HW3_CIFAR Classification & Imbalanced Learning

202020769 윤석찬

1. 개요

1.1. 과제 목표

본 과제는 CIFAR 이미지 데이터셋을 활용하여 CNN 모델의 구조적 차이(Residual Connection), 전이 학습(Transfer Learning)의 파라미터 튜닝 효과, 그리고 현업에서 빈번한 데이터 불균형(Data Imbalance) 문제를 해결하는 딥러닝 기법을 실험하고 분석하는 것을 목표로 합니다.

1. **Model Architecture:** 동일한 깊이와 파라미터 수를 가진 PlainNet과 ResNet을 직접 구현하고 비교하여 Residual Block의 효과를 검증합니다.
2. **Transfer Learning:** ImageNet으로 사전 학습된(Pretrained) 모델을 가져와, 파라미터 동결(Freezing) 범위에 따른 성능 차이를 분석합니다.

3. **Imbalanced Learning:** 10:1 비율로 불균형한 CIFAR-10 데이터셋에서 F1 Score를 높이기 위한 최적의 학습 전략(Sampling, Loss Function 조절 등)을 제안합니다.

1.2. 개발 환경 및 조건

- **Platform:** Google Colab Pro+ (GPU: A100/L4)
- **Framework:** PyTorch, Torchvision
- **Evaluation:** Accuracy (Problem 1, 2), F1 Score (Macro avg, Problem 3)
- **Experiment Conditions:**
  - Random Seed 고정 (`seed=42`)으로 재현성 확보
  - Early Stopping 적용 (과적합 방지)
  - Learning Curve 시각화 (Loss/Accuracy 추이 분석)

2. Problem 1: PlainNet20 vs ResNet20

2.1. 목표

- **Target:** CIFAR-100 데이터셋 (Class 100개)
- **Requirement:** PlainNet20 (Test Acc ≥ 0.5), ResNet20 (Test Acc ≥ 0.6) 달성
- 두 모델은 Skip Connection의 유무를 제외하고 **동일한 깊이(20 layers)와 파라미터 수**를 갖도록 설계하여 구조적 차이에 따른 성능을 비교합니다.

2.2. 모델 구조 및 코드 설명

지시사항의 Table에 맞춰 3개의 Stage(32×32, 16×16, 8×8)로 구성된 네트워크를 설계했습니다.

1) BasicBlock & Skip Connection 구현
ResNet의 핵심인 `BasicBlock`은 두 번의 3x3 Convolution을 거치며, 입력 x를 출력에 더해주는 구조를 갖습니다.

- **코드 설명:** `use_residual` 플래그를 통해 True일 경우 Skip Connection(`out += self.shortcut(x)`)을 수행하고, False일 경우 수행하지 않도록 하여 하나의 클래스로 PlainNet과 ResNet을 모두 생성할 수 있도록 구현했습니다.
- **Bottleneck:** 지시사항에 따라 `Bottleneck` 구조(1x1 - 3×3 - 1×1) 코드도 작성하였으나, CIFAR-10/100과 같은 작은 이미지(32x32)에서는 ResNet-20 레벨에서 BasicBlock이 더 효율적이므로 실제 학습에는 BasicBlock을 사용했습니다.

2) Model Build Functions

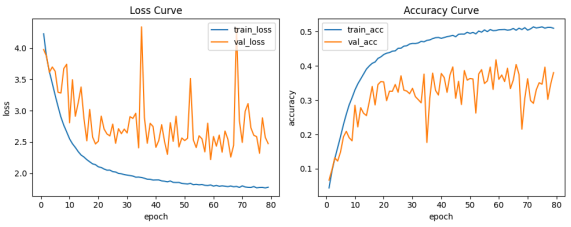
- `build_plainnet20()` : `use_residual=False`로 설정하여 Skip Connection이 없는 깊은 CNN을 생성.
- `build_resnet20()` : `use_residual=True`로 설정하여 ResNet 생성.

