서울시립대학교
UNIVERSITY OF SEOUL

서울시립대학교 자연과학대학 수학과

산업수학 연구인턴십

2020.10.09

고 지 형

2020 IEEE International Conference on Big Data and Smart Computing (BigComp)

Gene Expression Prediction using Stacked Temporal Convolutional Network

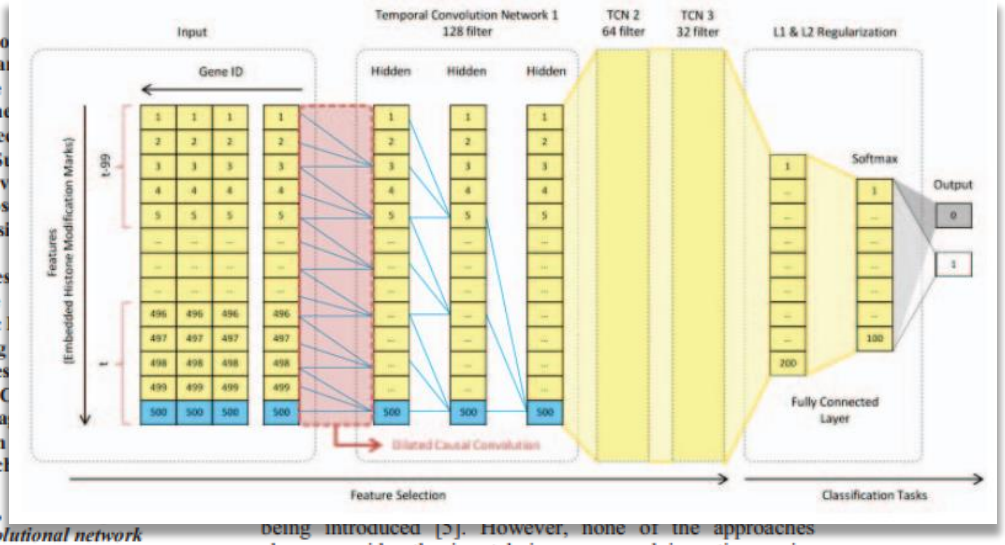
Imam Mustafa Kamal
Big Data Department
Pusan National University
Busan, South Korea
imamkamal@pusan.ac.kr

Nur Ahmad Wahid
Big Data Department
Pusan National University
Busan, South Korea
nawa410@pusan.ac.kr

Hyerim Bae*
Industrial Engineering Department
Pusan National University
Busan, South Korea
hrbae@pusan.ac.kr

***Abstract*— Predicting gene expressio**
important tasks in molecular biology a
Studying the complex combinatorial code
could lead to a better understanding of gene
i.e., how a gene increase or decrease spe
(protein and RNA) through translations. S
be useful to study the origins of cancer, dev
certain disease, etc. In this study, we propos
Histone Modification data into one-dimensi
predicted the gene expression by
Convolutional Networks. Previous studies
methods ranging from classical machine
(e.g., Support Vector Machine and Logistic
as the most recent machine learning
DeepChrome and DeepNN). Experiment res
approach is superior in terms of AUC
precision, recall, f-score, and specificity a
the-art-method, and only slightly worst in
and specificity against Support Vector Mach

Keywords — histone modifications, classification, deep learning, temporal convolutional network



I. Introduction

Gene regulation is the process to control the expression

being introduced [5]. However, none of the approaches above consider the input being processed in a time-series manner.

In this paper, we propose a deep learning approach for

Main Points

TCN(Temporal Convolutional Network) 모델 소개

유전자 발현 예측 모델 Stacked TCN 모델 학습 및 결과

TCN: Temporal Convolutional Network

서울시립대학교 UNIVERSITY OF SEOUL

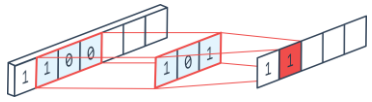
TCN: CNN 계열 시퀀스 학습 모델

- 시퀀스 학습을 위한 CNN 계열 파생 연구 활발
- 이러한 파생 모델 중 하나가 TCN

> Recently, Deep Learning practitioners have been using a variation of Convolutional Neural Network architecture for the sequence modelling tasks, **Temporal Convolutional Networks**. This is a simple descriptive term to refer to a family of architectures.

