**DNN**

**목차**

[제 1 장 개요 4](#_Toc66107161)

[1.1 개발 배경 5](#_Toc66107162)

[1.2 기능 설명 5](#_Toc66107163)

[제 2 장 설계 6](#_Toc66107164)

[2.1 요구 사항 7](#_Toc66107165)

[2.1.1 DNN, MLP 공통화 7](#_Toc66107166)

[2.1.2 ECMiner AI 표준화 7](#_Toc66107167)

[2.2 개발 요소 7](#_Toc66107168)

[2.2.1 NeuralNet 클래스 7](#_Toc66107169)

[2.2.2 Layer 클래스 7](#_Toc66107170)

[2.2.3 ActivationFunction 클래스 7](#_Toc66107171)

[2.2.4 Weights 클래스 8](#_Toc66107172)

[2.2.5 Optimizer 클래스 8](#_Toc66107173)

[2.2.6 DataBatchManager 클래스 8](#_Toc66107174)

[2.2.7 LossFunction 클래스 8](#_Toc66107175)

[2.3 Pseudo Code 8](#_Toc66107176)

[2.3.1 신경망 학습 8](#_Toc66107177)

[2.3.2 신경망 계산 9](#_Toc66107178)

[제 3 장 개발 프로세스 11](#_Toc66107179)

[3.1 개발 진행률 12](#_Toc66107180)

[3.2 변경 사항 12](#_Toc66107181)

[3.2.1 설계 변경 사항 12](#_Toc66107182)

[3.2.2 개발 완료 후 변경 사항 13](#_Toc66107183)

[제 4 장 구현 설명 14](#_Toc66107184)

[4.1 Class Diagram 15](#_Toc66107185)

[4.2 Sequence Diagram 16](#_Toc66107186)

[4.3 Pseudo Code Sequence Diagram 비교 17](#_Toc66107187)

[4.3.1 신경망 학습 17](#_Toc66107188)

[4.3.2 신경망 계산 17](#_Toc66107189)

[제 5 장 개발 결과 19](#_Toc66107190)

[5.1 사용 예제 20](#_Toc66107191)

[5.1.1 DNN 학습 사용 예제 20](#_Toc66107192)

