```
from fastai import *
from fastai.vision import *
import matplotlib.pyplot as plt
```

```
! git clone https://github.com/ankitbhadu/deep-learning.git
```

```
Cloning into 'deep-learning'...
remote: Enumerating objects: 27581, done.
remote: Counting objects: 100% (27581/27581), done.
remote: Compressing objects: 100% (27576/27576), done.
remote: Total 27581 (delta 2), reused 27578 (delta 2), pack-reused 0
Receiving objects: 100% (27581/27581), 357.27 MiB | 50.80 MiB/s, done.
Resolving deltas: 100% (2/2), done.
Checking out files: 100% (27566/27566), done.
```

```
cd /content/deep-learning/
```

```
/content/deep-learning
```

```
!rm -rf README.md
```

```
cd /content/deep-learning/CV
```

```
/content/deep-learning/CV
```

```
path = Path('/content/deep-learning/CV/Data')
```

```
fnames = get_image_files(path)
```

```
fnames[:5]
```

```
[PosixPath('/content/deep-learning/CV/Data/parasitized (9637).png'),
 PosixPath('/content/deep-learning/CV/Data/uninfected (7698).png'),
 PosixPath('/content/deep-learning/CV/Data/parasitized (7642).png'),
 PosixPath('/content/deep-learning/CV/Data/uninfected (2067).png'),
 PosixPath('/content/deep-learning/CV/Data/parasitized (4118).png')]
```

```
np.random.seed(2)
pat= r'/([^/])\s\S+.png$'
```

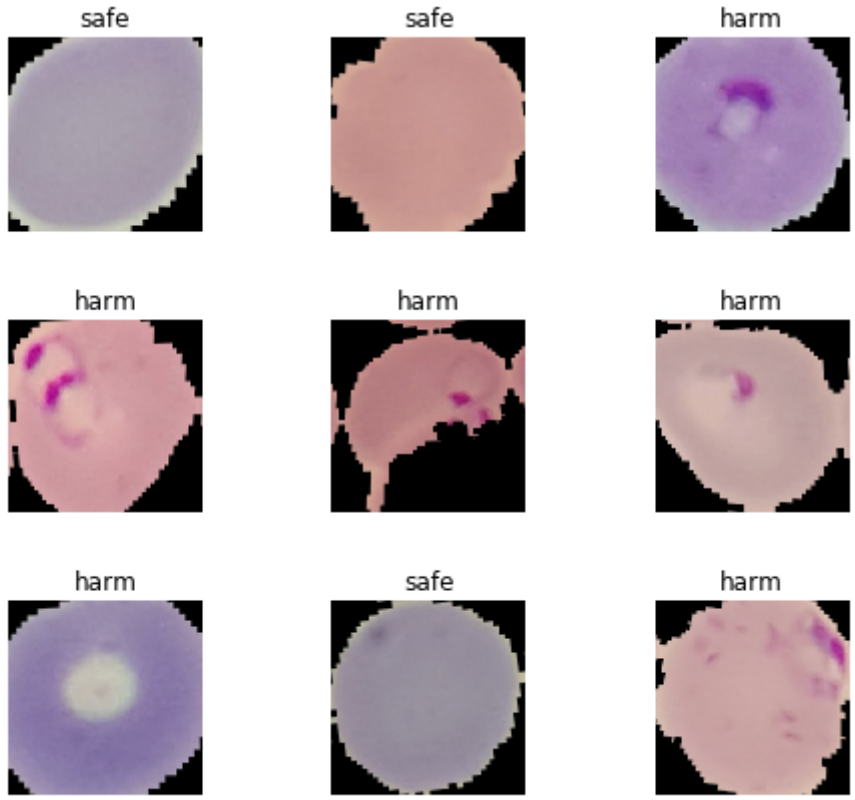
```
data = ImageDataBunch.from_name_func(path, fnames, label_func = lambda x: 'safe' if '/uninfected
data.normalize(imagenet_stats)
```

```
ImageDataBunch;

Train: LabelList (16000 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image
y: CategoryList
harm,safe,safe,safe,safe
Path: /content/deep-learning/CV/Data;

Valid: LabelList (4000 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image
y: CategoryList
harm,harm,safe,harm,harm
```

```
data.show_batch(rows=3, figsize=(7,6))
```

