딥러닝 과제

RNN 및 LSTM 기반 CNC 공구 마모 예측 분류 모델

20223114 서예은

CONTENTS

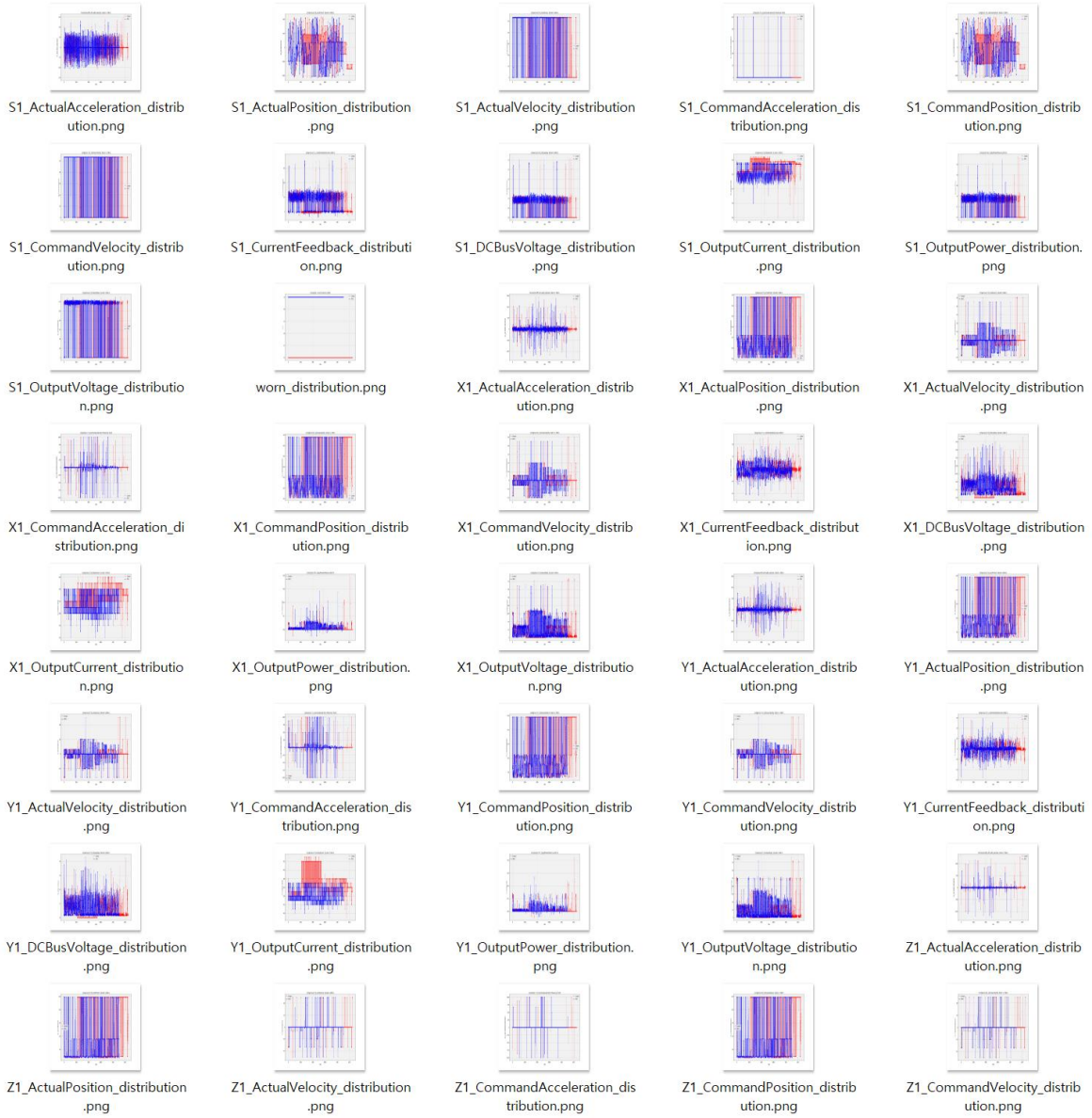
1. 개요

- CNC 밀링 머신을 사용하여 왁스 블록을 대상으로 일련의 가공 실험이 진행
- 다양한 공구 상태, 이송 속도, 및 클램프 압력 조건에서 가공 데이터 수집
- 공구 마모 감지
  - 마모된 공구와 마모되지 않은 공구를 식별하기 위한 감독 학습 이진 분류

▲ CNC

2. 칼럼별 데이터 분포

- BLUE : CNC 공구가 마모되지 않은 상태(worn_df)
  - Experiment 01 ~ 05
- RED : CNC 공구가 마모된 상태 (unworn_df)
  - Experiment 06 ~ 10

