프로젝트 보고서

시계열 데이터 분석

국민대학교 소프트웨어융합대학원
인공지능전공

최 현 규

국민대학교 소프트웨어 융합대학원 인공지능전공 최현규 K2020515

1. Time Series Analysis

1-1. 시계열 - TimeSeries

- 시계열(time series)란 일정 시간 간격(연도별, 분기별, 월별, 일별, 시간별 등)으로 시간의 경과(흐름)에 따라 관측되는 자료를 말한다. 시계열들은 생성되는 특성에 따라 연속적으로 생성되는 연속시계열과 이산적 시점에서 생성되는 이산시계열로 구분할 수 있다.

- 시계열 분석(time series analysis)라고 하는 것은 시계열을 해석하고 이해하는 데 쓰이는 여러가지 방법을 연구하는 분야를 말하며, 시계열 예측(time series prediction)이라고 하는 것은 주어진 시계열에 대해 수학적인 모델을 만들고, 과거 시계열이 미래에 일어날 사건을 예측하는 것을 말한다.

- 딥러닝 연구분야에서 가장 기본적으로 시도되는 분야가 시계열 분석(time series analysis)이며 데이터의 양이 이미지, 음성, 영상 데이터의 양에 비해 상대적으로 작기 때문이다.

1-2. 시계열 분석 기법

- AR(AutoreRression) 모델: 자기 회귀 모델이라고 불리는 모델이며, 이전의 과거 데이터를 사용하여 회귀한다. 과거의 데이터가 미래의 데이터에 영향을 준다는 점에서 RNN(Recurrent Neural Network)와 비슷하다.

- MA(Moving Average) 모델: 이동 평균 모델이라고 불리는 모델이며, 트렌드(Trend, 평균 혹은 시계열 그래프에서 y값)가 변화하는 상황에서 적합한 회귀 모델이다.

- ARMA(AutoRegression Moving Average) 모델: AR 모델과 MA 모델을 결합한 모델이다.

- ARIMA(AutoRegression Integrated Moving Average): ARMA에서 Integrated 개념을 추가한 모델이다. 불규칙한 시계열 데이터(소량의 데이터)를 분석하는 모델이다.

- Deep Learning(LSTM): LSTM은 RNN의 변형된 종류로, 시퀀스가 긴 데이터를 분석할 때 매우 용이한 방법이다.

2. Pytorch

2-1. Pytorch란?

- Pytorch는 2017년에 공개된 딥러닝 프레임워크로 개발자, 연구자들이 GPU를 활용하여 인공 신경망을 만들고 딥러닝 연구를 용이하게 할 수 있게 만든 Python 기반 라이브러리이다.

- Pytorch는 Facebook의 인공지능 연구부서에서 관리하며 독자적으로 운영되는 Pytorch Forum은 사람들이 질문을 던지면 여러 사용자는 물론 연구부서에서도 답글을 게시할 만큼 활발하게 운영되고 있다.

2-2. Pytorch와 다른 FrameWork의 비교

- Pytorch는 기본적으로 Numpy를 사용한다. x, y, z 세 변수에 대해 학습하는 과정을 예로 들면, 신경망 학습을 할 때 기울기를 계산하기 위해서는 연산 그래프를 따라서 미분해야한다. Numpy만을 사용한다면 모든 미분 식을 직접 계산하고 코드로 작성해야 하므로 변수 하나당 두 줄씩 필요하게 되므로 총 여섯 줄이 필요하게 된다. 반면 Pytorch를 사용하게 되면 backward() 함수를 통해 자동으로 연산이 가능하기 때문에 간편하다는 장점이 있다.

- Numpy를 사용하는 것 이외에 또 다른 장점은 GPU의 사용여부에 있다. Numpy만으로는 GPU로 값을 내보내고 다시 돌려받는 것이 불가능하다. 반면 Pytorch는 CUDA, cuDNN이라는 API를 통해 GPU 연산을 할 수 있다. cuDNN은 CUDA를 이용해 딥러닝 연산을 가속해주는 라이브러리이다. 병렬 연산에서 GPU의 속도는 CPU 속도보다 월등히 빠르며 CUDA와 cuDNN을 동시에 사용할 경우 연산 속도가 CPU의 15배 이상이 된다고 알려져 있다. 심층 신경망을 정의할 때 함수와 기울기 계산 그리고 GPU를 이용한 연산 가속 등의 장점이 우수하기 때문에 딥러닝 연구 시 GPU의 사용 여부는 필수적이라고 할 수 있다.

- Pytorch와 Tensorflow는 모두 GPU를 사용하는 딥러닝 프레임워크이다. 차이점이 있다면 Pytorch는 그래프를 만듦과 동시에 값이 할당되는 'Define by Run' 방식이고, Tensorflow는 연산 그래프를 먼저 만들고 실제 연산할 때 값을 전달하여 결과를 얻는 'Define and Run' 방식이다.

  Pytorch : Define by Run | Tensorflow : Define and Run

- 주로 연구목적으로는 Pytorch, 현업에서는 Tensorflow를 주로 사용하고 있다고 알려져있다.

출처: https://jfun.tistory.com/238

3 신경망 모델의 종류

3-1 RNN

- 가장 기본적인 MLP 모델의 경우 전부 은닉층에서 활성화 함수(activation function)를 지난 값은 오직 출력층 방향으로만 향한다. 이와 같은 신경망들을 피드 포워드 신경망(Feed Forward Neural Network)라고 한다. 이와 별개인 모델 중에서 RNN이라는 모델이 있다.

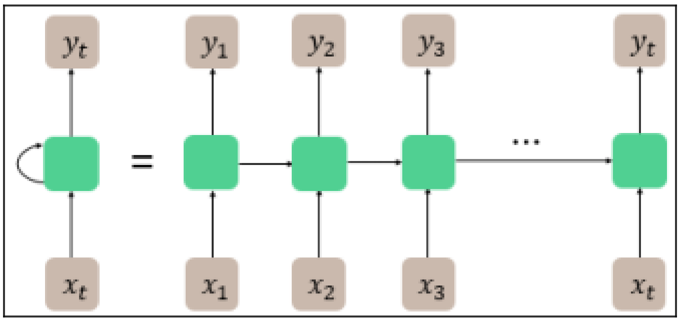
- RNN(Recurrent Neural Network)은 시퀀스(Sequence) 모델이다. 입력과 출력을 시퀀스 단위로 처리하는 모델이며, 번역을 예로 들었을 때 입력은 번역하고자 하는 문장(시퀀스)이다. 출력에 해당되는 번역된 문장 또한 단어 시퀀스이다. 이러한 시퀀스를 처리하기 위해 고안된 모델을 시퀀스 모델이라고 한다. 그 중에서 RNN은 딥러닝에 있어서 가장 기본적인 시퀀스 모델이다.

- RNN은 은닉층의 노드에서 활성화 함수를 통해 나온 결과값을 출력층 방향으로 보내면서 은닉층 노드의 다음 입력으로 전달하는 특징이 있다. 이를 그림으로 표현한 것이 아래와 같다. x는 입력층의 입력 벡터, y는 출력층의 출력 벡터이다(편향 b는 그림에서 생략되어 있음). RNN은 은닉층에서 활성화 함수를 통해 결과를 내보내는 역할을 하는 노드를 셀(Cell)이라고 한다. 셀은 과거의 값을 기억하는 일종의 메모리 역할을 수항하기 때문에 메모리 셀 또는 RNN 셀이라고 표현된다.

- 피드 포워드 신경망에서는 뉴런이라는 단위를 주로 사용하지만 RNN에서는 뉴런이라는 단위보다는 입력층과 출력층에서는 입력 벡터와 출력벡터, 은닉층에서는 은닉 상태라는 표현을 주로 사용한다.

- RNN은 입력과 출력의 길이를 다르게 설계할 수 있으므로 다양한 용도로 사용될 수 있다. 하나의 입력에 대해서 여러 개의 출력(one-to-many)의 모델은 하나의 이미지 입력에 대해서 사진의 제목을 출력하는 이미지 캡셔닝(Image Captioning) 작업에 사용할 수 있다. 또한 단어 시퀀스에 대해서 하나의 출력(may-to-one)의 모델은 입력 문서가 긍정적인지 부정적인지를 판별하는 감성 분류(sentiment classification) 등에 사용될 수 있다. 다 대 다(many-to-many)의 모델의 경우에는 입력 문장으로 부터 대답 문장을 출력하는 챗봇과 입력 문장으로부터 번역된 문장을 출력하는 번역기, 개체명 인식이나 품사 태깅과 같은 작업들에 사용될 수 있다.

In [1]:
```
# RNN
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('./data/image/RNN.png', cv2.IMREAD_UNCHANGED)
plt.imshow(image)
plt.xticks([])
plt.yticks([])
plt.show()
```



3-2 LSTM

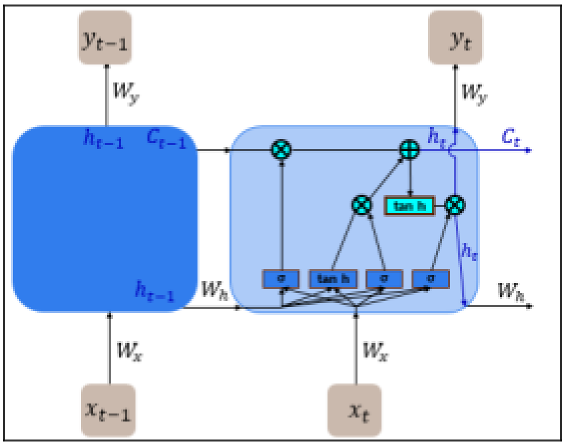
- RNN은 기본적으로 출력이 이전의 계산 결과에 의존하게 된다. 다시 말하면, RNN의 시점(time step)이 길어질 수록 과거의 정보가 미래로 충분히 전달되지 못하는 현상이 발생한다는 것이다. 이를 장기 의존성 문제(the problem of Long-Term Dependencies)라고 한다. 이러한 장기 의존성 문제를 해결하기 위해 RNN의 단점을 보완한 장단기 메모리(Long Short Term Memory)-RNN이 개발되었으며 이를 LSTM이라고 부른다. LSTM은 은닉층의 메모리 셀에 입력 게이트, 망각 게이트, 출력 게이트를 추가하여 불필요한 메모리를 지우고 기억해야할 것을 정한다.

- $h_t = tanh(W_x x_t + W_h h_(t-1) + b)$

- 입력, 출력과 셀을 수식으로 표현하면 위와 같다. x_t와 $h_(t-1)$이라는 두 개의 입력이 각각의 가중치와 곱해진 후 메모리의 입력이 된다. 이 값은 은닉층의 출력인 은닉 상태가 된다. 아래의 그림은 LSTM 내부의 전체적인 그림이다. LSTM은 은닉 상태(hidden state)를 계산하는 방법이 기존의 RNN보다 조금 더 복잡하며 셀 상태(cell state)라는 값이 추가되었다. 아래의 그림에서는 t 시점의 셀 상태를 C_t로 표현한다. LSTM은 기존의 RNN과 비교하여 긴 시퀀스(Long Sequence)의 입력을 계산하는 방법에 있어서 탁월한 성능을 보인다.

In [2]:
```
import cv2
import matplotlib.pyplot as plt
```

```
image = cv2.imread('./data/image/LSTM.png', cv2.IMREAD_UNCHANGED)
plt.imshow(image)
plt.xticks([])
plt.yticks([])
plt.show()
```



4. scikit-learn

- scikit-learn이란 python을 대표하는 머신러닝 라이브러리로 '사이킷런'이라고 부르기도 한다. 기본적으로 오픈소스이며 개인, 비즈니스, 등 관계없이 누구나 무료로 사용가능하다. 현재도 개발과 업데이트가 활발하게 이루어지고 있으며 많은 사람들이 이용하기 때문에 관련 정보도 얻기가 용이하다.
- 현재 프로젝트에서는 Pytorch를 이용하여 직접 구현하므로 scikit-learn에는 많은 함수 기능이 있지만 preprocessing 부분에서 scaler 부분만 다루도록 한다.

Code

In [3]:
```
'''
필요한 모듈 임포트
'''
from sklearn.preprocessing import MaxAbsScaler
from torch.utils.data import TensorDataset, DataLoader

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

In [4]:
```
'''
데이터 불러오기

데이터는 기상청 기상자료개방포털에서
서울지역의 1954년부터 2020년까지의 데이터를
일 단위로 다운로드했다

연도 부분에서 1900년도와 2000년도를 구분하기 위해
```

```
천의 자리, 백의 자리 / 십의 자리 일의 자리를 구분하여
각각 frontyear, backyear로 column 이름을 지정했다
'''
original_data_df = pd.read_csv('./data/original_data.csv')
```

In [5]:
```
original_data_df.head()
```

Out[5]:

	frontyear	backyear	month	day	location	temp_avg	temp_min	temp_max
0	19	54	1	18	seoul	1.4	-2.2	4.8
1	19	54	1	19	seoul	1.5	-1.3	5.8
2	19	54	1	20	seoul	4.7	-1.5	10.7
3	19	54	1	21	seoul	2.8	0.5	5.5
4	19	54	1	22	seoul	-2.1	-6.4	1.6

In [6]:
```
original_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24336 entries, 0 to 24335
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   frontyear  24336 non-null  int64
 1   backyear   24336 non-null  int64
 2   month      24336 non-null  int64
 3   day        24336 non-null  int64
 4   location   24336 non-null  object
 5   temp_avg   24336 non-null  float64
 6   temp_min   24335 non-null  float64
 7   temp_max   24334 non-null  float64
dtypes: float64(3), int64(4), object(1)
memory usage: 1.5+ MB
```

In [7]:
```
'''
해당 기상자료 데이터는 서울지역에 국한된 데이터이기 때문에
시계열 분석에서 의미가 없는 데이터이므로 삭제한다
'''
data_df = original_data_df.drop(['location'], axis = 1)
```

In [8]:
```
data_df.head()
```

Out[8]:

	frontyear	backyear	month	day	temp_avg	temp_min	temp_max
0	19	54	1	18	1.4	-2.2	4.8
1	19	54	1	19	1.5	-1.3	5.8
2	19	54	1	20	4.7	-1.5	10.7
3	19	54	1	21	2.8	0.5	5.5
4	19	54	1	22	-2.1	-6.4	1.6

In [9]:
```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24336 entries, 0 to 24335
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   frontyear  24336 non-null  int64
 1   backyear   24336 non-null  int64
```

```
 2   month      24336 non-null  int64
 3   day        24336 non-null  int64
 4   temp_avg   24336 non-null  float64
 5   temp_min   24335 non-null  float64
 6   temp_max   24334 non-null  float64
dtypes: float64(3), int64(4)
memory usage: 1.3 MB
```

In [10]:

```
'''
전처리, 신경망 학습 과정에서 결측치가 존재하면 오류가 생기기 때문에
확인한 후 다른 값으로 변경하거나 해당 행을 제거한다
'''
data_df.isnull().sum()
```

Out[10]:

```
frontyear    0
backyear     0
month        0
day          0
temp_avg     0
temp_min     1
temp_max     2
dtype: int64
```

In [11]:

```
'''
결측치는 시계열데이터이기 때문에 바로 앞과 뒤의 데이터는
크게 차이가 나지 않는다는 점을 생각해서 결측치를 바로 앞 데이터로 대체한다
'''
data_df = data_df.fillna(method = 'pad')
```

In [12]:

```
data_df.isnull().sum()
```

Out[12]:

```
frontyear    0
backyear     0
month        0
day          0
temp_avg     0
temp_min     0
temp_max     0
dtype: int64
```

In [13]:

```
'''
데이터프레임의 데이터를 수치적으로 계산하기 위해 numerical data로 변경한다
'''
data_df = data_df.apply(pd.to_numeric)
```

In [14]:

```
'''
각 계절을 구분하고 가중치를 두기 위해서 각 계절 별 데이터를 추가한다

Month column에 있는 데이터를 별도로 데이터프레임으로 저장한다

Spring: 3 ~ 5 - 0.05
Summer: 6 ~ 8 - 0.25
Fall: 9 ~ 11 - 0.42
Winter: 12 ~ 2 - 0.28
'''
season_data_df = pd.DataFrame(data_df['month'])
```

In [15]:

```
season_data_df.head()
```

Out[15]:

	month
0	1
1	1

	month
2	1
3	1
4	1

In [16]:
```
'''
column의 이름을 season으로 변경한다
'''
season_data_df.columns = ['season']
```

In [17]:
```
season_data_df.head()
```

Out[17]:

	season
0	1
1	1
2	1
3	1
4	1

In [18]:
```
'''
기존의 데이터프레임과 별도로 구분한 season column 데이터프레임을 결합한다
'''
new_data_df = pd.concat([data_df, season_data_df], axis = 1)
```

In [19]:
```
new_data_df.head()
```

Out[19]:

	frontyear	backyear	month	day	temp_avg	temp_min	temp_max	season
0	19	54	1	18	1.4	-2.2	4.8	1
1	19	54	1	19	1.5	-1.3	5.8	1
2	19	54	1	20	4.7	-1.5	10.7	1
3	19	54	1	21	2.8	0.5	5.5	1
4	19	54	1	22	-2.1	-6.4	1.6	1

In [20]:
```
new_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24336 entries, 0 to 24335
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   frontyear  24336 non-null  int64  
 1   backyear   24336 non-null  int64  
 2   month      24336 non-null  int64  
 3   day        24336 non-null  int64  
 4   temp_avg   24336 non-null  float64
 5   temp_min   24336 non-null  float64
 6   temp_max   24336 non-null  float64
 7   season     24336 non-null  int64  
dtypes: float64(3), int64(5)
memory usage: 1.5 MB
```

In [21]:
```
'''
계절 별 데이터를 정해진 수치별로 클러스터링한다
'''
for season in new_data_df:
    new_data_df.loc[(new_data_df['season'] >= 1) & (new_data_df['season'] < 3), 'seas
    new_data_df.loc[(new_data_df['season'] >= 3) & (new_data_df['season'] < 6), 'seas
    new_data_df.loc[(new_data_df['season'] >= 6) & (new_data_df['season'] < 9), 'seas
    new_data_df.loc[(new_data_df['season'] >= 9) & (new_data_df['season'] < 12), 'sea
    new_data_df.loc[(new_data_df['season'] >= 12), 'season'] = 0.05
```

In [22]:
```
new_data_df.head()
```

Out[22]:

	frontyear	backyear	month	day	temp_avg	temp_min	temp_max	season
0	19	54	1	18	1.4	-2.2	4.8	0.05
1	19	54	1	19	1.5	-1.3	5.8	0.05
2	19	54	1	20	4.7	-1.5	10.7	0.05
3	19	54	1	21	2.8	0.5	5.5	0.05
4	19	54	1	22	-2.1	-6.4	1.6	0.05

In [23]:
```
'''
scikit-learn의 MaxAbsScaler() 함수를 scaler 변수로 저장한다
'''
scaler = MaxAbsScaler()
```

In [24]:
```
'''
scalering을 진행하게되면 데이터프레임의 column 이름이 초기화되기 때문에 별도로 다시 이
'''
scaled_data_df = pd.DataFrame(scaler.fit_transform(new_data_df))
scaled_data_df.columns = ['frontyear', 'backyear', 'month', 'day', 'temp_avg', 'temp_
```

In [25]:
```
scaled_data_df.head()
```

Out[25]:

	frontyear	backyear	month	day	temp_avg	temp_min	temp_max	season
0	0.95	0.545455	0.083333	0.580645	0.041543	-0.072607	0.121212	0.119048
1	0.95	0.545455	0.083333	0.612903	0.044510	-0.042904	0.146465	0.119048
2	0.95	0.545455	0.083333	0.645161	0.139466	-0.049505	0.270202	0.119048
3	0.95	0.545455	0.083333	0.677419	0.083086	0.016502	0.138889	0.119048
4	0.95	0.545455	0.083333	0.709677	-0.062315	-0.211221	0.040404	0.119048

In [26]:
```
scaled_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24336 entries, 0 to 24335
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   frontyear  24336 non-null  float64
 1   backyear   24336 non-null  float64
 2   month      24336 non-null  float64
 3   day        24336 non-null  float64
 4   temp_avg   24336 non-null  float64
 5   temp_min   24336 non-null  float64
 6   temp_max   24336 non-null  float64
 7   season     24336 non-null  float64
```

```
dtypes: float64(8)
memory usage: 1.5 MB
```

In [27]:

```
'''
train, validation, test data를 분리한다
'''
total_size = int(len(scaled_data_df))

train_data_df = scaled_data_df[0:int(total_size * 0.7)].reset_index(drop = True)
validation_data_df = scaled_data_df[:int(total_size * 0.15):].reset_index(drop = True
test_data_df = scaled_data_df[int(total_size * 0.85):].reset_index(drop = True)
```

In [28]:

```
'''
분리된 train, validation, test data의 크기를 확인한다
'''
print("Total Length: {}".format(len(scaled_data_df)))
print("Train Length: {} | Validation Length: {} | Test Length: {}".format(len(train_da
print("{} + {} + {} = {}".format(len(train_data_df), len(validation_data_df), len(test
```

```
Total Length: 24336
Train Length: 17035 | Validation Length: 3650 | Test Length: 3651
17035 + 3650 + 3651 = 24336
```

In [29]:

```
'''
data의 shape를 확인한다
'''
print("Total Length: {}".format(total_size))
print("Total DataFrame Shape: {}".format(scaled_data_df.shape))
print("Train DataFrame Shape: {}".format(train_data_df.shape))
print("Validation DataFrame Shape: {}".format(validation_data_df.shape))
print("Test DataFrame Shape: {}".format(test_data_df.shape))
```

```
Total Length: 24336
Total DataFrame Shape: (24336, 8)
Train DataFrame Shape: (17035, 8)
Validation DataFrame Shape: (3650, 8)
Test DataFrame Shape: (3651, 8)
```

In [30]:

```
'''
train, validation, test data를 input, target data로 분리한다
'''
input_feature_list = ['frontyear', 'backyear', 'month', 'day', 'season']
target_feature_list = ['temp_avg', 'temp_min', 'temp_max']

x_train_data_df = train_data_df[input_feature_list]
y_train_data_df = train_data_df[target_feature_list]
x_validation_data_df = validation_data_df[input_feature_list]
y_validation_data_df = validation_data_df[target_feature_list]
x_test_data_df = test_data_df[input_feature_list]
y_test_data_df = test_data_df[target_feature_list]
```

In [31]:

```
'''
분리된 train, validation, test의 input, target data shape를 확인한다
'''
print("X Train DataFrame Shape: {}".format(x_train_data_df.shape))
print("Y Train DataFrame Shape: {}".format(y_train_data_df.shape))
print("X Validation DataFrame Shape: {}".format(x_validation_data_df.shape))
print("Y Validation DataFrame Shape: {}".format(y_validation_data_df.shape))
print("X Test DataFrame Shape: {}".format(x_test_data_df.shape))
print("Y Test DataFrame Shape: {}".format(y_test_data_df.shape))
```

```
X Train DataFrame Shape: (17035, 5)
Y Train DataFrame Shape: (17035, 3)
X Validation DataFrame Shape: (3650, 5)
Y Validation DataFrame Shape: (3650, 3)
```

```
X Test DataFrame Shape: (3651, 5)
Y Test DataFrame Shape: (3651, 3)
```

In [32]:

```
'''
GPU 사용 여부를 확인한다
'''
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

In [33]:

```
print("{} has been operated".format(device))
```

```
cuda has been operated
```

In [34]:

```
'''
프로그래밍된 코드를 일관된 데이터로써 확인하기 위해 랜덤시드를 고정한다
'''
torch.manual_seed(515)
```

Out[34]: <torch._C.Generator at 0x2872ac05f50>

In [35]:

```
'''
하이퍼파라미터를 정의한다

input shape : (batch size, sequence length, input dimension)
'''
num_years = 21
batch_size = 12 * num_years # 21years data every batch
sequence_length = 1
input_size = 5 # input data has five features
hidden_size = 32
num_layers = 3
output_size = 3 # output data has three features
learning_rate = 1e-5
max_norm = 5 # gradient clipping
nb_epochs = 1000
```

In [36]:

```
'''
dataset function를 정의한다
'''
def MakeDataSet(x_data_df, y_data_df):
    x_ts = torch.FloatTensor(np.array(x_data_df))
    y_ts = torch.FloatTensor(np.array(y_data_df))
    dataset_ts = TensorDataset(x_ts, y_ts)
    return dataset_ts
```

In [37]:

```
'''
dataloader function를 정의한다
'''
def MakeDataLoader(dataset, batch_size):
    dataloader = DataLoader(dataset, batch_size = batch_size, shuffle = True)
    return dataloader
```

In [38]:

```
'''
dataset을 구성한다
'''
train_dataset_ts = MakeDataSet(x_train_data_df, y_train_data_df)
validation_dataset_ts = MakeDataSet(x_validation_data_df, y_validation_data_df)
test_dataset_ts = MakeDataSet(x_test_data_df, y_test_data_df)
```

In [39]:

```
'''
dataloader를 구성한다
'''
train_dataloader = MakeDataLoader(train_dataset_ts, batch_size)
```

```
validation_dataloader = MakeDataLoader(validation_dataset_ts, batch_size)
test_dataloader = MakeDataLoader(test_dataset_ts, batch_size)
```

In [40]:

```
'''
사용될 데이터의 일부분을 확인한다
'''
for index, value in enumerate(train_dataloader):
    while index < 6:
        x, y = value
        print("{} Batch".format(index))
        print("Input: {}".format(x.shape))
        print("Target: {}".format(y.shape))

        break
```

```
0 Batch
Input: torch.Size([252, 5])
Target: torch.Size([252, 3])
1 Batch
Input: torch.Size([252, 5])
Target: torch.Size([252, 3])
2 Batch
Input: torch.Size([252, 5])
Target: torch.Size([252, 3])
3 Batch
Input: torch.Size([252, 5])
Target: torch.Size([252, 3])
4 Batch
Input: torch.Size([252, 5])
Target: torch.Size([252, 3])
5 Batch
Input: torch.Size([252, 5])
Target: torch.Size([252, 3])
```

In [41]:

```
'''
모델의 양방향 LSTM layer 를 정의한다

'''
class BiLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers):
        super(BiLSTM, self).__init__()

        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.num_layers = num_layers

        self.lstm = nn.LSTM(
            input_size = self.input_size,
            hidden_size = self.hidden_size,
            num_layers = self.num_layers,
            dropout = 0.3,
            batch_first = True,
            bidirectional = True)

        self.fc = nn.Linear(
            in_features = hidden_size * 2,
            out_features = output_size,
            bias = True)

    def forward(self, x):
        # init hidden and cell state
        hidden_state_0 = torch.zeros(self.num_layers * 2, x.size(0), self.hidden_size
        cell_state_0 = torch.zeros(self.num_layers * 2, x.size(0), self.hidden_size).

        # forward pass
```

```
        out, _ = self.lstm(x, (hidden_state_0, cell_state_0))
        out = self.fc(out[:, -1, :])

        return out
```

In [42]:
```
'''
모델, 비용함수, 옵티마이저를 구성한다
'''
model = BiLSTM(input_size, hidden_size, output_size, num_layers).to(device)
criterion = nn.MSELoss().to(device)
optimizer = optim.Adam(model.parameters(), lr = learning_rate, weight_decay = 1e-5)
```

In [43]:
```
print(model)
print(criterion)
print(optimizer)
```

```
BiLSTM(
  (lstm): LSTM(5, 32, num_layers=3, batch_first=True, dropout=0.3, bidirectional=True)
  (fc): Linear(in_features=64, out_features=3, bias=True)
)
MSELoss()
Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    eps: 1e-08
    lr: 1e-05
    weight_decay: 1e-05
)
```

In [44]:
```
'''
구성된 데이터, 모델, 비용함수, 옵티마이저를 통해 테스트를 진행한다
'''
x, y = list(train_dataloader)[0]
x = x.view(-1, sequence_length, input_size).to(device)
y = y.to(device)
hypothesis = model(x)
loss = criterion(hypothesis, y)
```

In [45]:
```
print("x: {} | {} | x demension: {}".format(x[0], x.shape, x.dim()))
print("y: {} | {} | y demension: {}".format(y[0], y.shape, y.dim()))
print("hypothesis: {} | {}".format(hypothesis.shape, hypothesis.dim()))
print("loss: {}".format(loss))
```

```
x: tensor([[0.9500, 0.6364, 0.5833, 0.4516, 1.0000]], device='cuda:0') | torch.Size([2
52, 1, 5]) | x demension: 3
y: tensor([0.7448, 0.7261, 0.7475], device='cuda:0') | torch.Size([252, 3]) | y demens
ion: 2
hypothesis: torch.Size([252, 3]) | 2
loss: 0.19864310324192047
```