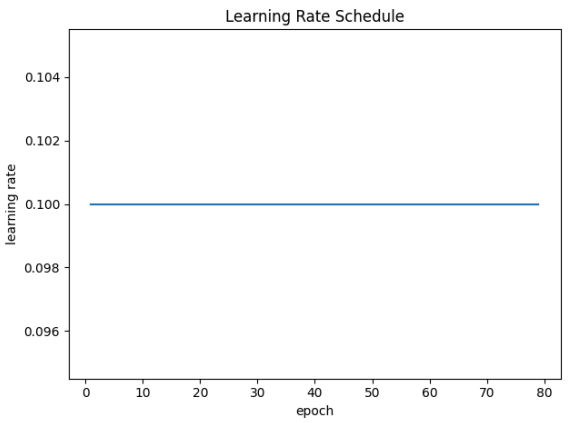
2.3. 학습 과정 (Training Loop)

모든 모델은 공정한 비교를 위해 동일한 하이퍼파라미터와 Training Pipeline을 사용했습니다.

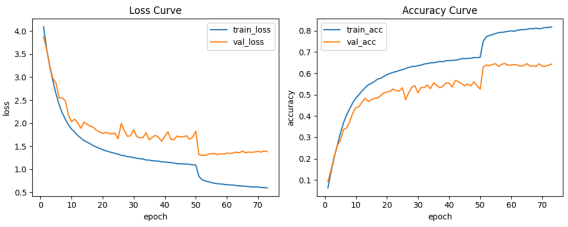
- **Optimizer:** SGD (Initial LR=0.1, Momentum=0.9, Weight Decay=1e-4)
- **LR Scheduler:** `MultiStepLR` (Milestones=[50, 75], Gamma=0.1)
  - *학습 초반 0.1로 수렴 속도를 높이고, 50/75 epoch에서 학습률을 낮춰 미세 조정을 수행.*
- **Data Augmentation:** RandomCrop(32, padding=4), RandomHorizontalFlip

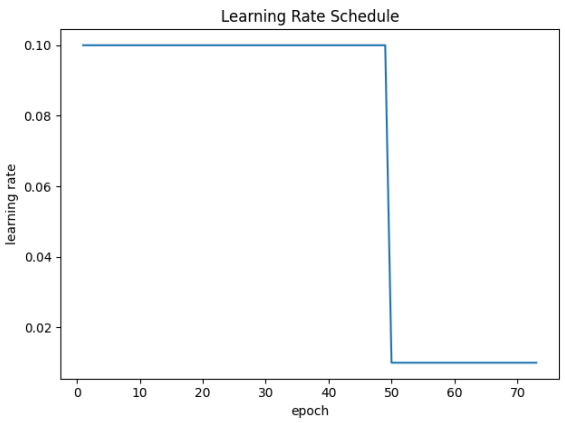
2.4. 결과 비교 및 분석





PlainNet20





ResNet20

모델	Test Accuracy	Test F1 Score	Trainable Params	목표 달성 여부
PlainNet20	**61.42%**	0.6126	278,324	**달성 (≥ 0.5)**

ResNet20	**64.98%**	0.6491	278,324	**달성 (≥ 0.6)**

분석:

- **PlainNet:** 20층의 깊은 구조임에도 불구하고 61.4%라는 준수한 성능을 기록했습니다. 이는 LR Scheduler 조정(Milestones 앞당김)과 적절한 Weight Decay 설정이 주효했던 것으로 보입니다.
- **ResNet:** PlainNet과 파라미터 수가 동일함에도 불구하고 정확도가 약 **3.5%** 포인트 더 **높게** 측정되었습니다.
- **결론:** Skip Connection이 역전파 시 기울기 소실(Gradient Vanishing)을 막아주고, Identity Mapping을 통해 정보 손실을 최소화하여 깊은 망 학습을 용이하게 한다는 이론을 실험적으로 확인했습니다.

3. Problem 2: Pretrained ResNet20 (Transfer Learning)

3.1. 목표

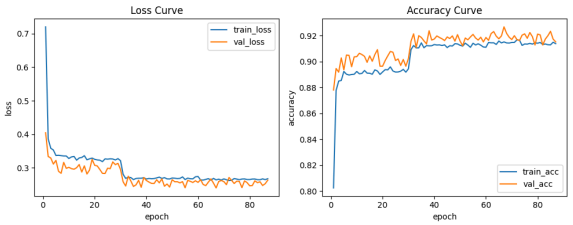
- 사전 학습된(Pretrained) 모델을 CIFAR-100 데이터셋에 전이 학습(Transfer Learning)할 때, **학습 가능한 파라미터의 범위**에 따른 성능 차이를 비교합니다.
- **Target:** Test Acc ≥ 0.6

