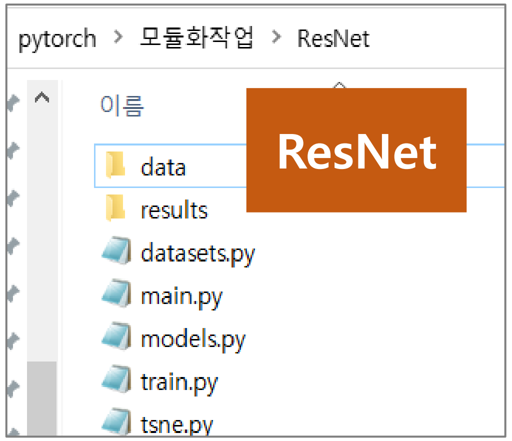
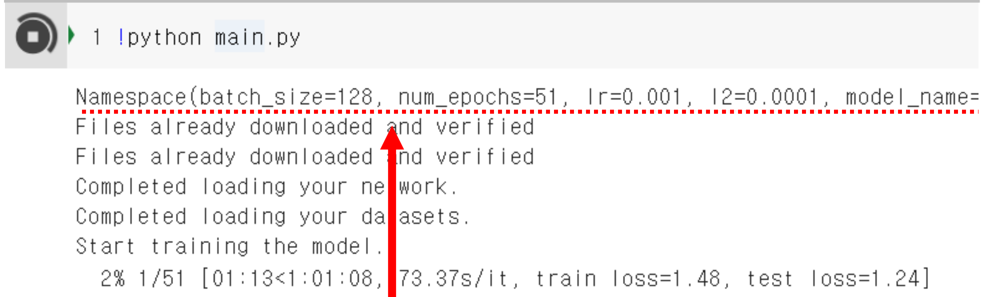


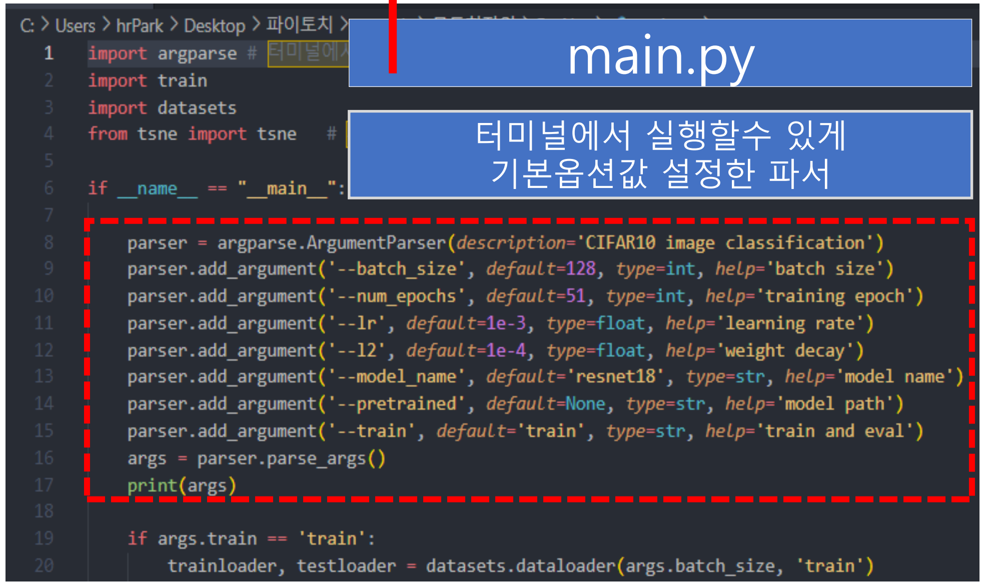
각 파일별로 기능이 별도로 작성되어 있고 main.py에 터미널에서 작업할수 있도록 파서가 설정되어 있음.

vs코드를 실행하고



```
1 !python main.py
```

```
Namespace(batch_size=128, num_epochs=51, lr=0.001, l2=0.0001, model_name=
Files already downloaded and verified
Files already downloaded and verified
Completed loading your network.
Completed loading your datasets.
Start training the model.
  2% 1/51 [01:13<1:01:08, 73.37s/it, train loss=1.48, test loss=1.24]
```