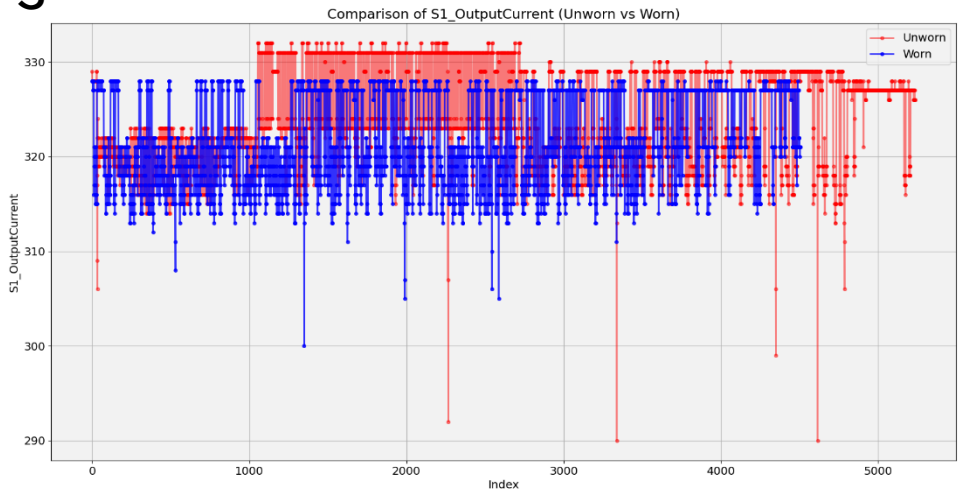


▲ 칼럼별 데이터

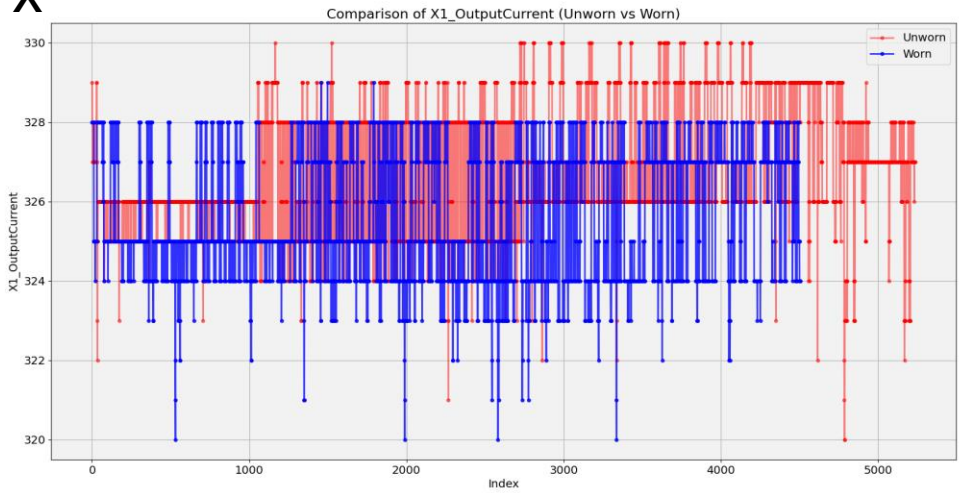
2. 칼럼별 데이터 분포

- OutputCuurent : 출력 전류(A)
  - X, Y축 / 스핀들

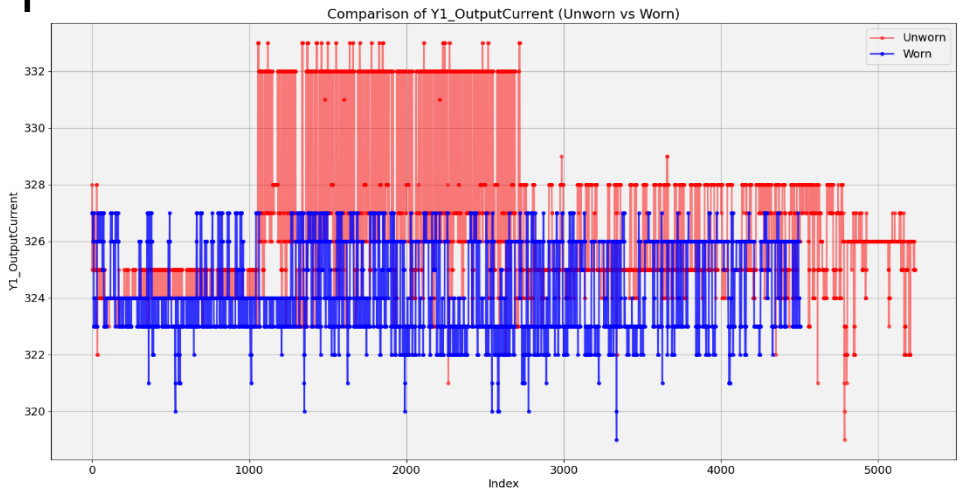
S



X



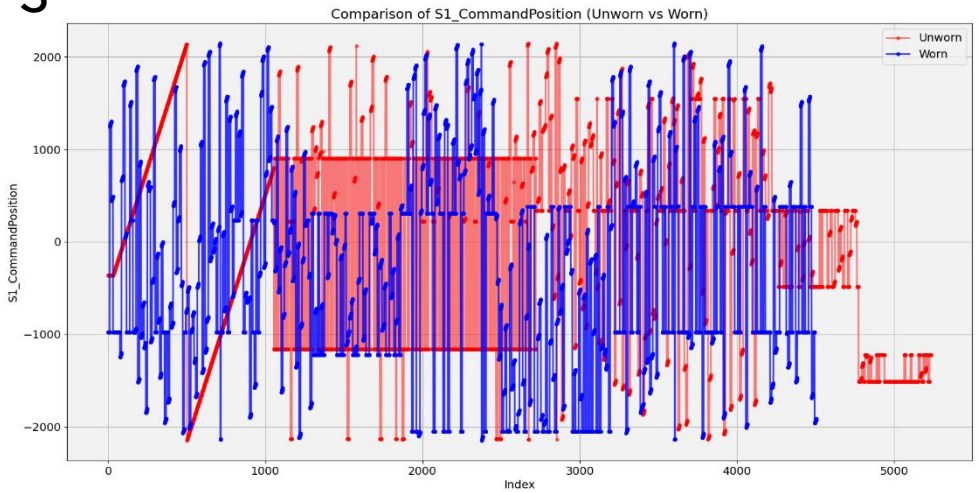
Y



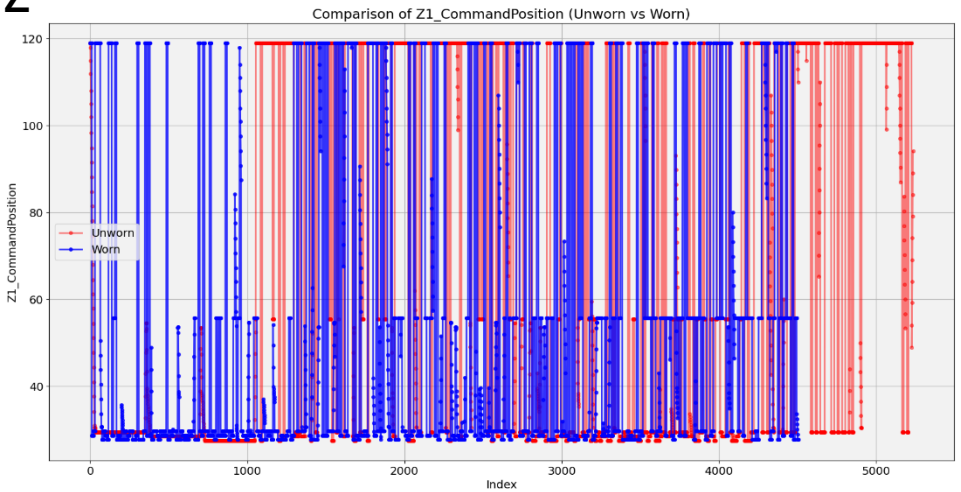
2. 칼럼별 데이터 분포

- Command Position : 기준 위치(mm)

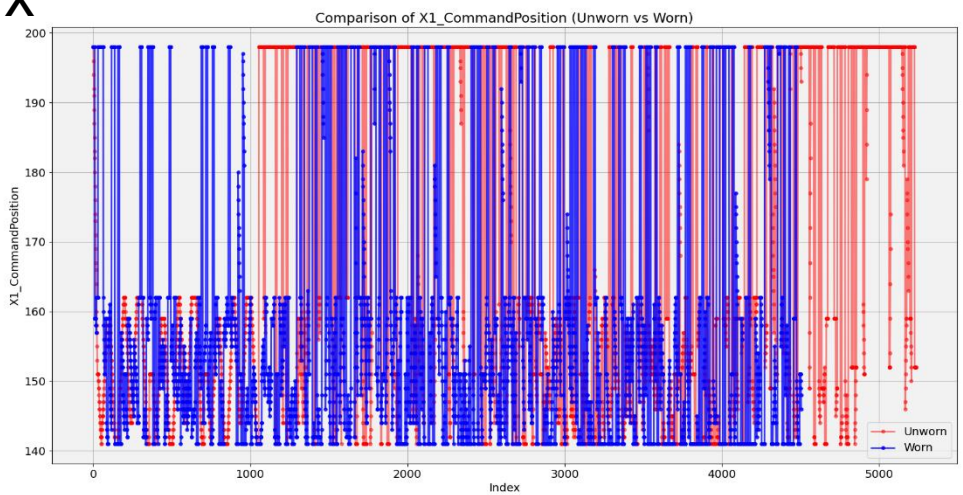
S



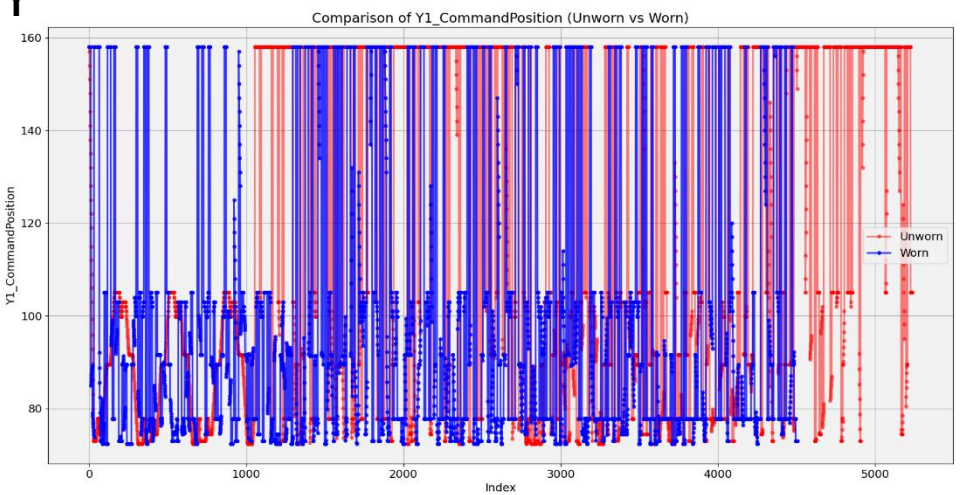
Z



X



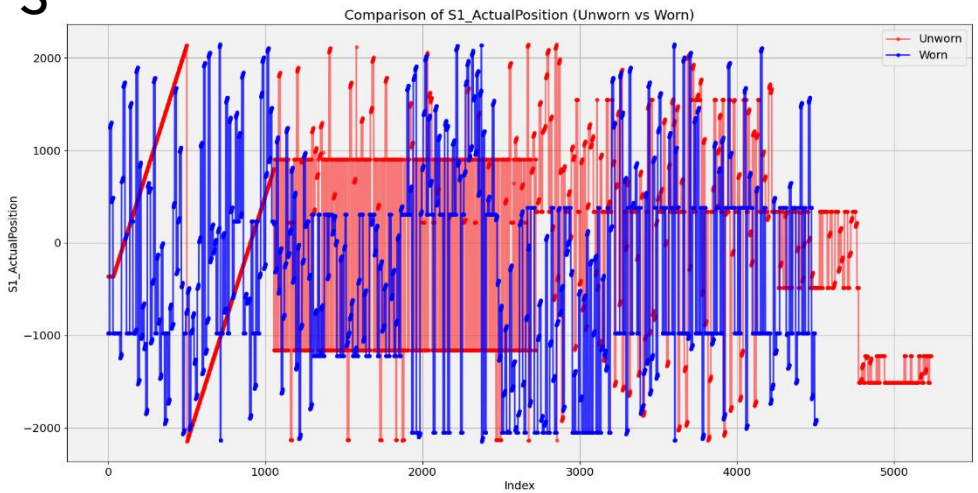
Y



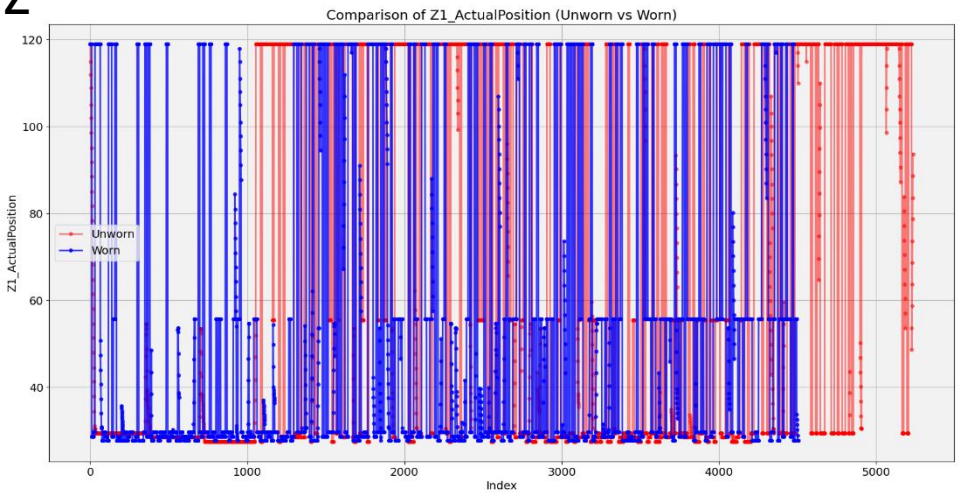
2. 칼럼별 데이터 분포

- Actual Position : 실제 위치(mm)

