


Advancement of Time Series Forecasting Model

2023712330 데이터사이언스융합학과 이루다

2024710673 데이터사이언스융합학과 신재인

2023711351 통계학과 김민경

2023712901 인공지능융합학과 이새하

- 
1. Problem definition
 2. Related research
 3. Proposed method
 4. Data and experiment
 5. Result analysis
 6. Appendix



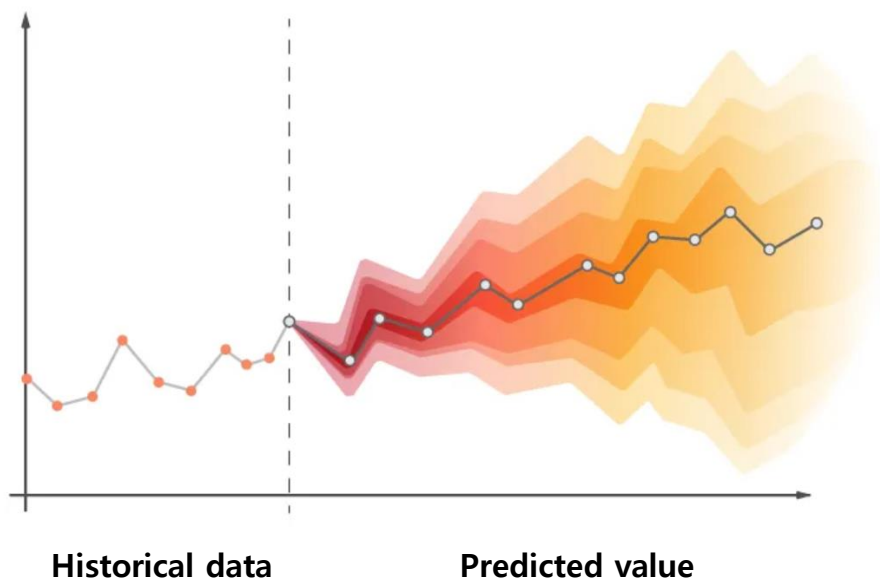
1

Problem

Definition

Time series forecasting

- Task: 과거 또는 현재 데이터를 활용하여 미래의 일정 기간, 또는 특정 시점 값을 예측
- Goal: 과거에 저장한 데이터를 통해 향후 동향을 이해하고, 이를 전략적으로 의사결정에 사용



$$\hat{y}_{t+1:t+\tau} = f(y_{t-k:t}, x_{t-k:t}, u_{t-k:t+\tau}, s, \tau)$$

- $\hat{y}_{t+1:t+\tau}$: Predicted target data
- $y_{t-k:t}, x_{t-k:t}$: Historical data
- $u_{t-k:t+\tau}$: Known future inputs (어느정도 알고 있는 사전 정보)
- s : Static meta data (시간에 따라 변동하지 않고 고정된 값)
- τ : Discrete forecast horizon

< Basic formula >

Time series forecasting

- Look-back window / Forecasting horizon

$$\hat{y}_{t+1:t+\tau} = f(y_{t-k:t}, x_{t-k:t}, u_{t-k:t+\tau}, s, \tau)$$

- 현재가 t시점이라고 했을 때,
 - Look-back window : 현재 시점(t)에서 과거로 몇 시점(-k)까지 데이터를 이용해 모델링 할 것인지 정의
 - Forecasting horizon : 모델을 통해 t+1 시점부터 어느 시점(=τ)까지 데이터를 예측할 것인지 정의

Goal of Time series forecasting

- ***What is the goal?***

- 타겟팅하는 길이의 시계열 데이터 예측의 정확도를 향상
- 데이터 내에서 추세 내지 패턴을 학습하고 노이즈, 계절적 변동, 패턴학습 및 오차를 효과적으로 처리

- ***Why should we solve this problem?***

- 전통 통계 모델(ARIMA 등): 장기적 상관관계를 포착하거나 비선형 패턴 처리에 한계를 가짐
- 딥러닝 모델(LSTM 등): 장기적 의존 학습에 gradient 소실 문제가 발생함
- Transformer 모델: 긴 입력 시퀀스 값을 학습할 때 비용 증가 및 메모리 한계로 성능이 저하됨



2

Related
Research

(1) Sequentially Trained Many-to-one LSTMs

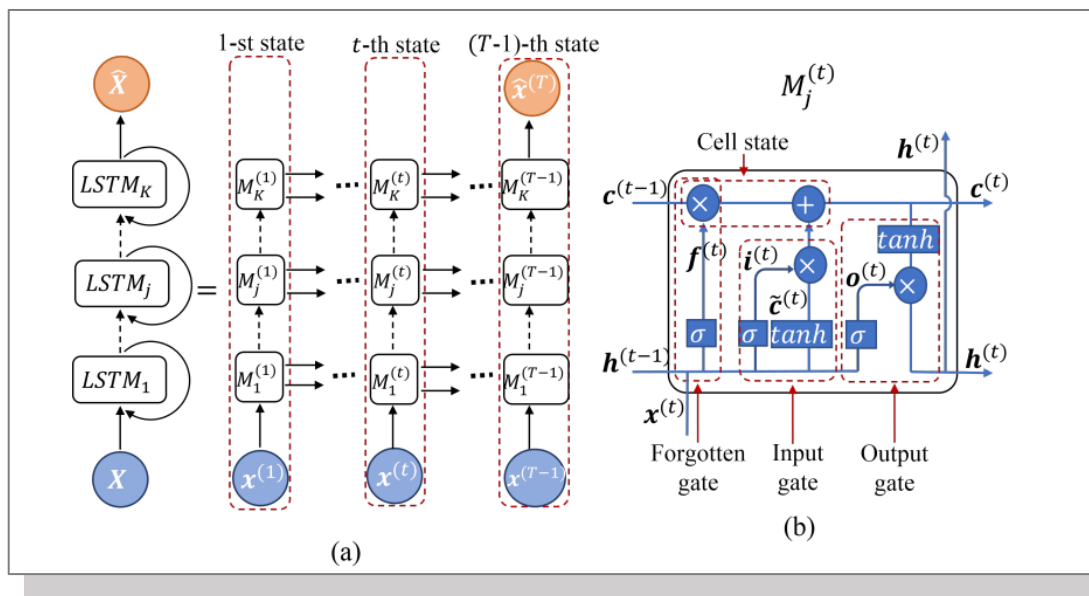
- 목적 : 금융 시장 데이터와 같은 비선형적 관계를 학습하여 실시간 미래 값을 예측하기 위해 LSTM(Long Short-Term Memory) 모델 도입하여 새로운 데이터가 모델에 통합되었을 때 지속적으로 학습 및 예측을 하고자 함
- 기존 모델과의 차이점
 - Sequential Training : 각 예측 후 새로운 데이터를 받아들여, 이를 기존 학습된 모델에 연속적으로 통합. 즉, 각 시간 단계에서 이전 예측 결과를 다음 학습의 데이터로 사용하여, 시간이 진행됨에 따라 모델이 지속적으로 업데이트
 - Many-to-one 구조 : 과거의 여러 데이터 포인트를 입력으로 사용하여 한 번에 하나의 미래 시점만을 예측
 - 실시간 업데이트 : 새로운 데이터가 모델에 지속적으로 입력되어 실시간 예측
- 모델의 의의 : 시간 단계의 데이터를 반복적으로 학습하여 더욱 정확한 예측을 가능하게 하며, 자주 패턴이 변하거나 예측하기 어려운 경향이 있는 주식 데이터와 같이 비정형적 데이터를 처리하는 데 유용

Reference : Real-time Forecasting of time series in Financial Markets Using Sequentially Trained Many-to-one LSTMs

(1) Sequentially Trained Many-to-one LSTMs

- K-stacked LSTM(Long Short-term Memory) 모델 :

