Chapter 2 Part 3 요약 (Page 70-78) 090120 강인승

From Data to DataLoaders

- DataLoaders 는 DataLoader 의 object 를 포함하는 class 이다
  - 이렇게 지정하면 train 과 valid 함수를 사용할 수 있는 형태가 됨
  - 비슷한 성향을 가지고 있는 Dataset 과 Datasets class 도 있다
- DataLoaders 에 다운로드한 데이터를 집어넣을 때 고려해야하는 부분
  - 어떤 type 의 데이터인지?
  - 아이템을 나열할 수 있는지
  - 아이템을 라벨링 하는 방법
  - Validation set 를 만드는 방법
- 이런 DataLoaders 를 만든느데 있어 FastAI 의 data block API 를 사용할 수 있다

```
In [ ]: bears = DataBlock(
            blocks=(ImageBlock, CategoryBlock),
            get_items=get_image_files,
            splitter=RandomSplitter(valid_pct=0.2, seed=42),
            get_y=parent_label,
            item_tfms=Resize(128))
```

- 인풋과 아웃풋의 형태를 지정한다 (independent and dependent variable)

```
blocks=(ImageBlock, CategoryBlock)
```

- 데이터의 파일들을 리스트의 형태로 불러온다 (`get_image_files` 함수는 주어진 path 안의 모든 이미지 파일을 불러냄)

```
get_items=get_image_files
```

- 어떤 데이터는 미리 training 과 validation 데이터가 나눠져서 제공되는 경우도 있음
- 기본적인 방법은 무작위로 나누는 방법이다 (seed 는 random 한 리스트를 생성하는데 사용)

```
splitter=RandomSplitter(valid_pct=0.2, seed=42)
```

- 아래와 같이 fastai 에서 데이터에 라벨링을하는 함수를 지정한다

```
get_y=parent_label
```

  - 이 parent_label 함수는 해당 데이터가 저장되어있는 폴더의 이름을 제공
  - 일반적으로 해당 데이터 이미지를 관련 폴더에 저장하기에 이 방법으로 라벨링을 함

- 딥러닝 모델을 학습시킬대 여러가지 데이터를 mini-batch 형태로 제공하면서 학습시키기 때문에 데이터 사이즈가 같아야 한다
  - Transform 을 통해 이미지를 resize 한다 (fastai 에는 여러가지 형태의 transform 이 제공된다)
  - Resize 를 통해 128 픽셀로 변형한다

```
item_tfms=Resize(128)
```

- 그렇게해서 bear datablock object 를 생성한 후 dataloaders 를 이용해서 실제 데이터를 기반한 class 를 만든다

```
dls = bears.dataloaders(path)
```

- 이 클래스는 기본적으로 batch 크기가 64 인데 (학습에 사용되는 묶음, 64 개의 데이터 stack 이 하나의 tensor 이다)
- Show_batch 를 통해 이런 batch 의 형태를 볼 수 있다

```
In [ ]: dls.valid.show_batch(max_n=4, nrows=1)
```

- Resize 함수는 기본적으로 크로핑을 하는데 (이미지를 정사각형으로 만듬) 이런 경우 데이터 유실의 가능성이 있다
  - 비슷한 방법으로 fastai 에서는 padding (이미지를 zero (검정색)로 채움) 또는 줄이고/늘리는 것도 가능
    - 패딩

```
bears = bears.new(item_tfms=Resize(128, ResizeMethod.Pad, pad_mode='zeros'))
dls = bears.dataloaders(path)
dls.valid.show_batch(max_n=4, nrows=1)
```

    - 줄이기/늘리기

```
bears = bears.new(item_tfms=Resize(128, ResizeMethod.Squish))
dls = bears.dataloaders(path)
dls.valid.show_batch(max_n=4, nrows=1)
```

- 이런 방법들은 대체로 효율적이지 않음
  - 강아지와 고양이의 종류를 구분하는데 중요한 부분이 크로핑 될 경우
  - 패딩을 통해 불필요한 공간이 모델 계산에 들어가는 경우

- 모델을 학습할 때 이미지의 특정 부분을 randomly crop 해서 batch 에 넣어 학습시킴 (Resize/RandomResizedCrop)
    - 매번 epoch 시 이 cropping 되는 부분이 randomize 됨
    - 모델이 매 iteration 마다 이미지의 다른 부분을 학습하며 generalize 될 수 있음
- 이런 resizing 방법에 중요한 파라미터인 min_scale (매번 epoch 시에 최소 선택 되는 이미지의 크기)

```
bears = bears.new(item_tfms=RandomResizedCrop(128, min_scale=0.3))
dls = bears.dataloaders(path)
dls.train.show_batch(max_n=4, nrows=1, unique=True)
```

