Idea Factory Intensive Program #2

딥러닝 홀로서기

#16

이론강의/PyTorch실습/코드리뷰

딥러닝(Deep Learning)에 관심이 있는 학생 발굴을 통한
딥러닝의 이론적 배경 강의 및 오픈소스 딥러닝 라이브러리 PyTorch를 활용한 실습

Acknowledgement

Sung Kim's 모두를 위한 머신러닝/딥러닝 강의
- https://hunkim.github.io/ml/
- https://www.youtube.com/playlist?list=PLlMkM4tgfjnLSOjrEJN31gZATbcj_MpUm

Andrew Ng's and other ML tutorials
- https://class.coursera.org/ml-003/lecture
- http://www.holehouse.org/mlclass/ (note)
- Deep Learning Tutorial
- Andrej Karpathy's Youtube channel

WooYeon Kim & SeongOk Ryu's KAIST CH485 Artificial Intelligence and Chemistry
- https://github.com/SeongokRyu/CH485---Artificial-Intelligence-and-Chemistry

SungJu Hwang's KAIST CS492 Deep Learning Course Material

Many insightful articles, blog posts and Youtube channels

Facebook community
- Tensorflow KR (https://www.facebook.com/groups/TensorFlowKR/)
- Pytorch KR (https://www.facebook.com/groups/PyTorchKR/)

Medium Channel and Writers
- Toward Data Science (https://towardsdatascience.com/)

How was Assignment #2?

How was Assignment #2?

1. Colab GPU 써도 왜 이렇게 느림..?
2. Regularization들 어떻게 구현해야 함..?
3. 결과 비교는 어떻게 해야 혀…?

Today`s Time Schedule

Today`s Time Schedule

Assignment #2 Review — 40 mins

Improvement in DL Optimizer — 40 mins

How to Save Experiment Results

How to Load and Manipulate Experiment Results

How to Visualize Results

1.5 hour

Today`s Time Schedule

Assignment #2 Review — 40 mins

Improvement in DL Optimizer — 40 mins

How to Save Experiment Results

How to Load and Manipulate Experiment Results

How to Visualize Results

1.5 hour

Assignment #2 Review

Assignment #2 Review

- Cifar-10 Dataset (10-way classification)
- 3 channel(RGB), 32 x 32 size (total 3072 dim)
- Use GPU
- Apply various regularization techniques

Assignment #2 Review

Model Architecture

```python
##### My Code ######

class MLP(nn.Module):
    def __init__(self,in_dim,hid_dim,out_dim,n_layer,act):
        super(MLP, self).__init__() # super는 다중상속시 슈퍼 클래스의 method를 호출하라는 의미
        self.in_dim = in_dim
        self.hid_dim = hid_dim
        self.out_dim = out_dim
        self.n_layer = n_layer
        self.act = act

        # input --> act(output)
        self.fc = nn.Linear(self.in_dim, self.hid_dim)
        # act(output) --> hid_dim을 가진 linear(fc) layer에 n-1번 통과
        self.linears = nn.ModuleList()
        # n-1번 fc layer를 통과
        for i in range(self.n_layer-1):
            self.linears.append(nn.Linear(self.hid_dim, self.hid_dim))
        # out_dim은 최종 클래스 갯수대로
        self.fc2 = nn.Linear(self.hid_dim, self.out_dim)

        if self.act == 'relu':
            self.act = nn.ReLU()

    def forward(self, x):
        x = self.act(self.fc(x))
        for fc in self.linears: ### 여기 잘 이해 안 가..이렇게만 써도 레이어를 통과하게 되는 거야?
            x = self.act(fc(x))
        x = self.fc2(x)
        return x

net = MLP(3072, 100, 10, 5, 'relu') # 3072 = 32*32(pixel)*3(RGB)
```

https://github.com/soheekim911/Standalone_DL/blob/master/assignment02.ipynb

Assignment #2 Review

Model Architecture

```
##### My Code ######

class MLP(nn.Module):
    def __init__(self,in_dim,hid_dim,out_dim,n_layer,act):
        super(MLP, self).__init__() # super는 다중상속시 슈퍼 클래스의 method를 호출하라는 의미
        self.in_dim = in_dim
        self.hid_dim = hid_dim
        self.out_dim = out_dim
        self.n_layer = n_layer
        self.act = act

        # input --> act(output)
        self.fc = nn.Linear(self.in_dim, self.hid_dim)
        # act(output) --> hid_dim을 가진 linear(fc) layer에 n-1번 통과
        self.linears = nn.ModuleList()
        # n-1번 fc layer를 통과
        for i in range(self.n_layer-1):
            self.linears.append(nn.Linear(self.hid_dim, self.hid_dim))
        # out_dim은 최종 클래스 갯수대로
        self.fc2 = nn.Linear(self.hid_dim, self.out_dim)

        if self.act == 'relu':
            self.act = nn.ReLU()

    def forward(self, x):
        x = self.act(self.fc(x))
        for fc in self.linears: ### 여기 잘 이해 안 가..이렇게만 써도 레이어를 통과하게 되는 거야?
            x = self.act(fc(x))
        x = self.fc2(x)
        return x

net = MLP(3072, 100, 10, 5, 'relu') # 3072 = 32*32(pixel)*3(RGB)
```

