```
pip install fastai
```

```
pip install fastbook
```

```
from fastai import *
from fastbook import *
```

Getting Deep into NN's

- Look inside NN as it trains
- Look inside NN as it makes predictions
- Find Possible Problem
- Find Optimal Good Solutions

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

```
100.00% [811712512/811706944 01:01<00:00]
```

```
Path.BASE_PATH = path
```

```
path.ls()
```

```
(#2) [Path('images'),Path('annotations')]
```

```
(path/'images').ls()
```

```
(#7393) [Path('images/Abyssinian_18.jpg'),Path('images
/yorkshire_terrier_184.jpg'),Path('images/yorkshire_terrier_61.jpg'),Path('images
/american_pit_bull_terrier_187.jpg'),Path('images/Abyssinian_73.jpg'),Path('images
/Maine_Coon_193.jpg'),Path('images/Bengal_99.jpg'),Path('images
/Bengal_45.jpg'),Path('images/Russian_Blue_57.jpg'),Path('images
/newfoundland_51.jpg')...]
```

DataSet

- First Letter UPPERCASE -> Cat
- First Letter lowecase -> Dog
- Label with its breed - using its structure

Regular Expressions Technique

- recognizing patters
- using to extract info from strings

```
fname=(path/'images').ls()[0]
fname
```

```
Path('images/Abyssinian_18.jpg')
```

Description

'(.+)_\d+.jpg$'

- get anything inside ()
- .+ -> may repeat n number of times
- _ followed by a underscore
- \ dont treat backslashes special
- d followed by a digit
- +. repeat n number of times
- .jpg followed by this
- $ end of file name

```
re.findall( r'(.+)_\d+.jpg$', fname.name)
```

```
['Abyssinian']
```

DataBlock

- Telling Fastai about
- data, labells (independent,dependent)
- Splitting Techniques
- labels

Presizing approach by Fastai

- Resize Crop
- Data Augmentation
- batch size

Two Step Approach

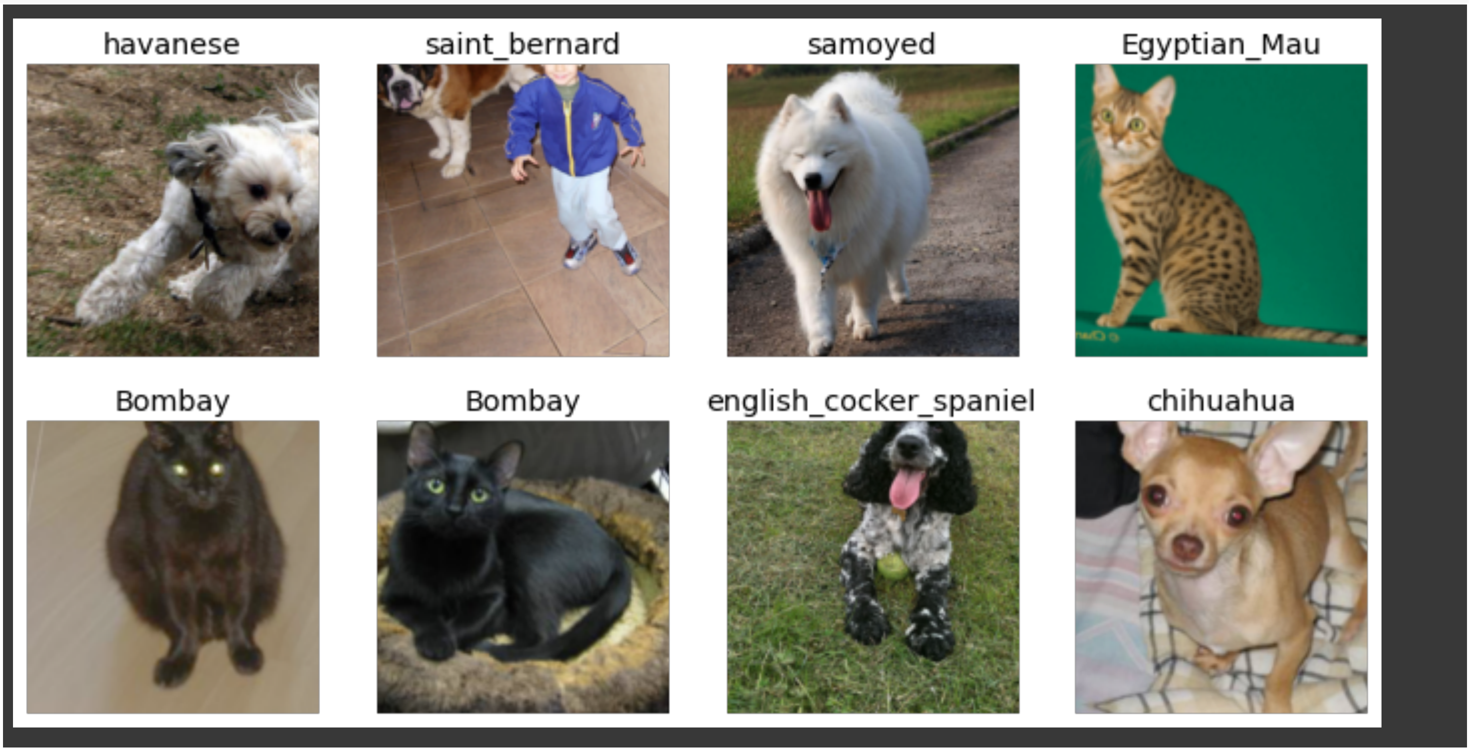
- item_tfms -> Crop full width or height
- batch_tfm -> Random crop and augment
- https://github.com/fastai/fastbook/blob/master/05_pet_breeds.ipynb

```
DataBlock??
```