TCN 모델의 특징

(1) RNN계열 모델과 같이 어떤 길이의 시퀀스를 받더라도 같은 길이의 시퀀스로 맵핑할 수 있음
→ 1D FCN(1D Fully-Convolutional Network 활용)



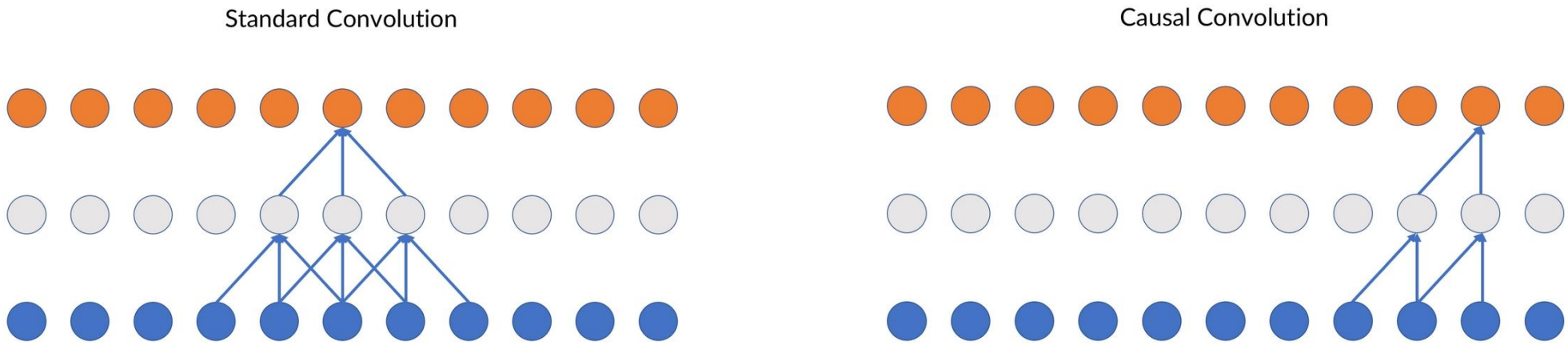
\+ Zero Padding

(2) 미래로부터 과거까지 정보 누출(leakage)문제가 없는 'Causal(인과관계적)' 특성을 지님
→ Causal Convolution 연산 활용

> The distinguishing characteristics of TCNs are:
>
> ① 어떤 길이의 시퀀스라도 받을수있다 + RNN처럼 같은 길이의 시퀀스로 맵핑할 수있다. ⇒ FCN 사용.
>
> - The architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNNs.
>
> ② Convolution 연산이 'Causal'하다. 즉. 과거와 미래 간 누출 (leakage) 문제가 없다.
>
> - The convolutions in the architecture are **causal**, meaning that there is no information "leakage" from future to past.

출처: https://medium.com/@raushan2807/temporal-convolutional-networks-bfea16e6d7d2

Causal Convolution

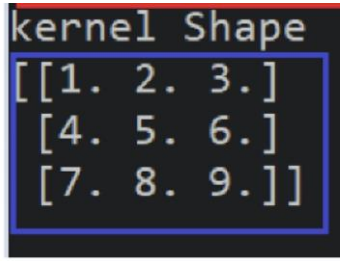


- 데이터가 시계열성을 지니고 있는 경우 과거에 대한 학습 과정에 미래 시점이 포함될 수 없음
- 이를 위해 RNN계열 모델을 활용하지만 학습 및 검증이 오래 걸린다는 단점
- 시계열 문제를 위한 CNN계열 모델 활용 시 이러한 문제를 보완할 수 있음

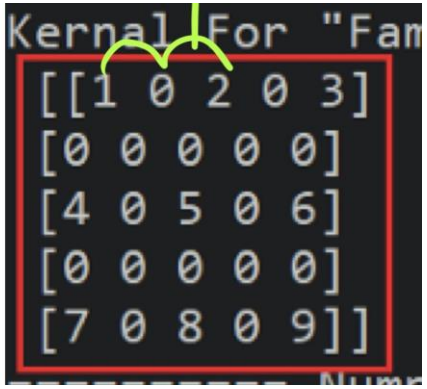
출처: https://subscription.packtpub.com/book/machine_learning/9781789136364/4/ch04lvl1sec59/dilated-and-causal-convolution