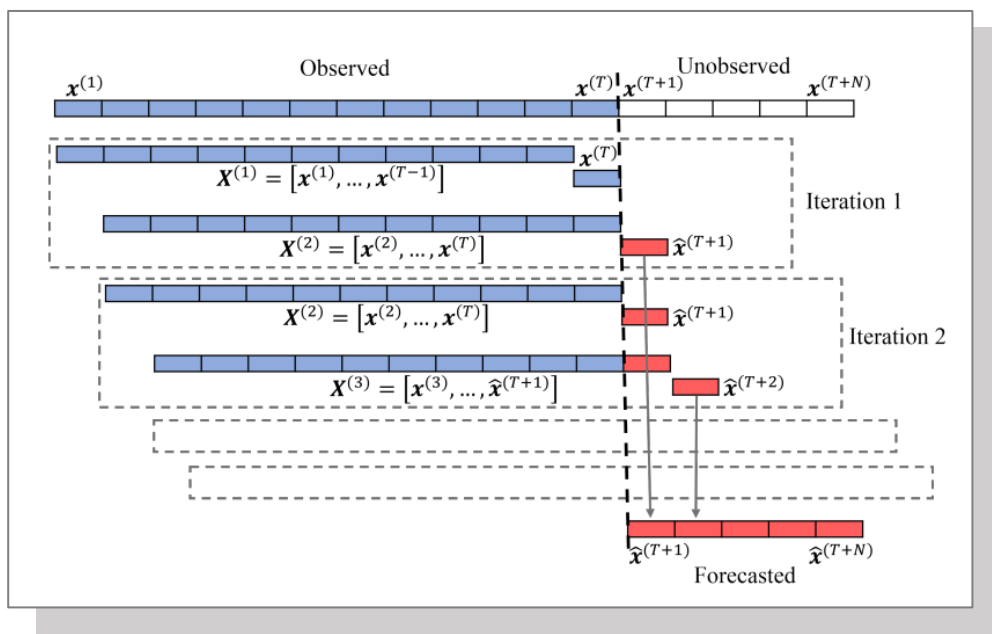
여러 층의 LSTM이 쌓여 있는 구조로 복잡한 데이터의 패턴을 학습하는 데 도움이 되며, 각 층은 입력 데이터의 다른 측면을 처리하고, 모든 층을 통과한 데이터는 시계열의 다음 시점을 예측



- ① **망각게이트**: 이전 은닉상태($h^{(t-1)}$)와 현재 입력($x^{(t)}$)을 결합하여 잊을 정보 결정
- ② **입력게이트**: 현재 입력에서 새롭게 저장할 정보를 결정
- ③ **새로운 후보 셀 상태**: 새로운 장기 정보를 계산
- ④ **셀 상태 업데이트**: 이전 셀 상태($c^{(t-1)}$)와 새로운 정보를 결합하여 장기 메모리를 업데이트
- ⑤ **출력게이트**: 시그모이드 레이어를 통해 출력할 정보를 결정
- ⑥ **은닉상태 업데이트**: 셀 상태($c^{(t)}$)에서 정보를 필터링하여 다음 은닉 상태로 전달

(1) Sequentially Trained Many-to-one LSTMs

- Sequentially Trained Many-to-one LSTMs 모델:
고정된 길이의 입력 시퀀스 데이터를 기반으로 한 번의 iteration에서 단일 미래 시점을 예측하도록 설정되고,
필요한 예측 길이를 충족할 때까지 반복하는 방식



첫 번째 반복 iteration:

- ① 입력 데이터 $X^{(1)} = [x^{(1)}, \dots, x^{(T-1)}]$ 로 모델 F 를 학습하여 $x^{(T)} = F(X^{(1)})$ 예측
- ② 이후, $X^{(2)} = [x^{(2)}, \dots, x^{(T)}]$ 를 입력하여 ($x^{(T)}$ 사용) $(T+1)$ 시점을 예측

두 번째 반복 iteration:

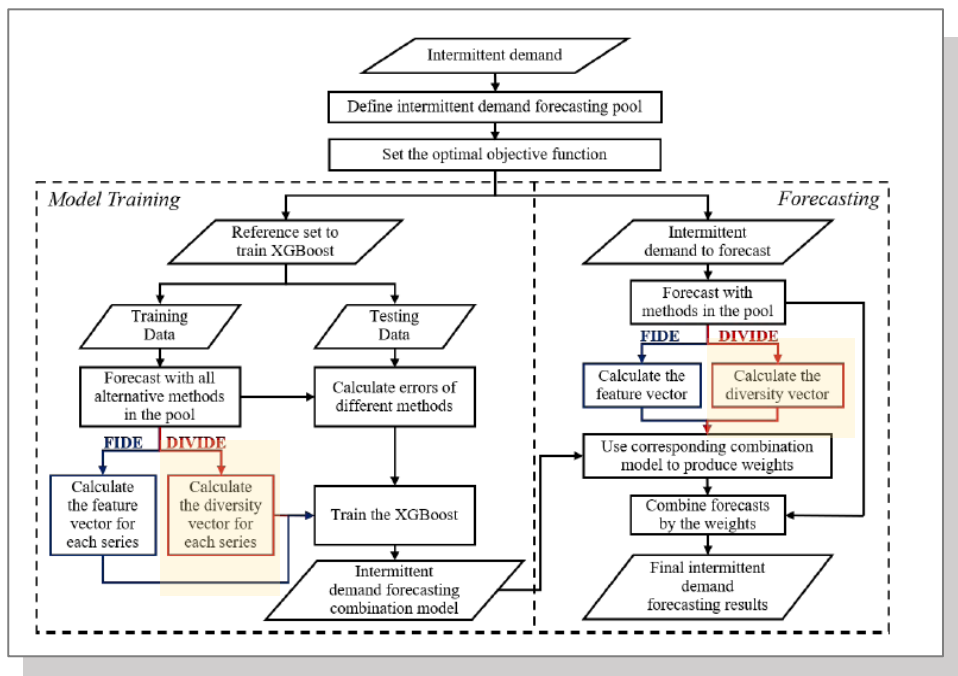
- ① 동일한 모델 F 를 $X^{(2)} = [x^{(2)}, \dots, x^{(T)}]$ $\hat{x}^{(T+1)} = F(X^{(2)})$ 를 계산
- ② 다음으로, $X^{(3)} = [x^{(3)}, \dots, \hat{x}^{(T+1)}]$ 를 입력하여 $(T+2)$ 시점 예측 ($\hat{x}^{(T+2)} = F(X^{(3)})$)
- ③ 이 과정을 모든 N 시점에 대한 예측이 완료될 때까지 계속 수행

(2) Intermittent Demand Forecasting

- 목적 : 시계열 데이터의 특징(수요 간격, 변동성 등)과 다양한 예측 모델의 조합을 활용해서 Intermittent Demand Forecasting(여러 기간동안 수요가 0이 되는 패턴인 간헐적 수요 예측) 정확도를 향상
- 기존 모델과의 차이점
 - 특징 기반 접근 : 전통 모델에서 간단한 평균 또는 중앙값을 사용하는 방법과는 대조적으로 시계열 데이터의 특징을 기반으로 조합한 가중치를 활용하여 데이터의 내재적 특성과 변동성을 더 잘 파악하고 이를 예측해 효과적으로 통합
 - 다양한 종류의 예측 모델 조합 : 간헐적 수요 예측에 특히 유용한 크로스컨 방법 및 최신 기계 학습 방법까지 포함하여 다양한 예측 방법의 장점을 결합하고 단점을 보완
- 모델의 의의 : 간헐적 수요는 수요 자체가 희소하거나 불규칙적으로 발생해 해당 모델을 표준 예측 모델로 활용하기는 어려우나, 비정기적 상황에서의 수요 패턴 정확하게 가늠해야 하는 재고 관리와 공급망 최적화에 적합

Reference : Feature-based intermittent demand forecast combinations: accuracy and inventory implications

(2) Intermittent Demand Forecasting



(Training)

- Intermittent demand 예측을 위한 pool 설정, 최적의 목표 함수 설정
- XGBoost 모델을 훈련하기 위한 Reference set 준비
- pool에 포함된 모든 대안적 예측 방법을 사용해 데이터 예측
- 각 시리즈에 대해 feature vector(**FIDE**)와 diversity vector(**DIVIDE**) 계산
- XGBoost 통해 분석, 각 시리즈에 대한 최적의 예측 조합 가중치 결정에 활용

(Forecasting)

