GDGoC SCH

심층 신경망 훈련에서
활성화 함수, 가중치 초기화, 배치정규화

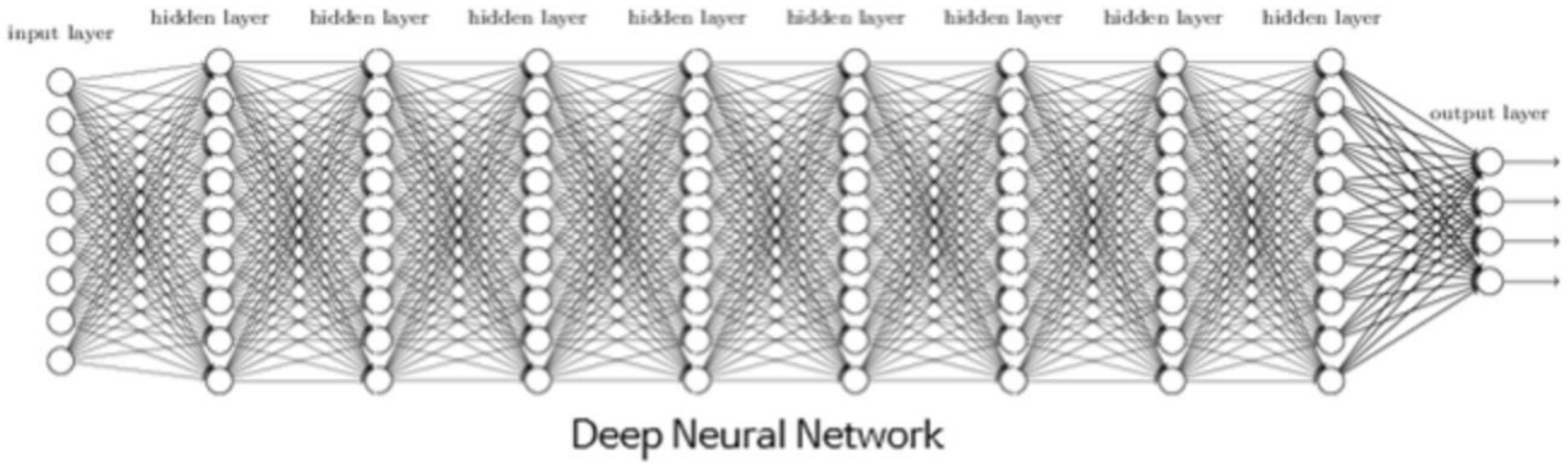
Week4-task1

임상병리학과 20215140 장아영

Contents

01 심층 신경망 훈련

심층 신경망(DNN, Deep Neural Network)



- 여러 층의 뉴런으로 구성된 인공지능 모델
- 복잡한 문제를 해결하기 위해 사용
- 입력층, 여러 개의 은닉층, 출력층
- 많은 데이터를 학습하고 일반화 가능