In [46]:
```
'''
학습을 진행한다
'''
trn_loss_list = []
val_loss_list = []
for epoch in range(nb_epochs):

    # Train
    trn_loss = 0.0
    for i, train_samples in enumerate(train_dataloader):

        # train data setting
        x_train, y_train = train_samples
        x_train = x_train.view(-1, sequence_length, input_size).to(device)
```

```
        y_train = y_train.to(device)

        # train
        model.train()
        hypothesis = model(x_train)
        optimizer.zero_grad()
        train_loss = criterion(hypothesis, y_train)
        train_loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm)
        optimizer.step()

        # train loss
        trn_loss += train_loss.item() / len(train_dataloader)

    trn_loss_list.append(trn_loss)

    # Evaluation
    with torch.no_grad():
        val_loss = 0.0
        for j, validation_samples in enumerate(validation_dataloader):

            # validatoin data setting
            x_validation, y_validation = validation_samples
            x_validation = x_validation.view(-1, sequence_length, input_size).to(devi
            y_validation = y_validation.to(device)

            # evaluation
            model.eval()
            prediction = model(x_validation)
            validation_loss = criterion(prediction, y_validation)

            # validation loss
            val_loss += validation_loss.item() / len(validation_dataloader)
    val_loss_list.append(val_loss)

    print("Epoch: {:3d} | Train Loss: {:.6f} | Val Loss: {:.6f}".format(epoch + 1, tr

torch.save(model, './data/temperature_model.pt')
```

```
Epoch:   1 | Train Loss: 0.219380 | Val Loss: 0.213915
Epoch:   2 | Train Loss: 0.217031 | Val Loss: 0.210942
Epoch:   3 | Train Loss: 0.214400 | Val Loss: 0.210408
Epoch:   4 | Train Loss: 0.211895 | Val Loss: 0.207402
Epoch:   5 | Train Loss: 0.209547 | Val Loss: 0.205251
Epoch:   6 | Train Loss: 0.206984 | Val Loss: 0.202386
Epoch:   7 | Train Loss: 0.204331 | Val Loss: 0.200600
Epoch:   8 | Train Loss: 0.201812 | Val Loss: 0.197473
Epoch:   9 | Train Loss: 0.199078 | Val Loss: 0.195296
Epoch:  10 | Train Loss: 0.196472 | Val Loss: 0.192633
Epoch:  11 | Train Loss: 0.193868 | Val Loss: 0.189713
Epoch:  12 | Train Loss: 0.190986 | Val Loss: 0.187165
Epoch:  13 | Train Loss: 0.188046 | Val Loss: 0.184730
Epoch:  14 | Train Loss: 0.185035 | Val Loss: 0.181143
Epoch:  15 | Train Loss: 0.182002 | Val Loss: 0.179290
Epoch:  16 | Train Loss: 0.178791 | Val Loss: 0.175426
Epoch:  17 | Train Loss: 0.175501 | Val Loss: 0.171754
Epoch:  18 | Train Loss: 0.171898 | Val Loss: 0.168422
Epoch:  19 | Train Loss: 0.168349 | Val Loss: 0.164839
Epoch:  20 | Train Loss: 0.164755 | Val Loss: 0.161874
Epoch:  21 | Train Loss: 0.160848 | Val Loss: 0.157282
Epoch:  22 | Train Loss: 0.156815 | Val Loss: 0.153904
Epoch:  23 | Train Loss: 0.152721 | Val Loss: 0.149386
Epoch:  24 | Train Loss: 0.148445 | Val Loss: 0.145868
Epoch:  25 | Train Loss: 0.144050 | Val Loss: 0.142113
Epoch:  26 | Train Loss: 0.139673 | Val Loss: 0.138127
Epoch:  27 | Train Loss: 0.135176 | Val Loss: 0.133486
Epoch:  28 | Train Loss: 0.130856 | Val Loss: 0.129438
```

```
Epoch:  29 | Train Loss: 0.126323 | Val Loss: 0.124875
Epoch:  30 | Train Loss: 0.122040 | Val Loss: 0.120729
Epoch:  31 | Train Loss: 0.117827 | Val Loss: 0.117501
Epoch:  32 | Train Loss: 0.113765 | Val Loss: 0.113030
Epoch:  33 | Train Loss: 0.110146 | Val Loss: 0.110785
Epoch:  34 | Train Loss: 0.106826 | Val Loss: 0.107442
Epoch:  35 | Train Loss: 0.103527 | Val Loss: 0.104458
Epoch:  36 | Train Loss: 0.101076 | Val Loss: 0.102249
Epoch:  37 | Train Loss: 0.098503 | Val Loss: 0.100307
Epoch:  38 | Train Loss: 0.096643 | Val Loss: 0.098486
Epoch:  39 | Train Loss: 0.094890 | Val Loss: 0.097290
Epoch:  40 | Train Loss: 0.093739 | Val Loss: 0.096036
Epoch:  41 | Train Loss: 0.092479 | Val Loss: 0.094862
Epoch:  42 | Train Loss: 0.091818 | Val Loss: 0.094576
Epoch:  43 | Train Loss: 0.091337 | Val Loss: 0.094230
Epoch:  44 | Train Loss: 0.090736 | Val Loss: 0.094101
Epoch:  45 | Train Loss: 0.090411 | Val Loss: 0.092931
Epoch:  46 | Train Loss: 0.090167 | Val Loss: 0.093371
Epoch:  47 | Train Loss: 0.089707 | Val Loss: 0.092253
Epoch:  48 | Train Loss: 0.089766 | Val Loss: 0.091990
Epoch:  49 | Train Loss: 0.089470 | Val Loss: 0.092403
Epoch:  50 | Train Loss: 0.089290 | Val Loss: 0.091954
Epoch:  51 | Train Loss: 0.089048 | Val Loss: 0.091618
Epoch:  52 | Train Loss: 0.088906 | Val Loss: 0.091965
Epoch:  53 | Train Loss: 0.088550 | Val Loss: 0.091168
Epoch:  54 | Train Loss: 0.088428 | Val Loss: 0.091801
Epoch:  55 | Train Loss: 0.088124 | Val Loss: 0.090760
Epoch:  56 | Train Loss: 0.088010 | Val Loss: 0.090826
Epoch:  57 | Train Loss: 0.087956 | Val Loss: 0.090280
Epoch:  58 | Train Loss: 0.087609 | Val Loss: 0.090588
Epoch:  59 | Train Loss: 0.087246 | Val Loss: 0.089964
Epoch:  60 | Train Loss: 0.087164 | Val Loss: 0.089986
Epoch:  61 | Train Loss: 0.086931 | Val Loss: 0.089734
Epoch:  62 | Train Loss: 0.086568 | Val Loss: 0.088985
Epoch:  63 | Train Loss: 0.086346 | Val Loss: 0.089354
Epoch:  64 | Train Loss: 0.086355 | Val Loss: 0.088443
Epoch:  65 | Train Loss: 0.085713 | Val Loss: 0.088691
Epoch:  66 | Train Loss: 0.085519 | Val Loss: 0.088061
Epoch:  67 | Train Loss: 0.085045 | Val Loss: 0.088150
Epoch:  68 | Train Loss: 0.084945 | Val Loss: 0.087741
Epoch:  69 | Train Loss: 0.084616 | Val Loss: 0.087533
Epoch:  70 | Train Loss: 0.084353 | Val Loss: 0.086901
Epoch:  71 | Train Loss: 0.084198 | Val Loss: 0.086220
Epoch:  72 | Train Loss: 0.083568 | Val Loss: 0.085964
Epoch:  73 | Train Loss: 0.083277 | Val Loss: 0.085577
Epoch:  74 | Train Loss: 0.082700 | Val Loss: 0.085443
Epoch:  75 | Train Loss: 0.082637 | Val Loss: 0.084959
Epoch:  76 | Train Loss: 0.082356 | Val Loss: 0.084416
Epoch:  77 | Train Loss: 0.081667 | Val Loss: 0.083997
Epoch:  78 | Train Loss: 0.081326 | Val Loss: 0.083464
Epoch:  79 | Train Loss: 0.080727 | Val Loss: 0.082810
Epoch:  80 | Train Loss: 0.080221 | Val Loss: 0.082363
Epoch:  81 | Train Loss: 0.080003 | Val Loss: 0.082152
Epoch:  82 | Train Loss: 0.079541 | Val Loss: 0.081791
Epoch:  83 | Train Loss: 0.078812 | Val Loss: 0.081235
Epoch:  84 | Train Loss: 0.078427 | Val Loss: 0.080692
Epoch:  85 | Train Loss: 0.077921 | Val Loss: 0.080434
Epoch:  86 | Train Loss: 0.077289 | Val Loss: 0.079669
Epoch:  87 | Train Loss: 0.076995 | Val Loss: 0.079007
Epoch:  88 | Train Loss: 0.076305 | Val Loss: 0.078295
Epoch:  89 | Train Loss: 0.075611 | Val Loss: 0.077707
Epoch:  90 | Train Loss: 0.075063 | Val Loss: 0.076937
Epoch:  91 | Train Loss: 0.074670 | Val Loss: 0.076610
Epoch:  92 | Train Loss: 0.073950 | Val Loss: 0.075972
Epoch:  93 | Train Loss: 0.073290 | Val Loss: 0.075319
Epoch:  94 | Train Loss: 0.072471 | Val Loss: 0.074476
Epoch:  95 | Train Loss: 0.071907 | Val Loss: 0.073522
Epoch:  96 | Train Loss: 0.071323 | Val Loss: 0.073216
Epoch:  97 | Train Loss: 0.070765 | Val Loss: 0.072034
```

```
Epoch:  98 | Train Loss: 0.069895 | Val Loss: 0.071210
Epoch:  99 | Train Loss: 0.069183 | Val Loss: 0.070639
Epoch: 100 | Train Loss: 0.068371 | Val Loss: 0.069831
Epoch: 101 | Train Loss: 0.067812 | Val Loss: 0.069836
Epoch: 102 | Train Loss: 0.067005 | Val Loss: 0.068414
Epoch: 103 | Train Loss: 0.066382 | Val Loss: 0.067166
Epoch: 104 | Train Loss: 0.065535 | Val Loss: 0.066635
Epoch: 105 | Train Loss: 0.064705 | Val Loss: 0.065740
Epoch: 106 | Train Loss: 0.064301 | Val Loss: 0.064646
Epoch: 107 | Train Loss: 0.063078 | Val Loss: 0.063969
Epoch: 108 | Train Loss: 0.062235 | Val Loss: 0.063256
Epoch: 109 | Train Loss: 0.061351 | Val Loss: 0.061913
Epoch: 110 | Train Loss: 0.060454 | Val Loss: 0.061687
Epoch: 111 | Train Loss: 0.059513 | Val Loss: 0.060173
Epoch: 112 | Train Loss: 0.058980 | Val Loss: 0.059303
Epoch: 113 | Train Loss: 0.057836 | Val Loss: 0.058432
Epoch: 114 | Train Loss: 0.057008 | Val Loss: 0.057831
Epoch: 115 | Train Loss: 0.056244 | Val Loss: 0.056603
Epoch: 116 | Train Loss: 0.055541 | Val Loss: 0.055398
Epoch: 117 | Train Loss: 0.054196 | Val Loss: 0.054775
Epoch: 118 | Train Loss: 0.053319 | Val Loss: 0.053870
Epoch: 119 | Train Loss: 0.052637 | Val Loss: 0.052806
Epoch: 120 | Train Loss: 0.051799 | Val Loss: 0.051727
Epoch: 121 | Train Loss: 0.051009 | Val Loss: 0.051007
Epoch: 122 | Train Loss: 0.049997 | Val Loss: 0.050241
Epoch: 123 | Train Loss: 0.049316 | Val Loss: 0.048913
Epoch: 124 | Train Loss: 0.048622 | Val Loss: 0.048047
Epoch: 125 | Train Loss: 0.047718 | Val Loss: 0.047030
Epoch: 126 | Train Loss: 0.046716 | Val Loss: 0.046297
Epoch: 127 | Train Loss: 0.046049 | Val Loss: 0.045485
Epoch: 128 | Train Loss: 0.045171 | Val Loss: 0.044629
Epoch: 129 | Train Loss: 0.044724 | Val Loss: 0.043775
Epoch: 130 | Train Loss: 0.044036 | Val Loss: 0.042943
Epoch: 131 | Train Loss: 0.043160 | Val Loss: 0.041780
Epoch: 132 | Train Loss: 0.042293 | Val Loss: 0.041115
Epoch: 133 | Train Loss: 0.041676 | Val Loss: 0.040644
Epoch: 134 | Train Loss: 0.041144 | Val Loss: 0.039804
Epoch: 135 | Train Loss: 0.040713 | Val Loss: 0.038911
Epoch: 136 | Train Loss: 0.039610 | Val Loss: 0.038311
Epoch: 137 | Train Loss: 0.039334 | Val Loss: 0.037735
Epoch: 138 | Train Loss: 0.038655 | Val Loss: 0.036799
Epoch: 139 | Train Loss: 0.038483 | Val Loss: 0.036243
Epoch: 140 | Train Loss: 0.037771 | Val Loss: 0.035819
Epoch: 141 | Train Loss: 0.037263 | Val Loss: 0.035044
Epoch: 142 | Train Loss: 0.036504 | Val Loss: 0.034653
Epoch: 143 | Train Loss: 0.036432 | Val Loss: 0.034259
Epoch: 144 | Train Loss: 0.036010 | Val Loss: 0.033890
Epoch: 145 | Train Loss: 0.035713 | Val Loss: 0.033118
Epoch: 146 | Train Loss: 0.035190 | Val Loss: 0.032605
Epoch: 147 | Train Loss: 0.034715 | Val Loss: 0.032351
Epoch: 148 | Train Loss: 0.034681 | Val Loss: 0.032083
Epoch: 149 | Train Loss: 0.034012 | Val Loss: 0.031859
Epoch: 150 | Train Loss: 0.034253 | Val Loss: 0.031190
Epoch: 151 | Train Loss: 0.033739 | Val Loss: 0.030881
Epoch: 152 | Train Loss: 0.033825 | Val Loss: 0.030901
Epoch: 153 | Train Loss: 0.033314 | Val Loss: 0.030539
Epoch: 154 | Train Loss: 0.033058 | Val Loss: 0.030249
Epoch: 155 | Train Loss: 0.033054 | Val Loss: 0.029842
Epoch: 156 | Train Loss: 0.032723 | Val Loss: 0.029846
Epoch: 157 | Train Loss: 0.032252 | Val Loss: 0.029559
Epoch: 158 | Train Loss: 0.032237 | Val Loss: 0.029418
Epoch: 159 | Train Loss: 0.032252 | Val Loss: 0.029144
Epoch: 160 | Train Loss: 0.032048 | Val Loss: 0.029082
Epoch: 161 | Train Loss: 0.032006 | Val Loss: 0.028579
Epoch: 162 | Train Loss: 0.031872 | Val Loss: 0.028726
Epoch: 163 | Train Loss: 0.031503 | Val Loss: 0.028548
Epoch: 164 | Train Loss: 0.031885 | Val Loss: 0.028461
Epoch: 165 | Train Loss: 0.031678 | Val Loss: 0.028227
Epoch: 166 | Train Loss: 0.031639 | Val Loss: 0.028296
```

```
Epoch: 167 | Train Loss: 0.031300 | Val Loss: 0.028188
Epoch: 168 | Train Loss: 0.031431 | Val Loss: 0.028235
Epoch: 169 | Train Loss: 0.031159 | Val Loss: 0.027949
Epoch: 170 | Train Loss: 0.030999 | Val Loss: 0.028028
Epoch: 171 | Train Loss: 0.031151 | Val Loss: 0.027996
Epoch: 172 | Train Loss: 0.031071 | Val Loss: 0.027804
Epoch: 173 | Train Loss: 0.031023 | Val Loss: 0.027978
Epoch: 174 | Train Loss: 0.030715 | Val Loss: 0.027854
Epoch: 175 | Train Loss: 0.030877 | Val Loss: 0.027690
Epoch: 176 | Train Loss: 0.030694 | Val Loss: 0.027797
Epoch: 177 | Train Loss: 0.030594 | Val Loss: 0.027617
Epoch: 178 | Train Loss: 0.030949 | Val Loss: 0.027891
Epoch: 179 | Train Loss: 0.030681 | Val Loss: 0.027358
Epoch: 180 | Train Loss: 0.030793 | Val Loss: 0.027550
Epoch: 181 | Train Loss: 0.030491 | Val Loss: 0.027513
Epoch: 182 | Train Loss: 0.030659 | Val Loss: 0.027506
Epoch: 183 | Train Loss: 0.030576 | Val Loss: 0.027479
Epoch: 184 | Train Loss: 0.030425 | Val Loss: 0.027706
Epoch: 185 | Train Loss: 0.030439 | Val Loss: 0.027388
Epoch: 186 | Train Loss: 0.030296 | Val Loss: 0.027407
Epoch: 187 | Train Loss: 0.030315 | Val Loss: 0.027420
Epoch: 188 | Train Loss: 0.030125 | Val Loss: 0.027370
Epoch: 189 | Train Loss: 0.030148 | Val Loss: 0.027087
Epoch: 190 | Train Loss: 0.030189 | Val Loss: 0.027272
Epoch: 191 | Train Loss: 0.030321 | Val Loss: 0.027318
Epoch: 192 | Train Loss: 0.030173 | Val Loss: 0.027050
Epoch: 193 | Train Loss: 0.030092 | Val Loss: 0.027207
Epoch: 194 | Train Loss: 0.030019 | Val Loss: 0.027486
Epoch: 195 | Train Loss: 0.029960 | Val Loss: 0.027230
Epoch: 196 | Train Loss: 0.030049 | Val Loss: 0.027353
Epoch: 197 | Train Loss: 0.029874 | Val Loss: 0.027163
Epoch: 198 | Train Loss: 0.030290 | Val Loss: 0.027285
Epoch: 199 | Train Loss: 0.029917 | Val Loss: 0.027323
Epoch: 200 | Train Loss: 0.029723 | Val Loss: 0.027175
Epoch: 201 | Train Loss: 0.029823 | Val Loss: 0.027107
Epoch: 202 | Train Loss: 0.029993 | Val Loss: 0.027160
Epoch: 203 | Train Loss: 0.029668 | Val Loss: 0.027205
Epoch: 204 | Train Loss: 0.029601 | Val Loss: 0.027090
Epoch: 205 | Train Loss: 0.029589 | Val Loss: 0.026996
Epoch: 206 | Train Loss: 0.029708 | Val Loss: 0.027377
Epoch: 207 | Train Loss: 0.029784 | Val Loss: 0.027000
Epoch: 208 | Train Loss: 0.029672 | Val Loss: 0.027268
Epoch: 209 | Train Loss: 0.029568 | Val Loss: 0.027117
Epoch: 210 | Train Loss: 0.029630 | Val Loss: 0.027313
Epoch: 211 | Train Loss: 0.029649 | Val Loss: 0.027135
Epoch: 212 | Train Loss: 0.029515 | Val Loss: 0.027287
Epoch: 213 | Train Loss: 0.029743 | Val Loss: 0.027159
Epoch: 214 | Train Loss: 0.029326 | Val Loss: 0.027114
Epoch: 215 | Train Loss: 0.029395 | Val Loss: 0.027019
Epoch: 216 | Train Loss: 0.029481 | Val Loss: 0.027109
Epoch: 217 | Train Loss: 0.029397 | Val Loss: 0.027299
Epoch: 218 | Train Loss: 0.029454 | Val Loss: 0.027148
Epoch: 219 | Train Loss: 0.029237 | Val Loss: 0.026979
Epoch: 220 | Train Loss: 0.029283 | Val Loss: 0.027069
Epoch: 221 | Train Loss: 0.029269 | Val Loss: 0.027012
Epoch: 222 | Train Loss: 0.029350 | Val Loss: 0.026930
Epoch: 223 | Train Loss: 0.029301 | Val Loss: 0.026991
Epoch: 224 | Train Loss: 0.029303 | Val Loss: 0.027055
Epoch: 225 | Train Loss: 0.029145 | Val Loss: 0.026791
Epoch: 226 | Train Loss: 0.029036 | Val Loss: 0.026785
Epoch: 227 | Train Loss: 0.029153 | Val Loss: 0.027213
Epoch: 228 | Train Loss: 0.029183 | Val Loss: 0.027077
Epoch: 229 | Train Loss: 0.029162 | Val Loss: 0.026856