↘넵! 이렇게만 써도 레이어에 통과됩니다!

https://github.com/soheekim911/Standalone_DL/blob/master/assignment02.ipynb

Assignment #2 Review

```python
##### My Code ######

import argparse

np.random.seed(seed)
torch.manual_seed(seed)

parser = argparse.ArgumentParser()
args = parser.parse_args("")
print(type(args))

#### model related parameters ####
args.in_dim = 3072
args.hid_dim = 100
args.out_dim = 10
args.n_layer = 5

#### Hyperparameters ####
args.act = 'relu'
args.lr = 0.001
args.mm = 0.9
args.epoch = 3

layer_list = [3,4,5]
hid_dim_list = [50,100,150]

list_epoch = []
list_train_loss = []
list_val_loss = []
list_acc = []
list_acc_epoch = []

for layer in layer_list:
    for dim in hid_dim_list:
        args.n_layer = layer
        args.hid_dim = dim
        result = experiment(args)
        print(result)
```

```python
print(list_epoch)
print(list_train_loss)
print(list_val_loss)
print(list_acc)
print(list_acc_epoch)
```

```
[0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]
[718.4838404655457, 702.5457305908203, 663.583244562149, 716.0431571006775, 686.204131603241, 643.3055664300919, 71
5.5012757778168, 685.0886516571045, 637.2816002368927, 721.2688143253326, 718.9383387565613, 715.5292422771454, 719.
786628484726, 715.6561961174011, 701.5076425075531, 719.8854279518127, 716.2080438137054, 704.1175994873047, 721.838
5496139526, 720.5331976413727, 719.5170781612396, 721.1488587856293, 720.5819962024689, 720.1053557395935, 721.11438
65585327, 720.4742543697357, 719.8597626686096]
[2.2781264268899264, 2.196911377242849, 2.0455130821541894, 2.259309524222265, 2.1231741452518897, 2.009038704860059
5, 2.2590633585483215, 2.111594918407971, 1.9870776149291027, 2.3008092324944993, 2.292842569230478, 2.2772972281975
083, 2.295247252983383, 2.27392411835586, 2.197103759910487, 2.2952077388763428, 2.277197822739806, 2.21005919009824
36, 2.3044583163683927, 2.3009407731551157, 2.2972732978531076, 2.3032811653764944, 2.3017285594457313, 2.3001226594
17647, 2.303093463559694, 2.3012663714493375, 2.299001008649415 5]
[26.76, 29.48, 29.67, 15.45, 18.54, 20.04, 13.5, 13.42, 17.75]
[2, 2, 2, 2, 2, 2, 2, 2, 2]
```

Epoch에 따른 실험 결과를 저장하는 리스트가 바깥 쪽에서 생성됨
→ 각 실험의 결과들이 계속 쌓임 ㅠ

https://github.com/soheekim911/Standalone_DL/blob/master/assignment02.ipynb

Assignment #2 Review

```python
##### My Code ######

import argparse

np.random.seed(seed)
torch.manual_seed(seed)

parser = argparse.ArgumentParser()
args = parser.parse_args("")
print(type(args))

#### model related parameters ####
args.in_dim = 3072
args.hid_dim = 100
args.out_dim = 10
args.n_layer = 5

#### Hyperparameters ####
args.act = 'relu'
args.lr = 0.001
args.mm = 0.9
args.epoch = 3

layer_list = [3,4,5]
hid_dim_list = [50,100,150]

list_epoch = []
list_train_loss = []
list_val_loss = []
list_acc = []
list_acc_epoch = []

for layer in layer_list:
    for dim in hid_dim_list:
        args.n_layer = layer
        args.hid_dim = dim
        result = experiment(args)
        print(result)
```

```python
print(list_epoch)
print(list_train_loss)
print(list_val_loss)
print(list_acc)
print(list_acc_epoch)
```

```
[0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]
[718.4838404655057, 702.5457305908203, 663.583244562149, 716.0431571006775, 686.204131603241, 643.3055664300919, 715.5012757778168, 685.0886516571045, 637.2816002368927, 721.2688143253326, 718.9383387565613, 715.5292422771454, 719.786628484726, 715.6561961174011, 701.5076425075531, 719.8854279518127, 716.2080438137054, 704.1175994873047, 721.8385496139526, 720.5331976413727, 719.5170781612396, 721.1488587856293, 720.5819962024689, 720.1053557395935, 721.1143865585327, 720.4742543697357, 719.8597626686096]
[2.2781264268899264, 2.196911377242849, 2.0455130821541894, 2.259309524222265, 2.1231741452518897, 2.0090387048600595, 2.2590633585483215, 2.111594918407971, 1.9870776149291027, 2.3008092324944993, 2.292842569230478, 2.2772972281975083, 2.295247252983383, 2.273924111835586, 2.197103759910487, 2.2952077388763428, 2.277197822739806, 2.2100591900982436, 2.3044583163683927, 2.3009407731551157, 2.2972732978531076, 2.3032811653764944, 2.3017285594457313, 2.3001226594 17647, 2.303093463559694, 2.3012663714493375, 2.2990010086494155]
[26.76, 29.48, 29.67, 15.45, 18.54, 20.04, 13.5, 13.42, 17.75]
[2, 2, 2, 2, 2, 2, 2, 2, 2]
```