심층 신경망 훈련 시 발생하는 문제점

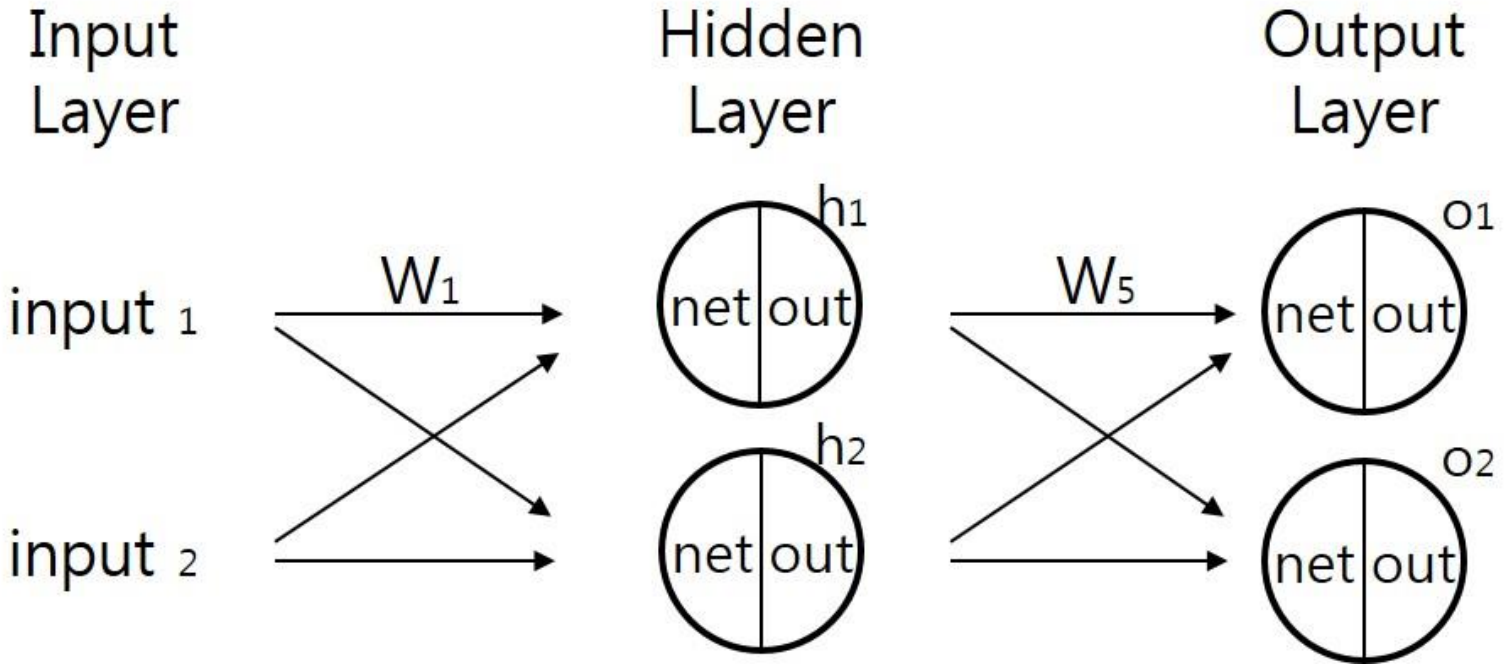
- DNN 모델이 복잡해질수록 hidden layer 개수 증가
- 그래디언트 소실이나 폭주와 같은 문제 발생
- 학습시간 지연
- 오버피팅 위험 증가

02 그래디언트 소실과 폭주

오차역전파 알고리즘

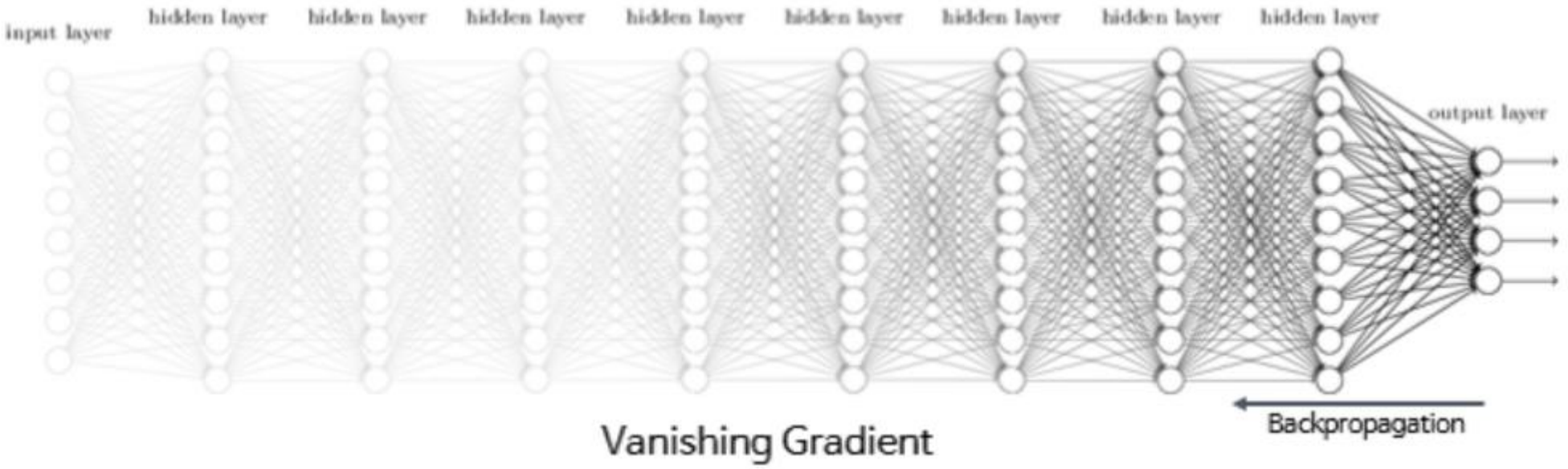
- 출력층에서 입력층으로 오차 그래디언트를 전달하여 각 뉴런의 손실함수의 그래디언트 계산