Dilated Convolution

- 1D FCN과 Causal Convolution 연산 활용 기본 TCN 모델은 Long Term 타임라인 반영 한계
- 이를 해결하기 위한 방법이 Dilated Convolution 연산
- Convolution 연산을 수행하는 Filter를 듬성듬성 넓히는 개념
- 이때, Filter의 Value 간 간격을 Dilated Factor라고 함
- Convolution 연산 방식은 기존 CNN의 연산과 같음



<Dilated Factor = 1 (일반적인 CNN의 Filter와 같다)>



<Dilated Factor = 2>

출처: https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25

Causal Dilated Convolution

- Causal Convolution 연산과 Dilated Convolution 연산을 결합한 형태
- TCN에서 활용할 수 있는 Convolution 연산 중 하나
- ‘Gene Expression Prediction Using Stacked TCN’ 논문의 TCN 모델에 활용

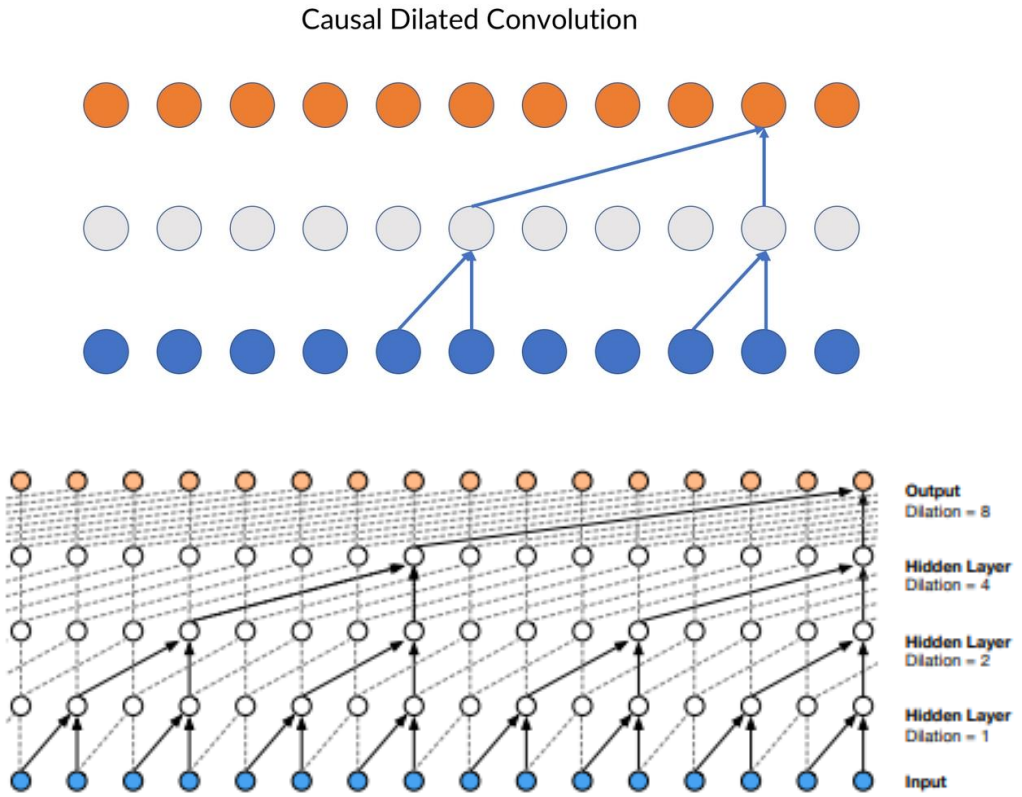


Figure 3: Visualization of a stack of *dilated* causal convolutional layers.

