Time-series price forecasting

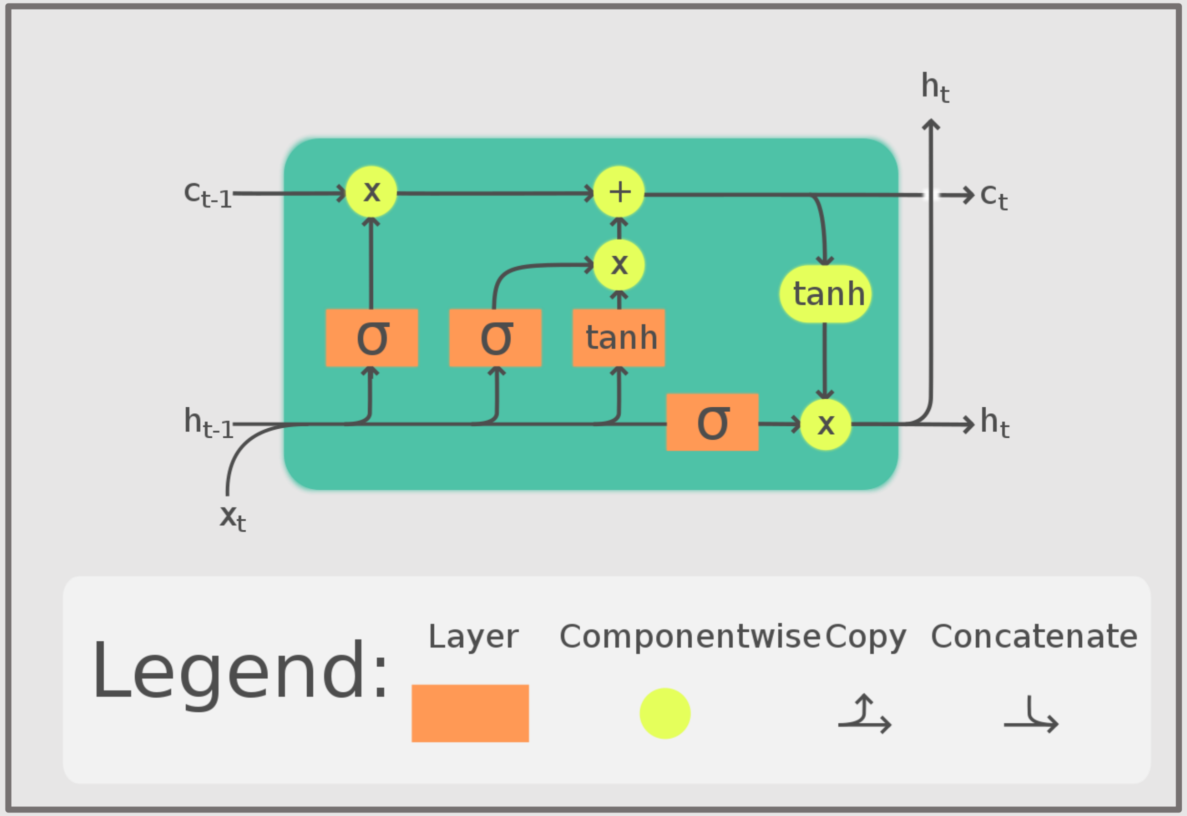
경상국립대학교 석사과정 오서영

1. LSTM (Long short term memory)

LSTM architecture

$$f_t = \sigma(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f})$$
$$i_t = \sigma(W_{xh\_i}x_t + W_{hh\_i}h_{t-1} + b_{h\_i})$$
$$o_t = \sigma(W_{xh\_o}x_t + W_{hh\_o}h_{t-1} + b_{h\_o})$$
$$\hat{c}_t = tanh(W_{xh\_\hat{c}}x_t + W_{hh\_\hat{c}}h_{t-1} + b_{h\_\hat{c}})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$
$$h_t = o_t \odot \tanh(c_t)$$