- 계산된 그래디언트는 경사 하강법 단계에서 가중치 매개변수를 업데이트하는 데 사용



그래디언트 소실

- 역전파 과정에서 그래디언트가 점점 작아져 가중치 매개변수가 업데이트 X
- 가중치 업데이트에 필요했던 기울기 값이 너무 작아지는 현상
- 학습이 매우 느려지거나 멈출 수 있음

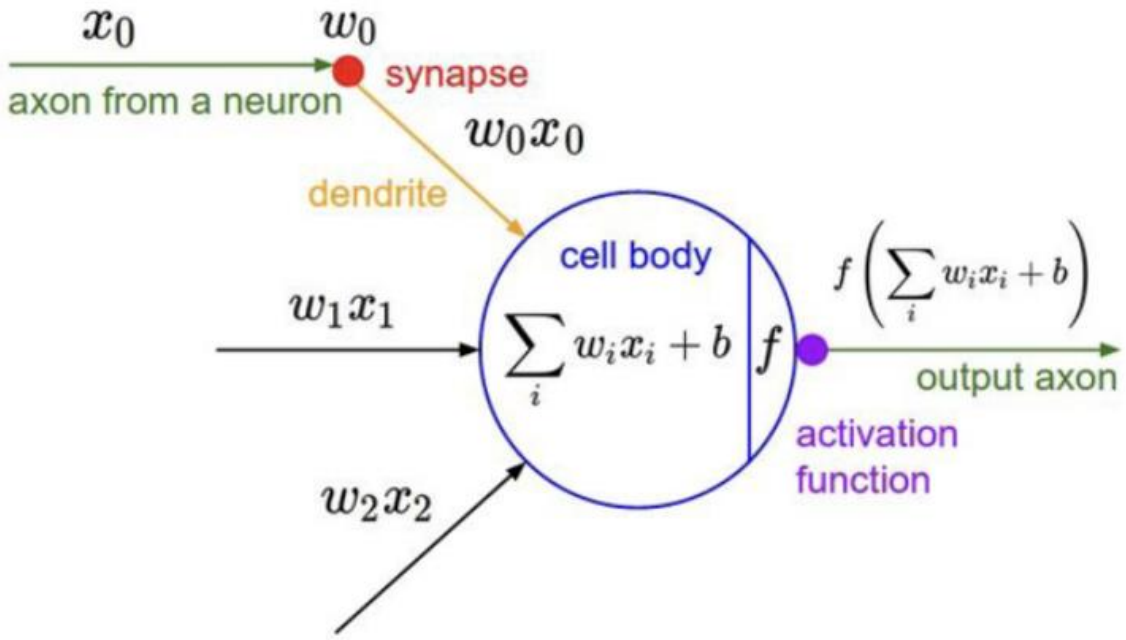


그래디언트 폭주

- 그래디언트가 커지는 현상
- 기울기 값이 너무 커져서 가중치 매개변수가 기하급수적으로 커짐
- 학습이 제대로 이루어지지 않고 모델이 불안정