- 실제 Intermittent demand 데이터에 대한 예측 준비
- 모델 pool 내의 방법을 사용해 Intermittent demand 예측,
- FIDE, DIVIDE 사용해 각 방법의 feature and diversity vector 계산
- XGBoost 모델로부터 생성된 가중치를 사용해 각 방법의 예측 결과 조합
- 조합된 예측을 통해 최종 Intermittent demand 예측 결과를 얻음

❖ **FIDE**: 시계열 특징 학습, **DIVIDE**: 모델의 다양성 학습

(2) Intermittent Demand Forecasting

- **DIVIDE (Diversity-based Intermittent Demand Forecasting) :**

- ❖ **다양성 계산**

- 각 예측 모델이 생성한 예측값의 차이를 기반으로 계산 ①
- 모든 예측 모델 pair간 다양성을 계산하여 벡터로 저장
(ex. N개의 모델이면 다양성벡터는 N(N-1)/2 차원)

- ❖ **가중치 학습**

- 모델 간 상호보완성을 반영해 조합 예측의 정확도를 최대화하기 위해 다양성 벡터를 기반으로 모델별 가중치를 학습
- 최적화 문제를 통해 학습되는 다양성 기반 조합 가중치 ②
RMSSE(root mean squared scaled error)를 사용한 예측오차 계산
- XGBoost 모델이 다양성 벡터와 최적화된 가중치의 관계를 학습하고, 결과를 토대로 새 데이터의 조합 가중치를 생성

①

$$DIV_{ij} = \frac{1}{H} \sum_{h=1}^H \frac{(\hat{y}_{i,h} - \hat{y}_{j,h})^2}{\left(\frac{1}{T} \sum_{t=1}^T |y_t|\right)^2}$$

H : 예측 구간의 길이

$\hat{y}_{i,h}$: 모델 i 의 h 단계 예측값

y_t : 관측값

T : 전체 train data 길이

②

$$\underset{w^D}{\operatorname{argmin}} \sum_{n=1}^N \sum_{i=1}^M w(D_n)_i \times error_{n,i}$$

$w(D_n)_i$: 다양성 벡터 D_n 에 따라 학습된 모델 i 의 가중치

$error_{n,i}$: n 번째 시계열 데이터와 i 번째 모델의 예측 오차

N : 전체 시계열 데이터의 수

M : 예측 모델의 수

(3) Linear Model (Dlinear, Nlinear)

- 목적 : Transformer 기반 방법론들이 LTSF task를 해결하는데 실제로 효과적인지에 대해 의문, simple 1-layer linear model 과 Transformer 기반 방법론들의 LTSF 성능을 비교하여 Transformer 기반 방법론의 효용성 확인
- 기존 Transformer 모델과의 차이점 :
 - 1-layer linear model: 단일 선형 계층으로 구성해 연산을 간소화한 모델
 - Informer: 효율적인 시계열 예측을 위한 Transformer 변형 모델
 - Autoformer: 시계열의 추세와 계절성을 자동으로 학습하는 Transformer 기반 모델
 - Pyraformer: 피라미드 구조를 활용한 시계열 예측 모델
 - FEDformer: 주파수 영역에서의 효율적인 디컴пози션을 활용한 Transformer 모델
- 모델의 의의 : Linear Model은 구조가 단순하여 구현이 매우 쉽고 복잡한 하이퍼파라미터 튜닝이 필요 없으며, 계산량이 Transformer 기반 모델보다 훨씬 적어 대규모 데이터셋에서도 빠르게 학습하고 예측할 수 있음

Reference : Are Transformer Effective for Time Series Forecasting?

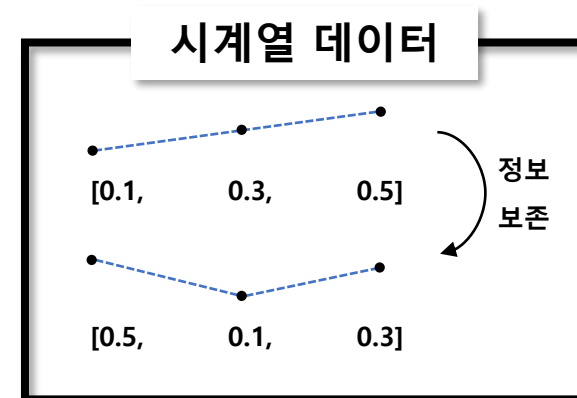
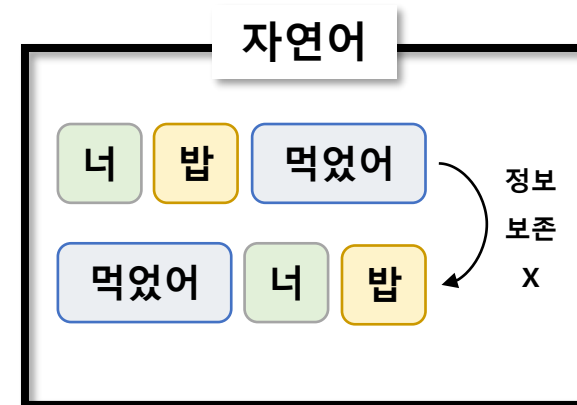
(3) Linear Model (Dlinear, Nlinear)

- Transformer model 구조적 문제

- 시계열 데이터는 semantic 정보가 부족한 수치형 데이터이기 때문에 순서가 변경되면 의미 정보가 보존되지 않음 → temporal change를 모델링하는 것이 매우 중요
- Transformer self-attention은 semantic correlation 추출하는 데 매우 효과적이거나, permutation-invariant 특성을 가짐 → temporal information loss가 발생할 수 있음

- Transformer based LTSF solution

- Transformer model에 positional/temporal embedding을 활용해 시간 정보 추가 시도
- Input에 shuffling 적용해 성능 비교한 결과, Transformer 기반 방법론은 성능 차이 미비 → temporal embedding이 시간 순서를 보존하지 못했다는 것을 의미

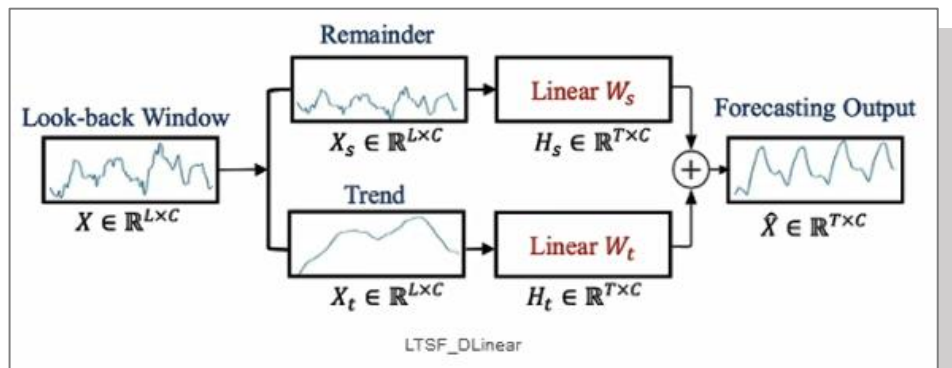


(3) Linear Model (Dlinear, Nlinear)

- LTSF-Linear 모델 (DLinear, NLinear):

M개의 변수로 이루어진 L길이의 데이터를 input으로 받아 linear layer를 통해 T길이의 미래 예측을 도출

- DLinear : Autoformer, FEDFormer에서 사용된 time series decomposition을 LTSF-Linear에 적용한 모델
- NLinear : 마지막 시점의 값을 빼는 normalization을 LTSF-Linear에 적용한 모델



(1) $\hat{X} = H_s + H_t$

(2) Remainder $H_s = W_s X_s$ (3) Trend $H_t = W_t X_t$

단일 선형 계층 구조 :

- 과거 시계열 데이터 입력받아 선형 연산을 통해 미래 시계열을 직접적으로 예측
- 복잡한 비선형 함수나 Attention 메커니즘을 사용하지 않음

독립적 변수 처리 :

- 다변량(multivariate) 시계열 데이터에서 각 변수는 독립적으로 처리됨
- 각 변수에 대해 선형 가중치가 학습되며, 변수 간 상호작용은 고려하지 않음
- 모델의 단순성 및 효율성을 높이지만, 변수 간의 상관관계를 학습하지 못함

