Oxford-pets, Resnet34

```
from fastai import *
from fastai.vision import *
import matplotlib as plt
from PIL import Image
```

```
path = untar_data(URLs.PETS) # getting data(FAI)
```

```
Downloading https://s3.amazonaws.com/fast-ai-imageclas/oxford-iiit-pet
```

```
path
```

```
PosixPath('/root/.fastai/data/oxford-iiit-pet')
```

```
path.ls() # checking paths
```

```
[PosixPath('/root/.fastai/data/oxford-iiit-pet/images'),
 PosixPath('/root/.fastai/data/oxford-iiit-pet/annotations')]
```

```
path_anno = path/'annotations'
path_img = path/'images'
```

```
fnames = get_image_files(path_img)
fnames[:5]
```

```
[PosixPath('/root/.fastai/data/oxford-iiit-pet/images/american_pit_bull_terrier_60.jp
 PosixPath('/root/.fastai/data/oxford-iiit-pet/images/basset_hound_96.jpg'),
 PosixPath('/root/.fastai/data/oxford-iiit-pet/images/staffordshire_bull_terrier_73.j
 PosixPath('/root/.fastai/data/oxford-iiit-pet/images/samoyed_56.jpg'),
 PosixPath('/root/.fastai/data/oxford-iiit-pet/images/yorkshire_terrier_6.jpg')]
```

```
pattern_label = r'/([^/]+)_\d+.jpg$'
np.random.seed(1)
```

```
data = ImageDataBunch.from_name_re(path_img, fnames, pattern_label, ds_tfms=get_transforms
```

```
data.normalize(imagenet_stats) # normalize to make the mean and SD of different channels(R
ImageDataBunch;

Train: LabelList (5912 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image
y: CategoryList
american_pit_bull_terrier,basset_hound,staffordshire_bull_terrier,samoyed,yorkshire_t
Path: /root/.fastai/data/oxford-iiit-pet/images;

Valid: LabelList (1478 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image
y: CategoryList
american_bulldog,miniature_pinscher,miniature_pinscher,shiba_inu,keeshond
Path: /root/.fastai/data/oxford-iiit-pet/images;

Test: None
```

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

```
print(data.classes)
```

```
['Abyssinian', 'Bengal', 'Birman', 'Bombay', 'British_Shorthair', 'Egyptian_Mau', 'Ma
```

```
len(data.classes)
```

```
37
```

```
learn = create_cnn(data, models.resnet34, metrics=error_rate)
```

```
/usr/local/lib/python3.6/dist-packages/fastai/vision/learner.py:106: UserWarning: `cr
  warn("`create_cnn` is deprecated and is now named `cnn_learner`.")
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /root/.ca
100%|██████████| 83.3M/83.3M [00:03<00:00, 22.3MB/s]
```

```
learn.fit_one_cycle(5)
```