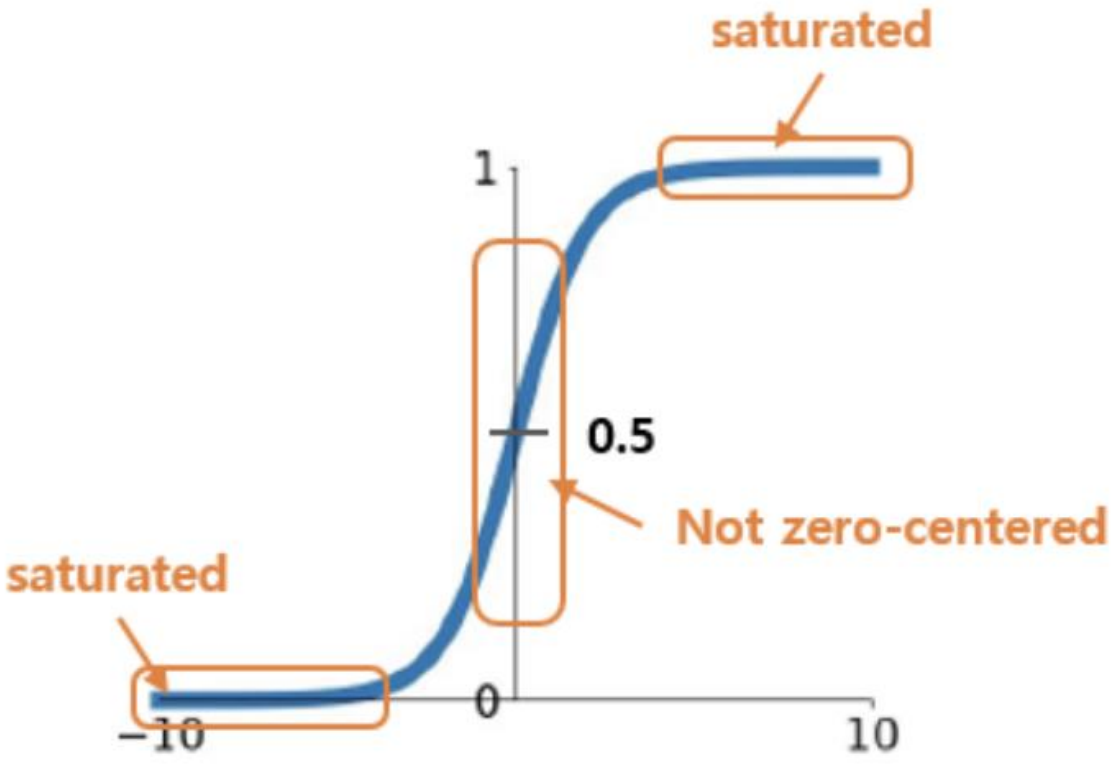
03 활성화함수의 종류와 특징

활성화 함수(activation function)



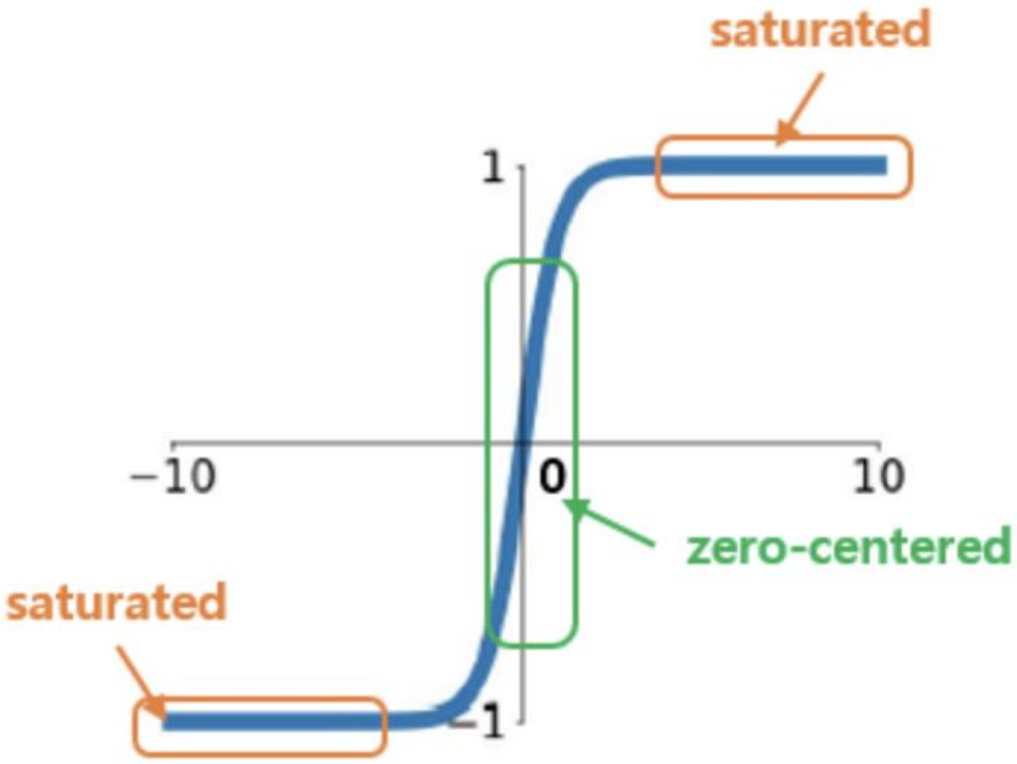
- 신경망의 각 뉴런이 입력 값에 따른 반응 결정
- 입력 신호의 총합을 출력 신호로 변환
- 뉴런이 작동할지 아닌지를 판단
- 예) 시그모이드 함수, 하이퍼볼릭 탄젠트 함수

시그모이드 함수



- 입력 신호의 총합을 0에서 1 사이로 변환
- 입력의 절대값이 크면 그래디언트가 소멸되어 역전파에서 전달 막힘
- 주로 두 가지 상태(켜짐/꺼짐)의 확률을 표현