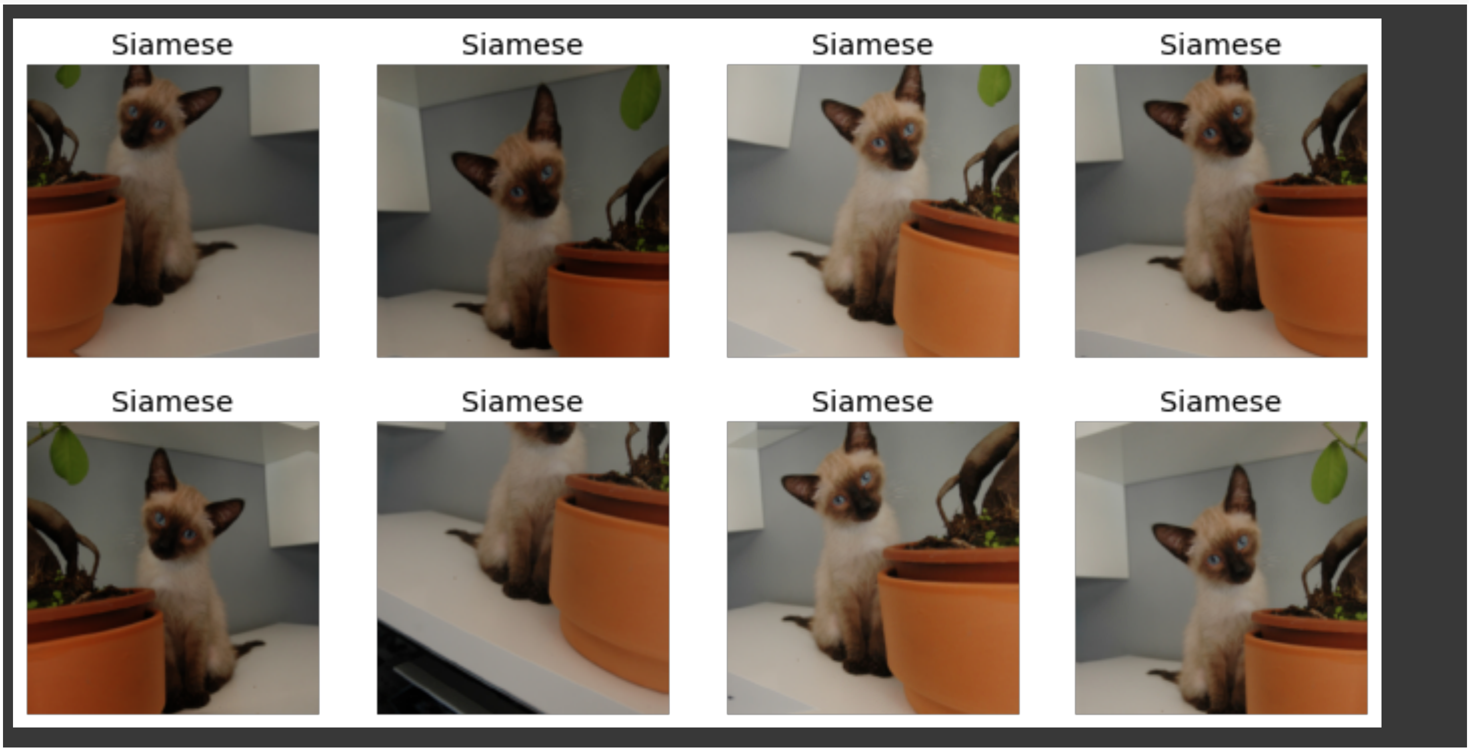
```
pets= DataBlock(
              #structure of dataBlock (independent,dependent)
                blocks=(ImageBlock,CategoryBlock),
              #dets independent variable
```

```
        get_items=get_image_files,
       #splitts shuffles dataset
        splitter= RandomSplitter(seed=42),
       #extracts labels form file names
        get_y=using_attr(RegexLabeller(r'(.+)_\d+.jpg$'),'name'),
       # Random Sesize crop large images into 460 squares ,
        item_tfms=Resize(460),
       # data augmentation (multi perspectives )
       # now applied upper small squres (460 pxl)
        batch_tfms=aug_transforms(size=224,min_scale=0.75))
dls=pets.dataloaders(path/'images')
```

```
dls.show_batch(nrows=2,ncols=4)
```



```
dls.show_batch(unique=True,nrows=2,ncols=4)
```