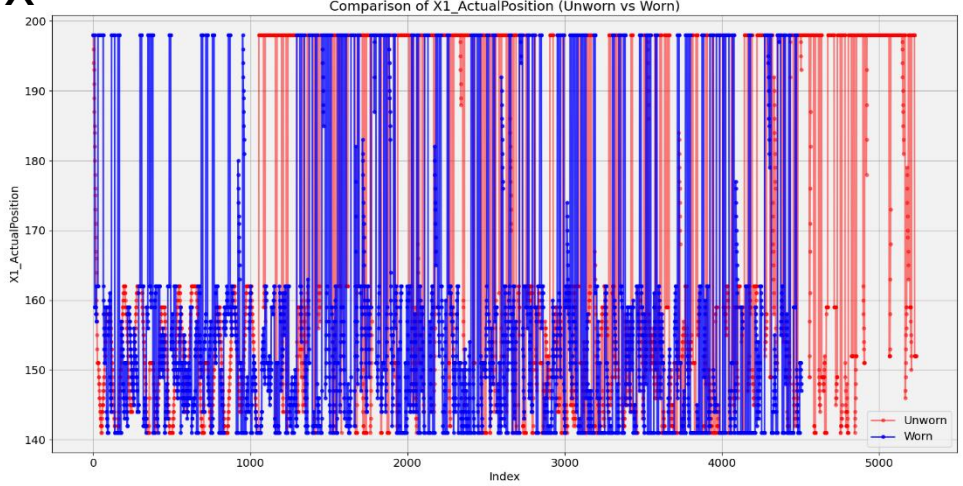
S



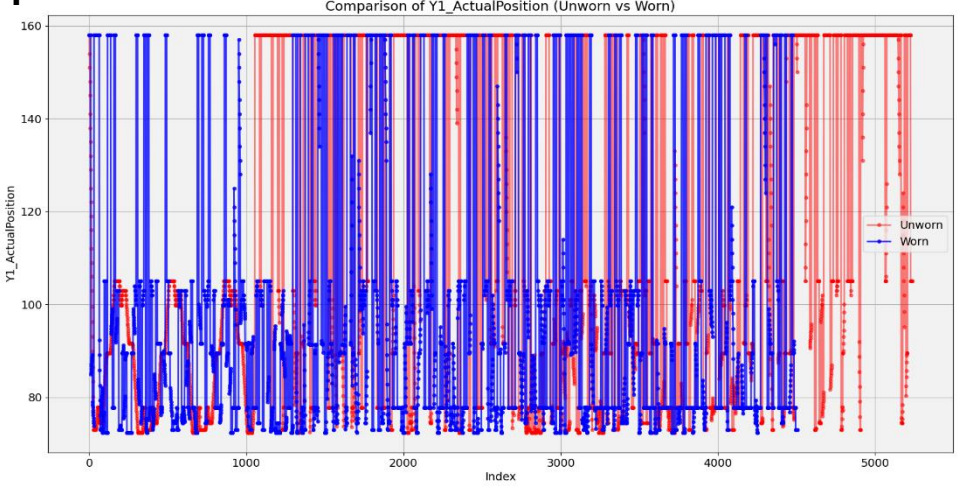
Z



X



Y

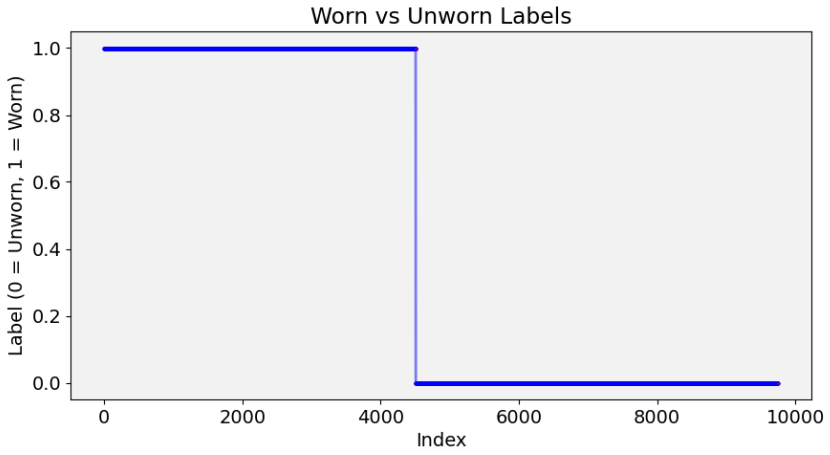


3. 데이터 전처리

- 피처 스케일링
  - 서로 다른 피처 값의 범위가 일치하도록 조정하는 작업
  - 값의 범위가 데이터마다 다르면 모델 훈련이 제대로 되지 않는 원인 제공

```
def Data_scaling(worn, unworn):
    scaler = MinMaxScaler()
    scaler.fit(worn_df)
    worn_scaled = scaler.transform(worn_df)
#WORN
    worn_tensor = torch.tensor(worn_scaled, dtype=torch.float32)
    worn_tensor[:, 39] = 1
#UNWORN
    scaler.fit(unworn_df)
    unworn_scaled = scaler.transform(unworn_df)
    unworn_tensor = torch.tensor(unworn_scaled, dtype=torch.float32)
    unworn_tensor[:, 39] = 0
#MIX
    mix_tensor = torch.cat((worn_tensor, unworn_tensor), dim=0)

    return worn_tensor, unworn_tensor, mix_tensor
```

3. 데이터 전처리




```python
def Data_Isolation(tensor, batch_size):
    target = tensor[:, 39]
    data = tensor[:, :39]

    ...

    if tensor is worn_tensor:
        train_dataset = TensorDataset(sequences_tensor[:4046], targets_tensor[:4046])
        test_dataset = TensorDataset(sequences_tensor[4046:], targets_tensor[4046:])

    elif tensor is unworn_tensor:
        train_dataset = TensorDataset(sequences_tensor[:3937], targets_tensor[:3937])
        test_dataset = TensorDataset(sequences_tensor[3938:], targets_tensor[3938:])

    else:
        test_indices = list(range(4046, 4508)) + list(range(8444, dataset_size))
        train_indices = list(set(range(dataset_size)) - set(test_indices))
        train_dataset = Subset(dataset, train_indices)
        test_dataset = Subset(dataset, test_indices)

    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
    return train_loader, test_loader
```

4. Model - RNN

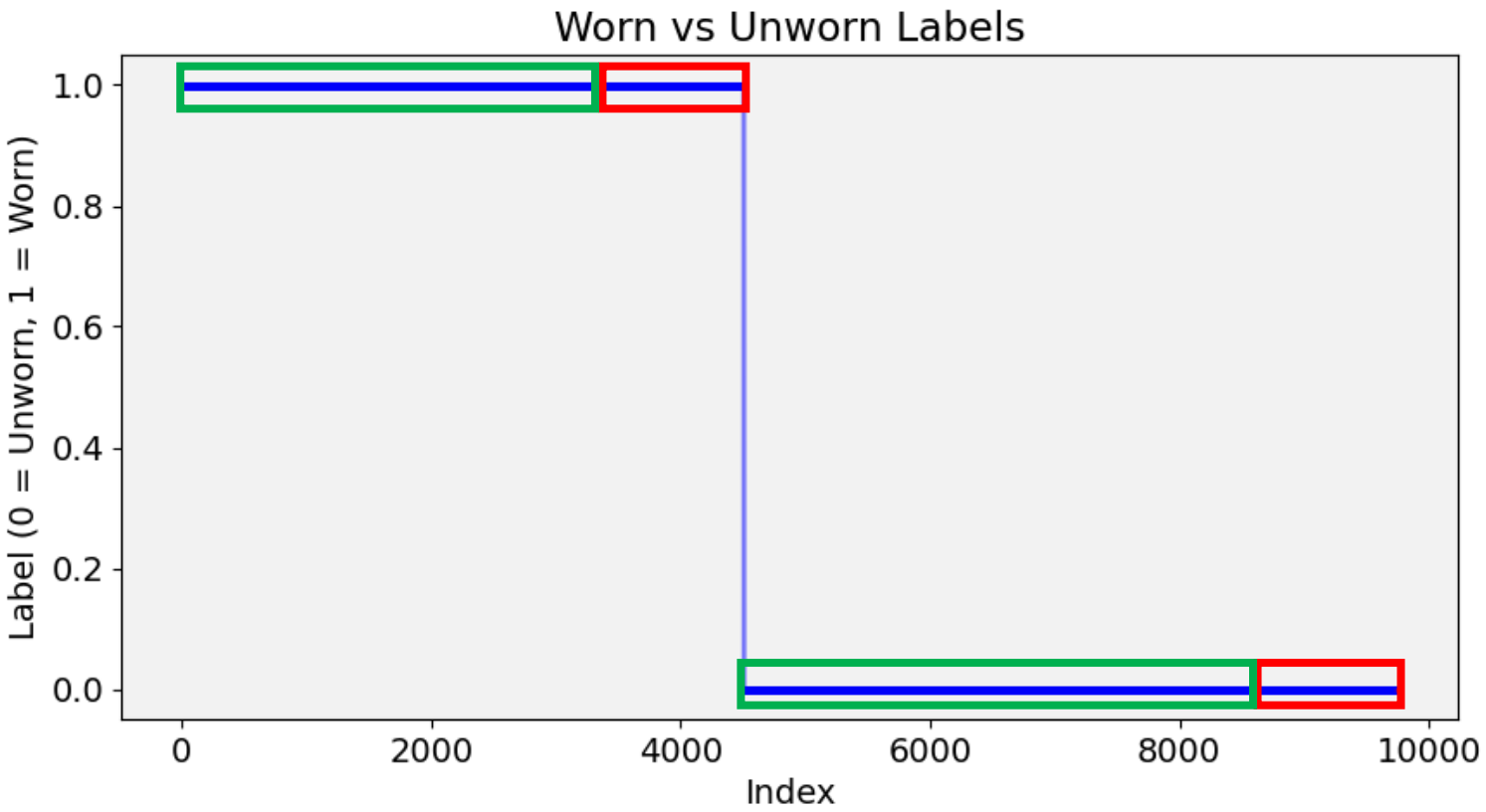
```python
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers, dropout_p):
        super(RNN, self).__init__()
        self.n_layers = num_layers
        self.hidden_dim = hidden_size
        self.rnn = nn.RNN(input_size, hidden_size, num_layers=num_layers, batch_first=True)
        self.dropout = nn.Dropout(dropout_p)
        self.out = nn.Linear(hidden_size, output_size)
        self.init_weights()

    def init_weights(self):
        for name, param in self.named_parameters():
            if 'weight' in name:
                nn.init.kaiming_normal_(param)
            elif 'bias' in name:
                nn.init.zeros_(param)

    def forward(self, x):
        rnn_out, hidden = self.rnn(x)
        last_out = rnn_out[:, -1, :]
        out = self.dropout(last_out)
        out = self.out(out)
        return out

    def predict(self, x):
        with torch.no_grad():
            x = self.forward(x)
        return x

    def reset_state(self):
        self.hidden = None
```

4. Model - LSTM

```python
class LSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers, dropout_p):
        super(LSTM, self).__init__()
        self.n_layers = num_layers
        self.hidden_dim = hidden_size
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers=num_layers, batch_first=True)
        self.dropout = nn.Dropout(dropout_p)
        self.fc = nn.Linear(hidden_size, output_size, bias=True)
        self.init_weights()

    def init_weights(self):
        for name, param in self.named_parameters():
            if 'weight' in name:
                nn.init.kaiming_normal_(param)
            elif 'bias' in name:
                nn.init.zeros_(param)

    def forward(self, x):
        lstm_out, (hidden, cell) = self.lstm(x)
        last_out = lstm_out[:, -1, :]
        out = self.dropout(last_out)
        out = self.fc(out)
        return out

    def predict(self, x):
        with torch.no_grad():
            x = self.forward(x)
        return x

    def reset_state(self):
        self.hidden = None
```