하이퍼볼릭 탄젠트 함수(tanh)



- 시그모이드 함수의 대체제
- 입력을 -1에서 1 사이의 값으로 변환
- 원점을 중심으로 대칭이 되어 출력이 더 균형 있게 분포
- 입력의 절대값이 클 경우 그래디언트 소멸

ReLU(Rectified Linear Unit) 함수

- 입력이 0 이하일 때 0을 출력

  0 이상일 때는 입력값을 그대로 출력

- 간단하면서도 계산 및 수렴 속도가 빠름

- 수렴하는 구간이 없어서 시그모이드나 tanh에 비해 그래디언트 소실 문제 적음

04 가중치 초기화의 중요성과 방법

가중치 초기화

- 신경망의 학습을 시작하기 전에 가중치 값을 설정하는 과정

- 초기값에 따라 모델의 학습 속도와 성능이 크게 달라짐

- 가중치는 평균이 0이고 표준편차가 0.01인 정규분포를 따르는 값으로 랜덤하게 초기화하는 것이 일반적

* 가중치 초기값이 모두 0이거나 동일: 역전파 단계에서 모든 뉴런이 동일한 그래디언트 값을 가지게 되어 학습이 이루어지지 않음

* 가중치 초기값이 작은 난수: 활성화 함수에 따라 그래디언트 소실 또는 dead ReLU

Xavier 초기화와 He 초기화

- 심층 신경망의 활성화 값을 고르게 분포시키기 위한 방법으로 제안

(1) Xavier 초기화

- 활성화 함수가 선형일 때 효과적

(2) He 초기화

- ReLU와 같은 비선형 활성화 함수에 적합

- 가중치가 더 넓게 분포되도록 한다.

- 그래디언트 소실 감소

05 배치 정규화의 개념과 장점

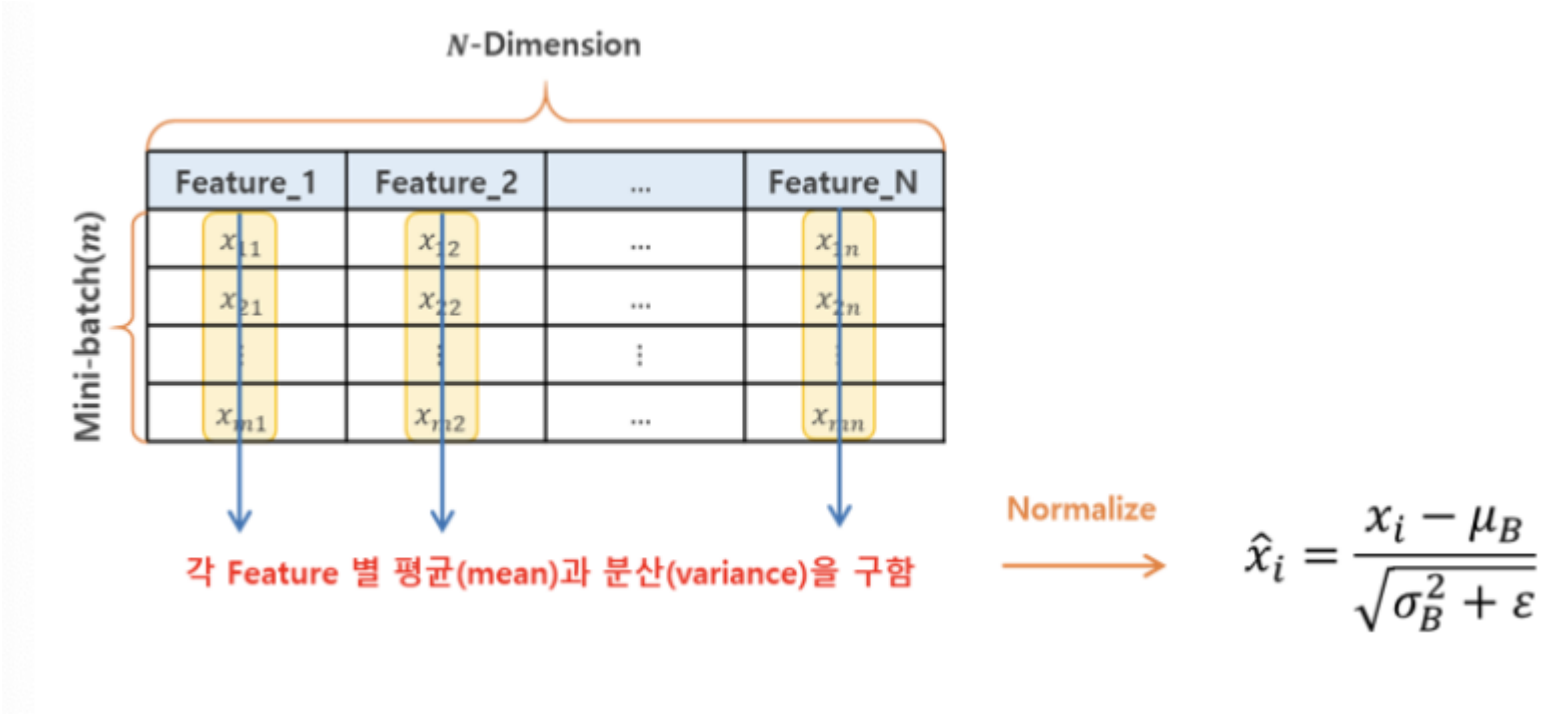
배치 정규화(BN, Batch Normalization)