Summary of DataBlcok

```
pets.summary(path/'images')
```

```
Setting-up type transforms pipelines
Collecting items from /root/.fastai/data/oxford-iiit-pet/images
Found 7390 items
2 datasets of sizes 5912,1478
Setting up Pipeline: PILBase.create
Setting up Pipeline: partial -> Categorize -- {'vocab': None, 'sort': True, 'add_na':

Building one sample
  Pipeline: PILBase.create
    starting from
      /root/.fastai/data/oxford-iiit-pet/images/Siamese_138.jpg
    applying PILBase.create gives
      PILImage mode=RGB size=333x500
  Pipeline: partial -> Categorize -- {'vocab': None, 'sort': True, 'add_na': False}
    starting from
      /root/.fastai/data/oxford-iiit-pet/images/Siamese_138.jpg
    applying partial gives
      Siamese
    applying Categorize -- {'vocab': None, 'sort': True, 'add_na': False} gives
      TensorCategory(10)

Final sample: (PILImage mode=RGB size=333x500, TensorCategory(10))


Collecting items from /root/.fastai/data/oxford-iiit-pet/images
Found 7390 items
2 datasets of sizes 5912,1478
Setting up Pipeline: PILBase.create
Setting up Pipeline: partial -> Categorize -- {'vocab': None, 'sort': True, 'add_na':
Setting up after_item: Pipeline: Resize -- {'size': (460, 460), 'method': 'crop', 'pad
Setting up before_batch: Pipeline:
Setting up after_batch: Pipeline: IntToFloatTensor -- {'div': 255.0, 'div_mask': 1} ->

Building one batch
Applying item_tfms to the first sample:
  Pipeline: Resize -- {'size': (460, 460), 'method': 'crop', 'pad_mode': 'reflection',
    starting from
      (PILImage mode=RGB size=333x500, TensorCategory(10))
    applying Resize -- {'size': (460, 460), 'method': 'crop', 'pad_mode': 'reflection'
      (PILImage mode=RGB size=460x460, TensorCategory(10))
    applying ToTensor gives
      (TensorImage of size 3x460x460, TensorCategory(10))

Adding the next 3 samples

No before_batch transform to apply

Collating items in a batch

Applying batch_tfms to the batch built
  Pipeline: IntToFloatTensor -- {'div': 255.0, 'div_mask': 1} -> Flip -- {'size': None
    starting from
      (TensorImage of size 4x3x460x460, TensorCategory([10,  4, 18,  0], device='cuda:
    applying IntToFloatTensor -- {'div': 255.0, 'div_mask': 1} gives
      (TensorImage of size 4x3x460x460, TensorCategory([10,  4, 18,  0], device='cuda:
    applying Flip -- {'size': None, 'mode': 'bilinear', 'pad_mode': 'reflection', 'mode
      (TensorImage of size 4x3x460x460, TensorCategory([10,  4, 18,  0], device='cuda:
    applying RandomResizedCropGPU -- {'size': (224, 224), 'min_scale': 0.75, 'ratio':
```