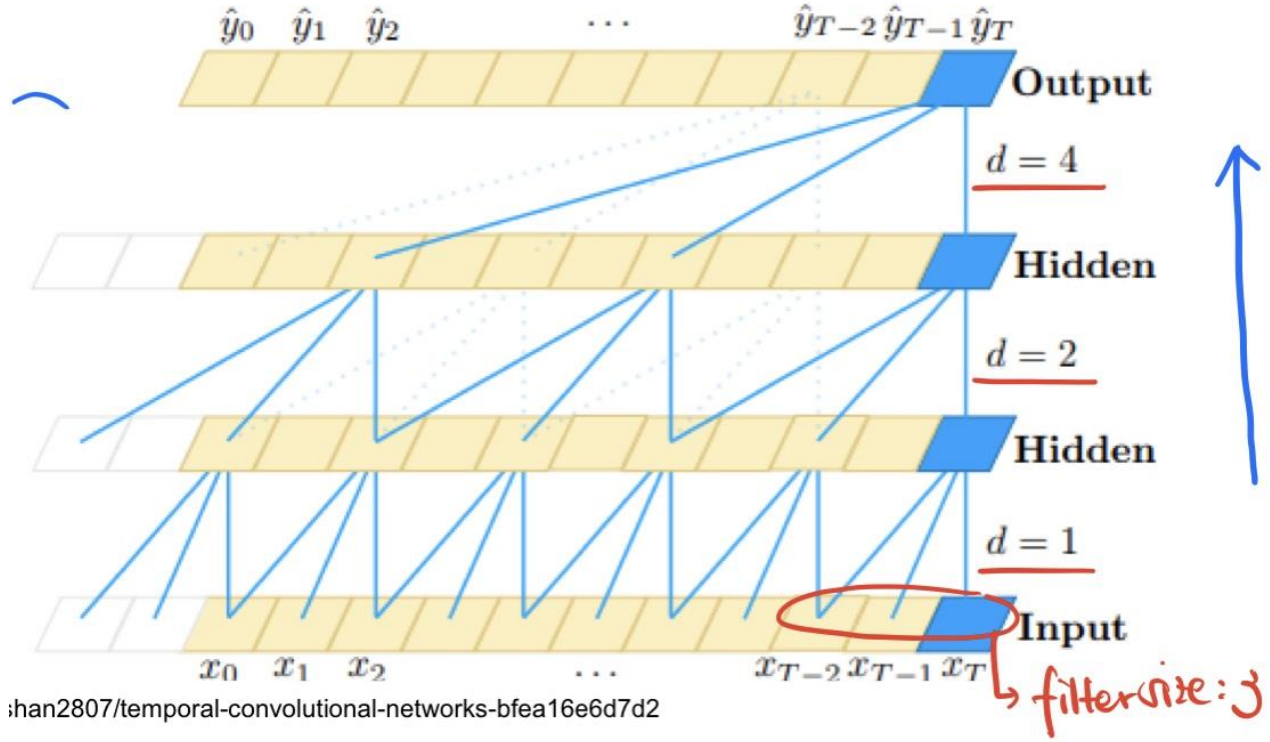
출처: https://subscription.packtpub.com/book/machine_learning/9781789136364/4/ch04lvl1sec59/dilated-and-causal-convolution

출처: https://dataplay.tistory.com/29

TCN: Temporal Convolutional Network

1D Causal Dilated Convolution

- 논문의 TCN에 활용한 Convolution 연산
- Dilated Convolution 연산으로 시퀀스에 대한 전체적 패턴 파악이 유리해짐
- 오른쪽 그림의 경우, 크기가 3인 Filter가 레이어 순서에 따라 1, 2, 4의 Dilated Factor를 가지고 Dilated Convolution 연산을 진행



han2807/temporal-convolutional-networks-bfea16e6d7d2

출처: https://medium.com/@raushan2807/temporal-convolutional-networks-bfea16e6d7d2

TCN 모델 장점

- 병렬처리
  RNN계열 모델은 마지막 타임스텝이 이전 타임스텝들의 연산에 종속되기 때문에 병렬처리 불가
  TCN 모델은 Convolution 연산을 활용하여 병렬처리 가능

- 필드 사이즈 가용성
  Filter의 크기 또는 Dilated Factor를 조절하여 큰 필드 사이즈가 입력되더라도 대처가 편하다

- 낮은 메모리 비용
  TCN 모델은 각각의 레이어에 Filter가 독립적 존재
  가중치 업데이트 시 메모리 비용이 적게 듦

- 뛰어난 Local Information 수집
  Convolution 연산의 장점
  Temporal Information으로부터 Local Information을 잡아내기에 용이

유전자 발현 예측 Stacked TCN 모델 구조

서울시립대학교 UNIVERSITY OF SEOUL

학습 데이터

- 특징 변수: 1~500
  유전자 내 히스톤 변형과 관련된 표식의 개수
  유전자를 500개 구간으로 나누어 관찰한 것이므로 순서(Sequence)를 지님 → TCN 활용
  최대·최소 Scaling