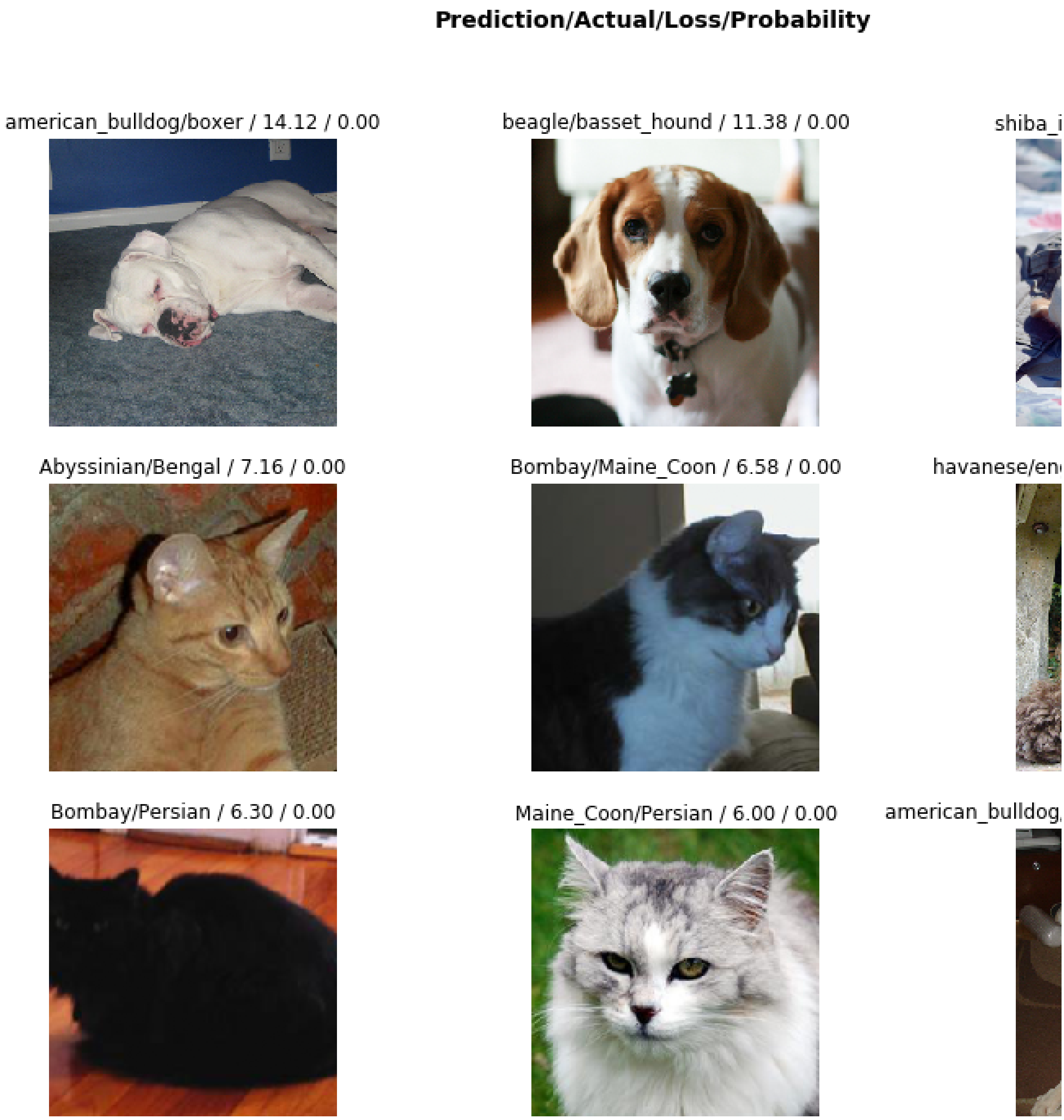
epoch	train_loss	valid_loss	error_rate	time
0	1.519426	0.309154	0.100135	01:22
1	0.637294	0.309648	0.092693	01:21
2	0.435239	0.272449	0.087957	01:21
3	0.311092	0.228633	0.070365	01:22
4	0.229652	0.226257	0.063599	01:22

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

Results

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

```
interp.plot_top_losses(9, figsize=(15, 11))
```


Prediction/Actual/Loss/Probability
american_bulldog/boxer / 14.12 / 0.00
beagle/basset_hound / 11.38 / 0.00
shiba_i
Abyssinian/Bengal / 7.16 / 0.00
Bombay/Maine_Coon / 6.58 / 0.00
havanese/en
Bombay/Persian / 6.30 / 0.00
Maine_Coon/Persian / 6.00 / 0.00
american_bulldog

```
doc(interp.plot_top_losses)
```

_cl_int_plot_top_losses

`_cl_int_plot_top_losses`(**k**, **largest**=*True*, **figsize**=*(12, 12)*, **heatmap**:bool=*False*, **heatm**
return_fig:bool=*None*) → Optional[Figure]