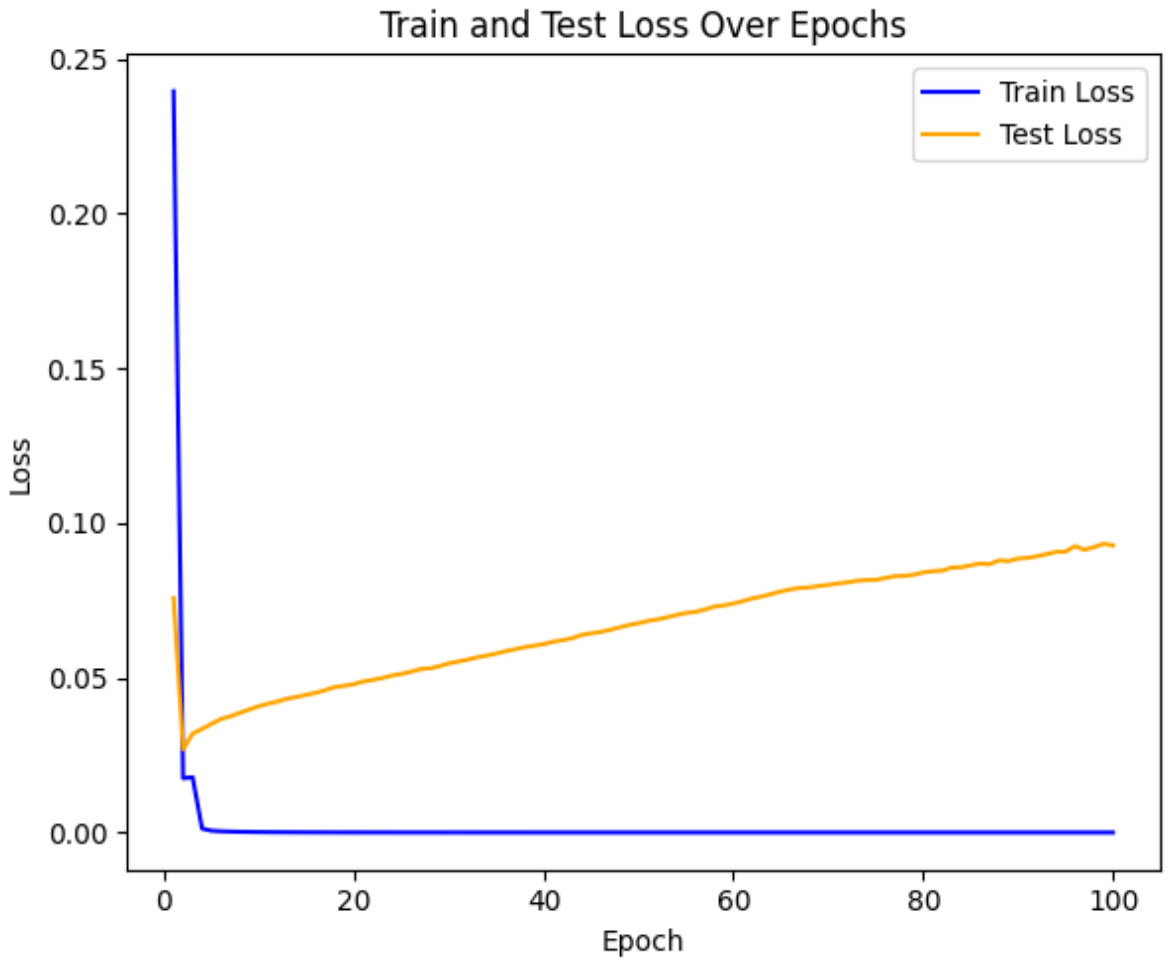
5. 학습 결과

- 입력 데이터
  - Worn, Unworn을 붙인 데이터 생성
  - 각 experiment 05, experiment 10을 TEST
  - 나머지 데이터를 TRAIN



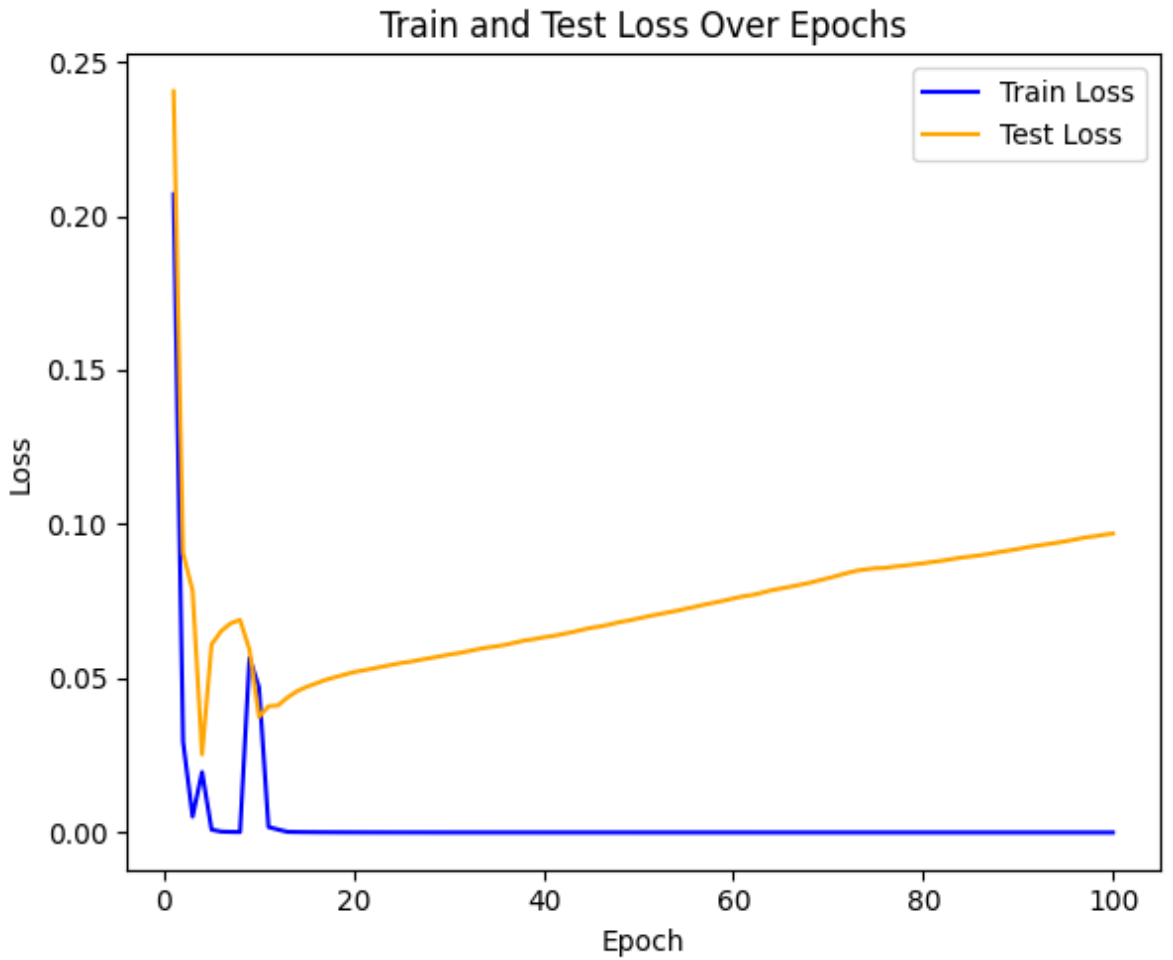
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	64
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



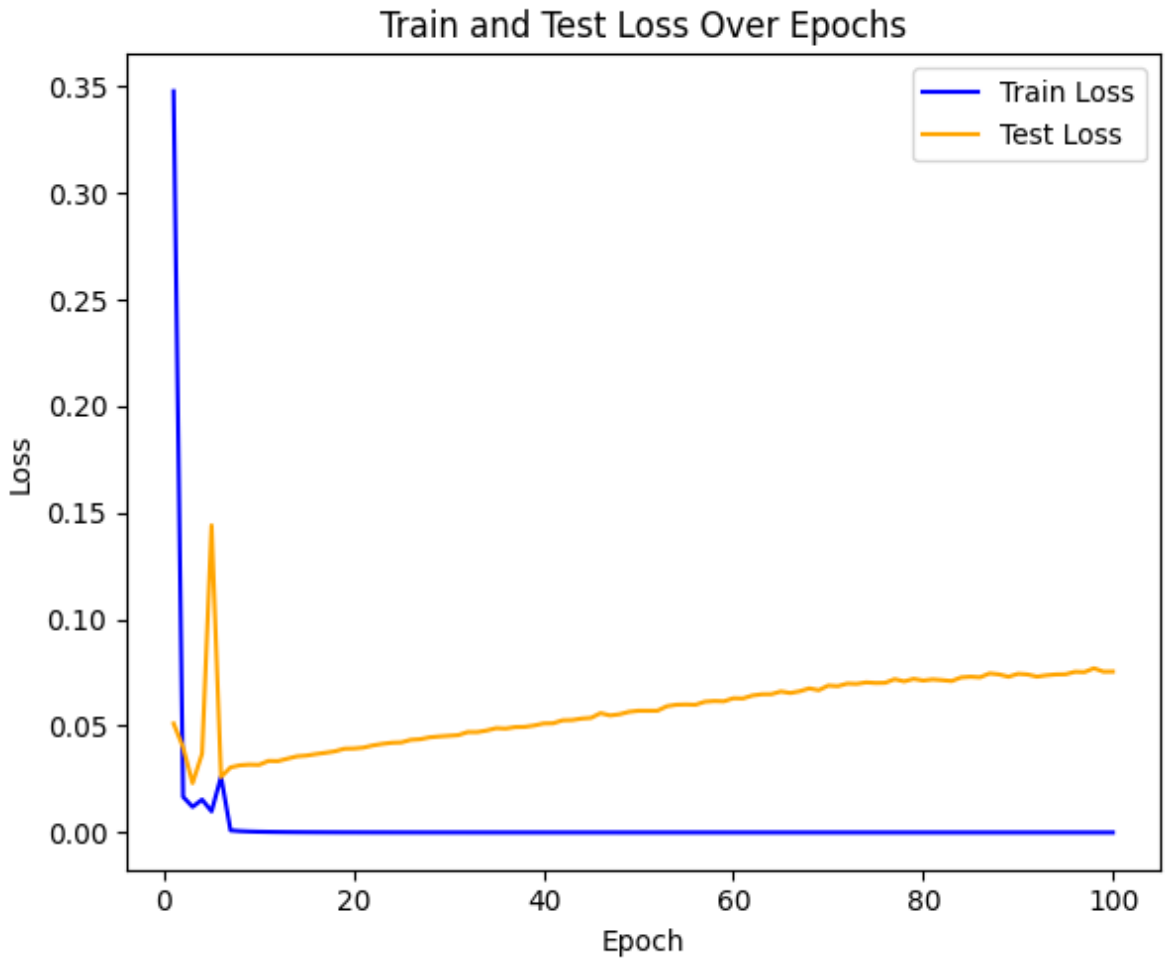
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	80
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



