



Deep Learning for Nucleotide Sequencing Data

DeepBind (2015, Nature biotechnology)

Presenter

Gwangho Lee

Computer Science Engineering, Pusan National University
Machine Learning & Bioinformatics Lab

Contents

1. 실습 연구 소개
2. 실습 및 예시 코드
3. 후속 연구 소개

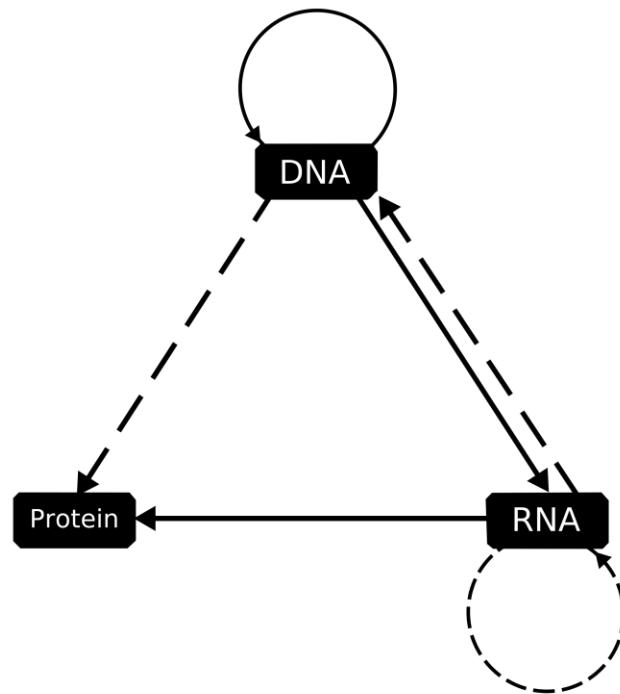
1. 실습 연구 소개 - DeepBind (2015, Nature biotechnology)

Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning

Published paper : <https://www.nature.com/articles/nbt.3300>

분자생물학의 중심 원리 (Central Dogma of Molecular Biology)

단백질로 만들어진 (유전)정보는 다른 단백질이나 핵산으로 전달될 수 없다



분자생물학의 중심 원리.
유전 정보가 전달되는 과정



일반적인 전이 과정

1. 세포내 유전정보 DNA가 스스로를 복제
2. DNA가 RNA를 생성하는 전사(transcription)
3. RNA에서 단백질(protein)을 생성하는 번역(translation)



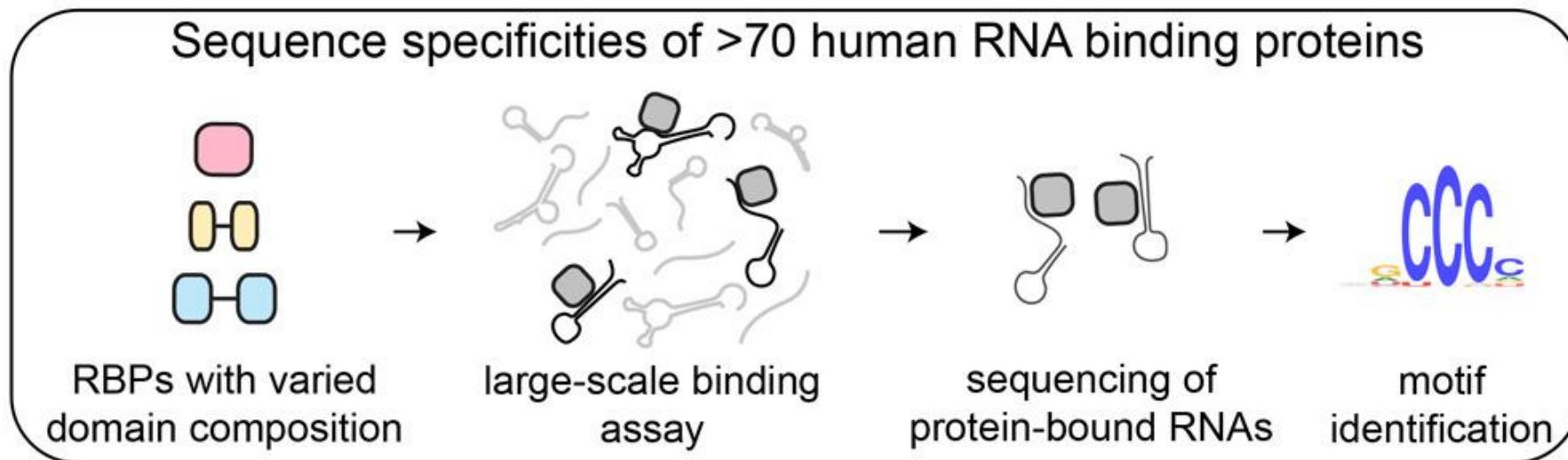
특수한 전이 과정

- 추정, 연구단계에서 논의되는 전이 현상
- 예를 들어 RNA에서 DNA를 생성하는 과정

그 외, 센트럴 도그마에 위배되는 사례

- 역전사를 통한 유전정보 전달되는 예 - [레트로바이러스](#)
- 유전정보 없이 형질 전달되는 예 - [프리온](#)
- 후성유전학 - [일란성 쌍둥이](#)

RNA Binding Proteins (RBPs)



“Knowing the sequence specificities of DNA- and RNA-binding proteins is essential for developing models of the regulatory processes in biological systems and for identifying causal disease variants.”

DNA- 또는 RNA-결합 단백질들에 대한 서열 특이성을 밝혀내는 것은 생물학적 시스템들이나 질병의 인과관계를 규명하는 것에 아주 핵심적인 역할을 한다

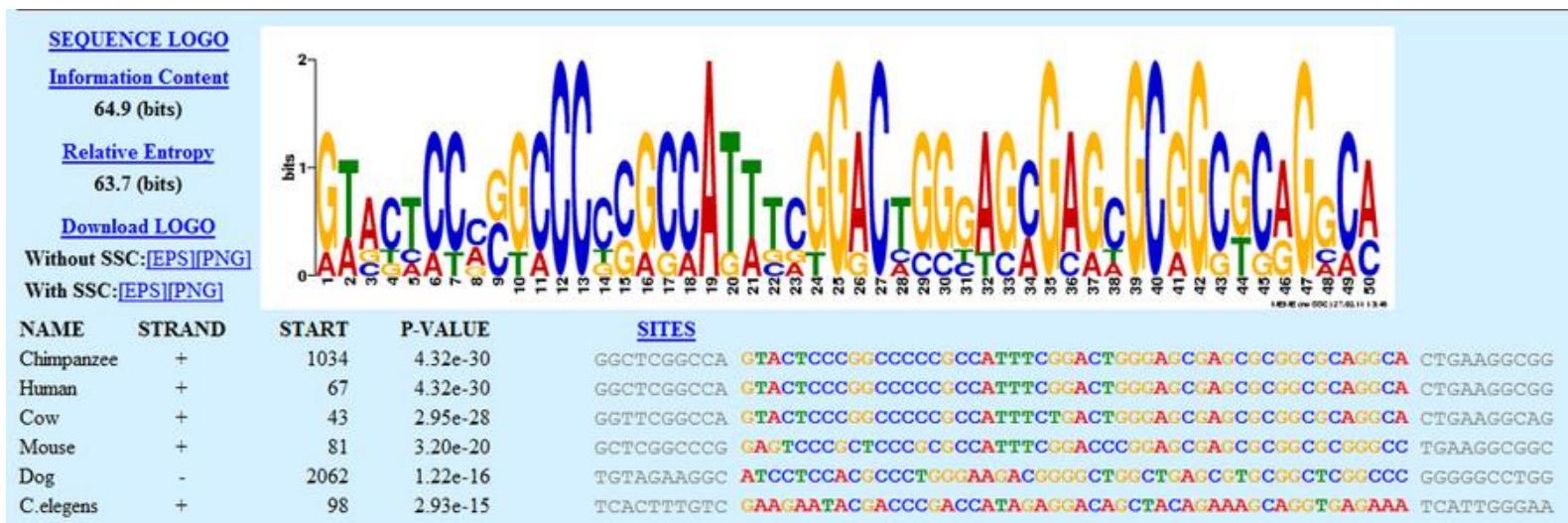
DeepBind (2015)

✓ 연구의 목표

대부분의 DNA- and RNA-binding proteins 연구와 유사하게 해당 단백질에 대한 motif를 찾는 것

✓ Motif(모티프)란 무엇인가?

유전학에서 뉴클레오타이드 또는 아미노산 서열 패턴을 의미, 이를 sequence motif라고 함



DeepBind (2015)

✓ 연구의 동기

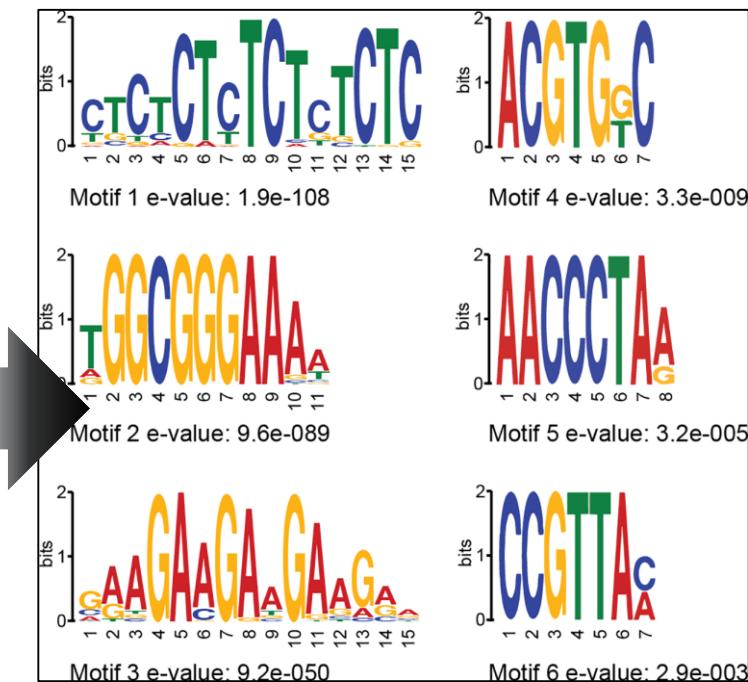
Q. 부분 서열 탐색 알고리즘으로 패턴 찾으면 될 일이 아닌가? 🤔

A. 전통적인 연구 방향성이 맞음, 하지만 최근의 머신러닝 & 딥 러닝 기술의 발전을 통해 새로운 접근을 시도하는 중

```

1 FoldID→EventID seq=Bound
2 A seq_0001_peak=GCTGGAAAGCCTTCGGGTGTCAGAACACTCACTGGCAAGGATCCCTCTGAACACAGCATTTGGAACCCCTTCAGTAGGGAGTCCTCATTGTTTCAGC=1
3 A seq_0003_peak=GAGTTCTTGATGTGCAACAAGCTGAGAGCTCAGCTGAAGGATTCCCACACTGATTACATTCAAAGGGTTTCTCTGTGAGTTCTGTGAGCA=1
4 A seq_0005_peak=ACCTACAAATGTCGGAGGTGGGAGAGGTTCCAGCAGCAGCTCACCTTACCTGAGCATCACAGATCACACAGGTAAGAAACCATATGAATGTTCTG=1
5 A seq_0007_peak=TTGCCACATTCCACATTGGTAGGTTCTCGCAGTGAATTCTTCTATGTAAGGTATGAGAACAGGTTAAAGCTTGGCCACATTCTCAC=1
6 A seq_0009_peak=GCCTGAGTCAGCATGCTGAGTCACTGGAGAGAAGCCTTAAATGCAACAGTGTAAAAGCCTTGGTTAGTTCAGGGTATTCCGACATCAG=1
7 A seq_0011_peak=ACTCAGCTCATCGGCATCTGCAAATTCACTGGGAGAAGCGTACAAATGCAACTAGTGCATAAAAGCCTTGCAGGAGCTCTACCTGTGATGCA=1
8 A seq_0013_peak=CACTGACTGTGGAAAAGCCTTCATCGATCAGTCATCCCTAAGAAACACACAGCTCACACTGGAGAGAAGCCTTATGAGTGTAAAGCTGTGAAAGT=1
9 A seq_0015_peak=GTGATCGGTCAAGGTCTTACAGCACAGAGAACTCATACTGGGAGCGGCATATGAGTGTAAAGAATGTGGAAAGCCTTGGCTACTGCTCAGCCCTG=1
10 A seq_0017_peak=CAGTGACCGGTTCTCTCAACCAGCAGAGGCAACTCACGGGGAGAACCCCTATGAAATGTAAGCAGTGTGGGAGAGCCTTCAGCCAGGGTCTCC=1
11 A seq_0019_peak=TGAGACAACTGAAAGCCTGGCACATTCACATGACACATACTGGTTCTCCAGTGTGAAGGTTCTCCAGTGTGAAGGAACTTGGAGTACAGGAAGGCT=1
12 A seq_0021_peak=GCCAGGATATTCTGTTAAAGCTTCTGACACTGTTGCAAGCTTCTCGCAGCATAGGGCTCTCGCAGGATGAGTCTCTGATGCTGAGGTACGACTCTGATG=1
13 A seq_0023_peak=TAATGACTGTGGGAGAGACTTCTGACATTACAGACCTTACTGACCATGAGGATCATACTGCAGAGAACCCCTATGTTGAGCAGGCTTGTAGTC=1
14 A seq_0025_peak=CCCTATCGTGTCTGAATGTAAGCCTTACAGGGGGAGCTTCCCTATCTACATGAGAACTCATACTGGAGAGAACCCCTATGTAAGGT=1
15 A seq_0027_peak=CTACACATCAGAGAAGTCATACTGGAGAGAGACCTACAATGTAAGGATGTGCAAGGCTTCAACTCTAGGTACATCCACTACACATCGGAGAAGA=1
16 A seq_0029_peak=ATGCCAACAAATTGCAAGGTTGGTAAAGCCTTCCACATTCTACAAACATAGGGCTCTCCAGTGTGAATCTCATGTCAGGTGAAGGAAG=1
17 A seq_0031_peak=AGCACAGGACTGTCTCGAACCGTAACTCGTAGGGCTTCACCGTACAGTGTGAGCTCTGGTGCAGATGAGGCCCGATGTTCT=1
18 A seq_0033_peak=TAAGTCTGAGCCAGACTAACGGCCCTCCACATTCTTACATTCTAGGGTTCTCCAGTGTGAGCTCTGAGTGTGAGGTTGAGCCAGA=1
19 A seq_0035_peak=AACCATGATGTAAGAATGTGGAAGGCTTCACTGCTGTCAGTGTGAAACATGGAGAGGTTCATCTGGTGAAGAATCTTGTGTC=1
20 A seq_0037_peak=TAGGGCTCTCTCAGTGTGTTCTGAGTGTGACTCGGATTGAAGGGCTTCCACATGCTTACACTCATAGGGTTTCTCAGT=1
21 A seq_0039_peak=CCAGAGAATTCATACAGGAGAGCGACCCATGAGTGTAAAGAATGTGCAAAACCTTCTTAAGAAGTCAACCTTATCATACATCGAGAAGATCAGGG=1
22 A seq_0041_peak=AACATCGAGAATTCACTAGGAGAGAACCCATAAATGTCAGGAAAGCCTTAAAGCTTCTGAGTGTAAAGTCACTGCTTACCCGGCATCAGAGAATTCA=1
23 A seq_0043_peak=AGACCCCTACGAATGTAAGGAATGTAGGAAAGCCTTCAGGCCAGTATGCACACCTTGCTCAACATCAGAGAGTTCATCTGGAGAAAAACCTTATGAATGTA=1
24 A seq_0045_peak=CAACATCTGGGATTCTACATGGAGAACGGCTTAAATGTAATCTGTTGAAAGCTTCTGGCGCATCACATCCCTACTGAACATCAGACTCTA=1
25 A seq_0047_peak=GTGGCTCAGCACTTACATCAGAGAATTCACTCTGGAGAACCCCTATGATTGTAAGGAATGTGGAAGGCTTACTCAGGTCACAGCTCTG=1
26 A seq_0049_peak=GAGAACACACAGGGGAGAACCTGATGAATGTCAGATGAGGAATGCTTCACTTGTGAGTGTGAAAGTCACACTCATGGTCAAGAGAACTCACAGGGGAG=1
27 A seq_0051_peak=GCATTCAGTGAATCTCTGTCAGTAAACGACATGAGAGAATTCACTGGAGAGAACCCATATGAGTGCCTGATGTGGAAAGCCTTACTGAATCTC=1
28 A seq_0053_peak=TTGTCACATTCTGTCAGTCAATGTTCTGAGTGTGTTCTGATGTTCTGAGTGTGTTCTGATGTTCTGAGTGTGAGGTTCTGGAGAACGCTTACACTGCA=1
29 A seq_0055_peak=TGTGCAAGCCTCAACCGGGCTTACATCTCACTGGACACAAAGAACTCATACTGGAGAGAACAAATACATGTGAAGAATGTGCAAGAACCTTTAA=1
30 A seq_0057_peak=GTACTGAGTGCAGGCAACCTTCAGCCAGGGTCAACTCTAGTATTACACTTGTGCAATCCACAGGAGAGAACCCATATGAGTGTGCAAGATGTGGAAG=1

```



DeepBind (2015)

DeepBind (2015) 모델 연구에 대한 개요 (그림1)

1. High-throughput experiments – 유전체, 분자생물학 실험 데이터 수집 및 가공 단계
2. Massively parallel deep learning – 실험 데이터를 사용한 머신러닝, 딥러닝 모델링 단계 (학습)
3. Community needs – 학습된 모델을 바탕으로 유의미한 분석을 수행하는 단계

