1. 네트워크 크기 & 네트워크 아키텍쳐 조정

- train accuracy를 높이기 위해 시도.

1) 기존 모델에서 SeperableConv2D 층을 네 번 쌓은 블록을 하나 더 추가한 모델을 만듦
2) 기존 모델에서 SeperableConv2D를 Conv2D로 바꾼 모델을 만듦.
3) SeperableConv2D를 Conv2D로 바꾸고 블록이 네 개인 모델을 만듦.

결과

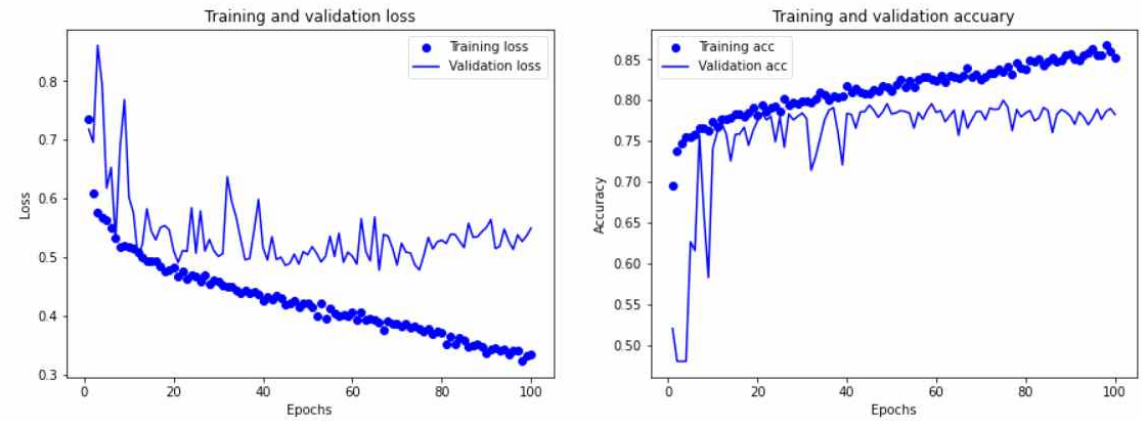
기존 모델과 성능을 비교하였을 때 1번 모델이 기존 모델과 성능이 유사하였고 2, 3번 모델은 에포크마다 성능 변화가 컸음. 따라서 1번 모델은 기존 모델과 성능이 유사하면서 학습 시간이 더 길기 때문에 기존 아키텍쳐를 유지하기로 하였음.

2. 최적화 알고리즘 적용

- Adagrad, Momentum, RMSProp, Adadelta, Adam 최적화 알고리즘 비교

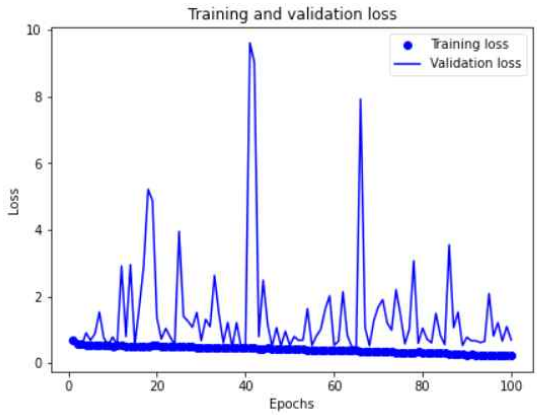
1) Adagrad

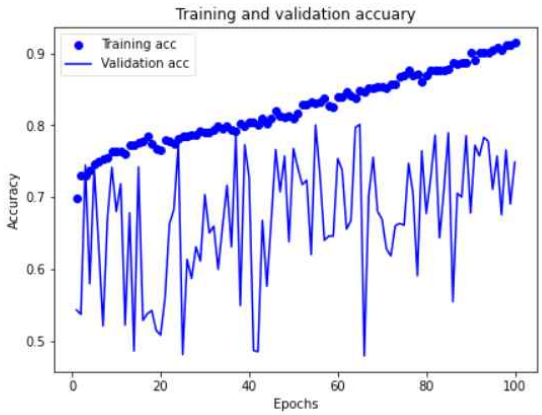
학습을 진행하면서 학습률을 점차 줄여가는 방식. 개별 매개변수 각각에 대해 학습률을 조정하며 학습. 갈수록 보폭을 줄여 세밀하게 탐색하는 셈.



