```
%reload_ext autoreload
%autoreload 2
%matplotlib inline

%config InlineBackend.figure_format = 'retina'


from fastai.vision import *
from fastai.metrics import error_rate
from pathlib import Path
from glob2 import glob
from sklearn.metrics import confusion_matrix

import pandas as pd
import numpy as np
import os
import zipfile as zf
import shutil
import re
import seaborn as sns


from zipfile import ZipFile
file_name="dataset-resized.zip"

with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print("done")
```

```
done
```

# New section

```
os.listdir(os.path.join(os.getcwd(),"dataset-resized"))
```

```
['.DS_Store', 'plastic', 'paper', 'trash', 'cardboard', 'metal', 'glass']
```

```
## helper functions ##

## splits indices for a folder into train, validation, and test indices with random sampli
    ## input: folder path
    ## output: train, valid, and test indices
def split_indices(folder,seed1,seed2):
    n = len(os.listdir(folder))
    full_set = list(range(1,n+1))

    ## train indices
    random.seed(seed1)
    train = random.sample(list(range(1,n+1)),int(.5*n))
```

```
    ## temp
    remain = list(set(full_set)-set(train))

    ## separate remaining into validation and test
    random.seed(seed2)
    valid = random.sample(remain,int(.5*len(remain)))
    test = list(set(remain)-set(valid))

    return(train,valid,test)

## gets file names for a particular type of trash, given indices
    ## input: waste category and indices
    ## output: file names
def get_names(waste_type,indices):
    file_names = [waste_type+str(i)+".jpg" for i in indices]
    return(file_names)

## moves group of source files to another folder
    ## input: list of source files and destination folder
    ## no output
def move_files(source_files,destination_folder):
    for file in source_files:
        shutil.move(file,destination_folder)


## paths will be train/cardboard, train/glass, etc...
subsets = ['train','valid']
waste_types = ['cardboard','glass','metal','paper','plastic','trash']

## create destination folders for data subset and waste type
for subset in subsets:
    for waste_type in waste_types:
        folder = os.path.join('data',subset,waste_type)
        if not os.path.exists(folder):
            os.makedirs(folder)

if not os.path.exists(os.path.join('data','test')):
    os.makedirs(os.path.join('data','test'))

## move files to destination folders for each waste type
for waste_type in waste_types:
    source_folder = os.path.join('dataset-resized',waste_type)
    train_ind, valid_ind, test_ind = split_indices(source_folder,1,1)

    ## move source files to train
    train_names = get_names(waste_type,train_ind)
    train_source_files = [os.path.join(source_folder,name) for name in train_names]
    train_dest = "data/train/"+waste_type
    move_files(train_source_files,train_dest)

    ## move source files to valid
    valid_names = get_names(waste_type,valid_ind)
    valid_source_files = [os.path.join(source_folder,name) for name in valid_names]
    valid_dest = "data/valid/"+waste_type
```

```
        move_files(valid_source_files,valid_dest)

        ## move source files to test
        test_names = get_names(waste_type,test_ind)
        test_source_files = [os.path.join(source_folder,name) for name in test_names]
        ## I use data/test here because the images can be mixed up
        move_files(test_source_files,"data/test")


    ## get a path to the folder with images
    path = Path(os.getcwd())/"data"
    path
```

```
PosixPath('/content/data')
```