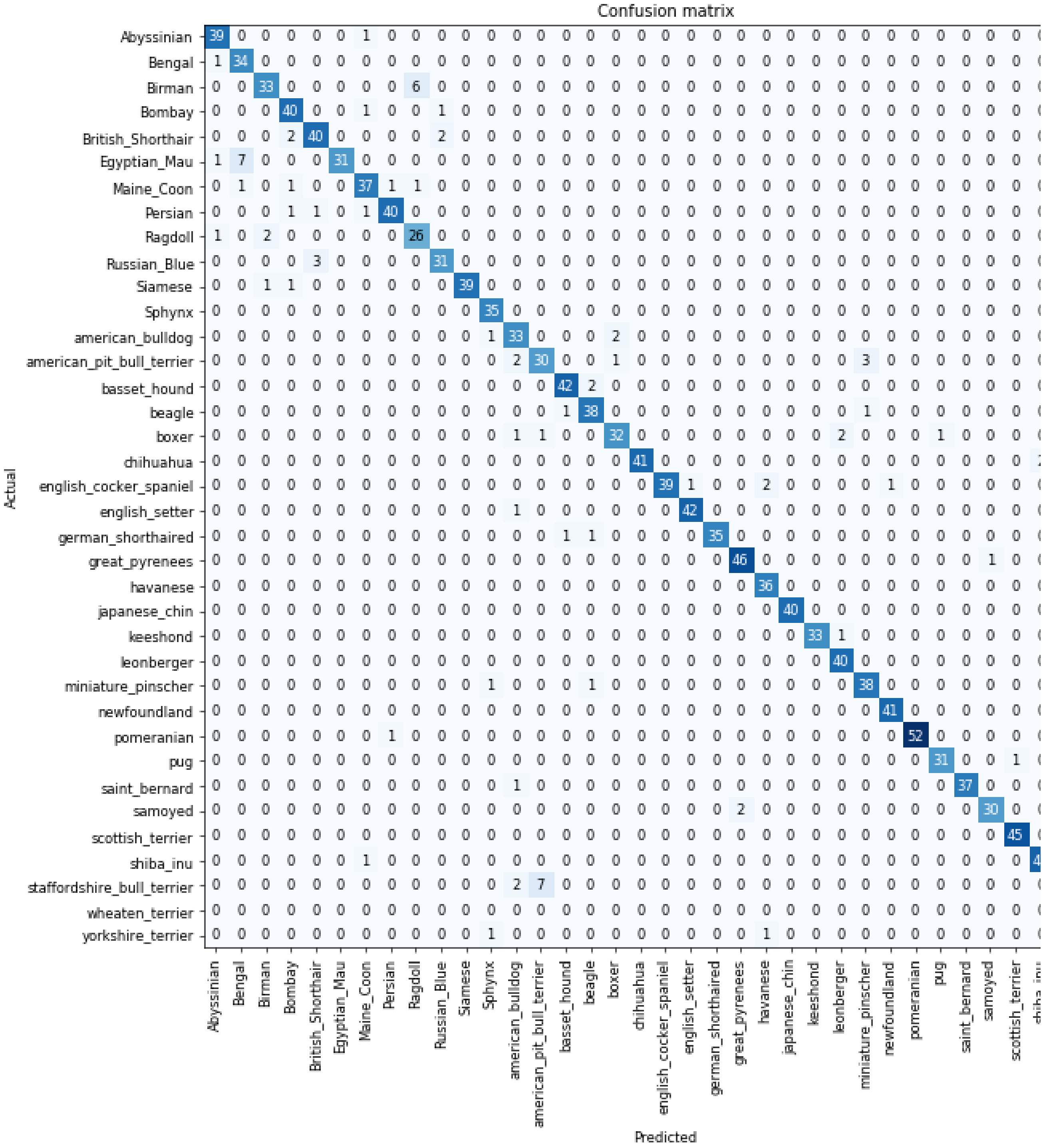
×

No tests found for `_cl_int_plot_top_losses`. To contribute a test please refer to this guide and this disc

Show images in `top_losses` along with their prediction, actual, loss, and probability of actual class.