Epoch이 반복되는 모습.
List_acc에도 여러 실험 결과가 순서대로 다 저장되어 있음

Epoch에 따른 실험 결과를 저장하는 리스트가 바깥 쪽에서 생성됨
→ 각 실험의 결과들이 계속 쌓임 ㅠ

https://github.com/soheekim911/Standalone_DL/blob/master/assignment02.ipynb

Assignment #2 Review

```python
##### My Code ######

import argparse

np.random.seed(seed)
torch.manual_seed(seed)

parser = argparse.ArgumentParser()
args = parser.parse_args("")
print(type(args))

#### model related parameters ####
args.in_dim = 3072
args.hid_dim = 100
args.out_dim = 10
args.n_layer = 5

#### Hyperparameters ####
args.act = 'relu'
args.lr = 0.001
args.mm = 0.9
args.epoch = 3

layer_list = [3,4,5]
hid_dim_list = [50,100,150]

list_epoch = []
list_train_loss = []
list_val_loss = []
list_acc = []
list_acc_epoch = []

for layer in layer_list:
    for dim in hid_dim_list:
        args.n_layer = layer
        args.hid_dim = dim
        result = experiment(args)
        print(result)
```

리스트들이 각 실험 마다 생성되고
하나의 리스트는 하나의 실험 결과들만 저장해야 함

https://github.com/soheekim911/Standalone_DL/blob/master/assignment02.ipynb

Assignment #2 Review

Model Architecture

```python
import torch.nn as nn
import torch.nn.functional as F


class MLP(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim, n_layer, act, dropout, init_xavier, batch_norm):
        super(MLP, self).__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.hid_dim = hid_dim
        self.n_layer = n_layer
        self.act = act
        self.dropout = dropout
        self.init_xavier = init_xavier
        self.batch_norm = batch_norm

        self.fc = nn.Linear(self.in_dim, self.hid_dim)
        self.linears = nn.ModuleList()

        for i in range(self.n_layer-1):
            self.linears.append(nn.Linear(self.hid_dim, self.hid_dim))
        self.fc2 = nn.Linear(self.hid_dim, self.out_dim)


        if self.act == 'relu':
            self.act = nn.ReLU()

        self.batch = nn.BatchNorm1d(self.hid_dim)
        self.drop = nn.Dropout()

    def forward(self, x):
        x = self.act(self.fc(x))
        for fc in self.linears:
            x = self.act(fc(x))
            if self.batch_norm == True: x = self.batch(x)     # Batch Normalization
        x = self.fc2(x)          # 마지막 layer는 activation function 먹이지 않기!
        if self.dropout == True: x = self.drop(x)     # Dropout
        if self.init_xavier == True: x = nn.init.xavier_uniform_(x)     # Xavier Initializtion

        return x
```

1. 하나의 BatchNorm 레이어가 여러 번 재사용됨
   → 한 BatchNorm은 하나의 Linear 레이어만 담당!

https://github.com/kwonsoonjae/happyhappyDL/blob/master/Lec3_Assignment_CIFAR10.ipynb

Assignment #2 Review

Model Architecture

```python
import torch.nn as nn
import torch.nn.functional as F


class MLP(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim, n_layer, act, dropout, init_xavier, batch_norm):
        super(MLP, self).__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.hid_dim = hid_dim
        self.n_layer = n_layer
        self.act = act
        self.dropout = dropout
        self.init_xavier = init_xavier
        self.batch_norm = batch_norm

        self.fc = nn.Linear(self.in_dim, self.hid_dim)
        self.linears = nn.ModuleList()

        for i in range(self.n_layer-1):
            self.linears.append(nn.Linear(self.hid_dim, self.hid_dim))
        self.fc2 = nn.Linear(self.hid_dim, self.out_dim)


        if self.act == 'relu':
            self.act = nn.ReLU()

        self.batch = nn.BatchNorm1d(self.hid_dim)
        self.drop = nn.Dropout()

    def forward(self, x):
        x = self.act(self.fc(x))
        for fc in self.linears:
            x = self.act(fc(x))
            if self.batch_norm == True: x = self.batch(x)     # Batch Normalization
        x = self.fc2(x)          # 마지막 layer는 activation function 먹이지 않기!
        if self.dropout == True: x = self.drop(x)     # Dropout
        if self.init_xavier == True: x = nn.init.xavier_uniform_(x)     # Xavier Initializtion

        return x
```

1. 하나의 BatchNorm 레이어가 여러 번 재사용됨
   → 한 BatchNorm은 하나의 Linear 레이어만 담당!