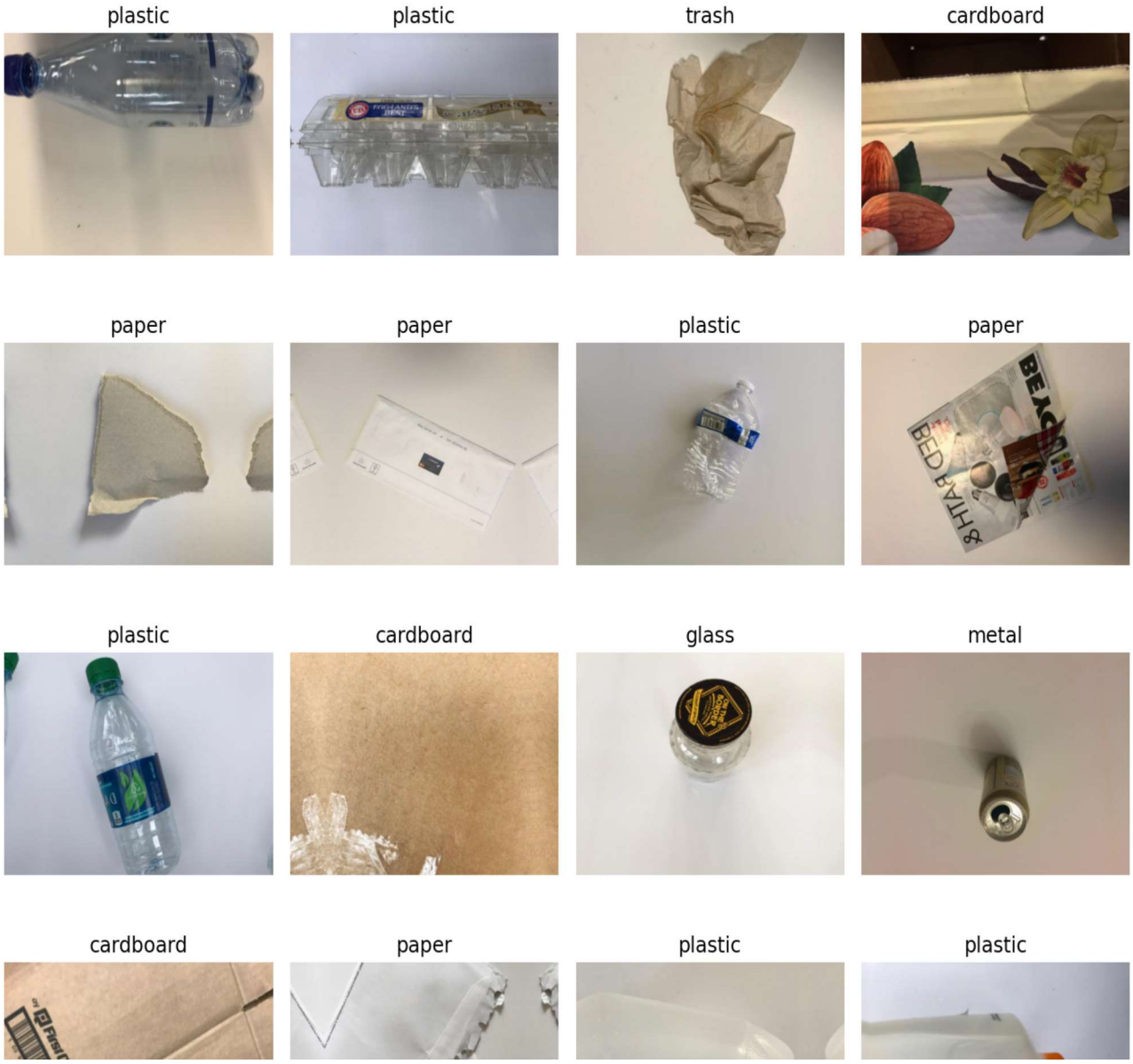
```
    tfms = get_transforms(do_flip=True,flip_vert=True)
    data = ImageDataBunch.from_folder(path,test="test",ds_tfms=tfms,bs=16)


    data
```

```
ImageDataBunch;

Train: LabelList (1262 items)
x: ImageList
Image (3, 384, 512),Image (3, 384, 512),Image (3, 384, 512),Image (3, 384, 512),Imag
y: CategoryList
plastic,plastic,plastic,plastic,plastic
Path: /content/data;

Valid: LabelList (630 items)
x: ImageList
Image (3, 384, 512),Image (3, 384, 512),Image (3, 384, 512),Image (3, 384, 512),Imag
y: CategoryList
plastic,plastic,plastic,plastic,plastic
Path: /content/data;

Test: LabelList (635 items)
x: ImageList
Image (3, 384, 512),Image (3, 384, 512),Image (3, 384, 512),Image (3, 384, 512),Imag
y: EmptyLabelList
,,,,
Path: /content/data
```

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

```
['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
```

```
    data.show_batch(rows=10,figsize=(10,10))
```




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

```
/usr/local/lib/python3.6/dist-packages/fastai/vision/learner.py:109: UserWarning: `c
  warn("`create_cnn` is deprecated and is now named `cnn_learner`.")
```

```
learn.model
```

```
Sequential(
  (0): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
      )
      (2): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_s
      )
    )
    (5): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
      )
      (2): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
      )
      (3): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_
```

```
    )
  )
  (6): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
    )
    (2): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
    )
    (3): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
    )
    (4): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
    )
    (5): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_
    )
  )
  (7): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
      )
    )
```

```
  )
  (1): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
  )
  (2): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
  )
```

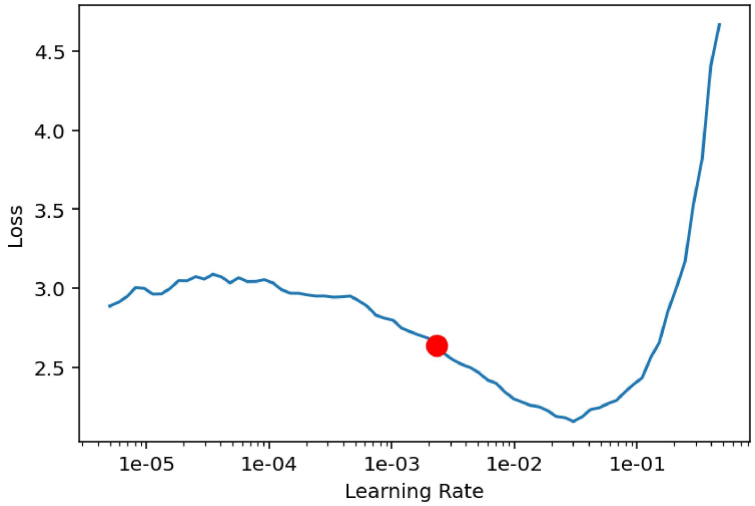
```
learn.freeze()
learn.lr_find(start_lr=1e-6, end_lr=1e1, wd=5e-3)
learn.recorder.plot(suggestion=True)
```

50.00% [1/2 00:43<00:43]

| epoch | train_loss | valid_loss | error_rate | time |
|---|---|---|---|---|
| 0 | 3.168776 | #na# | | 00:43 |

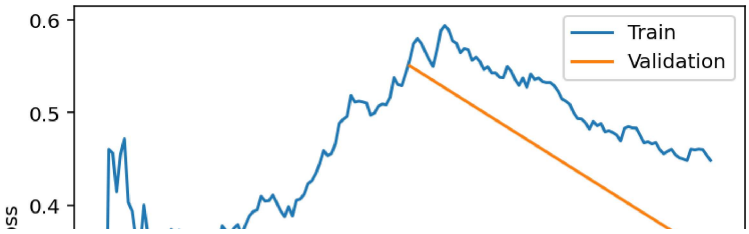
10.26% [8/78 00:05<00:44 8.3158]

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
Min numerical gradient: 2.29E-03
Min loss divided by 10: 3.02E-03
```