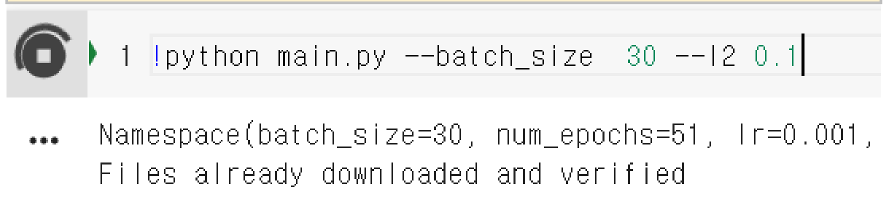


main.py

터미널에서 실행할수 있게
기본옵션값 설정한 파서

```
import argparse # 터미널에서
import train
import datasets
from tsne import tsne    #

if __name__ == "__main__":

    parser = argparse.ArgumentParser(description='CIFAR10 image classification')
    parser.add_argument('--batch_size', default=128, type=int, help='batch size')
    parser.add_argument('--num_epochs', default=51, type=int, help='training epoch')
    parser.add_argument('--lr', default=1e-3, type=float, help='learning rate')
    parser.add_argument('--l2', default=1e-4, type=float, help='weight decay')
    parser.add_argument('--model_name', default='resnet18', type=str, help='model name')
    parser.add_argument('--pretrained', default=None, type=str, help='model path')
    parser.add_argument('--train', default='train', type=str, help='train and eval')
    args = parser.parse_args()
    print(args)

    if args.train == 'train':
        trainloader, testloader = datasets.dataloader(args.batch_size, 'train')
```

옵션값 변경할때 --옵션명 옵션



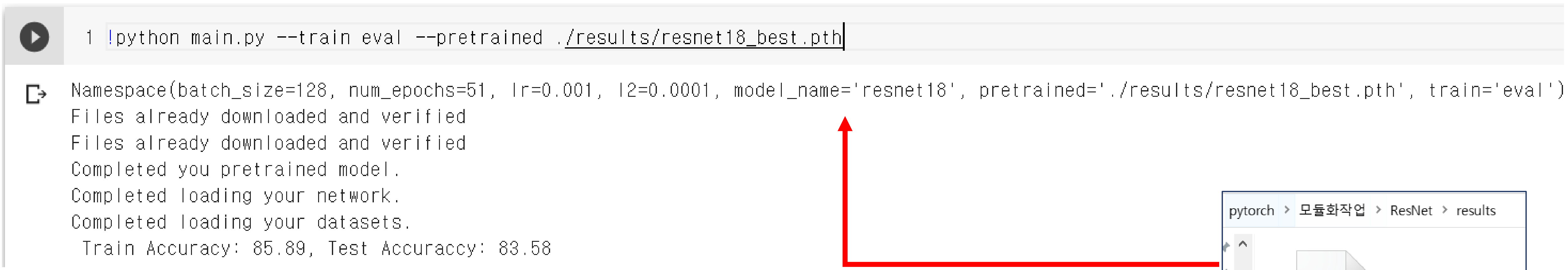
```
1 !python main.py --batch_size 30 --l2 0.1
```

```
Namespace(batch_size=30, num_epochs=51, lr=0.001,
Files already downloaded and verified
```

resnet layer를 구현한 모델임

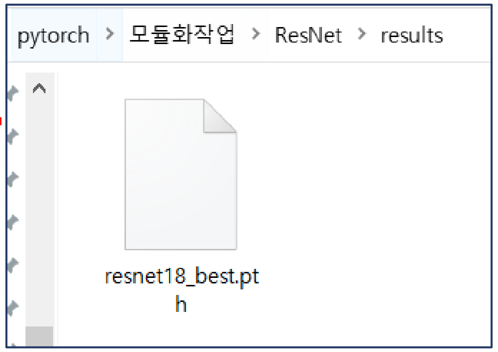
(참고) tps://velog.io/@yooniverseis/YOLO-v5-%EC%8B%A4%EC%8A%B5%ED%95%B4%EB%B3%B4%EA%B8%B0%B0

작업되어 있는 모델이 있을시 선택하여 w값 변경하지 않고 모델에 적용만 진행(전이학습과 같음).