Epoch: 230 | Train Loss: 0.029116 | Val Loss: 0.027029
Epoch: 231 | Train Loss: 0.029251 | Val Loss: 0.026921
Epoch: 232 | Train Loss: 0.028944 | Val Loss: 0.026913
Epoch: 233 | Train Loss: 0.028979 | Val Loss: 0.027154
Epoch: 234 | Train Loss: 0.029097 | Val Loss: 0.027210
Epoch: 235 | Train Loss: 0.028910 | Val Loss: 0.027054
```

```
Epoch: 236 | Train Loss: 0.029018 | Val Loss: 0.026846
Epoch: 237 | Train Loss: 0.028896 | Val Loss: 0.026836
Epoch: 238 | Train Loss: 0.028745 | Val Loss: 0.026931
Epoch: 239 | Train Loss: 0.028978 | Val Loss: 0.026692
Epoch: 240 | Train Loss: 0.028690 | Val Loss: 0.026860
Epoch: 241 | Train Loss: 0.028934 | Val Loss: 0.026775
Epoch: 242 | Train Loss: 0.028584 | Val Loss: 0.026776
Epoch: 243 | Train Loss: 0.028790 | Val Loss: 0.026925
Epoch: 244 | Train Loss: 0.028938 | Val Loss: 0.026683
Epoch: 245 | Train Loss: 0.028978 | Val Loss: 0.026898
Epoch: 246 | Train Loss: 0.028801 | Val Loss: 0.026781
Epoch: 247 | Train Loss: 0.028783 | Val Loss: 0.026832
Epoch: 248 | Train Loss: 0.028702 | Val Loss: 0.026712
Epoch: 249 | Train Loss: 0.028674 | Val Loss: 0.026784
Epoch: 250 | Train Loss: 0.028729 | Val Loss: 0.026917
Epoch: 251 | Train Loss: 0.028547 | Val Loss: 0.026660
Epoch: 252 | Train Loss: 0.028694 | Val Loss: 0.026834
Epoch: 253 | Train Loss: 0.028841 | Val Loss: 0.026838
Epoch: 254 | Train Loss: 0.028695 | Val Loss: 0.026743
Epoch: 255 | Train Loss: 0.028708 | Val Loss: 0.026854
Epoch: 256 | Train Loss: 0.028564 | Val Loss: 0.026975
Epoch: 257 | Train Loss: 0.028662 | Val Loss: 0.026734
Epoch: 258 | Train Loss: 0.028564 | Val Loss: 0.026777
Epoch: 259 | Train Loss: 0.028469 | Val Loss: 0.026797
Epoch: 260 | Train Loss: 0.028418 | Val Loss: 0.026933
Epoch: 261 | Train Loss: 0.028439 | Val Loss: 0.026738
Epoch: 262 | Train Loss: 0.028569 | Val Loss: 0.026653
Epoch: 263 | Train Loss: 0.028385 | Val Loss: 0.026622
Epoch: 264 | Train Loss: 0.028384 | Val Loss: 0.026930
Epoch: 265 | Train Loss: 0.028304 | Val Loss: 0.026588
Epoch: 266 | Train Loss: 0.028393 | Val Loss: 0.027026
Epoch: 267 | Train Loss: 0.028600 | Val Loss: 0.026989
Epoch: 268 | Train Loss: 0.028429 | Val Loss: 0.026779
Epoch: 269 | Train Loss: 0.028439 | Val Loss: 0.026842
Epoch: 270 | Train Loss: 0.028392 | Val Loss: 0.026733
Epoch: 271 | Train Loss: 0.028195 | Val Loss: 0.026639
Epoch: 272 | Train Loss: 0.028229 | Val Loss: 0.026542
Epoch: 273 | Train Loss: 0.028333 | Val Loss: 0.026642
Epoch: 274 | Train Loss: 0.028248 | Val Loss: 0.026727
Epoch: 275 | Train Loss: 0.028213 | Val Loss: 0.026658
Epoch: 276 | Train Loss: 0.028465 | Val Loss: 0.026605
Epoch: 277 | Train Loss: 0.028311 | Val Loss: 0.026712
Epoch: 278 | Train Loss: 0.028416 | Val Loss: 0.026764
Epoch: 279 | Train Loss: 0.028242 | Val Loss: 0.026746
Epoch: 280 | Train Loss: 0.028292 | Val Loss: 0.026608
Epoch: 281 | Train Loss: 0.028407 | Val Loss: 0.026603
Epoch: 282 | Train Loss: 0.028229 | Val Loss: 0.026638
Epoch: 283 | Train Loss: 0.028225 | Val Loss: 0.026596
Epoch: 284 | Train Loss: 0.028148 | Val Loss: 0.026800
Epoch: 285 | Train Loss: 0.028190 | Val Loss: 0.026939
Epoch: 286 | Train Loss: 0.027850 | Val Loss: 0.026695
Epoch: 287 | Train Loss: 0.028146 | Val Loss: 0.026924
Epoch: 288 | Train Loss: 0.028020 | Val Loss: 0.026554
Epoch: 289 | Train Loss: 0.028042 | Val Loss: 0.026691
Epoch: 290 | Train Loss: 0.028259 | Val Loss: 0.026624
Epoch: 291 | Train Loss: 0.027990 | Val Loss: 0.026739
Epoch: 292 | Train Loss: 0.028165 | Val Loss: 0.026821
Epoch: 293 | Train Loss: 0.028096 | Val Loss: 0.026971
Epoch: 294 | Train Loss: 0.028129 | Val Loss: 0.026658
Epoch: 295 | Train Loss: 0.028190 | Val Loss: 0.026508
Epoch: 296 | Train Loss: 0.027909 | Val Loss: 0.026787
Epoch: 297 | Train Loss: 0.028014 | Val Loss: 0.026596
Epoch: 298 | Train Loss: 0.028041 | Val Loss: 0.026648
Epoch: 299 | Train Loss: 0.027869 | Val Loss: 0.026674
Epoch: 300 | Train Loss: 0.028029 | Val Loss: 0.026591
Epoch: 301 | Train Loss: 0.027970 | Val Loss: 0.026616
Epoch: 302 | Train Loss: 0.027957 | Val Loss: 0.026586
Epoch: 303 | Train Loss: 0.027901 | Val Loss: 0.026527
Epoch: 304 | Train Loss: 0.027978 | Val Loss: 0.026636
```

```
Epoch: 305 | Train Loss: 0.027992 | Val Loss: 0.026581
Epoch: 306 | Train Loss: 0.027784 | Val Loss: 0.026564
Epoch: 307 | Train Loss: 0.027862 | Val Loss: 0.026861
Epoch: 308 | Train Loss: 0.028021 | Val Loss: 0.026776
Epoch: 309 | Train Loss: 0.027963 | Val Loss: 0.026533
Epoch: 310 | Train Loss: 0.027946 | Val Loss: 0.026644
Epoch: 311 | Train Loss: 0.027689 | Val Loss: 0.026475
Epoch: 312 | Train Loss: 0.027844 | Val Loss: 0.026606
Epoch: 313 | Train Loss: 0.028052 | Val Loss: 0.026870
Epoch: 314 | Train Loss: 0.027880 | Val Loss: 0.026583
Epoch: 315 | Train Loss: 0.027787 | Val Loss: 0.026450
Epoch: 316 | Train Loss: 0.027734 | Val Loss: 0.026738
Epoch: 317 | Train Loss: 0.027982 | Val Loss: 0.026694
Epoch: 318 | Train Loss: 0.027881 | Val Loss: 0.026503
Epoch: 319 | Train Loss: 0.027634 | Val Loss: 0.026666
Epoch: 320 | Train Loss: 0.027784 | Val Loss: 0.026602
Epoch: 321 | Train Loss: 0.027772 | Val Loss: 0.026564
Epoch: 322 | Train Loss: 0.027592 | Val Loss: 0.026590
Epoch: 323 | Train Loss: 0.027657 | Val Loss: 0.026714
Epoch: 324 | Train Loss: 0.027830 | Val Loss: 0.026936
Epoch: 325 | Train Loss: 0.027730 | Val Loss: 0.026617
Epoch: 326 | Train Loss: 0.027768 | Val Loss: 0.026649
Epoch: 327 | Train Loss: 0.027732 | Val Loss: 0.026575
Epoch: 328 | Train Loss: 0.027765 | Val Loss: 0.026609
Epoch: 329 | Train Loss: 0.027714 | Val Loss: 0.026507
Epoch: 330 | Train Loss: 0.027869 | Val Loss: 0.026755
Epoch: 331 | Train Loss: 0.027571 | Val Loss: 0.026281
Epoch: 332 | Train Loss: 0.027620 | Val Loss: 0.026533
Epoch: 333 | Train Loss: 0.027648 | Val Loss: 0.026527
Epoch: 334 | Train Loss: 0.027751 | Val Loss: 0.026616
Epoch: 335 | Train Loss: 0.027667 | Val Loss: 0.026426
Epoch: 336 | Train Loss: 0.027890 | Val Loss: 0.026534
Epoch: 337 | Train Loss: 0.027655 | Val Loss: 0.026590
Epoch: 338 | Train Loss: 0.027643 | Val Loss: 0.026607
Epoch: 339 | Train Loss: 0.027687 | Val Loss: 0.026557
Epoch: 340 | Train Loss: 0.027721 | Val Loss: 0.026707
Epoch: 341 | Train Loss: 0.027663 | Val Loss: 0.026515
Epoch: 342 | Train Loss: 0.027605 | Val Loss: 0.026280
Epoch: 343 | Train Loss: 0.027618 | Val Loss: 0.026585
Epoch: 344 | Train Loss: 0.027570 | Val Loss: 0.026661
Epoch: 345 | Train Loss: 0.027634 | Val Loss: 0.026526
Epoch: 346 | Train Loss: 0.027569 | Val Loss: 0.026476
Epoch: 347 | Train Loss: 0.027437 | Val Loss: 0.026592
Epoch: 348 | Train Loss: 0.027439 | Val Loss: 0.026631
Epoch: 349 | Train Loss: 0.027262 | Val Loss: 0.026370
Epoch: 350 | Train Loss: 0.027594 | Val Loss: 0.026505
Epoch: 351 | Train Loss: 0.027567 | Val Loss: 0.026623
Epoch: 352 | Train Loss: 0.027498 | Val Loss: 0.026464
Epoch: 353 | Train Loss: 0.027611 | Val Loss: 0.026568
Epoch: 354 | Train Loss: 0.027475 | Val Loss: 0.026350
Epoch: 355 | Train Loss: 0.027625 | Val Loss: 0.026705
Epoch: 356 | Train Loss: 0.027456 | Val Loss: 0.026506
Epoch: 357 | Train Loss: 0.027493 | Val Loss: 0.026611
Epoch: 358 | Train Loss: 0.027480 | Val Loss: 0.026450
Epoch: 359 | Train Loss: 0.027441 | Val Loss: 0.026461
Epoch: 360 | Train Loss: 0.027464 | Val Loss: 0.026643
Epoch: 361 | Train Loss: 0.027522 | Val Loss: 0.026604
Epoch: 362 | Train Loss: 0.027664 | Val Loss: 0.026563
Epoch: 363 | Train Loss: 0.027466 | Val Loss: 0.026546
Epoch: 364 | Train Loss: 0.027704 | Val Loss: 0.026702
Epoch: 365 | Train Loss: 0.027383 | Val Loss: 0.026485
Epoch: 366 | Train Loss: 0.027318 | Val Loss: 0.026610
Epoch: 367 | Train Loss: 0.027480 | Val Loss: 0.026312
Epoch: 368 | Train Loss: 0.027466 | Val Loss: 0.026522
Epoch: 369 | Train Loss: 0.027489 | Val Loss: 0.026686
Epoch: 370 | Train Loss: 0.027403 | Val Loss: 0.026455
Epoch: 371 | Train Loss: 0.027665 | Val Loss: 0.026445
Epoch: 372 | Train Loss: 0.027527 | Val Loss: 0.026563
Epoch: 373 | Train Loss: 0.027412 | Val Loss: 0.026227
```

```
Epoch: 374 | Train Loss: 0.027479 | Val Loss: 0.026721
Epoch: 375 | Train Loss: 0.027481 | Val Loss: 0.026420
Epoch: 376 | Train Loss: 0.027499 | Val Loss: 0.026562
Epoch: 377 | Train Loss: 0.027402 | Val Loss: 0.026517
Epoch: 378 | Train Loss: 0.027287 | Val Loss: 0.026497
Epoch: 379 | Train Loss: 0.027408 | Val Loss: 0.026493
Epoch: 380 | Train Loss: 0.027388 | Val Loss: 0.026442
Epoch: 381 | Train Loss: 0.027244 | Val Loss: 0.026520
Epoch: 382 | Train Loss: 0.027404 | Val Loss: 0.026341
Epoch: 383 | Train Loss: 0.027269 | Val Loss: 0.026304
Epoch: 384 | Train Loss: 0.027372 | Val Loss: 0.026415
Epoch: 385 | Train Loss: 0.027236 | Val Loss: 0.026524
Epoch: 386 | Train Loss: 0.027495 | Val Loss: 0.026705
Epoch: 387 | Train Loss: 0.027421 | Val Loss: 0.026500
Epoch: 388 | Train Loss: 0.027504 | Val Loss: 0.026542
Epoch: 389 | Train Loss: 0.027317 | Val Loss: 0.026436
Epoch: 390 | Train Loss: 0.027262 | Val Loss: 0.026618
Epoch: 391 | Train Loss: 0.027342 | Val Loss: 0.026635
Epoch: 392 | Train Loss: 0.027277 | Val Loss: 0.026801
Epoch: 393 | Train Loss: 0.027337 | Val Loss: 0.026422
Epoch: 394 | Train Loss: 0.027200 | Val Loss: 0.026395
Epoch: 395 | Train Loss: 0.027499 | Val Loss: 0.026422
Epoch: 396 | Train Loss: 0.027185 | Val Loss: 0.026360
Epoch: 397 | Train Loss: 0.027349 | Val Loss: 0.026444
Epoch: 398 | Train Loss: 0.027370 | Val Loss: 0.026591
Epoch: 399 | Train Loss: 0.027023 | Val Loss: 0.026413
Epoch: 400 | Train Loss: 0.027403 | Val Loss: 0.026441
Epoch: 401 | Train Loss: 0.027135 | Val Loss: 0.026462
Epoch: 402 | Train Loss: 0.027252 | Val Loss: 0.026560
Epoch: 403 | Train Loss: 0.027107 | Val Loss: 0.026504
Epoch: 404 | Train Loss: 0.027191 | Val Loss: 0.026538
Epoch: 405 | Train Loss: 0.027203 | Val Loss: 0.026641
Epoch: 406 | Train Loss: 0.027091 | Val Loss: 0.026646
Epoch: 407 | Train Loss: 0.027235 | Val Loss: 0.026514
Epoch: 408 | Train Loss: 0.027266 | Val Loss: 0.026390
Epoch: 409 | Train Loss: 0.027396 | Val Loss: 0.026292
Epoch: 410 | Train Loss: 0.027304 | Val Loss: 0.026482
Epoch: 411 | Train Loss: 0.027210 | Val Loss: 0.026405
Epoch: 412 | Train Loss: 0.027429 | Val Loss: 0.026574
Epoch: 413 | Train Loss: 0.027241 | Val Loss: 0.026385
Epoch: 414 | Train Loss: 0.027314 | Val Loss: 0.026351
Epoch: 415 | Train Loss: 0.027228 | Val Loss: 0.026429
Epoch: 416 | Train Loss: 0.027130 | Val Loss: 0.026620
Epoch: 417 | Train Loss: 0.027157 | Val Loss: 0.026709
Epoch: 418 | Train Loss: 0.027131 | Val Loss: 0.026504
Epoch: 419 | Train Loss: 0.027320 | Val Loss: 0.026496
Epoch: 420 | Train Loss: 0.027247 | Val Loss: 0.026576
Epoch: 421 | Train Loss: 0.027250 | Val Loss: 0.026508
Epoch: 422 | Train Loss: 0.027292 | Val Loss: 0.026634
Epoch: 423 | Train Loss: 0.027370 | Val Loss: 0.026378
Epoch: 424 | Train Loss: 0.027086 | Val Loss: 0.026433
Epoch: 425 | Train Loss: 0.027126 | Val Loss: 0.026714
Epoch: 426 | Train Loss: 0.027049 | Val Loss: 0.026355
Epoch: 427 | Train Loss: 0.027254 | Val Loss: 0.026279
Epoch: 428 | Train Loss: 0.027331 | Val Loss: 0.026369
Epoch: 429 | Train Loss: 0.027062 | Val Loss: 0.026146
Epoch: 430 | Train Loss: 0.027010 | Val Loss: 0.026237
Epoch: 431 | Train Loss: 0.027007 | Val Loss: 0.026446
Epoch: 432 | Train Loss: 0.027035 | Val Loss: 0.026280
Epoch: 433 | Train Loss: 0.027123 | Val Loss: 0.026268
Epoch: 434 | Train Loss: 0.027181 | Val Loss: 0.026391
Epoch: 435 | Train Loss: 0.027221 | Val Loss: 0.026453
Epoch: 436 | Train Loss: 0.027058 | Val Loss: 0.026439
Epoch: 437 | Train Loss: 0.027235 | Val Loss: 0.026440
Epoch: 438 | Train Loss: 0.027139 | Val Loss: 0.026246
Epoch: 439 | Train Loss: 0.027262 | Val Loss: 0.026597
Epoch: 440 | Train Loss: 0.027310 | Val Loss: 0.026346
Epoch: 441 | Train Loss: 0.027229 | Val Loss: 0.026370
Epoch: 442 | Train Loss: 0.027106 | Val Loss: 0.026509
```

```
Epoch: 443 | Train Loss: 0.027104 | Val Loss: 0.026339
Epoch: 444 | Train Loss: 0.027072 | Val Loss: 0.026417
Epoch: 445 | Train Loss: 0.027013 | Val Loss: 0.026220
Epoch: 446 | Train Loss: 0.027055 | Val Loss: 0.026282
Epoch: 447 | Train Loss: 0.027197 | Val Loss: 0.026706
Epoch: 448 | Train Loss: 0.027146 | Val Loss: 0.026594
Epoch: 449 | Train Loss: 0.027052 | Val Loss: 0.026418
Epoch: 450 | Train Loss: 0.026913 | Val Loss: 0.026542
Epoch: 451 | Train Loss: 0.027074 | Val Loss: 0.026330
Epoch: 452 | Train Loss: 0.026916 | Val Loss: 0.026335
Epoch: 453 | Train Loss: 0.027128 | Val Loss: 0.026397
Epoch: 454 | Train Loss: 0.027009 | Val Loss: 0.026306
Epoch: 455 | Train Loss: 0.026945 | Val Loss: 0.026334
Epoch: 456 | Train Loss: 0.027102 | Val Loss: 0.026220
Epoch: 457 | Train Loss: 0.027102 | Val Loss: 0.026514
Epoch: 458 | Train Loss: 0.027051 | Val Loss: 0.026246
Epoch: 459 | Train Loss: 0.027005 | Val Loss: 0.026276
Epoch: 460 | Train Loss: 0.026907 | Val Loss: 0.026291
Epoch: 461 | Train Loss: 0.027043 | Val Loss: 0.026236
Epoch: 462 | Train Loss: 0.027157 | Val Loss: 0.026210
Epoch: 463 | Train Loss: 0.027049 | Val Loss: 0.026144
Epoch: 464 | Train Loss: 0.026863 | Val Loss: 0.026183
Epoch: 465 | Train Loss: 0.026991 | Val Loss: 0.026188
Epoch: 466 | Train Loss: 0.026911 | Val Loss: 0.026123
Epoch: 467 | Train Loss: 0.026911 | Val Loss: 0.026511
Epoch: 468 | Train Loss: 0.026945 | Val Loss: 0.026455
Epoch: 469 | Train Loss: 0.026849 | Val Loss: 0.026336
Epoch: 470 | Train Loss: 0.026942 | Val Loss: 0.026376
Epoch: 471 | Train Loss: 0.026928 | Val Loss: 0.026218
Epoch: 472 | Train Loss: 0.026933 | Val Loss: 0.026360
Epoch: 473 | Train Loss: 0.026983 | Val Loss: 0.026234
Epoch: 474 | Train Loss: 0.027018 | Val Loss: 0.026290
Epoch: 475 | Train Loss: 0.027133 | Val Loss: 0.026411
Epoch: 476 | Train Loss: 0.027032 | Val Loss: 0.026548
Epoch: 477 | Train Loss: 0.027092 | Val Loss: 0.026207
Epoch: 478 | Train Loss: 0.026977 | Val Loss: 0.026248
Epoch: 479 | Train Loss: 0.026863 | Val Loss: 0.026245
Epoch: 480 | Train Loss: 0.026943 | Val Loss: 0.026223
Epoch: 481 | Train Loss: 0.027034 | Val Loss: 0.026199
Epoch: 482 | Train Loss: 0.026925 | Val Loss: 0.026290
Epoch: 483 | Train Loss: 0.026987 | Val Loss: 0.026188
Epoch: 484 | Train Loss: 0.026971 | Val Loss: 0.026258
Epoch: 485 | Train Loss: 0.026907 | Val Loss: 0.026083
Epoch: 486 | Train Loss: 0.026997 | Val Loss: 0.026139