- 타깃 변수: Label
  유전자가 발현되었다 → 1
  유전자가 발현되지 않았다 → 0

TABLE I. MULTI-DIMENSIONAL DATA OF GENE EXPRESSION

Row	*GeneID*	*HM1*	*HM2*	*HM3*	*HM4*	*HM5*	*Label*
1	1	0	7	0	5	2	1 (on)
2	1	0	0	1	7	0	1 (on)
…	…	…	…	…	…	…	…
100	1	1	2	0	8	9	1 (on)
1	2	0	1	4	0	0	0 (off)
2	2	0	1	8	1	3	0 (off)
…	…	…	…	…	…	…	…
100	2	4	0	1	3	7	0 (off)

<raw 데이터>

TABLE II. ONE-DIMENSIONAL DATA OF GENE EXPRESSION

Row	*GeneID*	*1*	*2*	…	…	…	*499*	*500*	*Label*
1	1	0	7	…	…	…	8	9	1 (on)
1	2	2	0	…	…	…	3	7	0 (off)

<학습 데이터>

유전자 발현 예측 Stacked TCN 모델 구조

서울시립대학교 UNIVERSITY OF SEOUL

모델 구조

- 입력층

  길이가 500인 시퀀스 데이터를 입력

  은닉층으로 넘어갈 때 Dilated Causal Convolution

- 은닉층

  3개의 Stacked TCN Layer

	레이어 개수	Filter 개수	Filter 크기	Dilated Factor
#1 TCN	3	128	3	2 → 4
#2 TCN	3	64	3	2 → 4
#3 TCN	3	32	3	2 → 4

  Fully Connected Layer(Dense Layer)

  #1 Layer: Unit 개수 200

  #2 Layer: Unit 개수 100, SoftMax 활성화 함수

- 출력층

  Unit 개수 1, Sigmoid 활성화 함수

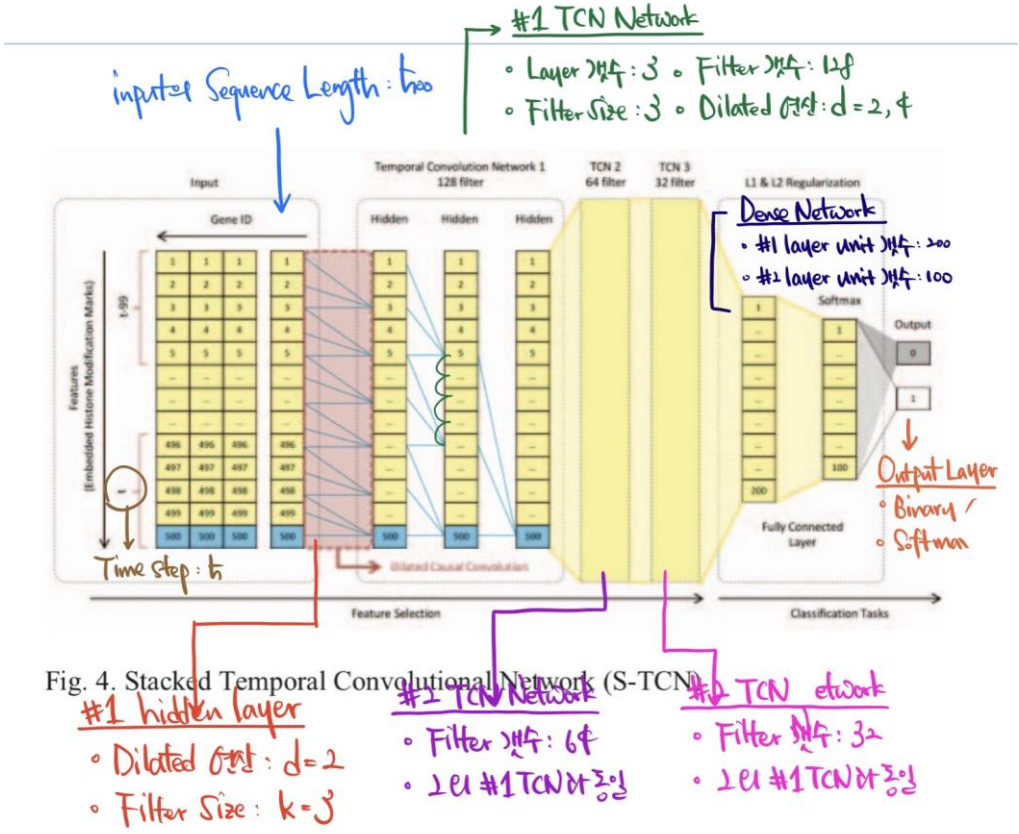


Fig. 4. Stacked Temporal Convolutional Network (S-TCN)