f : forget gate
i : input gate
o : output gate



이미지 출처 : https://en.wikipedia.org/wiki/Long_short-term_memory

2. Wavenet

Wavenet architecture

Dilated causal convolution

- Dilated convolution layer + causal convolution layer

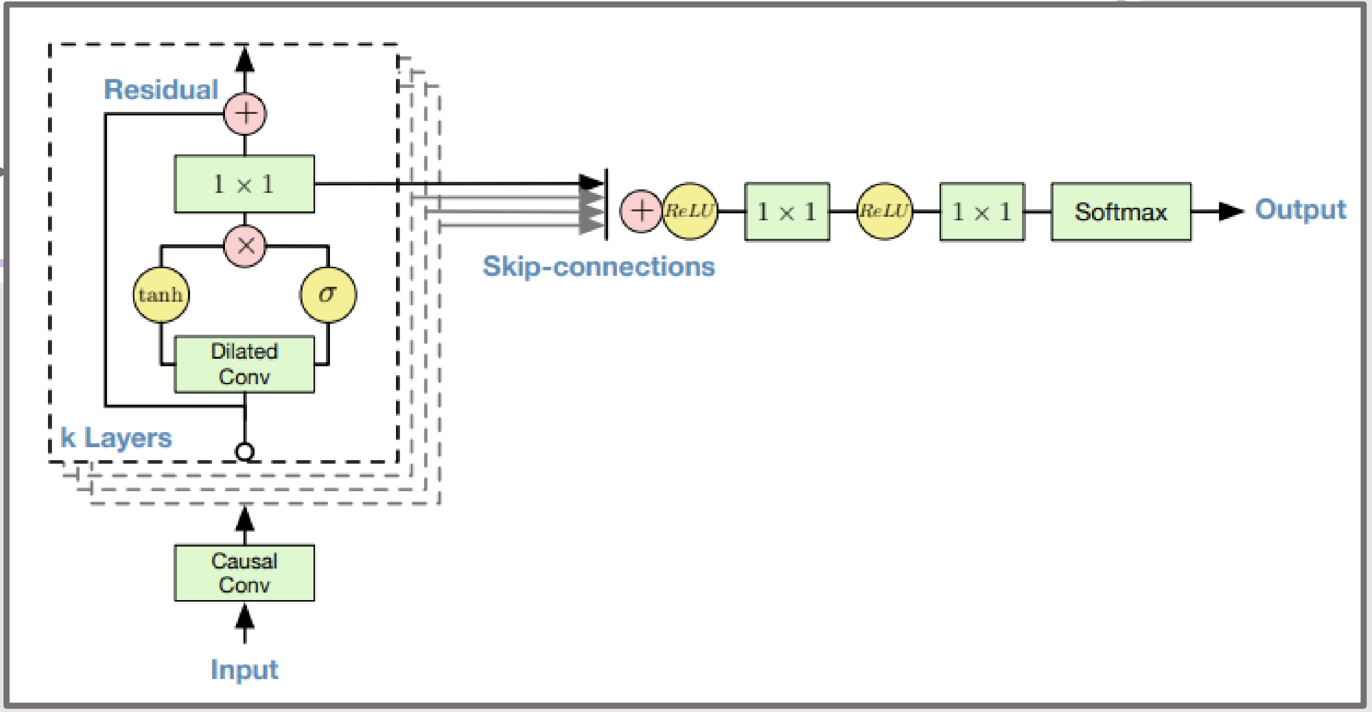
→ Dilated convolution은 추출 간격 (dilation)을 조절하여 특정단계로 입력값을 건너뛰어, 더 넓은 receptive field를 갖게하는 convolution layer (적은 layer, 넓은 receptive field)