# 제 1 장 개요

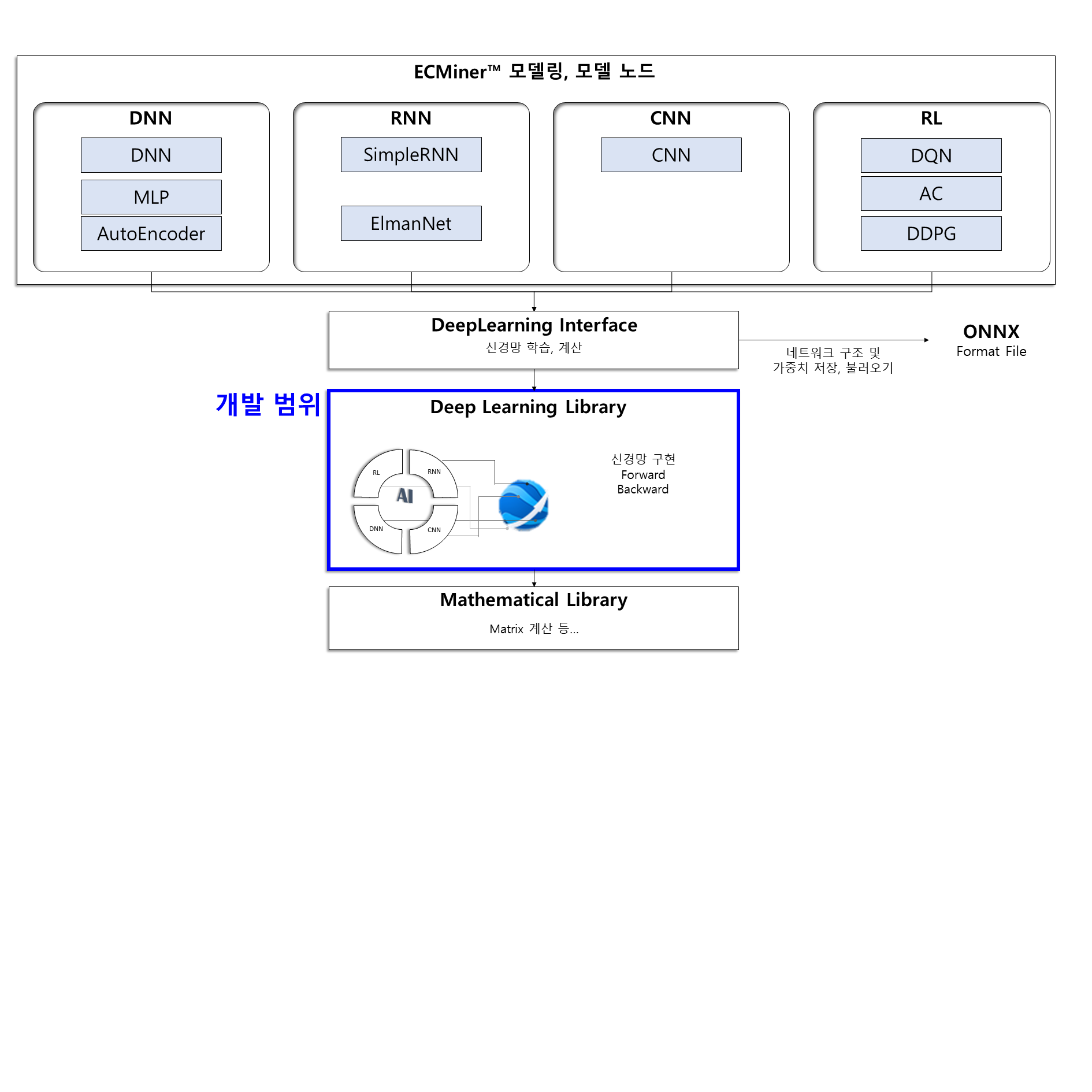
**1.1 개발 배경**

**1.2 기능 설명**

## 1.1 개발 배경

ECMiner™에서 인공신경망을 이용하여 Deep Learning을 수행하는 모델링 노드 및 모델 노드에 대해, 공통적으로 수행하는 기능을 표준화로 현재 개발된 노드들과 앞으로 개발될 노드를 Deep Learning Library를 통해 개발합니다.

## 1.2 기능 설명



ECMiner™ 모델링, 모델 노드 중에 신경망을 이용할 경우, Deep Learning Library를 이용하여 개발할 수 있도록 신경망에 관한 전반적인 기능을 구성합니다.

# 제 2 장 설계

**2.1 요구 사항**

**2.2 개발 요소**

**2.3 Pseudo Code**

## 2.1 요구 사항

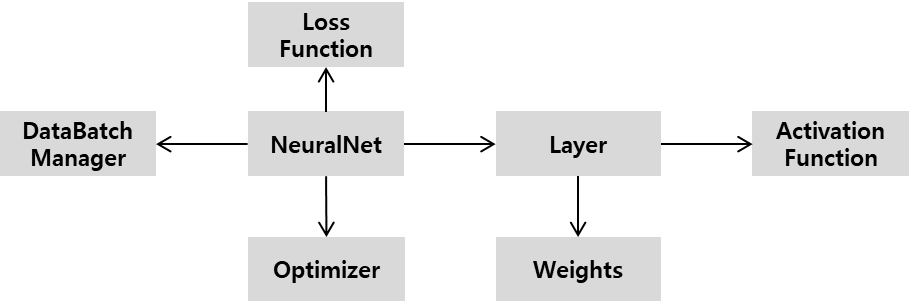
### 2.1.1 DNN, MLP 공통화

DNN 노드를 개발하되, 이 노드가 기존의 DNN 노드와, MLP 노드 기능들을 포함합니다.

### 2.1.2 ECMiner AI 표준화

딥러닝이 사용되는 노드들이 사용할 수 있는 공통된 기능들을 개발합니다.