2) Momentum

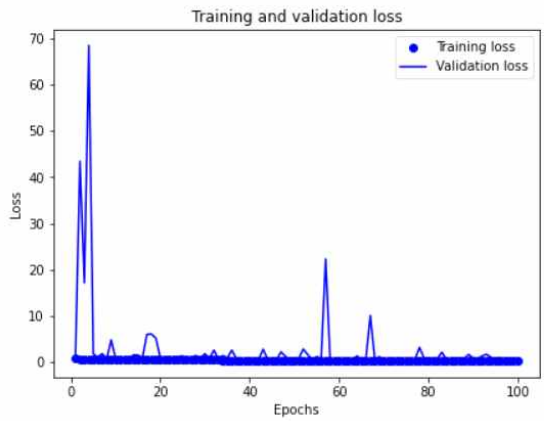
iteration에 따라 방향이 반대로 계속 바뀌는 경로를 지나온 경우 해당 방향으로 update하는 변화량이 점차 작아지고, iteration에 따라 방향이 계속 유지되면 가속을 붙여주는 방식. 관성을 따라 진행.

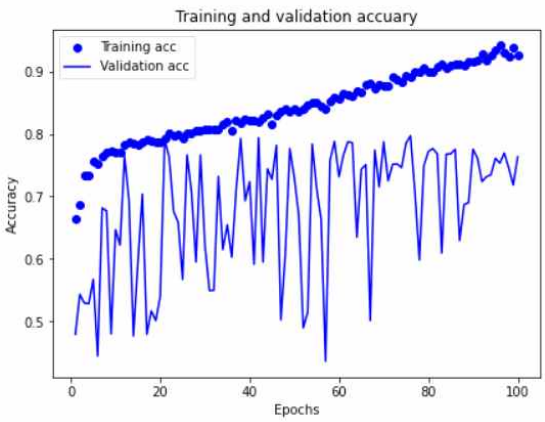




3) RMSProp

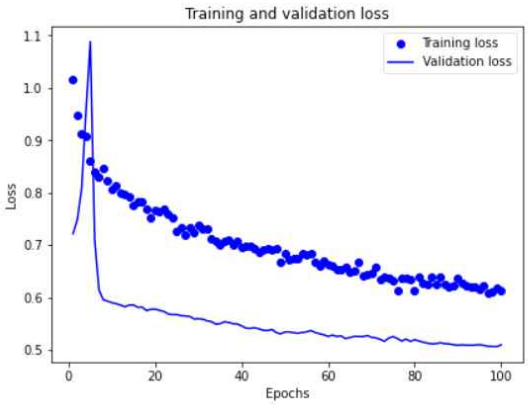
Gradient의 방향을 이용하지 않고 그 크기만을 이용해 업데이트 해주고자 하는 각 parameter에 대한 학습 속도를 조절한다. Adagrad보다 이전 맥락을 고려해 학습률을 조절.