→ Causal convolution은 시간순서를 고려한 convolution

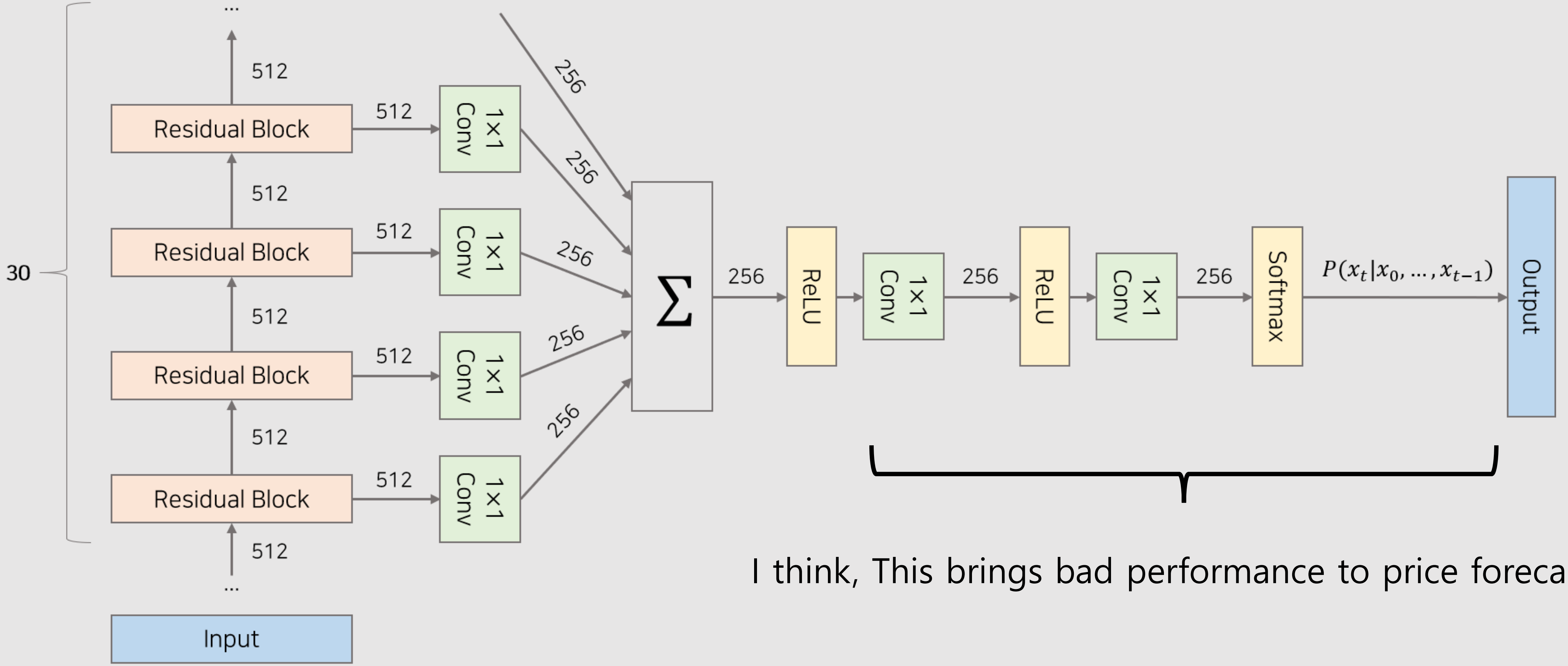
Gated activation units

$$\mathbf{z} = \tanh\left(W_{f,k} * \mathbf{x}\right) \odot \sigma\left(W_{g,k} * \mathbf{x}\right)$$