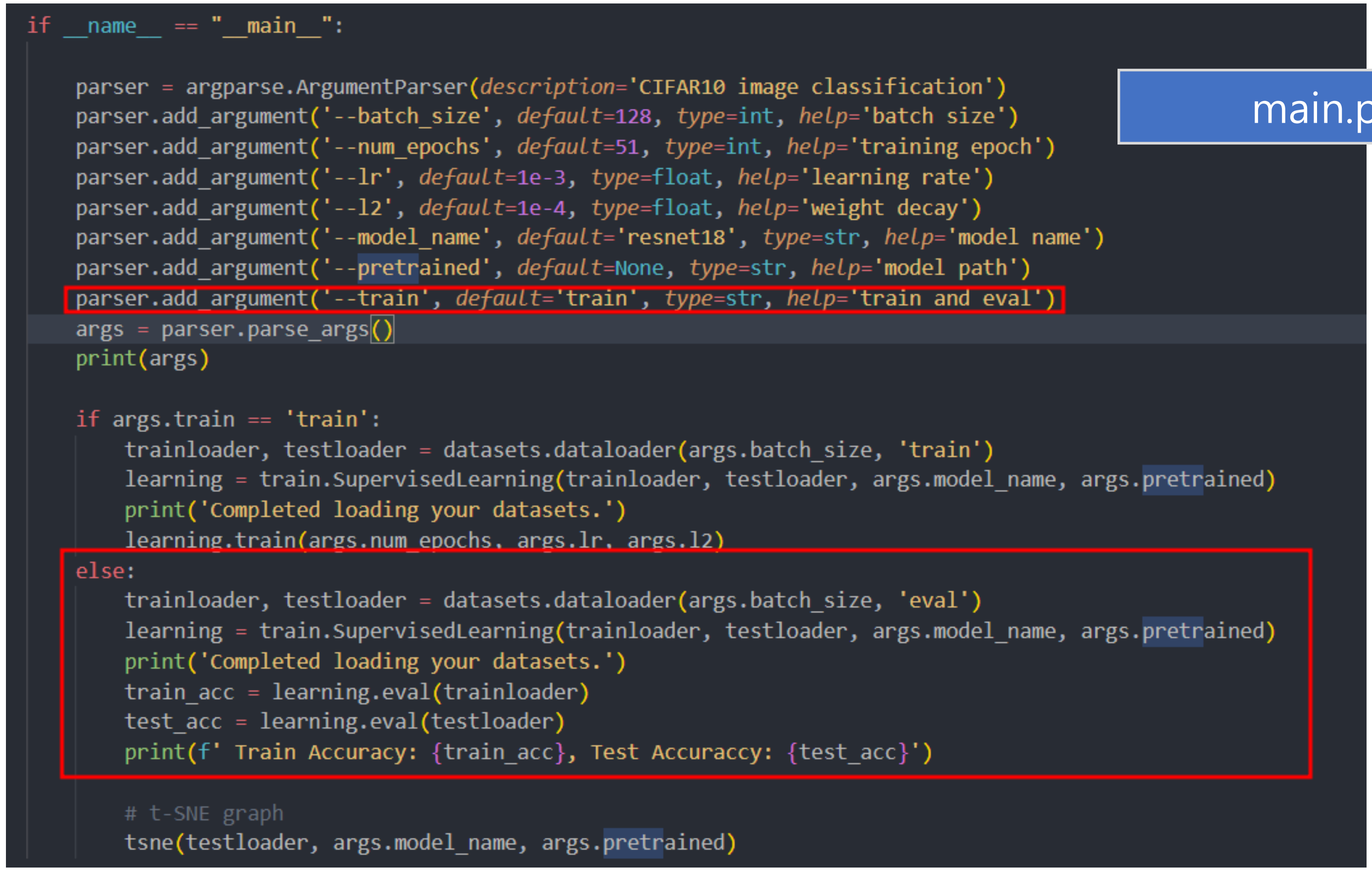


```
!python main.py --train eval --pretrained ./results/resnet18_best.pth
```