2. Dropout이 마지막 레이어에만 있음
   → Dropout은 중간에 있어야
   비로소 오버피팅을 막을 수 있음!

https://github.com/kwonsoonjae/happyhappyDL/blob/master/Lec3_Assignment_CIFAR10.ipynb

Assignment #2 Review

Model Architecture

```python
import torch.nn as nn
import torch.nn.functional as F


class MLP(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim, n_layer, act, dropout, init_xavier, batch_norm):
        super(MLP, self).__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.hid_dim = hid_dim
        self.n_layer = n_layer
        self.act = act
        self.dropout = dropout
        self.init_xavier = init_xavier
        self.batch_norm = batch_norm

        self.fc = nn.Linear(self.in_dim, self.hid_dim)
        self.linears = nn.ModuleList()

        for i in range(self.n_layer-1):
            self.linears.append(nn.Linear(self.hid_dim, self.hid_dim))
        self.fc2 = nn.Linear(self.hid_dim, self.out_dim)


        if self.act == 'relu':
            self.act = nn.ReLU()

        self.batch = nn.BatchNorm1d(self.hid_dim)
        self.drop = nn.Dropout()

    def forward(self, x):
        x = self.act(self.fc(x))
        for fc in self.linears:
            x = self.act(fc(x))
            if self.batch_norm == True: x = self.batch(x)      # Batch Normalization
        x = self.fc2(x)          # 마지막 layer는 activation function 먹이지 않기!
        if self.dropout == True: x = self.drop(x)      # Dropout
        if self.init_xavier == True: x = nn.init.xavier_uniform_(x)      # Xavier Initializtion

        return x
```

1. 하나의 BatchNorm 레이어가 여러 번 재사용됨
   → 한 BatchNorm은 하나의 Linear 레이어만 담당!

2. Dropout이 마지막 레이어에만 있음
   → Dropout은 중간에 있어야
   비로소 오버피팅을 막을 수 있음!

3. Xavier 초기화는 맨 처음 레이어를 생성할 때!
   __init__ 안에서 이루어져야 함

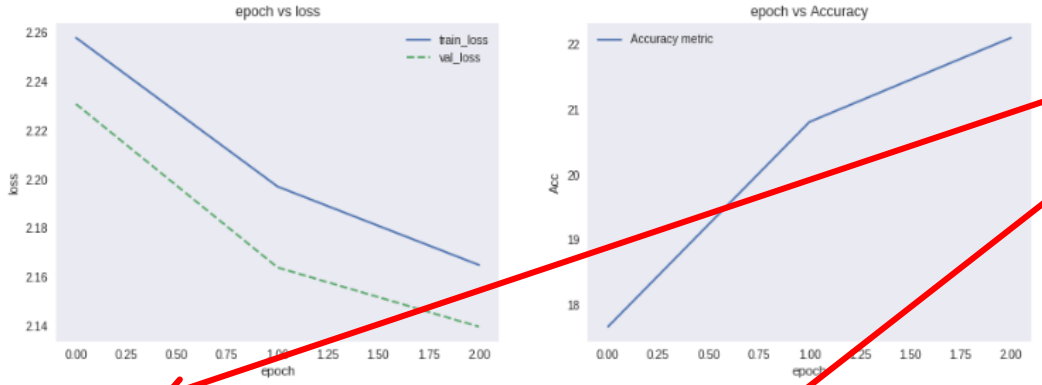
https://github.com/kwonsoonjae/happyhappyDL/blob/master/Lec3_Assignment_CIFAR10.ipynb

Assignment #2 Review

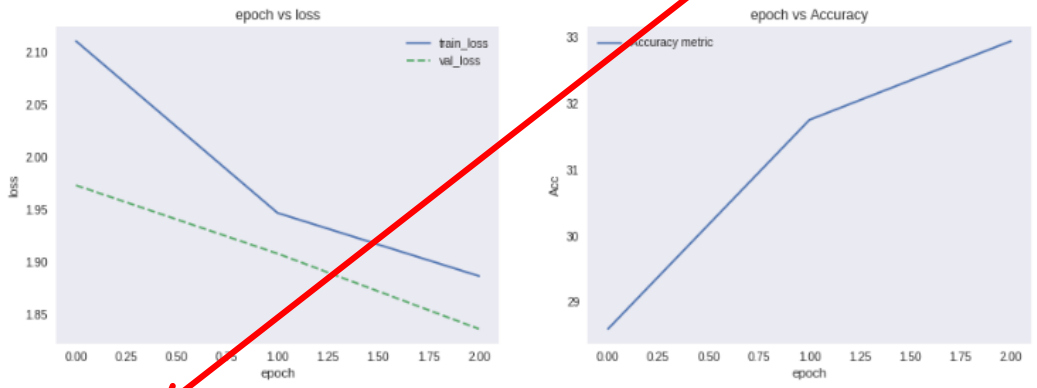
```
list_change_l2 = [True, False]
list_change_dropout = [True, False]
list_change_init_xavier = [True, False]
list_change_batch_norm = [True, False]

for var1 in list_change_batch_norm:
        args.batch_norm = var1
        result = experiment(args)
        draw_result(result[0], result[1], result[2], result[3], result[4])
```

```
Epoch 0, Train Loss: 2.26, Val Loss: 2.23, Val Acc: 17.67
Epoch 1, Train Loss: 2.20, Val Loss: 2.16, Val Acc: 20.81
Epoch 2, Train Loss: 2.16, Val Loss: 2.14, Val Acc: 22.10
```




```
Test Acc: 21.98
Epoch 0, Train Loss: 2.11, Val Loss: 1.97, Val Acc: 28.59
Epoch 1, Train Loss: 1.95, Val Loss: 1.91, Val Acc: 31.75
Epoch 2, Train Loss: 1.89, Val Loss: 1.84, Val Acc: 32.94
```




```
Test Acc: 31.85
```