Epoch: 487 | Train Loss: 0.026931 | Val Loss: 0.026129
Epoch: 488 | Train Loss: 0.026933 | Val Loss: 0.026449
Epoch: 489 | Train Loss: 0.026946 | Val Loss: 0.026236
Epoch: 490 | Train Loss: 0.026848 | Val Loss: 0.026425
Epoch: 491 | Train Loss: 0.026885 | Val Loss: 0.026253
Epoch: 492 | Train Loss: 0.026838 | Val Loss: 0.026287
Epoch: 493 | Train Loss: 0.026739 | Val Loss: 0.026159
Epoch: 494 | Train Loss: 0.026948 | Val Loss: 0.026082
Epoch: 495 | Train Loss: 0.026879 | Val Loss: 0.026165
Epoch: 496 | Train Loss: 0.026845 | Val Loss: 0.026302
Epoch: 497 | Train Loss: 0.026816 | Val Loss: 0.026220
Epoch: 498 | Train Loss: 0.026874 | Val Loss: 0.026128
Epoch: 499 | Train Loss: 0.026909 | Val Loss: 0.026456
Epoch: 500 | Train Loss: 0.026864 | Val Loss: 0.026307
Epoch: 501 | Train Loss: 0.026902 | Val Loss: 0.026245
Epoch: 502 | Train Loss: 0.027037 | Val Loss: 0.026270
Epoch: 503 | Train Loss: 0.026767 | Val Loss: 0.026141
Epoch: 504 | Train Loss: 0.026869 | Val Loss: 0.026373
Epoch: 505 | Train Loss: 0.026754 | Val Loss: 0.026154
Epoch: 506 | Train Loss: 0.026857 | Val Loss: 0.026185
Epoch: 507 | Train Loss: 0.026906 | Val Loss: 0.026191
Epoch: 508 | Train Loss: 0.026851 | Val Loss: 0.026405
Epoch: 509 | Train Loss: 0.026929 | Val Loss: 0.026321
Epoch: 510 | Train Loss: 0.026868 | Val Loss: 0.026284
Epoch: 511 | Train Loss: 0.026734 | Val Loss: 0.026243
```

```
Epoch: 512 | Train Loss: 0.026881 | Val Loss: 0.026356
Epoch: 513 | Train Loss: 0.026897 | Val Loss: 0.026070
Epoch: 514 | Train Loss: 0.026818 | Val Loss: 0.026243
Epoch: 515 | Train Loss: 0.026886 | Val Loss: 0.026250
Epoch: 516 | Train Loss: 0.026824 | Val Loss: 0.026066
Epoch: 517 | Train Loss: 0.026837 | Val Loss: 0.026183
Epoch: 518 | Train Loss: 0.026780 | Val Loss: 0.026289
Epoch: 519 | Train Loss: 0.026705 | Val Loss: 0.026213
Epoch: 520 | Train Loss: 0.026806 | Val Loss: 0.026271
Epoch: 521 | Train Loss: 0.026867 | Val Loss: 0.026099
Epoch: 522 | Train Loss: 0.026647 | Val Loss: 0.026193
Epoch: 523 | Train Loss: 0.026850 | Val Loss: 0.026297
Epoch: 524 | Train Loss: 0.026768 | Val Loss: 0.026341
Epoch: 525 | Train Loss: 0.026843 | Val Loss: 0.026238
Epoch: 526 | Train Loss: 0.026848 | Val Loss: 0.026415
Epoch: 527 | Train Loss: 0.026785 | Val Loss: 0.026119
Epoch: 528 | Train Loss: 0.026919 | Val Loss: 0.026166
Epoch: 529 | Train Loss: 0.026745 | Val Loss: 0.026044
Epoch: 530 | Train Loss: 0.026816 | Val Loss: 0.026354
Epoch: 531 | Train Loss: 0.026824 | Val Loss: 0.026159
Epoch: 532 | Train Loss: 0.026971 | Val Loss: 0.026221
Epoch: 533 | Train Loss: 0.026855 | Val Loss: 0.026317
Epoch: 534 | Train Loss: 0.026864 | Val Loss: 0.026229
Epoch: 535 | Train Loss: 0.026794 | Val Loss: 0.026312
Epoch: 536 | Train Loss: 0.026787 | Val Loss: 0.026165
Epoch: 537 | Train Loss: 0.026726 | Val Loss: 0.026201
Epoch: 538 | Train Loss: 0.026702 | Val Loss: 0.025967
Epoch: 539 | Train Loss: 0.026777 | Val Loss: 0.026230
Epoch: 540 | Train Loss: 0.026811 | Val Loss: 0.026236
Epoch: 541 | Train Loss: 0.026785 | Val Loss: 0.026172
Epoch: 542 | Train Loss: 0.026719 | Val Loss: 0.026163
Epoch: 543 | Train Loss: 0.026762 | Val Loss: 0.026175
Epoch: 544 | Train Loss: 0.026812 | Val Loss: 0.026491
Epoch: 545 | Train Loss: 0.026703 | Val Loss: 0.026065
Epoch: 546 | Train Loss: 0.026629 | Val Loss: 0.026202
Epoch: 547 | Train Loss: 0.026726 | Val Loss: 0.026199
Epoch: 548 | Train Loss: 0.026570 | Val Loss: 0.026274
Epoch: 549 | Train Loss: 0.026606 | Val Loss: 0.026113
Epoch: 550 | Train Loss: 0.026661 | Val Loss: 0.026013
Epoch: 551 | Train Loss: 0.026830 | Val Loss: 0.026205
Epoch: 552 | Train Loss: 0.026681 | Val Loss: 0.026173
Epoch: 553 | Train Loss: 0.026687 | Val Loss: 0.026415
Epoch: 554 | Train Loss: 0.026530 | Val Loss: 0.026147
Epoch: 555 | Train Loss: 0.026703 | Val Loss: 0.026081
Epoch: 556 | Train Loss: 0.026743 | Val Loss: 0.026148
Epoch: 557 | Train Loss: 0.026646 | Val Loss: 0.026397
Epoch: 558 | Train Loss: 0.026662 | Val Loss: 0.026293
Epoch: 559 | Train Loss: 0.026492 | Val Loss: 0.026100
Epoch: 560 | Train Loss: 0.026781 | Val Loss: 0.026363
Epoch: 561 | Train Loss: 0.026786 | Val Loss: 0.026092
Epoch: 562 | Train Loss: 0.026662 | Val Loss: 0.026177
Epoch: 563 | Train Loss: 0.026663 | Val Loss: 0.025969
Epoch: 564 | Train Loss: 0.026722 | Val Loss: 0.025968
Epoch: 565 | Train Loss: 0.026718 | Val Loss: 0.026044
Epoch: 566 | Train Loss: 0.026721 | Val Loss: 0.026232
Epoch: 567 | Train Loss: 0.026758 | Val Loss: 0.026066
Epoch: 568 | Train Loss: 0.026599 | Val Loss: 0.026058
Epoch: 569 | Train Loss: 0.026685 | Val Loss: 0.026159
Epoch: 570 | Train Loss: 0.026714 | Val Loss: 0.026326
Epoch: 571 | Train Loss: 0.026619 | Val Loss: 0.026200
Epoch: 572 | Train Loss: 0.026636 | Val Loss: 0.026119
Epoch: 573 | Train Loss: 0.026592 | Val Loss: 0.026371
Epoch: 574 | Train Loss: 0.026542 | Val Loss: 0.026181
Epoch: 575 | Train Loss: 0.026681 | Val Loss: 0.026103
Epoch: 576 | Train Loss: 0.026587 | Val Loss: 0.026082
Epoch: 577 | Train Loss: 0.026610 | Val Loss: 0.026038
Epoch: 578 | Train Loss: 0.026742 | Val Loss: 0.026179
Epoch: 579 | Train Loss: 0.026587 | Val Loss: 0.026179
Epoch: 580 | Train Loss: 0.026512 | Val Loss: 0.026292
```

```
Epoch: 581 | Train Loss: 0.026674 | Val Loss: 0.026212
Epoch: 582 | Train Loss: 0.026632 | Val Loss: 0.026132
Epoch: 583 | Train Loss: 0.026479 | Val Loss: 0.026197
Epoch: 584 | Train Loss: 0.026546 | Val Loss: 0.026113
Epoch: 585 | Train Loss: 0.026586 | Val Loss: 0.026066
Epoch: 586 | Train Loss: 0.026648 | Val Loss: 0.026052
Epoch: 587 | Train Loss: 0.026576 | Val Loss: 0.026082
Epoch: 588 | Train Loss: 0.026761 | Val Loss: 0.026127
Epoch: 589 | Train Loss: 0.026607 | Val Loss: 0.026318
Epoch: 590 | Train Loss: 0.026638 | Val Loss: 0.026055
Epoch: 591 | Train Loss: 0.026574 | Val Loss: 0.026055
Epoch: 592 | Train Loss: 0.026602 | Val Loss: 0.026130
Epoch: 593 | Train Loss: 0.026495 | Val Loss: 0.026148
Epoch: 594 | Train Loss: 0.026526 | Val Loss: 0.025974
Epoch: 595 | Train Loss: 0.026541 | Val Loss: 0.025871
Epoch: 596 | Train Loss: 0.026529 | Val Loss: 0.025836
Epoch: 597 | Train Loss: 0.026593 | Val Loss: 0.026131
Epoch: 598 | Train Loss: 0.026624 | Val Loss: 0.026093
Epoch: 599 | Train Loss: 0.026637 | Val Loss: 0.026224
Epoch: 600 | Train Loss: 0.026641 | Val Loss: 0.026330
Epoch: 601 | Train Loss: 0.026510 | Val Loss: 0.026053
Epoch: 602 | Train Loss: 0.026547 | Val Loss: 0.026019
Epoch: 603 | Train Loss: 0.026524 | Val Loss: 0.026049
Epoch: 604 | Train Loss: 0.026467 | Val Loss: 0.026002
Epoch: 605 | Train Loss: 0.026619 | Val Loss: 0.025967
Epoch: 606 | Train Loss: 0.026575 | Val Loss: 0.026033
Epoch: 607 | Train Loss: 0.026656 | Val Loss: 0.026116
Epoch: 608 | Train Loss: 0.026477 | Val Loss: 0.026069
Epoch: 609 | Train Loss: 0.026702 | Val Loss: 0.026050
Epoch: 610 | Train Loss: 0.026464 | Val Loss: 0.026496
Epoch: 611 | Train Loss: 0.026487 | Val Loss: 0.026184
Epoch: 612 | Train Loss: 0.026450 | Val Loss: 0.026454
Epoch: 613 | Train Loss: 0.026529 | Val Loss: 0.026423
Epoch: 614 | Train Loss: 0.026590 | Val Loss: 0.026050
Epoch: 615 | Train Loss: 0.026541 | Val Loss: 0.026275
Epoch: 616 | Train Loss: 0.026576 | Val Loss: 0.026019
Epoch: 617 | Train Loss: 0.026462 | Val Loss: 0.026034
Epoch: 618 | Train Loss: 0.026525 | Val Loss: 0.026062
Epoch: 619 | Train Loss: 0.026512 | Val Loss: 0.026237
Epoch: 620 | Train Loss: 0.026544 | Val Loss: 0.026113
Epoch: 621 | Train Loss: 0.026364 | Val Loss: 0.025860
Epoch: 622 | Train Loss: 0.026424 | Val Loss: 0.026141
Epoch: 623 | Train Loss: 0.026445 | Val Loss: 0.026076
Epoch: 624 | Train Loss: 0.026628 | Val Loss: 0.026126
Epoch: 625 | Train Loss: 0.026526 | Val Loss: 0.026250
Epoch: 626 | Train Loss: 0.026480 | Val Loss: 0.025982
Epoch: 627 | Train Loss: 0.026475 | Val Loss: 0.026021
Epoch: 628 | Train Loss: 0.026557 | Val Loss: 0.026163
Epoch: 629 | Train Loss: 0.026313 | Val Loss: 0.025871
Epoch: 630 | Train Loss: 0.026488 | Val Loss: 0.026010
Epoch: 631 | Train Loss: 0.026477 | Val Loss: 0.026091
Epoch: 632 | Train Loss: 0.026453 | Val Loss: 0.025887
Epoch: 633 | Train Loss: 0.026480 | Val Loss: 0.026115
Epoch: 634 | Train Loss: 0.026435 | Val Loss: 0.026088
Epoch: 635 | Train Loss: 0.026422 | Val Loss: 0.026095
Epoch: 636 | Train Loss: 0.026499 | Val Loss: 0.025902
Epoch: 637 | Train Loss: 0.026454 | Val Loss: 0.026158
Epoch: 638 | Train Loss: 0.026460 | Val Loss: 0.025963
Epoch: 639 | Train Loss: 0.026443 | Val Loss: 0.026061
Epoch: 640 | Train Loss: 0.026443 | Val Loss: 0.025945
Epoch: 641 | Train Loss: 0.026502 | Val Loss: 0.026042
Epoch: 642 | Train Loss: 0.026514 | Val Loss: 0.026053
Epoch: 643 | Train Loss: 0.026503 | Val Loss: 0.025885
Epoch: 644 | Train Loss: 0.026490 | Val Loss: 0.026079
Epoch: 645 | Train Loss: 0.026434 | Val Loss: 0.026114
Epoch: 646 | Train Loss: 0.026441 | Val Loss: 0.026036
Epoch: 647 | Train Loss: 0.026394 | Val Loss: 0.026008
Epoch: 648 | Train Loss: 0.026485 | Val Loss: 0.026049
Epoch: 649 | Train Loss: 0.026252 | Val Loss: 0.026169
```

```
Epoch: 650 | Train Loss: 0.026539 | Val Loss: 0.025931
Epoch: 651 | Train Loss: 0.026335 | Val Loss: 0.026014
Epoch: 652 | Train Loss: 0.026422 | Val Loss: 0.025943
Epoch: 653 | Train Loss: 0.026354 | Val Loss: 0.025976
Epoch: 654 | Train Loss: 0.026403 | Val Loss: 0.025972
Epoch: 655 | Train Loss: 0.026442 | Val Loss: 0.026133
Epoch: 656 | Train Loss: 0.026346 | Val Loss: 0.026072
Epoch: 657 | Train Loss: 0.026445 | Val Loss: 0.025960
Epoch: 658 | Train Loss: 0.026267 | Val Loss: 0.025909
Epoch: 659 | Train Loss: 0.026332 | Val Loss: 0.026232
Epoch: 660 | Train Loss: 0.026317 | Val Loss: 0.025949
Epoch: 661 | Train Loss: 0.026392 | Val Loss: 0.026145
Epoch: 662 | Train Loss: 0.026361 | Val Loss: 0.025977
Epoch: 663 | Train Loss: 0.026358 | Val Loss: 0.026024
Epoch: 664 | Train Loss: 0.026271 | Val Loss: 0.025816
Epoch: 665 | Train Loss: 0.026408 | Val Loss: 0.026080
Epoch: 666 | Train Loss: 0.026417 | Val Loss: 0.025857
Epoch: 667 | Train Loss: 0.026413 | Val Loss: 0.026019
Epoch: 668 | Train Loss: 0.026413 | Val Loss: 0.025913
Epoch: 669 | Train Loss: 0.026371 | Val Loss: 0.025940
Epoch: 670 | Train Loss: 0.026475 | Val Loss: 0.025783
Epoch: 671 | Train Loss: 0.026387 | Val Loss: 0.026020
Epoch: 672 | Train Loss: 0.026212 | Val Loss: 0.025942
Epoch: 673 | Train Loss: 0.026436 | Val Loss: 0.026045
Epoch: 674 | Train Loss: 0.026322 | Val Loss: 0.026058
Epoch: 675 | Train Loss: 0.026374 | Val Loss: 0.025884
Epoch: 676 | Train Loss: 0.026368 | Val Loss: 0.026032
Epoch: 677 | Train Loss: 0.026337 | Val Loss: 0.025862
Epoch: 678 | Train Loss: 0.026321 | Val Loss: 0.025748
Epoch: 679 | Train Loss: 0.026280 | Val Loss: 0.026027
Epoch: 680 | Train Loss: 0.026434 | Val Loss: 0.025935
Epoch: 681 | Train Loss: 0.026327 | Val Loss: 0.025942
Epoch: 682 | Train Loss: 0.026299 | Val Loss: 0.025965
Epoch: 683 | Train Loss: 0.026291 | Val Loss: 0.025768
Epoch: 684 | Train Loss: 0.026437 | Val Loss: 0.025934
Epoch: 685 | Train Loss: 0.026418 | Val Loss: 0.025947
Epoch: 686 | Train Loss: 0.026329 | Val Loss: 0.026079
Epoch: 687 | Train Loss: 0.026232 | Val Loss: 0.025835
Epoch: 688 | Train Loss: 0.026347 | Val Loss: 0.025814
Epoch: 689 | Train Loss: 0.026287 | Val Loss: 0.025885
Epoch: 690 | Train Loss: 0.026218 | Val Loss: 0.025754
Epoch: 691 | Train Loss: 0.026385 | Val Loss: 0.025802
Epoch: 692 | Train Loss: 0.026267 | Val Loss: 0.026056
Epoch: 693 | Train Loss: 0.026252 | Val Loss: 0.025863
Epoch: 694 | Train Loss: 0.026281 | Val Loss: 0.025911
Epoch: 695 | Train Loss: 0.026262 | Val Loss: 0.025772
Epoch: 696 | Train Loss: 0.026217 | Val Loss: 0.025848
Epoch: 697 | Train Loss: 0.026265 | Val Loss: 0.026068
Epoch: 698 | Train Loss: 0.026422 | Val Loss: 0.026155
Epoch: 699 | Train Loss: 0.026294 | Val Loss: 0.025919
Epoch: 700 | Train Loss: 0.026213 | Val Loss: 0.025807
Epoch: 701 | Train Loss: 0.026231 | Val Loss: 0.026003
Epoch: 702 | Train Loss: 0.026344 | Val Loss: 0.025741
Epoch: 703 | Train Loss: 0.026227 | Val Loss: 0.025983
Epoch: 704 | Train Loss: 0.026346 | Val Loss: 0.025898
Epoch: 705 | Train Loss: 0.026373 | Val Loss: 0.025822
Epoch: 706 | Train Loss: 0.026249 | Val Loss: 0.025834
Epoch: 707 | Train Loss: 0.026223 | Val Loss: 0.025973
Epoch: 708 | Train Loss: 0.026403 | Val Loss: 0.025860
Epoch: 709 | Train Loss: 0.026280 | Val Loss: 0.026080
Epoch: 710 | Train Loss: 0.026129 | Val Loss: 0.025820
Epoch: 711 | Train Loss: 0.026285 | Val Loss: 0.025908
Epoch: 712 | Train Loss: 0.026246 | Val Loss: 0.026010
Epoch: 713 | Train Loss: 0.026307 | Val Loss: 0.025847
Epoch: 714 | Train Loss: 0.026311 | Val Loss: 0.025806
Epoch: 715 | Train Loss: 0.026312 | Val Loss: 0.025869
Epoch: 716 | Train Loss: 0.026307 | Val Loss: 0.025813
Epoch: 717 | Train Loss: 0.026336 | Val Loss: 0.025803
Epoch: 718 | Train Loss: 0.026297 | Val Loss: 0.025969
```

```
Epoch: 719 | Train Loss: 0.026318 | Val Loss: 0.025785
Epoch: 720 | Train Loss: 0.026258 | Val Loss: 0.025793
Epoch: 721 | Train Loss: 0.026202 | Val Loss: 0.025778
Epoch: 722 | Train Loss: 0.026220 | Val Loss: 0.025836
Epoch: 723 | Train Loss: 0.026181 | Val Loss: 0.025918
Epoch: 724 | Train Loss: 0.026325 | Val Loss: 0.025921
Epoch: 725 | Train Loss: 0.026174 | Val Loss: 0.026016
Epoch: 726 | Train Loss: 0.026277 | Val Loss: 0.025860
Epoch: 727 | Train Loss: 0.026227 | Val Loss: 0.025709
Epoch: 728 | Train Loss: 0.026236 | Val Loss: 0.025915
Epoch: 729 | Train Loss: 0.026200 | Val Loss: 0.025834
Epoch: 730 | Train Loss: 0.026098 | Val Loss: 0.025966
Epoch: 731 | Train Loss: 0.026230 | Val Loss: 0.025775
Epoch: 732 | Train Loss: 0.026146 | Val Loss: 0.026089
Epoch: 733 | Train Loss: 0.026298 | Val Loss: 0.025834
Epoch: 734 | Train Loss: 0.026241 | Val Loss: 0.025725
Epoch: 735 | Train Loss: 0.026216 | Val Loss: 0.025791
Epoch: 736 | Train Loss: 0.026145 | Val Loss: 0.025930
Epoch: 737 | Train Loss: 0.026121 | Val Loss: 0.025897
Epoch: 738 | Train Loss: 0.026221 | Val Loss: 0.025841
Epoch: 739 | Train Loss: 0.026198 | Val Loss: 0.025751
Epoch: 740 | Train Loss: 0.026123 | Val Loss: 0.025980
Epoch: 741 | Train Loss: 0.026248 | Val Loss: 0.025763
Epoch: 742 | Train Loss: 0.026177 | Val Loss: 0.025927
Epoch: 743 | Train Loss: 0.026129 | Val Loss: 0.025840