- 각 층의 활성화 함수출력값이 정규분포를 이루도록 함
- 미니배치단위의 평균과 분산을 구한 후 이를 정규화해 데이터 분포를 평균 0, 분산 1
- 내부의 데이터 분포가 안정적이어서 학습이 빠르고 효율적으로 진행
- 내부 공변량 변화 문제 감소
- 내부 공변량 변화(internal covariate shift): 각 층의 입력 분포가 학습하는 동안 계속 변화
- 이로 인해 학습이 느려질 수 있으며, 배치 정규화가 이를 감소

BN의 장점

- 활성화 함수입력값에 대한 스케일(gamma) 및 이동(beta) 조정을 통해 비선형성 유지

- 그래디언트 소실문제가 감소하고, 가중치 초기화에 덜 민감해지며, 학습률을 크게 잡아도 잘 수렴



Week4-task1

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, ReLU, Dropout

# 배치 정규화를 포함한 딥러닝 모델 정의 함수
def create_model():
    model = Sequential([
        # 첫 번째 Dense 층(유닛 수: 256)과 적절한 입력 형태 추가
        Dense(256, input_shape=(784,)),
        # 첫 번째 Dense 층 다음에 배치 정규화 추가
        BatchNormalization(),
        # 첫 번째 배치 정규화 다음에 ReLU 활성화 함수 추가
        ReLU(),
        # 유닛 수가 128인 Dense 층 추가
        Dense(128),
        # 해당 층에 배치 정규화와 ReLU 추가
        BatchNormalization(),
        ReLU(),
        # 드롭아웃 층(드롭아웃 비율: 0.3) 추가
        Dropout(0.3),
        # 유닛 수가 64인 Dense 층 추가
        Dense(64),
        # 해당 층에 배치 정규화와 ReLU 추가
        BatchNormalization(),
        ReLU(),
        # 유닛 수가 10인 출력층(Dense)과 softmax 활성화 함수 추가
        Dense(10, activation='softmax')
    ])
    return model
```

Week4-task1

```
# MNIST 데이터셋을 로드하고 전처리
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# 데이터 형태를 (784,)로 변환하고, 0~1 범위로 정규화
x_train = x_train.reshape(-1, 784).astype('float32') / 255.0
x_test = x_test.reshape(-1, 784).astype('float32') / 255.0

# Adam 옵티마이저, sparse categorical crossentropy 손실 함수, accuracy 메트릭으로 모델 컴파일
model = create_model()
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# 배치 크기를 64로, 에포크 수를 5로 설정하여 모델 훈련
history = model.fit(x_train, y_train, batch_size=64, epochs=5, validation_split=0.2)

# 모델을 평가하고 테스트 정확도 출력
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

Week4-task1

```
Epoch 1/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - accuracy: 0.8367 - loss: 0.5768 - val_accuracy: 0.9607 - val_loss: 0.1276
Epoch 2/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.9572 - loss: 0.1401 - val_accuracy: 0.9723 - val_loss: 0.0907
Epoch 3/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 0.9681 - loss: 0.0990 - val_accuracy: 0.9716 - val_loss: 0.0909
Epoch 4/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.9742 - loss: 0.0809 - val_accuracy: 0.9754 - val_loss: 0.0798
Epoch 5/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.9810 - loss: 0.0641 - val_accuracy: 0.9749 - val_loss: 0.0879
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9727 - loss: 0.0930
Test Accuracy: 0.9767
```