TermProroct

2019028313 / Kim Hyewon

Index

TermProject

- Input Image: 28 x 28 x 3
- 주어진 이미지의 class를 분류하는 model을 구축하고 학습한다.
- 0 ~ 9의 숫자를 분류하는 좋은 성능의 neural network를 학습한다.

Dataset

- MNIST
- SVHN
- MNIST-C

위의 3가지 dataset을 기반으로 이외에도 다양한 폰트의 숫자 이미지를 수집 및 변형하여 학습 데이터를 구축하였습니다.

Dataset 변화

학습 초반에는 RGB의 3 channel 이미지로 학습데이터를 구성하였으나 이들을 transforms.Grayscale을 통해 1 channel로 변경하여 학습한 결과가 성능이 더 좋았으며 더 간단한 model을 통해 학습할 수 있었습니다. 때문에, 입력 받은 컬러 이미지를 흑백으로 변환하여 학습하였습니다.
(input channel: 3 -> 1)

Model

early model

- 이전 filter의 크기가 각각 5, 3인 2개의 CONV와 POOL, 3개의 FC layer로 이루어진 model
- MNIST dataset으로만 이루어지는 경우 충분한 학습이 가능하지만 svhn과 다양한 숫자 폰트로 이루어진 dataset은 Accuracy(<0.94)와 loss(> 0.08)값으로 확인한 성능이 좋지 않았습니다.
- 분석: 다양한 학습 데이터를 학습하기에 model이 간단하다(더 깊은 model이 필요하다).

final model

- layer를 하나 더 추가하여 model이 더 깊어질 수 있도록 조정해주었습니다.
- 크기가 3인 filter로 구성된 3개의 CONV를 이용하여 model의 깊이를 증가시켰습니다. (28 x 28 의 크기가 작은 image를 학습하기 때문에 VGG16, AlexNet과 같은 network처럼 깊은 신경망을 구성하지 않았습니다.)
- train data기반의 loss값에 비해 validation data의 accuracy와 loss가 낮아 overfitting을 막기위해 dropout layer를 추가하였습니다(아래 표에서는 생략하였습니다).
  (test data의 accuracy는 감소하지만 train data의 accuracy와 loss는 성능의 향상을 나타내 overfitting을 확인하였으며 이를 막기위해 추가하였습니다.)
- Gradient Vanishing을 억제하고 초기 가중치값의 영향을 줄이기위해 BatchNorm을 추하였습니다.

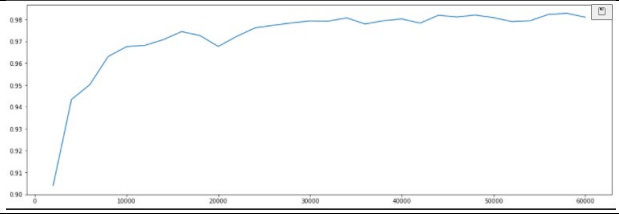
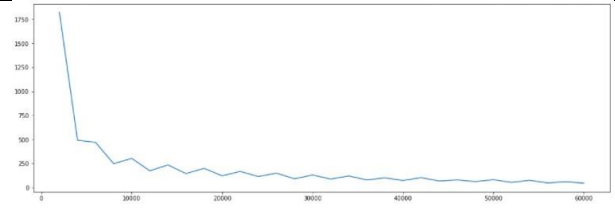
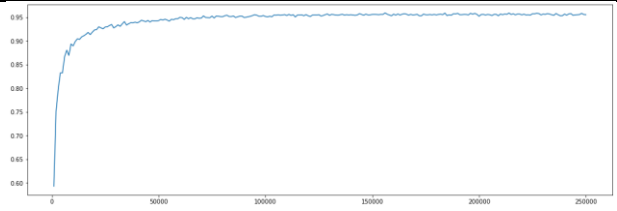
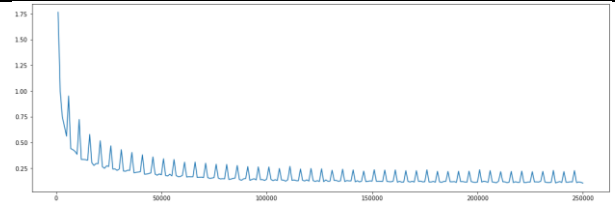
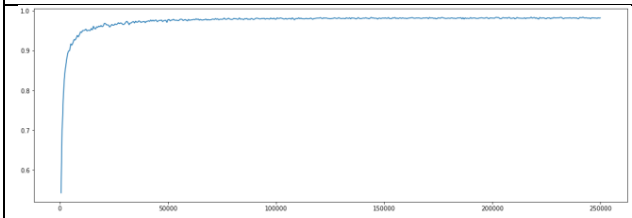
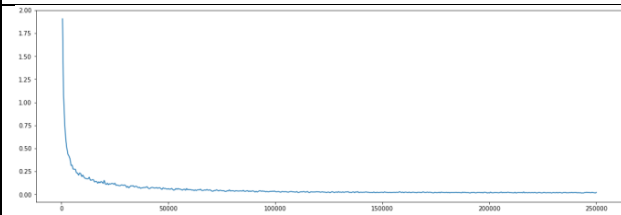
Input	RGB image	28 x 28 x 3
DataLoad	Grayscale	28 x 28 x 1
CONV1	[f = 3, s = 1, p = 1]	28 x 28 x 6
BatchNorm		
POOL	maxpooling	14 x 14 x 6
CONV2	[f = 3, s = 1, p = 0]	12 x 12 x 16
BatchNorm		
POOL	maxpooling	6 x 6 x 16
CONV3	[f = 3, s = 1, p = 0]	4 x 4 x 64
BatchNorm		
POOL	maxpooling	2 x 2 x 64
FC4		120 x 1
FC5		80 x 1
FC6		10 x 1