- 여기서 `unique=True` 를 지정함으로써 같은 사진을 반복적으로 resize crop 하게 한다

Data Augmentation

- 데이터 증강은 인풋 데이터에 random variation 을 주는 것
    - 이미지의 rotation, flipping, perspective warping, brightness and contrast changing
    - 이런 데이터 증강에는 FastAI 의 `aug_transforms` 함수 사용
- 한 batch 의 데이터가 같은 사이즈로 정의됐기 때문에 GPU 를 이용해서 한번에 증강 가능
    - batch_tfms 파라미터 사용

```
bears = bears.new(item_tfms=Resize(128), batch_tfms=aug_transforms(mult=2))
dls = bears.dataloaders(path)
dls.train.show_batch(max_n=8, nrows=2, unique=True)
```

Training Your Model, and Using It to Clean Your Data

- bear classification 의 예제 (곰의 종류 당 150 가지의 이미지가 있음)

```
bears = bears.new(
    item_tfms=RandomResizedCrop(224, min_scale=0.5),
    batch_tfms=aug_transforms())
dls = bears.dataloaders(path)
```

- 만들어진 dataloader object 의 bear 데이터를 이용하여 resnet18 모델 학습

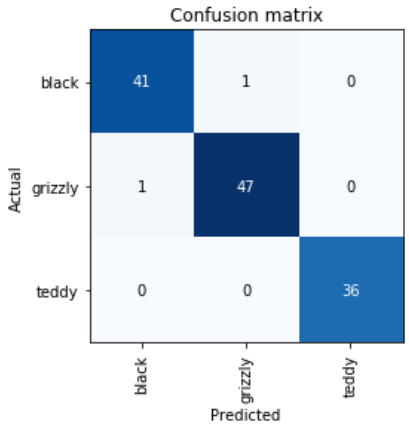
```
learn = cnn_learner(dls, resnet18, metrics=error_rate)
learn.fine_tune(4)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.235733	0.212541	0.087302	00:05

epoch	train_loss	valid_loss	error_rate	time
0	0.213371	0.112450	0.023810	00:05
1	0.173855	0.072306	0.023810	00:06
2	0.147096	0.039068	0.015873	00:06
3	0.123984	0.026801	0.015873	00:06

- 모델의 퍼포먼스를 *confusion matrix* 를 통해서 볼 수 있다 (validation set 을 이용)

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



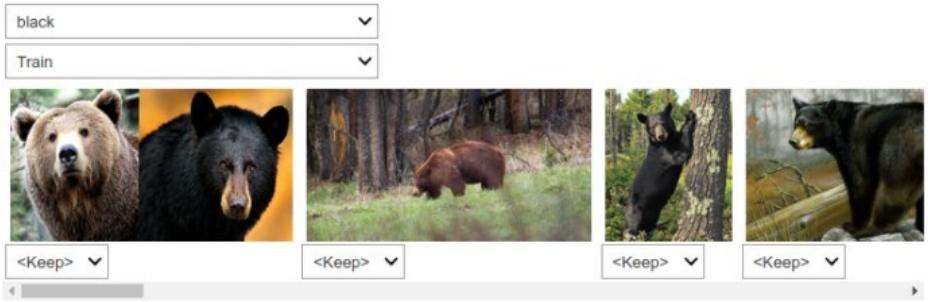
- 이러한 방법은 에러가 어디서 발생하는지 확일 할 수 있다
  - 데이터의 문제 (라벨링이 잘못되었거나, 전혀 다른 이미지가 껴있는 경우)
  - 학습된 모델의 문제 (사용된 이미지의 빛번짐, 각도 등)
- Loss 값을 이용하여 이미지를 분류할 수 있음
  - Plot_top_losses 함수는 데이터 안에서 가장 큰 에러를 발생하는 이미지를 보여준다
  - Probability 는 모델이 예측하는 결과에 대한 confidence level 을 0 ~ 1 사이의 값으로 표현한다

```
interp.plot_top_losses(5, nrows=1)
```

- 첫번째 이미지의 경우 confidence level 과 loss 가 높음:라벨링이 잘못된 경우
- 일반적으로 데이터를 정제한 후 모델을 학습함
    - 간단한 모델을 만들고 evaluate 해서 데이터를 정제하는 방법도 있음
- FastAI 는 GUI 기반의 데이터 정제 툴을 제공함: `ImageClassifierCleaner`

```
cleaner = ImageClassifierCleaner(learn)
cleaner
```

```
VBox(children=(Dropdown(options=('black', 'grizzly', 'teddy'), value='black'), Dropdown(options=('Train', 'Val…
```



- 실제로 이미지를 삭제하는 것이 아닌, 해당 관련 파일의 index 를 제공함 (unlink 사용)

```
for idx in cleaner.delete(): cleaner.fns[idx].unlink()
```

- 다른 카테고리로 이미지를 이동시키려면

```
for idx,cat in cleaner.change(): shutil.move(str(cleaner.fns[idx]),
path/cat)
```