Homework 3 - HTC Deep Q

學號： **B05901082** 系級： 電機三 姓名**:** 楊晟甫

Problem 1. 做了那些比較的實驗

- Pretrained： True or False
  - Result : pretrained model 的 Acc 明顯高出很多
- Optimizer： SGD、Adam、Adamax
  - Result：Adamax 收斂得較好
- Learning rate： 1e-3, 1e-4, 1e-5
  - Result：1e-3 step太大、1e-5逼近的不夠快，所以選1e-4效果最佳
- Batch_size： 32, 64
  - Result： 32 的Acc較好
- Data Preprocessing：
  - Result：有Data Augmentation的performace較好，如flip、shear、scale、translate
- Batch_normalization：None, 0.1
  - Result：並沒有很顯著的差距
- Dropout：None, 0.3, 0.5
  - Result：0.3 has higher accuracy

Problem 2. 最後選擇此值 **/** 此 **scheduling** 的原因

- 根據初始的假設以及實驗結果(Valid Score)，以及Kaggle上Public Score，分別對Mobilenetv2以及resnet50進行微調，最後兩個model分別選擇以下的scheduling：
- Mobilenetv2：

```
def get_random_seed():
    seed = None
    return seed
def get_model_spec():
    return {"model_name": "mobilenetv2", "pretrained": True}
def get_optimizer(params):
    optimizer = torch.optim.Adamax(params, lr = 1e-4, weight_decay= 0.01)
    return optimizer
def get_eval_spec():
    transform =  transforms.Compose([
        transforms.Resize(224),
        transforms.ToTensor()
        ])
    return {"transform": transform, "batchsize": 32}
def before_epoch(train_history, validation_history):
    transform = transforms.Compose([
        transforms.Resize(224),
        transforms.RandomHorizontalFlip(p=0.3),
```

```
        transforms.RandomAffine(0, translate=(0.1, 0.1), scale=None, shear=15,
resample=False, fillcolor=0),
        transforms.ToTensor()
        ])
    n_epoch = len(train_history)
    return {"transform": transform, "batchsize": 32}
def before_batch(train_history, validation_history):
    return {"optimizer": {"lr": 1e-4}, "batch_norm": 0.1, "drop_out": 0.3}
```

- ResNet50 :

```
def get_random_seed():
    seed = None
    return seed
def get_model_spec():
    return {"model_name": "resnet50", "pretrained": True}
def get_optimizer(params):
    optimizer = torch.optim.Adamax(params, lr = 5e-5, weight_decay= 0.01)
    return optimizer
def get_eval_spec():
    transform =  transforms.Compose([
        transforms.Resize(224),
        transforms.ToTensor()
        ])
    return {"transform": transform, "batchsize": 64}
def before_epoch(train_history, validation_history):
    transform = transforms.Compose([
        transforms.Resize(224),
        transforms.RandomHorizontalFlip(p=0.3),
        transforms.RandomAffine(0, translate=(0.1, 0.1), scale=None, shear=15,
resample=False, fillcolor=0),
        transforms.ToTensor()
        ])
    n_epoch = len(train_history)
    return {"transform": transform, "batchsize": 64}
def before_batch(train_history, validation_history):
    return {"optimizer": {"lr": 5e-5}, "batch_norm": 0.1, "drop_out": None}
```