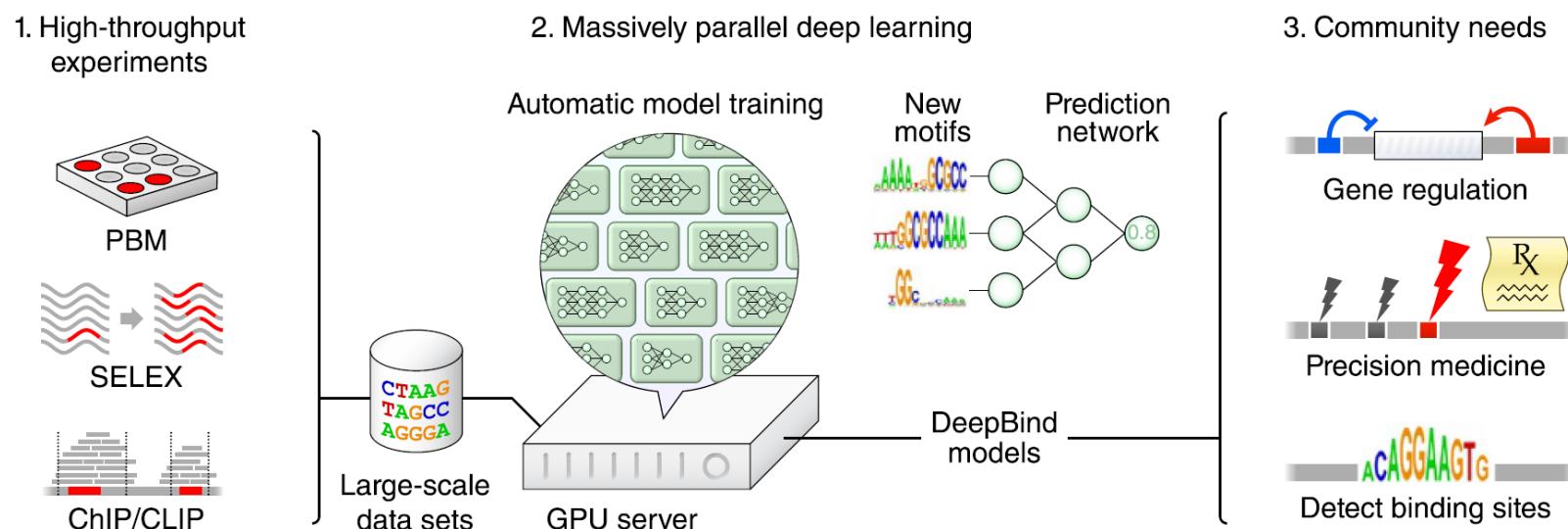


그림1. DeepBind's input data, training procedure and applications

DeepBind (2015)

DeepBind (2015) 모델 연구에 대한 개요 (그림1)

1. High-throughput experiments – 유전체, 분자생물학 실험 데이터 수집 및 가공 단계
2. Massively parallel deep learning – 실험 데이터를 사용한 머신러닝, 딥러닝 모델링 단계 (학습)
3. Community needs – 학습된 모델을 바탕으로 유의미한 분석을 수행하는 단계

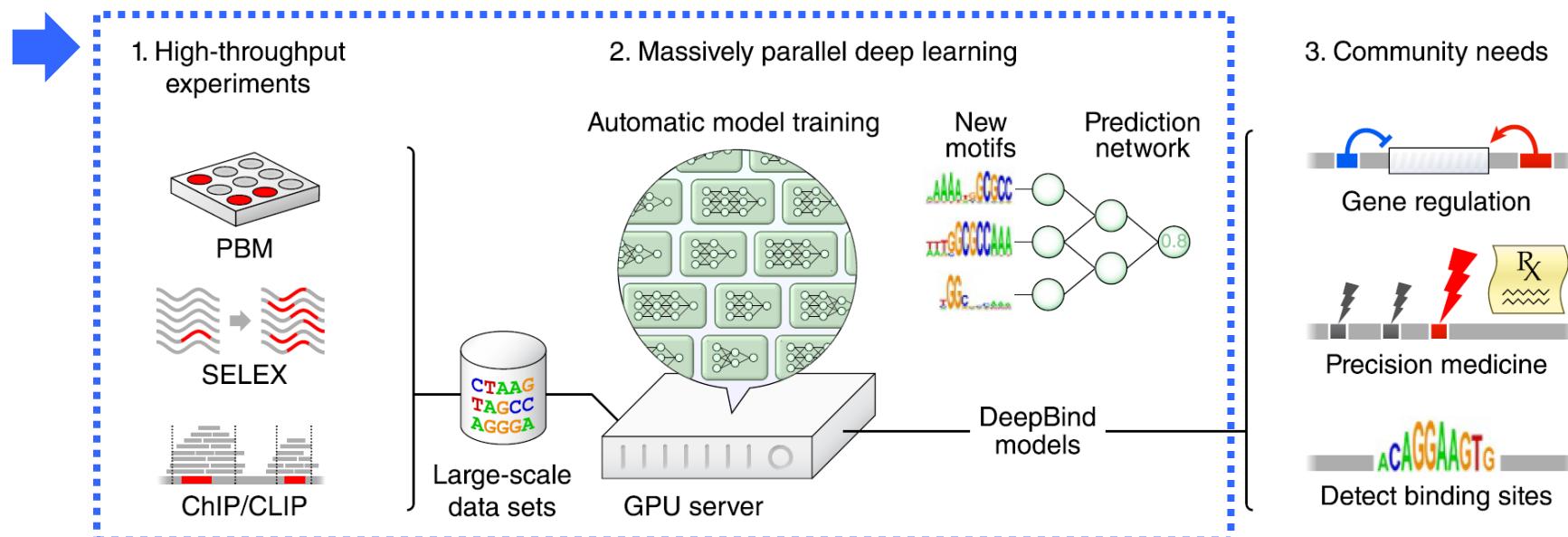
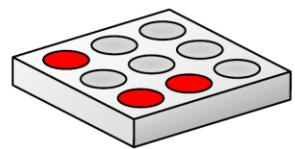


그림1. DeepBind's input data, training procedure and applications

High-throughput experiments, 용어 및 내용 정리

1. High-throughput experiments



PBM



SELEX



ChIP/CLIP

High-throughput

- 일반적으로 1회 정도 할 수 있던 일을 100회 이상 대용량고효율로 처리하는 것을 의미
- High-throughput platform을 갖추다 ~ 라는 표현으로 많이 사용

Sequencing

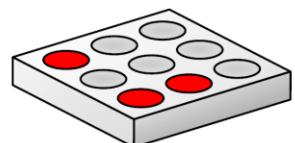
- 시퀀싱(sequencing)이란 DNA 서열을 분석하기 위해 생화학적 방법으로 모든 세포의 DNA 사슬을 구성하는 염기(A, C, G, T)가 결합된 순서를 분석하는 기술

NGS

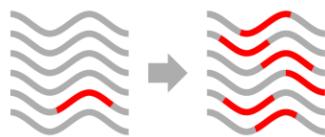
- Next Generation Sequencing 의 약자, 유전체의 염기서열을 고속으로 분석하는 기법
- 하나의 유전체를 수많은 조각으로 분해하여 각 조각을 동시에 읽은 후 전산 기술(ex. alignment algorithm)으로 조합하여 유전체 정보를 해독하는 방법

High-throughput experiments, 용어 및 내용 정리

1. High-throughput experiments



PBM



SELEX



ChIP/CLIP

PBM

- Protein (Binding) Microarray 실험의 약자 (high-throughput 실험)
- 특정 리간드(ligand)에 결합하는 수용체(receptor)를 검출하는데 사용 가능

SELEX

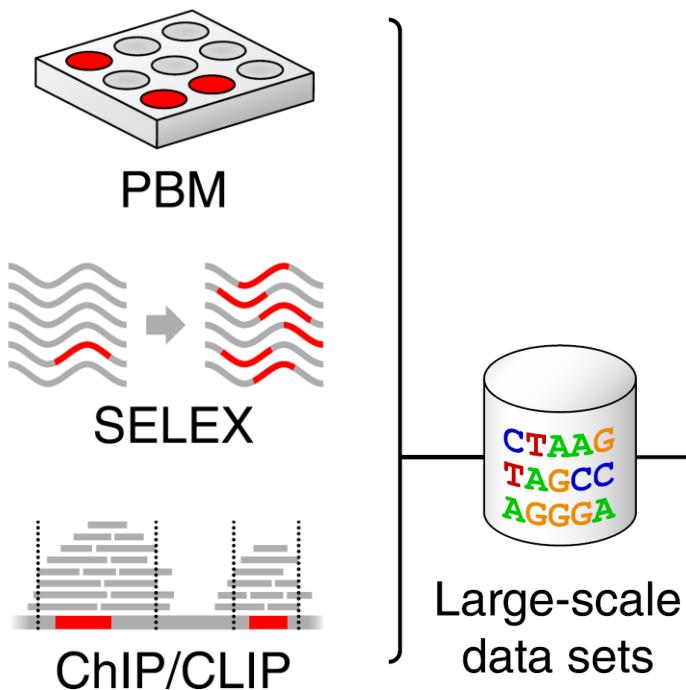
- 특정 수용체(ex. Protein)에 결합하는 인공적인 DNA/RNA 서열 생성 실험

ChIP-seq

- 특정 단백질이 결합(binding)하는 DNA서열을 NGS로 시퀀싱
- 유전체 전체에서 결합 가능한 motif 를 탐색하기위해 사용

High-throughput Sequencing Data

1. High-throughput experiments



Large-scale data sets

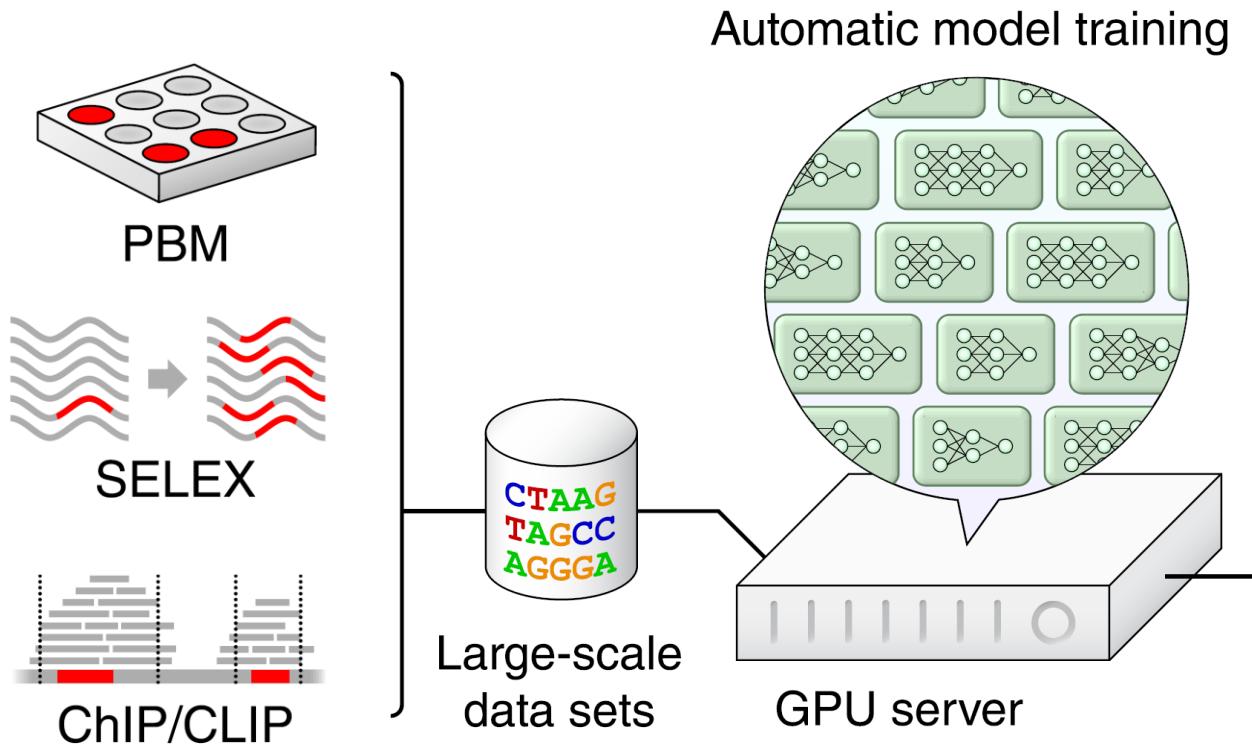
- 표적 단백질에 결합한 DNA/RNA 서열 정보를 포함한 텍스트 파일들

1 FoldID="EventID seq=Bound
2 A="seq_00001_peak"=GCTGGAAAGCCTTCGGGTGCCCCAGAACACTCTGGCAAG
3 A="seq_00003_peak"=GAGTTCTTGTGTCACAAAGCTGGAGACCTCCAGCTG
4 A="seq_00005_peak"=ACKCTTACAATGTCGGAGTGTGGGAAAGATTCTGGACAG
5 A="seq_00007_peak"=TTTCGACCATTCATCTTACATTGTTGAAGGTTCTGGCGAGT
6 A="seq_00009_peak"=GCCGTAGTCAGCATCTGCAGTATTACACTGGAGAGA
7 A="seq_00011_peak"=ACTACGCTCATCAGGCATCTGAACTTCACACTGGGAGA
8 A="seq_00013_peak"=CAGTGTGGAAAGCCTTCATGCTAGTCACTGGACAG
9 A="seq_00015_peak"=GTGATCGCTCAGGTCTTACAGCACAGGAGAACACTCAT
10 A="seq_00017_peak"=CAGTGTGGACCTCTCTTCAACAGCACGGCAGA
11 A="seq_00019_peak"=TGAGACAACTGAGGAGCTTCCACATTCTAGCAGCACAT
12 A="seq_00021_peak"=GCCAGGATTAATTCGGAAAAGTTCTTGCACATGGTTCAG
13 A="seq_00023_peak"=TAATGACTGTGGGAAGACTTTAGTCACATTACAGACTT
14 A="seq_00025_peak"=CCCTATCTGCTGTGCTGAATGTGATAAGGCTTCAGGCG
15 A="seq_00027_peak"=CTACATCACAGAAAGTCACTACTGGAGAGGACCTCC
16 A="seq_00029_peak"=ATGCCGAACAAAATTTCAGGTTGGGAAAGCCTTCC
17 A="seq_00031_peak"=AGACAGGAGCTGCTCTGAAGGCTTGGACACCGATT
18 A="seq_00033_peak"=TAATGTCGGACCGACCAAGGCTTGGACACCGATT
19 A="seq_00035_peak"=AAACCTCTATGATGTTGAAAGATGTTGGAGGCTTCAGT
20 A="seq_00037_peak"=TAGGGCTCTCTCCAGATGAGTCTTGTGTCAGT
21 A="seq_00039_peak"=CCAGAAGATTCTACAGGAGGACGCCATTAGTGTAA
22 A="seq_00041_peak"=AACATCAGAGAATTCTACATGGAGGAAACCTATAAT
23 A="seq_00043_peak"=AGACCCCTCAAGATGAAAGGATGGAGAACCTCCAGG
24 A="seq_00045_peak"=CAACATCTGGAGATTCACTAGTGGAAAAGCCCTTAA
25 A="seq_00047_peak"=GTGCTCAGACCTTACTAACATCAGAGAAATTCACTG
26 A="seq_00049_peak"=GAGACACACACAGGGAGGAAAGCTGTGATGTCAG
27 A="seq_00051_peak"=GCATTCTACAGTACCTTGTGCTTAAAGGAGCATGAGAG
28 A="seq_00053_peak"=TTGTCACATTCTAGTCATTCTACATGTTCACTG
29 A="seq_00055_peak"=TGTGGCAAAAGCTTCAACAGGCGCTTACACTCTAG
30 A="seq_00057_peak"=GTACTGAGTGGCGGCAAAACTTCAGCAGGAGGCAACT

A→seq_0001_peak→GCTGGGAAGCCTCGGGTGTCCCAGAAC
A→seq_0003_peak→GAGTTCTTGATGTGCAACAAGCTGAGA
A→seq_0005_peak→ACCCCTACAAATGTCCCGAGTGTGGGAAG
A→seq_0007_peak→TTTGCCACATTCCCTCACATTGTAGGGT
A→seq_0009_peak→GCCTGAGTCAGCATCAGCTGATTCACACT
A→seq_0011_peak→ACTCAGCTCATCAGGCATCTGCAAATTCA
A→seq_0013_peak→CAGTGACTGTGGAAAAGCCTTCATCGATC
A→seq_0015_peak→GTGATCGGTCAAGGTCTTATCCAGCACCA
A→seq_0017_peak→CAGTGACCGTTCCCTCTCAACCAGCAC
A→seq_0019_peak→TGAGACAACTGAAAGCTTGCCACATTCA
A→seq_0021_peak→GCCAGCGATATTCCGTAAAAGTTCTGA
A→seq_0023_peak→TAATGACTGTGGGAAGACTTTAGTCAC

How to find motifs? 🤔

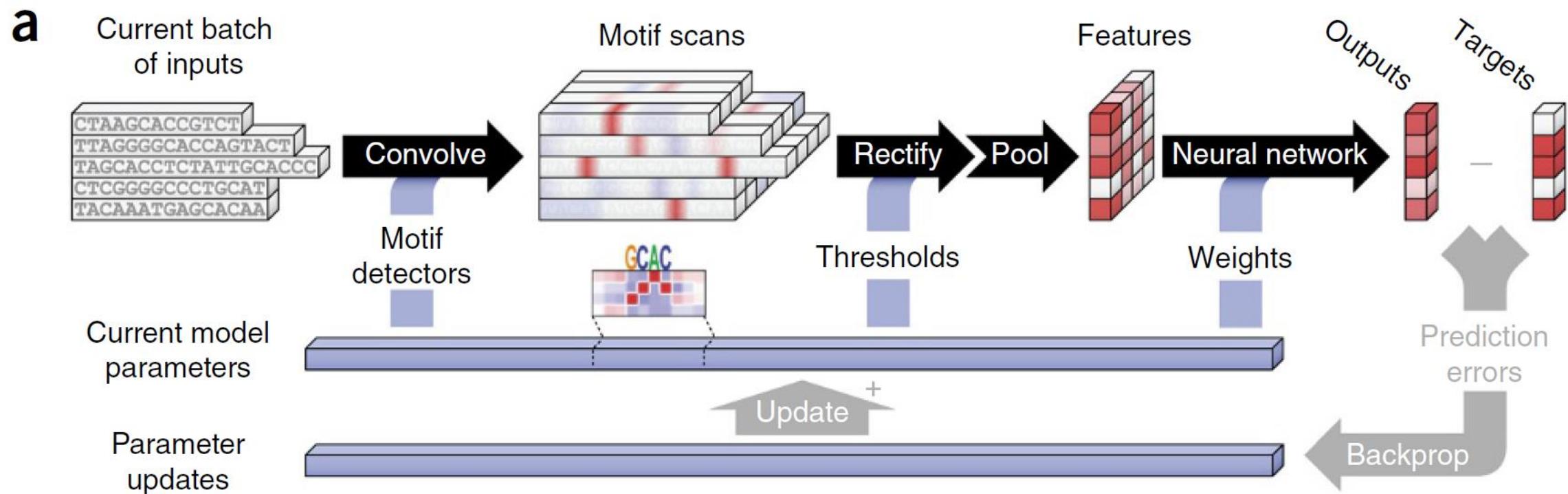
1. High-throughput experiments



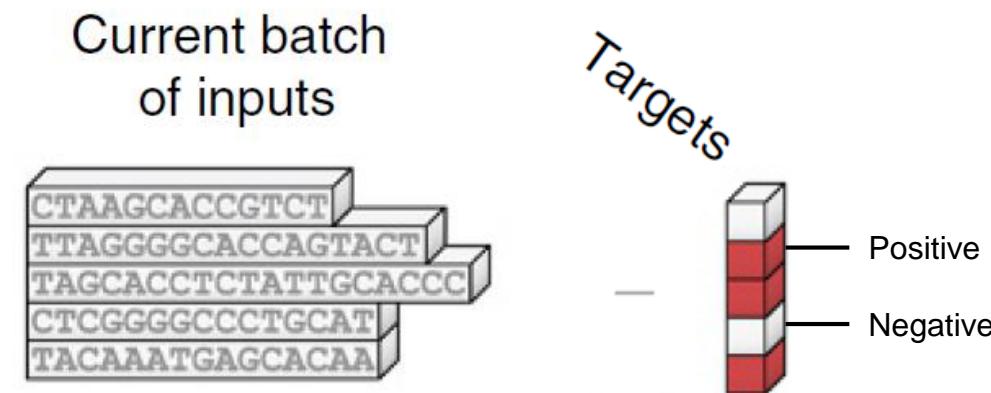
Automatic model training

- 표적 단백질에 결합한 DNA/RNA 서열들을 학습시킨다???