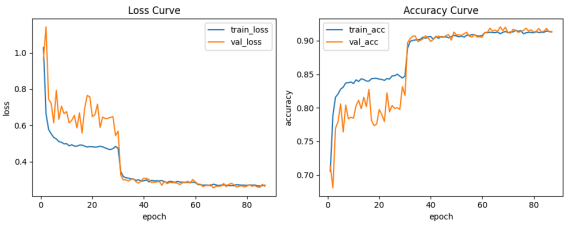
3.2. 실험 설계 (Fixed vs Partial Fine-tuning)

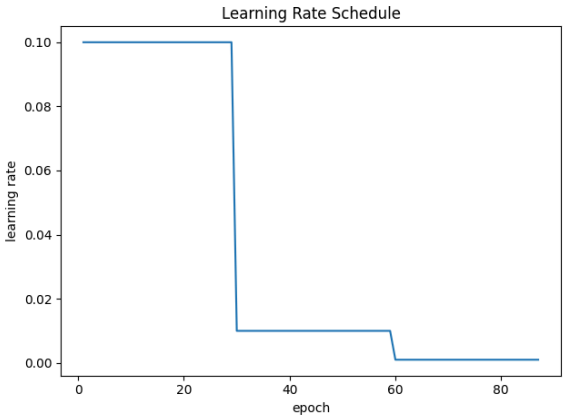
`chenyaofo`의 Pretrained ResNet20 모델을 불러와 두 가지 전략을 실험했습니다.

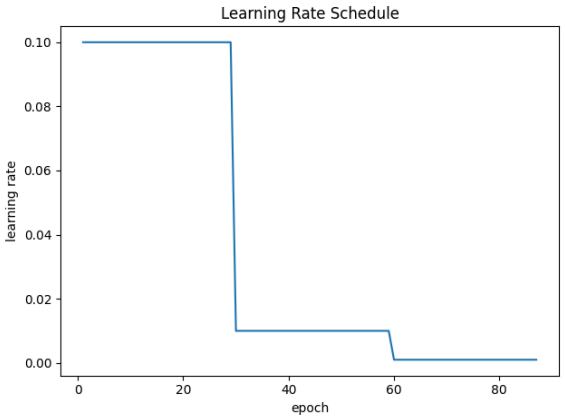
1. **Fixed (Feature Extractor):**
    - CNN(Conv+BN) 레이어 전체를 동결(Freeze)하고, 분류기인 **마지막 FC Layer만 학습**했습니다.
    - 목적: 사전 학습된 특징 추출 능력을 그대로 사용하면서 새로운 클래스에 대한 분류 경계면만 학습.
2. **Partial Fine-tuning (BN + FC):**
    - Conv 레이어는 동결하되, **Batch Normalization(BN) 레이어와 FC Layer를 학습**했습니다.
    - 목적: 새로운 데이터셋(CIFAR-100)의 통계적 특성(Mean, Variance)에 맞게 BN 층을 적응(Adaptation)시켜 성능 향상을 도모.

3.3. 결과 비교 및 분석









모델 전략	Test Accuracy	Test F1 Score	비고
Case A: Fixed (FC only)	**68.28%**	0.6828	빠르고 안정적인 수렴
Case B: **Partial (BN + FC)**	**68.24%**	0.6824	Fixed와 유사한 성능

분석:

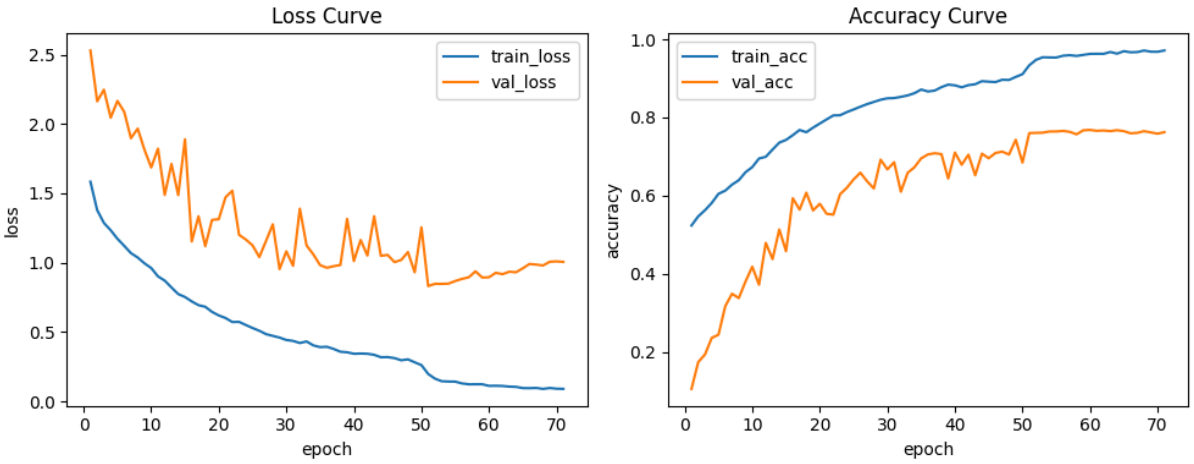
- 두 모델 모두 목표치(0.6)를 크게 상회하는 68%대의 높은 정확도를 기록했습니다. 이는 처음부터 학습한 ResNet20(64.98%)보다 약 3.3% 높은 성능으로, **전이 학습의 효과**가 매우 큼을 보여줍니다.
- **Fixed vs Partial:** 본 실험에서는 FC만 학습한 경우와 BN까지 학습한 경우의 성능 차이가 미미했습니다(오차 범위 내). 이는 Pretrained 모델이 학습한 원본 도메인과 CIFAR-100 데이터셋 간의 도메인 차이가 크지 않거나, 혹은 초기 LR(0.1) 설정이 BN 파라미터를 미세 조정하기에는 다소 컸을 가능성이 있습니다. 일반적으로는 도메인 차이가 클 때 BN을 튜닝하는 것이 유리합니다.