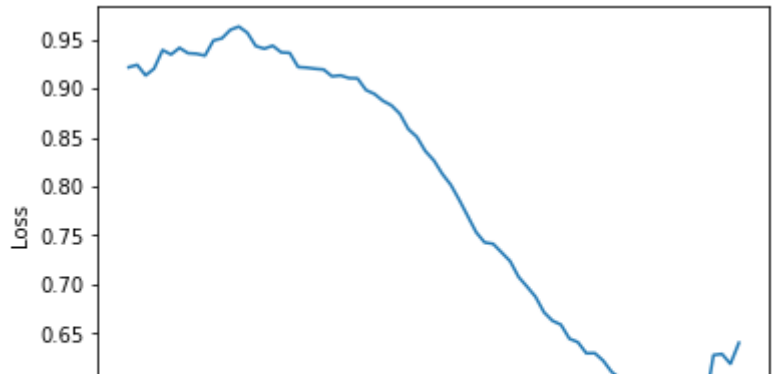



```
learn = cnn_learner(data,models.resnet50,metrics=error_rate)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.ca
100%|██████████| 102502400/102502400 [00:00<00:00, 128370674.61it/s]
```

```
learn.lr_find()
learn.recorder.plot()
```

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

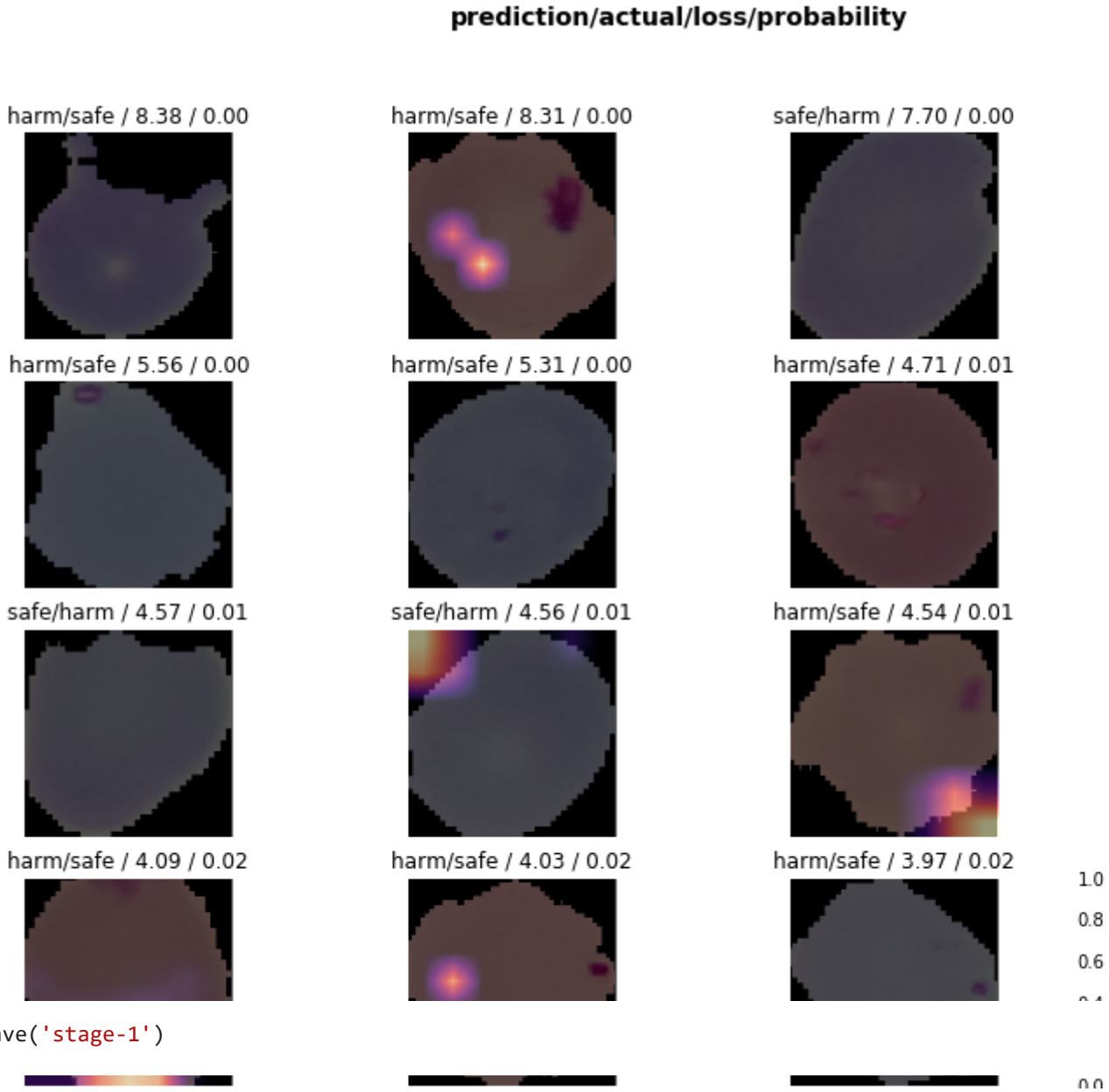



```
learn.fit_one_cycle(8,max_lr=slice(1e-5,1e-2))
```