Sol. Deep Convolutional Neural Network & Binary Classification



Sol. Deep Convolutional Neural Network & Binary Classification



- ① 실험 결과, 발생한 서열들은 Positive class, 그렇지 못한 서열들 (없으면 무작위 서열)은 Negative class로 지정

Sol. Deep Convolutional Neural Network & Binary Classification



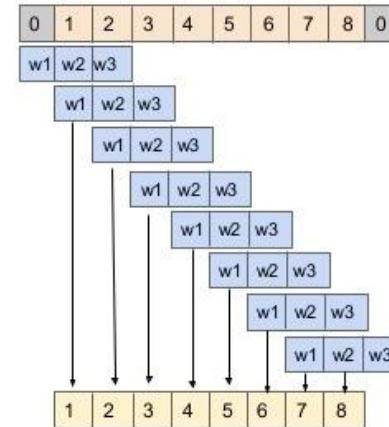
- ① 실험 결과, 발생한 서열들은 Positive class, 그렇지 못한 서열들 (없으면 무작위 서열)은 Negative class로 지정
- ② 만약 어떤 방법(혹은 모델)이 이 두가지 class들을 분류할 수 있다면 해당 모델은 입력 서열에서 유의미한 패턴 정보를 파악한 것

Sol. Deep Convolutional Neural Network & Binary Classification



1D Convolutions

When we add zero padding, we normally do so on both sides of the sequence (as in image padding)

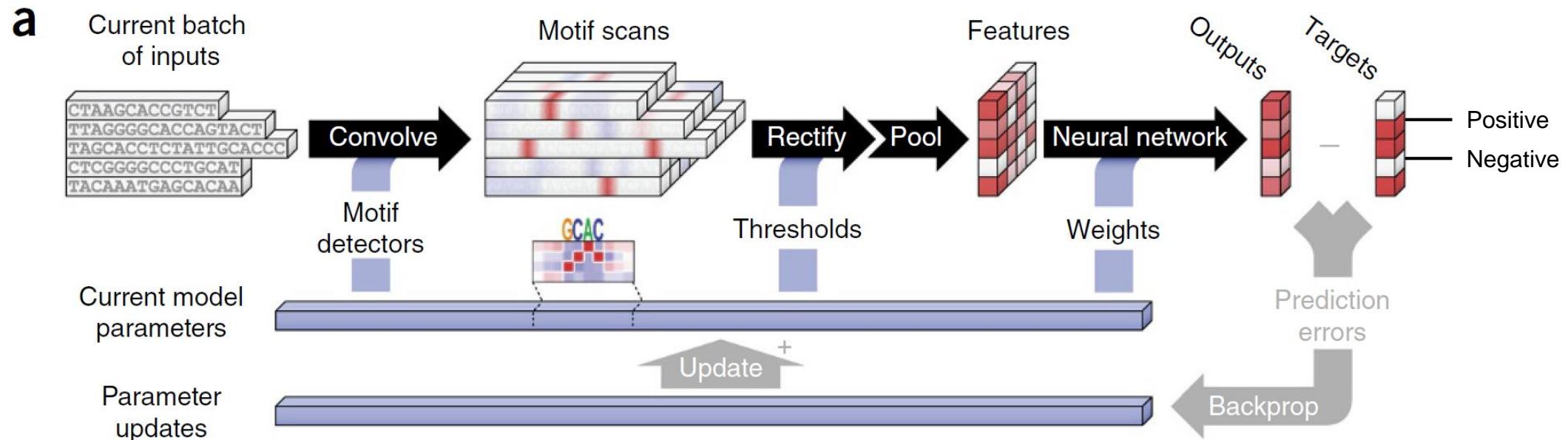


The length result of the convolution is well known to be:
 $\text{seqlength} - \text{kwidth} + 1 = 10 - 3 + 1 = 8$

So the output matrix will be (8, 100)
because we had padding

- ① 실험 결과, 발생한 서열들은 Positive class, 그렇지 못한 서열들 (없으면 무작위 서열)은 Negative class로 지정
- ② 만약 어떤 방법(혹은 모델)이 이 두가지 class들을 분류할 수 있다면 해당 모델은 입력 서열에서 유의미한 패턴 정보를 파악한 것
- ③ 머신러닝/딥러닝 방법론으로 이를 접근한다면, 어느 모델 구조 (architecture)가 가장 적합한가? → 1D-CNN

Sol. Deep Convolutional Neural Network & Binary Classification



모델은 입력 서열에서 유의미한 패턴 정보를 파악한 것

③ 머신러닝/딥러닝 방법론으로 이를 접근한다면, 어느 모델 구조 (architecture)가 가장 적합한가? → 1D-CNN

④ 결론: 1D-CNN 모델과 실험 데이터를 바탕으로 이진분류모델을 학습시키면 CNN 구조의 filter 정보로 motif를 추론할 수 있다

<http://tools.genes.toronto.edu/deepbind/>

DeepBind

Welcome to the searchable database of DeepBind models!

Example searches:

- [gata selex type:tf](#) - GATA TFs trained on SELEX data
- ["mus musculus" type:rbp](#) - mouse RBPs

For more help, see frequently asked questions.

[Copy IDs](#) [« prev](#) [next »](#)

Protein	Type	Species	Family	Experiment	Logos	ID
A1CF	RBP	Gallus gallus	RRM	RNAcompete		D00288.001
A1CF	RBP	Homo sapiens	RRM	RNAcompete		D00084.001
A2BP1	RBP	Drosophila melanogaster	RRM	RNAcompete		D00175.001
Ahctf1	TF	Mus musculus	AT hook	PBM		 D00053.001

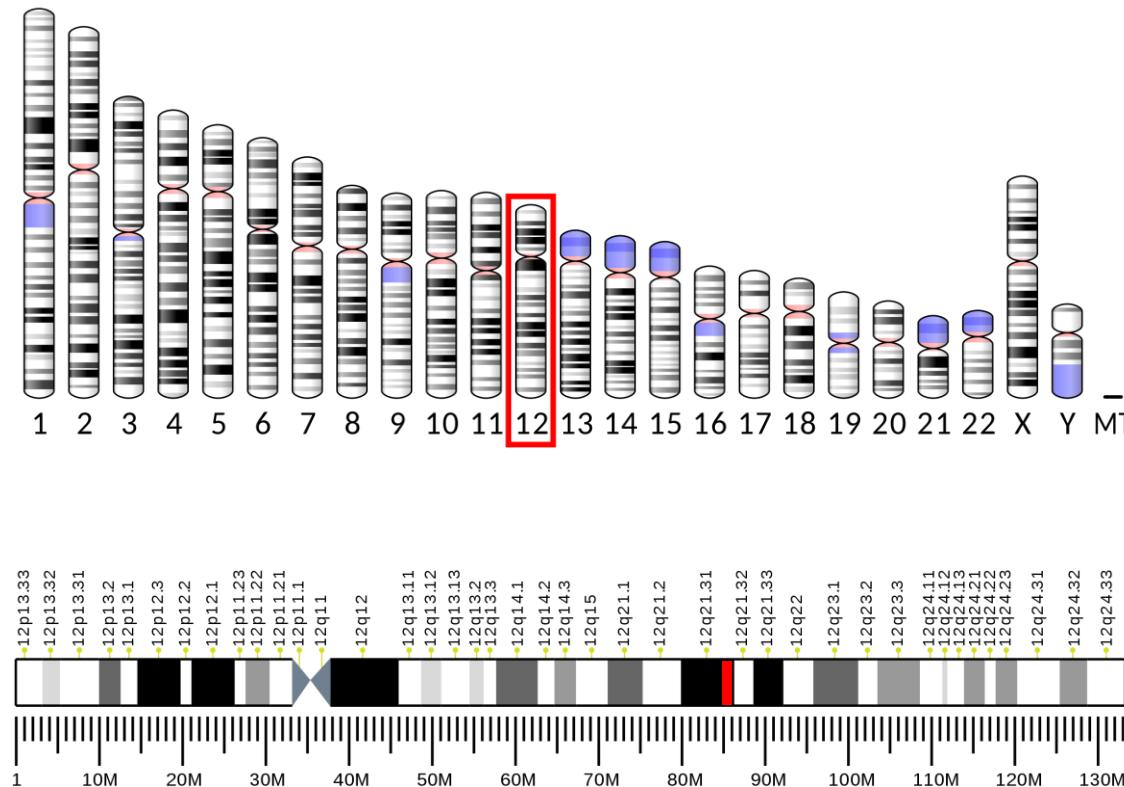
2. 실습 및 예시 코드 - ML/DL for sequencing data

실습 목표 확인, 데이터 준비, etc ...

실습 과제 목표

1. 이진 분류 모델 학습을 위한 sequencing 데이터 준비하기
2. Sequencing 데이터를 Deep Learning 모델 (ex. 1D-CNN)에 사용하기
3. 이진 분류 모델의 학습에 대한 올바른 평가 수행하기
4. Motif 분석 – Position Frequency Matrix

Download raw data



ALX1

ALX homeobox protein 1 is a [protein](#) that in humans is encoded by the [ALX1 gene](#).

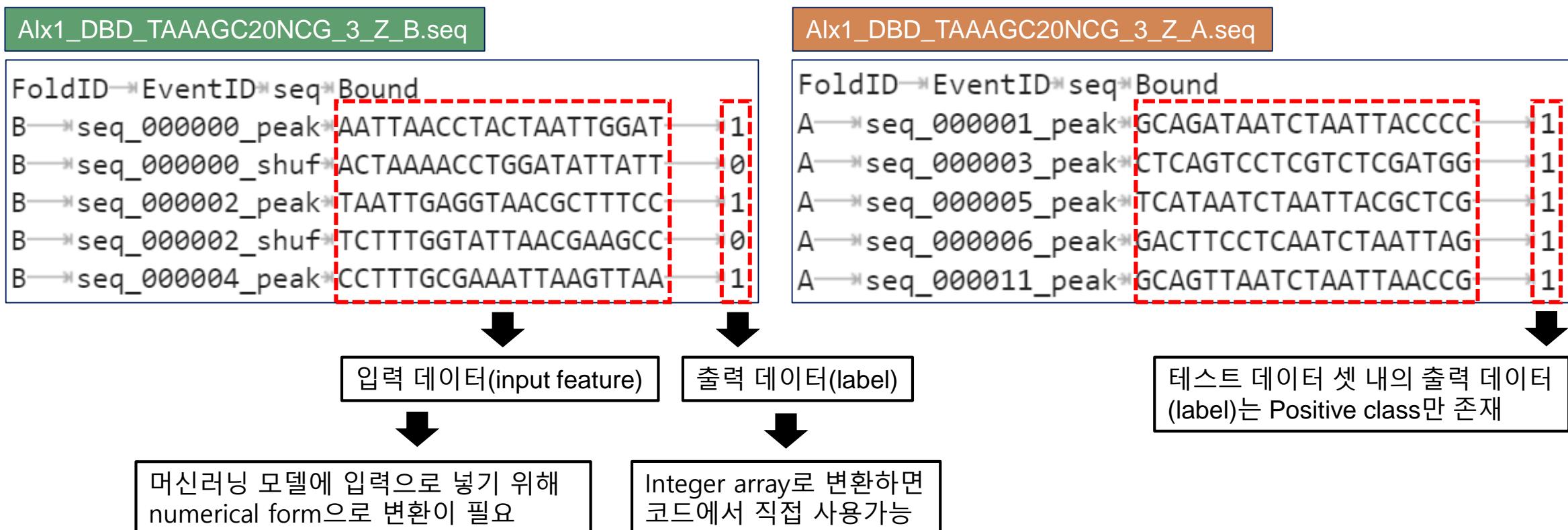
- ID: [D00289.002](#)
- Type: [TF](#)
- Species: [Mus musculus](#)
- Family: [Homeodomain](#)
- Experiment: [SELEX](#)
- Model: [deepbind 0.1](#)
- Cite: [PMID 23332764](#)
- Wiki: <https://en.wikipedia.org/wiki/ALX1>
- DeepBind info: ([link](#))

Sequence Dataset for Binary Classifier (Source: DeepBind)

- Download type-A ([link](#))
- Download type-B ([link](#))

About the raw data sequences

- Alx1_DBDB_TAAAGC20NCG_3_Z_B.seq (학습 및 평가용 데이터)
- Alx1_DBDB_TAAAGC20NCG_3_Z_A.seq (테스트용 데이터)



Load the downloaded raw data

- Alx1_DBDB_TAAAGC20NCG_3_Z_B.seq (학습 및 평가용 데이터)
- Alx1_DBDB_TAAAGC20NCG_3_Z_A.seq (테스트용 데이터)

```

from collections import defaultdict
import numpy as np

def raw_seq_parser(path):
    f = open(path, "r")
    seqs = []
    seq_lens = []
    labels = []
    flines = f.readlines()
    cols = None
    for i, line in enumerate(flines):
        if not i:
            cols = line.replace("\n", "").split("\t")
        else:
            fold_id, event_id, seq, bound = line.replace("\n", "").split("\t")
            seqs.append(seq)
            labels.append(int(bound))
            seq_lens.append(len(seq))

    num_pos = sum(labels)
    num_neg = len(labels) - num_pos
    print("Load {} finish ({}) lines, #positive: {}, #negative: {}".format(path, len(labels), num_pos, num_neg))
    print("- sequence sizes : {}".format(set(seq_lens)))

    return seqs, labels

# path of raw data
TRAIN_RAW = "Alx1_DBDB_TAAAGC20NCG_3_Z_B.seq"
TEST_RAW = "Alx1_DBDB_TAAAGC20NCG_3_Z_A.seq"

train_seqs, train_labels = raw_seq_parser(TRAIN_RAW)
test_seqs, test_labels = raw_seq_parser(TEST_RAW)

```

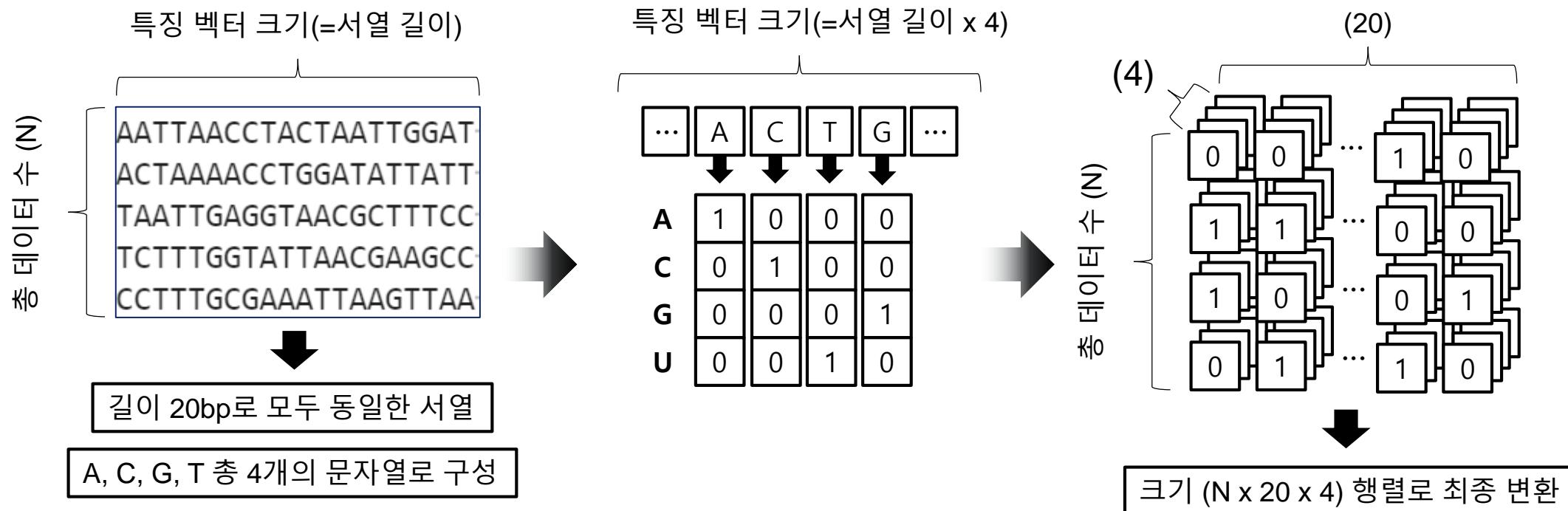
```

Load Alx1_DBDB_TAAAGC20NCG_3_Z_B.seq finish (255508 lines, #positive: 127754, #negative: 127754)
- sequence sizes : {20}
Load Alx1_DBDB_TAAAGC20NCG_3_Z_A.seq finish (128012 lines, #positive: 128012, #negative: 0)
- sequence sizes : {20}