Building The Model

Building The Model

Loss Funtion Picked by Fastai

- we have image data
- categorial outcome
- so it picked

Cross Entropy loss

- same as mnist loss i created in previous model -> minnist...(sigmoid , 1- predictions)

Additional Benifits

  - Works even when dependent varaible has more than two categories **(because here we have n number of breeds cant just use 1/0 binary prediction - we have n number of labels)**
  - Faster and more Reliable Training

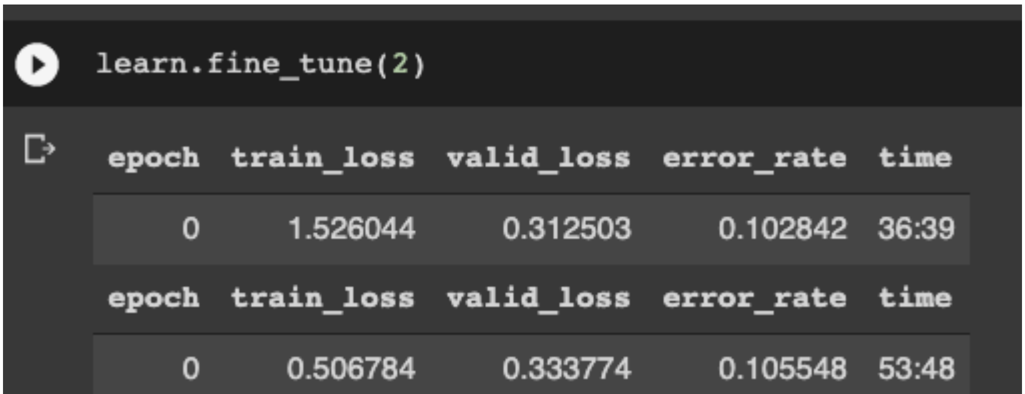
```
# fastai automatically picked loss funtion as we didnt passed
learn=cnn_learner(dls,resnet34,metrics=error_rate)
```

```
/usr/local/lib/python3.7/dist-packages/fastai/vision/learner.py:287: UserWarning: `cnn_
  warn("`cnn_learner` has been renamed to `vision_learner` -- please update your code"
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:136: UserWarning: 
  f"Using {sequence_to_str(tuple(keyword_only_kwargs.keys()), separate_last='and ')} a
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWarning: 
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet34-b627a593.pth" to /root/.cac
100% 83.3M/83.3M [00:01<00:00, 84.9MB/s]
```

```
learn.fine_tune(2)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.513739	0.313420	0.100135	01:11

epoch	train_loss	valid_loss	error_rate	time
0	0.497575	0.318589	0.092016	01:13
1	0.323638	0.243381	0.069012	01:14



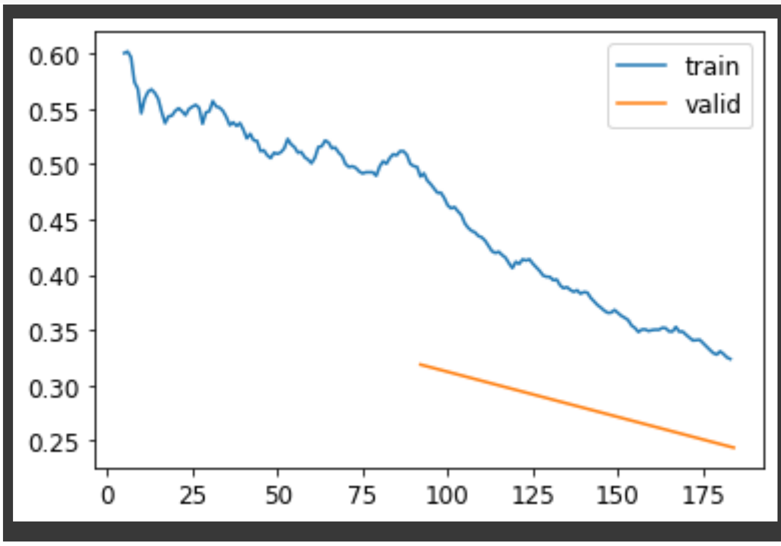
```
learn.fine_tune(2)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.526044	0.312503	0.102842	36:39

epoch	train_loss	valid_loss	error_rate	time
0	0.506784	0.333774	0.105548	53:48

```
      1      0.309370      0.222299      0.071042    54:59
```

```
learn.recorder.plot_loss()
```

