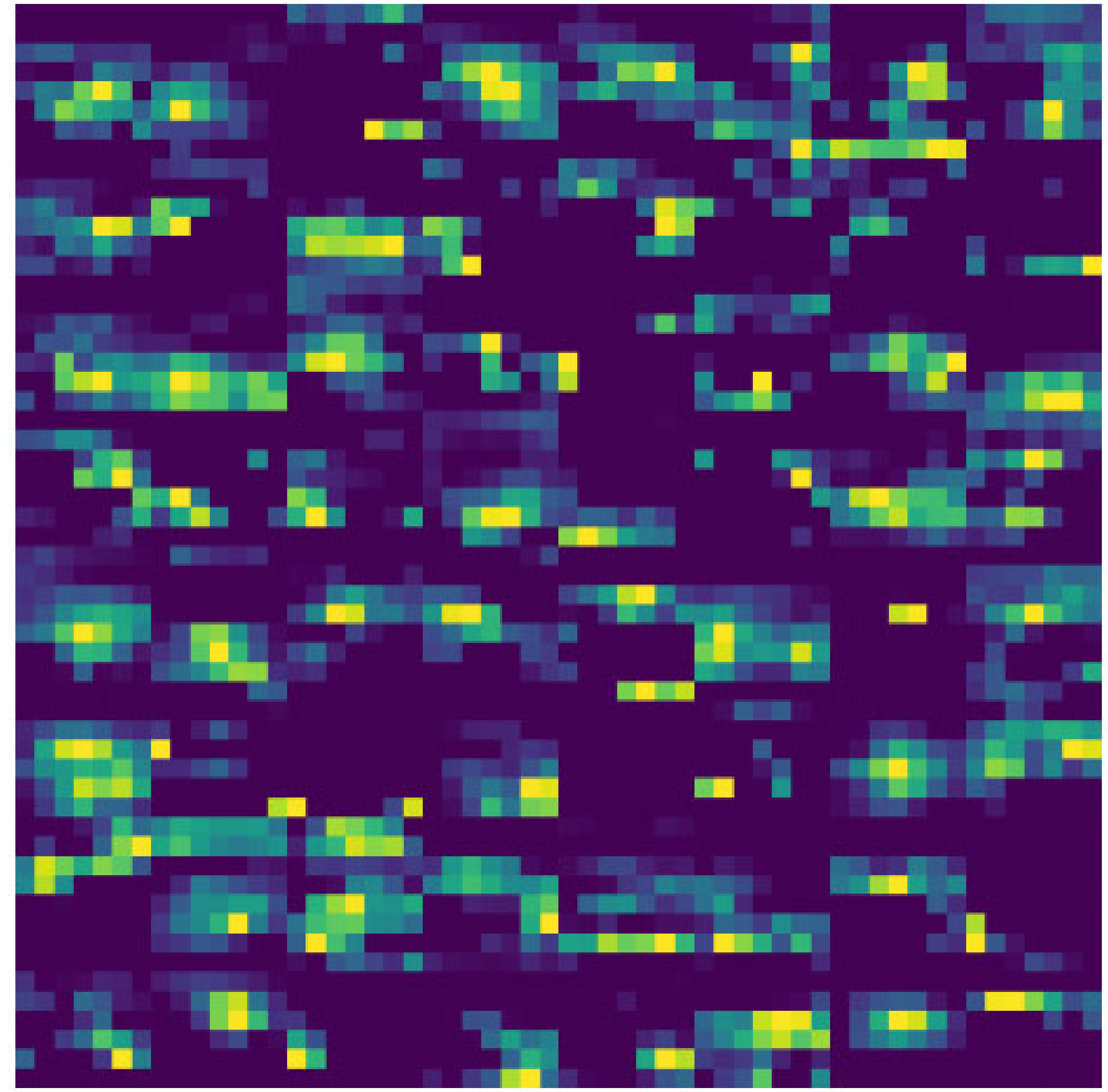
```
activs = sfs[2].features.detach().cpu().numpy()[0]
fig, axes = plt.subplots(8,8, figsize=(15,15))
fig.subplots_adjust(hspace=0.0, wspace=0, left=0, right=1, top=1, bottom=0)
for i, ax in enumerate(axes.flat):
    ax.imshow(activs[i])
    ax.set_axis_off()
```



In [0]:
```python
activs = sfs[3].features.detach().cpu().numpy()[0]
fig, axes = plt.subplots(8,8, figsize=(15,15))
fig.subplots_adjust(hspace=0.0, wspace=0, left=0, right=1, top=1, bottom=0)
for i, ax in enumerate(axes.flat):
    ax.imshow(activs[i])
    ax.set_axis_off()
```



In [0]:
```python
activs = sfs[7].features.detach().cpu().numpy()[0]
fig, axes = plt.subplots(8,8, figsize=(15,15))
fig.subplots_adjust(hspace=0.0, wspace=0, left=0, right=1, top=1, bottom=0)
for i, ax in enumerate(axes.flat):
    ax.imshow(activs[i])
    ax.set_axis_off()
```



TSNE

In [0]:
```python
log_preds,y = learn233.TTA()
```

In [0]:
```python
log_preds[500]
```

Out[0]:
```
tensor([3.0233e-04, 6.8169e-03, 8.1317e-01, 2.6513e-04, 5.9113e-03, 8.5206e-02,
        2.5864e-02, 1.6664e-03, 4.8960e-04, 4.0825e-02, 3.7604e-03, 1.0820e-02,
        6.6200e-04, 4.2439e-03])
```

In [0]:
```python
F.softmax(log_preds[500], dim=0)
```

Out[0]:
```
tensor([0.0648, 0.0652, 0.1460, 0.0648, 0.0651, 0.0705, 0.0664, 0.0648, 0.0648,
        0.0674, 0.0650, 0.0654, 0.0648, 0.0650])
```

t-SNE is performed on model's output vectors. As these vectors are from the final classification, we would expect them to cluster well.

In [0]:
```python
probs_trans = manifold.TSNE(n_components=2, perplexity=15).fit_transform(log_preds)
```

In [0]:
```python
prob_df = pd.DataFrame(np.concatenate((probs_trans, y[:,None]), axis=1), columns=['x','y','labels'])
```

```
In [0]: g = sns.lmplot('x', 'y', data=prob_df, hue='labels', fit_reg=False, legend=True, height=20)
```