```
learn.fit_one_cycle(2, max_lr=3e-3, pct_start=0.1, div_factor=10, final_div=30, wd=5e-3, m
learn.save('stage-1')
learn.recorder.plot_losses()
```

| epoch | train_loss | valid_loss | error_rate | time |
|---|---|---|---|---|
| 0 | 0.541931 | 0.550473 | 0.163492 | 00:51 |
| 1 | 0.448527 | 0.343839 | 0.106349 | 00:51 |



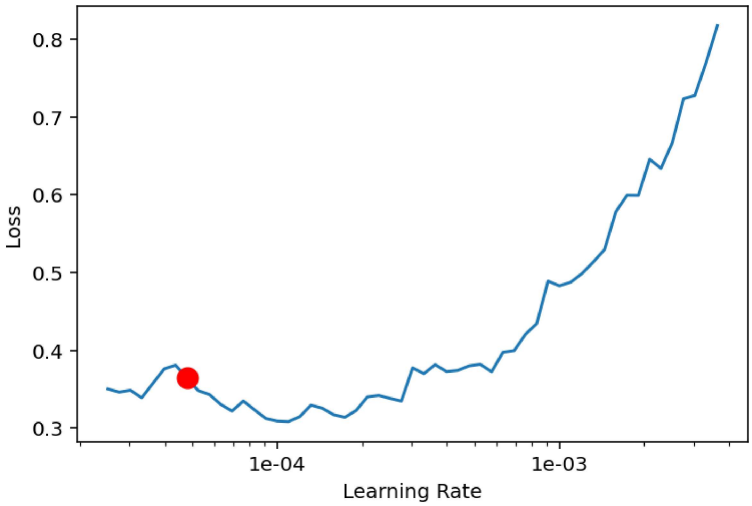

```
# unfreeze and search appropriate learning rate for full training
learn.unfreeze()
learn.lr_find(start_lr=slice(1e-6, 1e-5), end_lr=slice(1e-2, 1e-1), wd=1e-3)
learn.recorder.plot(suggestion=True)
```

0.00% [0/2 00:00<00:00]

| epoch | train_loss | valid_loss | error_rate | time |
|---|---|---|---|---|

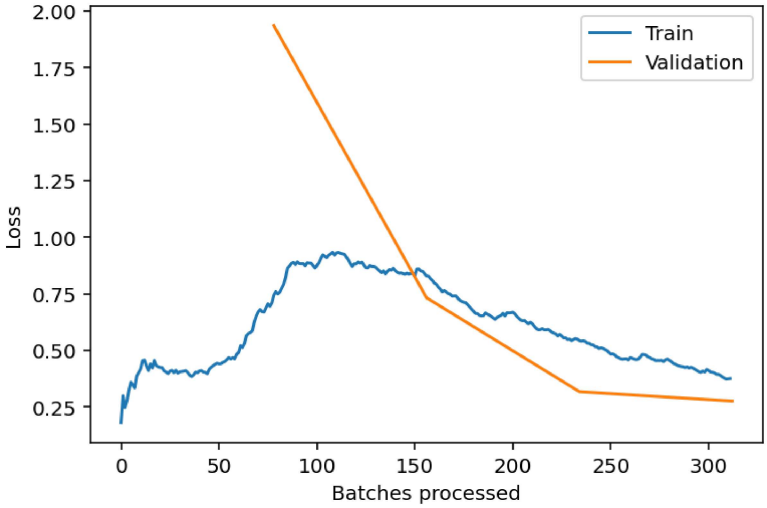
88.46% [69/78 00:42<00:05 0.9763]

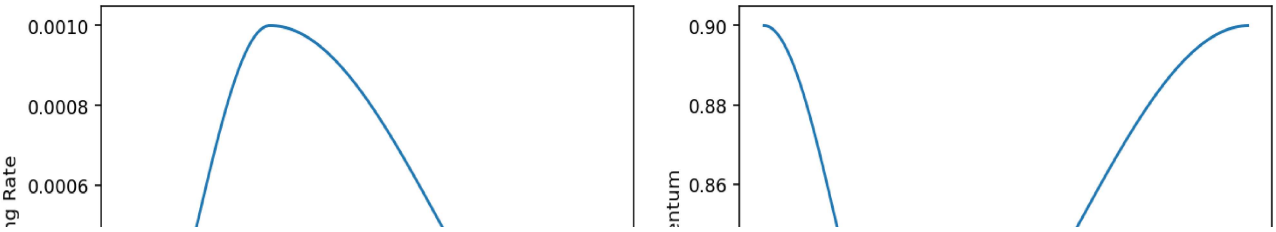
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
Min numerical gradient: 4.79E-05
Min loss divided by 10: 1.10E-05
```




```
# train all layers
learn.fit_one_cycle(4, max_lr=slice(1e-4, 1e-3), div_factor=50, final_div=100, wd=1e-3, mc
learn.save('stage-2')
learn.recorder.plot_losses()
# schedule of the lr (left) and momentum (right) that the 1cycle policy uses
learn.recorder.plot_lr(show_moms=True)
```

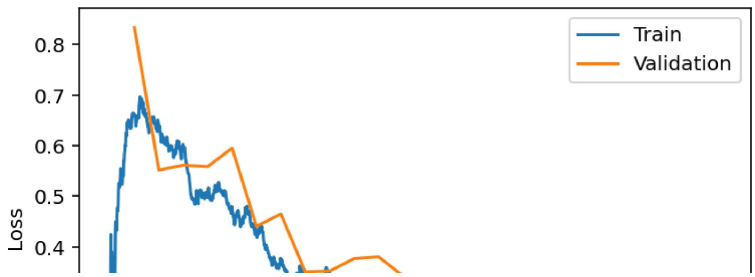
| epoch | train_loss | valid_loss | error_rate | time |
|---|---|---|---|---|
| 0 | 0.710440 | 1.935624 | 0.403175 | 00:55 |
| 1 | 0.838917 | 0.732250 | 0.273016 | 00:55 |
| 2 | 0.549303 | 0.317220 | 0.098413 | 00:55 |
| 3 | 0.375120 | 0.275464 | 0.085714 | 00:55 |