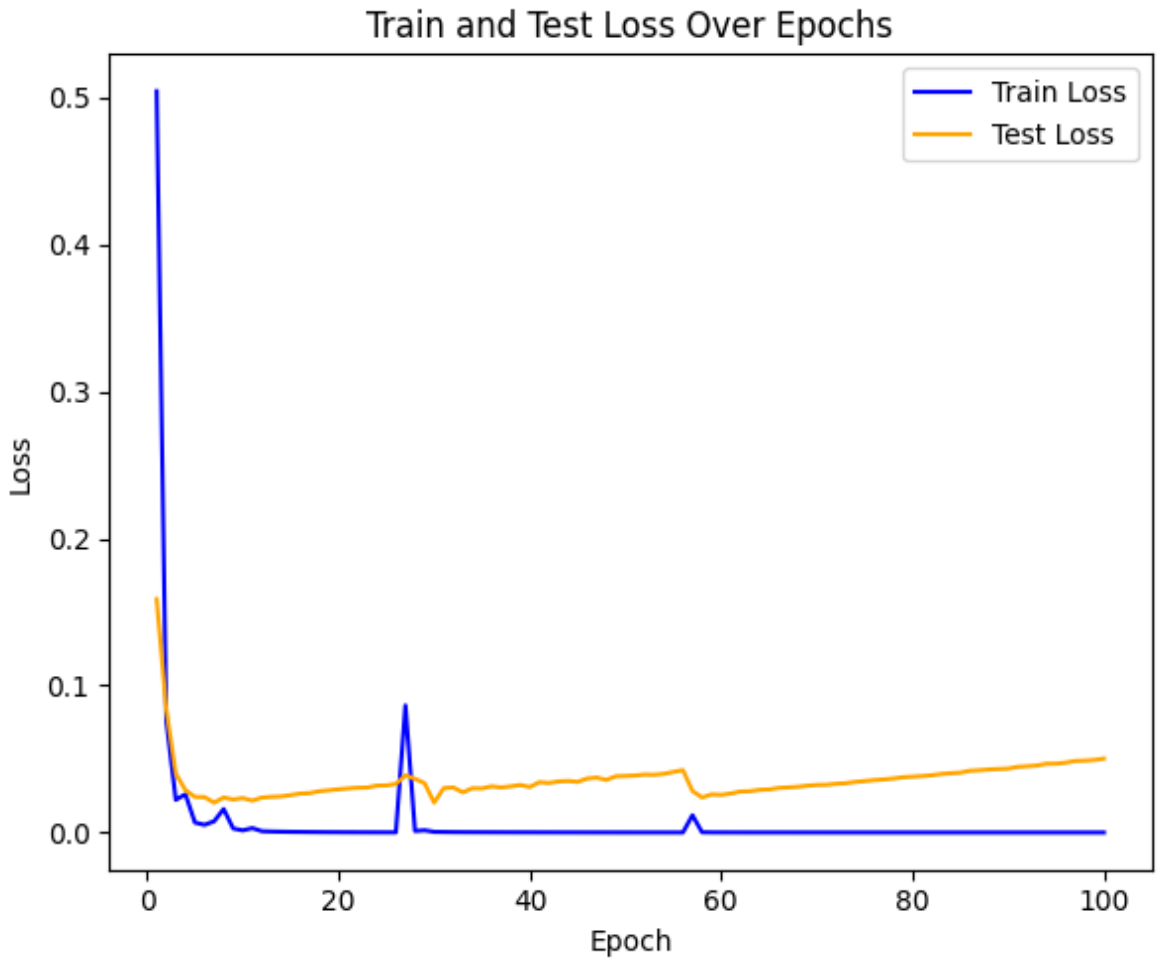
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	30
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



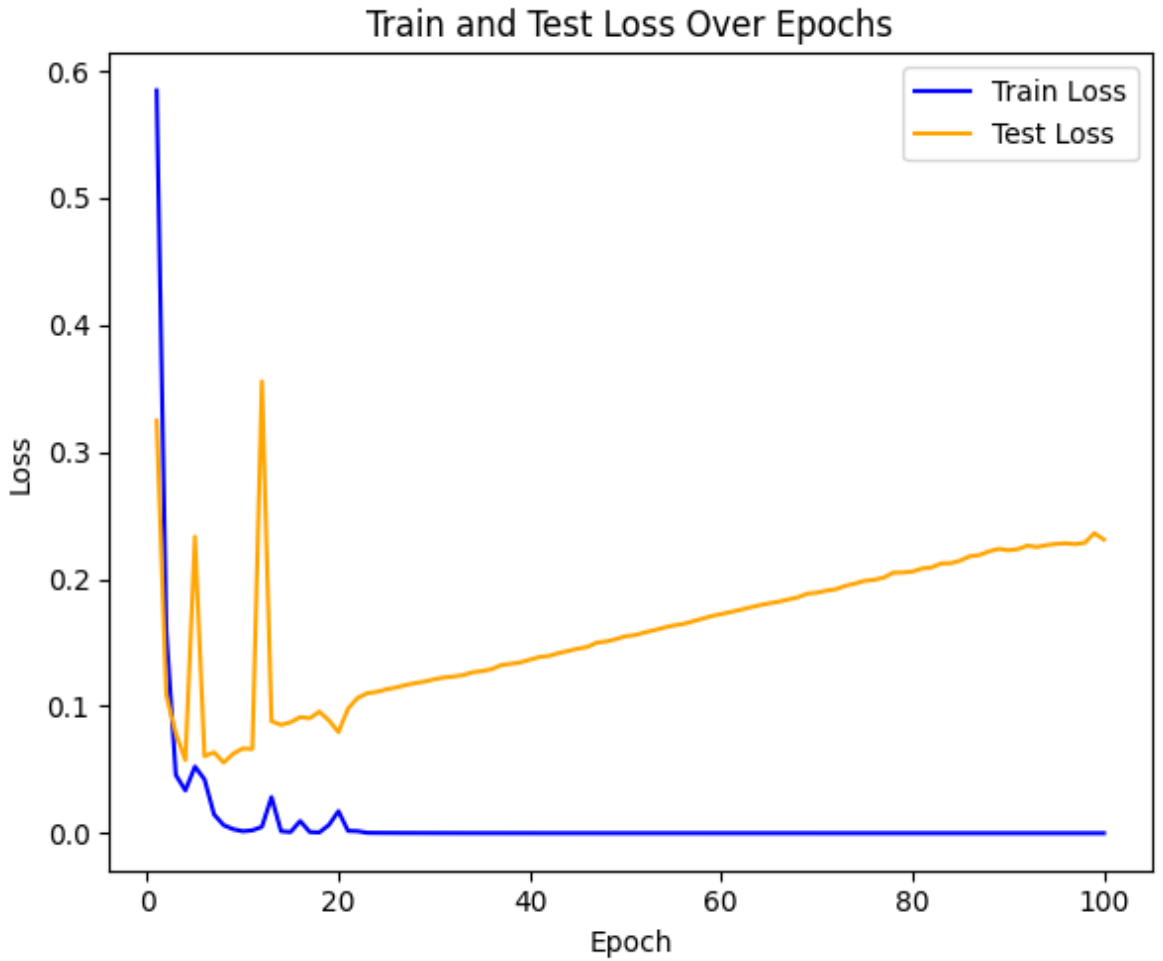
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	16
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



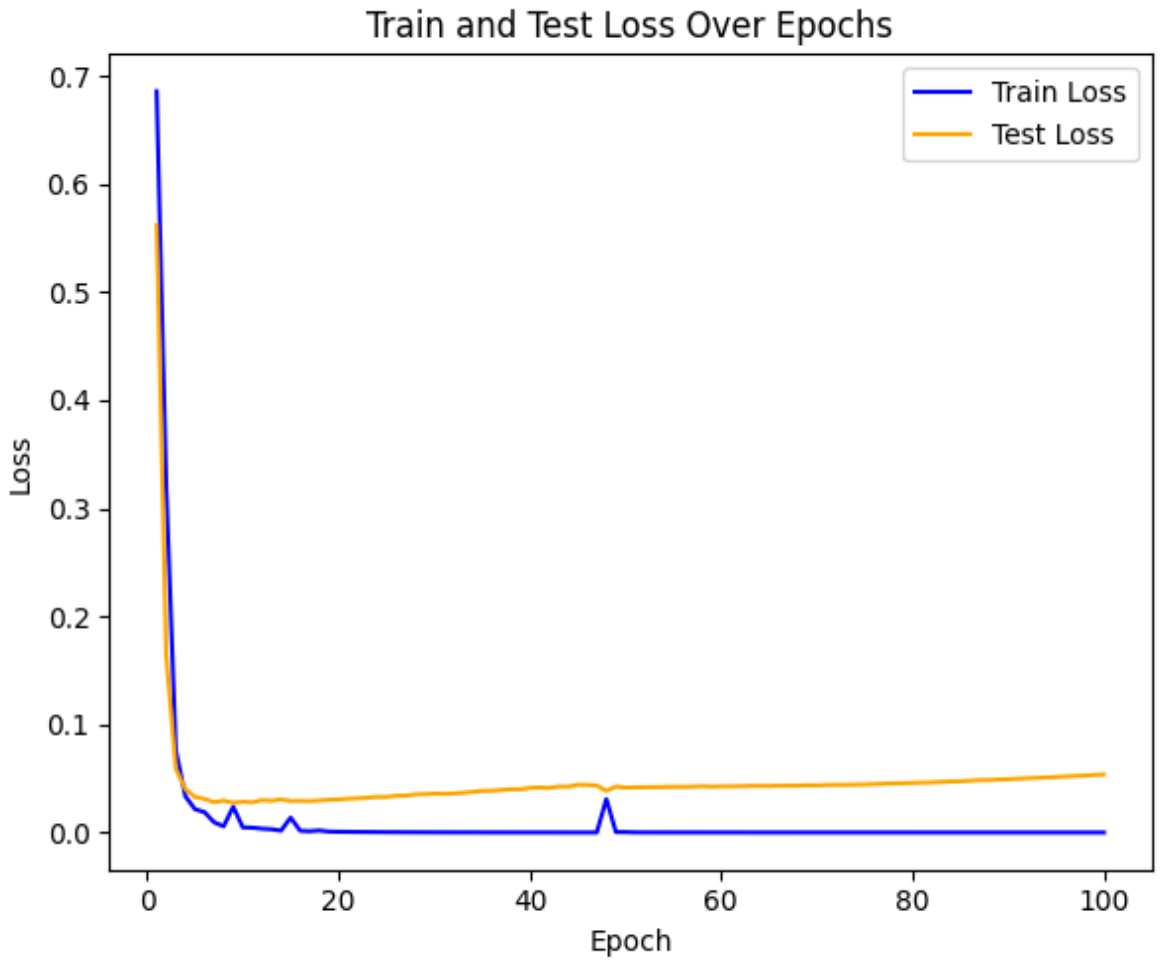
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	16
OUTPUT_SIZE	1
NUM_LAYERS	4
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