Show in docs

```
interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```



```
interp.most_confused(min_val=3)
```

```
[('Egyptian_Mau', 'Bengal', 7),
 ('staffordshire_bull_terrier', 'american_pit_bull_terrier', 7),
 ('Birman', 'Ragdoll', 6),
 ('american_pit_bull_terrier', 'staffordshire_bull_terrier', 6),
 ('Russian_Blue', 'British_Shorthair', 3),
 ('american_pit_bull_terrier', 'miniature_pinscher', 3)]
```

Unfreezing and fine tuning

```
learn.unfreeze()
learn.fit_one_cycle(1)
```

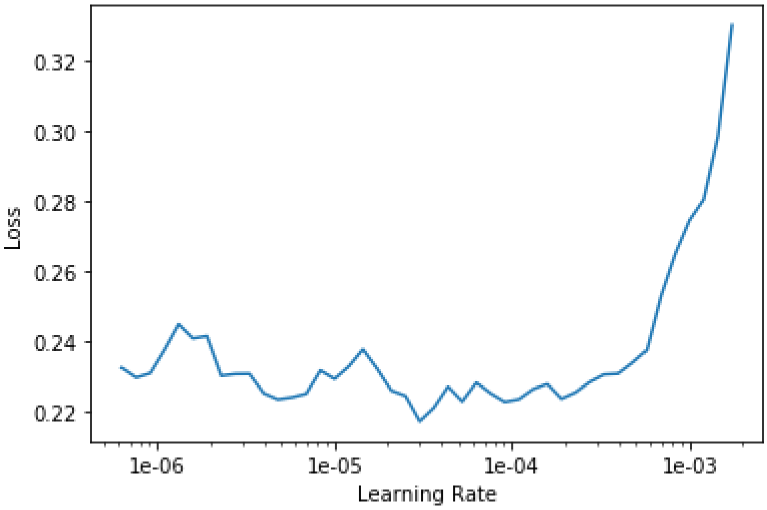
epoch	train_loss	valid_loss	error_rate	time
0	0.684500	0.370274	0.115020	01:23

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

```
learn.lr_find()
```

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

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



```
learn.unfreeze()
```

```
learn.fit_one_cycle(2, max_lr=slice(1e-6, 5e-5))
```

epoch	train_loss	valid_loss	error_rate	time
0	0.229646	0.224551	0.066982	01:22
1	0.208496	0.214111	0.065629	01:20

Resnet50 model, bigger images, bigger batch size for transfe

```
data1 = ImageDataBunch.from_name_re(path_img, fnames, pattern_label, ds_tfms=get_transform
```

```
data1.normalize(imagenet_stats)
```

```
learn1 = create_cnn(data1, models.resnet50, metrics=error_rate)
```

```
learn1.fit_one_cycle(5)
```

```
epoch  train_loss  valid_loss  error_rate  time
    0    0.909988    0.327232    0.094723  01:58
    1    0.461359    0.260119    0.076455  01:54
    2    0.292678    0.219529    0.062923  01:57
    3    0.196122    0.190867    0.056834  01:59
    4    0.135056    0.184373    0.053451  01:58
```