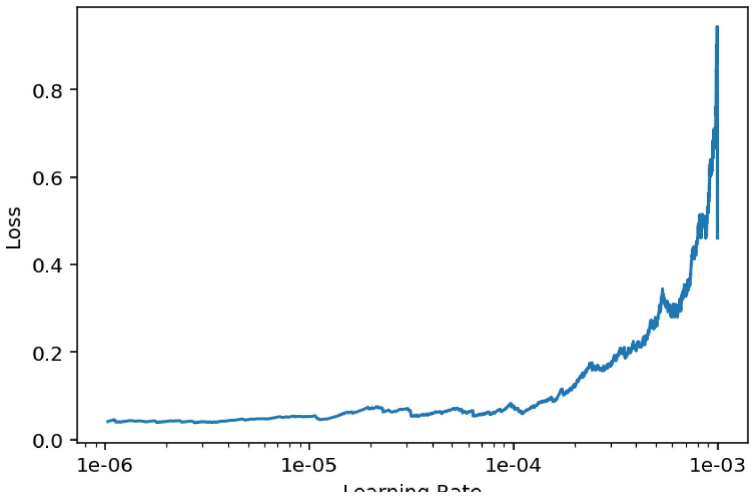



```
# train all layers
learn.fit_one_cycle(cyc_len=25, max_lr=slice(1e-4, 1e-3), pct_start=0, final_div=1000, wd=
learn.save('stage-3')
learn.recorder.plot_losses()
# # schedule of the lr (left) and momentum (right) that the 1cycle policy uses
learn.recorder.plot_lr(show_moms=True)
```

| epoch | train_loss | valid_loss | error_rate | time |
|---|---|---|---|---|
| 0 | 0.661865 | 0.833219 | 0.250794 | 00:56 |
| 1 | 0.644664 | 0.551600 | 0.207937 | 00:56 |
| 2 | 0.602549 | 0.561101 | 0.188889 | 00:56 |
| 3 | 0.499956 | 0.558382 | 0.195238 | 00:56 |
| 4 | 0.479834 | 0.594895 | 0.200000 | 00:55 |
| 5 | 0.426385 | 0.440468 | 0.130159 | 00:55 |
| 6 | 0.377105 | 0.464676 | 0.171429 | 00:55 |
| 7 | 0.339981 | 0.350086 | 0.114286 | 00:55 |
| 8 | 0.344789 | 0.351899 | 0.126984 | 00:55 |
| 9 | 0.312505 | 0.376408 | 0.126984 | 00:55 |
| 10 | 0.269394 | 0.380059 | 0.120635 | 00:56 |
| 11 | 0.217099 | 0.342207 | 0.107937 | 00:56 |
| 12 | 0.171804 | 0.253995 | 0.080952 | 00:56 |
| 13 | 0.173610 | 0.242147 | 0.079365 | 00:55 |
| 14 | 0.156049 | 0.267856 | 0.080952 | 00:55 |
| 15 | 0.127287 | 0.219792 | 0.068254 | 00:57 |
| 16 | 0.115073 | 0.202695 | 0.061905 | 00:56 |
| 17 | 0.093572 | 0.207282 | 0.071429 | 00:57 |
| 18 | 0.077892 | 0.215668 | 0.058730 | 00:56 |
| 19 | 0.054730 | 0.207211 | 0.068254 | 00:56 |
| 20 | 0.064356 | 0.179681 | 0.061905 | 00:56 |
| 21 | 0.048653 | 0.191036 | 0.066667 | 00:56 |
| 22 | 0.057065 | 0.195278 | 0.065079 | 00:55 |
| 23 | 0.076381 | 0.175961 | 0.057143 | 00:55 |
| 24 | 0.053429 | 0.178601 | 0.053968 | 00:56 |