Residual and skip connection

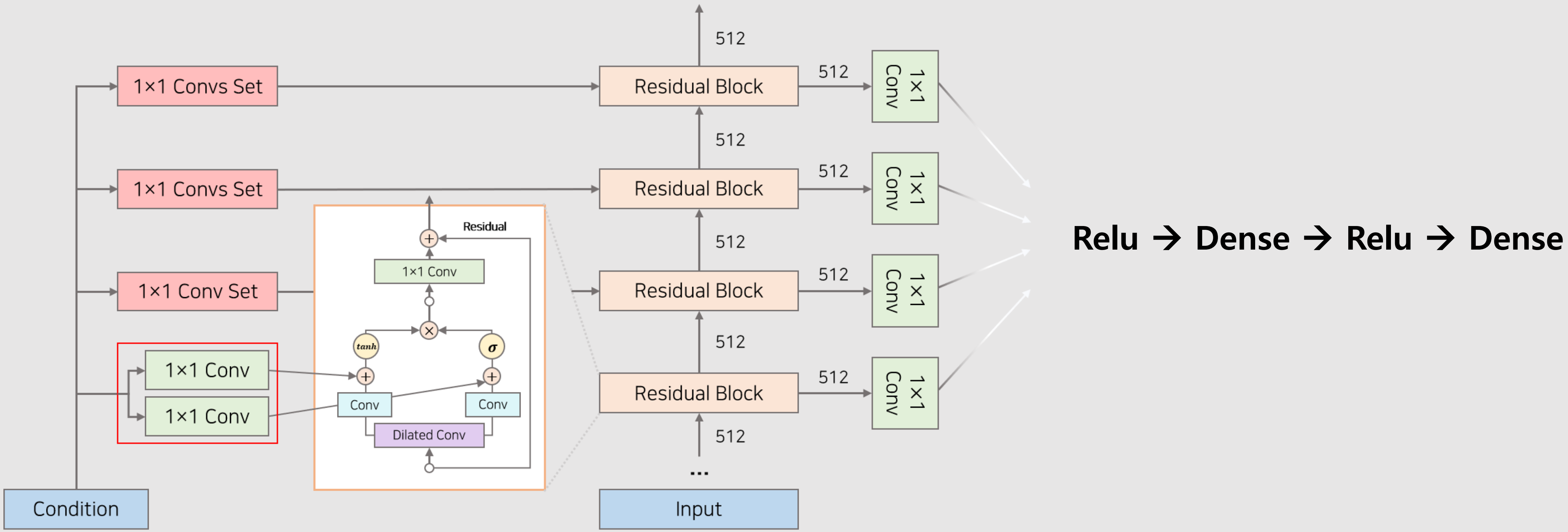


2. Wavenet



I think, This brings bad performance to price forecasting

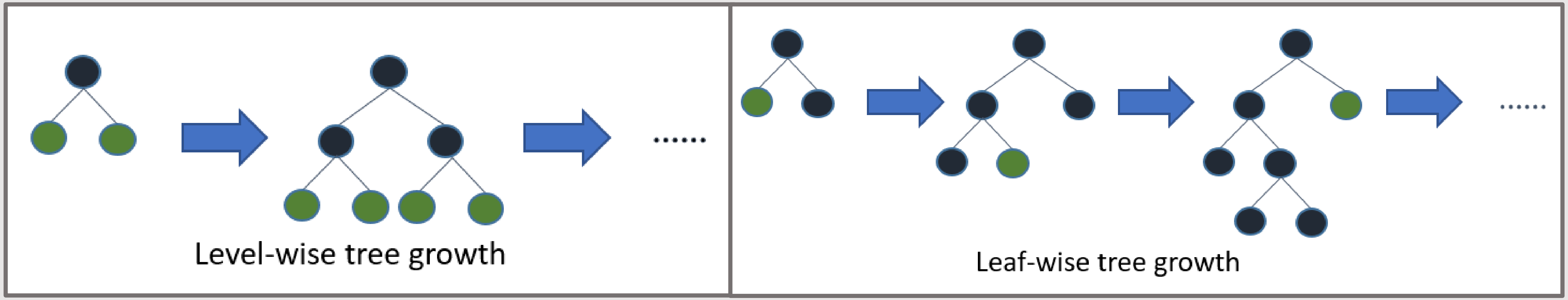
2. Wavenet (Modified)



이미지 출처 : https://joungheekim.github.io/2020/09/17/paper-review/

3. LightGBM

LightGBM
= GOSS + EFB
GOSS (Gradient-based One-Side Sampling
- 데이터의 샘플 수를 줄임
EFB (Exclusive Feature Bundling)
- 데이터의 feature 수를 줄임