```
learn1.save('stage-1-50')
```

```
learn1.unfreeze()
```

```
learn1.fit_one_cycle(1)
```

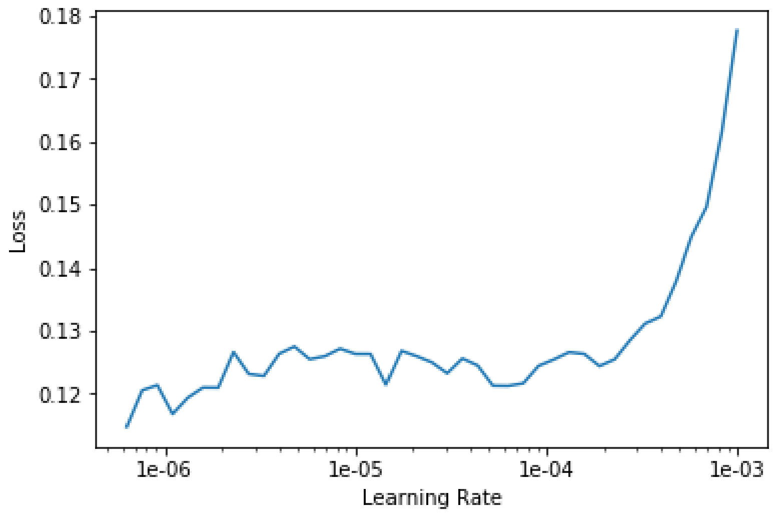
```
epoch  train_loss  valid_loss  error_rate  time
    0    0.675920    0.313637    0.106225  02:03
```

```
learn1.load('stage-1-50')
```

```
learn1.lr_find()
```

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

```
learn1.recorder.plot()
```



```
learn1.unfreeze()
```

```
learn1.fit_one_cycle(2, max_lr=slice(1e-6, 5e-4))
```

epoch	train_loss	valid_loss	error_rate	time
0	0.142892	0.193107	0.060893	02:02
1	0.106905	0.162742	0.045332	02:04

```
learn1.save('stage-1-50-mod')
```

MNIST_SAMPLE(3 and 7) creating databunch using 'from_fc

```
path = untar_data(URLs.MNIST_SAMPLE)
```

```
Downloading http://files.fast.ai/data/examples/mnist_sample
```

```
path.ls()
```

```
[PosixPath('/root/.fastai/data/mnist_sample/labels.csv'),
 PosixPath('/root/.fastai/data/mnist_sample/valid'),
 PosixPath('/root/.fastai/data/mnist_sample/train')]
```

```
path_labels = path/'labels'
path_valid = path/'valid'
path_train = path/'train'
```

```
path_train.ls()
```

```
[PosixPath('/root/.fastai/data/mnist_sample/train/3'),
 PosixPath('/root/.fastai/data/mnist_sample/train/7')]
```

```
tfms = get_transforms(do_flip=False)
data = ImageDataBunch.from_folder(path, ds_tfms=tfms, size=26)
```

```
data
```

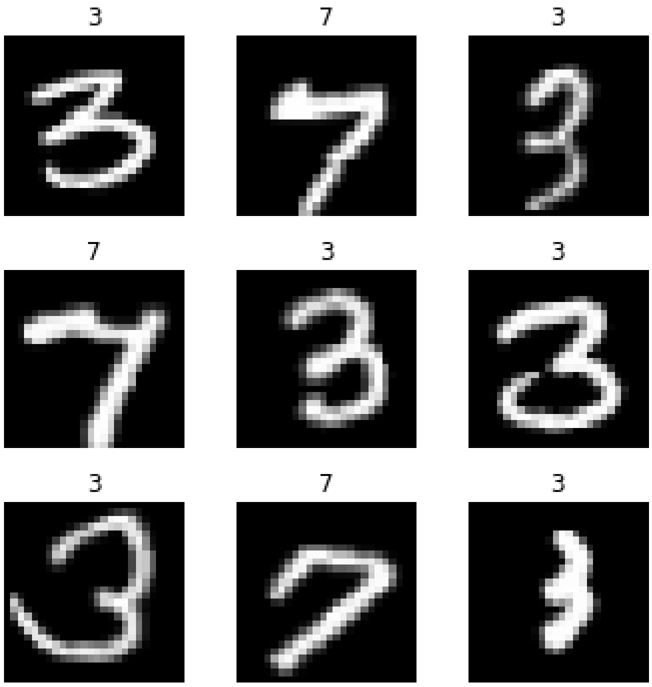
```
ImageDataBunch;

Train: LabelList (12396 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
3,3,3,3,3
Path: /root/.fastai/data/mnist_sample;