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

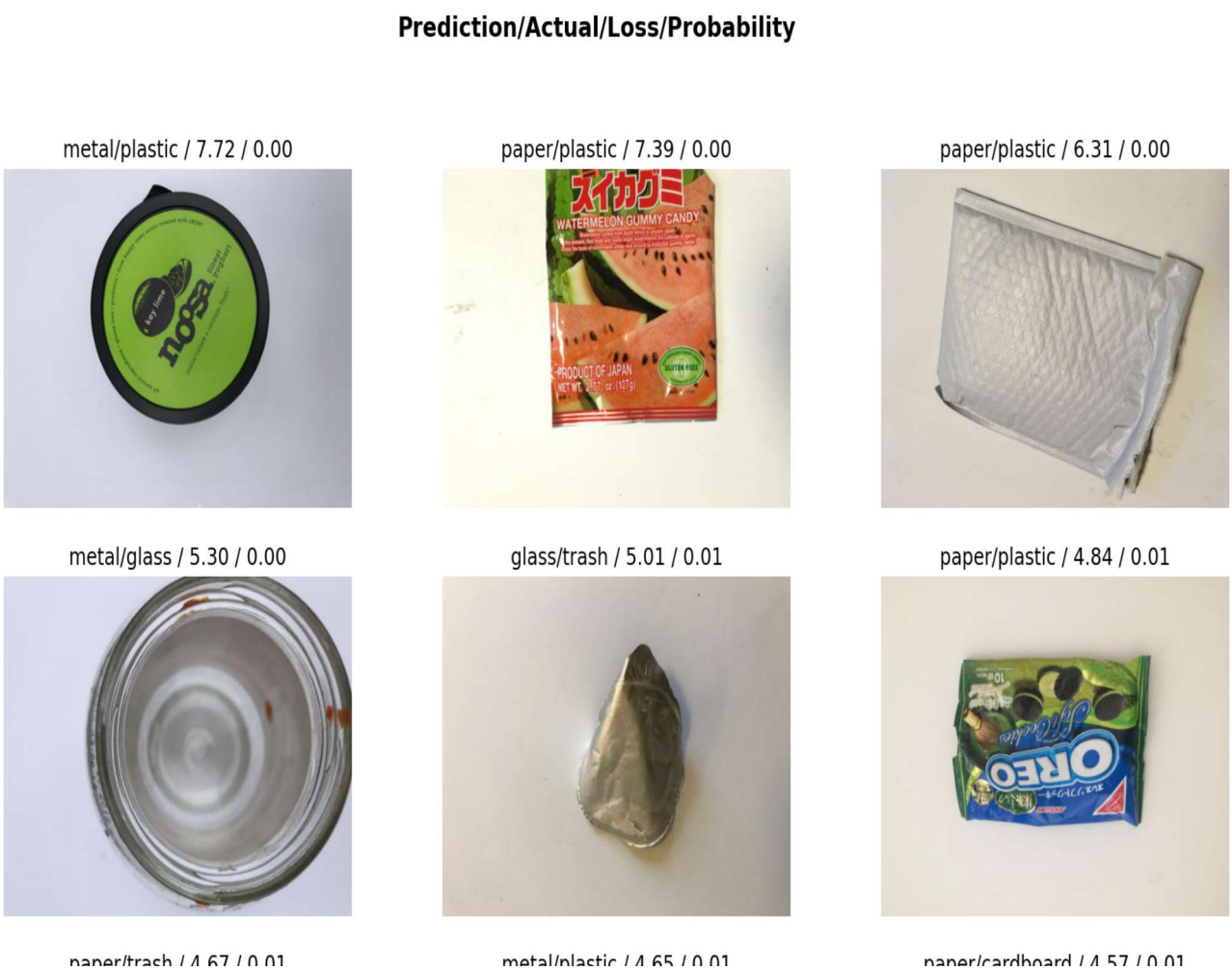



```
learn.fit_one_cycle(10,max_lr=2.29E-03)
```

| epoch | train_loss | valid_loss | error_rate | time |
|---|---|---|---|---|
| 0 | 1.925746 | 0.744957 | 0.263492 | 00:51 |
| 1 | 1.184447 | 0.544262 | 0.188889 | 00:51 |
| 2 | 0.827479 | 0.505477 | 0.152381 | 00:51 |
| 3 | 0.789958 | 0.493378 | 0.147619 | 00:51 |
| 4 | 0.614714 | 0.392201 | 0.123810 | 00:51 |
| 5 | 0.502765 | 0.341864 | 0.106349 | 00:51 |
| 6 | 0.475884 | 0.313752 | 0.106349 | 00:50 |
| 7 | 0.410975 | 0.315682 | 0.106349 | 00:50 |
| 8 | 0.317734 | 0.266906 | 0.082540 | 00:50 |
| 9 | 0.274001 | 0.275533 | 0.080952 | 00:50 |

```
interp = ClassificationInterpretation.from_learner(learn)
losses,idxs = interp.top_losses()
```

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




```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

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

**_cl_int_plot_top_losses** [test] [source]

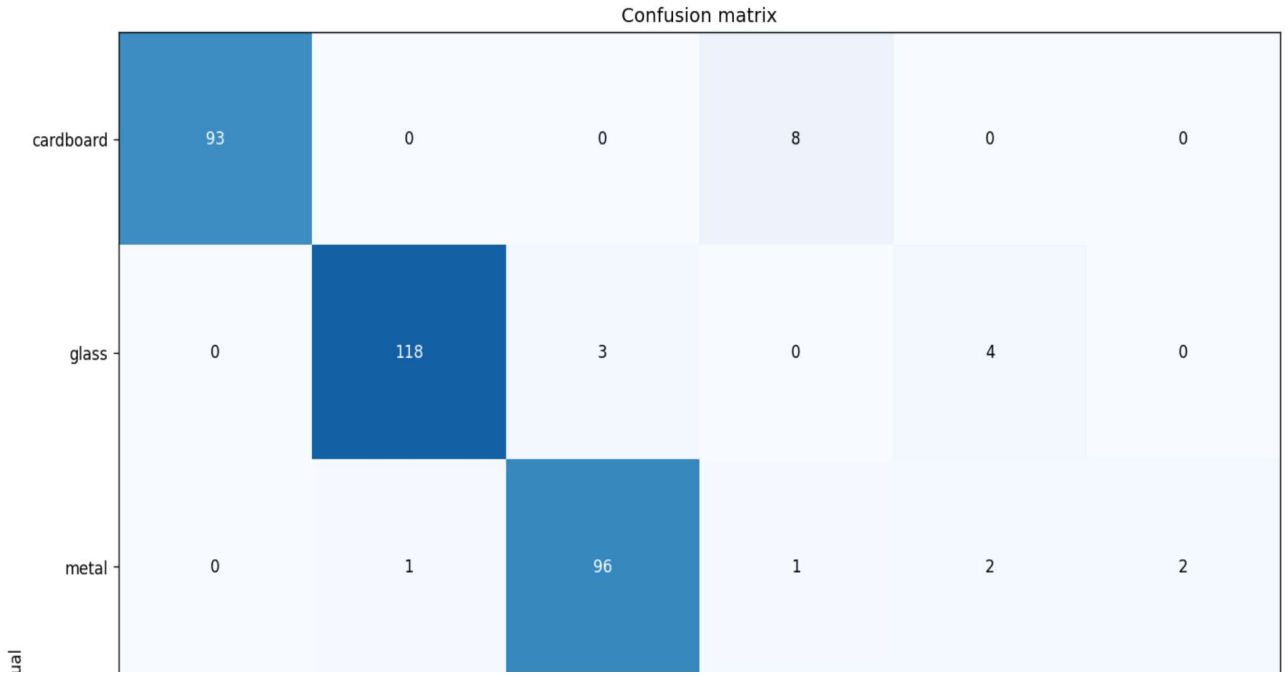
_cl_int_plot_top_losses(**k**, **largest**=***True***, **figsize**=***(12, 12)***, **heatmap**:bool=***False***, **heatmap_thresh**:int=***16***, **alpha**:float=***0.6***, **cmap**:str=***'magma'***, **show_text**:bool=***True***, **return_fig**:bool=***None***) → Optional[Figure]

×

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

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

Show in docs




