Homework 3

Gheorghita Stefana

November 4, 2024

1 Introduction

For this assignment I created a training pipeline with the following characteristics:

- it is device agnostic, can work on CPU and GPU;
- can support any dataset, currently the training on MNIST, CIFAR10, CIFAR100;
- it includes a caching component;
- the DataLoaders are configurable for training and testing;
- it can support any model; currently, it supports loading:
  1. for CIFAR: resnet18 from timm (pretrained or not), resnet50 from timm (pretrained or not), preactresnet18 from lab 2, resnet18 resized and resnet50 resized (an up-sampling layer is added to the original models from timm), resnet18_cifar10;
  2. for MNIST: MLP (a predefined class) and CustomMLP (customizable number of layers, activation function) and LeNet;
- can be configured to use any optimizer, such as (SGD, SGD with momentum, SGD with Nesterov, SGD with weight decay, Adam, AdamW, RMSProp;
- can be configured to use any scheduler, currently accepts: StepLR, ReduceLROnPlateau, CosineAnnealingLR, CosineAnnealingLRWarmRestarts, ExponentialLR, LinearLR or None;
- supports early stopping (accepting two modes: *min* - for loss and *max* - accuracy);
- offers different types of data augmentation: it accepts custom augmentations as input and it also have predefined augmentation schemes;
- accepts warmup;

- has wandb support for metrics reporting;
- takes a yaml or json configuration file as input;

2 Results:

Experiment Number	Test Accuracy Achieved
1	59.72%
2	38.42%
3	59.42%
4	40.55%
5	79.4%
6	72.12%
7	81.88%
8	64.94%
9	80.61%
10	69.63%

Table 1: Experiment Results

1. **Experiment 1**:

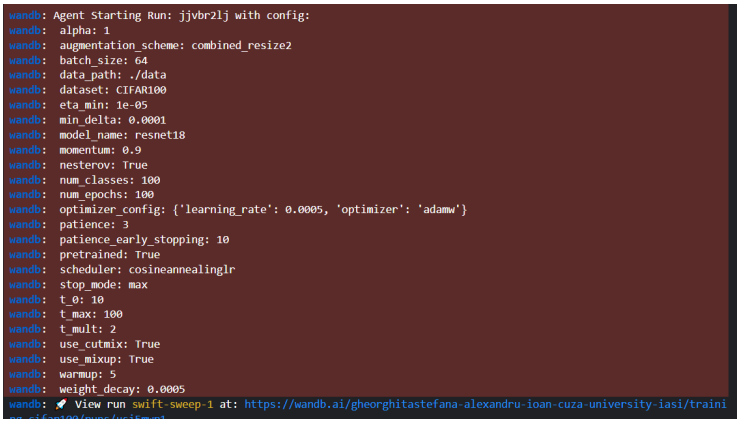

Figure 1: Configuration

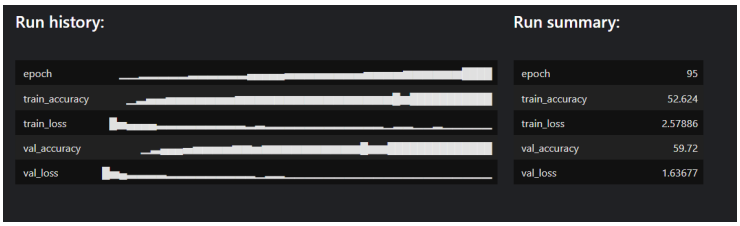

Figure 2: Result

Configuration:

- Model: ResNet18 pretrained;
- Optimizer: AdamW, Learning rate: 0.0005;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandomRotation(10),
  - RandomResizedCrop(32, scale=(0.9, 1.1)),
  - RandomHorizontalFlip(),
  - RandomAffine(degrees=0, shear=10),
  - RandomCrop(32, padding=3),
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
- Batch size: 64;
- Early stopping;
- Number of epochs: 100;

2. **Experiment 2**:

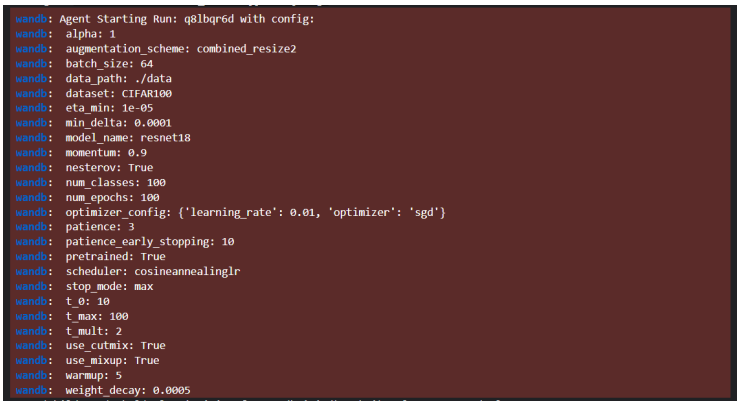

Figure 3: Configuration

Figure 4: Result

Configuration:

- Model: ResNet18 pretrained;
- Optimizer: SGD, Learning rate: 0.01;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandomRotation(10),
  - RandomResizedCrop(32, scale=(0.9, 1.1)),
  - RandomHorizontalFlip(),
  - RandomAffine(degrees=0, shear=10),
  - RandomCrop(32, padding=3),
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
- Batch size: 64;

- Early stopping;
- Number of epochs: 100;

3. **Experiment 3**:

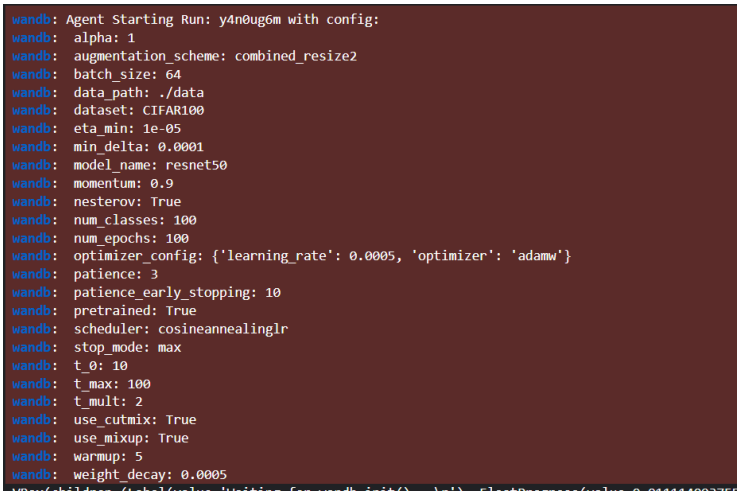

Figure 5: Configuration

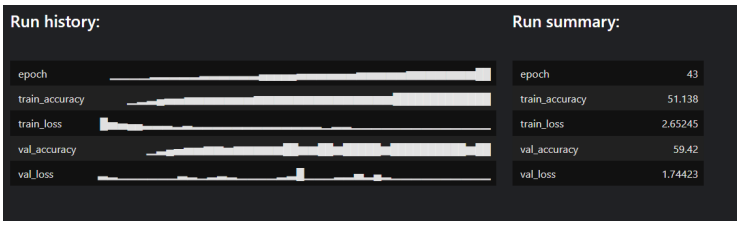

Figure 6: Result

Configuration:

- Model: ResNet50 pretrained;
- Optimizer: AdamW, Learning rate: 0.0005;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandomRotation(10),
  - RandomResizedCrop(32, scale=(0.9, 1.1)),

  - RandomHorizontalFlip(),
  - RandomAffine(degrees=0, shear=10),
  - RandomCrop(32, padding=3),
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
- Batch size: 64;
- Early stopping;
- Number of epochs: 100;

4. **Experiment 4**:

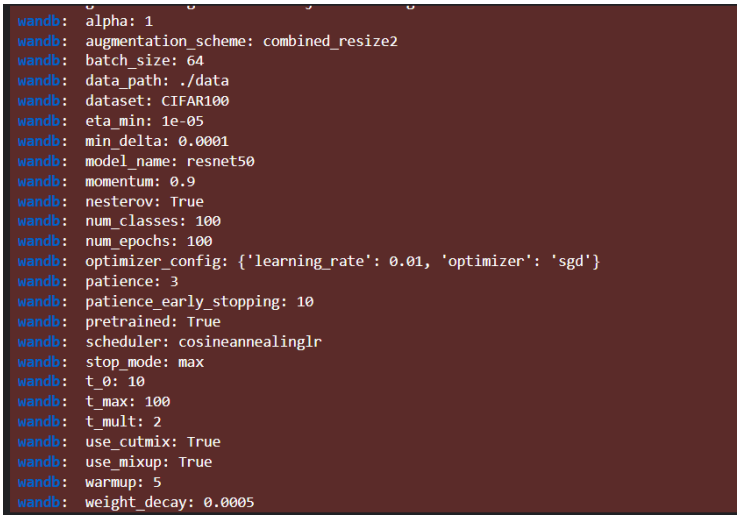

Figure 7: Configuration

Figure 8: Result

Configuration:

- Model: ResNet50 pretrained;

- Optimizer: SGD, Learning rate: 0.01;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandomRotation(10),
  - RandomResizedCrop(32, scale=(0.9, 1.1)),
  - RandomHorizontalFlip(),
  - RandomAffine(degrees=0, shear=10),
  - RandomCrop(32, padding=3),
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
- Batch size: 64;
- Early stopping;
- Number of epochs: 100;

5. **Experiment 5**:

```
wandb: Sweep Agent: Waiting for job.
wandb: Job received.
wandb: Agent Starting Run: 6dgaym2g with config:
wandb:  alpha: 1
wandb:  augmentation_scheme: combined_resize2
wandb:  batch_size: 64
wandb:  data_path: ./data
wandb:  dataset: CIFAR100
wandb:  eta_min: 1e-05
wandb:  min_delta: 0.0001
wandb:  model_name: resnet18_resize
wandb:  momentum: 0.9
wandb:  nesterov: True
wandb:  num_classes: 100
wandb:  num_epochs: 100
wandb:  optimizer_config: {'learning_rate': 0.0005, 'optimizer': 'adamw'}
wandb:  patience: 3
wandb:  patience_early_stopping: 10
wandb:  pretrained: True
wandb:  scheduler: cosineannealinglr
wandb:  stop_mode: max
wandb:  t_0: 10
wandb:  t_max: 100
wandb:  t_mult: 2
wandb:  use_cutmix: True
wandb:  use_mixup: True
wandb:  warmup: 5
wandb:  weight_decay: 0.0005
```

Figure 9: Configuration

Figure 10: Result

Configuration:

- Model: ResNet18 pretrained + Up-sampling to (224, 224);
- Optimizer: AdamW, Learning rate: 0.0005;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandomRotation(10),
  - RandomResizedCrop(32, scale=(0.9, 1.1)),
  - RandomHorizontalFlip(),
  - RandomAffine(degrees=0, shear=10),
  - RandomCrop(32, padding=3),
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
- Batch size: 64;
- Early stopping;
- Number of epochs: 20;

6. **Experiment 6**:

```
wandb: Agent Starting Run: bb1xm74z with config:
wandb:  alpha: 1
wandb:  augmentation_scheme: randaugment
wandb:  batch_size: 64
wandb:  data_path: ./data
wandb:  dataset: CIFAR100
wandb:  eta_min: 1e-05
wandb:  min_delta: 0.0001
wandb:  model_name: preactresnet18
wandb:  momentum: 0.9
wandb:  nesterov: True
wandb:  num_classes: 100
wandb:  num_epochs: 50
wandb:  optimizer_config: {'learning_rate': 0.0005, 'optimizer': 'adamw'}
wandb:  patience: 3
wandb:  patience_early_stopping: 10
wandb:  pretrained: True
wandb:  scheduler: cosineannealinglr
wandb:  stop_mode: max
wandb:  t_0: 10
wandb:  t_max: 100
wandb:  t_mult: 2
wandb:  use_cutmix: True
wandb:  use_mixup: True
wandb:  warmup: 5
wandb:  weight_decay: 0.0005
wandb: Currently logged in as: gheorghitastefana (gheorghitastefana-alexandru-ioan-cuza-university-iasi). Use `wandb
login --relogin` to force relogin
```

Figure 11: Configuration

Figure 12: Result

Configuration:

- Model: PreActResNet18;
- Optimizer: AdamW, Learning rate: 0.0005;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandAugment()
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.5,), (0.5,))
- Batch size: 64;
- Early stopping;
- Number of epochs: 50;

7. **Experiment 7**:

Figure 13: Configuration

Figure 14: Result

Configuration:

- Model: ResNet18 pretrained + Up-sampling to (224, 224);
- Optimizer: AdamW, Learning rate: 0.0005;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandAugment()
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.5,), (0.5,))
- Batch size: 64;
- Early stopping;

- Number of epochs: 20;

8. **Experiment 8**:

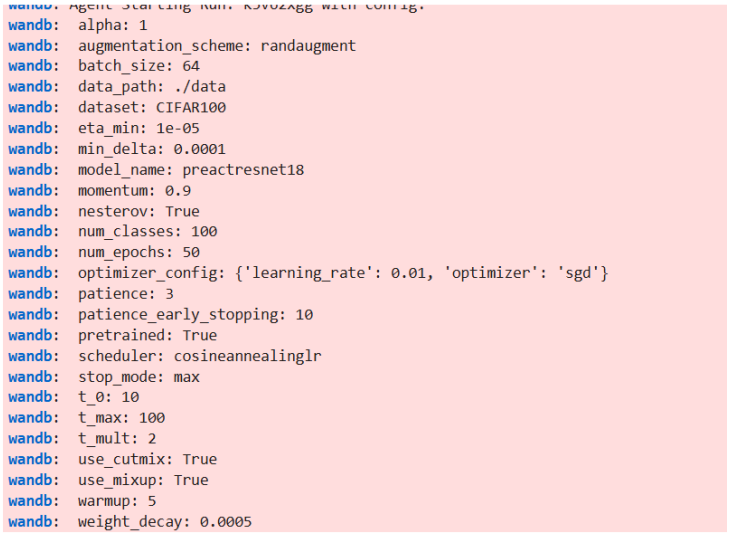

Figure 15: Configuration

Figure 16: Result

Configuration:

- Model: PreActResNet18;
- Optimizer: SGD, Learning rate: 0.01;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandAugment()
  - ToImage(),

  - ToDtype(torch.float32, scale=True),
  - Normalize((0.5,), (0.5,))
- Batch size: 64;
- Early stopping;
- Number of epochs: 50;

9. **Experiment 9**:

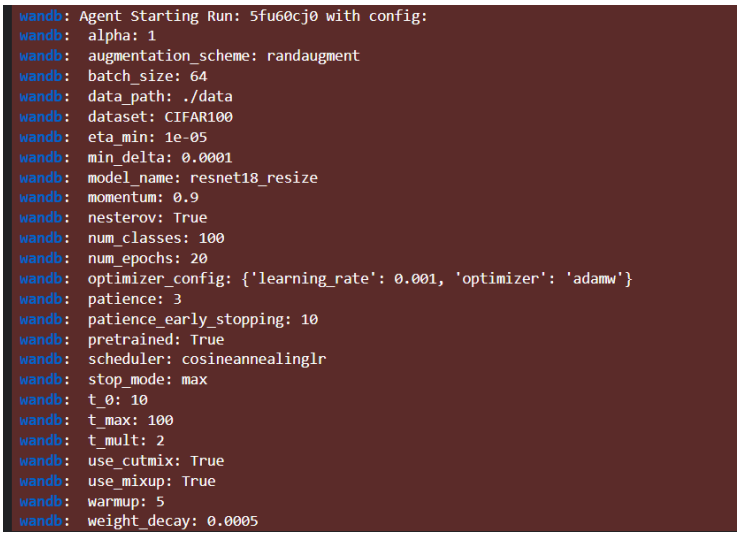

Figure 17: Configuration

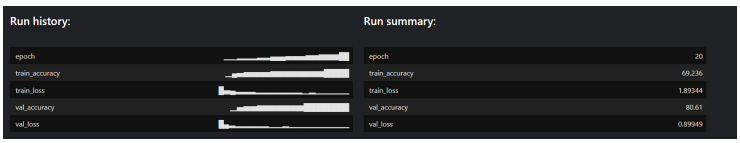

Figure 18: Result

Configuration:

- Model: PreActResNet18;
- Optimizer: AdamW, Learning rate: 0.001;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp

  - RandAugment()
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.5,), (0.5,))
- Batch size: 64;
- Early stopping;
- Number of epochs: 20;

10. **Experiment 10**:

```
wandb: Agent Starting Run: sjsmddpt with config:
wandb:  alpha: 1
wandb:  augmentation_scheme: combined
wandb:  batch_size: 64
wandb:  data_path: ./data
wandb:  dataset: CIFAR100
wandb:  eta_min: 1e-05
wandb:  min_delta: 0.0001
wandb:  model_name: preactresnet18
wandb:  momentum: 0.9
wandb:  nesterov: True
wandb:  num_classes: 100
wandb:  num_epochs: 50
wandb:  optimizer_config: {'learning_rate': 0.0005, 'optimizer': 'adamw'}
wandb:  patience: 3
wandb:  patience_early_stopping: 10
wandb:  pretrained: True
wandb:  scheduler: cosineannealinglr
wandb:  stop_mode: max
wandb:  t_0: 10
wandb:  t_max: 100
wandb:  t_mult: 2
wandb:  use_cutmix: True
wandb:  use_mixup: True
wandb:  warmup: 5
wandb:  weight_decay: 0.0005
```