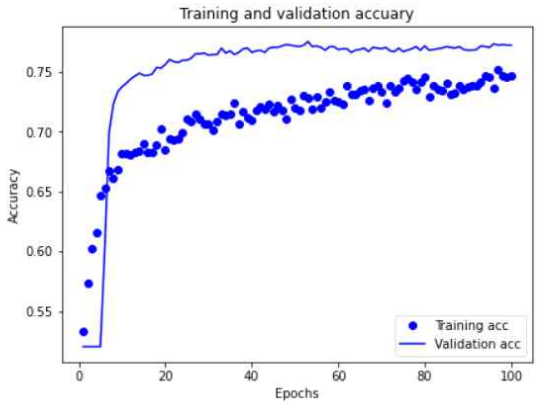




4) Adadelta

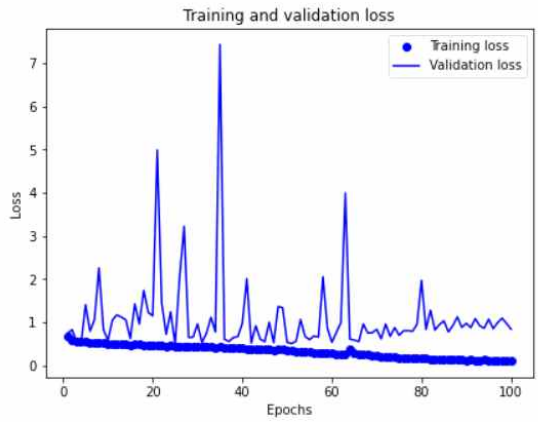
iteration마다 learning rate가 작아지는 Adagrad의 문제점을 해결하고자 고안된 알고리즘.

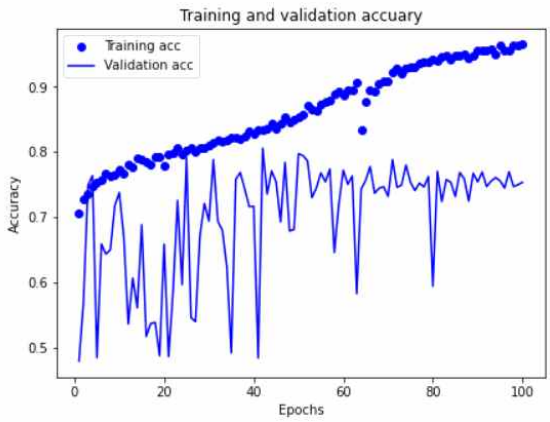




5) Adam

RMSProp과 Momentum의 융합. gradient descent를 진행하는 방향도, 스텝의 크기도 적당하게 택하고자 하는 알고리즘.





결론

Adagrad와 Adadelta 최적화 알고리즘 채택.

3. 네트워크 정규화

- 과대적합 방지 목적
- L2 정규화, 드롭아웃 비교

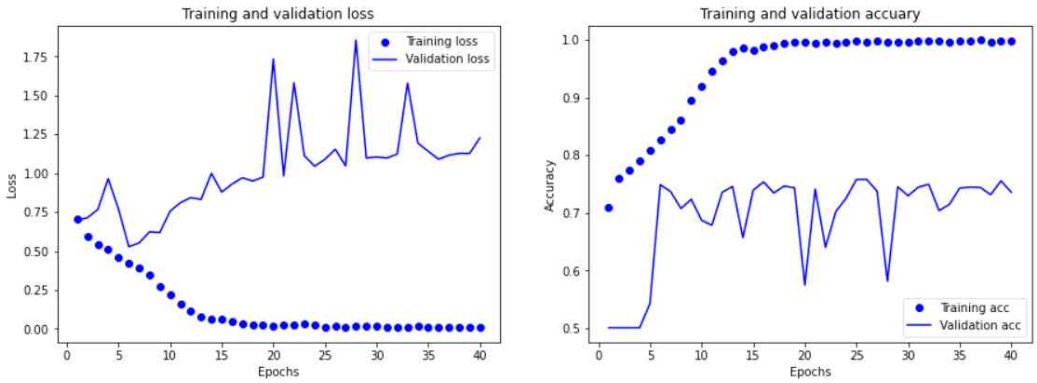
1) L2 정규화

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

$$\begin{aligned} w \rightarrow w - \eta \frac{\partial C_0}{\partial w} - \frac{\eta \lambda}{n} w \\ = \left(1 - \frac{\eta \lambda}{n}\right) w - \eta \frac{\partial C_0}{\partial w}. \end{aligned}$$

* 위 식에서 C는 Error를 의미함

- λ 값을 크게 만들어서 가중치 행렬 w를 0에 가깝게 설정할 수 있음. 그 결과 간단하고 작은 신경망이 되기에 과대적합이 덜 일어남.