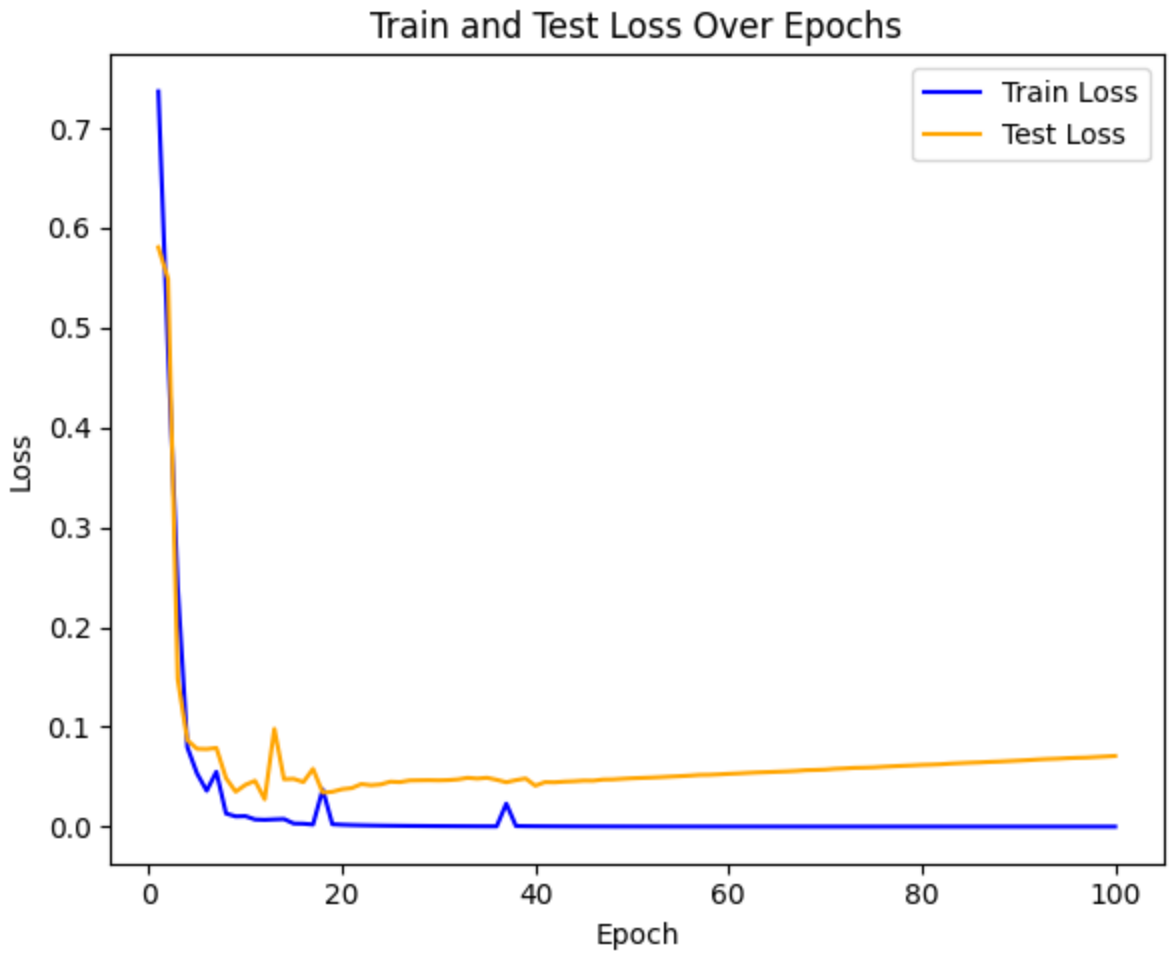
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	8
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



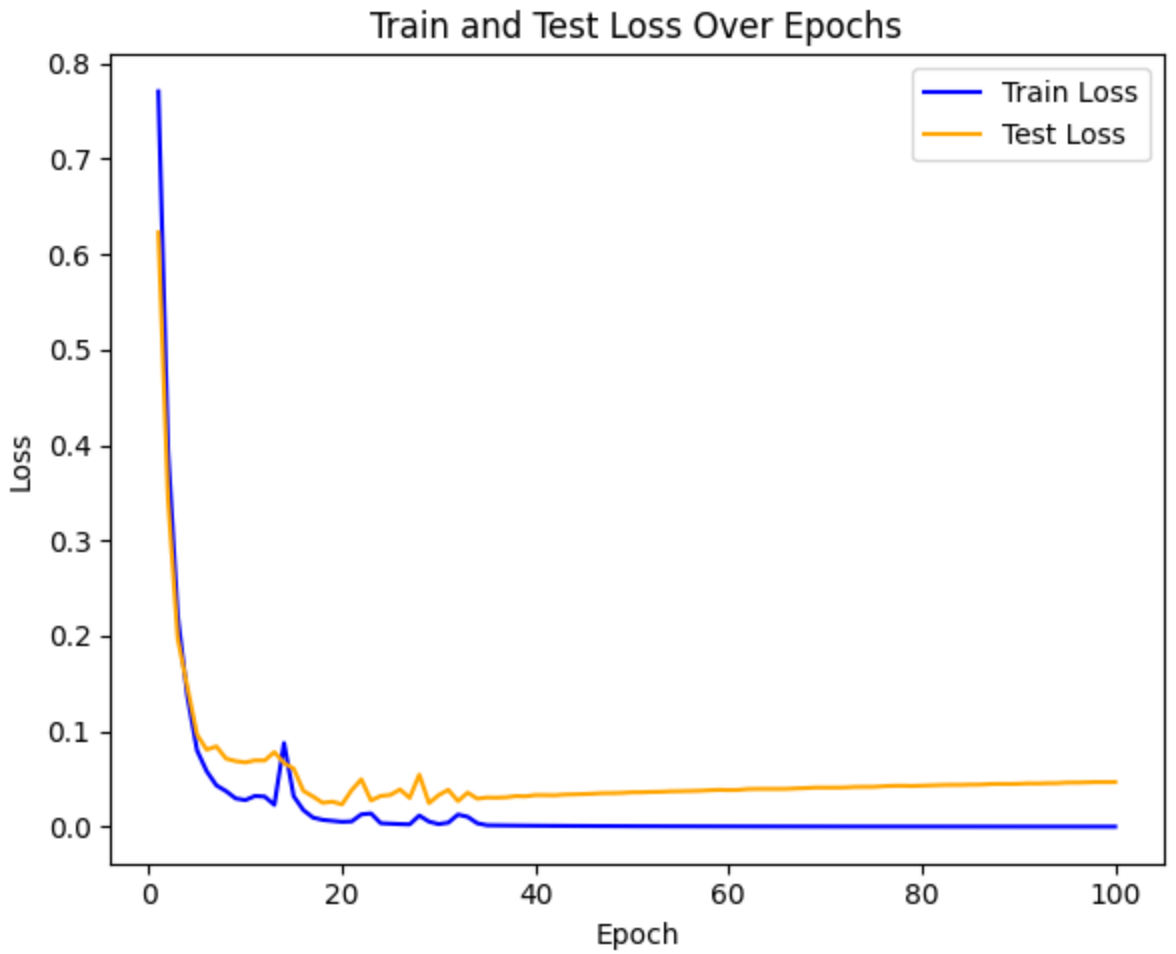
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	8
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.7



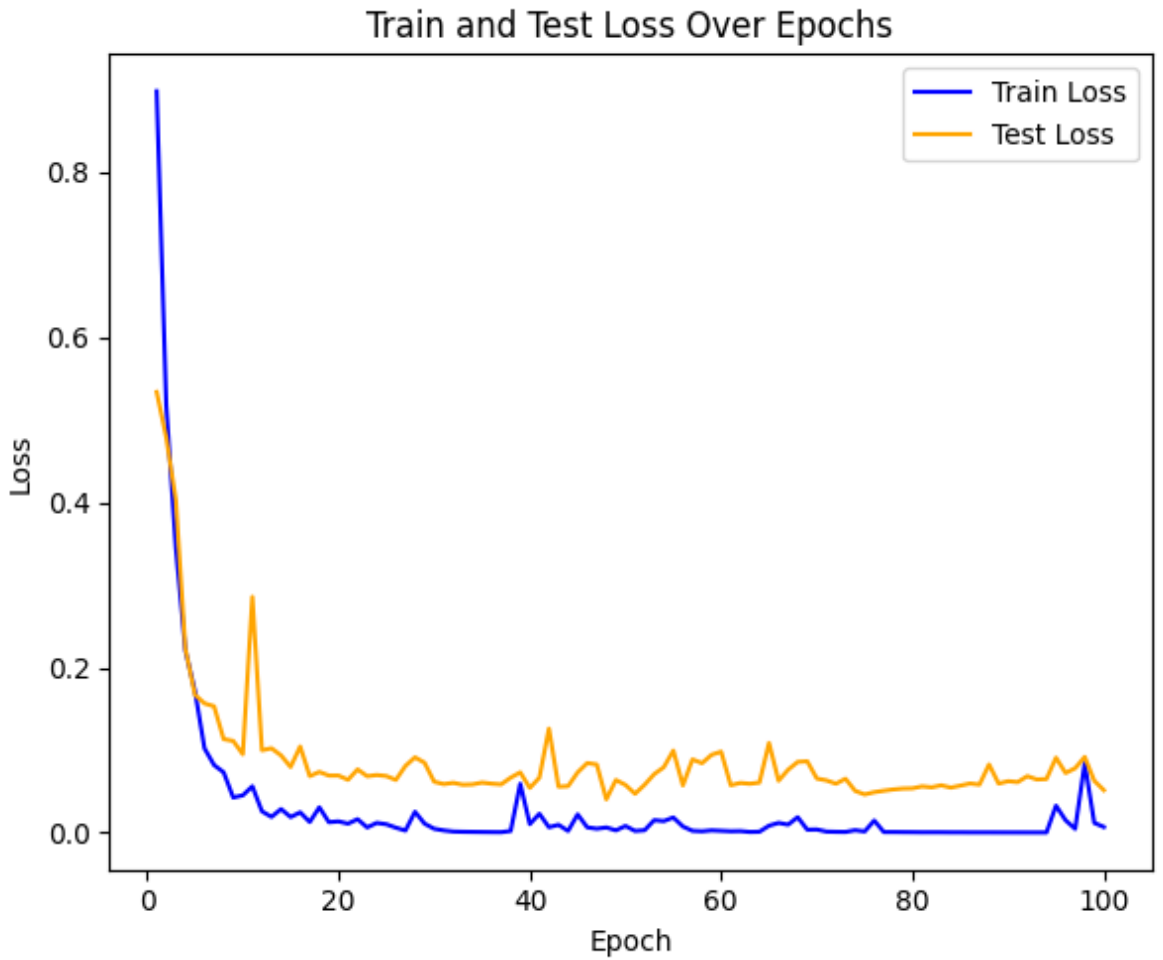
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	8
OUTPUT_SIZE	1
NUM_LAYERS	4
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.7