```

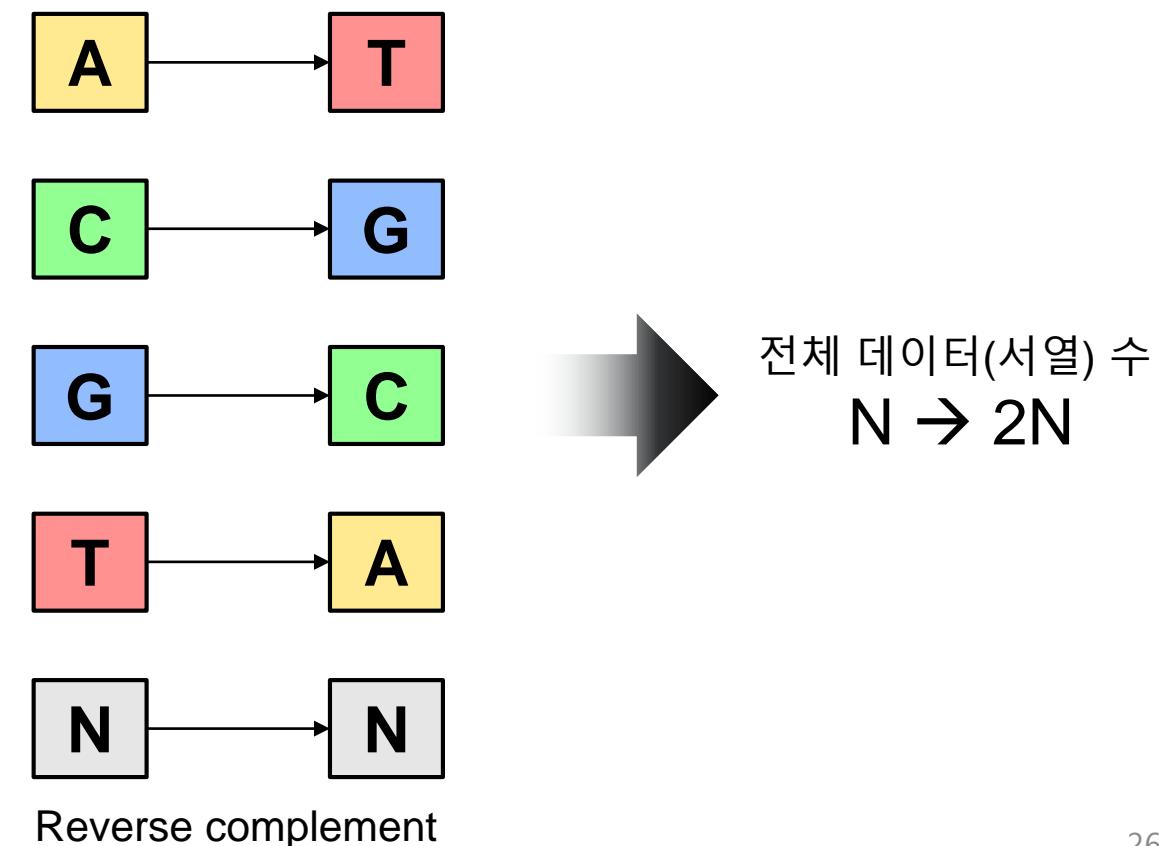
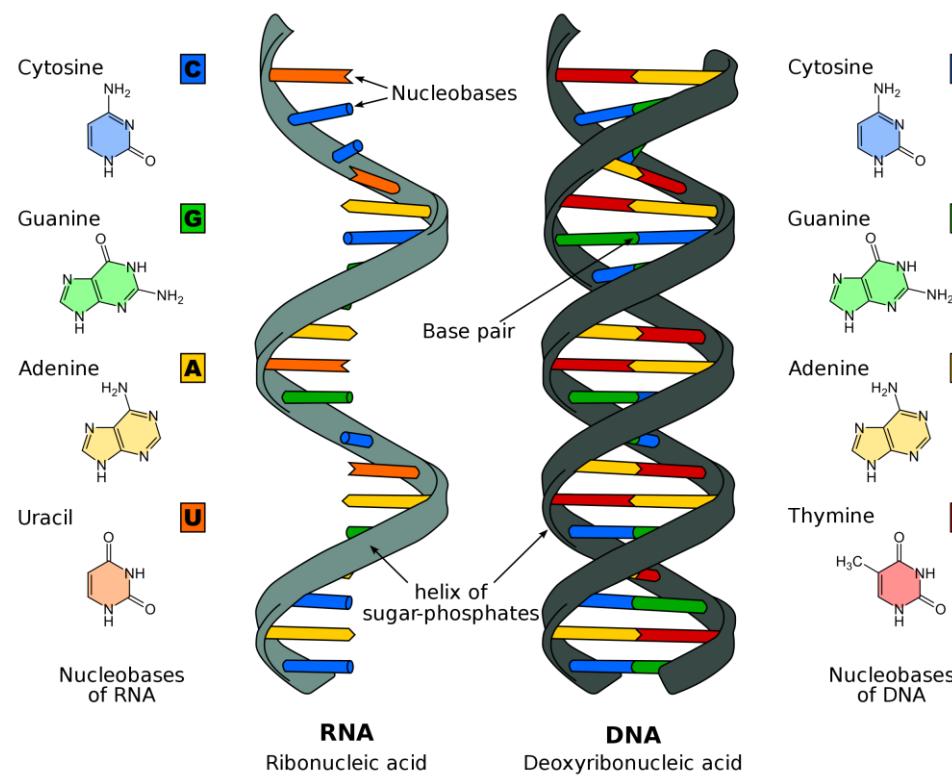
Sequence Encoding (one-hot encoding)

- 별도의 변환 없이, 데이터 그 자체를 표현할 수 있는 간단한 방법
- 보통 classification 데이터의 label 변환에 자주 사용된다 (ex. 4 classes, class-4 → [0,0,0,1])
- 생물정보학 분야에서 염기서열을 표현할 때 많이 사용되나, 길이가 다른 서열들을 표현하기엔 어려움



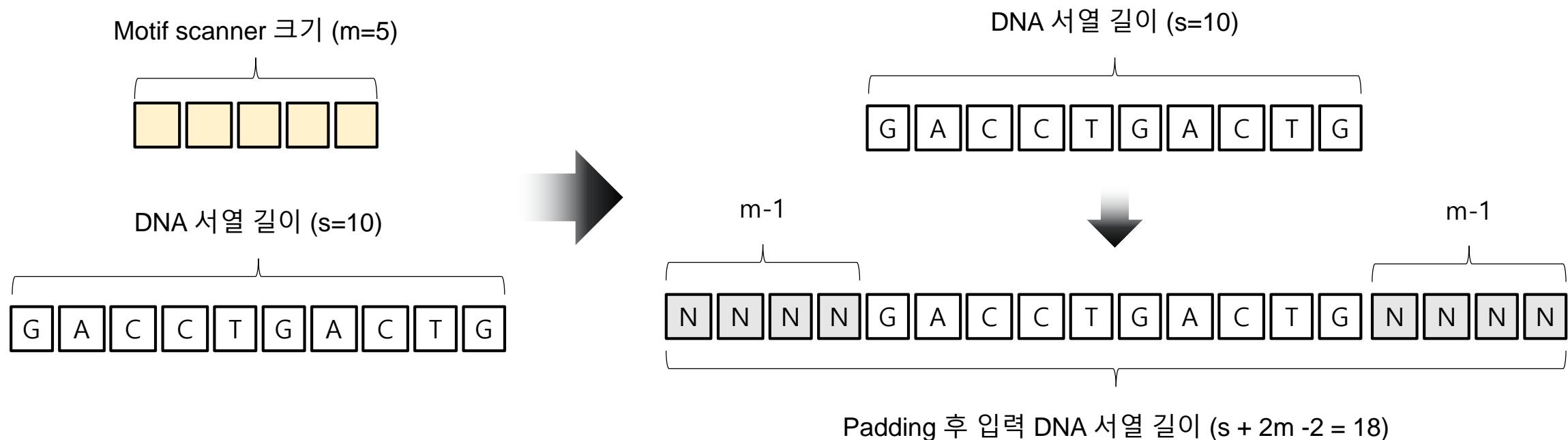
Reverse Complement (~ Bioinformatics)

- 모델에 사용하는 데이터가 DNA 서열인 경우
- DNA 서열은 single-strand 인 RNA와 달리 double-strand
- 어떤 strand 가 표적 단백질과 결합을 한 것인지 모호하므로 둘다 학습데이터로 사용
- DNA 서열의 base pair는 지정되어 있다



Padding for motif scanner (~ DeepBind)

- DeepBind 연구에서 제안한 방법
- 1D-CNN 모델의 필터를 motif scanner 처럼 사용하기 위해 입력 서열의 양 끝에 padding 을 추가
- motif scanner 의 크기 m / 서열의 길이 s
 - padding 후 서열의 길이 $(s + 2m - 2)$ / one-hot encoding 후 서열 벡터의 크기 $(s + 2m - 2) * 4$



Sequence Encoding (one-hot encoding + reverse complement + padding)

```
# ref [complement,reverse_complement] from [https://github.com/MedChaabane/DeepBind-with-PyTorch/]
def complement(seq):
    complement = {'A': 'T', 'C': 'G', 'G': 'C', 'T': 'A', 'N': 'N'}
    complseq = [complement[base] for base in seq]
    return complseq

def reverse_complement(seq):
    seq = list(seq)
    seq.reverse()
    return ''.join(complement(seq))

def onehot_encoder_with_padding_and_reverse_complement(seqs, labels, letters="ACGT", motif_len=10):
    print("Original sequence length {}".format(len(seqs[0])))
    enc = defaultdict(lambda: np.array([1/len(letters)]*len(letters))) # default. [0.25,0.25,0.25,0.25]
    for i,l in enumerate(letters):
        f      = np.zeros(len(letters))
        f[i]   = 1
        enc[l] = f

    # reverse-complement (note that it have to apply for DNA seqs)
    _seqs = []
    _labels = []
    for seq, label in zip(seqs, labels):
        _seqs.append(seq)
        _labels.append([label])
        _seqs.append(reverse_complement(seq))
        _labels.append([label])

    seqs = _seqs
    labels = _labels
```

Source code for the reverse complement

Sequence Encoding (one-hot encoding + reverse complement + padding)

```

# one-hot encoding
onehot_features = []
for seq in seqs:
    # Append padding letter 'N' according to the <length of motif-1>
    seq = 'N' * (motif_len-1) + seq + 'N' * (motif_len-1)
    feature = []
    for c in seq.upper():
        feature.append(enc[c])
    onehot_features.append(feature)

onehot_features = np.array(onehot_features)
print("One-hot encoding finished, with feature shape {}".format(onehot_features.shape))
return onehot_features, labels

LEN_MOTIF      = 10
X, y           = onehot_encoder_with_padding_and_reverse_complement(train_seqs, y, motif_len=LEN_MOTIF)
test_x, test_y = onehot_encoder_with_padding_and_reverse_complement(test_seqs, test_y, motif_len=LEN_MOTIF)

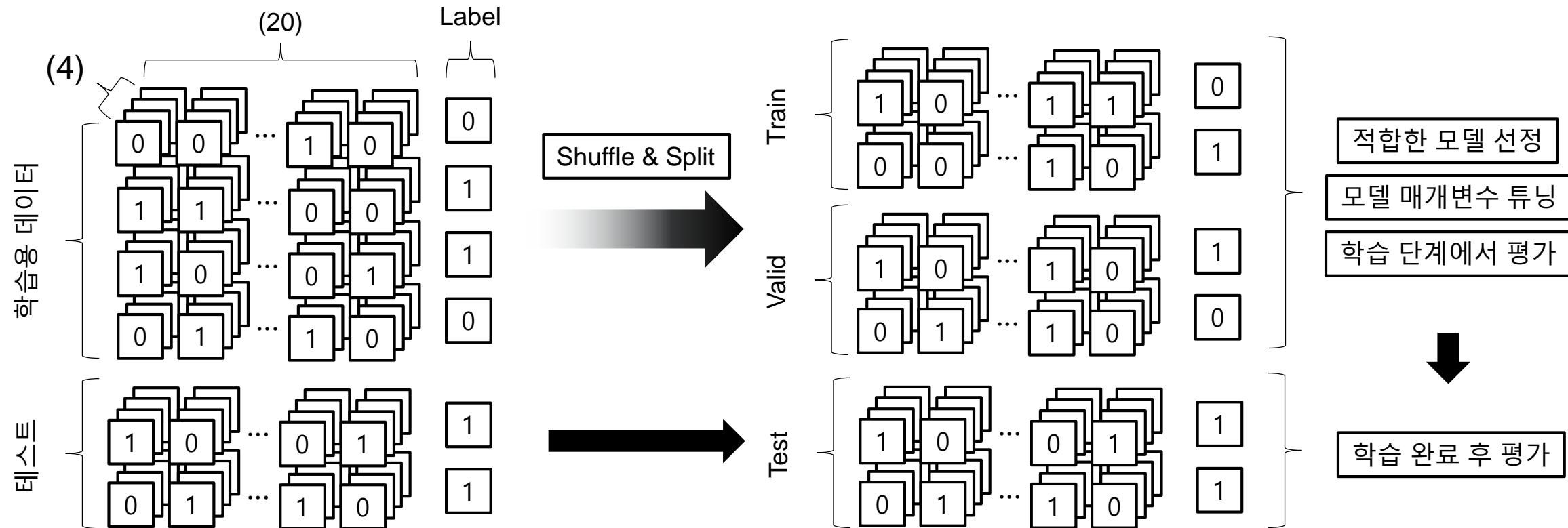
```

Original sequence length 20
 One-hot encoding finished, with feature shape (511016, 38, 4)
 Original sequence length 20
 One-hot encoding finished, with feature shape (256024, 38, 4)

Source code for the padding & one-hot encoding

Split Training & Validation datasets

- 테스트 데이터 셋은 모델 학습 후 최종 평가 시 사용
- 학습 데이터를 학습용(training) 및 평가용(validation)으로 분리하여 overfitting 방지 및 hyper-parameter tuning에 사용
- Scikit-learn 패키지의 train_test_split ([ref-link](#))으로 쉽게 코딩 가능



Split Training & Validation datasets

- 테스트 데이터 셋은 모델 학습 후 최종 평가 시 사용
- 학습 데이터를 학습용(training) 및 평가용(validation)으로 분리하여 overfitting 방지 및 hyper-parameter tuning에 사용
- Scikit-learn 패키지의 train_test_split ([ref-link](#))으로 쉽게 코딩 가능

```
from sklearn.model_selection import train_test_split

X, y = np.array(X), np.array(y) # convert python-list to numpy-array

train_x, valid_x, train_y, valid_y = train_test_split(X, y,
                                                    test_size=0.33, # ratio of size of train-test set splitting
                                                    random_state=42) # random-seed

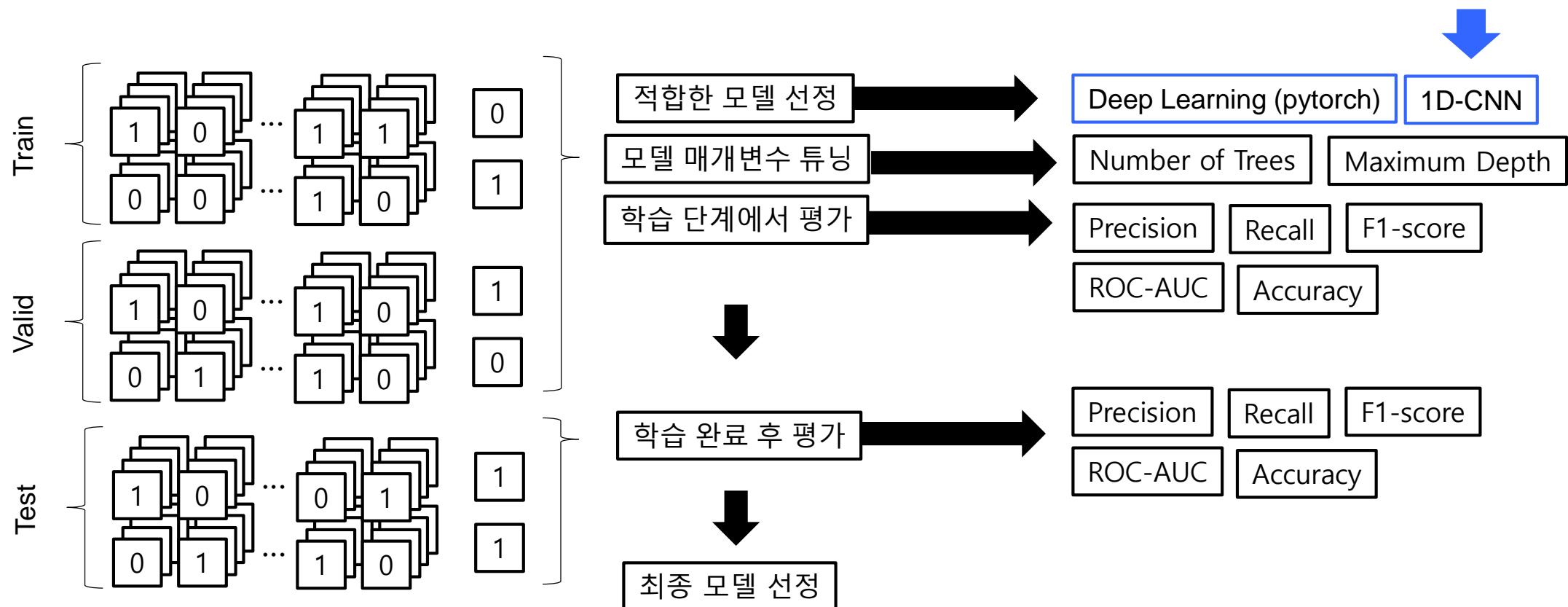
print("Train set | x: {}, y: {}\nValid set | x: {}, y: {}".format(train_x.shape,
                                                               train_y.shape,
                                                               valid_x.shape,
                                                               valid_y.shape))
```



```
Train set | x: (171190, 20, 4), y: (171190,)
Valid set | x: (84318, 20, 4), y: (84318,)
```

Binary Classifier using the Deep Learning

- 기존 binary classifier 학습 준비 과정 및 평가 단계를 모두 숙지한 상태에서 binary classifier 모델만 변경
- Pytorch 프레임워크를 사용하여 1D-CNN 기반 모델을 구성



DeepBind (2015) – Basic Architecture

- 논문에서 제시한 모델 구조는 다음과 같다
- 다만 실제 구현 단계에서 차별성이 존재함

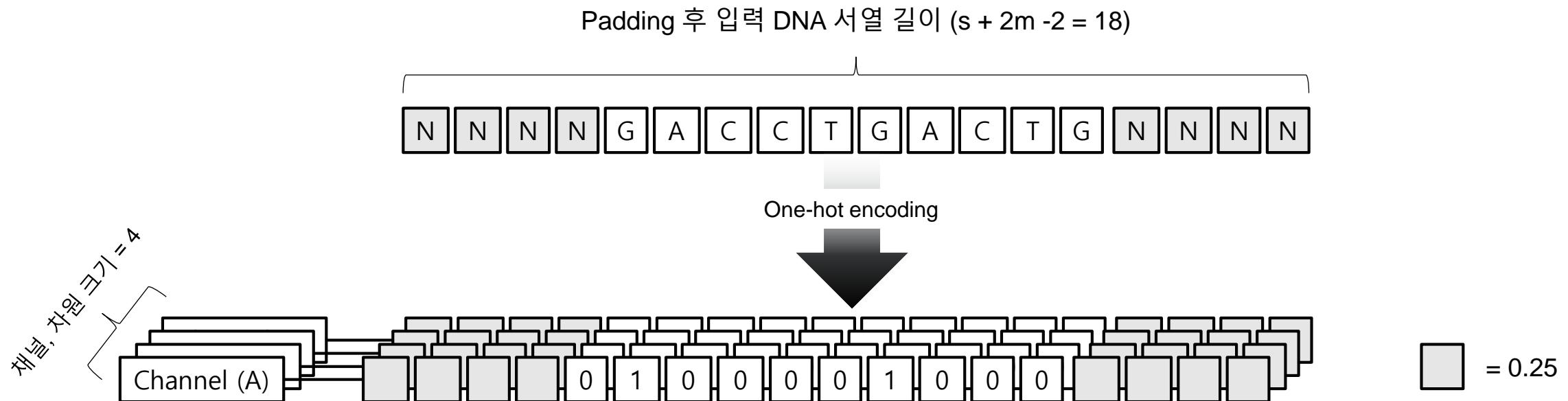
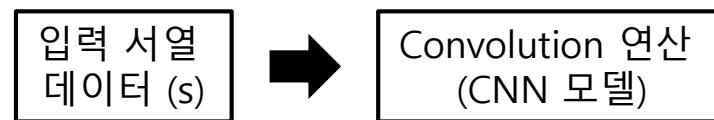
$$f(s) = \text{net}_W(\text{pool}(\text{rect}_b(\text{conv}_M(s))))$$

