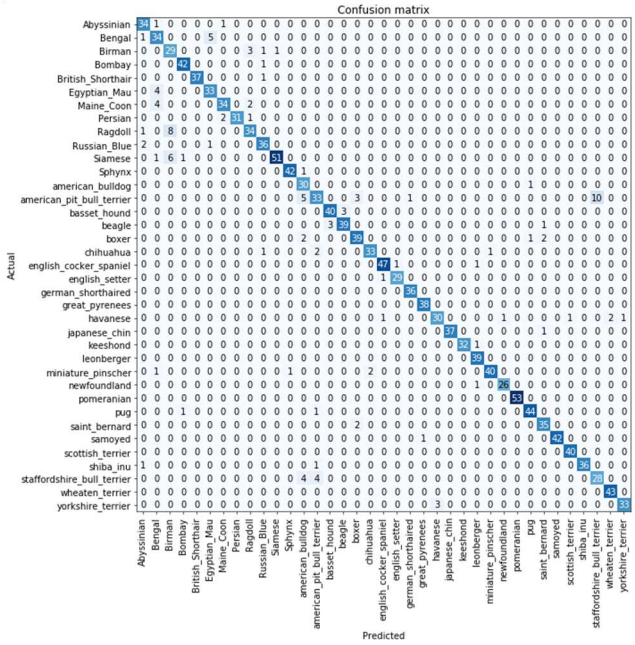
Chapter 5 Part 3 요약 (Page 203-210) 092320 강인승

Model Interpretation

- Loss 함수는 직관적으로 이해하기 힘들다 (사람이 이해하기 쉬운 metric 으로 퍼포먼스를 보여줌)
- 이런 metric 을 이용한 퍼포먼스가 어디서 나쁘고 좋은지 알 수 있음 (Confusion Matrix)

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



- Label 이 많기 때문에 (37) 이해하기 안좋음
  - Classification error 가 높은 상위 class 를 보여줄 수 있음

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

```
[('american_pit_bull_terrier', 'staffordshire_bull_terrier', 10),
 ('Ragdoll', 'Birman', 8),
 ('Siamese', 'Birman', 6),
 ('Bengal', 'Egyptian_Mau', 5),
 ('american_pit_bull_terrier', 'american_bulldog', 5)]
```

Improving Our Model

- Baseline 에서 더 좋은 퍼포먼스를 위해 추가적으로 Model 을 학습 할 수 있음 (Transfer Learning)
  - 미리 학습 된 weight 을 사용함

The Learning Rate Finder

- Model 학습에 가장 중요한 부분 중 하나는 적합한 learning rate 을 찾는 것
  - 학습시간이 증가할 수 있음
  - Model 이 Overfitting 될 수 있음
- Learning rate 가 높았을 때의 예

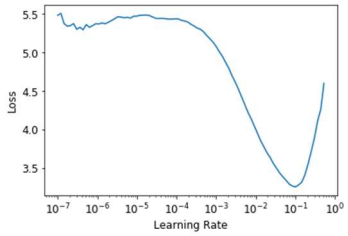
```
learn = cnn_learner(dls, resnet34, metrics=error_rate)
learn.fine_tune(1, base_lr=0.1)
```

epoch	train_loss	valid_loss	error_rate	time
0	2.778816	5.150732	0.504060	00:20
epoch	train_loss	valid_loss	error_rate	time
0	4.354680	3.003533	0.834235	00:24

  - Optimizer 는 맞는 방향으로 갔으나, 너무 step 이 커서 min loss 를 뛰어서 에러가 올라감

- 2015 Leslie Smith 의 learning rate finder
    - 처음에 작은 learning rate 로 시작해서 1 mini-batch 마다 점진적으로 증가 (e.g. 두배로 만든다)
    - Loss 가 내려가다 다시 올라가기 시작하면 그 직전의 포인트의 learning rate 를 사용

```
learn = cnn_learner(dls, resnet34, metrics=error_rate)
lr_min,lr_steep = learn.lr_find()
```



    - 위의 예에서는 Learning rate 가 로그 스케일인걸 생각할 때 3e-3 정도를 정함

Unfreezing and Transfer Learning

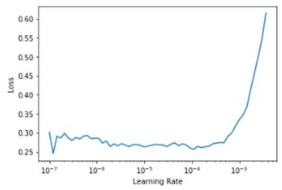
- 기존의 학습이 된 model 을 (예를 들어 ImagNet) 이용하여 fine-tune 할 수 있음
- 일반적인 CNN model 의 구조는 nonlinear activation function 을 가진 여러개의 layer 기반인 Conv 모델에 마지막에 아웃풋과 사이즈가 맞는 (number of class) linear layer 를 이어놓는 구조임
    - 마지막 layer 는 application specific 할 가능성이 많음
    - 전이학습을 하게되면 기존 모델의 마지막 linear layer 를 원하는 application 에 맞춰서 변형시키면 됨
        - 기존 앞의 layer 들은 미리 학습된 모델의 weight 들을 사용하고 마지막 새로 변경된 layer 만 random initial weight 을 가지게 됨
- CNN 의 앞부분 layer 들은 일반적인 컨셉을 extract 하는 layer 들임
    - Gradient, edge detection 등등
- CNN 의 마지막 layer 들은 application 의 특화되어 있는 내용을 classify 하는 layer 임
    - Eyeball, fur 등등
- 이런 구조의 학습은 (마지막/추가된 layer 의 weight 만 학습) 앞의 layer 들을 freezing 해야 함
    - Fastai 는 미리 학습된 model 을 불러오면 자동적으로 freeze 함
    - Fine_tune 함수는 두가지 기능을 가지고 있음
        - 추가 layer 를 1 epoch 학습함 (앞의 layer 들은 freeze 한 상태)
        - 전체 layer 를 다 unfreeze 하고 유저가 지정한 epoch 개수만큼 학습함
- Transfer learning 의 예
    - Fit_one_cycle 함수를 사용할 수도 있음 (마지막 layer 만 추가적으로 학습)
        - 처음에 낮은 learning rate 로 시작하여 점진적으로 learning rate 를 크게하다가 마지막에는 다시 learning rate 를 줄이는 방법

```
learn = cnn_learner(dls, resnet34, metrics=error_rate)
learn.fit_one_cycle(3, 3e-3)
```

epoch	train_loss	valid_loss	error_rate	time
0	1.188042	0.355024	0.102842	00:20
1	0.534234	0.302453	0.094723	00:20
2	0.325031	0.222268	0.074425	00:20

```
learn.unfreeze()
```

        - 그리고 다시 learning rate 를 찾는다 (unfreeze 이후에는 전체 model 을 학습시키기에 기존 learning rate 이 적합하지 않을 수 있음)



- Model 이 기존에 학습되었던 model 이기에 낮은 learning rate 일 때 loss 가 flat 함
- 새로운 learning rate 로 추가 학습을 시킴

```
1 learn.fit_one_cycle(6, lr_max=1e-5)
```

epoch	train_loss	valid_loss	error_rate	time
0	0.263579	0.217419	0.069012	00:24
1	0.253060	0.210346	0.062923	00:24
2	0.224340	0.207357	0.060217	00:24
3	0.200195	0.207244	0.061570	00:24
4	0.194269	0.200149	0.059540	00:25
5	0.173164	0.202301	0.059540	00:25

- Model 의 학습에 있어 layer 의 위치별로 (앞부분, 뒷부분 layer) learning rate 가 다르게 적용할 수 있음
  - Discriminative Learning Rate