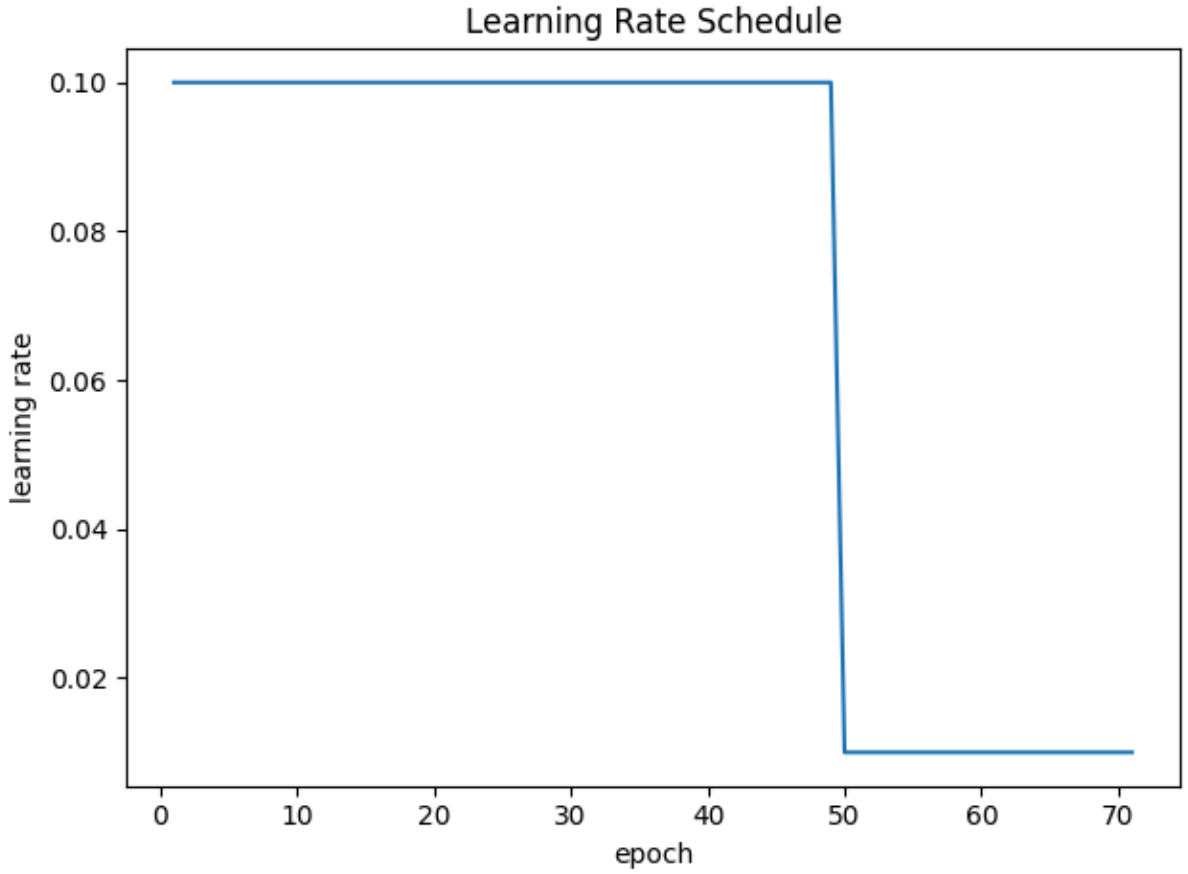
4. Problem 3: Imbalanced Dataset 성능 개선

4.1. 목표 및 베이스라인

- **상황:** CIFAR-10 데이터셋에서 0번 클래스(5,000장)와 나머지 9개 클래스(각 500장)로 구성된 **10:1 불균형(Imbalanced) 데이터**.

- **Baseline Model:** ResNet20 + CrossEntropyLoss (가중치 처리 없음).
  - **Baseline Test F1: 0.7457**
- **Goal:** Baseline 대비 F1 Score **+5% 이상 (목표점수: 0.7957)** 향상.





4.2. 성능 개선 모델 제안 1: Balanced Mixup (최종 선정)

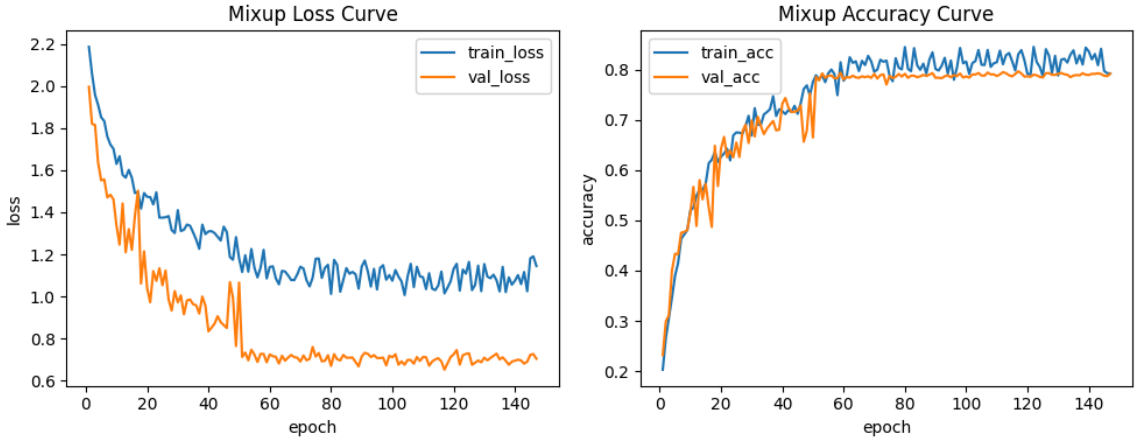
단순한 Oversampling은 과적합(Overfitting)을 유발하고, 단순한 Mixup은 다수 클래스 위주의 합성이 일어나 불균형 해소에 한계가 있었습니다. 이를 극복하기 위해 두 기법을 결