(4) Patch Time Series Transformer

- 목적 : 시계열 데이터는 고차원이고 긴 경우가 많기 때문에 계산 효율성이 중요, 데이터를 패치(patch) 단위로 나누어 처리하여 데이터의 국소적 패턴을 잘 파악하면서도 장기적 종속성을 효과적으로 학습하도록 함
- 기존 모델과의 차이점
 - Patching: 기존 시계열 예측 연구들은 대부분 point-wise input token을 사용하였으나, patchTST는 time-series를 sub-series patch로 집계하고 이를 input으로 사용하여 지역적 특성 반영하도록 함. 이를 통해 메모리 사용량을 크게 줄임
 - Channel-independence: CNN이나 linear model 연구에서도 변수 간 correlation을 따로 모델링하지 않아도 효과적인 결과를 낸다는 점에 착안. Transformer model에서도 각 입력 토큰이 단일 채널 정보만 포함하도록 하여 연산량을 줄임
- 모델의 의의 : 패치 단위로 데이터를 나누어 더 많은 local semantic information을 반영하며, input token 길이 감소로 연산량과 메모리 사용량 감소시킴. 동일한 GPU와 학습 시간 내에서 더 긴 길이의 sequence 활용 가능

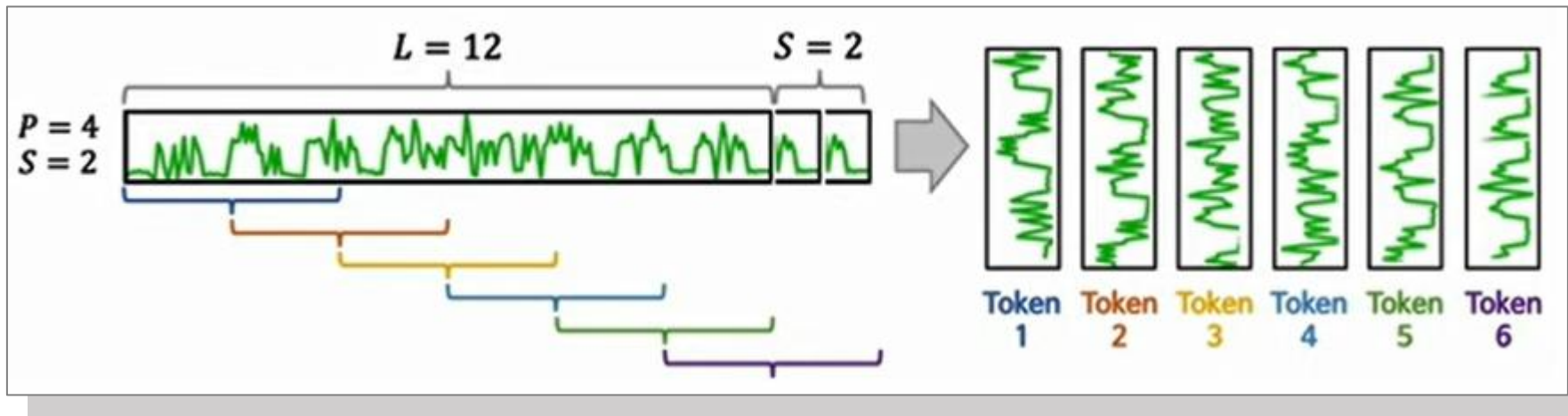
Reference : A Time Series is Worth 64 words: Long-term Forecasting with Transformers

(4) Patch Time Series Transformer

- Patching 개념 :

단변량 input $x^{(i)} \in \mathbb{R}^{1 \times L}$ 가 주어지면, $x^{(i)}$ 에 sliding window 방식을 적용하여 P길이의 sub-series patch N개로 구성된 patch

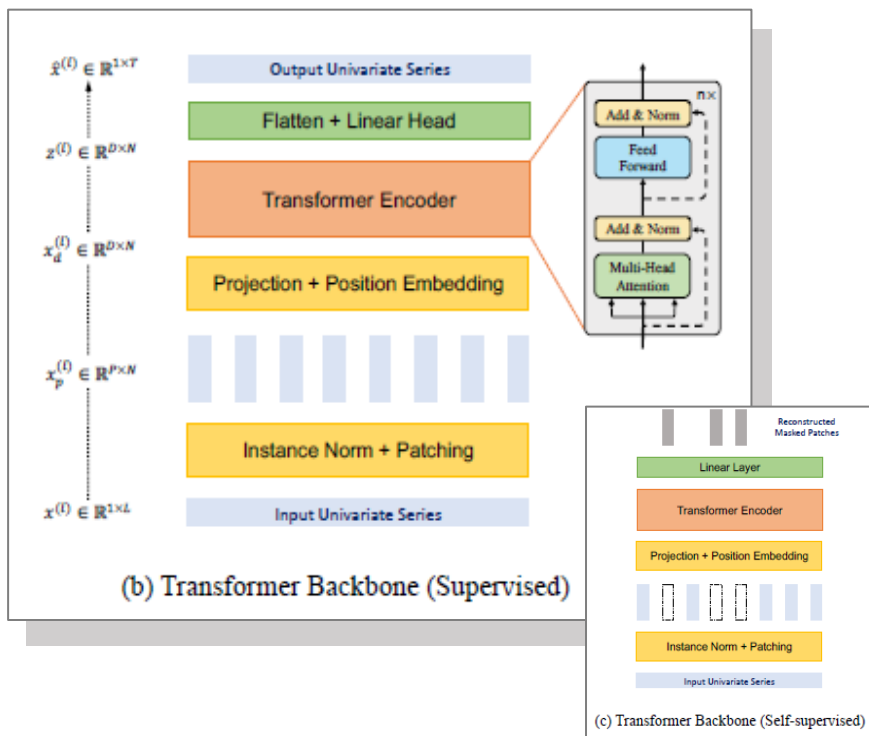
sequence $x_p^{(i)} \in \mathbb{R}^{P \times N}$ 을 생성하는 방법, $N = \left\lfloor \frac{(L-P)}{S} \right\rfloor + 2$ 개의 patch 생성되어 input token 길이 $1 \rightarrow L/S$ 개로 감소



- ❖ P: 패치의 길이, S: Sliding window step
- ❖ 최종적으로 $L(\text{input 데이터 길이}) + S$ 길이의 데이터

(4) Patch Time Series Transformer

- PatchTST 모델 :



- ① **Input Univariate Series:** 다변량 데이터를 단변량 시리즈로 분할
- ② **Instance Normalization+Patching:** 변수마다 정규화, N개의 패치로 나눔*
Patching: sliding window 방식을 적용하여 P길이의 sub-series patch N개 생성
- ③ **Projection+Positional Embedding:** trainable linear projection과 learnable position encoding을 더하여 시간 순서를 반영한 patch embedding 생성
- ④ **Vanilla Transformer Encoder:** self-attention 통해 patch representation 도출
- ⑤ **Flatten+Linear Head:** representation을 flatten해서 prediction output 도출
- ⑥ **Loss Function:** 각 변수별 MSE loss 산출 후 평균을 loss로 활용하여 모델 학습
- ⑦ **Representation Learning:** masking imputation* 활용해 self-supervised 모델 적용
Patch를 Masking한 후 representation을 활용하여 masked patch 복원하는 방법



3

Proposed
Method

1398

Baseline model 한계점

- **Sequentially Trained Many-to-one LSTMs** : 변동성 큰 자산 데이터 예측 시 여전히 오차가 크며, Many to one 방식을 취하여 한 시점 예측에 초점을 맞추고 있고 여러 기간을 예측하려면 계산 비용이 많이 필요
- **Intermittent Demand Forecasting (DIVIDE)** : 모델 간 다양성에 초점을 맞추므로 추세나 계절성을 반영하지 않아 데이터 결과가 왜곡될 가능성이 있으며, 작은 데이터는 다양성이 적으므로 방법론이 타당하지 않음
- **Linear Model (DLinear)** : LTSF-Linear는 단순한 선형 모델로, 복잡한 패턴이나 비선형성을 충분히 학습하지 못할 수 있으며, Transformer model의 장점인 병렬 처리 능력이나 복잡한 상관관계 학습 능력을 활용하지 못함
- **PatchTST** : 변수간 상관관계에 따라 학습 성능이 달라질 수 있으며, 도메인 영향을 받을 수 있으며, anomaly detection과 같은 과업에서는 모델 성능이 떨어져 예측이 어려울 수 있음

Proposed method

- ***What results do you want to obtain and how do you apply machine learning?***

- ✓ PatchTST 모델 예측성능 고도화

- (1) 장기의존성(과거의 데이터가 현재에 미치는 영향 반영) 기반 패턴을 학습할 수 있는 LSTM 모델과

- (2) Patch기반으로 지역적 정보와 범역적 정보를 모두 학습할 수 있는 PatchTST 모델을 앙상블(Ensemble)하여 예측 성능 고도화

- ✓ (extra) PatchTST 모델의 경량화 *appendix에서 다룸.