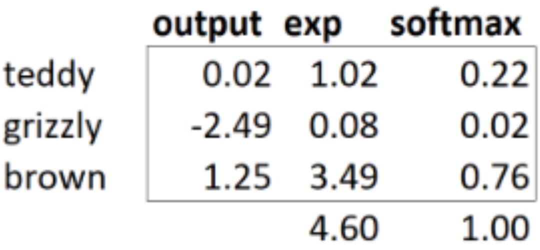


Inside Loss Funtion Used By Fastai

Cross Entropy Loss()

**Softmax - First Part of Cross Entropy Loss() **

- Extended Version of Sigmoid used forn number of labels .
- Works as per Probability Priniciple - all prediction/activation add to 1.0
- detail notes in register
- What does this function do in practice? Taking the exponential ensures all our numbers are positive, and then dividing by the sum ensures we are going to have a bunch of numbers that add up to 1.
- The exponential also has a nice property: if one of the numbers in our activations x is slightly bigger than the others, the exponential will amplify this (since it grows, well... exponentially), which means that in the softmax, that number will be closer to 1.



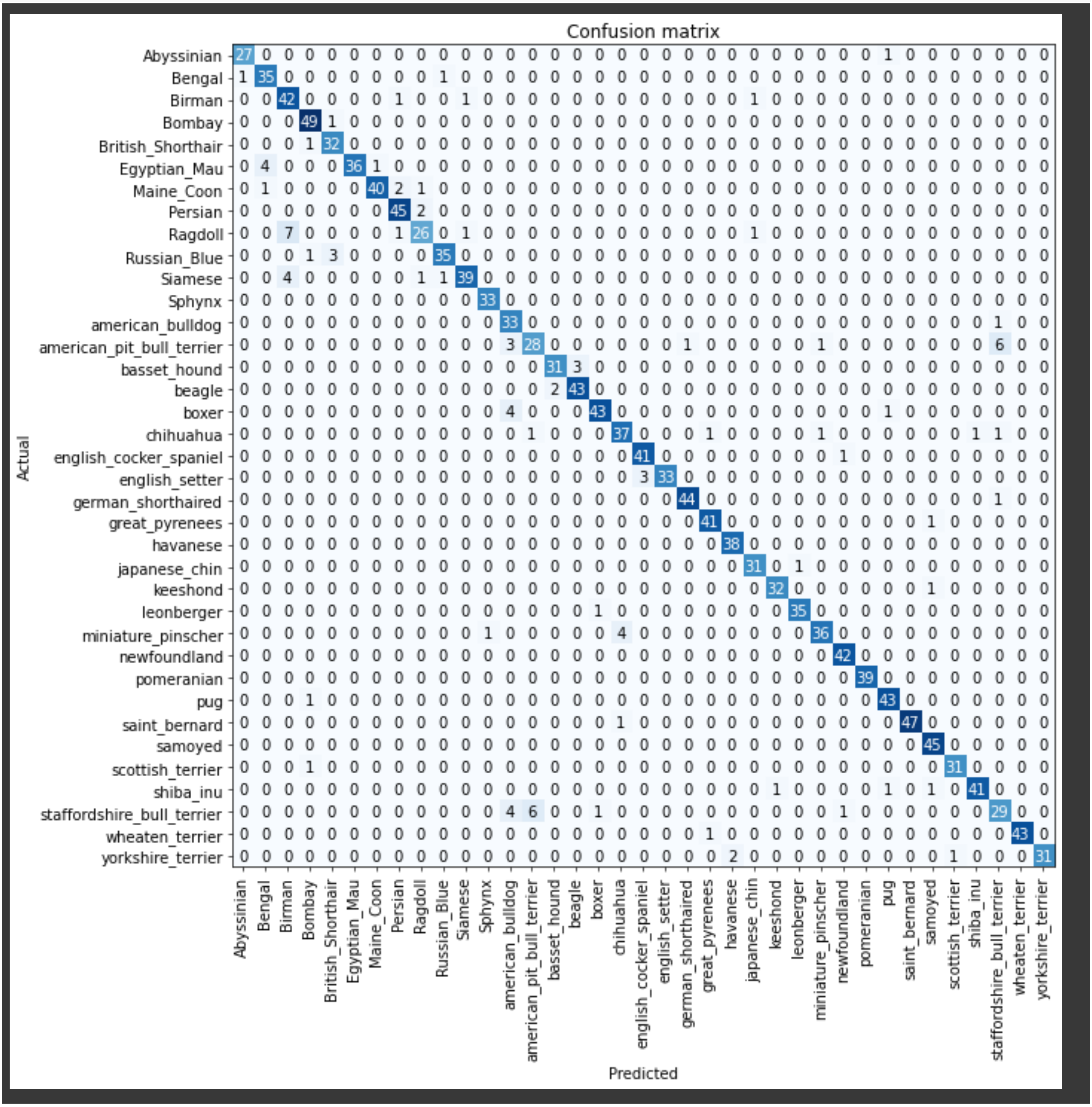
	output	exp	softmax
teddy	0.02	1.02	0.22
grizzly	-2.49	0.08	0.02
brown	1.25	3.49	0.76
		4.60	1.00