epoch	train_loss	valid_loss	error_rate	time
0	0.239269	0.201722	0.073000	02:31
1	0.196275	0.159721	0.053000	02:34
2	0.162477	0.139411	0.047250	02:31
3	0.150956	0.117020	0.043500	02:33
4	0.137664	0.118327	0.043750	02:37
5	0.133781	0.111159	0.042750	02:31
6	0.117217	0.109023	0.042000	02:32
7	0.125308	0.110227	0.041000	02:31

```
interp = ClassificationInterpretation.from_learner(learn)
```

```
interp.plot_top_losses(15,figsize=(15,10))
```




```
learn.save('stage-1')
```

```
learn.unfreeze()
learn.fit_one_cycle(8,max_lr=slice(1e-5,1e-2))
```

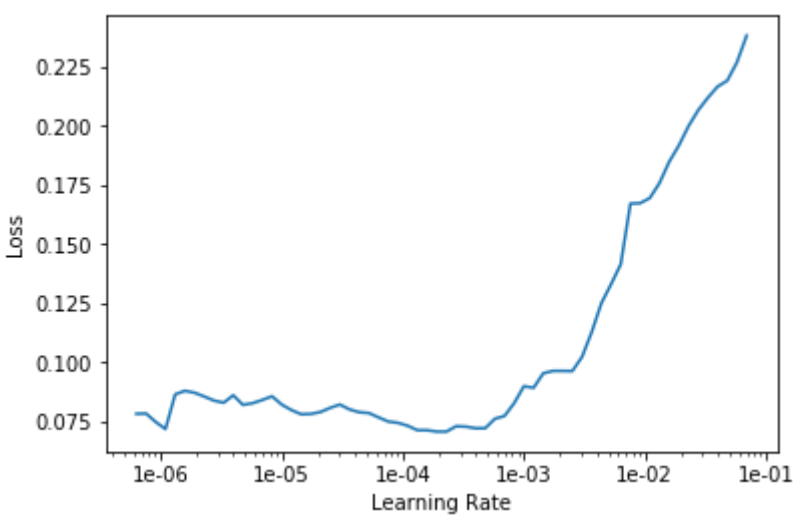
epoch	train_loss	valid_loss	error_rate	time
0	0.131713	0.155745	0.050500	03:11
1	0.144311	0.126511	0.043500	03:09
2	0.147690	0.126300	0.041500	03:09
3	0.123391	0.306476	0.045250	03:09
4	0.124153	0.100258	0.034750	03:09
5	0.106530	0.087759	0.031500	03:09
6	0.086368	0.085630	0.032000	03:09
7	0.081019	0.084219	0.031250	03:10

```
learn.save('stage-2')
```

```
learn.lr_find()
```

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

```
learn.recorder.plot()
```