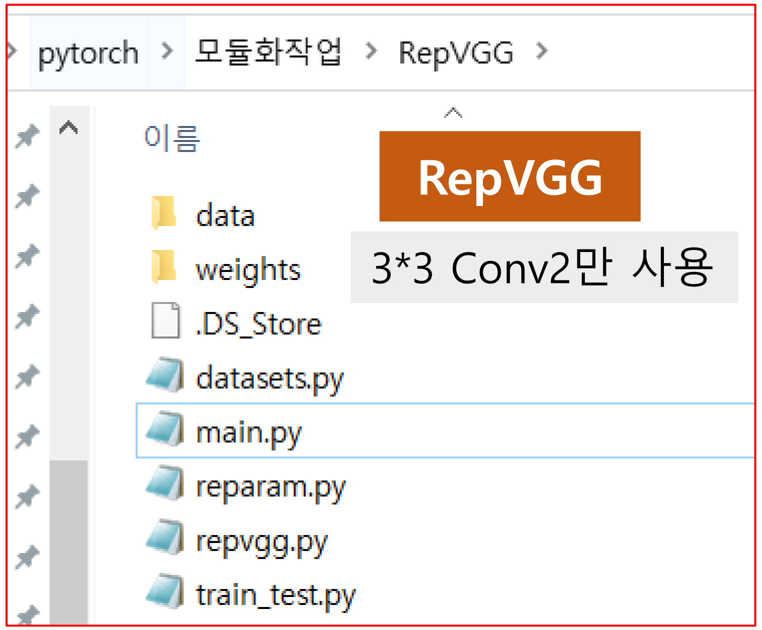
```
Namespace(batch_size=128, num_epochs=51, lr=0.001, l2=0.0001, model_name='resnet18', pretrained='./results/resnet18_best.pth', train='eval')
Files already downloaded and verified
Files already downloaded and verified
Completed you pretrained model.
Completed loading your network.
Completed loading your datasets.
 Train Accuracy: 85.89, Test Accuraccy: 83.58
```