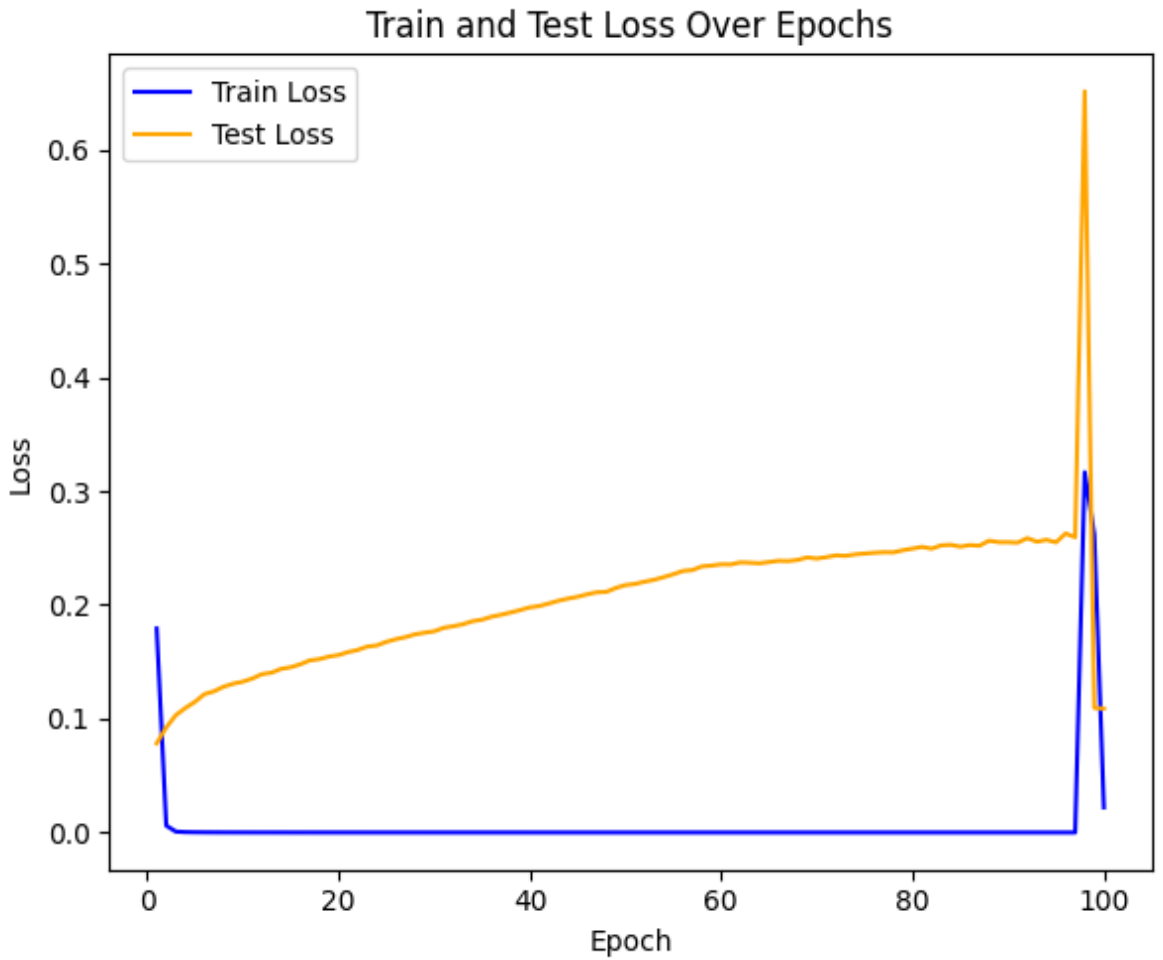
5. 학습 결과 - RNN

INPUT_SIZE	39
HIDDEN_SIZE	8
OUTPUT_SIZE	1
NUM_LAYERS	8
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.7



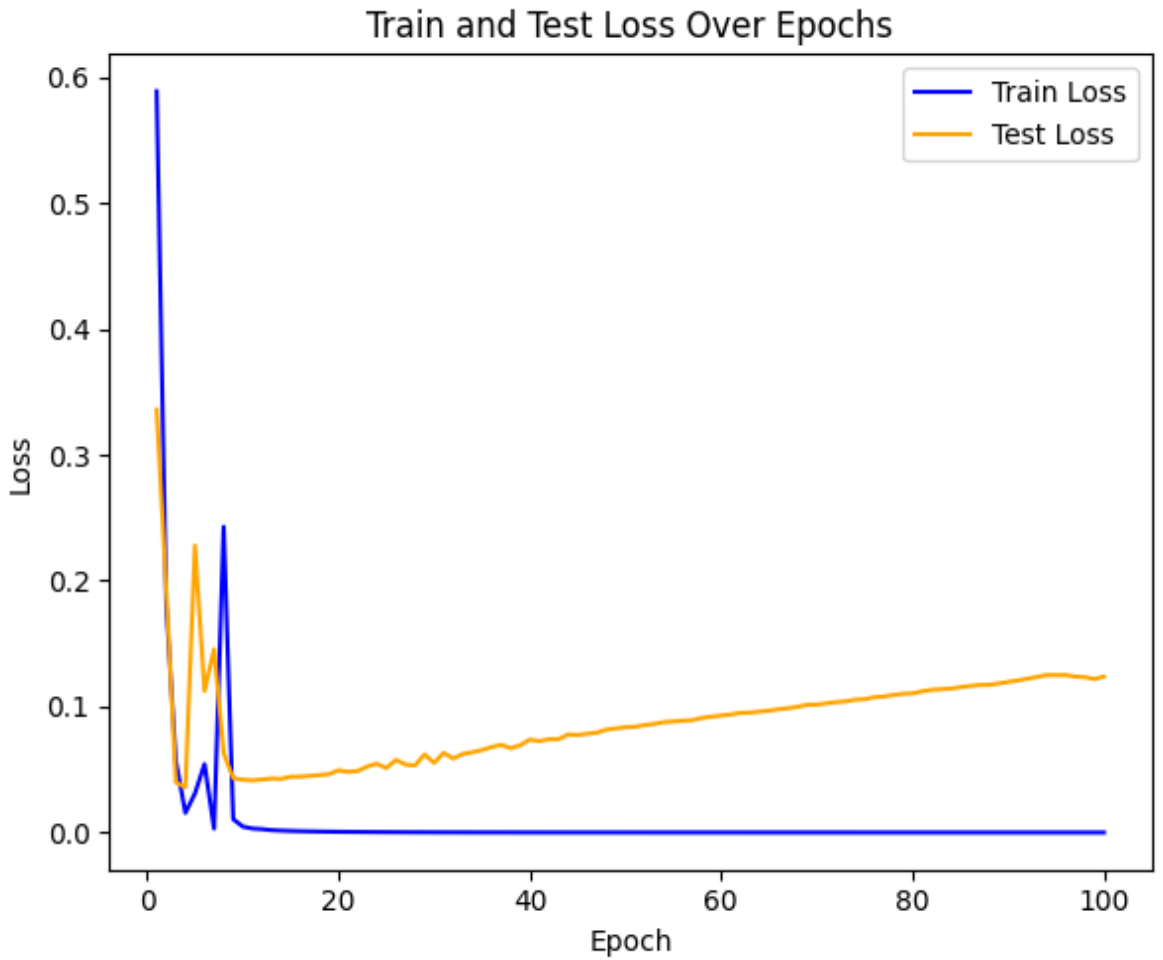
5. 학습 결과 - LSTM

INPUT_SIZE	39
HIDDEN_SIZE	64
OUTPUT_SIZE	1
NUM_LAYERS	2
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



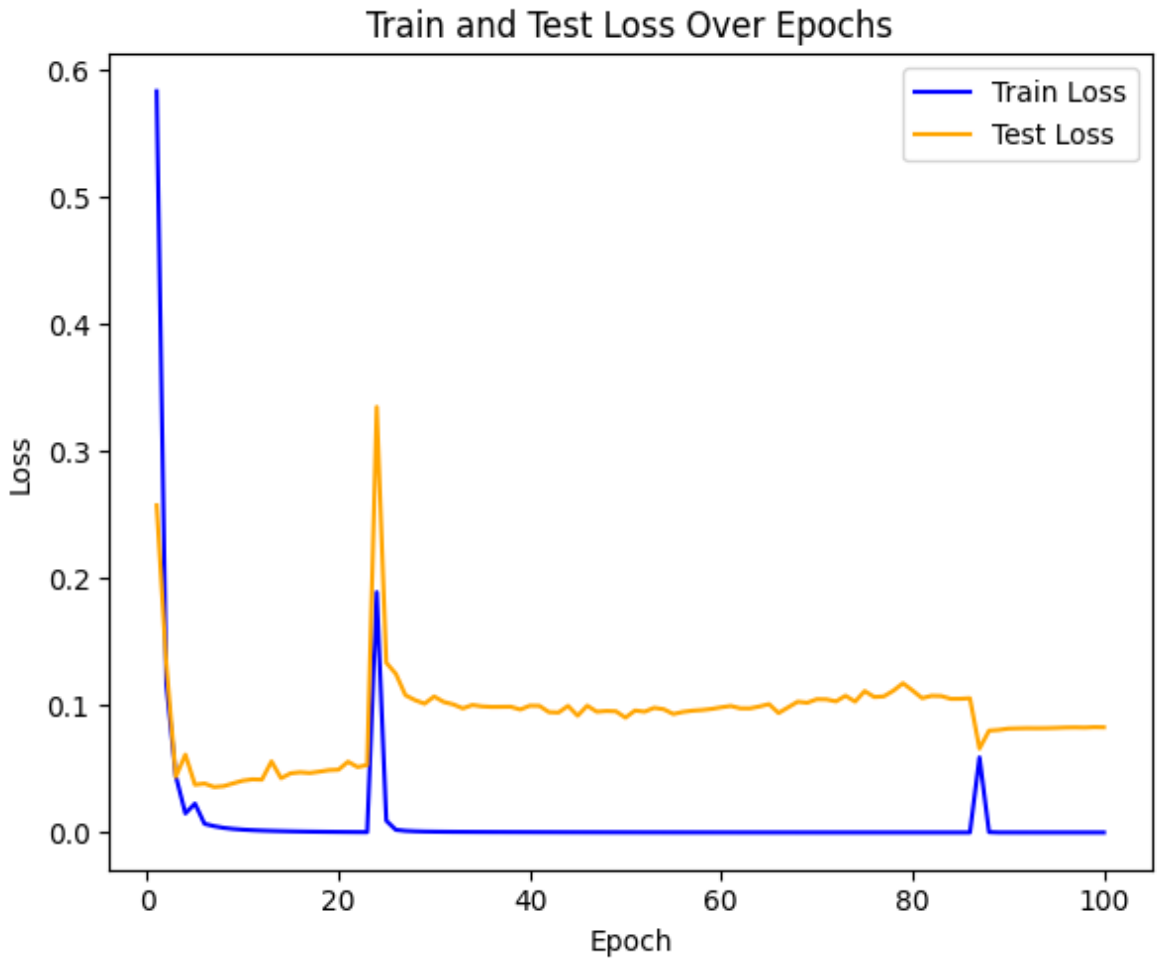
5. 학습 결과 - LSTM

INPUT_SIZE	39
HIDDEN_SIZE	8
OUTPUT_SIZE	1
NUM_LAYERS	4
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.5



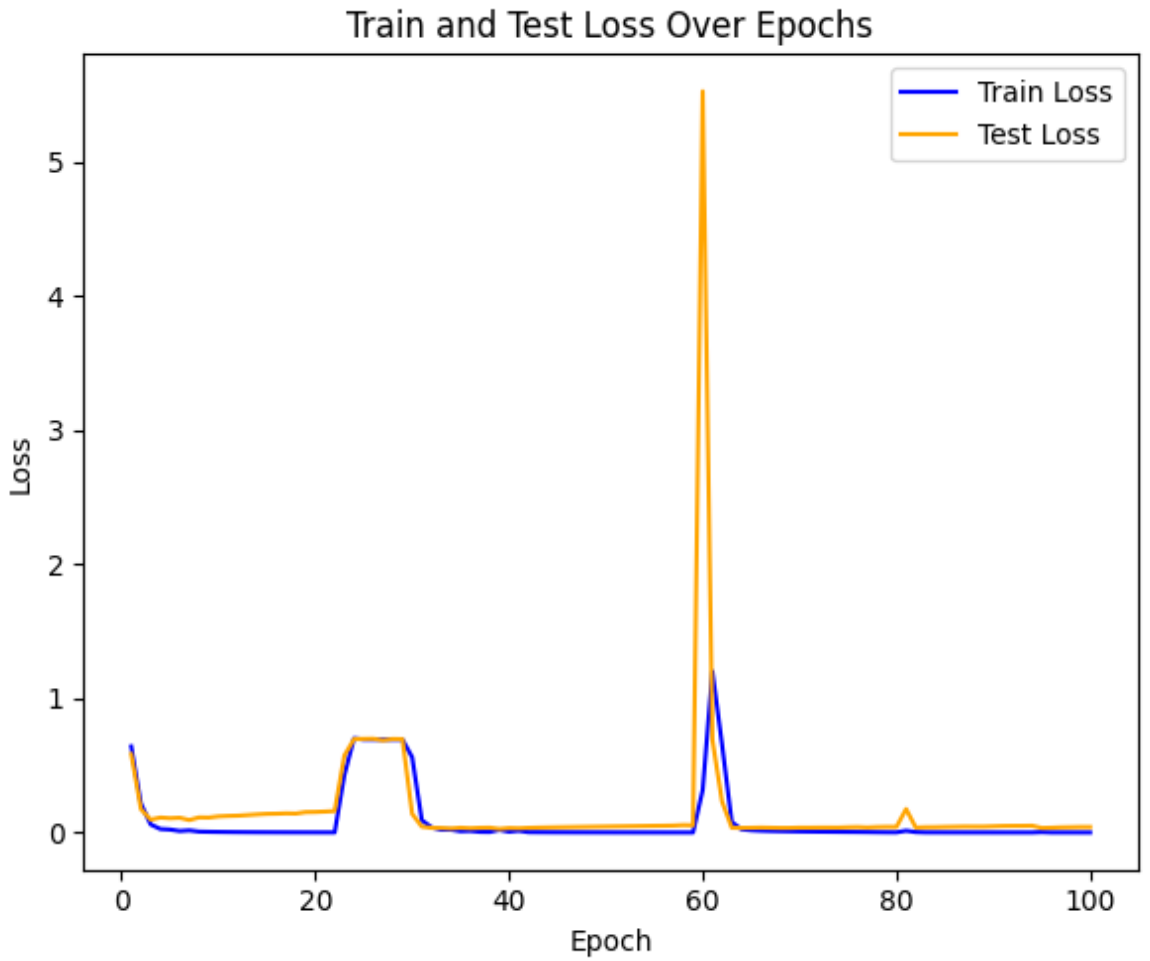
5. 학습 결과 - LSTM

INPUT_SIZE	39
HIDDEN_SIZE	4
OUTPUT_SIZE	1
NUM_LAYERS	4
BATCH_SIZE	50
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	15
DROPOUT_P	0.5