Valid: LabelList (2038 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
3,3,3,3,3
Path: /root/.fastai/data/mnist_sample;

Test: None
```

```
data.show_batch(rows=3, figsize=(5,5))
```



```
learn = create_cnn(data, models.resnet18, metrics=accuracy)
```

```
/usr/local/lib/python3.6/dist-packages/fastai/vision/learner.py:106: UserWarning: `cr
  warn("`create_cnn` is deprecated and is now named `cnn_learner`.")
Downloading: "https://download.pytorch.org/models/resnet18-5c106cde.pth" to /root/.ca
100%|██████████| 44.7M/44.7M [00:02<00:00, 23.2MB/s]
```

```
learn.fit_one_cycle(5)
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    0.108815    0.056137  0.980864  00:16
    1    0.077194    0.034737  0.988714  00:16
    2    0.066731    0.028009  0.989696  00:16
    3    0.061687    0.024061  0.992149  00:16
    4    0.047814    0.022234  0.993131  00:16
```

MNIST_SAMPLE(3 and 7) creating databunch using the 'labe

```
df = pd.read_csv(path/'labels.csv')
```

```
df.head()
```

	name	label
0	train/3/7463.png	0
1	train/3/21102.png	0
2	train/3/31559.png	0
3	train/3/46882.png	0
4	train/3/26209.png	0

```
data_csv = ImageDataBunch.from_csv(path, ds_tfms=tfms, size=26)
```

```
data_csv
```

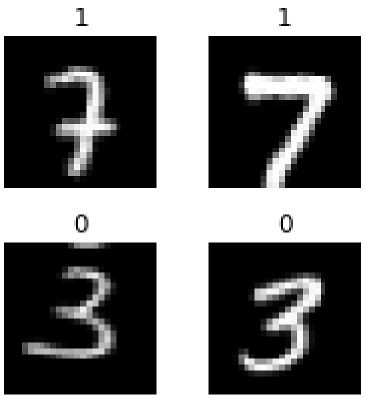
```
ImageDataBunch;

Train: LabelList (11548 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
0,0,0,0,0
Path: /root/.fastai/data/mnist_sample;

Valid: LabelList (2886 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
0,1,1,0,1
Path: /root/.fastai/data/mnist_sample;

Test: None
```

```
data_csv.show_batch(rows=2, figsize=(3,3))
```



```
data_csv.classes
```

```
[0, 1]
```

```
learn_csv = cnn_learner(data_csv, models.resnet18, metrics=accuracy)
```

```
learn_csv.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.358849	0.166654	0.935897	00:14
1	0.168769	0.103451	0.955994	00:14
2	0.114843	0.054705	0.979557	00:15
3	0.096019	0.033092	0.988912	00:14
4	0.079061	0.032406	0.988912	00:14

MNIST_SAMPLE(3 and 7) creating databunch using RegEx

```
pattern = r'/(\d)/\d+\.png$'
```

```
fn_paths = [path/name for name in df['name']]
```

```
fn_paths[:2]
```

```
[PosixPath('/root/.fastai/data/mnist_sample/train/3/7463.png'),
 PosixPath('/root/.fastai/data/mnist_sample/train/3/21102.png')]
```

```
data_reg = ImageDataBunch.from_name_re(path, fn_paths, pat=pattern, ds_tfms=tfms, size=26)
```

```
data_reg
```

