

스터디 주간 활동 보고서

팀명	Robo:Loop	제출자 성명	홍송은
참여 명단	전효재, 홍송은, 김사웅		
모임 일시	2025 년 04 월 22 일 16 시 40 분 ~ 18 시 00(총 1 시간 30 분)		
장소	온라인 구글 미팅	출석 인원	3
학습목표	<ul style="list-style-type: none">• MNIST 데이터셋의 구조와 클래스 라벨을 이해하고 시각화한다.• 각자 구현한 모델을 공유하고 성능을 비교 분석한다.• 모델의 구성 방식과 성능 지표(loss, accuracy 등)를 바탕으로 학습 과정을 분석한다.• 각 레이어의 역할을 이해하고, 직접 모델을 설계할 수 있는 기반을 다진다.		
학습내용	<ol style="list-style-type: none">1. MNIST 란?<ul style="list-style-type: none">○ MNIST (Modified National Institute of Standards and Technology database)○ 손으로 쓴 숫자(0~9) 이미지들을 모아 놓은 데이터셋○ 흑백 이미지(1 채널), 28x28 사이즈2. MNIST 분류 모델 설계 및 결과 분석, 성능 개선<ul style="list-style-type: none">• 홍송은<ul style="list-style-type: none">○ 선형 분류 모델(Logestic Regression)과 MLP 비교		

- Logistic Regression 구현

- 단일 선형 계층 사용
- 정확도 92.29%

- MLP 구현

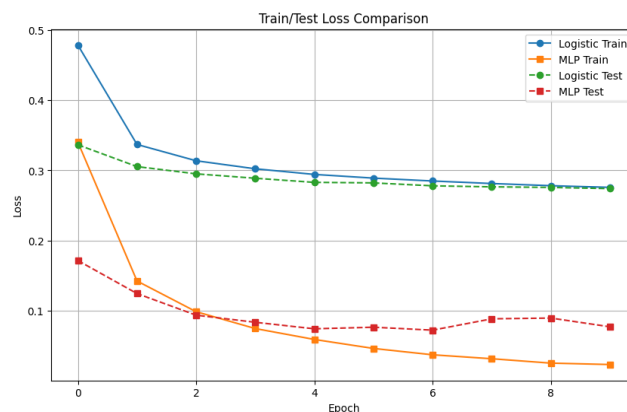
- hidden layer 2 개 (128 → 64), ReLU 활성화
- 정확도 97.29%

=> MLP 는 비선형성과 은닉층을 통해 표현력이 높아 더 복잡한 패턴을 학습할 수 있어 Logistic Regression 보다 성능이 높음

- 질의응답) DNN 과 MLP 의 구조가 비슷한데 어떤 차이?

MLP 는 DNN 의 하위 개념으로 hidden layer 가 1 개 이상 있는 완전 연결 신경망이며, DNN 은 은닉층이 2 개 이상 있는 신경망을 의미함.

- Train Loss / Test Loss 곡선 시각화를 통한 오버피팅 여부 확인



- Logistic: Train/Test Loss 차이 거의 없음
→ 과적합 없음

- MLP: Train Loss 는 감소하지만 Test Loss 는 정체
→ 약한 오버피팅 발생

- MLP 모델 성능 개선 실험

- hidden layer 크기 증가 (예: 256 → 128)
 - 모델 용량 증가 → 더 복잡한 패턴 학습 가능
- Dropout 추가
 - 오버피팅 방지를 위해 학습 중 일부 뉴런을 무작위 제거
 - 일반화 성능 향상
- Batch Normalization 추가
 - 학습 중 각 층의 출력을 정규화하여 학습을 빠르고 안정적으로 만드는 기법
 - 학습 안정화 및 수렴 속도 향상

- 김사웅

- nn. Module

- 모델 정의: `__init__()`에서 layer 들을 정의
- 순전파 로직 정의: `forward()` 메서드에서 데이터를 어떻게 처리할지 정의
- 파라미터 관리: 학습 가능한 weight 와 bias 등의 파라미터를 자동 추적
- GPU 전송 지원: `.to(device)` 한 줄로 모든 파라미터와 텐서를 GPU 로 전송
- 서브모듈 관리: Layer 를 다른 Module 로 쪼개 관리 가능 (ex: `self.block = MyBlock()`)

- 합성곱층

- 목적: 이미지에서 특징(feature)을 추출

- 동작: 필터(kernel)를 이미지에 슬라이딩하여 곱하고 합함
- 결과: 입력보다 작거나 같은 크기의 feature map 생성

- 풀링층

- 목적: feature map 의 크기를 줄이고, 계산량 감소
- 방식: 대표적으로 Max Pooling 영역내 최댓값 선택

- CNN 구성 이유

- 이미지는 공간 정보가 중요하다 → Conv 로 특징 추출
- 파라미터 수를 줄이기 위해
- MaxPooling 을 통해 대표적인 특징만 추출 + 계산량 줄임 + 과적합 완화
- 고차원 특징을 조합하기 위해 → Fully Connected
- 비선형 함수가를 추가하여 학습을 원활하게 하기 위해 → ReLU 사용

- Adam

- Adam 은 Gradient Descent 를 더 똑똑하게 한 최적화 알고리즘
- 이전 gradient 의 이동 평균 추적 (Momentum)
- gradient 제곱의 이동 평균도 추적 (RMSProp 비슷함)
- 학습률을 적응적으로 조절 (Adaptive Learning Rate)