- 예측력이 높으나 계산용량이 큰 PatchTST 모델을 경량화하여 모델용량을 줄이면서 예측성능 유지 목표



4

Data and Experiment

Dataset

- **ETT(Electricity Transformer Temperature)**

- 2021년에 발표된 논문 「*Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*」에서 처음 공개된 다변량 데이터셋
- ETTh는 한시간 단위, ETTm은 15분 간격으로 측정된 데이터
- 전력 변압기의 부하 상태와 온도로 구성됨. (주로 OT가 예측target)
 - OT : 변압기의 온도(Temperature), 주요 예측 대상
 - HUFL : 부하 고주파 부분 부하
 - HULL: 부하 저주파 부분 부하
 - MUFL: 주 중간 고주파 부하
 - MULL: 주 중간 저주파 부하
 - LUFL: 부하 저부 고주파
 - LULL: 부하 저부 저주파

Experiment

- **PatchTST** 모델 성능 고도화 방법론: **Real Time** 예측에 최적화된 **LSTM** 모델과 **PatchTST** 모델의 앙상블

1 step) PatchTST 모델을 활용하여 LSTM 모델에 입력될 시퀀스를 처리

2 step) Feature Projection Layer를 통해 PatchTST 모델의 출력값을

LSTM 모델의 입력형식으로 변환. 즉, Linear 레이어를 사용하여 차원 조정

3 step) 변환된 출력을 LSTM에 입력하여 최종 예측 값 생성

PatchTST 출력 값 = (batch_size, num_patches, patch_size)

LSTM 입력 값 = (batch_size, sequence_length, input_size)

시간축으로 정렬된 데이터로 병합 후

*(*lstm_input = patchtst_output.reshape(batch_size, -1, patch_size)*

3차원 텐서로 차원 조정

Experiment

- 기본조건

- 입력 크기 (input size) = 48 또는 96
- 은닉 상태 크기 (hidden size) = 64
- 레이어 수 (num layers) = 2
- 드롭아웃 비율 (dropout1) = 0.2
- 에포크 수 = 10

*patchTST 모델도 동일한 조건으로 세팅하여 비교

```
# 14. CombinedModel 학습
lstm_model = LSTM3(input_size=48, hidden_size=64, num_layers=2, output_size=48, dropout1=0.2).to(device)
combined_model = CombinedModel(
    patchtst_model, lstm_model, patchtst_output_size=48, lstm_input_size=48
).to(device)
combined_train_loss, combined_val_loss = train_and_validate(
    combined_model, train_loader, val_loader, n_epochs=10, lr=0.001, model_name="CombinedModel"
)
```

Experiment

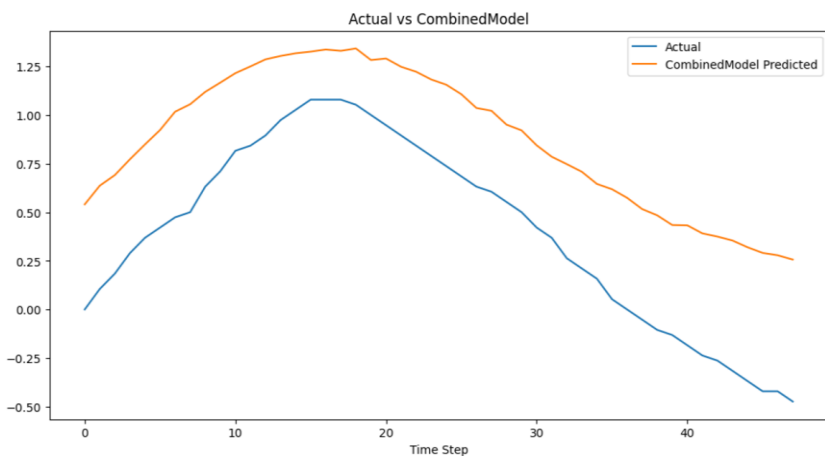
- 모델의 평가 : 예측 값과 실제 값 간의 평균 제곱 오차 (**MSE**) 손실을 계산하여 모델의 성능을 측정

[기대효과] 패치를 중심으로 전체 시계열 데이터의 구간별 특성(local feature)을 학습하고

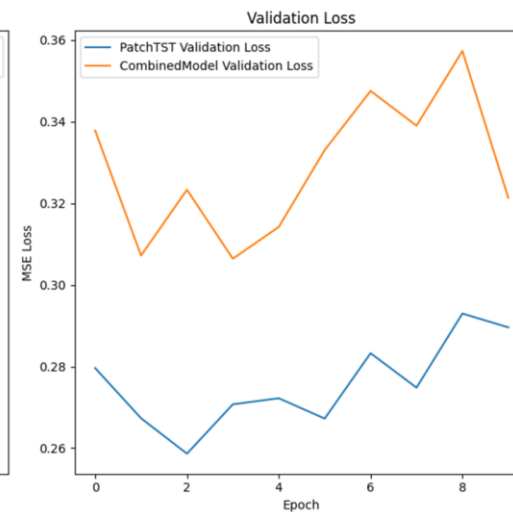
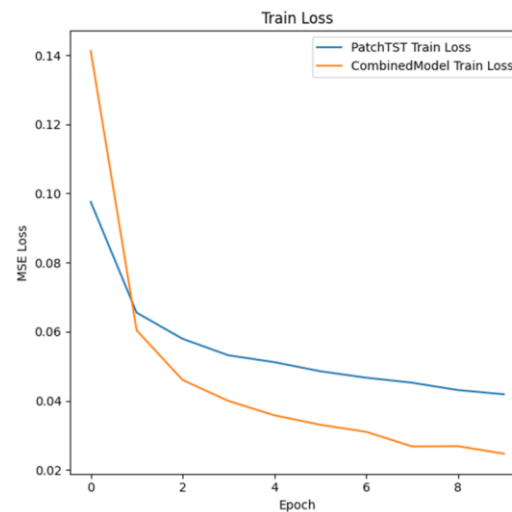
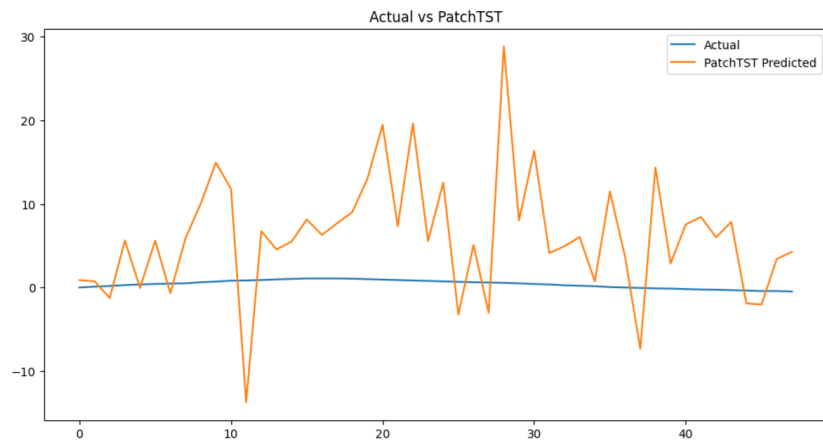
LSTM 모델로 장기 의존성을 추가 학습하여 데이터 예측력 및 학습안정성이 보다 향상될 것을 기대