```
interp.most_confused(min_val=2)
```

```
[('cardboard', 'paper', 8),
 ('plastic', 'glass', 7),
 ('plastic', 'paper', 6),
 ('plastic', 'metal', 5),
 ('glass', 'plastic', 4),
 ('trash', 'paper', 4),
 ('glass', 'metal', 3),
 ('metal', 'plastic', 2),
 ('metal', 'trash', 2),
 ('paper', 'trash', 2),
 ('plastic', 'trash', 2),
 ('trash', 'glass', 2)]
```

```
preds = learn.get_preds(ds_type=DatasetType.Test)
```

```
print(preds[0].shape)
preds[0]
```

```
torch.Size([635, 6])
tensor([[7.4347e-03, 4.0719e-02, 9.4808e-01, 2.7471e-03, 2.8936e-04, 7.2560e-04],
        [1.0555e-03, 1.3216e-01, 7.8278e-02, 9.5935e-05, 7.8803e-01, 3.7813e-04],
        [1.8167e-04, 6.3455e-02, 5.8674e-04, 5.0085e-06, 9.3543e-01, 3.4049e-04],
        ...,
        [5.4346e-01, 1.0564e-02, 2.1437e-03, 4.3707e-01, 6.1309e-03, 6.3481e-04],
        [3.5392e-07, 1.1130e-06, 9.9985e-01, 1.7509e-06, 1.3307e-04, 1.6053e-05]
```

```
data.classes
```

```
['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
```

```
## saves the index (0 to 5) of most likely (max) predicted class for each image
max_idxs = np.asarray(np.argmax(preds[0],axis=1))
```

```
yhat = []
for max_idx in max_idxs:
    yhat.append(data.classes[max_idx])
```

```
yhat
```

```
['metal',
 'plastic',
 'plastic',
 'metal',
 'paper',
 'trash',
 'metal',
 'glass',
 'paper',
 'plastic',
 'trash',
 'glass',
 'glass',
 'plastic',
 'cardboard',
 'paper',
 'paper',
 'paper',
 'plastic',
 'paper',
 'cardboard',
 'plastic',
 'paper',
 'glass',
 'plastic',
 'trash',
 'cardboard',
 'plastic',
 'plastic',
 'cardboard',
 'cardboard',
 'cardboard',
 'metal',
 'metal',
 'plastic',
 'glass',
 'cardboard',
 'cardboard',
 'glass',
 'cardboard',
 'paper',
 'paper',
 'plastic',
 'metal',
 'cardboard',
 'metal',
 'cardboard',
 'metal',
 'paper',
 'cardboard',
 'metal',
 'cardboard',
 'metal',
 'cardboard',
 'plastic',
 'plastic',
 'metal',
 'cardboard',
 'metal',
 'cardboard',
 'plastic',
```

```
        'glass',
        'glass',
        'paper',
        'glass',
        'cardboard',
        'paper',
        'glass',
        'paper',
        'paper',
        'metal',
        'metal',
        'glass',
        'glass',
        'plastic',
        'glass',
        'plastic',
        'metal',
        'paper',
        'metal',
        'glass',
        'plastic',
        'paper',
        'metal',
        'cardboard',
        'cardboard',
        'metal',
        'glass',
        'paper',
        'glass',
        'paper',
        'plastic',
        'glass',
        'cardboard',
        'paper',
        'paper',
        'paper',
        'cardboard',
        'plastic',
        'glass',
        'cardboard',
        'cardboard',
        'glass',
        'plastic',
        'plastic',
        'glass',
        'paper',
        'paper',
        'metal',
        'trash',
        'paper',
        'paper',
        'metal',
        'glass',
        'plastic',
        'glass',
        'plastic',
        'trash',
        'trash',
        'plastic',
        'paper',
        'metal',
        'glass',
```

```
 'glass',
 'plastic',
 'paper',
 'paper',
 'glass',
 'paper',
 'plastic',
 'metal',
 'glass',
 'paper',
 'metal',
 'metal',
 'plastic',
 'plastic',
 'paper',
 'glass',
 'paper',
 'plastic',
 'paper',
 'plastic',
 'paper',
 'cardboard',
 'cardboard',
 'paper',
 'paper',
 'cardboard',
 'glass',
 'glass',
 'cardboard',
 'metal',
 'plastic',
 'cardboard',
 'paper',
 'cardboard',
 'metal',
 'metal',
 'cardboard',
 'glass',
 'paper',
 'metal',
 'glass',
 'plastic',
 'plastic',
 'metal',
 'cardboard',
 'glass',
 'metal',
 'glass',
 'glass',
 'cardboard',
 'paper',
 'cardboard',
 'glass',
 'plastic',
 'cardboard',
 'paper',
 'glass',
 'cardboard',
 'paper',
 'paper',
 'cardboard',
 'metal',
```

