Downloading **DataSet**

using BingIMageSearch - crack

```
from IPython.core.display import Image
#image downloader bing search image crack without api
from bing_image_downloader import downloader
downloader.download('Black Bear', limit=100,  output_dir='dataset/Bear',
                    adult_filter_off=True, force_replace=False, timeout=600, verbose=

downloader.download('Teddy Bear', limit=100,  output_dir='dataset/Bear',
                    adult_filter_off=True, force_replace=False, timeout=600, verbose=

downloader.download('Grizzly Bear', limit=100,  output_dir='dataset/Bear',
                    adult_filter_off=True, force_replace=False, timeout=600, verbose=
```

**Output Directory will be like **

[Path('dataset/Bear/BlackBear/Image_1.jpg'),Path('dataset/Bear/TeddyBear/Image_1.jpg.jpg')...]

Showing Downloaded Data

```
# #showing images from directory
# from PIL import Image as im
# global str
# for i in range(20):
#   imgStr="Image_"
#   imgStr=(f'{imgStr}{i+1}')
#   print(imgStr)
#   Image.open('dataset/cat/cat/'+imgStr+'.jpg')
```

Building Model

- **Step 1**
  - Tell Fastai Type of Data and how it is Structured

```
from fastai import *
```

```
from fastai.vision import *
from fastbook import *

bears = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.3, seed=42),
    get_y=parent_label,
    item_tfms=Resize(128))

dls= bears.dataloaders('dataset/Bear')
#dls.valid.show_batch(max_n=30,nrows=6)  #can be used anywhere to see your data
```

Step 2

- Train Model
- Clean Data

Notes :

-Called CNN to create a architecture of 18 layers [resnet18]. -fine_tune will train pretrained model.

```
learn=vision_learner(dls,resnet34,metrics=error_rate)
learn.fine_tune(10)
```

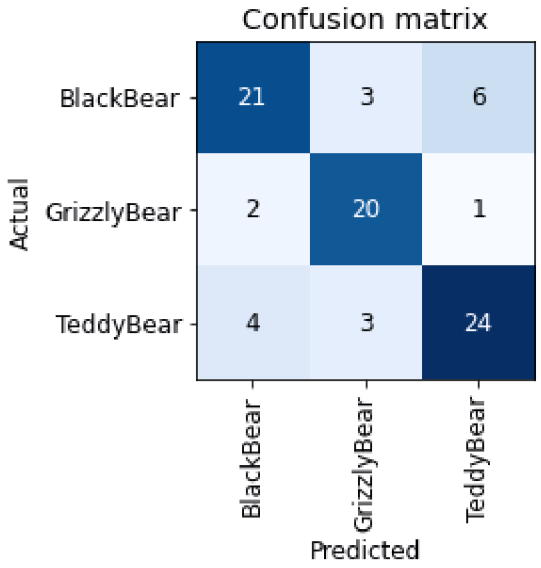
epoch	train_loss	valid_loss	error_rate	time
0	2.055591	1.994754	0.511905	00:34

```
/usr/local/lib/python3.7/dist-packages/PIL/Image.py:960: UserWarning: Palette i
  "Palette images with Transparency expressed in bytes should be "
```

epoch	train_loss	valid_loss	error_rate	time
0	1.371218	1.215280	0.464286	00:44
1	1.143828	0.895549	0.392857	00:45
2	0.922690	0.765632	0.297619	00:44
3	0.751675	0.792297	0.238095	00:45
4	0.626648	0.789966	0.273810	00:46
5	0.524576	0.827829	0.261905	00:50
6	0.455256	0.831248	0.238095	00:45

Confusion Matrix

```
interp=ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()
# interp.plot_top_losses(3,nrows=3)
```



Turning Model to a program / Testing

```
learn.export()
path=Path()
```

```
path.ls(file_exts='.pkl')
```

```
learn_inf=load_learner(path/'export.pkl')
```

```
learn_inf.dls.vocab
```

```
['BlackBear', 'GrizzlyBear', 'TeddyBear']
```

```
learn_inf.predict('Image_19.jpg')
```

```
('TeddyBear', TensorBase(2), TensorBase([1.3252e-05, 1.4879e-04,
9.9984e-01]))
```