다음과 같이 순차적으로 구성된다



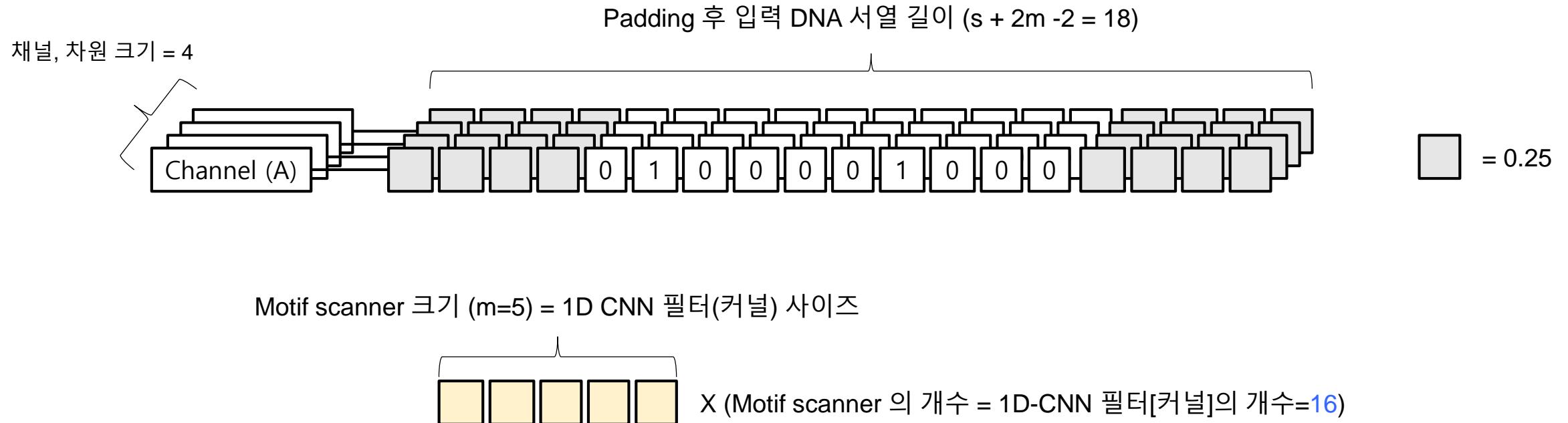
DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



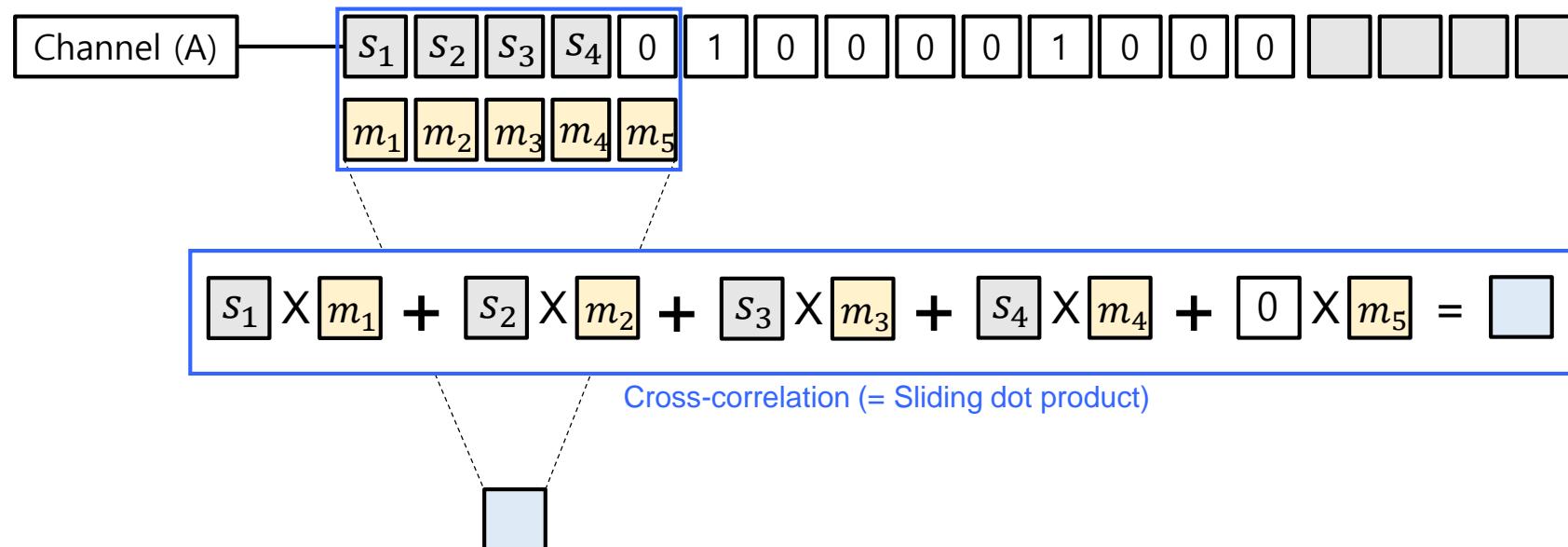
DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



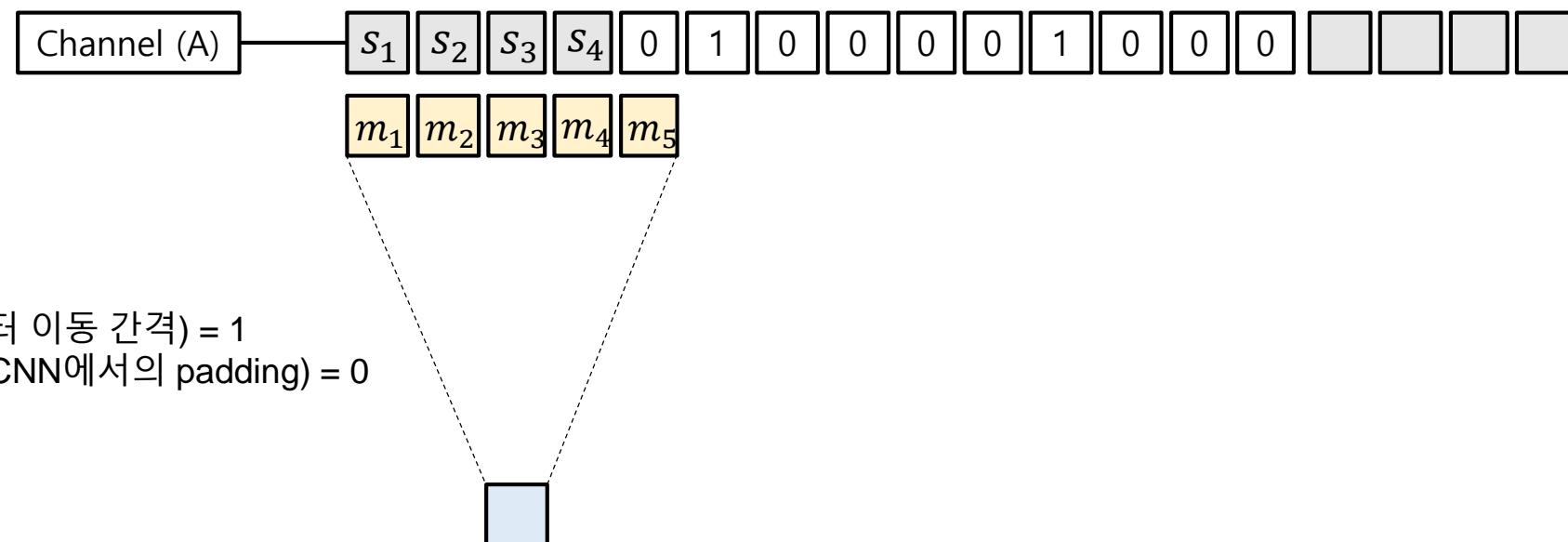
Cross-correlation (= Sliding dot product) for one channel for one motif scanner (=convolution filter)

Channel (A)



DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



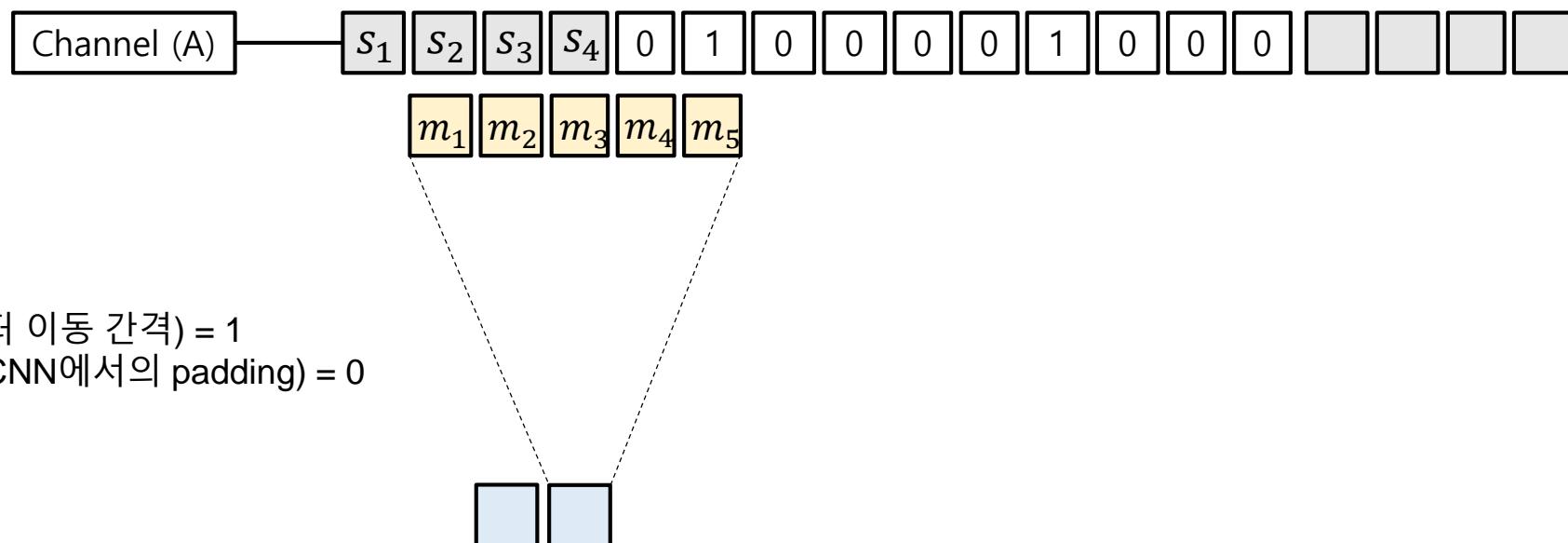
Cross-correlation (= Sliding dot product) for one channel for one motif scanner (=convolution filter)

Channel (A)



DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



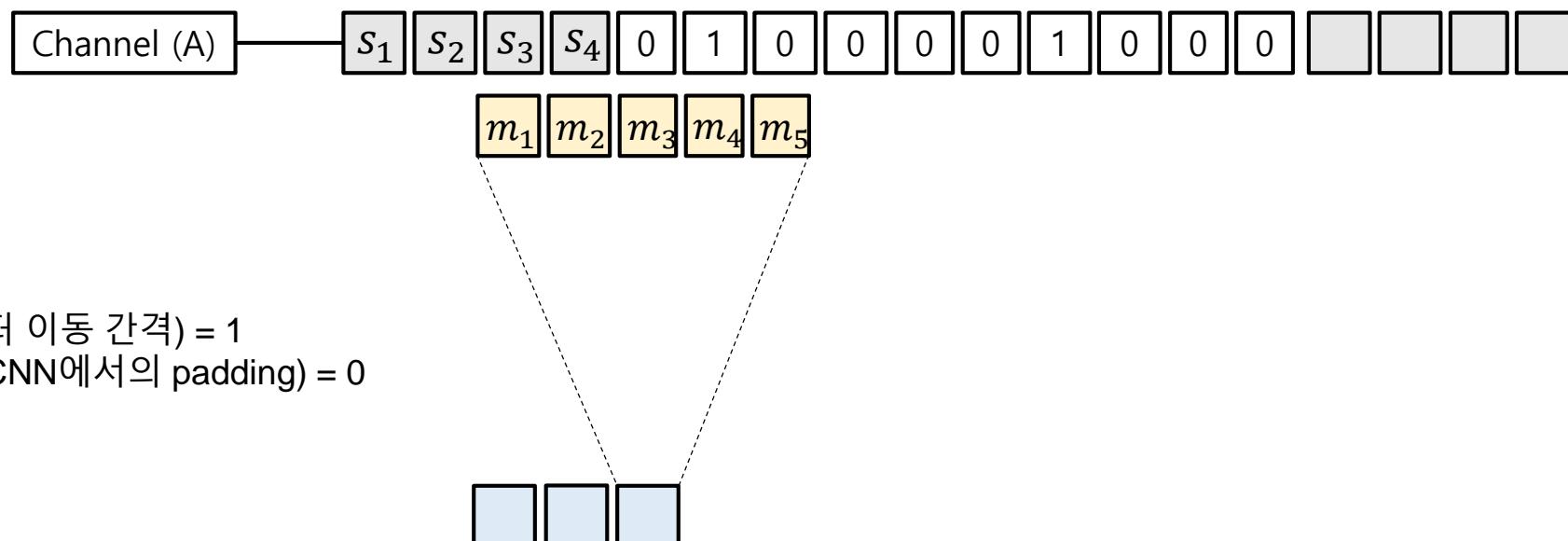
Cross-correlation (= Sliding dot product) for one channel for one motif scanner (=convolution filter)

Channel (A)



DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



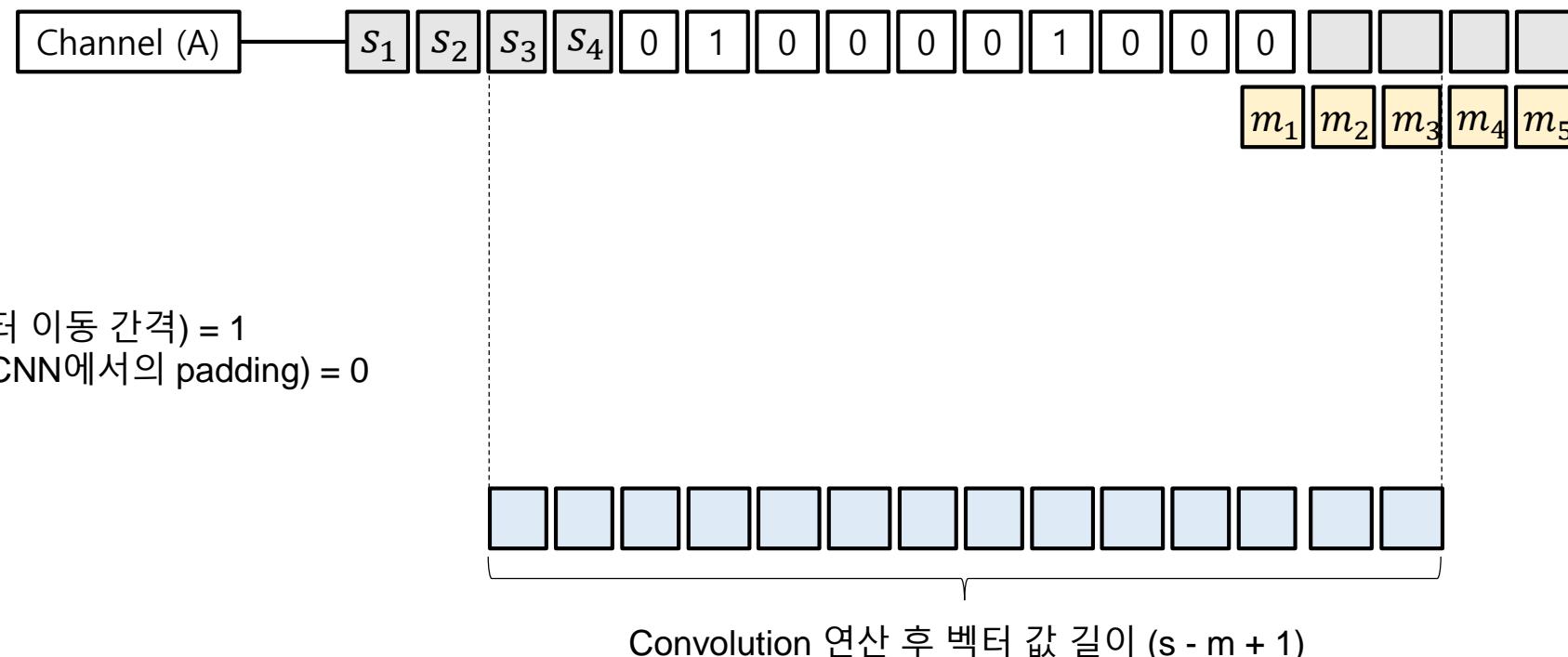
Cross-correlation (= Sliding dot product) for one channel for one motif scanner (=convolution filter)

Channel (A)



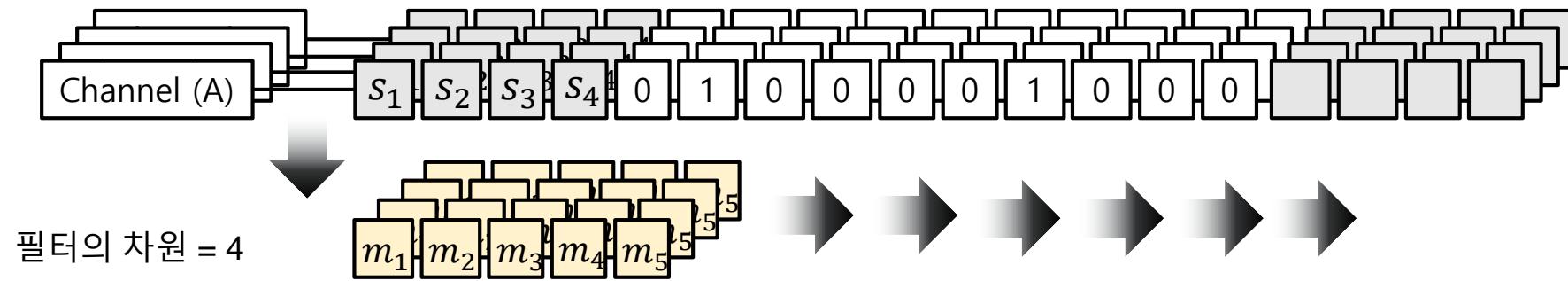
DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)