Batch Normalization 을 안 썼을 때

오히려 더 성능이 향상되는 결과

→ 하나의 Batch Normalization을 여러 번 썼기 때문에
→ 레이어 별로 다른 아웃풋들에 의해 혼란스러움

https://github.com/kwonsoonjae/happyhappyDL/blob/master/Lec3_Assignment_CIFAR10.ipynb

Assignment #2 Review

질문있습니다~

사실.. 과제 코드 제대로 못해봤는데요

올려주신 시작코드 training이 너무 오래걸려서 ㅜㅜ기다리다가 끝까지 못해봤어요

일단 시작코드부터 돌려보고 hyperparameter값들 변경해가면서 accuracy 올리고 그래프도 그려보고싶었는데..

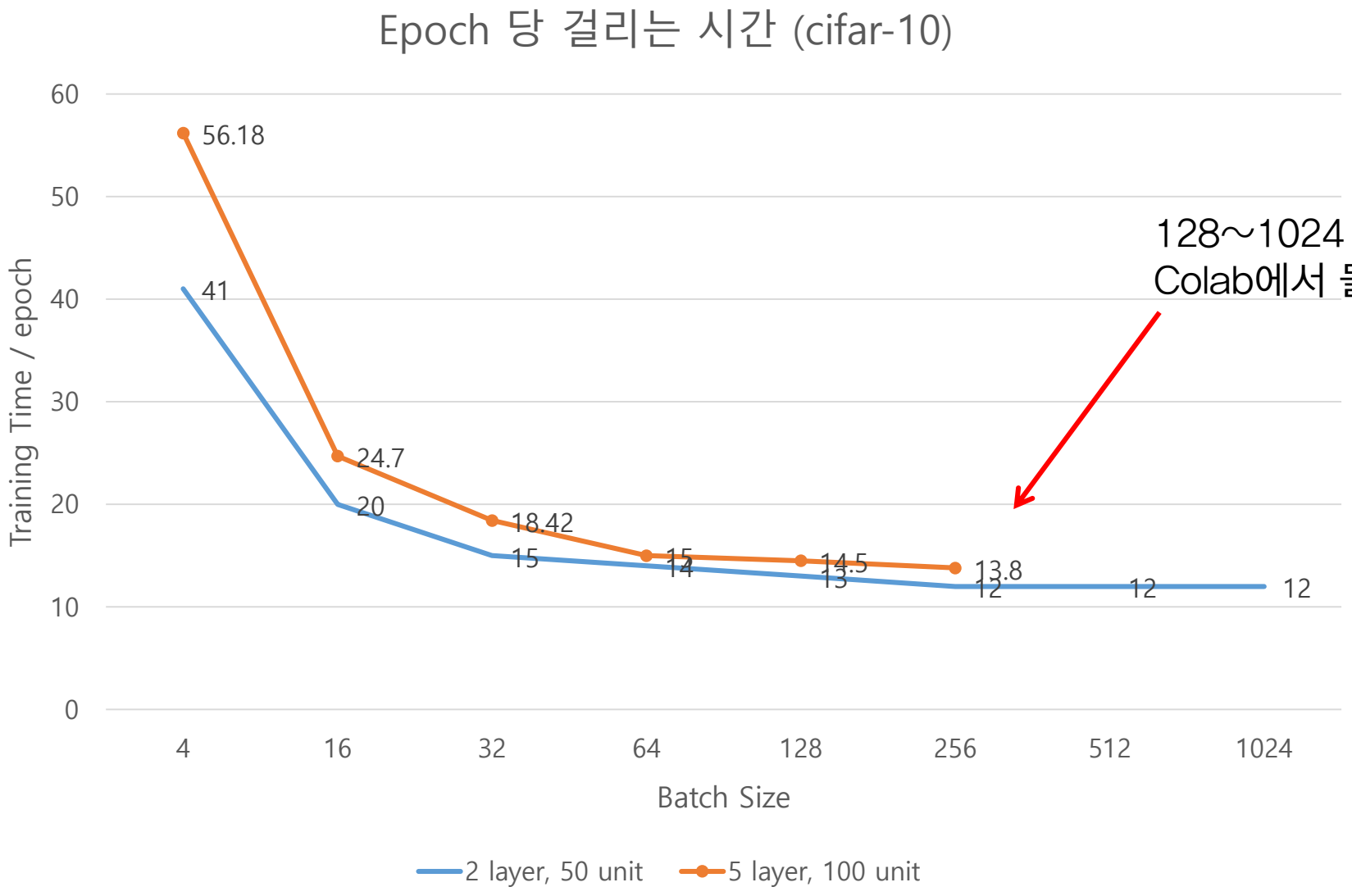
너무 느려서? GPU로 제대로 돌고있나?라는 생각이 들었습니당..