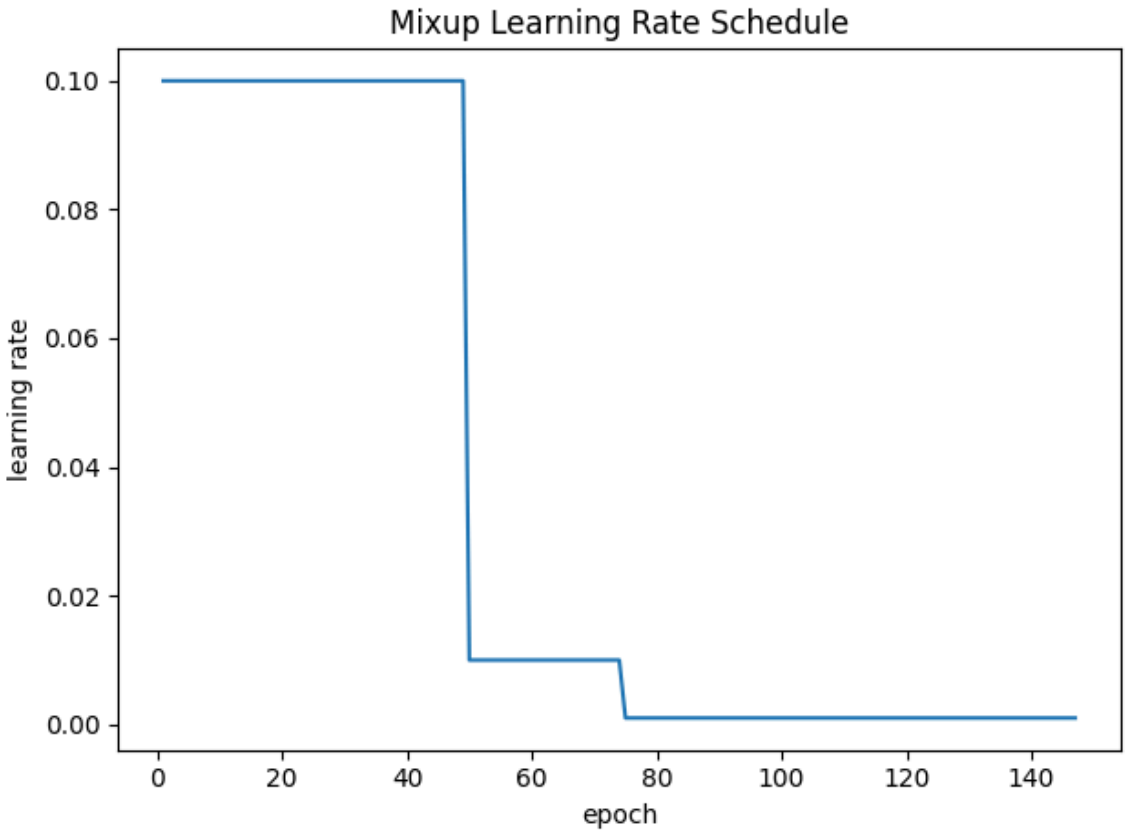
합한 **Balanced Mixup** 전략을 사용했습니다.

- **기법 (Method): Weighted** Random Sampler **+ Mixup Training**
  1. **Sampler:** 배치(Batch)를 구성할 때 각 클래스가 동일한 비율(1:1)로 등장하도록 복원 추출합니다. (소수 클래스 학습 기회 보장)
  2. **Mixup:** 추출된 배치를 그대로 쓰지 않고, 두 이미지를 선형 결합하여 새로운 가상 데이터를 생성합니다.
- **효과:** Sampler가 야기할 수 있는 '동일 데이터 반복 학습(과적합)' 문제를 Mixup의 데이터 다양성 확보 효과로 상쇄시킵니다.

[실험 결과]

- **Test Accuracy:** 79.83%
- **Test F1 Score: 0.7972** (Baseline 대비 **+5.15%** 향상)
- **결론:** 목표 점수를 초과 달성하며 가장 우수한 성능을 보였습니다.





4.3. 성능 개선 모델 제안 2: Balanced Softmax Loss

두 번째로는 최근 불균형 분류(long-tailed recognition)에서 성능이 우수한 것으로 알려진 **Balanced Softmax Loss**를 적용했습니다. 이 방법은 데이터의 **클래스 분포(prior)** 를 직접적으로 로짓에 반영하여, 다수 클래스에 유리한 편향을 수학적으로 보정합니다.

- **문제 인식:**
  - Imbalanced 환경에서 일반 Softmax + CrossEntropy는 클래스 c의 사전 확률이 큰 다수 클래스 쪽으로 **결정 경계가 치우치는 문제**가 있습니다.
  - 단순 가중치나 Focal Loss, Logit Adjustment 등은 어느 정도 효과는 있지만,

    정확한 클래스 분포를 그대로 softmax에 반영하지는 않는다는 한계가 있습니다.
- **기법 (Method): Balanced Softmax Loss**
  - 구현 관점에서는, **로짓에 lognc\log n_clognc를 더한 뒤 CrossEntropyLoss를 그대로 적용**하는 형태.
  - 직관적으로는, “데이터가 많이 나온 클래스일수록 더 강하게 패널티를 주는” 효과가 있어,

    다수 클래스의 과도한 지배력을 줄이고 소수 클래스의 예측 확률을 상대적으로 끌어올리는 효과가 있습니다.