```
        'cardboard',
        'paper',
        'plastic',
        'plastic',
        'glass',
        'plastic',
        'plastic',
        'glass',
        'paper',
        'glass',
        'paper',
        'trash',
        'paper',
        'paper',
        'metal',
        'paper',
        'glass',
        'glass',
        'paper',
        'paper',
        'glass',
        'cardboard',
        'plastic',
        'glass',
        'plastic',
        'plastic',
        'glass',
        'glass',
        'plastic',
        'cardboard',
        'glass',
        'metal',
        'paper',
        'metal',
        'plastic',
        'paper',
        'metal',
        'plastic',
        'plastic',
        'paper',
        'paper',
        'cardboard',
        'glass',
        'paper',
        'glass',
        'cardboard',
        'paper',
        'metal',
        'cardboard',
        'glass',
        'paper',
        'paper',
        'plastic',
        'glass',
        'paper',
        'plastic',
        'glass',
        'cardboard',
        'glass',
        'metal',
        'glass',
        'paper',
```

```
 'glass',
 'paper',
 'paper',
 'paper',
 'cardboard',
 'metal',
 'plastic',
 'plastic',
 'glass',
 'paper',
 'metal',
 'metal',
 'metal',
 'paper',
 'paper',
 'paper',
 'plastic',
 'glass',
 'paper',
 'metal',
 'plastic',
 'plastic',
 'glass',
 'glass',
 'metal',
 'plastic',
 'paper',
 'paper',
 'glass',
 'paper',
 'cardboard',
 'paper',
 'cardboard',
 'glass',
 'trash',
 'metal',
 'paper',
 'trash',
 'cardboard',
 'glass',
 'glass',
 'cardboard',
 'glass',
 'plastic',
 'cardboard',
 'metal',
 'paper',
 'metal',
 'plastic',
 'cardboard',
 'plastic',
 'trash',
 'glass',
 'plastic',
 'plastic',
 'metal',
 'metal',
 'glass',
 'paper',
 'metal',
 'metal',
```

```
        'glass',
        'paper',
        'trash',
        'paper',
        'plastic',
        'trash',
        'paper',
        'paper',
        'paper',
        'paper',
        'metal',
        'cardboard',
        'glass',
        'cardboard',
        'paper',
        'trash',
        'glass',
        'cardboard',
        'plastic',
        'glass',
        'paper',
        'cardboard',
        'trash',
        'plastic',
        'metal',
        'paper',
        'paper',
        'cardboard',
        'glass',
        'glass',
        'paper',
        'glass',
        'paper',
        'paper',
        'paper',
        'plastic',
        'glass',
        'paper',
        'glass',
        'glass',
        'cardboard',
        'metal',
        'plastic',
        'glass',
        'glass',
        'paper',
        'paper',
        'paper',
        'paper',
        'glass',
        'paper',
        'metal',
        'glass',
        'paper',
        'trash',
        'glass',
        'paper',
        'paper',
        'glass',
        'paper',
        'paper',
        'paper',
```

```
 'cardboard',
 'metal',
 'cardboard',
 'cardboard',
 'glass',
 'plastic',
 'metal',
 'plastic',
 'plastic',
 'glass',
 'paper',
 'trash',
 'plastic',
 'plastic',
 'trash',
 'glass',
 'plastic',
 'cardboard',
 'cardboard',
 'cardboard',
 'plastic',
 'paper',
 'trash',
 'plastic',
 'paper',
 'plastic',
 'metal',
 'glass',
 'plastic',
 'paper',
 'paper',
 'glass',
 'cardboard',
 'plastic',
 'trash',
 'trash',
 'cardboard',
 'glass',
 'cardboard',
 'cardboard',
 'glass',
 'cardboard',
 'paper',
 'trash',
 'paper',
 'plastic',
 'paper',
 'plastic',
 'glass',
 'plastic',
 'cardboard',
 'plastic',
 'plastic',
 'glass',
 'paper',
 'glass',
 'paper',
 'glass',
 'paper',
 'glass',
 'cardboard',
```

```
        'glass',
        'metal',
        'paper',
        'plastic',
        'glass',
        'paper',
        'glass',
        'glass',
        'paper',
        'metal',
        'paper',
        'glass',
        'glass',
        'plastic',
        'paper',
        'glass',
        'cardboard',
        'paper',
        'cardboard',
        'glass',
        'paper',
        'glass',
        'plastic',
        'plastic',
        'plastic',
        'trash',
        'plastic',
        'metal',
        'cardboard',
        'plastic',
        'paper',
        'plastic',
        'metal',
        'plastic',
        'cardboard',
        'metal',
        'metal',
        'plastic',
        'plastic',
        'plastic',
        'plastic',
        'glass',
        'metal',
        'metal',
        'glass',
        'plastic',
        'paper',
        'glass',
        'paper',
        'glass',
        'cardboard',
        'plastic',
        'glass',
        'paper',
        'plastic',
        'cardboard',
        'glass',
        'paper',
        'metal',
        'glass',
        'cardboard',
        'paper',
```

```
'paper',
'paper',
'paper',
'paper',
'glass',
'glass',
'paper',
'plastic',
'cardboard',
'glass',
'glass',
'plastic',
'plastic',
'glass',
'glass',
'paper',
'glass',
'trash',
'metal',
'cardboard',
'paper',
'cardboard',
'metal',
'plastic',
'paper',
'glass',
'metal',
'metal',
'paper',
'plastic',
'metal',
'plastic',
'glass',
'plastic',
'trash',
'paper',
'cardboard',
'metal',
'cardboard',
'cardboard',
'glass',
'paper',
'plastic',
'metal',
'metal',
'cardboard',
'paper',
'glass',
'glass',
'paper',
'metal',
'glass',
'plastic',
'plastic',
'paper',
'trash',
'glass',
'paper',
'glass',
'metal',
'trash',
```