<논문에 기재된 모델 구조>

Keras-TCN

- 현재 TensorFlow와 결합하여 활용하기에 가장 완성도 높은 TCN 라이브러리
- TensorFlow의 Keras API와 연동 가능
- 아래 코드와 같이 TCN 레이어를 구축할 수 있다

```
i = Input(batch_shape=(batch_size, timesteps, input_dim))

o = TCN(return_sequences=False)(i)  # The TCN layers are here.
o = Dense(1)(o)

m = Model(inputs=[i], outputs=[o])
m.compile(optimizer='adam', loss='mse')

tcn_full_summary(m, expand_residual_blocks=False)

x, y = get_x_y()
m.fit(x, y, epochs=10, validation_split=0.2)
```

<구현 예시>

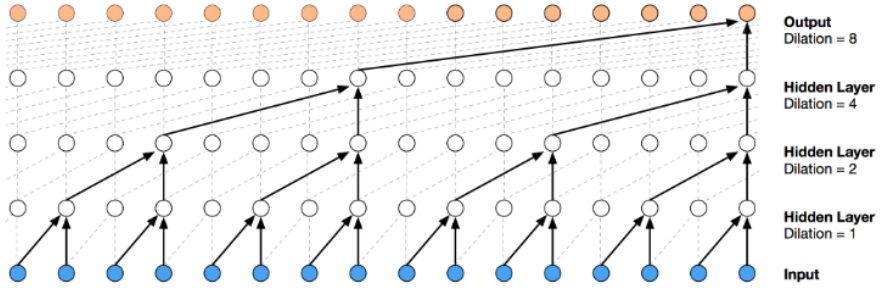
Keras TCN

Compatible with all the major/latest Tensorflow versions (from 1.14 to 2.2+).

downloads 100k | downloads/month 8k | Keras TCN CI passing

Why Temporal Convolutional Network?

- TCNs exhibit longer memory than recurrent architectures with the same capacity.
- Constantly performs better than LSTM/GRU architectures on a vast range of tasks (Seq. MNIST, Adding Problem, Copy Memory, Word-level PTB...).
- Parallelism, flexible receptive field size, stable gradients, low memory requirements for training, variable length inputs...

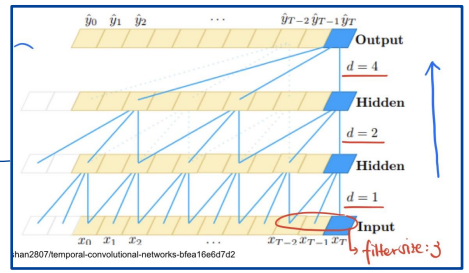


Visualization of a stack of dilated causal convolutional layers (Wavenet, 2016)

출처: https://github.com/philipperemy/keras-tcn

Keras-TCN 주요 파라미터

- nb_filters: Convolution 연산을 수행할 Filter의 개수
- kernel_size: Filter의 크기



- dilations: Dilated Convolution 연산을 위한 Dilated Factor의 값
- padding: 'causal'로 설정 시 Causal Dilated 연산

Arguments

```
TCN(nb_filters=64, kernel_size=2, nb_stacks=1, dilations=[1, 2, 4, 8, 16, 32], padding='causal',
use_skip_connections=False, dropout_rate=0.0, return_sequences=True, activation='relu',
kernel_initializer='he_normal', use_batch_norm=False, **kwargs)
```