Xgboost	**LightGBM**

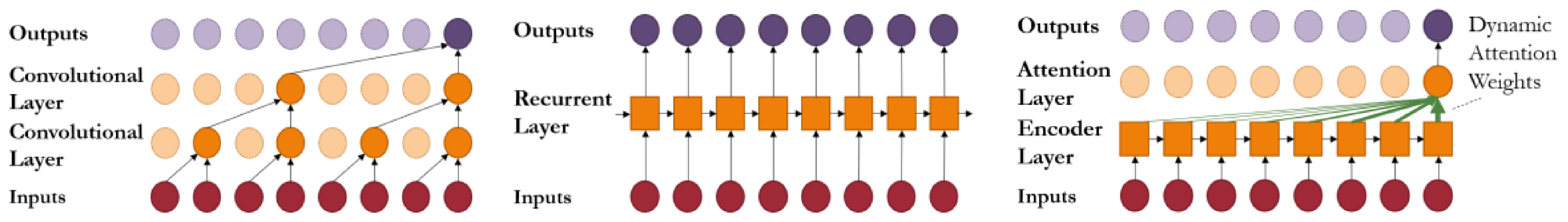
→ 최대 손실을 가지는 leaf 노드를 계속 분할

이미지 출처 : https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost

4. Experiment

시계열 예측 모델링은 금융에서 널리 적용됨
특히 머신러닝 기법은 순전히 데이터 중심의 방식으로 time dynamics를 이해할 수 있는 통찰력을 제공함
CNN, RNN, Attention-based model을 사용하여 시계열 가격을 예측할 수 있음

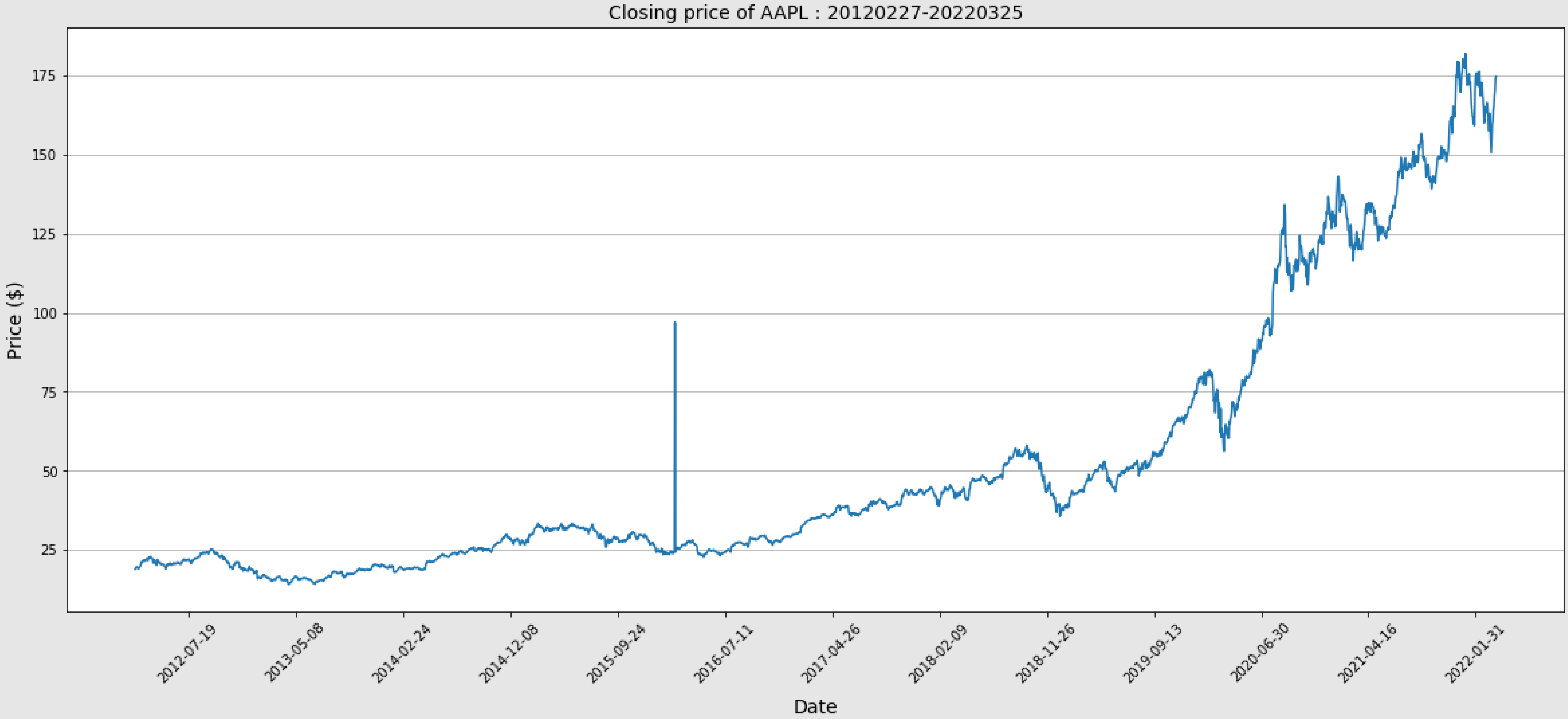
주식 및 ~~비트코인 가격 데이터~~(APPL, ~~BTC~~)를 활용하여 머신러닝, 딥러닝 기반의 시계열 가격 예측을 수행