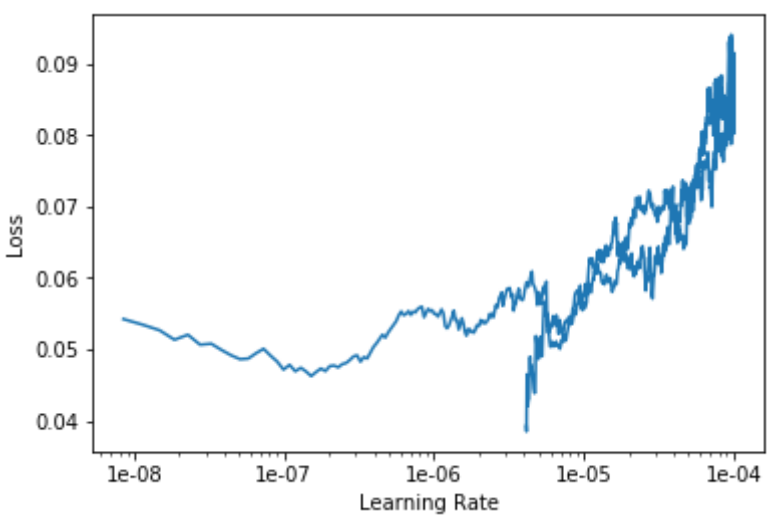

```
learn.load('stage-2')
```

```
learn.fit_one_cycle(6,max_lr=1e-4)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.072677	0.113770	0.033250	03:10
1	0.083040	0.090087	0.030750	03:09
2	0.088097	0.103396	0.034500	03:09
3	0.071160	0.083401	0.030250	03:09
4	0.062203	0.083172	0.026750	03:10
5	0.053603	0.082909	0.027500	03:10

```
learn.lr_find
```

```
learn.recorder.plot()
```




```
learn.save('stage-3')
```

```
learn.fit_one_cycle(2,max_lr=2e-7)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.050104	0.083129	0.027500	03:10
1	0.046099	0.085462	0.026750	03:09

```
interp = ClassificationInterpretation.from_learner(learn)
```

```
learn.save('stage-3')
```

```
learn.load('stage-3')
```

```
learn.export()
```

```
testdf=pd.read_csv('/content/deep-learning/CV/submission.csv')
```

```
testdf1.head()
```

	image	label
0	C84P45ThinF_IMG_20150818_101903_cell_10.png	
1	C116P77ThinF_IMG_20150930_171844_cell_110.png	
2	C234ThinF_IMG_20151112_162759_cell_22.png	
3	C99P60ThinF_IMG_20150918_141129_cell_113.png	
4	C130P91ThinF_IMG_20151004_141504_cell_72.png	

```
y=testdf1.to_csv(index=False)
```

```
test = ImageList.from_df(testdf1,"",folder='test')
```

```
test
```

```
ImageList (7558 items)
Image (3, 112, 100),Image (3, 121, 127),Image (3, 130, 133),Image (3, 163, 154),Image
Path: .
```

```
data.add_test(test)
```

```
learn = load_learner('/content/deep-learning/CV/Data/','export.pkl', test=test)
preds, _ = learn.get_preds(ds_type=DatasetType.Test)
```

```
preds[:5]
```

```
tensor([[9.9995e-01, 4.7743e-05],
        [9.9894e-01, 1.0569e-03],
        [6.3640e-04, 9.9936e-01],
        [1.0000e+00, 3.4597e-06],
        [1.6317e-02, 9.8368e-01]])
```

```
labelled_preds = torch.argmax(preds,1)
```

```
labelled_preds
```

```
tensor([0, 0, 1,  ..., 0, 0, 0])
```

```
fnames = [f for f in testdf1['image']]
```

```
fnames[:5]
```

```
['C84P45ThinF_IMG_20150818_101903_cell_10.png',
 'C116P77ThinF_IMG_20150930_171844_cell_110.png',
 'C234ThinF_IMG_20151112_162759_cell_22.png',
 'C99P60ThinF_IMG_20150918_141129_cell_113.png',
 'C130P91ThinF_IMG_20151004_141504_cell_72.png']
```

```
df = pd.DataFrame({'image':fnames, 'category':labelled_preds}, columns=['image', 'category'])

df.to_csv('submission.csv', index=True)
```