## 2.2 개발 요소



### 2.2.1 NeuralNet 클래스

개요 : 신경망을 생성 및 관리할 때 기본적으로 접근하는 클래스

기능 : 신경망 계산, 신경망 학습 등 인공신경망에서 사용되는 전반적인 기능

### 2.2.2 Layer 클래스

개요 : 신경망의 각 층의 종류, 노드 개수, 가중치, 활성화 함수 등을 관리하는 클래스

기능 : 층의 종류에 따른 feedforward, backward를 수행

### 2.2.3 ActivationFunction 클래스

개요 : 활성화 및 비활성화를 수행하는 클래스

기능 : 활성화 및 비활성화를 수행

### 2.2.4 Weights 클래스

개요 : 현재 층과 이전 층을 잇는 가중치

기능 : 가중치 값을 저장

### 2.2.5 Optimizer 클래스

개요 : 최적화 기법을 수행하는 클래스

기능 : 최적화 기법 수행

### 2.2.6 DataBatchManager 클래스

개요 : 입력, 출력 데이터를 미니배치로 나누어 주는 클래스

기능 : 데이터를 미니 배치로 나누어 관리

### 2.2.7 LossFunction 클래스

개요 : 목적함수를 계산하는 클래스

기능 : 목적함수를 미분한 값을 계산하여 을 구함

## 2.3 Pseudo Code

### 2.3.1 신경망 학습

B : the number of total mini Batches

N : minibatch size

M : the number of total Layer

α : learning rate

|  |
| --- |
| **Train** |
| **for** epoch = 1 to MaxEpoch step +1 **do** |
| **for** b = 1 to B step +1 **do**  set input, output\_target |
| **for** n = 1 to N step +1 **do** |
| **for** m = 1 to M step +1 **do** |
|  |
| **end**    **for** m = M to 1 step -1 **do** |
| / N |
| **end** |
| **end** |
| 𝐨𝐩𝐭\_𝐠= optimize(𝐠)  init |
| **end** |
| **end** |

### 2.3.2 신경망 계산

N : Data size

M : the number of total Layer

|  |
| --- |
| **Calculate Predicted** |
| **for** n=1 to N step +1 **do**  set input  **for** m=1 to M Step +1 **do**    **end**  **end** |

# 제 3 장 개발 프로세스

**3.1 개발 진행률**

**3.2 변경 사항**

## 3.1 개발 진행률

설계 : 2020-12-28 ~ 2021-01-15

개발 : 2021-01-18 ~

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **내용** | **현재**  **진척도** | **시작** | **완료** | **담당** |
| network generation | 신경망 구성 기능 | 100% |  | 2021-  01-22 | 이민곤 |
| Initialize | 각 층, 가중치 초기화 | 100% |  | 2021-  02-04 | 이민곤 |
| data management | 배치 불러오기 | 100% |  | 2021-  02-05 | 박우진 |
| loss | 손실함수 | 100% |  | 2021-  01-19 | 최환민 |
| forward | 신경망 계산 | 100% |  | 2021-  01-21 | 최환민 |
| backward | 신경망 학습 | 100% |  | 2021-  02-08 | 최환민,  이민곤 |
| update | deltaWeight 업데이트, Weight 업데이트 | 100% |  | 2021-  02-08 | 최환민 |
| 속성 값 불러오기 | UI로부터 속성값 불러오기 |  |  |  | 박우진 |
| 모델(다이아몬드) 생성 | 학습된 결과를 모델로 저장 |  |  |  | 박우진 |
| 학습된 모델 계산 | 모델로부터 계산 |  |  |  | 박우진 |
| 메시지 출력 | 학습 진행, 예외 처리 등 메시지 출력 |  |  |  | 박우진 |
| Serialize | 프로젝트 저장 등 저장요소 |  |  |  | 박우진 |
| 전체 |  |  |  |  |  |

## 3.2 변경 사항

### 3.2.1 설계 변경 사항

### 3.2.2 개발 완료 후 변경 사항

1) 오류수정 :Weights::randomizeWeights 랜덤부여 수정

- 문제점 : 가중치가 신경망을 생성할 때 마다 랜덤 값을 가지지만, 모든 가중치가 동일한 값을 가짐

- 개선 내용 : ECMiner #4771 (2021-02-26), ECMiner #4774 (2021-02-26)