(a) CNN Model. (b) RNN Model. (c) Attention-based Model.

이미지 출처 : Lim, Bryan, and Stefan Zohren. "Time-series forecasting with deep learning: a survey." *Philosophical Transactions of the Royal Society A* 379.2194 (2021): 20200209.

4. Experiment



4. Experiment

```
[ ]  model = Sequential()

     model.add(LSTM(50, return_sequences=True,input_shape=(X_train.shape[1], X_train.shape[2])))
     model.add(LSTM(50, return_sequences=True))
     model.add(LSTM(50))
     model.add(Dense(1, kernel_initializer=tf.keras.initializers.glorot_uniform(seed=seed_num)))
     model.compile(loss='mean_squared_error', optimizer='adam')
```

```
[ ]  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 1, 50)             10400

 lstm_1 (LSTM)               (None, 1, 50)             20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

LSTM

4. Experiment

```python
def gated_activation_units(x):
    tanh_out = Activation('tanh')(x)
    sig_out = Activation('sigmoid')(x)
    return keras.layers.multiply([tanh_out, sig_out])


def residual_block(x, i, num_filters, kernel_size, padding):
    # i: The dilation power of 2


    prev_x = x
    conv = Conv1D(filters=num_filters, kernel_size = kernel_size, dilation_rate = i, padding = padding)(x) # dilated conv
    x = gated_activation_units(conv)  # gated activation units

    x = Convolution1D(num_filters, 1, padding='same')(x)  # skip connection

    res_x = keras.layers.add([prev_x, x])

    return res_x, x  # residual, skip connection

# input : (batch_size, timestep, input_dim)
```

Wavenet – activation and residual block

4. Experiment

```
# wavenet
input = Input(shape=(X_train.shape[1], X_train.shape[2]))
x = Convolution1D(num_filters, 1, padding = padding)(input)  # causal conv

skip_connections = []
for k in range(num_stacks):
    for i in dilations:
        x, skip_out = residual_block(x, i, num_filters, kernel_size, padding)  # residual and skip connection
        skip_connections.append(skip_out)

x = keras.layers.add(skip_connections)

x = Activation('relu')(x)

# Since then, different from original Wavenet

x = Dense(16, activation="relu")(x)
# x = Dropout(0.1)(x)
output = Dense(1)(x)
```