Epoch: 744 | Train Loss: 0.026149 | Val Loss: 0.025934
Epoch: 745 | Train Loss: 0.026146 | Val Loss: 0.025725
Epoch: 746 | Train Loss: 0.026201 | Val Loss: 0.025805
Epoch: 747 | Train Loss: 0.026269 | Val Loss: 0.025856
Epoch: 748 | Train Loss: 0.026265 | Val Loss: 0.025906
Epoch: 749 | Train Loss: 0.026091 | Val Loss: 0.026026
Epoch: 750 | Train Loss: 0.026244 | Val Loss: 0.025858
Epoch: 751 | Train Loss: 0.026188 | Val Loss: 0.025733
Epoch: 752 | Train Loss: 0.026161 | Val Loss: 0.025791
Epoch: 753 | Train Loss: 0.026193 | Val Loss: 0.025694
Epoch: 754 | Train Loss: 0.026174 | Val Loss: 0.025752
Epoch: 755 | Train Loss: 0.026198 | Val Loss: 0.025879
Epoch: 756 | Train Loss: 0.026190 | Val Loss: 0.025724
Epoch: 757 | Train Loss: 0.026189 | Val Loss: 0.025961
Epoch: 758 | Train Loss: 0.026077 | Val Loss: 0.025704
Epoch: 759 | Train Loss: 0.026275 | Val Loss: 0.025876
Epoch: 760 | Train Loss: 0.026100 | Val Loss: 0.025564
Epoch: 761 | Train Loss: 0.026205 | Val Loss: 0.025925
Epoch: 762 | Train Loss: 0.026160 | Val Loss: 0.025804
Epoch: 763 | Train Loss: 0.026205 | Val Loss: 0.025822
Epoch: 764 | Train Loss: 0.026136 | Val Loss: 0.025865
Epoch: 765 | Train Loss: 0.026265 | Val Loss: 0.025780
Epoch: 766 | Train Loss: 0.026112 | Val Loss: 0.025877
Epoch: 767 | Train Loss: 0.026037 | Val Loss: 0.025744
Epoch: 768 | Train Loss: 0.026149 | Val Loss: 0.025824
Epoch: 769 | Train Loss: 0.026011 | Val Loss: 0.025657
Epoch: 770 | Train Loss: 0.026223 | Val Loss: 0.025729
Epoch: 771 | Train Loss: 0.026109 | Val Loss: 0.025558
Epoch: 772 | Train Loss: 0.026138 | Val Loss: 0.025691
Epoch: 773 | Train Loss: 0.026117 | Val Loss: 0.025637
Epoch: 774 | Train Loss: 0.026182 | Val Loss: 0.025665
Epoch: 775 | Train Loss: 0.026062 | Val Loss: 0.025703
Epoch: 776 | Train Loss: 0.026139 | Val Loss: 0.025617
Epoch: 777 | Train Loss: 0.026112 | Val Loss: 0.025736
Epoch: 778 | Train Loss: 0.026048 | Val Loss: 0.025570
Epoch: 779 | Train Loss: 0.026011 | Val Loss: 0.025703
Epoch: 780 | Train Loss: 0.026109 | Val Loss: 0.025827
Epoch: 781 | Train Loss: 0.026193 | Val Loss: 0.025718
Epoch: 782 | Train Loss: 0.026020 | Val Loss: 0.025797
Epoch: 783 | Train Loss: 0.025886 | Val Loss: 0.025695
Epoch: 784 | Train Loss: 0.026096 | Val Loss: 0.025687
Epoch: 785 | Train Loss: 0.026143 | Val Loss: 0.025698
Epoch: 786 | Train Loss: 0.026007 | Val Loss: 0.025876
Epoch: 787 | Train Loss: 0.026120 | Val Loss: 0.025745
```

```
Epoch: 788 | Train Loss: 0.026031 | Val Loss: 0.025631
Epoch: 789 | Train Loss: 0.026029 | Val Loss: 0.025591
Epoch: 790 | Train Loss: 0.026008 | Val Loss: 0.025811
Epoch: 791 | Train Loss: 0.025935 | Val Loss: 0.025738
Epoch: 792 | Train Loss: 0.026103 | Val Loss: 0.025729
Epoch: 793 | Train Loss: 0.026178 | Val Loss: 0.025644
Epoch: 794 | Train Loss: 0.025972 | Val Loss: 0.025568
Epoch: 795 | Train Loss: 0.026091 | Val Loss: 0.025998
Epoch: 796 | Train Loss: 0.026057 | Val Loss: 0.025817
Epoch: 797 | Train Loss: 0.026008 | Val Loss: 0.025738
Epoch: 798 | Train Loss: 0.026010 | Val Loss: 0.025641
Epoch: 799 | Train Loss: 0.025979 | Val Loss: 0.025609
Epoch: 800 | Train Loss: 0.026004 | Val Loss: 0.025628
Epoch: 801 | Train Loss: 0.026003 | Val Loss: 0.025702
Epoch: 802 | Train Loss: 0.026083 | Val Loss: 0.025662
Epoch: 803 | Train Loss: 0.026055 | Val Loss: 0.025897
Epoch: 804 | Train Loss: 0.026070 | Val Loss: 0.025433
Epoch: 805 | Train Loss: 0.026103 | Val Loss: 0.025752
Epoch: 806 | Train Loss: 0.026083 | Val Loss: 0.025608
Epoch: 807 | Train Loss: 0.026001 | Val Loss: 0.025920
Epoch: 808 | Train Loss: 0.025845 | Val Loss: 0.025663
Epoch: 809 | Train Loss: 0.026024 | Val Loss: 0.025420
Epoch: 810 | Train Loss: 0.026029 | Val Loss: 0.025656
Epoch: 811 | Train Loss: 0.026069 | Val Loss: 0.025608
Epoch: 812 | Train Loss: 0.026000 | Val Loss: 0.025537
Epoch: 813 | Train Loss: 0.025973 | Val Loss: 0.025626
Epoch: 814 | Train Loss: 0.025942 | Val Loss: 0.025546
Epoch: 815 | Train Loss: 0.025878 | Val Loss: 0.025696
Epoch: 816 | Train Loss: 0.026012 | Val Loss: 0.025539
Epoch: 817 | Train Loss: 0.025990 | Val Loss: 0.025587
Epoch: 818 | Train Loss: 0.025890 | Val Loss: 0.025619
Epoch: 819 | Train Loss: 0.025983 | Val Loss: 0.025976
Epoch: 820 | Train Loss: 0.026046 | Val Loss: 0.025589
Epoch: 821 | Train Loss: 0.026075 | Val Loss: 0.025498
Epoch: 822 | Train Loss: 0.025963 | Val Loss: 0.025719
Epoch: 823 | Train Loss: 0.025857 | Val Loss: 0.025625
Epoch: 824 | Train Loss: 0.026002 | Val Loss: 0.025750
Epoch: 825 | Train Loss: 0.026000 | Val Loss: 0.025593
Epoch: 826 | Train Loss: 0.025970 | Val Loss: 0.025697
Epoch: 827 | Train Loss: 0.025867 | Val Loss: 0.025636
Epoch: 828 | Train Loss: 0.025952 | Val Loss: 0.025605
Epoch: 829 | Train Loss: 0.025945 | Val Loss: 0.025639
Epoch: 830 | Train Loss: 0.025964 | Val Loss: 0.025609
Epoch: 831 | Train Loss: 0.025933 | Val Loss: 0.025890
Epoch: 832 | Train Loss: 0.025936 | Val Loss: 0.025597
Epoch: 833 | Train Loss: 0.025969 | Val Loss: 0.025636
Epoch: 834 | Train Loss: 0.025976 | Val Loss: 0.025509
Epoch: 835 | Train Loss: 0.025923 | Val Loss: 0.025562
Epoch: 836 | Train Loss: 0.025898 | Val Loss: 0.025648
Epoch: 837 | Train Loss: 0.025959 | Val Loss: 0.025485
Epoch: 838 | Train Loss: 0.025919 | Val Loss: 0.025655
Epoch: 839 | Train Loss: 0.026041 | Val Loss: 0.025608
Epoch: 840 | Train Loss: 0.025928 | Val Loss: 0.025715
Epoch: 841 | Train Loss: 0.025981 | Val Loss: 0.025660
Epoch: 842 | Train Loss: 0.026011 | Val Loss: 0.025559
Epoch: 843 | Train Loss: 0.025963 | Val Loss: 0.025755
Epoch: 844 | Train Loss: 0.025953 | Val Loss: 0.025421
Epoch: 845 | Train Loss: 0.025820 | Val Loss: 0.025634
Epoch: 846 | Train Loss: 0.025836 | Val Loss: 0.025618
Epoch: 847 | Train Loss: 0.025914 | Val Loss: 0.025732
Epoch: 848 | Train Loss: 0.025911 | Val Loss: 0.025586
Epoch: 849 | Train Loss: 0.025998 | Val Loss: 0.025533
Epoch: 850 | Train Loss: 0.025939 | Val Loss: 0.025411
Epoch: 851 | Train Loss: 0.025966 | Val Loss: 0.025577
Epoch: 852 | Train Loss: 0.025842 | Val Loss: 0.025508
Epoch: 853 | Train Loss: 0.025810 | Val Loss: 0.025520
Epoch: 854 | Train Loss: 0.025973 | Val Loss: 0.025623
Epoch: 855 | Train Loss: 0.026004 | Val Loss: 0.025671
Epoch: 856 | Train Loss: 0.025923 | Val Loss: 0.025664
```

```
Epoch: 857 | Train Loss: 0.025832 | Val Loss: 0.025485
Epoch: 858 | Train Loss: 0.025916 | Val Loss: 0.025764
Epoch: 859 | Train Loss: 0.025840 | Val Loss: 0.025672
Epoch: 860 | Train Loss: 0.025914 | Val Loss: 0.025591
Epoch: 861 | Train Loss: 0.025933 | Val Loss: 0.025570
Epoch: 862 | Train Loss: 0.025824 | Val Loss: 0.025680
Epoch: 863 | Train Loss: 0.026015 | Val Loss: 0.025476
Epoch: 864 | Train Loss: 0.025843 | Val Loss: 0.025569
Epoch: 865 | Train Loss: 0.025880 | Val Loss: 0.025776
Epoch: 866 | Train Loss: 0.025901 | Val Loss: 0.025478
Epoch: 867 | Train Loss: 0.025868 | Val Loss: 0.025408
Epoch: 868 | Train Loss: 0.025951 | Val Loss: 0.025524
Epoch: 869 | Train Loss: 0.025834 | Val Loss: 0.025394
Epoch: 870 | Train Loss: 0.025769 | Val Loss: 0.025519
Epoch: 871 | Train Loss: 0.025872 | Val Loss: 0.025604
Epoch: 872 | Train Loss: 0.025855 | Val Loss: 0.025440
Epoch: 873 | Train Loss: 0.025906 | Val Loss: 0.025669
Epoch: 874 | Train Loss: 0.025836 | Val Loss: 0.025552
Epoch: 875 | Train Loss: 0.025822 | Val Loss: 0.025460
Epoch: 876 | Train Loss: 0.025811 | Val Loss: 0.025615
Epoch: 877 | Train Loss: 0.025859 | Val Loss: 0.025438
Epoch: 878 | Train Loss: 0.025882 | Val Loss: 0.025709
Epoch: 879 | Train Loss: 0.025803 | Val Loss: 0.025527
Epoch: 880 | Train Loss: 0.025790 | Val Loss: 0.025422
Epoch: 881 | Train Loss: 0.025846 | Val Loss: 0.025473
Epoch: 882 | Train Loss: 0.025904 | Val Loss: 0.025403
Epoch: 883 | Train Loss: 0.025803 | Val Loss: 0.025622
Epoch: 884 | Train Loss: 0.025816 | Val Loss: 0.025383
Epoch: 885 | Train Loss: 0.025710 | Val Loss: 0.025541
Epoch: 886 | Train Loss: 0.025802 | Val Loss: 0.025399
Epoch: 887 | Train Loss: 0.025863 | Val Loss: 0.025561
Epoch: 888 | Train Loss: 0.025777 | Val Loss: 0.025676
Epoch: 889 | Train Loss: 0.025845 | Val Loss: 0.025584
Epoch: 890 | Train Loss: 0.025824 | Val Loss: 0.025537
Epoch: 891 | Train Loss: 0.025868 | Val Loss: 0.025582
Epoch: 892 | Train Loss: 0.025847 | Val Loss: 0.025372
Epoch: 893 | Train Loss: 0.025725 | Val Loss: 0.025577
Epoch: 894 | Train Loss: 0.025635 | Val Loss: 0.025583
Epoch: 895 | Train Loss: 0.025809 | Val Loss: 0.025437
Epoch: 896 | Train Loss: 0.025759 | Val Loss: 0.025544
Epoch: 897 | Train Loss: 0.025846 | Val Loss: 0.025372
Epoch: 898 | Train Loss: 0.025812 | Val Loss: 0.025385
Epoch: 899 | Train Loss: 0.025749 | Val Loss: 0.025449
Epoch: 900 | Train Loss: 0.025889 | Val Loss: 0.025432
Epoch: 901 | Train Loss: 0.025746 | Val Loss: 0.025499
Epoch: 902 | Train Loss: 0.025868 | Val Loss: 0.025506
Epoch: 903 | Train Loss: 0.025728 | Val Loss: 0.025443
Epoch: 904 | Train Loss: 0.025879 | Val Loss: 0.025457
Epoch: 905 | Train Loss: 0.025823 | Val Loss: 0.025385
Epoch: 906 | Train Loss: 0.025724 | Val Loss: 0.025340
Epoch: 907 | Train Loss: 0.025718 | Val Loss: 0.025470
Epoch: 908 | Train Loss: 0.025762 | Val Loss: 0.025601
Epoch: 909 | Train Loss: 0.025752 | Val Loss: 0.025519
Epoch: 910 | Train Loss: 0.025804 | Val Loss: 0.025415
Epoch: 911 | Train Loss: 0.025793 | Val Loss: 0.025396
Epoch: 912 | Train Loss: 0.025723 | Val Loss: 0.025525
Epoch: 913 | Train Loss: 0.025704 | Val Loss: 0.025551
Epoch: 914 | Train Loss: 0.025847 | Val Loss: 0.025429
Epoch: 915 | Train Loss: 0.025744 | Val Loss: 0.025355
Epoch: 916 | Train Loss: 0.025716 | Val Loss: 0.025357
Epoch: 917 | Train Loss: 0.025656 | Val Loss: 0.025501
Epoch: 918 | Train Loss: 0.025710 | Val Loss: 0.025603
Epoch: 919 | Train Loss: 0.025824 | Val Loss: 0.025389
Epoch: 920 | Train Loss: 0.025769 | Val Loss: 0.025317
Epoch: 921 | Train Loss: 0.025672 | Val Loss: 0.025332
Epoch: 922 | Train Loss: 0.025687 | Val Loss: 0.025353
Epoch: 923 | Train Loss: 0.025657 | Val Loss: 0.025395
Epoch: 924 | Train Loss: 0.025826 | Val Loss: 0.025496
Epoch: 925 | Train Loss: 0.025680 | Val Loss: 0.025327
```

```
Epoch: 926 | Train Loss: 0.025675 | Val Loss: 0.025456
Epoch: 927 | Train Loss: 0.025710 | Val Loss: 0.025358
Epoch: 928 | Train Loss: 0.025716 | Val Loss: 0.025540
Epoch: 929 | Train Loss: 0.025700 | Val Loss: 0.025467
Epoch: 930 | Train Loss: 0.025601 | Val Loss: 0.025291
Epoch: 931 | Train Loss: 0.025666 | Val Loss: 0.025658
Epoch: 932 | Train Loss: 0.025684 | Val Loss: 0.025358
Epoch: 933 | Train Loss: 0.025696 | Val Loss: 0.025506
Epoch: 934 | Train Loss: 0.025600 | Val Loss: 0.025362
Epoch: 935 | Train Loss: 0.025581 | Val Loss: 0.025443
Epoch: 936 | Train Loss: 0.025656 | Val Loss: 0.025514
Epoch: 937 | Train Loss: 0.025608 | Val Loss: 0.025281
Epoch: 938 | Train Loss: 0.025571 | Val Loss: 0.025353
Epoch: 939 | Train Loss: 0.025681 | Val Loss: 0.025395
Epoch: 940 | Train Loss: 0.025631 | Val Loss: 0.025383
Epoch: 941 | Train Loss: 0.025551 | Val Loss: 0.025377
Epoch: 942 | Train Loss: 0.025678 | Val Loss: 0.025439
Epoch: 943 | Train Loss: 0.025675 | Val Loss: 0.025398
Epoch: 944 | Train Loss: 0.025664 | Val Loss: 0.025326
Epoch: 945 | Train Loss: 0.025592 | Val Loss: 0.025427
Epoch: 946 | Train Loss: 0.025674 | Val Loss: 0.025679
Epoch: 947 | Train Loss: 0.025605 | Val Loss: 0.025486
Epoch: 948 | Train Loss: 0.025607 | Val Loss: 0.025479
Epoch: 949 | Train Loss: 0.025591 | Val Loss: 0.025345
Epoch: 950 | Train Loss: 0.025680 | Val Loss: 0.025496
Epoch: 951 | Train Loss: 0.025621 | Val Loss: 0.025192
Epoch: 952 | Train Loss: 0.025755 | Val Loss: 0.025448
Epoch: 953 | Train Loss: 0.025605 | Val Loss: 0.025216
Epoch: 954 | Train Loss: 0.025622 | Val Loss: 0.025205
Epoch: 955 | Train Loss: 0.025663 | Val Loss: 0.025421
Epoch: 956 | Train Loss: 0.025533 | Val Loss: 0.025459
Epoch: 957 | Train Loss: 0.025771 | Val Loss: 0.025473
Epoch: 958 | Train Loss: 0.025616 | Val Loss: 0.025264
Epoch: 959 | Train Loss: 0.025603 | Val Loss: 0.025251
Epoch: 960 | Train Loss: 0.025571 | Val Loss: 0.025271
Epoch: 961 | Train Loss: 0.025498 | Val Loss: 0.025486
Epoch: 962 | Train Loss: 0.025606 | Val Loss: 0.025435
Epoch: 963 | Train Loss: 0.025598 | Val Loss: 0.025541
Epoch: 964 | Train Loss: 0.025573 | Val Loss: 0.025258
Epoch: 965 | Train Loss: 0.025610 | Val Loss: 0.025291
Epoch: 966 | Train Loss: 0.025587 | Val Loss: 0.025348
Epoch: 967 | Train Loss: 0.025550 | Val Loss: 0.025304
Epoch: 968 | Train Loss: 0.025555 | Val Loss: 0.025399
Epoch: 969 | Train Loss: 0.025579 | Val Loss: 0.025384
Epoch: 970 | Train Loss: 0.025556 | Val Loss: 0.025382
Epoch: 971 | Train Loss: 0.025549 | Val Loss: 0.025228
Epoch: 972 | Train Loss: 0.025587 | Val Loss: 0.025293
Epoch: 973 | Train Loss: 0.025617 | Val Loss: 0.025451
Epoch: 974 | Train Loss: 0.025487 | Val Loss: 0.025428
Epoch: 975 | Train Loss: 0.025584 | Val Loss: 0.025447
Epoch: 976 | Train Loss: 0.025528 | Val Loss: 0.025210
Epoch: 977 | Train Loss: 0.025559 | Val Loss: 0.025315
Epoch: 978 | Train Loss: 0.025546 | Val Loss: 0.025472
Epoch: 979 | Train Loss: 0.025556 | Val Loss: 0.025395
Epoch: 980 | Train Loss: 0.025617 | Val Loss: 0.025210
Epoch: 981 | Train Loss: 0.025457 | Val Loss: 0.025372
Epoch: 982 | Train Loss: 0.025419 | Val Loss: 0.025493
Epoch: 983 | Train Loss: 0.025610 | Val Loss: 0.025434
Epoch: 984 | Train Loss: 0.025595 | Val Loss: 0.025392
Epoch: 985 | Train Loss: 0.025456 | Val Loss: 0.025225
Epoch: 986 | Train Loss: 0.025571 | Val Loss: 0.025172
Epoch: 987 | Train Loss: 0.025649 | Val Loss: 0.025526
Epoch: 988 | Train Loss: 0.025501 | Val Loss: 0.025152
Epoch: 989 | Train Loss: 0.025474 | Val Loss: 0.025412
Epoch: 990 | Train Loss: 0.025512 | Val Loss: 0.025096
Epoch: 991 | Train Loss: 0.025528 | Val Loss: 0.025085
Epoch: 992 | Train Loss: 0.025470 | Val Loss: 0.025258
Epoch: 993 | Train Loss: 0.025449 | Val Loss: 0.025326
Epoch: 994 | Train Loss: 0.025495 | Val Loss: 0.025174
```

```
Epoch: 995 | Train Loss: 0.025482 | Val Loss: 0.025171
Epoch: 996 | Train Loss: 0.025400 | Val Loss: 0.025580
Epoch: 997 | Train Loss: 0.025530 | Val Loss: 0.025216
Epoch: 998 | Train Loss: 0.025498 | Val Loss: 0.025340
Epoch: 999 | Train Loss: 0.025578 | Val Loss: 0.025283
Epoch: 1000 | Train Loss: 0.025535 | Val Loss: 0.025227
```