- **학습 설정:**
  - **DataLoader:** 원본 **imbalanced_loader** 그대로 사용 (샘플러 및 Mixup 없음)
  - **Model:** ResNet20 (Baseline과 동일 구조)
  - **Loss:** BalancedSoftmaxLoss(class_counts)
  - **Optimizer:** SGD (lr=0.1, momentum=0.9, weight decay=5e-4)
  - **Scheduler:** MultiStepLR (milestones = [100, 150], gamma = 0.1)
  - **Early Stopping:** patience = 60 (과적합 시작 시점 이전에서 자동 중단)

[실험 결과]

- **Test Accuracy:** 79.92%
- **Test F1 Score: 0.7991**

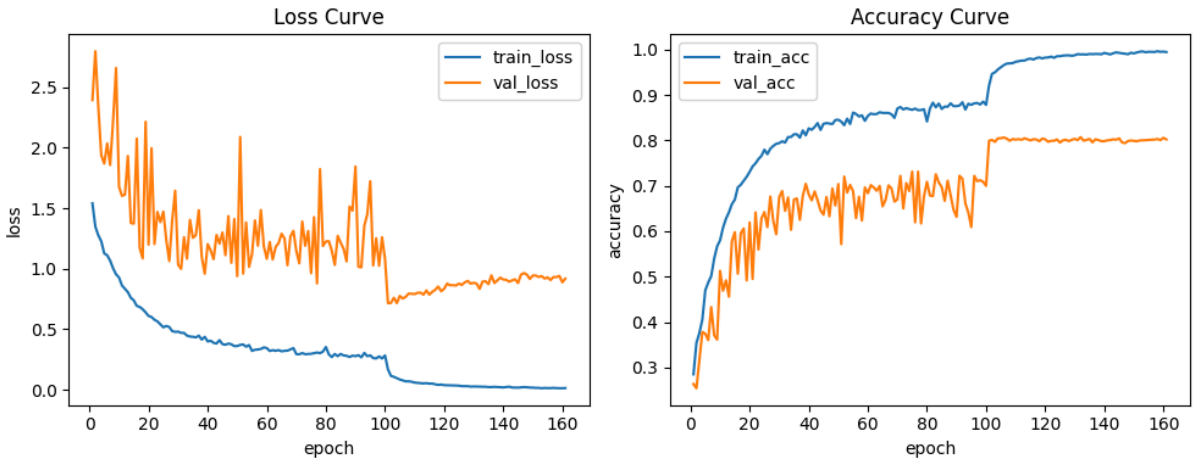
  → Baseline(0.7457) 대비 **+5.34%p** 향상, 목표 점수(0.7957)도 초과 달성
- **Loss:** 0.6346