Train

- Loss function: CrossEntropyLoss
- Optimizer: Adam
- Learning rate: Scheduler, ExponentioalLR

  고정적인 learning rate를 사용하지 않고 scheduler를 통해 learning rate가 변하도록 하였습니다.

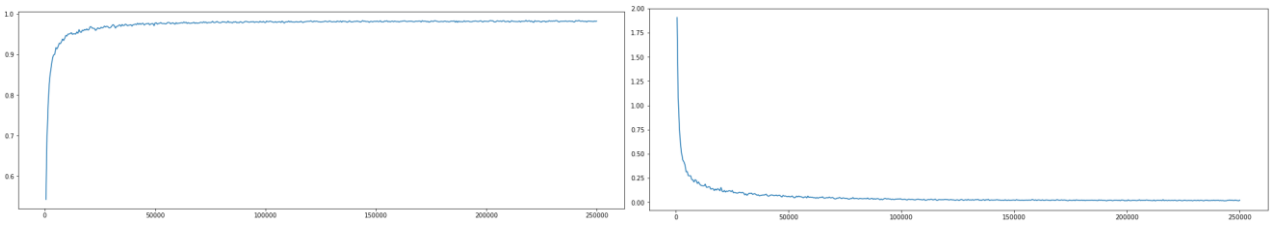
<Loss Function, Scheduler, optimizer, model 구조 등에 따른 Accuracy와 Loss값 그래프 차이>

Accuracy	Loss
	
	
	

- 대부분 loss값이 줄고, accuracy는 증가하는 양상을 보여주었지만 loss값이 크게 증가하는 구간이 있거나, 최소 loss값이 0.1 ~ 0.3으로 높은 값을 같기도 하였습니다.
- 때문에, loss값을 고정적인 값으로 설정하지 않고 scheduler를 이용해 감소할 수 있는 값으로 설정하였습니다. (LambdaLR, StepLR, ExponentialLR, CyclicLR 등 다양한 scheduler를 적용해보았으며 가정 깔끔하게 loss가 감소하는 것은 ExponentialLR이었습니다.)
- 또한, model의 깊이를 늘리고 dataset이 더 다양한 숫자 폰트를 포함하도록 조정해 주었습니다. (dataset이 증가함에 따라 과적합이 줄고, model의 깊이를 늘림에 따라 0.1이상이던 loss값이 0.02까지 감소할 수 있었습니다.)
- 학습 dataset 또한 변화를 주었습니다. train, validation dataset의 성능에 비해 test data의 정답율이 낮아 test data를 파악하여 다양한 폰트의 숫자 이미지를 추가하고 dataset를 재구축하였습니다.

Result

앞서 설명한 model, dataset 그리고 학습과정을 거쳐 아래와 같이 model을 학습시킬 수 있었습니다.



loss: 0.022 , Accuracy:0.982

Realization

loss function, batch size, learning rate, optimizer, batch normalization, model의 구조, dataset 등 다양한 요소들이 변함에 따라 학습 결과가 매우 크게 달라지며 validation과 test dataset에서도 좋은 성능 얻기 위해 고려할 사항이 많은 것을 경험한 프로젝트였습니다. 또한 dataset을 만들기 위해 data 수집, data augmentation, labeling 등 다양한 과정이 필요하며 오랜 시간이 걸린다는 것을 알게 되었습니다.