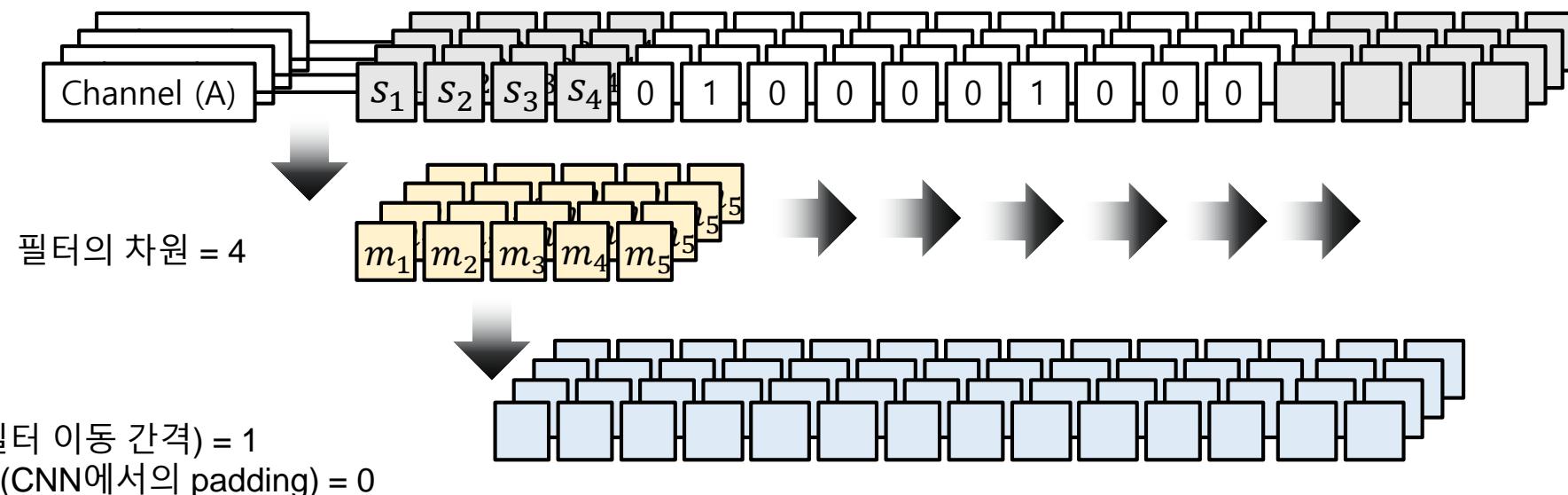


Stride (필터 이동 간격) =

Padding (CNN에서의 padding) = 0

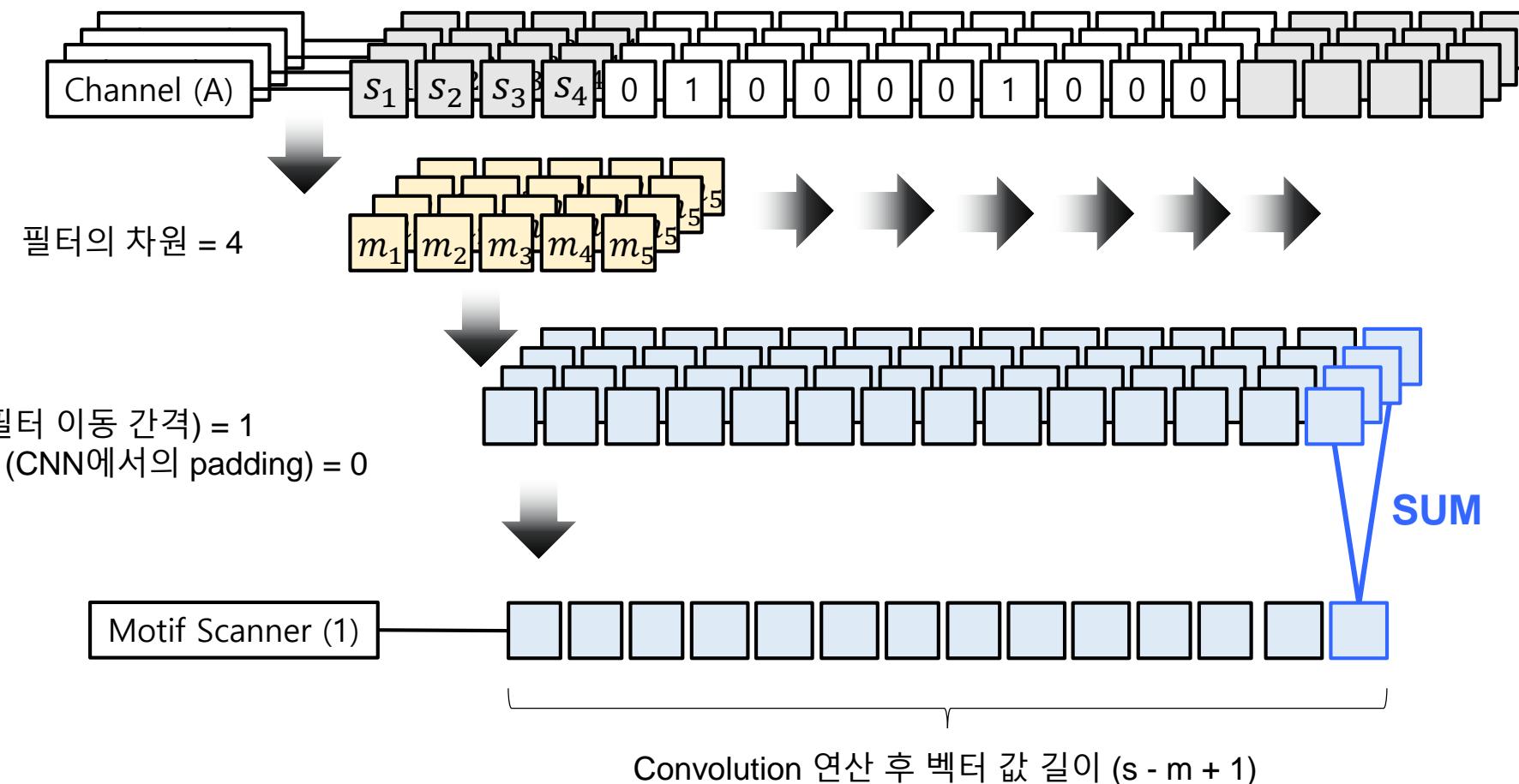
DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



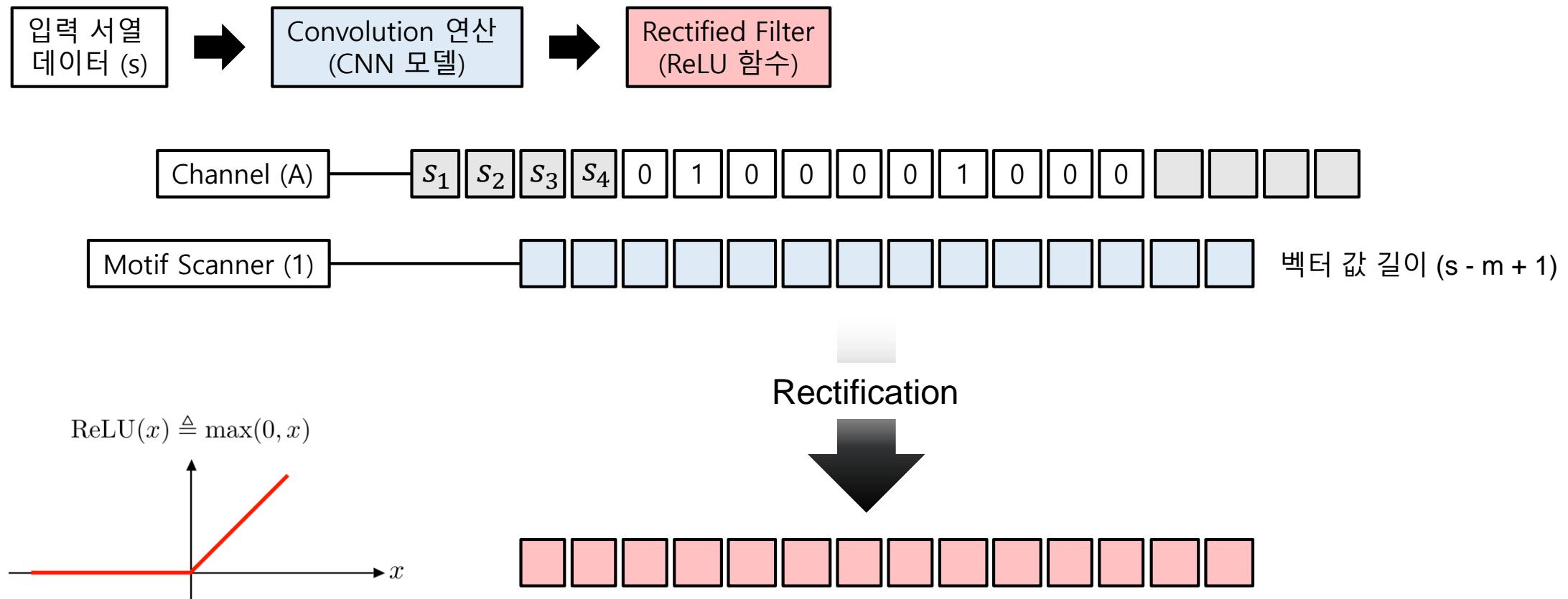
DeepBind (2015) – Convolution

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



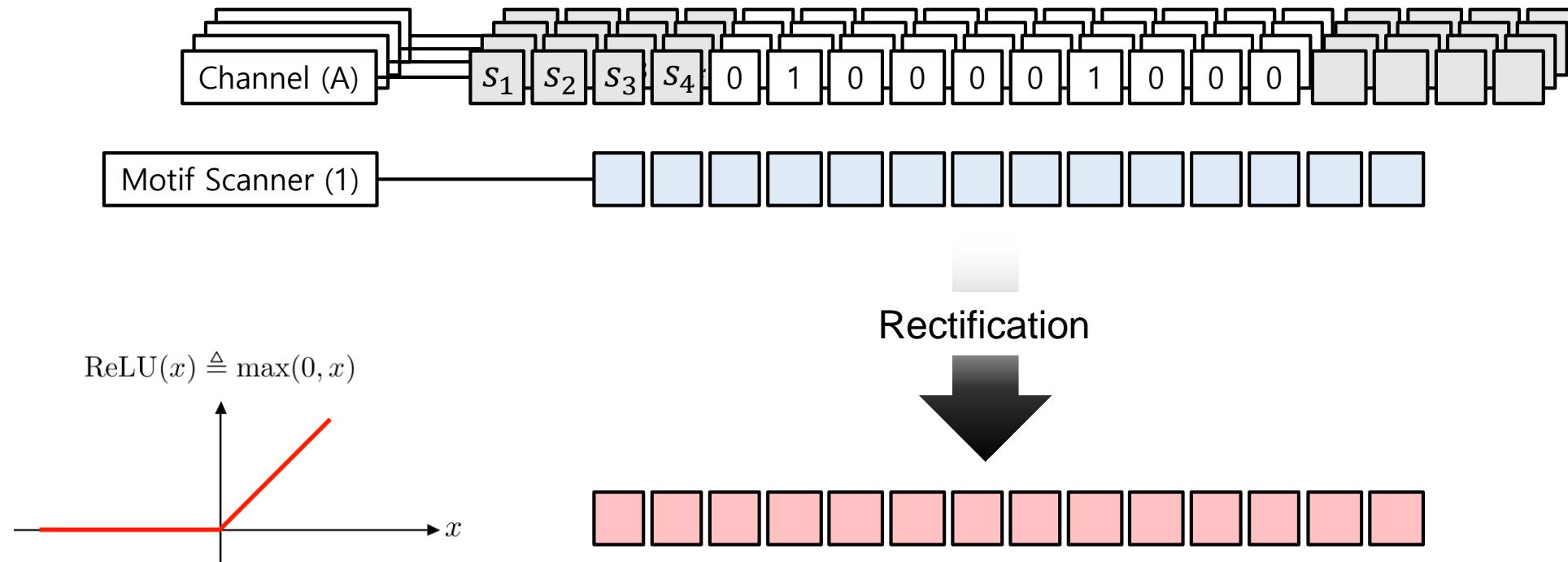
DeepBind (2015) – Rectification

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



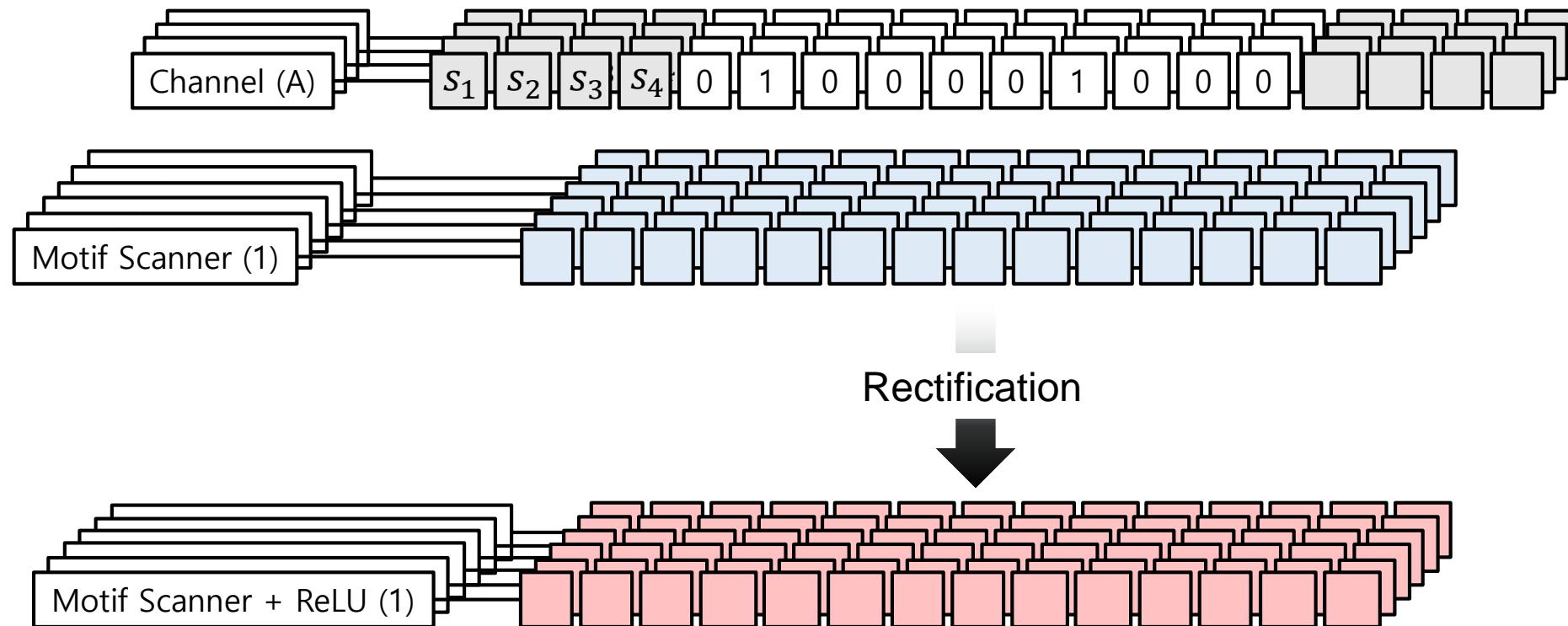
DeepBind (2015) – Rectification

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



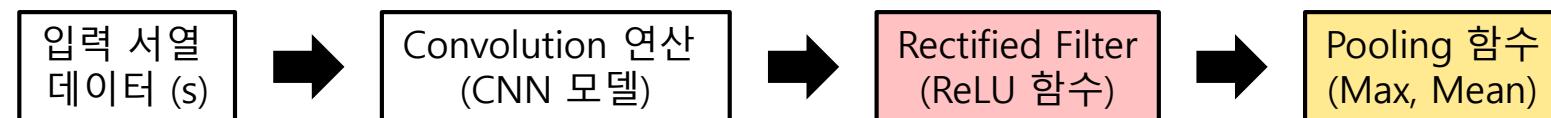
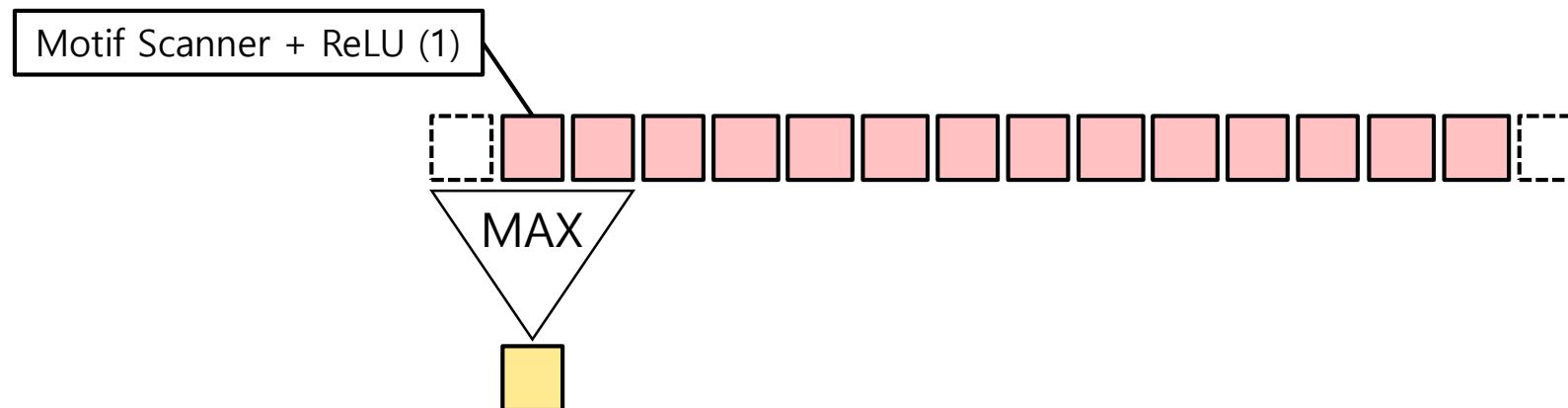
DeepBind (2015) – Rectification

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



DeepBind (2015) – Pooling (Max or Max + Avg)