main.py

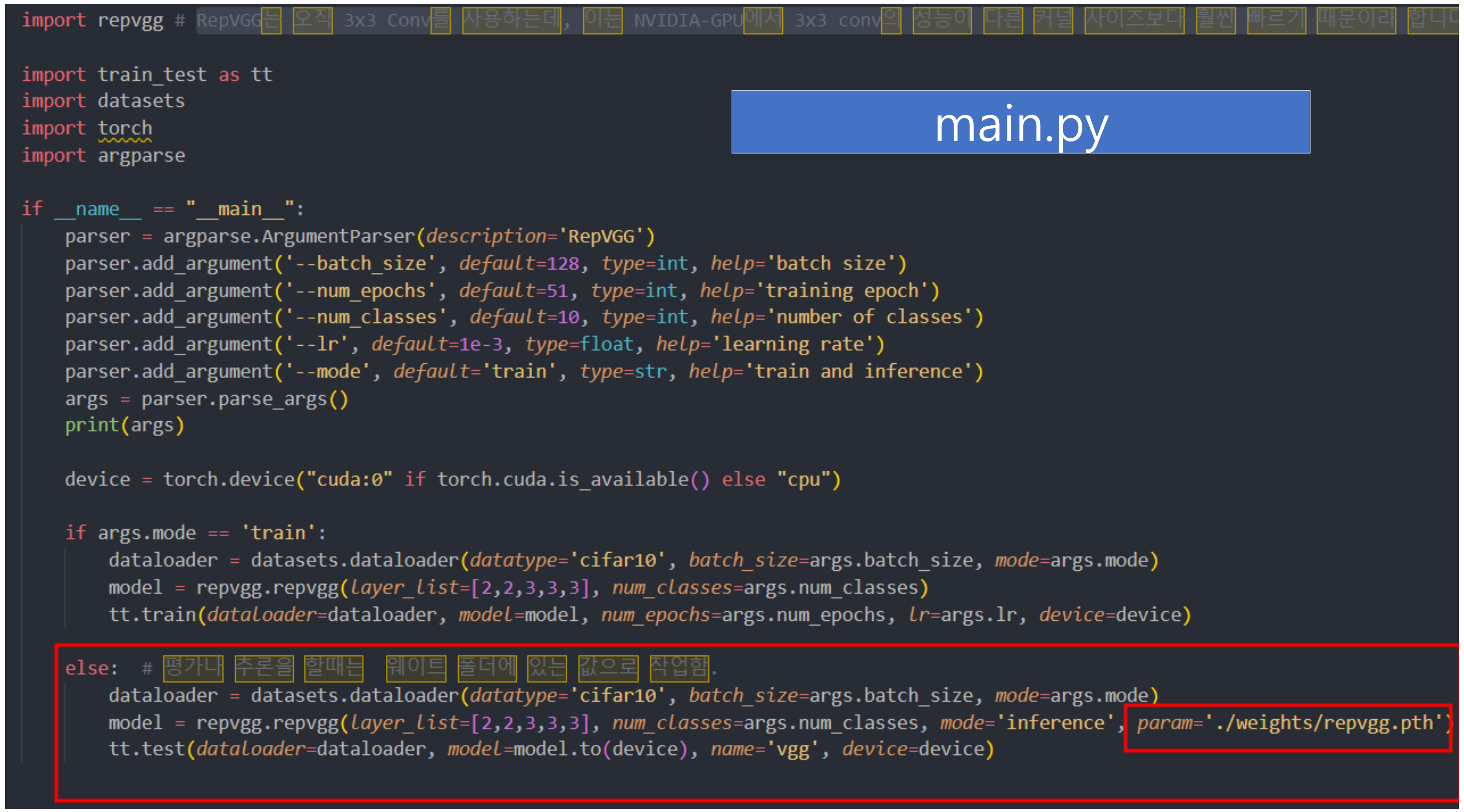


```python
if __name__ == "__main__":

    parser = argparse.ArgumentParser(description='CIFAR10 image classification')
    parser.add_argument('--batch_size', default=128, type=int, help='batch size')
    parser.add_argument('--num_epochs', default=51, type=int, help='training epoch')
    parser.add_argument('--lr', default=1e-3, type=float, help='learning rate')
    parser.add_argument('--l2', default=1e-4, type=float, help='weight decay')
    parser.add_argument('--model_name', default='resnet18', type=str, help='model name')
    parser.add_argument('--pretrained', default=None, type=str, help='model path')
    parser.add_argument('--train', default='train', type=str, help='train and eval')
    args = parser.parse_args()
    print(args)

    if args.train == 'train':
        trainloader, testloader = datasets.dataloader(args.batch_size, 'train')
        learning = train.SupervisedLearning(trainloader, testloader, args.model_name, args.pretrained)
        print('Completed loading your datasets.')
        learning.train(args.num_epochs, args.lr, args.l2)
    else:
        trainloader, testloader = datasets.dataloader(args.batch_size, 'eval')
        learning = train.SupervisedLearning(trainloader, testloader, args.model_name, args.pretrained)
        print('Completed loading your datasets.')
        train_acc = learning.eval(trainloader)
        test_acc = learning.eval(testloader)
        print(f' Train Accuracy: {train_acc}, Test Accuraccy: {test_acc}')

        # t-SNE graph
        tsne(testloader, args.model_name, args.pretrained)
```