```
dilations = [1, 2, 4, 8, 16, 32]
num_filters = 64
padding = "causal"
kernel_size = 2
num_stacks = 1
```

Wavenet

4. Experiment

LSTM
- Min-max scaling 진행
- 간단한 stacked LSTM으로도 높은 성능을 냄
- 약 30 epoch

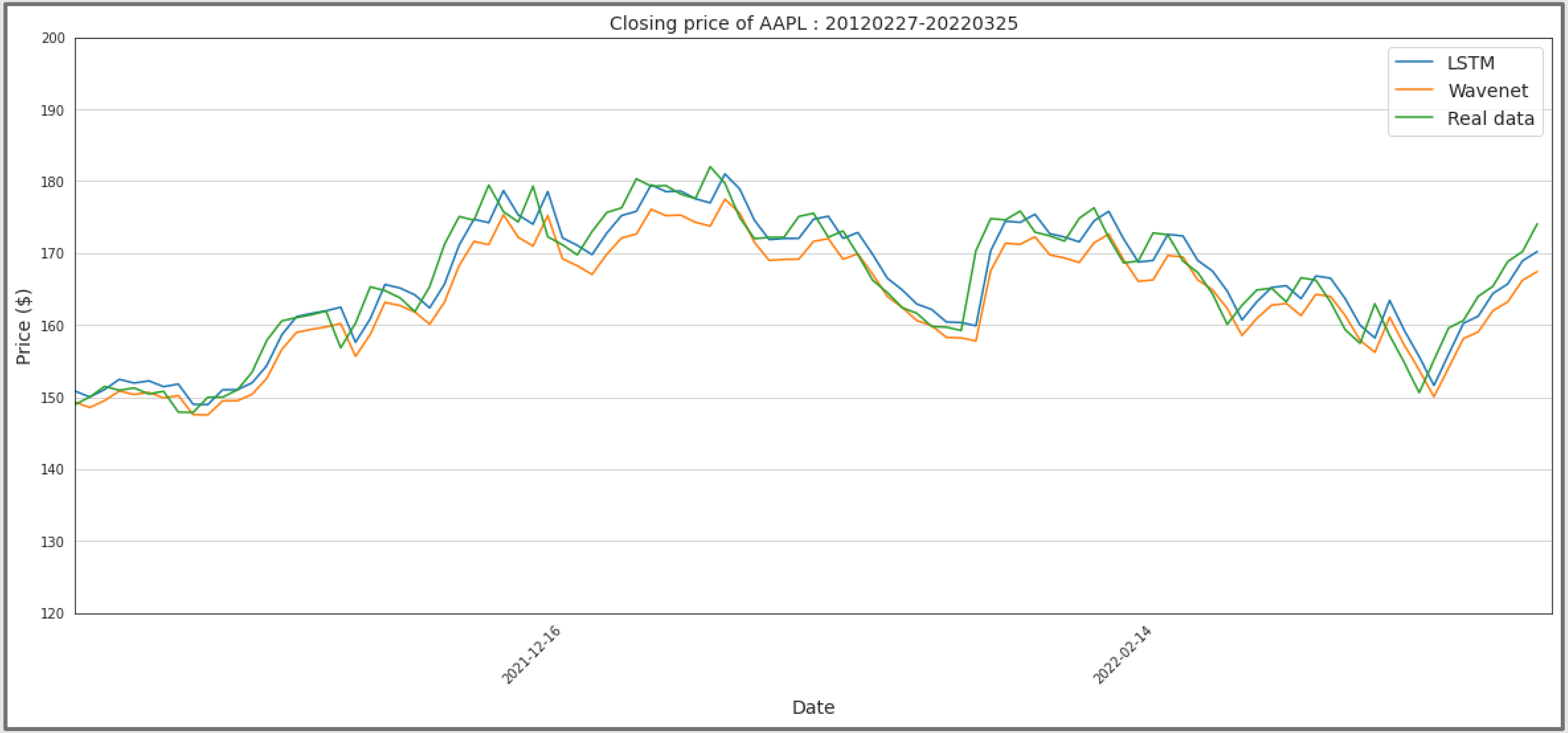
Wavenet
- Min-max scaling 진행
- Skip connection 이후 original Wavenet 대로 Relu->1x1 Conv 구조를 사용했으나 성능이 좋지 않아 Dense로 대체
- 약 10 epoch, Convolution 구조로 인해 더 빠른 학습속도

LightGBM
- Scaling 사용하지 않음
- 매우 빠른 속도
- 저조한 성능

```
Val RMSE of LSTM: 2.688
Val RMSE of Wavenet: 2.779
Val RMSE of LightGBM: 57.708
```

4. Experiment

4. Experiment



4. Experiment

5. Future work

1. 가격 데이터에 많이 사용되는 다른 전처리 기법 적용
- Log, return 등

2. 타 머신러닝 기법 적용
- ARIMA, Prophet, XGBoost 등
- Ensemble

3. 가격 예측 뿐만 아니라 매매에 도움을 줄 수 있는 다른 예측 방식을 고려
- price trend forecasting (falling, or not falling)

4. 자동 매매 프로그램 제작 (미정)

6. References

[1] Lim, Bryan, and Stefan Zohren. "Time-series forecasting with deep learning: a survey." *Philosophical Transactions of the Royal Society A* 379.2194 (2021): 20200209.

[2] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems* 30 (2017).