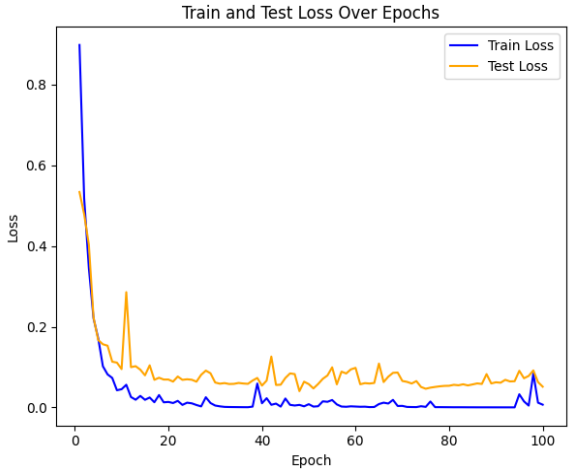
5. 학습 결과 - LSTM

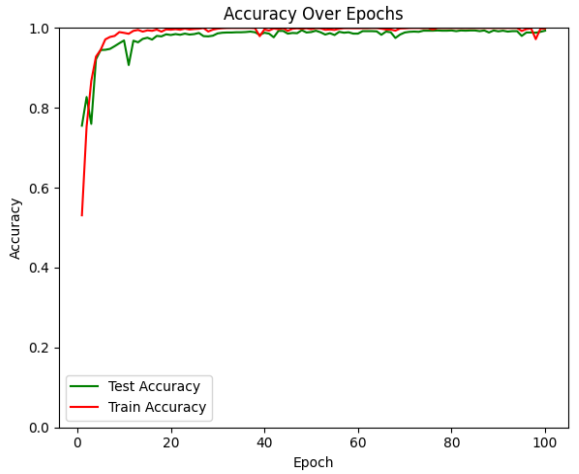
INPUT_SIZE	39
HIDDEN_SIZE	4
OUTPUT_SIZE	1
NUM_LAYERS	4
BATCH_SIZE	25
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	15
DROPOUT_P	0.5

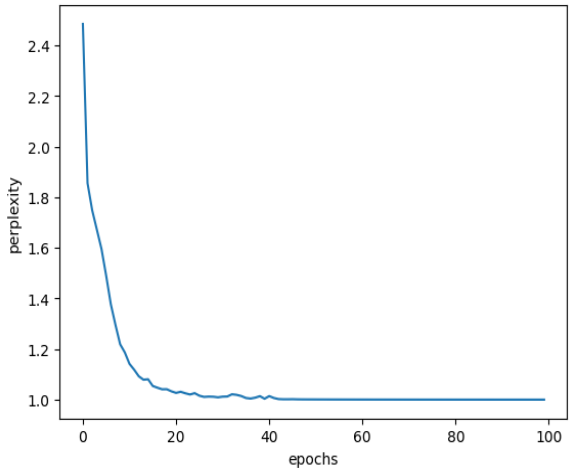


6. 정리

INPUT_SIZE	39
HIDDEN_SIZE	8
OUTPUT_SIZE	1
NUM_LAYERS	8
BATCH_SIZE	100
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	10
DROPOUT_P	0.7

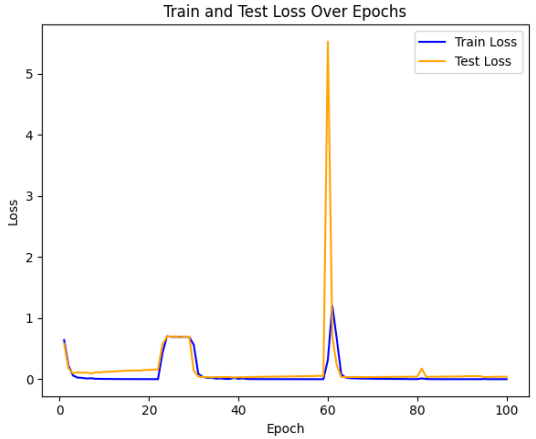


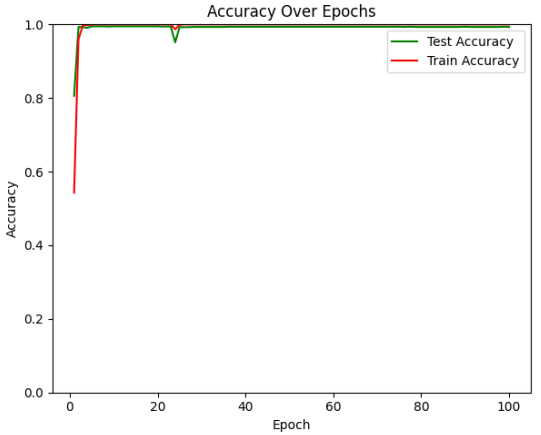


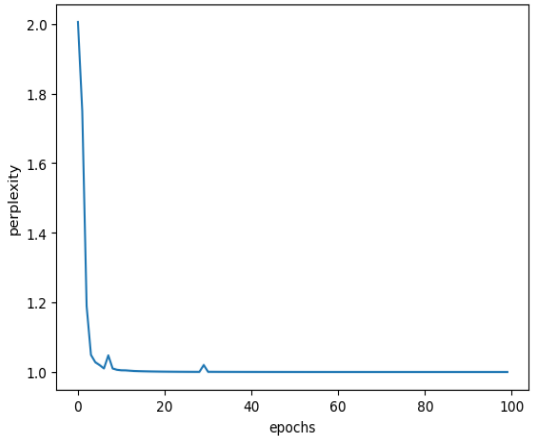


6. 정리

INPUT_SIZE	39
HIDDEN_SIZE	4
OUTPUT_SIZE	1
NUM_LAYERS	4
BATCH_SIZE	25
N_EPOCH	100
LEARNING_RATE	0.001
SEQUENCE_LENGTH	15
DROPOUT_P	0.5







THANK YOU!