- `nb_filters` : Integer. The number of filters to use in the convolutional layers. Would be similar to `units` for LSTM.
- `kernel_size` : Integer. The size of the kernel to use in each convolutional layer.
- `dilations` : List. A dilation list. Example is: [1, 2, 4, 8, 16, 32, 64].
- `nb_stacks` : Integer. The number of stacks of residual blocks to use.
- `padding` : String. The padding to use in the convolutions. 'causal' for a causal network (as in the original implementation) and 'same' for a non-causal network.
- `use_skip_connections` : Boolean. If we want to add skip connections from input to each residual block.
- `return_sequences` : Boolean. Whether to return the last output in the output sequence, or the full sequence.
- `dropout_rate` : Float between 0 and 1. Fraction of the input units to drop.
- `activation` : The activation used in the residual blocks o = activation(x + F(x)).
- `kernel_initializer` : Initializer for the kernel weights matrix (Conv1D).
- `use_batch_norm` : Whether to use batch normalization in the residual layers or not.
- `kwargs` : Any other arguments for configuring parent class Layer. For example "name=str", Name of the model. Use unique names when using multiple TCN.

출처: https://github.com/philipperemy/keras-tcn

TensorFlow를 활용한 구현

서울시립대학교 UNIVERSITY OF SEOUL

학습 결과

- 논문과 달리 저조한 성능
- 머신러닝을 활용한 학습보다 성능이 좋지 않았다

	XGB	LGB	RF	SVM	TCN
AUROC	0.9143	0.9141	0.9083	0.9067	0.8107
Recall	0.8657	0.8544	0.8601	0.8518	0.7357
Precision	0.8416	0.8406	0.8365	0.8452	0.7440
F1	0.8535	0.8474	0.8481	0.8485	0.7398

$$A = \int_{\infty}^{-\infty} TPR(T)\,FPR'(T)\,dt$$

$$F = 2 \times \frac{REC \times PRE}{REC + PRE}$$

F1-스코어

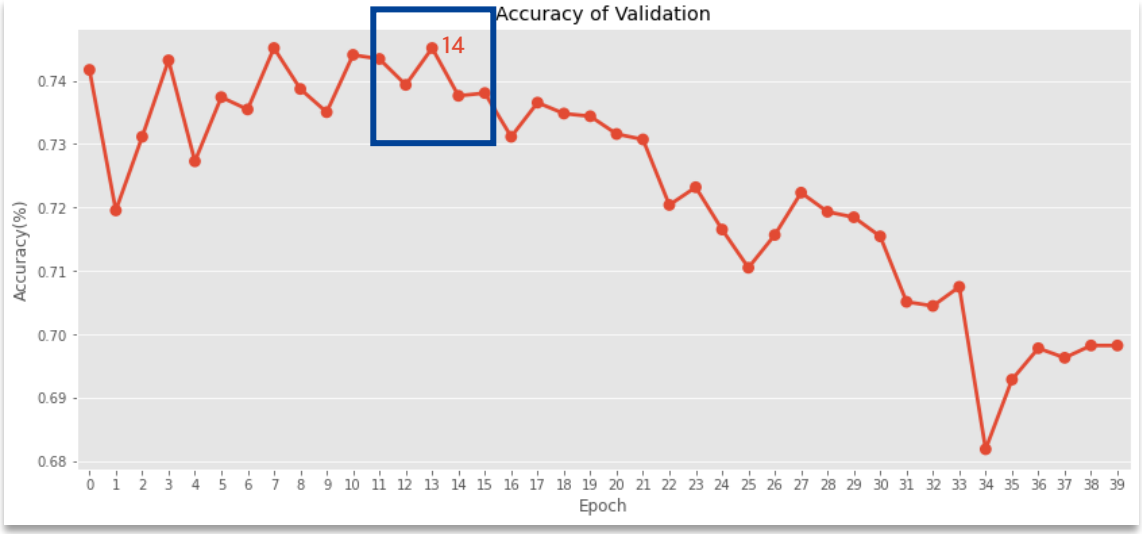
$$REC = \frac{TP}{TP + FN}$$

재현율

$$PRE = \frac{TP}{TP + FP}$$

정밀도

최적 Epoch에서 재학습



<Epoch에 따른 성능 평가>

향후 계획

- 알츠하이머 유전자 발현 데이터
- 췌장암 유전자 발현 데이터

Reference

Temporal Convolutional Networks: https://medium.com/@raushan2807/temporal-convolutional-networks-bfea16e6d7d2

Keras-TCN: https://github.com/philipperemy/keras-tcn

2D Dilated Convolution: https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25

Causal and Dilated Convolution: https://subscription.packtpub.com/book/machine_learning/9781789136364/4/ch04lvl1sec59/dilated-and-causal-convolution

감사합니다