내일봐요~ ㅜㅜ

Batch Size를 너무 작게 설정해서..

https://github.com/Meereh/DeepLearning_practice/blob/master/Standalone_deeplearning_Assignment2.ipynb

Assignment #2 Review



128~1024 사이의 Batch Size로
Colab에서 돌려줍시다.. ㅎ

Assignment #2 Review

Experiment

```python
# ==== Grid Test ==== #
seed = 123
np.random.seed(seed)
torch.manual_seed(seed)

parser = argparse.ArgumentParser()
args = parser.parse_args("")


args.n_layer = 5
args.in_dim = 3072
args.out_dim = 10
args.hid_dim = 100
args.act = 'relu'

args.lr = 0.001
args.mm = 0.9
args.epoch = 3


list_n_layer = [3, 4, 5, 6, 7]
list_hid_dim = [100, 200, 400, 800]
list_lr = [.1, .01, .001, .0001]

results = {"n_layer":[], "hid_dim":[], "lr":[], "train_loss":[], "val_loss":[], "val_acc":[], "test_acc":[]}


for var1 in list_n_layer:
    for var2 in list_hid_dim:
        for var3 in list_lr:
            args.n_layer = var1
            args.hid_dim = var2
            args.lr = var3
            result = experiment(args)
            print(args.n_layer, args.hid_dim, args.lr, result[0:2])
            results["n_layer"].append(args.n_layer)
            results["hid_dim"].append(args.hid_dim)
            results["lr"].append(args.lr)
            results["train_loss"].append(result[0])
            results["val_loss"].append(result[1])
            results["val_acc"].append(result[2])
            results["test_acc"].append(result[3])
```

Val Acc: 49.7

```python
# ==== Random Test ==== #
args.n_layer = 5
args.in_dim = 3072
args.out_dim = 10
args.hid_dim = 100
args.act = 'relu'

args.lr = 0.001
args.mm = 0.9
args.epoch = 5

for _ in range(10):
    var1 = np.random.randint(2, 10)
    var2 = 2 ** np.random.randint(3, 10)
    var3 = .1 ** np.random.randint(1, 5)
    args.n_layer = var1
    args.hid_dim = var2
    args.lr = var3
    result = experiment(args)
    print(var1, var2, var3, result)
    list_result.append((var1, var2, var3, result))
    results["n_layer"].append(args.n_layer)
    results["hid_dim"].append(args.hid_dim)
    results["lr"].append(args.lr)
    results["train_loss"].append(result[0])
    results["val_loss"].append(result[1])
    results["val_acc"].append(result[2])
    results["test_acc"].append(result[3])
```

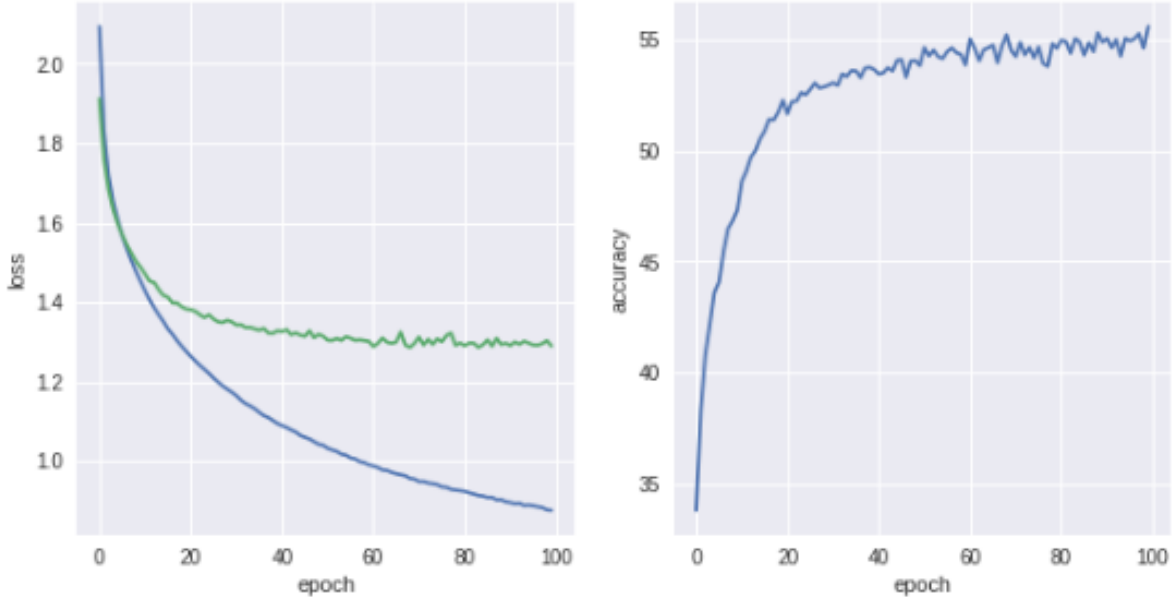
Val Acc: 50.85

Regularization Technique들을 적용해보았으면 더 좋았을 듯!