Experiment

- 실험결과 - 데이터셋이 ETTm, output 구간 48인 경우, 결합모델이 PatchTST 보다 우수한 예측력을 보임

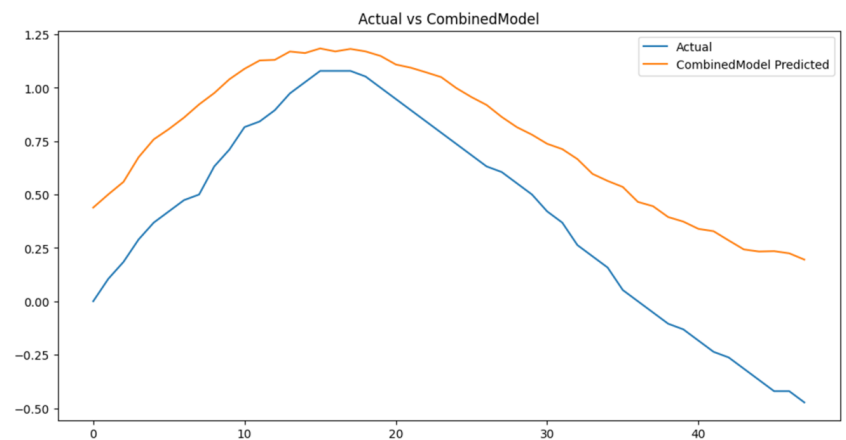


PatchTST Test Loss: 58.5627
CombinedModel Test Loss: 0.2814

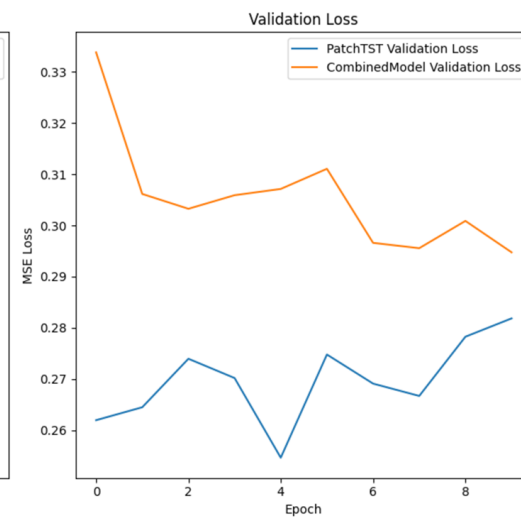
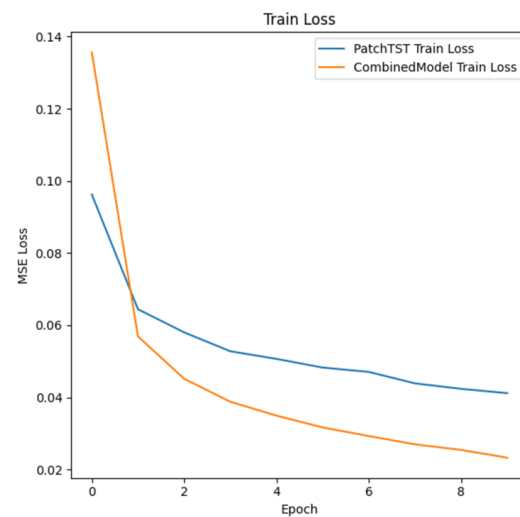
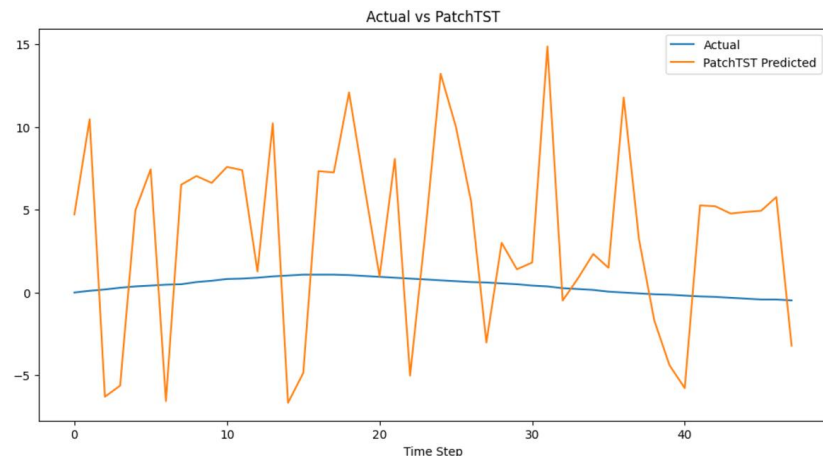


Experiment

- 실험결과 - 데이터셋이 ETTh인 경우에도, 과적합 우려가 다소 있으나 결합모델이 PatchTST 보다 우수한 예측력 보임.

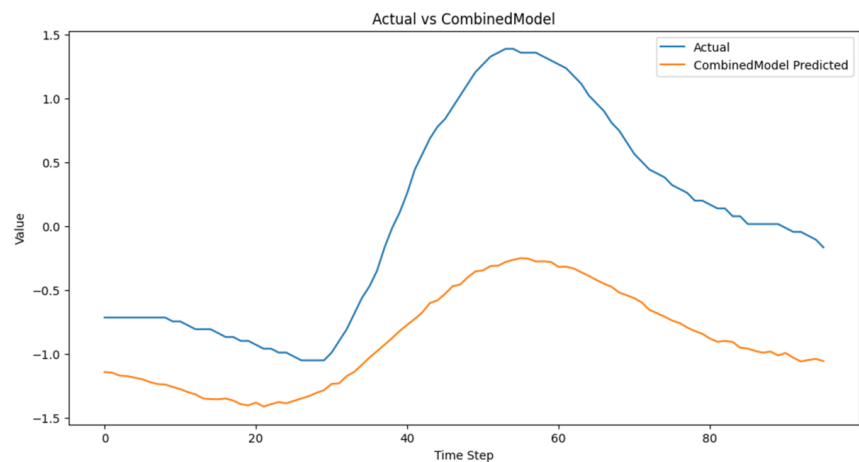


PatchTST Test Loss: 74.5963
CombinedModel Test Loss: 0.2886



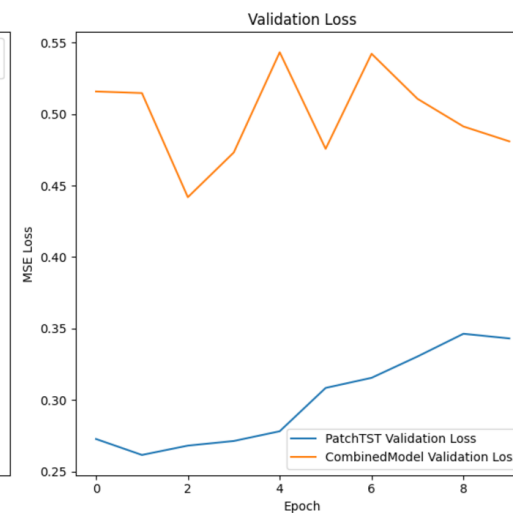
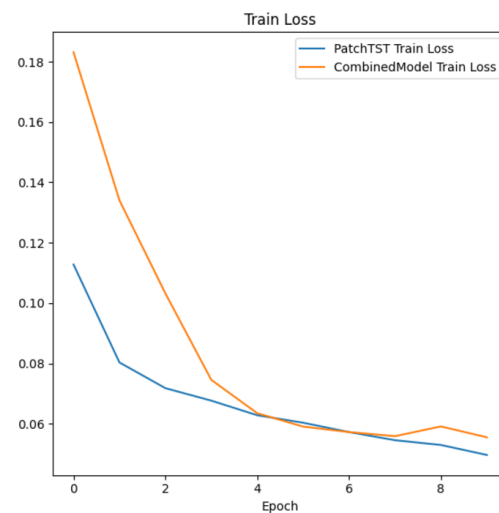
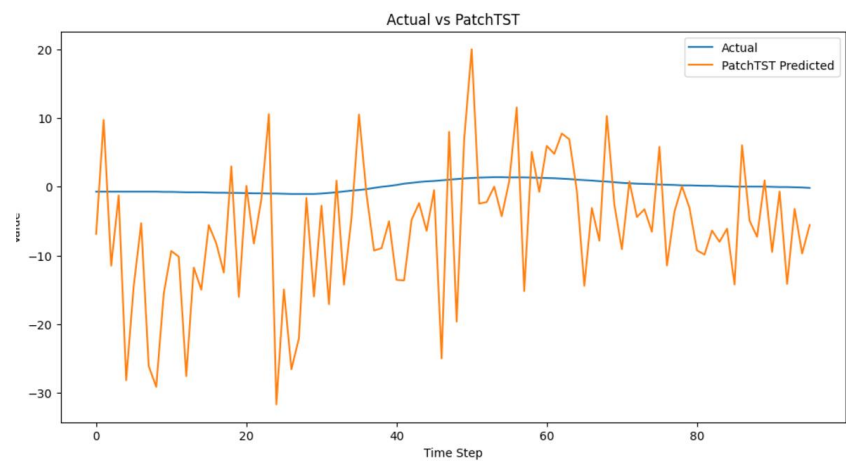
Experiment