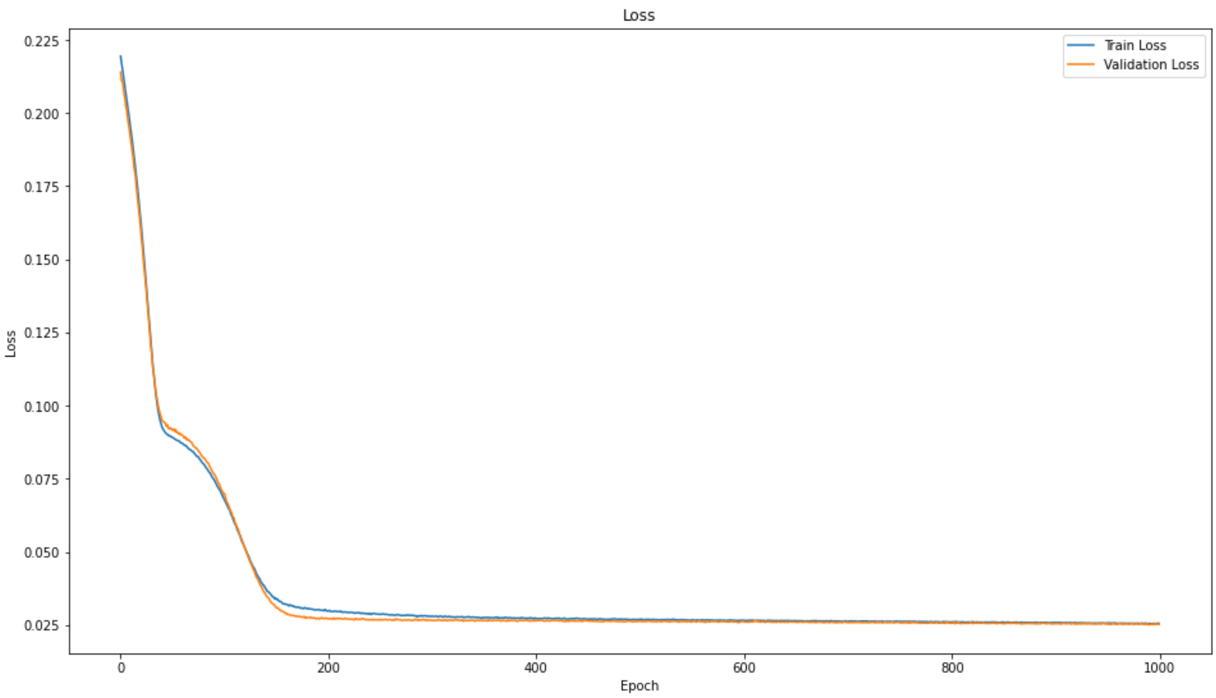
In [47]:

```
'''
train, validation loss를 확인한다
'''
print("train loss list length:", len(trn_loss_list))
print("validation loss list length:", len(val_loss_list))
```

```
train loss list length: 1000
validation loss list length: 1000
```