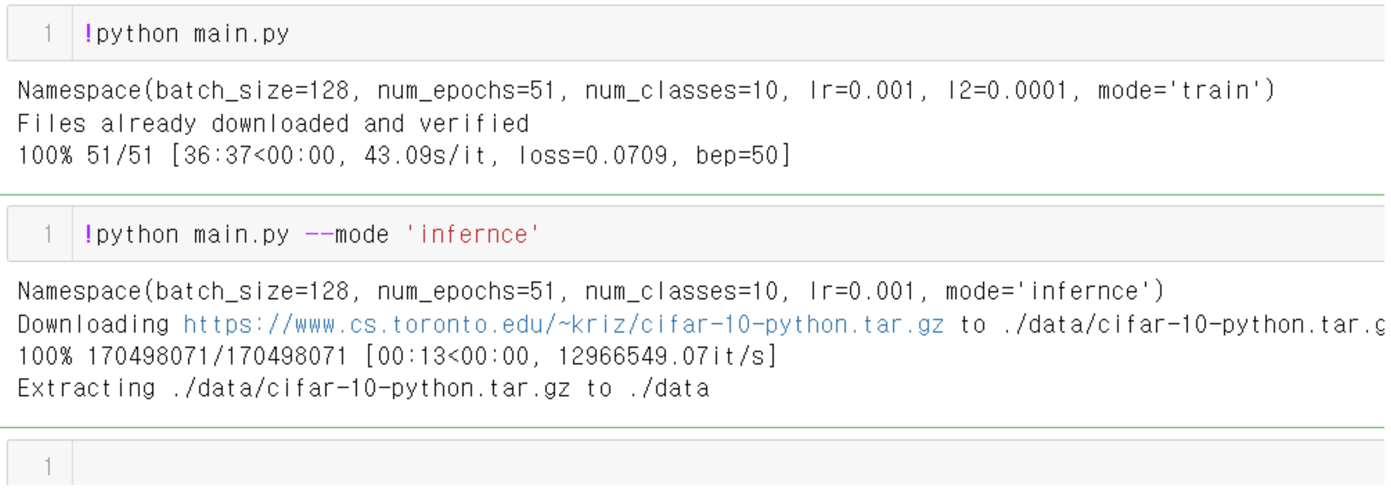
RepVGG

3*3 Conv2만 사용



main.py

```python
import repvgg # RepVGG는 오직 3x3 Conv를 사용하는데, 이는 NVIDIA-GPU에서 3x3 conv의 성능이 다른 커널 사이즈보다 훨씬 빠르기 때문이라 합니다

import train_test as tt
import datasets
import torch
import argparse

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='RepVGG')
    parser.add_argument('--batch_size', default=128, type=int, help='batch size')
    parser.add_argument('--num_epochs', default=51, type=int, help='training epoch')
    parser.add_argument('--num_classes', default=10, type=int, help='number of classes')
    parser.add_argument('--lr', default=1e-3, type=float, help='learning rate')
    parser.add_argument('--mode', default='train', type=str, help='train and inference')
    args = parser.parse_args()
    print(args)

    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

    if args.mode == 'train':
        dataloader = datasets.dataloader(datatype='cifar10', batch_size=args.batch_size, mode=args.mode)
        model = repvgg.repvgg(layer_list=[2,2,3,3,3], num_classes=args.num_classes)
        tt.train(dataloader=dataloader, model=model, num_epochs=args.num_epochs, lr=args.lr, device=device)

    else:  # 평가나 추론을 할때는  웨이트 폴더에 있는 값으로 작업함.
        dataloader = datasets.dataloader(datatype='cifar10', batch_size=args.batch_size, mode=args.mode)
        model = repvgg.repvgg(layer_list=[2,2,3,3,3], num_classes=args.num_classes, mode='inference', param='./weights/repvgg.pth')
        tt.test(dataloader=dataloader, model=model.to(device), name='vgg', device=device)
```



```
!python main.py
```

```
Namespace(batch_size=128, num_epochs=51, num_classes=10, lr=0.001, l2=0.0001, mode='train')
Files already downloaded and verified
100% 51/51 [36:37<00:00, 43.09s/it, loss=0.0709, bep=50]
```

```
!python main.py --mode 'infernce'
```

```
Namespace(batch_size=128, num_epochs=51, num_classes=10, lr=0.001, mode='infernce')
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.g
100% 170498071/170498071 [00:13<00:00, 12966549.07it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
```

재매개변수화
적용: 5.88초
미적용: 10.28초(1.75배)