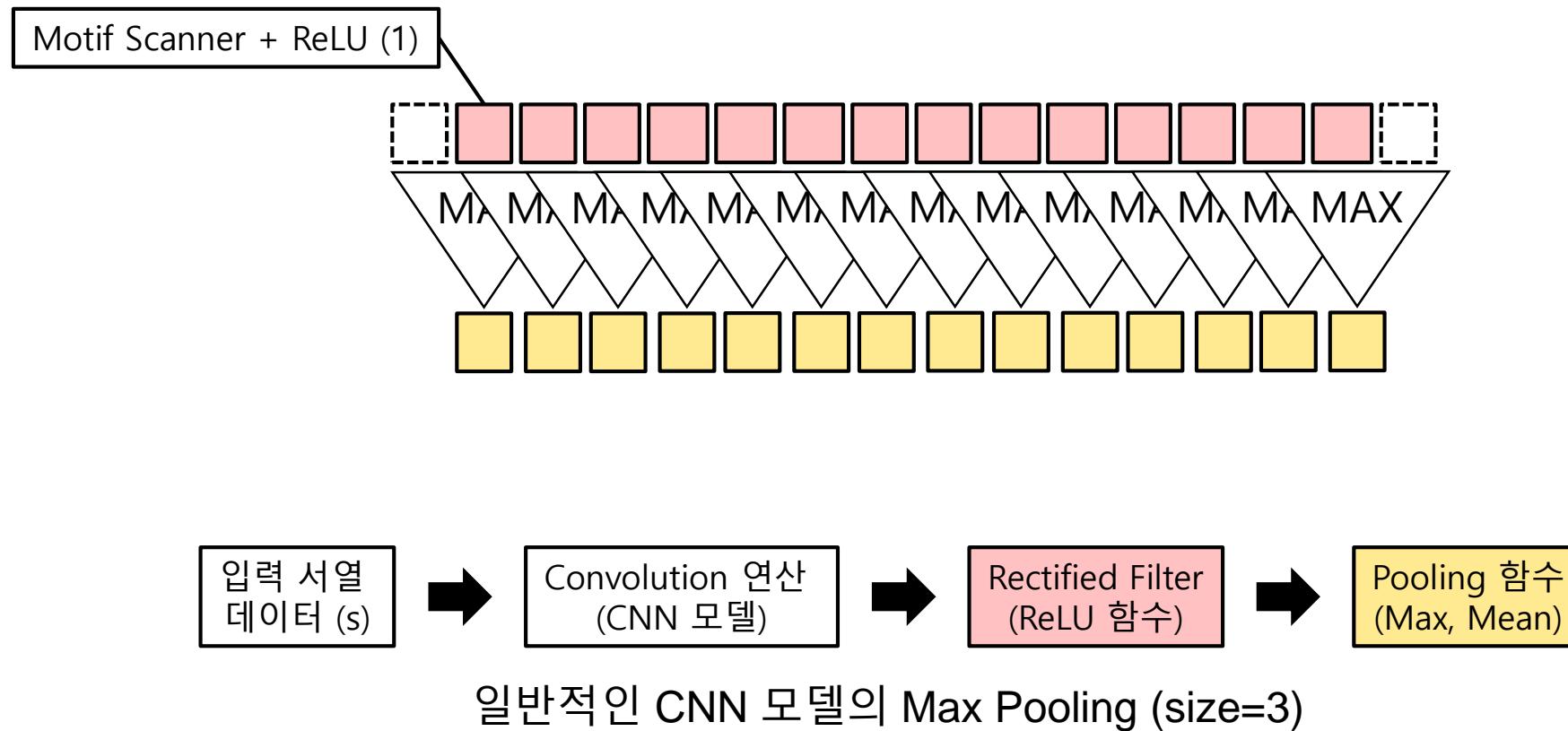
- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



일반적인 CNN 모델의 Max Pooling (size=3)

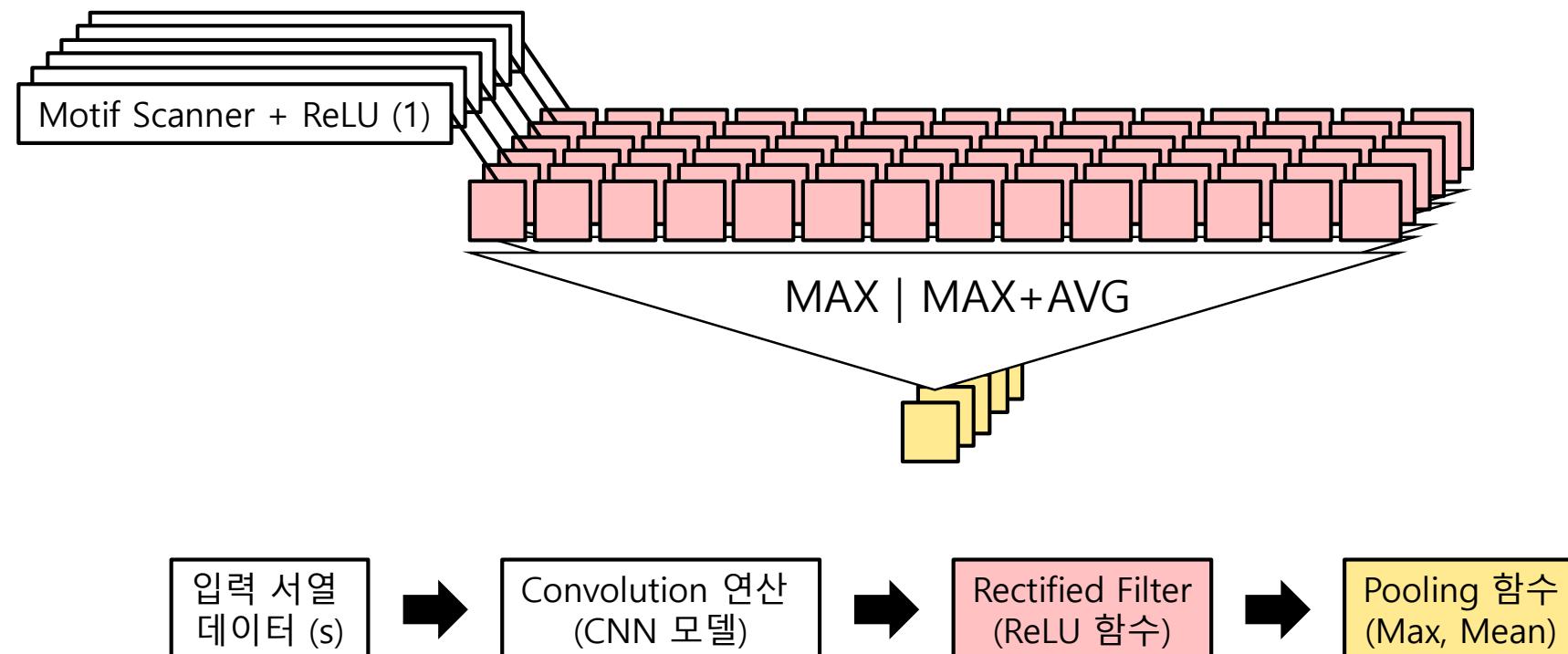
DeepBind (2015) – Pooling (Max or Max + Avg)

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



DeepBind (2015) – Pooling (Max or Max + Avg)

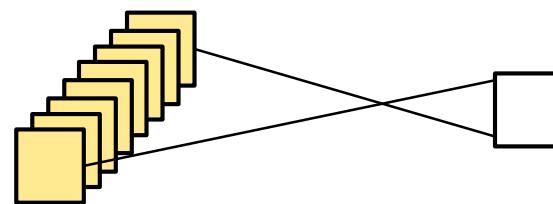
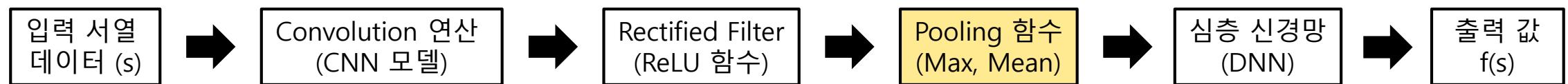
- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



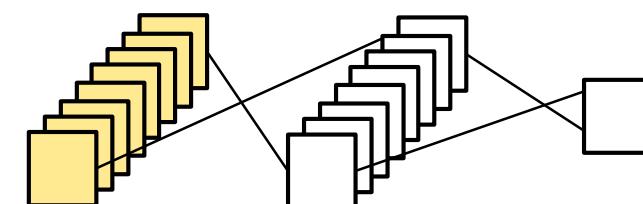
DeepBind에서 사용한 Max Pooling
각 Motif scanner 의 결과 전체에 대한 Pooling

DeepBind (2015) – Neural Network

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



(No hidden layer)

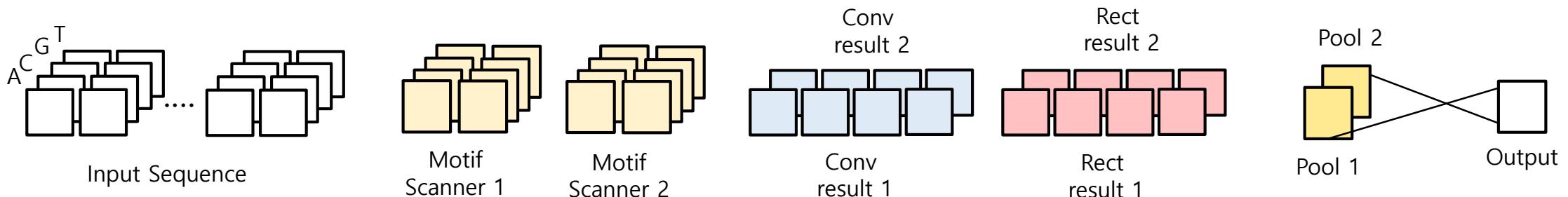


(One hidden layer with 32 units and Relu)

Motif scanner 및 Rectification 후,
각 motif 결과 값들의 pooling 결과들을 사용하여 최종 출력 (True/False)

DeepBind (2015) – Neural Network

- 해당 연구에서는 다층 구조로 쌓는 일반적인 Deep Learning 모델이 아닌 단일 계층의 CNN 모델을 사용
(단, 여러 층을 사용해도 상관은 없으나 motif scanner로 사용되는 부분은 첫번째 레이어이므로 1개 층을 추천)



Ex. Motif Scanner 가 2개인 경우 모델 구조

DeepBind (2015)

```
import torch
import torch.nn as nn
import torch.nn.functional as F
device = 'cpu'
```

모델에서 사용할 Layer 초기화 및 정의

```
class ConvNet(nn.Module):
    def __init__(self, num_motifs, len_motif, pooling_type="max"): # pool_type = [max, maxavg]
        super(ConvNet, self).__init__()
        self.l_conv = nn.Conv1d(in_channels = 4,
                             out_channels = num_motifs,
                             kernel_size = len_motif,
                             stride = 1,
                             padding = 0)

        self.pooling_type = pooling_type
        self.relu = nn.ReLU()
        if pooling_type == "max":
            self.l_nnet = nn.Linear(num_motifs,1)
        else:
            self.l_nnet = nn.Linear(num_motifs*2,1)

        print("- Motif scanners : {}".format(self.l_conv.weight.size()))
```

```
NUM_MOTIFS = 16
convnet = ConvNet(num_motifs = NUM_MOTIFS,
                  len_motif = LEN_MOTIF,
                  pooling_type = "maxavg")
output = convnet.forward_test(torch.randn(10, 4, 38))
```

Feed-forward 연산 방식 정의

```
def forward(self, x):
    x = self.l_conv(x)
    x = self.relu(x)
    m, _ = torch.max(x, dim=2)
    if self.pooling_type == "maxavg":
        a = torch.mean(x, dim=2)
        x = torch.cat((m,a), dim=1)
    else:
        x = m
    logit = self.l_nnet(x)
    y = torch.sigmoid(logit)
    return y
```

Train the DL model

```
import torch.optim as optim
from sklearn import datasets, metrics
epochs = 50
batch_size = 32
learning_rate = 0.01
criterion = torch.nn.BCELoss()
optimizer = optim.SGD(convnet.parameters(), lr=learning_rate)
print(criterion)
print(optimizer)
```

Epochs – 전체 반복 학습 횟수

Batch Size – 학습 데이터 분할 학습 시, 분할 데이터 크기

Learning Rate – 경사하강법을 사용한 최적화 알고리즘 사용 시, 모델 매개변수 갱신 계수

Criterion – 모델의 예측 값과 실제 정답 사이의 오차 계산을 위한 수식, 해당 문제는 이진 분류 (binary classification)이므로 (binary) cross-entropy 오차 함수가 사용됨

Optimizer – 경사하강법 기반 최적화 알고리즘, 여기서는 stochastic gradient descent 가 사용됨

Train the DL model

```
# numpy array to tensor
ts_train_x = torch.tensor(train_x, dtype=torch.float32)
ts_train_y = torch.tensor(train_y, dtype=torch.float32)
ts_valid_x = torch.tensor(valid_x, dtype=torch.float32)
ts_valid_y = torch.tensor(valid_y, dtype=torch.float32)
ts_test_x = torch.tensor(test_x, dtype=torch.float32)
ts_test_y = torch.tensor(test_y, dtype=torch.float32)
print(ts_train_x.size(), ts_train_y.size())
print(ts_valid_x.size(), ts_valid_y.size())
print(ts_test_x.size(), ts_test_y.size())

torch.Size([5000, 4, 38]) torch.Size([5000, 1])
torch.Size([500, 4, 38]) torch.Size([500, 1])
torch.Size([100, 4, 38]) torch.Size([100, 1])
```



```
train_data = [(_x,_y) for _x, _y in zip(ts_train_x, ts_train_y)]
valid_data = [(_x,_y) for _x, _y in zip(ts_valid_x, ts_valid_y)]
test_data = [(_x,_y) for _x, _y in zip(ts_test_x, ts_test_y)]
```



```
train_loader = torch.utils.data.DataLoader(dataset = train_data,
                                           batch_size = batch_size,
                                           shuffle = True)
```

Train the DL model

```

convnet.to(device)

convnet.train()
for epoch in range(epochs):
    # Training
    train_loss = []
    for index, (data, target) in enumerate(train_loader):
        data = data.to(device)
        target = target.to(device)
        optimizer.zero_grad()
        output = convnet(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()
        train_loss.append(loss.item())

    # Validation
    valid_output = convnet(ts_valid_x.to(device))
    valid_loss = criterion(valid_output.to(device), ts_valid_y.to(device))
    valid_loss = valid_loss.item()
    valid_roc_auc = metrics.roc_auc_score(ts_valid_y.detach().numpy(),
                                         valid_output.detach().numpy())

    print("Epoch {:>6} | Train Loss {:.5f} | Valid Loss {:.5f} | Valid ROC-AUC {:.5f}".format(epoch,
                                                                                           np.mean(train_loss),
                                                                                           valid_loss,
                                                                                           valid_roc_auc))

```

Epoch	36	Train Loss 0.53760	Valid Loss 0.56233	Valid ROC-AUC 0.78650
Epoch	37	Train Loss 0.53357	Valid Loss 0.55847	Valid ROC-AUC 0.78926
Epoch	38	Train Loss 0.53006	Valid Loss 0.55553	Valid ROC-AUC 0.79217
Epoch	39	Train Loss 0.52820	Valid Loss 0.55305	Valid ROC-AUC 0.79475
Epoch	40	Train Loss 0.52341	Valid Loss 0.54952	Valid ROC-AUC 0.79755
Epoch	41	Train Loss 0.52053	Valid Loss 0.54683	Valid ROC-AUC 0.79973
Epoch	42	Train Loss 0.51757	Valid Loss 0.54374	Valid ROC-AUC 0.80244
Epoch	43	Train Loss 0.51455	Valid Loss 0.54270	Valid ROC-AUC 0.80444
Epoch	44	Train Loss 0.51021	Valid Loss 0.54070	Valid ROC-AUC 0.80686
Epoch	45	Train Loss 0.50952	Valid Loss 0.53601	Valid ROC-AUC 0.80931
Epoch	46	Train Loss 0.50655	Valid Loss 0.53470	Valid ROC-AUC 0.81141
Epoch	47	Train Loss 0.50300	Valid Loss 0.53116	Valid ROC-AUC 0.81365
Epoch	48	Train Loss 0.49952	Valid Loss 0.52929	Valid ROC-AUC 0.81523
Epoch	49	Train Loss 0.49842	Valid Loss 0.52691	Valid ROC-AUC 0.81663