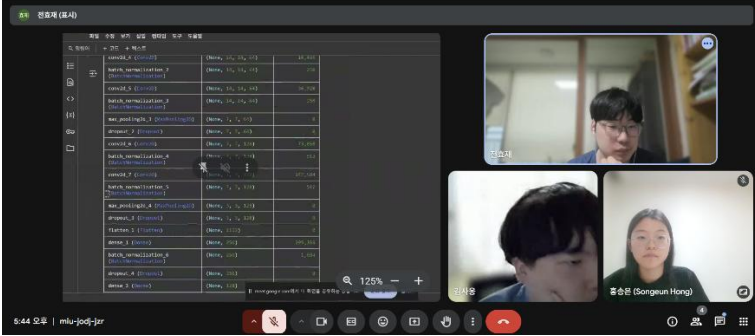
- nn.CrossEntropyLoss()

- 예측값과 정답(label) 사이의 차이를 측정하는 손실 함수(Loss Function)

- 전효재

	<ul style="list-style-type: none"> ○ Torch 와 Keras 둘 다 사용 ○ CNN 모델 사용 <ul style="list-style-type: none"> ▪ 처음엔 2 개의 컨볼루션 블록을 사용 ▪ 정확도를 향상해보기 위해 블록 하나 추가해서 3 개의 컨볼루션 블록으로 모델 구성 ○ 파라미터 <ul style="list-style-type: none"> ▪ 초기엔 에포크 15 회로 구성 ▪ 정확도를 향상해보기 위해 에포크를 50 회로 변경 ○ 과적합 방지 <ul style="list-style-type: none"> ▪ 처음엔 2 개의 컨볼루션 블록을 사용 ▪ Dropout 을 각 층마다 추가 ▪ Earlystop 를 이용해서 개선되지 않을 경우 학습 중지 ▪ ReduceLROnPlateau 를 사용 : 검증 손실이 3 번 연속 개선되지 않으면 학습률 0.5 배 감소 ▪ 모델 체크포인트로 가장 좋은 정확도의 모델을 선택 ○ 결과 <ul style="list-style-type: none"> ▪ 사용 프레임워크 및 변경 전후 정확도 ▪ Keras: 99.19 -> 99.64 ▪ Torch: 99.54 -> 99.58
활동평가	<div>전효재</div> <p>MNIST 를 이용해서 이미지 분류에 가장 많이 사용되는 CNN 을 이용해서 분류해보고 정확도를 추출해봤음. 정확도를 올리기 위해 모델구성이나 학습방식 등을 만져보면서 정확도를 올리긴 했지만 과적합으로 정확도가 오른것으로 예상됨. 과적합 이유로는 이미지 파일의 크기가 작고 단순한 이미지인 것이 이유로 생각됨. 조금 더</p>

		고차원 이미지를 이용한다면 모델 구성이나 다른 부분을 만질방향이 많을 것으로 예상됨.
	홍송은	첫 AI 스터디인 만큼 수업 내용과 연계하여 기본 개념을 탄탄히 다짐. 모델을 직접 설계하고 복잡한 구조를 이해하기 위해, 간단한 모델부터 시작하여 성능 차이를 비교하고 각 레이어의 역할과 텐서 크기 변화를 중심으로 학습함. 또한, loss 시각화를 통해 오버피팅 여부를 확인하고 Dropout, BatchNorm 등 다양한 성능 개선 기법을 실습에 적용함. 기초 모델을 바탕으로 구조적 흐름을 익힘으로써, 이후 더 복잡한 모델 구성에 대한 이해도를 높이고자 함. 향후에는 오버피팅을 중심 주제로 삼아 심도 있게 다루는 것도 의미 있을 것이라 판단함.
	김사웅	cnn 을 클래스를 만들어서 직접 구성해보고 구성한 model 을 epoch 만큼 training 하여 최적화를 하는 일련의 과정을 직접 해볼 수 있었음 . 데이터를 활용하기에 편의성이 좋은 mnist 데이터셋으로 비교적 쉽게 학습을 접근할 수 있었음 이 과정에서 CNN 의 합성곱과 풀링 ,fc 그 내부의 구성을 자세히 학습함으로써 모델 구성이 어떻게 이루어졌는지 확인할 수 있었음. 예측 결과 시각화와 학습한 Loss 를 분석하는 간단한 방법또한 알게되었음
과제	<ul style="list-style-type: none"> • Fashion MNIST 데이터셋의 구조 및 클래스 라벨 이해 • Fashion MNIST 분류 모델 구현 <ul style="list-style-type: none"> ◦ loss, accuracy 곡선 시각화 후 비교 분석 ◦ 오버피팅 여부 확인 후 개선 방안 탐색 ◦ 각 레이어의 역할과 의미를 해석하고, 직접 모델을 설계할 수 있도록 학습 	
향후 계획	<ul style="list-style-type: none"> • Fashion MNIST 데이터셋을 활용한 이미지 분류 실습 진행 • 기본 분류 모델을 구축하고 성능 확인 및 구조별 차이 이해 	

	<ul style="list-style-type: none"> • 학습/테스트 결과 분석으로 오버피팅 여부 확인 • 다양한 성능 개선 기법 적용 	
첨부 자료	스터디 화면 1	
	스터디 화면 2	
	결과물	<p>Github 주소:</p> <p>https://github.com/yellowHSE/ds_rokey4_study_team2/tree/main/codes/week6_MNIST</p> 