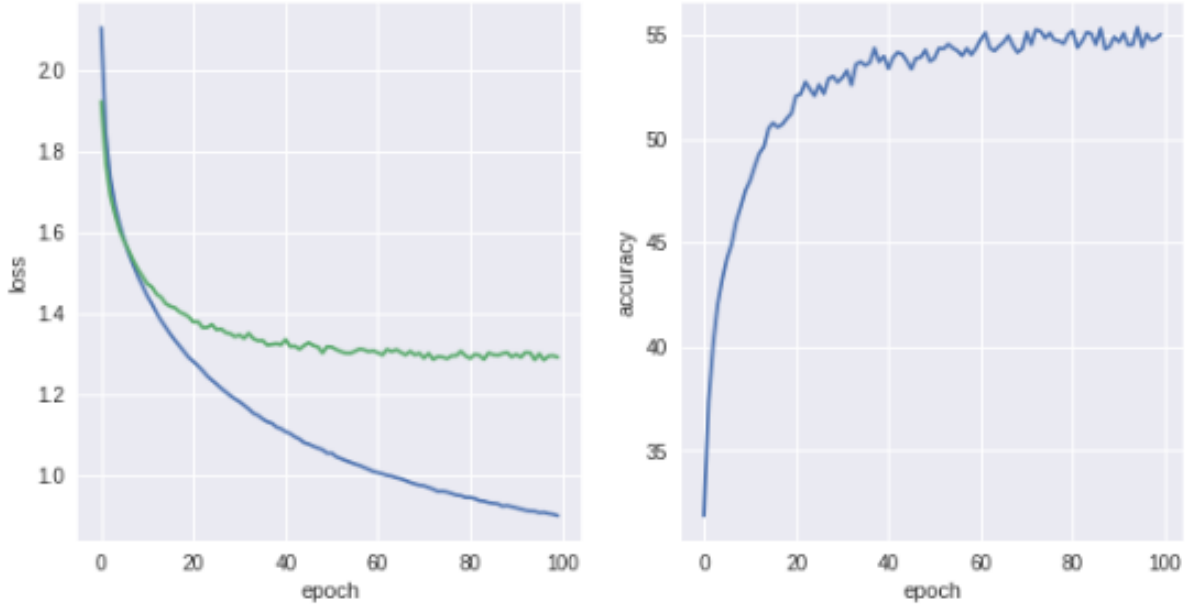
https://github.com/mekty2012/DeepLearning-StandAlone/blob/master/Assignment2.ipynb

Assignment #2 Review

< n_layer : 2 , hid_dim : 500 , dropoutRate : 0.1, L2 alpha : 0.01 > test_acc : 55.32%



< n_layer : 2 , hid_dim : 500 , dropoutRate : 0.2, L2 alpha : 0.01 > test_acc : 54.93%



Overfitting을 줄이면 좋을듯!

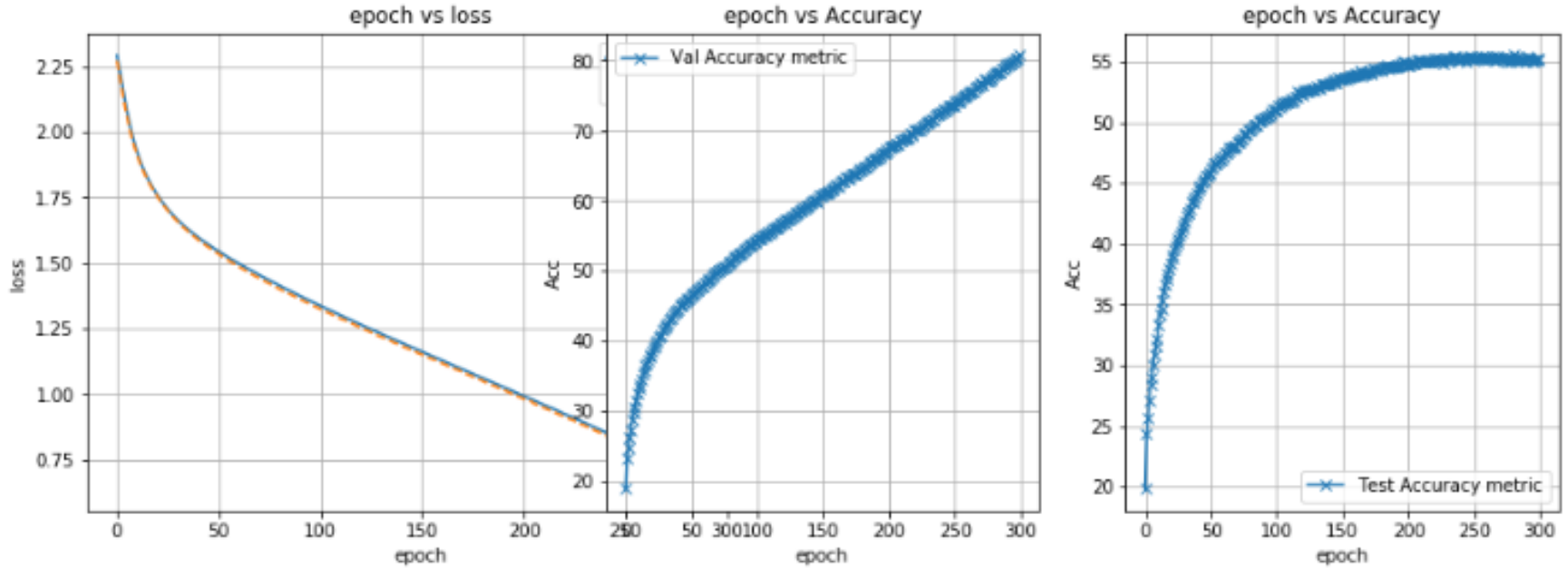
https://github.com/Hemistone/DeepLearning_StandAlone/blob/master/Assignment2_hemistone.ipynb