Figure 19: Configuration

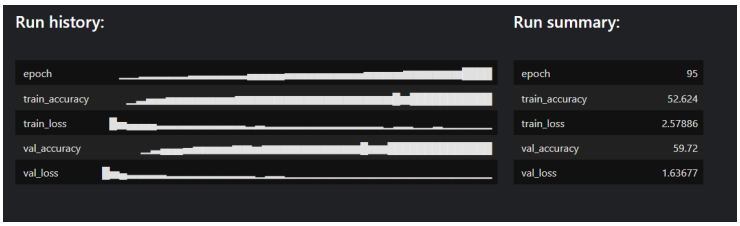

Figure 20: Result

Configuration:

- Model: PreActResNet18;

- Optimizer: AdamW, Learning rate: 0.0005;
- Scheduler: CosineAnnealingLR, t_max= 100, eta_min=1e-5;
- Augmentation:
  - CutMix
  - MixUp
  - RandomResizedCrop(28, scale=(0.8, 1.0))
  - RandomRotation(15)
  - RandAugment()
  - ToImage(),
  - ToDtype(torch.float32, scale=True),
  - Normalize((0.5,), (0.5,))
- Batch size: 64;
- Early stopping;
- Number of epochs: 50;

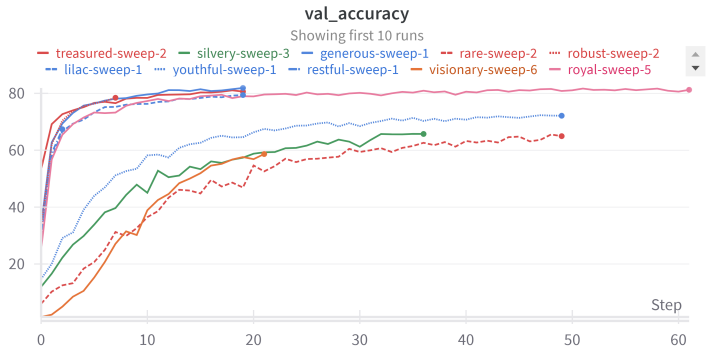

Figure 21: Accuracy

Figure 22: Caption

3 Description

During the training process, I varied several parameters, including the optimizer and its settings, the learning rate scheduler, data augmentation techniques, batch size, and the model architecture. For the optimizer, I used both AdamW and SGD, finding AdamW to be more effective in this case. The choice of scheduler also proved important; I experimented with CosineAnnealing, ReduceLROnPlateau, and StepLR.

In terms of data augmentation, I applied MixUp and CutMix alongside other techniques in various configurations, such as RandAugment, RandomResizedCrop, RandomHorizontalFlip, and RandomRotation.

An essential factor in achieving 81% accuracy was the model choice. Through my experiments, I observed that ResNet50 yielded better results than ResNet18, but neither variant achieved the desired accuracy level. Neither the regular nor the pretrained versions achieved the target test accuracy. Given that ResNet18 is pretrained on ImageNet, which uses images of size (224, 224), I decided to add an up-sampling layer, which led to the desired accuracy. Previously, I tried resizing the images to (224, 224) in the augmentation configuration, but this led to overfitting.

For enhancing the efficiency of the training pipeline, I implemented the following features:

- I use torch.amp.autocast to accelerate computation and reduce memory usage without significant loss in model accuracy.
- I use GradScaler to stabilize training.
- The model accepts custom data augmentation configurations leading to a flexible scheme selection.
- The num_workers parameter in DataLoader allows parallel data loading and preprocessing, reducing the CPU bottleneck during training.

- I added warmup and early stopping;
- The best model is saved;
- The use of RandomChoice between CutMix and MixUp avoids a fixed augmentation pattern. This makes training more robust and can enhance generalization without introducing excessive computational overhead.
- I also added torch.backends.cudnn.benchmark which is useful when the input size does not change.
- The input can be a configuration file of type yaml/json.
- In the sweep configuration the optimizer and learning rate are placed together to avoid not so efficient combinations.

W&B links:

- W&B Training CIFAR100 Project
- W&B Generalized Training Project

I can't provide an exact number of points I expect to receive, as I aimed to complete all the assignment requirements but may have made some mistakes while doing it.