2) 오류수정 : ADAM 초기화 수정

- 문제점 : ADAM의 beta1\_pow, beta2\_pow 값이 초기화 되지 않음

- 개선 내용 : ECMiner #4774 (2021-02-26)

3) 오류수정 : 최적화 기법 수정

- 문제점 : tmpDelta1, tmpDelta2 변수가 누적된 값을 사용하지 않고, 항상 초기화 됨

- 개선 내용 : ECMiner #4780 (2021-03-04)

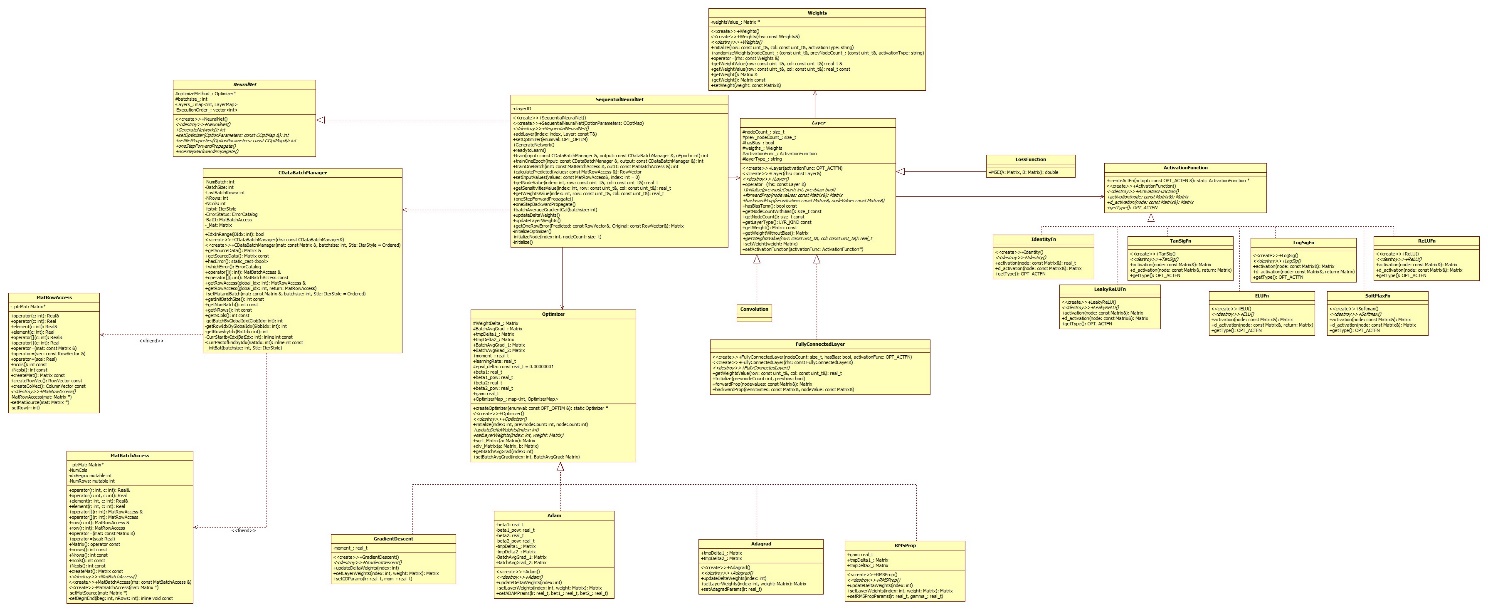
# 제 4 장 구현 설명

**4.1 Class Diagram**

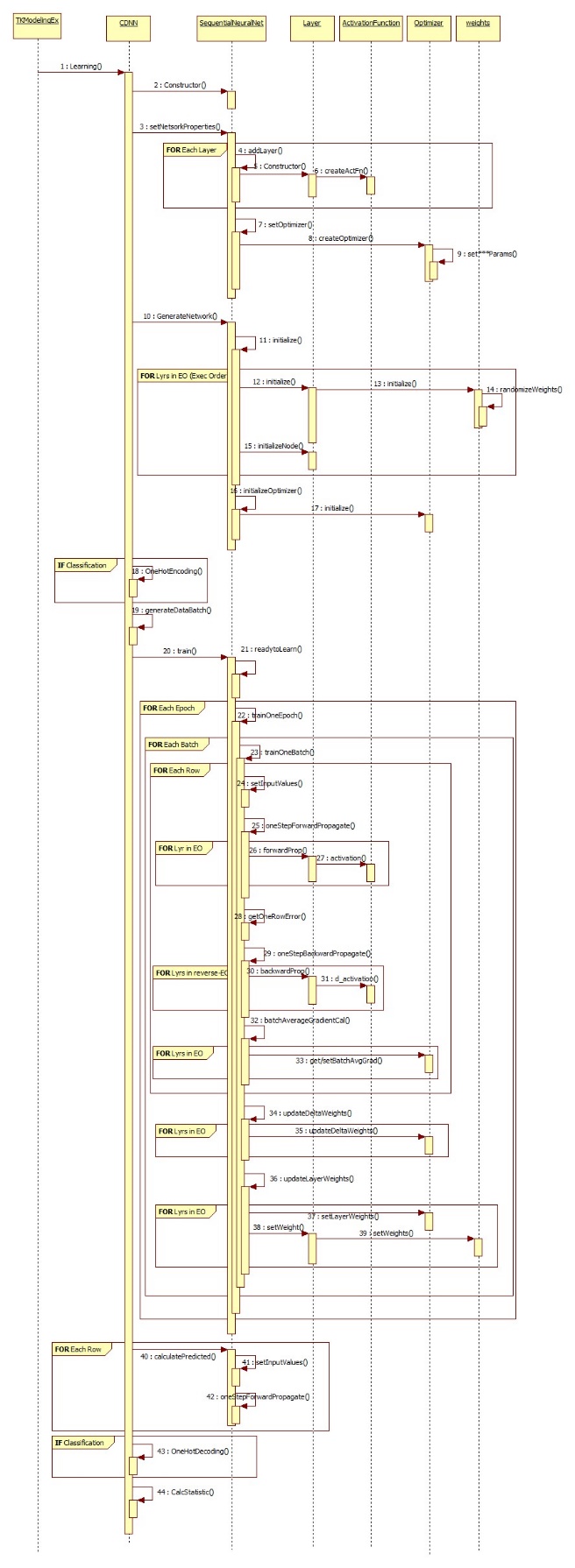