- Cancernet 모델에는 부적합한 것으로 보임.

2) 드롭아웃

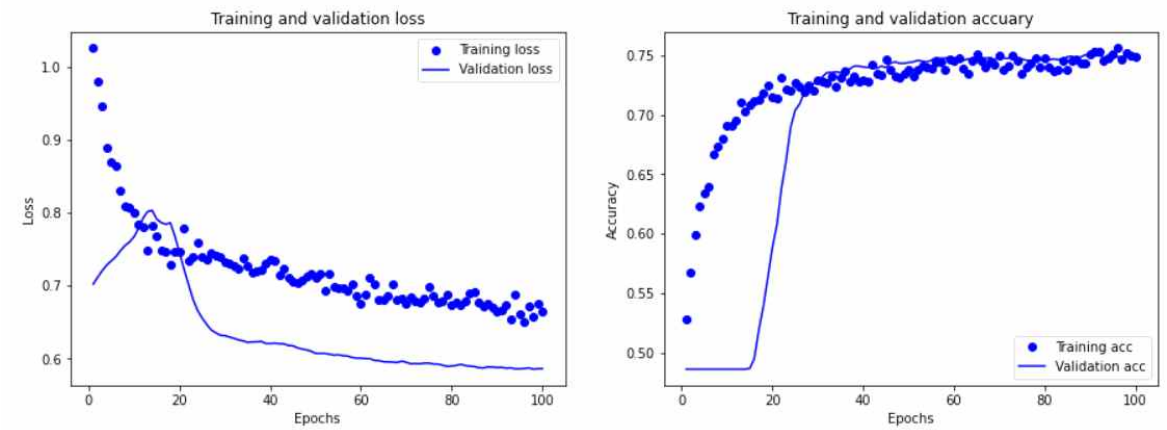
- 신경망의 각각의 층에 대해 노드를 삭제할 확률을 설정하고, 삭제할 노드를 랜덤으로 선정한 후 삭제된 노드에 들어가는 링크와 나가는 링크를 모두 삭제.
- 그 결과 더 작고 간소화된 네트워크가 만들어지고 이때 이 작아진 네트워크로 훈련을 진행하게 되므로 과대적합이 일어나는 것을 줄일 수 있음.
- 드롭아웃은 랜덤으로 노드를 삭제시키기 때문에 어떤 특성에도 의존할 수 없음. 즉 특정 입력에 특별히 큰 가중치를 부여하기가 곤란하므로, 모든 입력 각각에 가중치를 분산시키게 되고, 가중치의 노름의 제곱값이 줄어들게 됨. 즉 가중치가 작아지게 해서 과대적합 발생을 줄임.

3) L2 정규화와 드롭아웃의 차이점
- L2 정규화는 개별 가중치 각각에 적용되나 드롭아웃은 그렇지 않음.
- 컴퓨터비전에서는 드롭아웃이 대체로 더 효과적임.

결론
네트워크 정규화 기법으로 드롭아웃 채택.

4. 하이퍼파라미터 튜닝
- 미니 배치 크기와 learning rate 값을 랜덤하게 탐색하였음.
- 무작위 접근 방식의 타당성 : 튜닝 시 우선순위에 대해 관행은 존재할지라도, 특정 모델에서의 하이퍼 파라미터의 우선순위를 정확히 파악하기는 어렵고, grid search 방식의 경우 더 중요도가 높은 하이퍼파라미터의 몇 가지 값에 의해 사실상 결정되므로 탐색하는 경우의 수가 적기 때문.

Case 1)
optimizer : Adadelta
batch size = 128
learning rate = 0.003962703907659129



Case 2)

optimizer : Adagrad

batch size = 64

learning rate = 0.028213040859089513