In [48]:

```
'''
train, validation loss를 시각화한다
'''
plt.figure(figsize = (16, 9))
plt.plot(trn_loss_list, label = 'Train Loss')
plt.plot(val_loss_list, label = 'Validation Loss')
plt.legend(loc = 'upper right')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss')
plt.show()
```



In [49]:

```
'''
test를 진행한다
'''
original = []
result = []
for i, batch in enumerate(test_dataloader):
    x, y = batch
    x = x.view(-1, sequence_length, input_size).to(device)
    y = y.to(device)
    pred = model(x)
    label = y
    loss = criterion(pred, label)
    original.append(y.tolist())
    result.append(pred.tolist())
```

```
print(len(result))
print(len(original))

test_original_np = np.array(sum(sum(original, []), []))
test_result_np = np.array(sum(sum(result, []), []))
```

```
15
15
```

In [50]:

```
'''
reshape dataframe 확인
'''
test_original_df = pd.DataFrame(test_original_np.reshape(-1, 3))
test_result_df = pd.DataFrame(test_result_np.reshape(-1, 3))

print(test_original_df.shape)
print(test_result_df.shape)
```

```
(3651, 3)
(3651, 3)
```

In [51]:

```
'''
결과 데이터를 결합한다
'''
reshaped_test_original_df = pd.concat([x_test_data_df, test_original_df], axis = 1)
reshaped_test_result_df = pd.concat([x_test_data_df, test_result_df], axis = 1)
```

In [52]:

```
'''
데이터를 결합할 때 season column이 중간에 결합되었다

scikit-learn의 inverse transform을 진행하기 위해서는 데이터의 구성이 동일해야하기 때문
'''
reshaped_test_original_df.info
reshaped_test_original_df.columns = ['frontyear', 'backyear', 'month', 'day', 'season
reshaped_test_result_df.columns = ['frontyear', 'backyear', 'month', 'day', 'season',

reshaped_test_original_df = reshaped_test_original_df[['frontyear', 'backyear', 'montł
reshaped_test_result_df = reshaped_test_result_df[['frontyear', 'backyear', 'month',
```

In [53]:

```
print(reshaped_test_original_df.head())
print(reshaped_test_result_df.head())
```

```
   frontyear  backyear  month       day  O_temp_avg  O_temp_min  O_temp_max  ₩
0        1.0   0.10101   0.75  0.193548   -0.050445   -0.161716    0.035354
1        1.0   0.10101   0.75  0.225806    0.652819    0.630363    0.636364
2        1.0   0.10101   0.75  0.258065    0.219585    0.099010    0.318182
3        1.0   0.10101   0.75  0.290323    0.703264    0.739274    0.659091
4        1.0   0.10101   0.75  0.322581    0.860534    0.877888    0.833333

     season
0  0.666667
1  0.666667
2  0.666667
3  0.666667
4  0.666667
   frontyear  backyear  month       day  R_temp_avg  R_temp_min  R_temp_max  ₩
0        1.0   0.10101   0.75  0.193548   -0.027601   -0.160018    0.084288
1        1.0   0.10101   0.75  0.225806    0.385983    0.284728    0.458219
2        1.0   0.10101   0.75  0.258065    0.361920    0.257341    0.437202
3        1.0   0.10101   0.75  0.290323    0.384114    0.282533    0.456736
4        1.0   0.10101   0.75  0.322581    0.679456    0.632526    0.696669

     season
0  0.666667
1  0.666667
2  0.666667
```

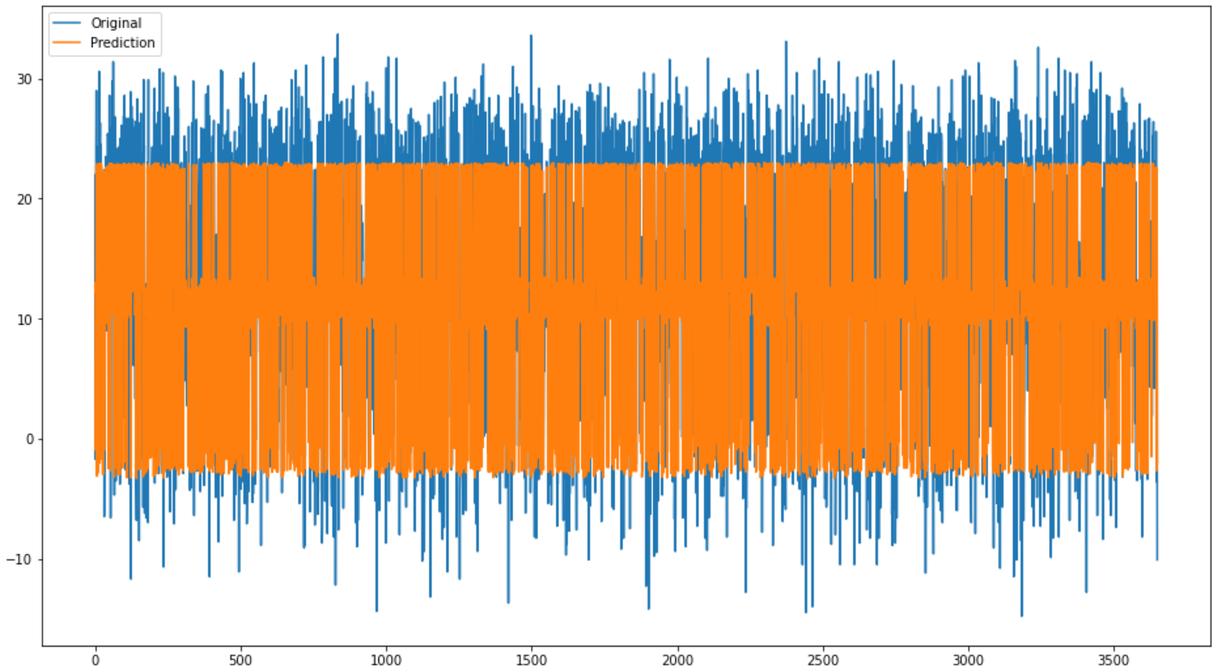
```
3   0.666667
4   0.666667
```

In [54]:
```
'''
예측된 데이터프레임에 inverse transform을 진행한다
'''
inversed_test_original_np = scaler.inverse_transform(reshaped_test_original_df)
inversed_test_original_df = pd.DataFrame(inversed_test_original_np)

inversed_test_result_np = scaler.inverse_transform(reshaped_test_result_df)
inversed_test_result_df = pd.DataFrame(inversed_test_result_np)
```

In [55]:
```
'''
scikit-learn의 scaler를 사용하면 데이터프레임의 column 이름이 초기화되기 때문에 데이터
'''
inversed_test_original_df.columns = [['frontyear', 'backyear', 'month', 'day', 'O_temp
inversed_test_result_df.columns = [['frontyear', 'backyear', 'month', 'day', 'P_temp_a

dropped_test_original_df = inversed_test_original_df[['O_temp_avg', 'O_temp_min', 'O_t
dropped_test_result_df = inversed_test_result_df[['P_temp_avg', 'P_temp_min', 'P_temp_
```

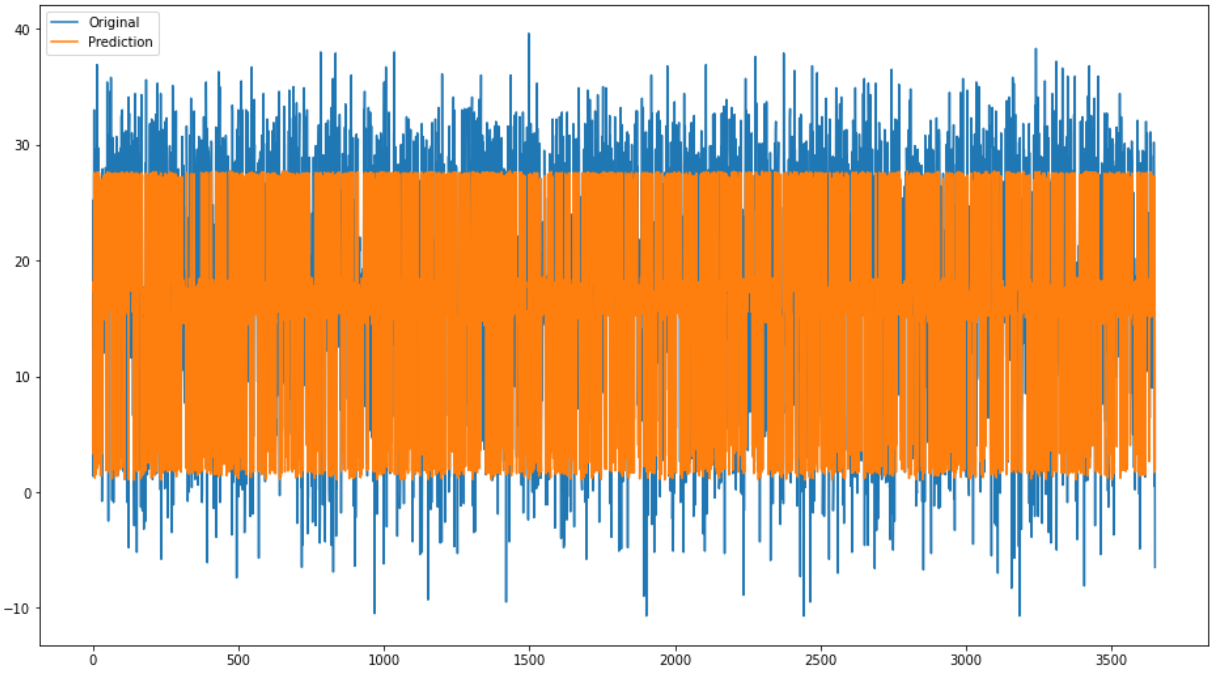
In [56]:
```
'''
결과 데이터를 시각화한다
'''

# average
print("Average temperature")
plt.figure(figsize = (16, 9))
plt.plot(dropped_test_original_df[['O_temp_avg']], label = 'Original')
plt.plot(dropped_test_result_df[['P_temp_avg']], label = 'Prediction')
plt.legend(loc = 'upper left')
plt.show()
```

Average temperature



In [57]:
```
# maximum
print("Maximum temperature")
plt.figure(figsize = (16, 9))
plt.plot(dropped_test_original_df[['O_temp_max']], label = 'Original')
plt.plot(dropped_test_result_df[['P_temp_max']], label = 'Prediction')
plt.legend(loc = 'upper left')
plt.show()
```

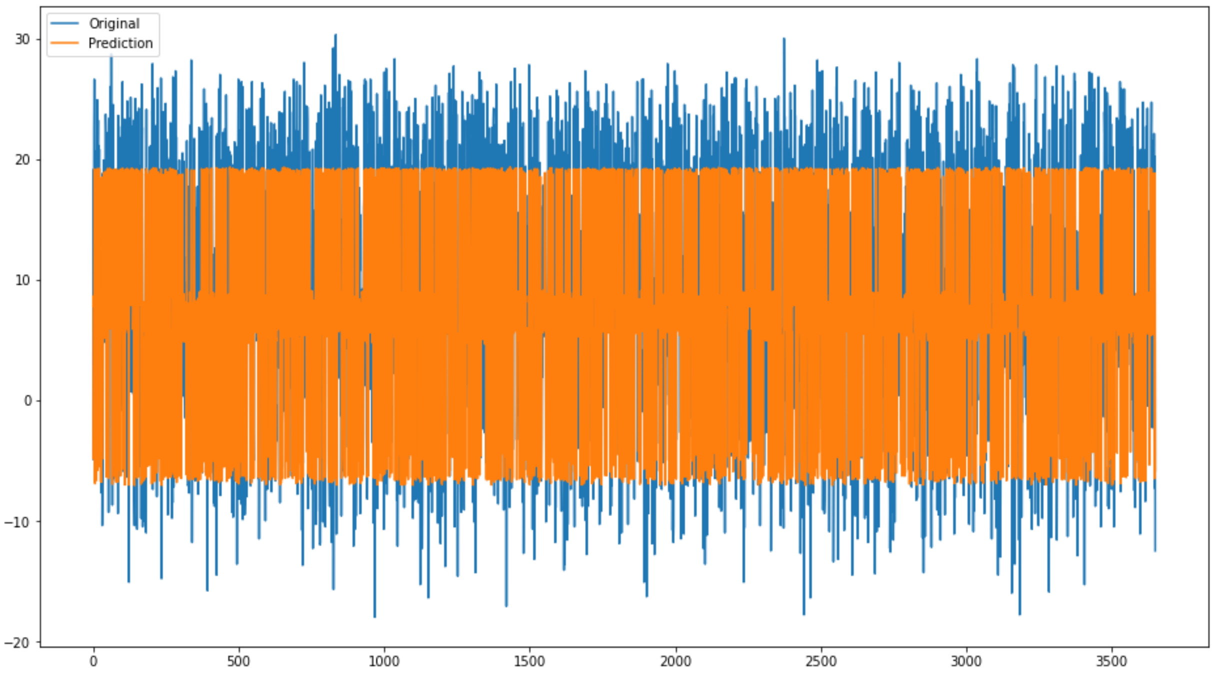
Maximum temperature



In [58]:

```
# minimum
print("Minimum temperature")
plt.figure(figsize = (16, 9))
plt.plot(dropped_test_original_df[['O_temp_min']], label = 'Original')
plt.plot(dropped_test_result_df[['P_temp_min']], label = 'Prediction')
plt.legend(loc = 'upper left')
plt.show()
```

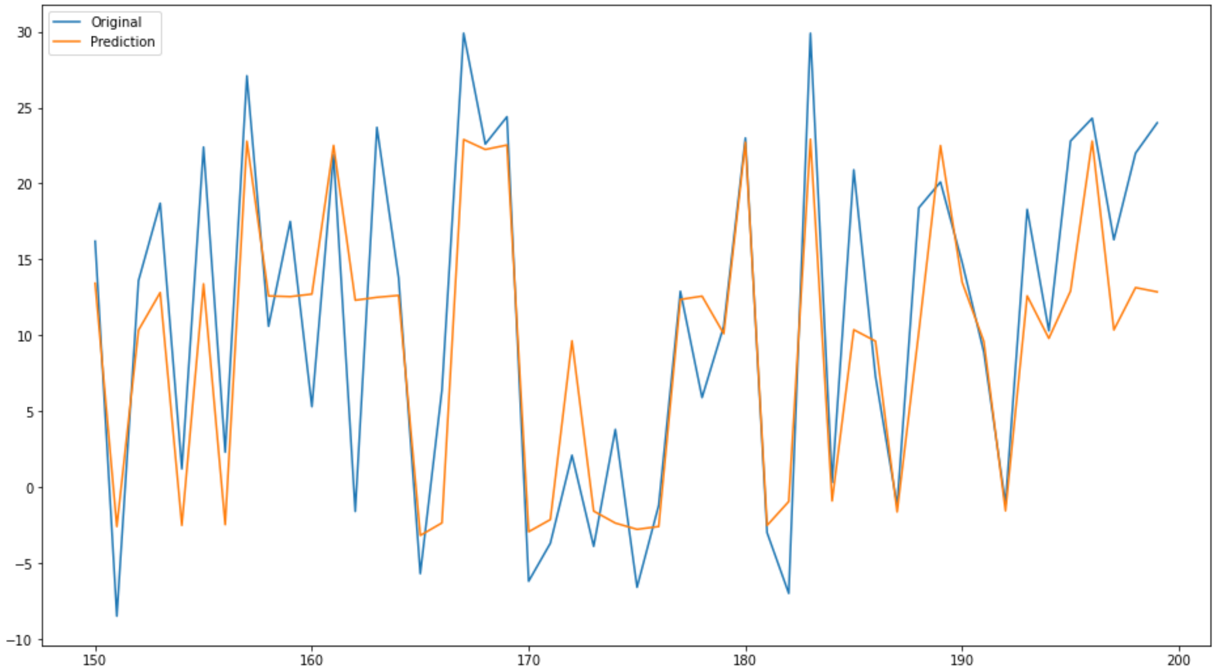
Minimum temperature



In [59]:

```
# average[Ni:Nj]
print("Average temperature")
plt.figure(figsize = (16, 9))
plt.plot(dropped_test_original_df[['O_temp_avg']][150:200], label = 'Original')
plt.plot(dropped_test_result_df[['P_temp_avg']][150:200], label = 'Prediction')
plt.legend(loc = 'upper left')
plt.show()
```

Average temperature



결론

1. 정확도

- ARIMA 기법 등 다른 통계적인 방법을 사용했을 때도 잘 예측하지만, 데이터가 방대해지면 시간이 오래걸림은 물론이고 정확도가 떨어지는 모습이 보이곤 한다. 마지막에 출력된 이미지를 보면 알 수 있다시피, 주기적인 모습은 어느정도 잘 맞추었으나 최대, 최소(Outliers)에 대한 부분은 잘 맞추지 못 한 것을 볼 수 있다.

2. Scaler

- 첫 시작 부분에서 Scaler는 scikit-learn의 MaxAbsScaler() 함수를 이용하였다. 아마 최대 최솟값이 -1 부터 1 사이로 지정되어있기 때문에 Original과 Prediction의 최대, 최소 부분이 차이가 나는 것으로 결론지었다.