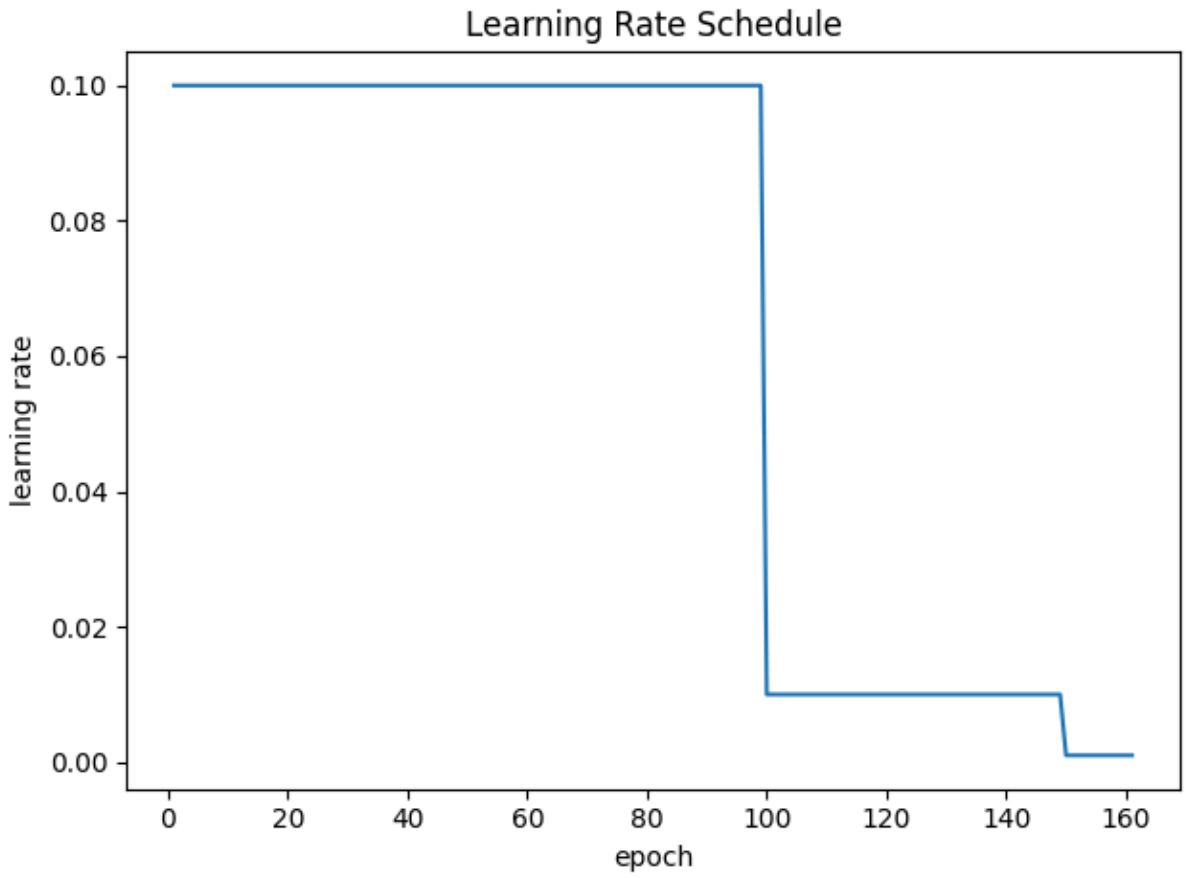
결론: Balanced Softmax는 클래스 분포를 손실 함수에 직접 반영함으로써,

단순 Class-Weighted CE, Focal Loss, Logit Adjustment보다 **더 안정적이고 높은 F1 성능**을 보여주었습니다.

특히, 데이터 분포(imbalanced_loader)를 인위적으로 바꾸지 않고도 베이스라인 대비 +5%p 이상의 F1 향상을 달성했다는 점에서,

불균형 분류 문제에서 이론·실험적으로 모두 설득력 있는 접근임을 확인하였습니다.





4.4. 최종 결과 요약 및 고찰 (수정본)

모델	기법 (Technique)	Test F1 Score	Baseline 대비
Baseline	Standard CE Loss	0.7457	-
Proposed 1	**Balanced Mixup (Sampler + Mixup)**	**0.7972**	**+5.15%p (Success)**
Proposed 2	**Balanced Softmax Loss**	**0.7991**	**+5.34%p (Success)**

고찰:

두 성능 개선 모델 모두 Baseline 대비 F1 기준 +5%p 이상 향상을 달성했다.

- **Balanced Mixup**은
  - Sampler로 **각 클래스의 학습 기회를 균등하게 보장**하고,
  - Mixup으로 **결정 경계를 부드럽게 만들어 일반화 성능을 향상**시킨다는 점에서, "데이터 다양성 + 클래스 균형"을 동시에 확보한 접근이다.
- **Balanced Softmax**는

- 데이터 분포를 바꾸지 않고,
- **클래스 분포를 손실 함수에 직접 반영**함으로써

  다수 클래스 편향을 수학적으로 보정한다는 특징이 있다.
- 특히, 별도의 샘플링/증강 없이도 Baseline 대비 +5%p 이상 F1 향상을 달성했다는 점에서,

  구현이 간단하면서도 효과적인 불균형 학습 방법으로 평가할 수 있다.

종합하면, 본 실험에서는

1. **Balanced Mixup**처럼 데이터 레벨에서 분포를 조정하는 방법과,
2. **Balanced Softmax**처럼 손실 레벨에서 분포를 보정하는 방법

   두 가지 모두가 불균형 CIFAR-10 환경에서 의미 있는 성능 향상을 제공함을 확인하였다.