```
        'glass',
        'paper',
        'plastic',
        'plastic',
        'plastic',
        'cardboard',
        'metal',
        'metal',
        'metal',
        'metal',
        'plastic',
        'paper',
        'paper',
        'metal',
        'cardboard',
        'glass',
        'cardboard',
        'metal',
        'cardboard',
        'trash',
        'glass',
        'plastic',
        'metal',
        'cardboard',
        'cardboard',
        'paper',
        'paper',
        'trash',
        'metal',
        'metal',
        'cardboard',
        'metal',
        'plastic',
        'paper',
        'cardboard',
        'metal',
        'metal',
        'plastic',
        'plastic',
        'metal',
```

```
learn.data.test_ds[0][0]
```

```
y = []

## convert POSIX paths to string first
for label_path in data.test_ds.items:
    y.append(str(label_path))

## then extract waste type from file path
pattern = re.compile("([a-z]+)[0-9]+")
for i in range(len(y)):
    y[i] = pattern.search(y[i]).group(1)
```

```
## predicted values
print(yhat[0:5])
## actual values
print(y[0:5])
```

```
['metal', 'plastic', 'plastic', 'metal', 'paper']
['metal', 'glass', 'plastic', 'metal', 'metal']
```

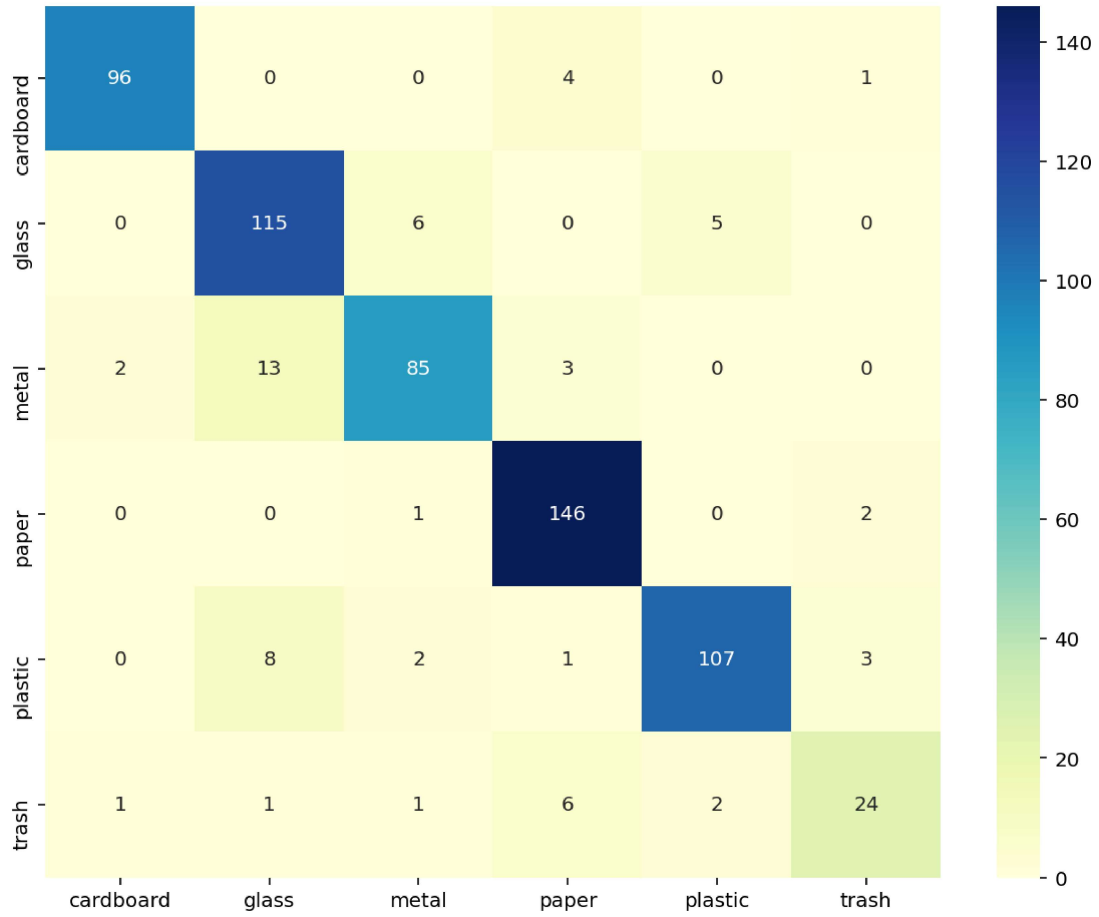
```
learn.data.test_ds[0][0]
```


```
cm = confusion_matrix(y,yhat)
print(cm)
```

```
[[ 96   0   0   4   0   1]
 [  0 115   6   0   5   0]
 [  2  13  85   3   0   0]
 [  0   0   1 146   0   2]
```

```
df_cm = pd.DataFrame(cm,waste_types,waste_types)

plt.figure(figsize=(10,8))
sns.heatmap(df_cm,annot=True,fmt="d",cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f46b2b901d0>
```




```
correct = 0

for r in range(len(cm)):
    for c in range(len(cm)):
        if (r==c):
            correct += cm[r,c]


accuracy = correct/sum(sum(cm))
accuracy
```

```
0.9023622047244094
```

```
## delete everything when you're done to save space
shutil.rmtree("data")
shutil.rmtree('dataset-resized')
```