**4.2 Sequence Diagram**

**4.3 Pseudo Code Sequence Diagram 비교**

## 4.1 Class Diagram

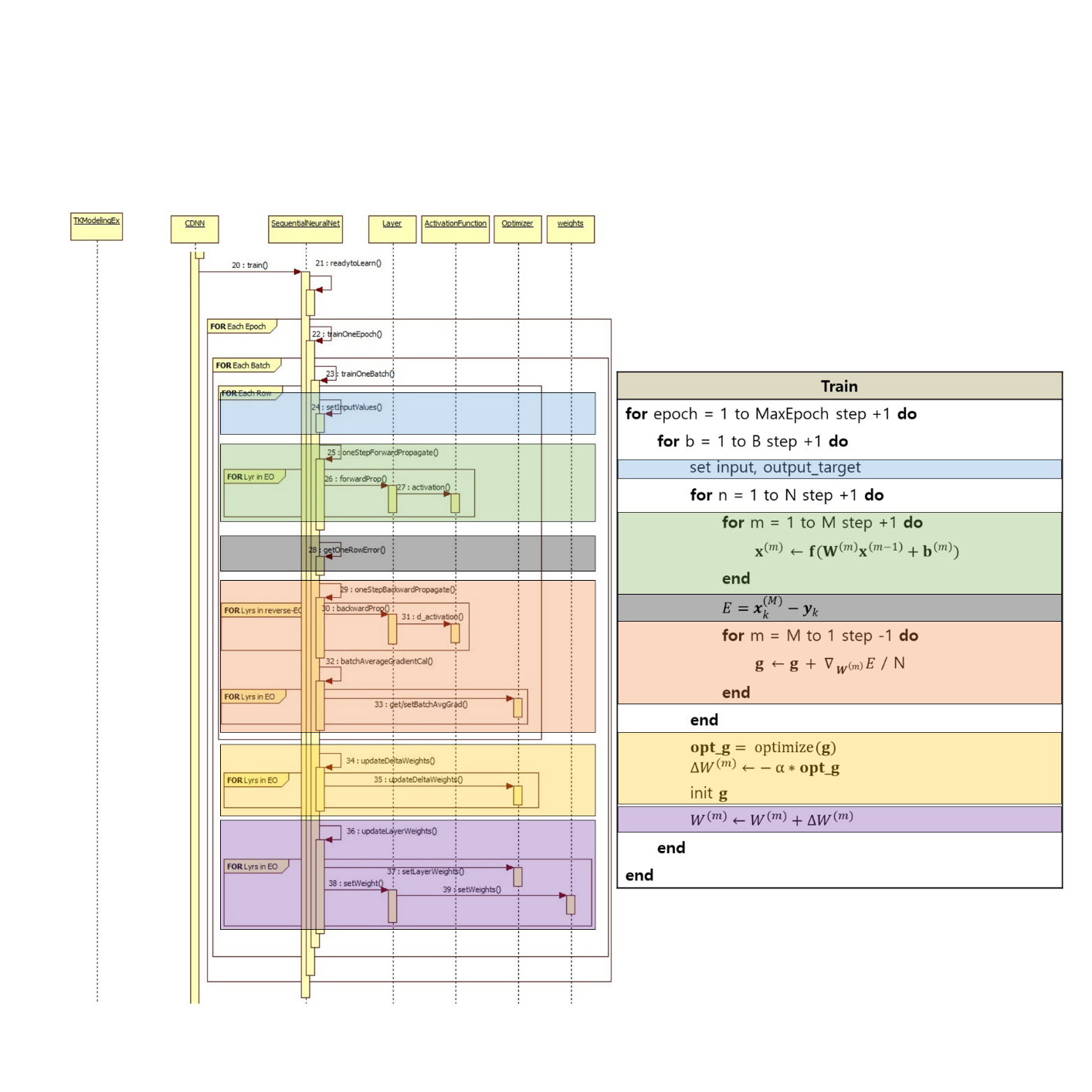
[](그림/Class.jpg)