Assignment #2 Review

Final Result

앞선 결과는 epoch을 100으로 했을 때의 결과이다. 위 결과를 바탕으로 lr: 0.001, hid: [1000, 1000], drop:0.5 가 가장 효율이 좋다고 판단하여 이 모델에 대해 epoch을 300까지 돌린 결과는 아래와 같다.

lr: 0.001, hid: [1000, 1000], drop:0.5, Test Acc: 55.12%



100 epoch으로 적절한 후보군을 찾은 후
300 epoch까지 학습시켜 좋은 결과!

https://github.com/kboseong/DL_KAIST/blob/master/HW2/Assignment2.ipynb

Assignment #2 Review

Model Architecture

```python
class MLP(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim, n_layer, act, dropout, use_bn, use_xavier):
        super(MLP, self).__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.hid_dim = hid_dim
        self.n_layer = n_layer
        self.act = act
        self.dropout = dropout
        self.use_bn = use_bn
        self.use_xavier = use_xavier

        # ==== Create Linear Layers ==== #
        self.fc1 = nn.Linear(self.in_dim, self.hid_dim)

        self.linears = nn.ModuleList()
        self.bns = nn.ModuleList()
        for i in range(self.n_layer-1):
            self.linears.append(nn.Linear(self.hid_dim, self.hid_dim))
            if self.use_bn:
                self.bns.append(nn.BatchNorm1d(self.hid_dim))

        self.fc2 = nn.Linear(self.hid_dim, self.out_dim)

        # ==== Create Activation Function ==== #
        if self.act == 'relu':
            self.act = nn.ReLU()
        elif self.act == 'tanh':
            self.act = nn.Tanh()
        elif self.act == 'sigmoid':
            self.act = nn.Sigmoid()
        else:
            raise ValueError('no valid activation function selected!')

        # ==== Create Regularization Layer ==== #
        self.dropout = nn.Dropout(self.dropout)
        if self.use_xavier:
            self.xavier_init()
```

```python
    def forward(self, x):
        x = self.act(self.fc1(x))
        for i in range(len(self.linears)):
            x = self.act(self.linears[i](x))
            x = self.bns[i](x)
            x = self.dropout(x)
        x = self.fc2(x)
        return x

    def xavier_init(self):
        for linear in self.linears:
            nn.init.xavier_normal_(linear.weight)
            linear.bias.data.fill_(0.01)

net = MLP(3072, 10, 100, 4, 'relu', 0.1, True, True) # Testing Model Construction
```

https://github.com/Hemistone/DeepLearning_StandAlone/blob/master/Assignment2_hemistone.ipynb

Assignment #2 Review

```
if args.optim == 'SGD':
    optimizer = optim.RMSprop(net.parameters(), lr=args.lr, weight_decay=args.l2)
elif args.optim == 'RMSprop':
    optimizer = optim.RMSprop(net.parameters(), lr=args.lr, weight_decay=args.l2)
elif args.optim == 'Adam':
    optimizer = optim.Adam(net.parameters(), lr=args.lr, weight_decay=args.l2)
else:
    raise ValueError('In-valid optimizer choice')
```

https://github.com/Hemistone/DeepLearning_StandAlone/blob/master/Assignment2_hemistone.ipynb

Summary

- Again, Nice work everyone!
- Dropout Layer should be in the middle of the Neural Net
- BatchNorm Layer should be created for each linear layer
- Increase Batch Size around 128~512
- Xavier initialization should be done in __init__ method
- Linear → activation → BatchNorm → Dropout Order for MLP