- https://github.com/fastai/fastbook/blob/master/05_pet_breeds.ipynb

[ ] ↳ *3 cells hidden*

Confusion Matrix

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



Most Confusion

- Method to only get where model is most confused
- this big confusion matrix is hard to read
- quick

```
interp.most_confused(min_val=5)
```

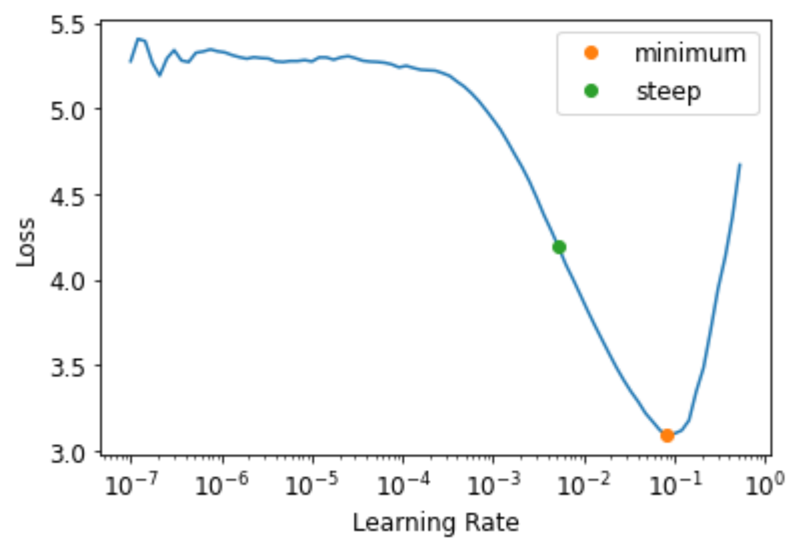
```
[('Ragdoll', 'Birman', 7),
 ('american_pit_bull_terrier', 'staffordshire_bull_terrier', 6),
 ('staffordshire_bull_terrier', 'american_pit_bull_terrier', 6)]
```

Learning Rate Finder

- method automaticlly finds the optimal lr for the batch
- To train quick

```
learn = vision_learner(dls, resnet34, metrics=error_rate)
lr_min,lr_steep = learn.lr_find(suggest_funcs=(minimum, steep))
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:136: UserWarning:
  f"Using {sequence_to_str(tuple(keyword_only_kwargs.keys()), separate_last='and ')} a
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWarning:
  warnings.warn(msg)
```



```
print(f"Minimum/10: {lr_min:.2e}, steepest point: {lr_steep:.2e}")
```

```
Minimum/10: 8.32e-03, steepest point: 5.25e-03
```

Using This lr now to fine tune the model

```
learn = vision_learner(dls, resnet34, metrics=error_rate)
learn.fine_tune(1, base_lr=25e-3)
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:136: UserWarning:
  f"Using {sequence_to_str(tuple(keyword_only_kwargs.keys()), separate_last='and ')} a
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWarning:
  warnings.warn(msg)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.111147	0.909922	0.169147	01:10

epoch	train_loss	valid_loss	error_rate	time
0	1.555150	0.613240	0.182679	01:14

Using Higher Learning Rate increased the error rate

- because higher lr makes bigger jumps than 0.001 used back