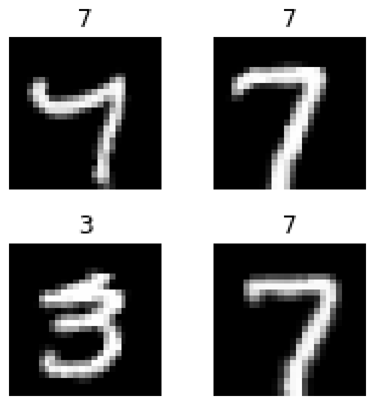
```
ImageDataBunch;

Train: LabelList (11548 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
3,3,3,3,3
Path: /root/.fastai/data/mnist_sample;

Valid: LabelList (2886 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
7,3,3,7,3
Path: /root/.fastai/data/mnist_sample;

Test: None
```

```
data_reg.show_batch(rows=2, figsize=(3,3))
```



```
learn_reg = cnn_learner(data_reg, models.resnet18, metrics=accuracy)
```

```
learn_reg.fit(5)
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    0.203526    0.094837  0.961538  00:16
    1    0.120946    0.051662  0.981636  00:16
    2    0.108039    0.037705  0.989605  00:15
    3    0.066713    0.021860  0.992377  00:15
    4    0.078224    0.018173  0.994110  00:16
```

MNIST_SAMPLE(3 and 7) creating databunch using RegEx w

```
pattern = r'/(\d)/\d+\.png$'
```

```
fn_paths = [path/name for name in df['name']]
```

```
data_lambda = ImageDataBunch.from_name_func(path, fn_paths, ds_tfms=tfms, size=26,
                                            label_func= lambda x : '3' if '/3/' in str(x) el
```

```
data_lambda
```

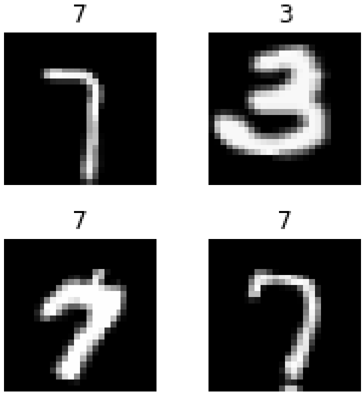
```
ImageDataBunch;

Train: LabelList (11548 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
3,3,3,3,3
Path: /root/.fastai/data/mnist_sample;

Valid: LabelList (2886 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
7,7,7,3,3
Path: /root/.fastai/data/mnist_sample;

Test: None
```

```
data_lambda.show_batch(rows=2, figsize=(3,3))
```



```
learn_lambda = cnn_learner(data_lambda, models.resnet18, metrics=accuracy)
```

```
learn_lambda.fit(5)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.197588	0.076690	0.972973	00:15
1	0.121993	0.054215	0.979903	00:15
2	0.092091	0.042031	0.986140	00:15
3	0.080076	0.023034	0.994802	00:15
4	0.054163	0.012242	0.995495	00:15

MNIST_SAMPLE(3 and 7) creating databunch using RegEx w

```
labels=[('3' if '/3/' in str(x) else '7') for x in fn_paths]
```

```
data_labels = ImageDataBunch.from_lists(path, fn_paths, ds_tfms=tfms, size=26, labels=labe
data_labels
```

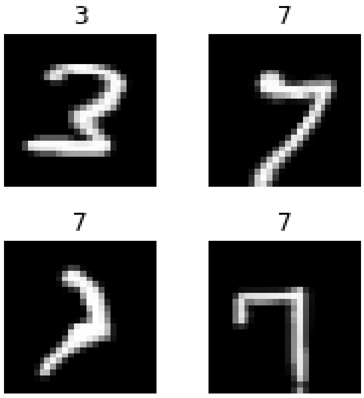
```
ImageDataBunch;

Train: LabelList (11548 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
3,3,3,3,3
Path: /root/.fastai/data/mnist_sample;

Valid: LabelList (2886 items)
x: ImageList
Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26, 26),Image (3, 26,
y: CategoryList
3,3,3,3,7
Path: /root/.fastai/data/mnist_sample;

Test: None
```

```
data_labels.show_batch(rows=2, figsize=(3,3))
```



```
learn_labels = cnn_learner(data_labels, models.resnet18, metrics=accuracy)
```

```
learn_labels.fit(5)
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    0.059016    0.021237  0.993070  00:16
    1    0.039150    0.014911  0.994456  00:16
    2    0.035309    0.015841  0.994456  00:16
    3    0.034600    0.010617  0.995842  00:16
    4    0.036799    0.012018  0.996189  00:16
```