- 실험결과 - 데이터셋이 ETTm이고, output 구간 96인 경우, 결합모델이 PatchTST 보다 우수한 예측력 보임.



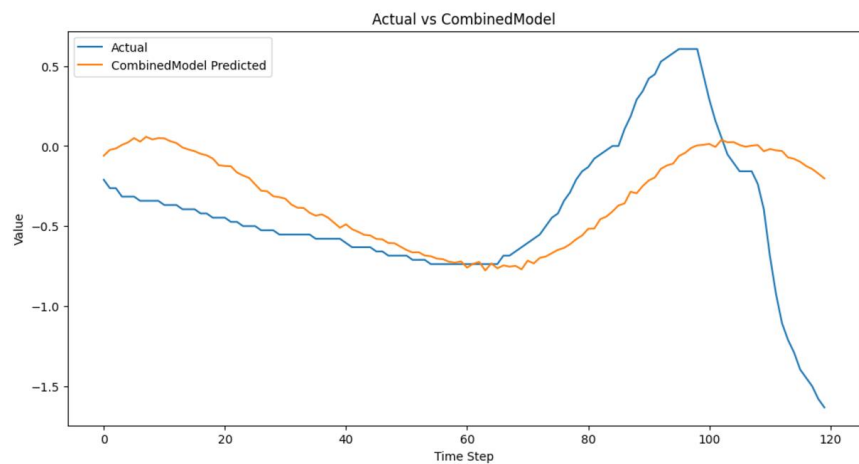
PatchTST Test Loss: 247.4025

CombinedModel Test Loss: 0.4186

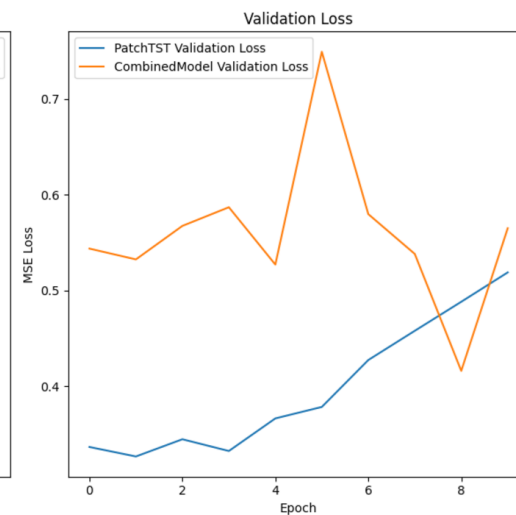
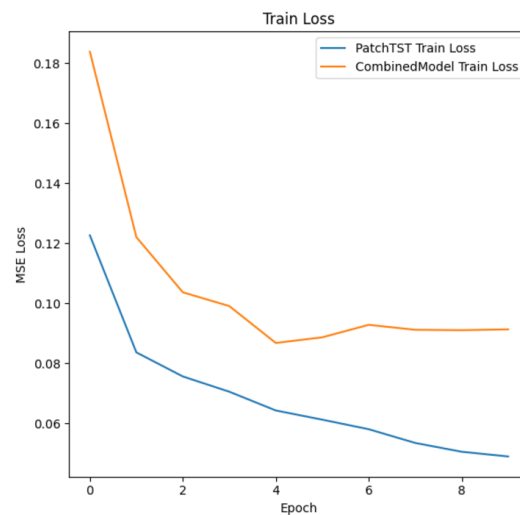
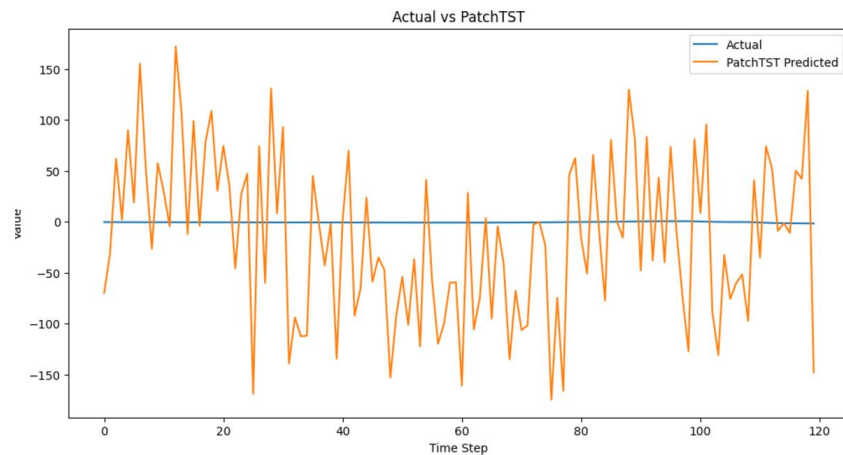


Experiment

- 실험결과 - 데이터셋이 ETTm, output 구간 120인 경우, 결합모델이 PatchTST 보다 우수한 예측력 보임.



PatchTST Test Loss: 938.8969
CombinedModel Test Loss: 0.5151





5

Result
Analysis

Result and analysis

- 다양한 예측길이에서 결합모델의 예측성능이 일관적으로 우수한 결과 보임. (단기데이터, 장기데이터 모두에서 WIN)
- 단, 96이상의 길이에서는 훈련손실과 검증손실의 움직임이 달라 과적합 우려가 있어 보이나, 예측길이의 변동에도 결합모델이 기존 patchTST 모델보다 전체적인 추세와 방향성을 정확히 예측하는 것이 확인됨. 즉, 모델의 성능과 무관한 것으로 판단
- 결론적으로, LSTM과 patchTST 모델을 앙상블하여 예측성능 고도화 목표 달성
- 추가로, 다양한 길이의 구간 예측도 가능한 것을 고려할 때 파라미터 조정이 뒷받침되면 다른 데이터셋으로 확장 가능할 것으로 판단



6

Appendix

Extra experiment

- 모델 성능을 고도화하는 하나의 방법으로 경량화도 함께 고려
- 예측성능이 우수한 patchTST 모델의 구조가
채널(데이터셋의 feature)별로 독립적으로 transformer encoding을 수행하여 용량이 상대적으로 큰 점에 착안,
경량화를 추가 목표로 삼고 이를 수행하였음