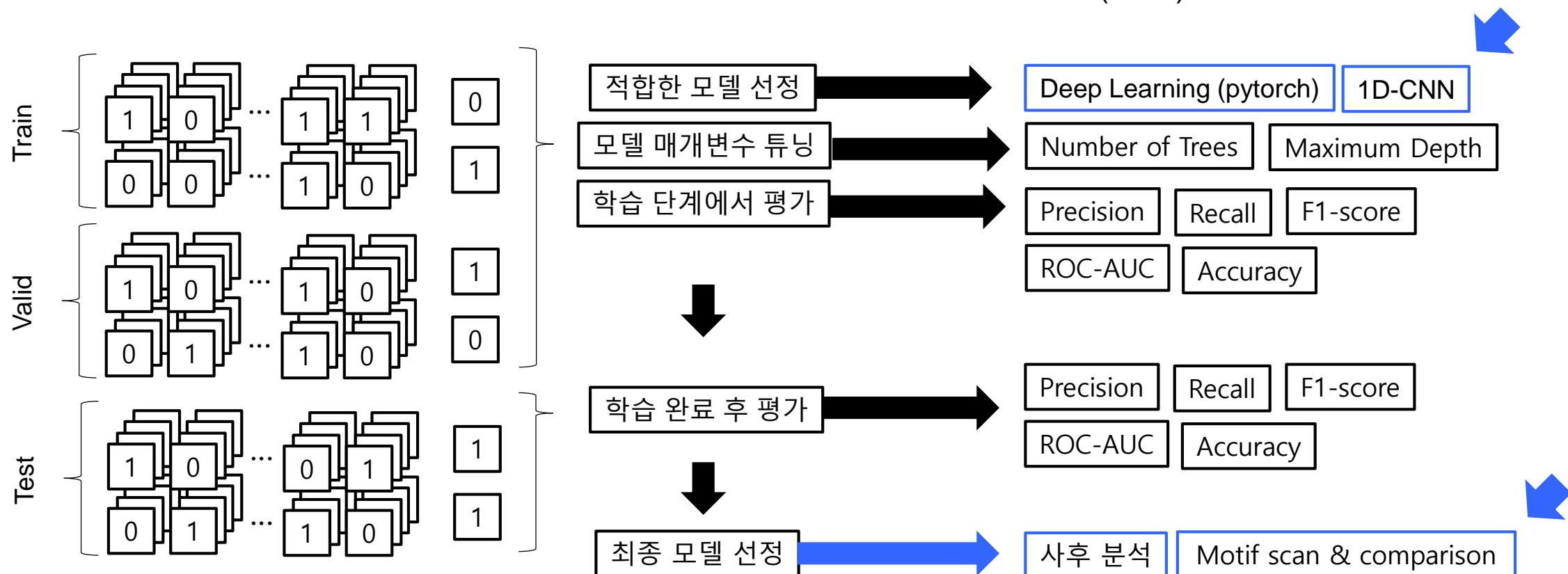
Test the DL model

```
# Testing
test_output = convnet(ts_test_x.to(device)).detach().numpy()
test_output = test_output > 0.5
print(metrics.classification_report(test_y, test_output))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	1.00	0.71	0.83	100
accuracy			0.71	100
macro avg	0.50	0.35	0.42	100
weighted avg	1.00	0.71	0.83	100

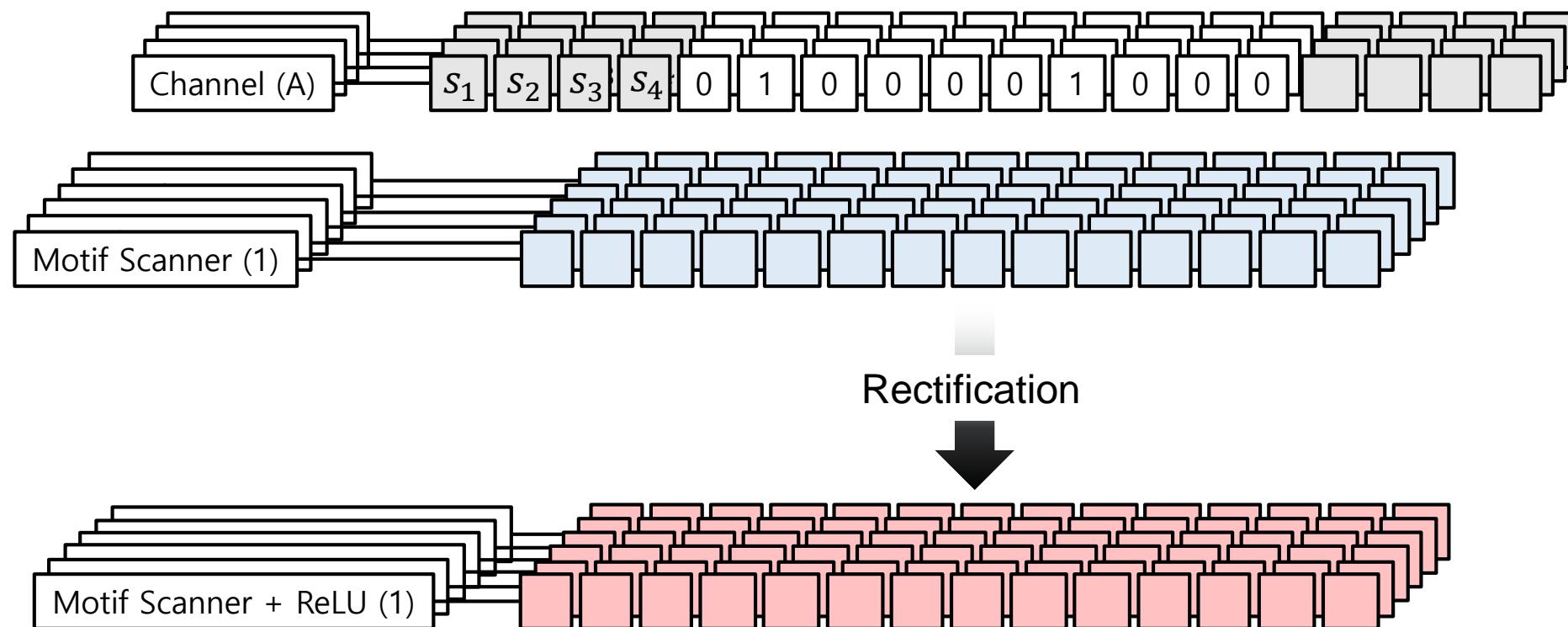
1D-CNN (Convolutional Neural Network) + Motif scan

- 모델 평가 및 선별 방식은 크게 변하지 않음
- 학습 모델을 딥 러닝 모델 사용
- 1D-Convolutional Neural Network를 사용하여 motif 를 찾는 것 까지 진행 (예정)



Motif Scanning

- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행



Motif Scanning

- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행

모델에서 사용할 Layer 초기화 및 정의

```
class ConvNet(nn.Module):
    def __init__(self, num_motifs, len_motif, pooling_type="max"): # pool_type = [max, maxavg]
        super(ConvNet, self).__init__()
        self.l_conv = nn.Conv1d(in_channels=4,
                             out_channels=num_motifs,
                             kernel_size=len_motif,
                             stride=1,
                             padding=0)

        self.pooling_type = pooling_type
        self.relu = nn.ReLU()
        if pooling_type == "max":
            self.l_nnet = nn.Linear(num_motifs, 1)
        else:
            self.l_nnet = nn.Linear(num_motifs * 2, 1)

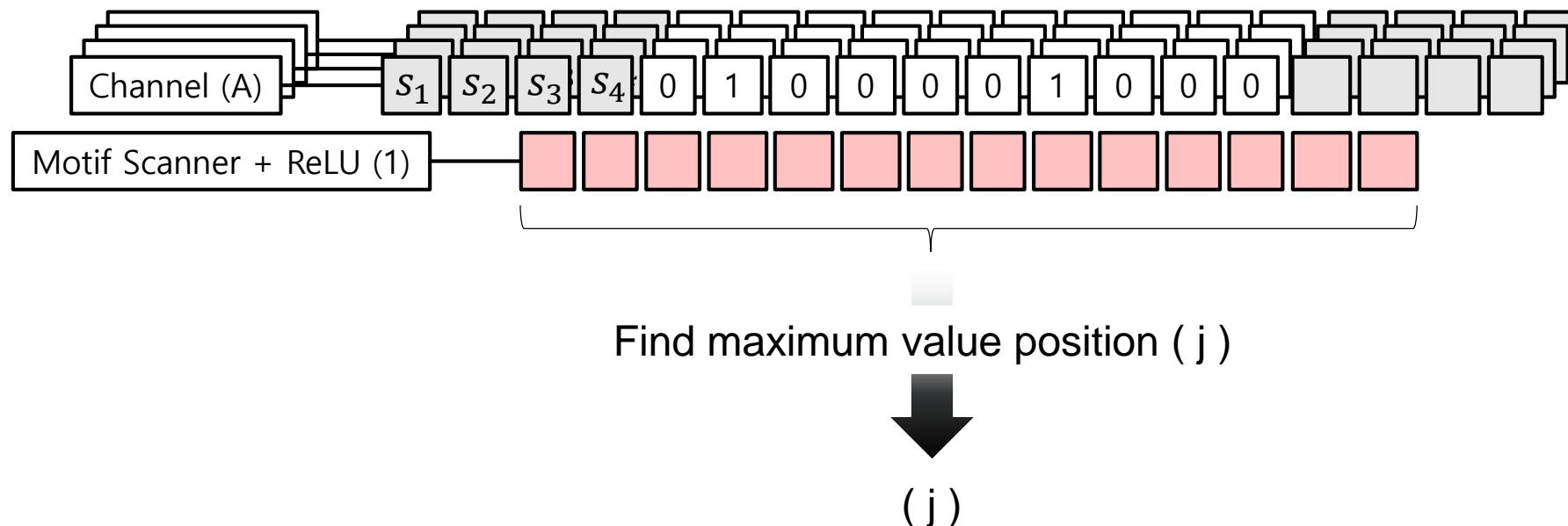
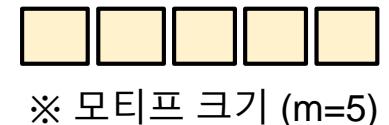
    print("- Motif scanners : {}".format(self.l_conv.weight.size()))
```

Convolution & Rectification only

```
def get_rectified_motif_scan(self, x):
    x = self.l_conv(x)
    x = self.relu(x)
    return x
```

Motif Scanning

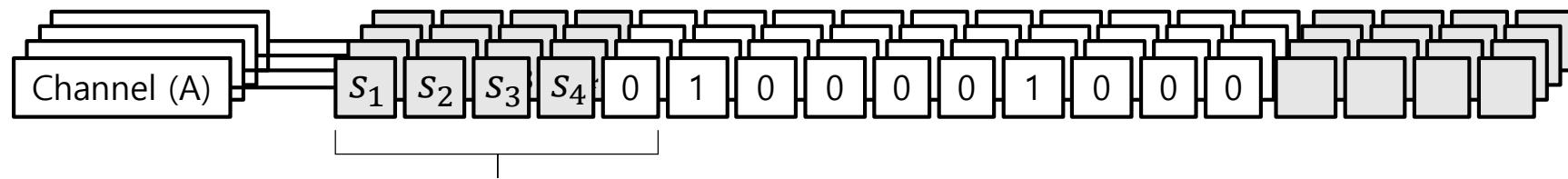
- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행
- 출력 결과(각 모티프 별)에서 가장 높은 값을 갖는 위치(j)를 찾는다



Motif Scanner 가 입력 서열에서 가장 중요하게 보는 지점을 찾는 과정

Motif Scanning

- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행
- 출력 결과(각 모티프 별)에서 가장 높은 값을 갖는 위치(j)를 찾는다
- 입력 서열에서 해당 위치(j)에 대하여, $(j-m+1 \sim j)$ 범위의 부분 서열을 추출한다

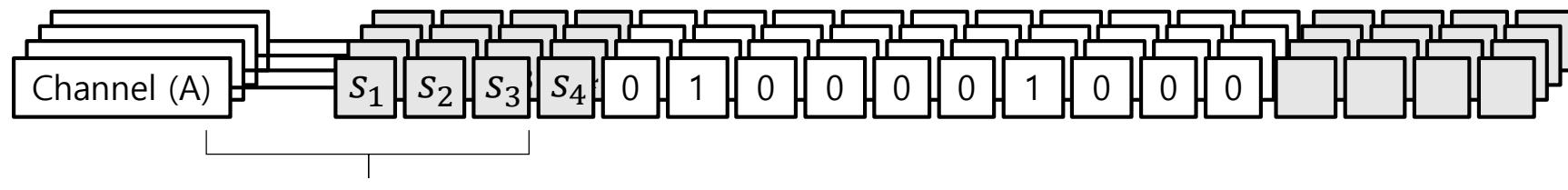


추출 된 부분 서열
(= 모티프 스캐너가 중요하다고 판단한 영역)

Motif Scanner 가 입력 서열에서 가장 중요하게 보는 지점을 찾는 과정

Motif Scanning

- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행
- 출력 결과(각 모티프 별)에서 가장 높은 값을 갖는 위치(j)를 찾는다
- 입력 서열에서 해당 위치(j)에 대하여, $(j-m+1 \sim j)$ 범위의 부분 서열을 추출한다

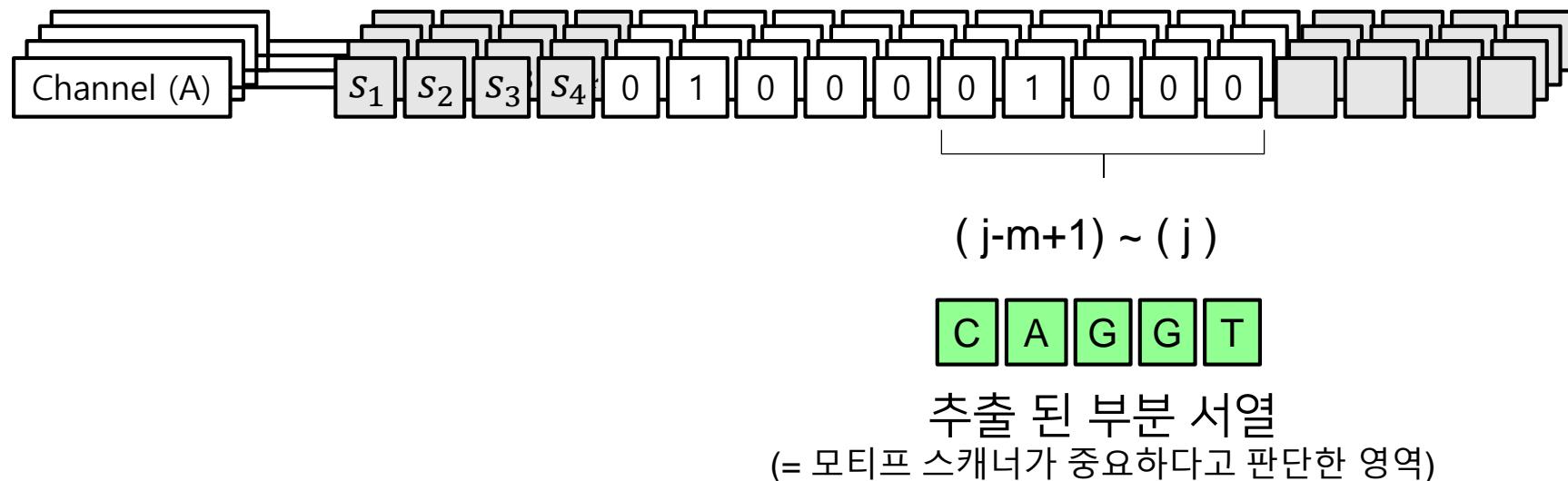


추출 된 부분 서열
 (= 모티프 스캐너가 중요하다고 판단한 영역)

Motif Scanner 가 입력 서열에서 가장 중요하게 보는 지점을 찾는 과정

Motif Scanning

- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행
- 출력 결과(각 모티프 별)에서 가장 높은 값을 갖는 위치(j)를 찾는다
- 입력 서열에서 해당 위치(j)에 대하여, $(j-m+1 \sim j)$ 범위의 부분 서열을 추출한다



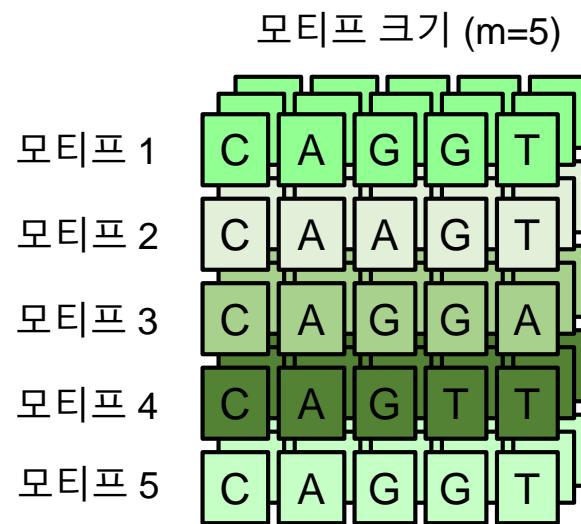
Motif Scanner 가 입력 서열에서 가장 중요하게 보는 지점을 찾는 과정

Motif Scanning

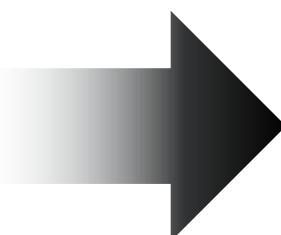
- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행
- 출력 결과(각 모티프 별)에서 가장 높은 값을 갖는 위치(j)를 찾는다
- 입력 서열에서 해당 위치(j)에 대하여, $(j-m+1 \sim j)$ 범위의 부분 서열을 추출한다
- 각 Motif Scanner 별로 부분 서열이 추출되고 이를 사용하여 Position Frequency Matrix (PFM)을 구성함



※ 모티프 크기 (m=5)



추출 된 부분 서열들
(= 모티프 스캐너가 중요하다고 판단한 영역)
모티프 스캐너마다 부분 서열들이 수집된다



read#1	A	T	T	T	G	C	G	T	C	G	A	...
read#2	T	T	-	C	C	G	T	-	-	A	...	
read#3	A	T	T	-	G	-	G	T	C	G	T	...
read#4		T	-	G	C	G	A	C	G	A	...	
read#5		A	-	G	C	G	A	C	G	A	...	
<i>REF_{pos}</i>	1	2	3	-	4	5	6	7	8	9	10	...
A	2	0	1	-	0	0	0	2	0	0	4	...
C	0	0	0	-	1	4	0	0	4	0	0	...
G	0	0	0	-	4	0	5	0	0	4	0	...
T	0	3	4	-	0	0	0	3	0	0	1	...

PFM =

$$\begin{bmatrix} A & 2 & 0 & 1 & - & 0 & 0 & 0 & 2 & 0 & 0 & 4 & \dots \\ C & 0 & 0 & 0 & - & 1 & 4 & 0 & 0 & 4 & 0 & 0 & \dots \\ G & 0 & 0 & 0 & - & 4 & 0 & 5 & 0 & 0 & 4 & 0 & \dots \\ T & 0 & 3 & 4 & - & 0 & 0 & 0 & 3 & 0 & 0 & 1 & \dots \end{bmatrix}$$

부분 서열 정보를 누적하여 PFM을 구성

Motif Scanning

- 모티프 검출을 위한 서열들을 준비 (ex. 테스트 용 데이터 서열)
- 모델의 Convolution 및 Rectification 단계까지 수행
- 출력 결과(각 모티프 별)에서 가장 높은 값을 갖는 위치(j)를 찾는다
- 입력 서열에서 해당 위치(j)에 대하여, $(j-m+1 \sim j)$ 범위의 부분 서열을 추출한다
- 각 Motif Scanner 별로 부분 서열이 추출되고 이를 사용하여 Position Frequency Matrix (PFM)을 구성함

```
# 1. Get rectified motif scan result (from model) from the test dataset sequences
rect_scans = convnet.get_rectified_motif_scan(ts_test_x.to(device)).detach().numpy()
print("Returned rectified motif scan data {}".format(rect_scans.shape))
```

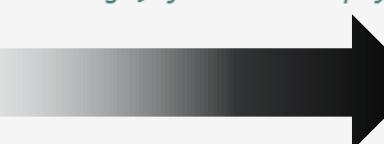
```
# 4. Accumulate all subsequences (length m) and count letter of each position
test_subseqs_list = defaultdict(lambda: [])

for test_seq, rect_scan in zip(ts_test_x, rect_scans):
    test_seq = test_seq.detach().numpy()

    for m_i, m_scans in enumerate(rect_scan):
        # 2. For each motif, find maximum valued position (j)
        j = np.argmax(m_scans)
        j_m_1 = j - LEN_MOTIF + 1

        # 3. Extract subsequence from input sequence where  $(j-m+1) \sim (j)$ 
        # 3-1 when the positions  $(j-m+1) \sim (j)$  beyond the sequence range, fill the empty
        subseq = get_subsequence(test_seq, j_m_1, j+1)

        # print(m_i+1, j_m_1, j, subseq, len(subseq))
        test_subseqs_list[m_i].append(subseq)
```



```
def get_subsequence(seq_x, start, end, letters="ACGT"):
    ret = ""
    ldict = {i:l for i,l in enumerate(letters)}
    if start < 0:
        ret = "N"*(-start) + ret
        for i in range(end):
            x = seq_x[:, i]
            base = ldict[np.argmax(x)]
            ret = ret + base
    else:
        for i in range(start, end):
            x = seq_x[:, i]
            base = ldict[np.argmax(x)]
            ret = ret + base

    return ret
```



Thank you

Q & A

DeepBind Example full script ([link](#))