## 4.2 Sequence Diagram

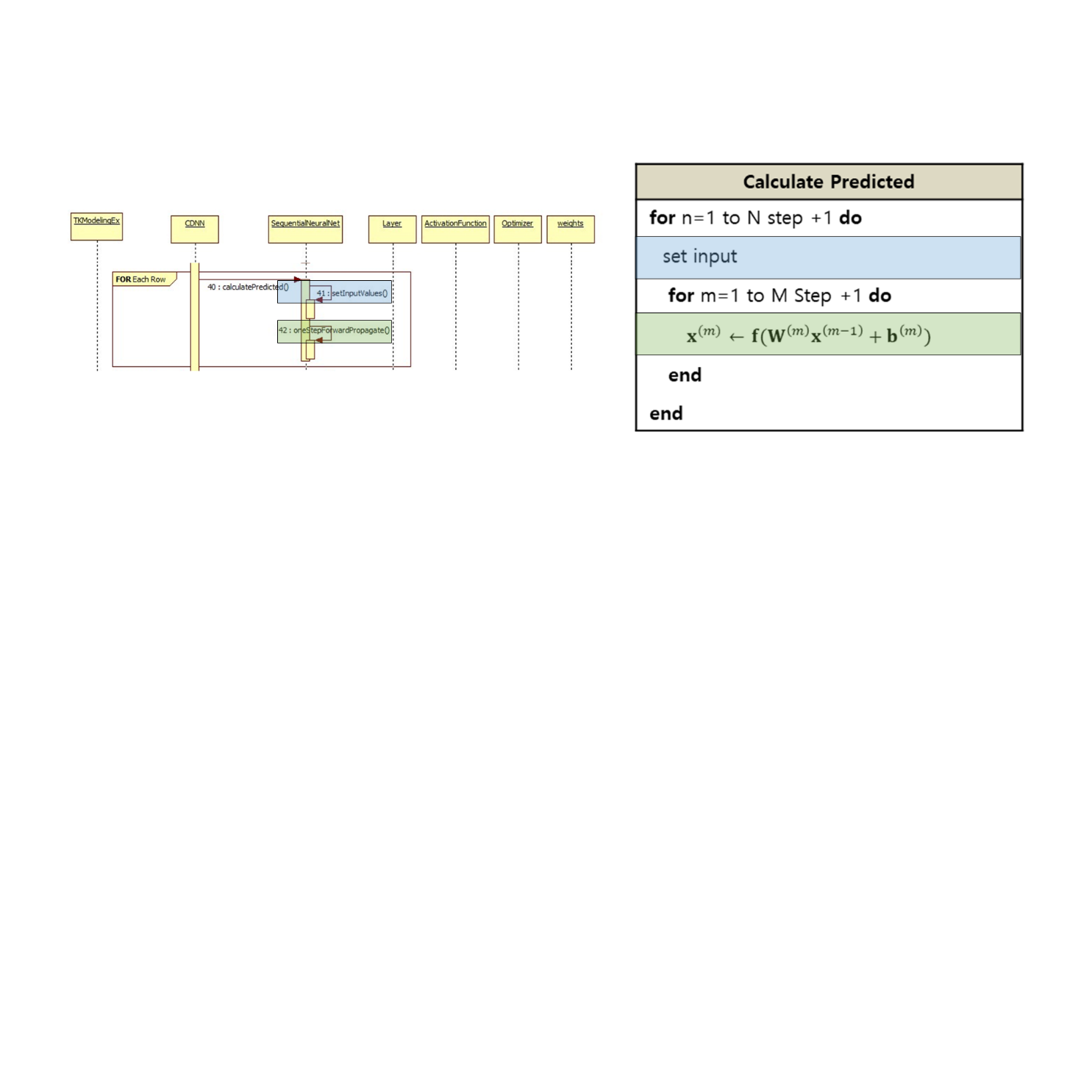
[](그림/Seq_DNN.jpg)

## 4.3 Pseudo Code Sequence Diagram 비교

### 4.3.1 신경망 학습

[](그림/Seq_compare_train.png)

### 4.3.2 신경망 계산

[](그림/Seq_compare_test.png)

# 제 5 장 개발 결과

**5.1 사용 예제**

## 5.1 사용 예제

### 5.1.1 DNN 학습 사용 예제

ECMDL::SequentialNeuralNet Net;

// 속성 값 설정 및 예외 처리

int bStop = Net.setNetProperties(OptionParameters);

if (bStop)

throw TExceptions("Neural Network 속성 설정에 실패하였습니다");

// 신경망 생성

Net.GenerateNetwork();

// 종속변수 크기 설정

int nRowSizeY = MY.Nrows();

int nColSizeY = MY.Ncols();

// 기본 알고리즘 실행

const auto & leanOpt = OptionParameters.at("learning").GetMap();

int nBatch = leanOpt.at("batch").GetInt();

// 출력 목표 onehot encodeing 설정

Matrix OutMat;

if (m\_bIsClassification)

OutMat = OneHotEncoding(MY, m\_nNoClass);

else

OutMat = MY;

// 입력 데이터와 배치 사이즈 설정

ECMDL::CDataBatchManager InpBat(MX, nBatch);

// 출력 목표 데이터와 배치 사이즈 설정

ECMDL::CDataBatchManager OutBat(OutMat, nBatch);

// 예외 처리

int ret = checkBatSizes(InpBat, OutBat);

if (ret != 0)

throw Exception("입력 데이터에 배치 생성에 이상이 있습니다.");

// 학습 수행

Net.train(InpBat, OutBat, epochMax\_);