Extra experiment

- **PatchTST** 모델 경량화 방법론: 정규화 과정 생략, **Flatten** 후 **Transformer** 입력

1 step) Patch 생성 후 정규화 생략

2 step) Patch Flatten(평탄화) *Patch 수 x (Patch 길이 x features 수) 형태

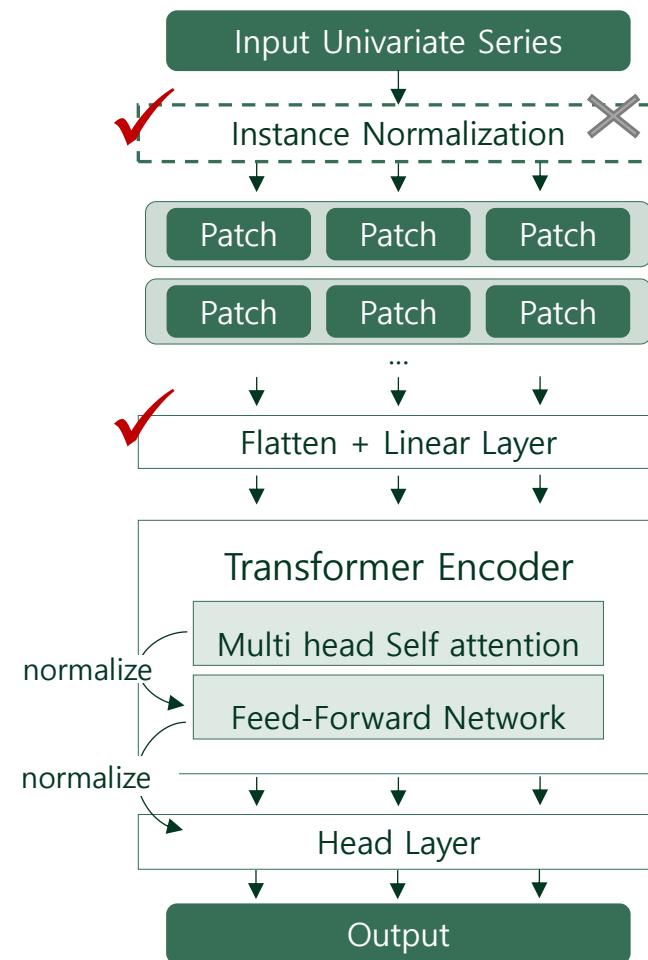
3 step) Linear Layer로 각 패치특성 압축하여 Transformer Encoder에 적합한 형태로 차원 조정

4 step) Pytorch 라이브러리에 내장된 nn.TransformerEncoder로 학습

5 step) Head Layer로 예측 길이로 변환해 출력

> transformer 입력 시퀀스를 생성하는 단계에서 계산복잡도를 줄임

> PatchTST의 출력을 모방하는 **[지식증류](knowledge distillation)** 방식 차용



Extra experiment

- 기본조건

데이터셋 특성 개수 (c_in) = 7

입력 크기 (context_window) = 366

예측 길이 (target_window) = 96

패치 길이 (patch_len) = 16

패치 이동간격 (stride) = 8

**patchTST 모델도 동일한 조건으로 세팅하여 비교*

- 경량화조건

transformer encoder layer 수 (n_layers) = 2 (*patchTST는 일반적으로 3 layers 이상*)

transformer 모델의 입출력 벡터차원(d_model) = 16 (*patchTST는 일반적으로 128 ~ 512*)

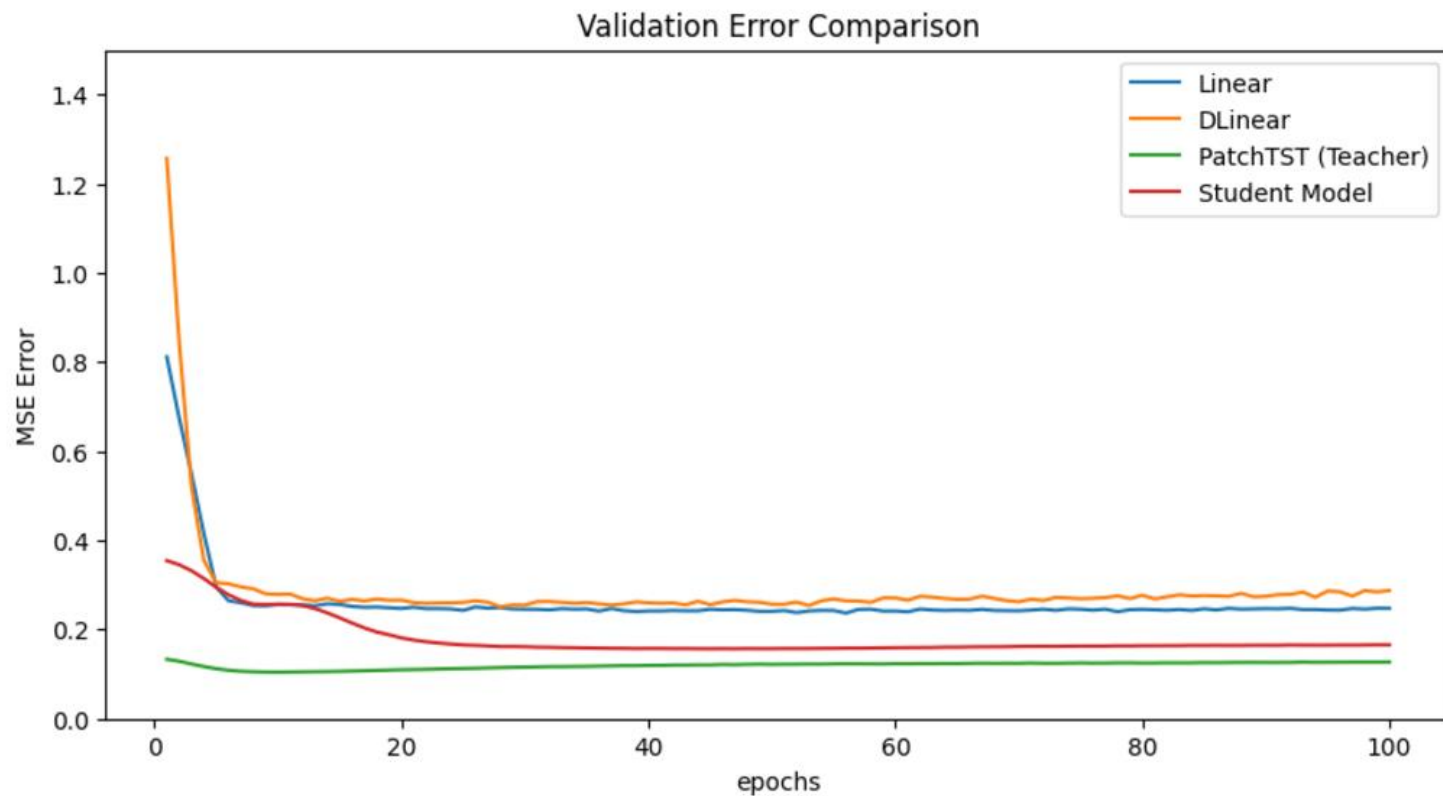
multi-head attention의 헤드 수 (n_heads) = 2 (*patchTST는 일반적으로 8 ~ 16*)

```
c_in = 7
context_window = 366
target_window = 96
patch_len = 16
stride = 8
```

```
student_model = StudentModel(
    num_features=num_features, # 입력 feature 수
    seq_len=seq_len,           # 입력 시계열 길이
    pred_len=pred_len,         # 예측 시계열 길이
    hidden_dim=hidden_dim,     # 은닉층 크기
    patch_len=patch_len,       # 패치 길이
    stride=stride,             # 스트라이드
    n_layers=2,                 # Transformer 레이어 수
    d_model=16,                 # 모델 차원
    n_heads=2,                  # 헤드 수
    dropout=0.1                 # 드롭아웃 비율
)
```

Extra experiment

- **실험결과** - MSE 값이 PatchTST와 크게 다르지 않은 것을 확인 (다른 경량화 모델인 Linear, D-Linear 과도 성능 비교할 때 우수)
반면, 모델용량은 **약 4.47배 감소 (ETTh 기준)**



Teacher Model Parameters: 468800
Student Model Parameters: 104848
Compression Ratio: 4.47x
Teacher Model Size: 1861.20 KB
Student Model Size: 420.38 KB

References

- Gajamannage, K., Park, Y., & Jayathilake, D. I. (2023). Real-time forecasting of time series in financial markets using sequentially trained dual-LSTMs. *Expert Systems with Applications*, 223, 119879.
- Li, L., Kang, Y., Petropoulos, F., & Li, F. (2023). Feature-based intermittent demand forecast combinations: accuracy and inventory implications. *International Journal of Production Research*, 61(22), 7557-7572.
- Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.
- Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2023, June). Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 37, No. 9, pp. 11121-11128).

감사합니다.

1398