Deep Learning Based DNA Sequence Fast Decoding

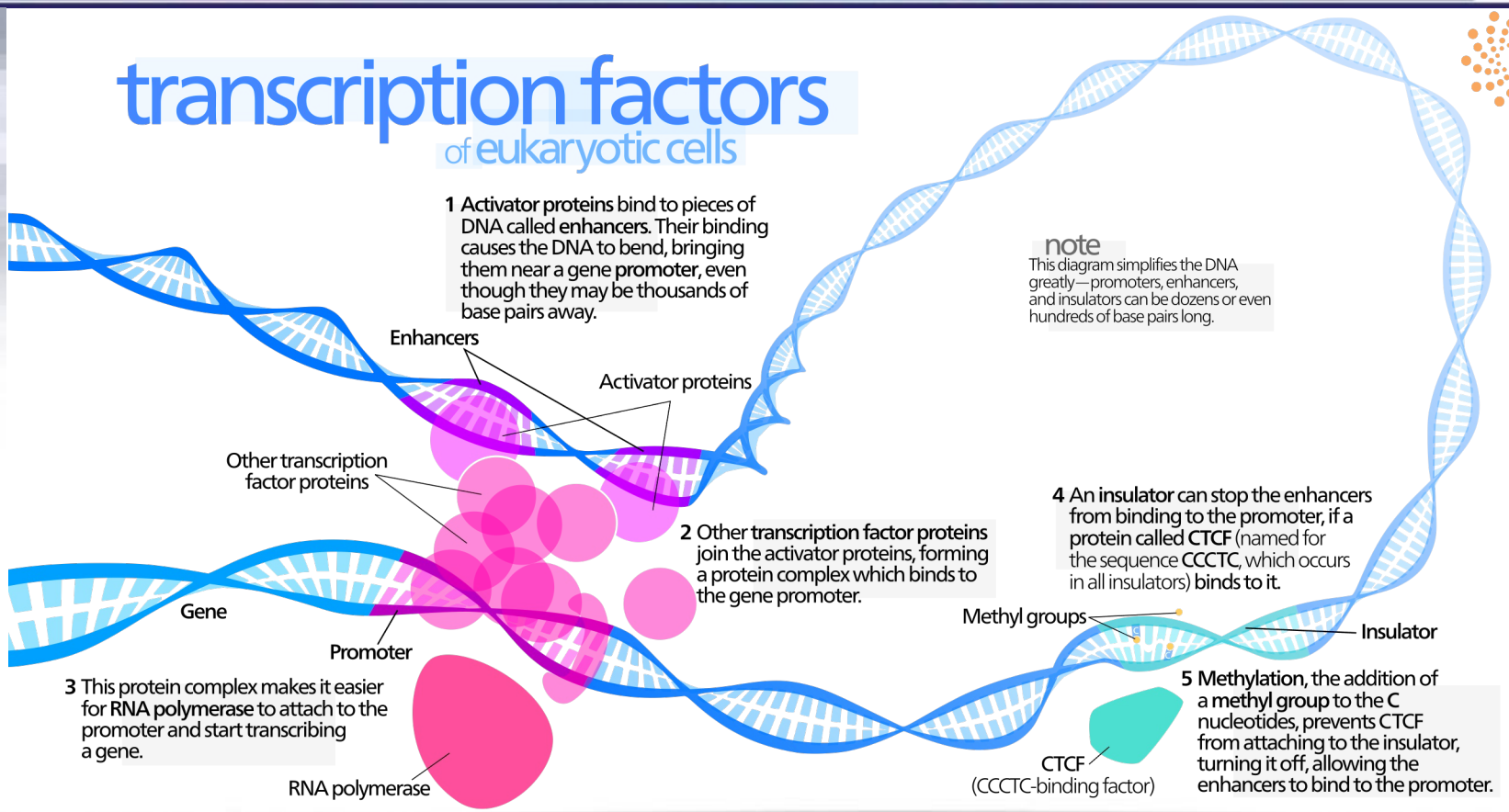
21.06.2022

Pengzhi Zhu, Ke Ding

Outline

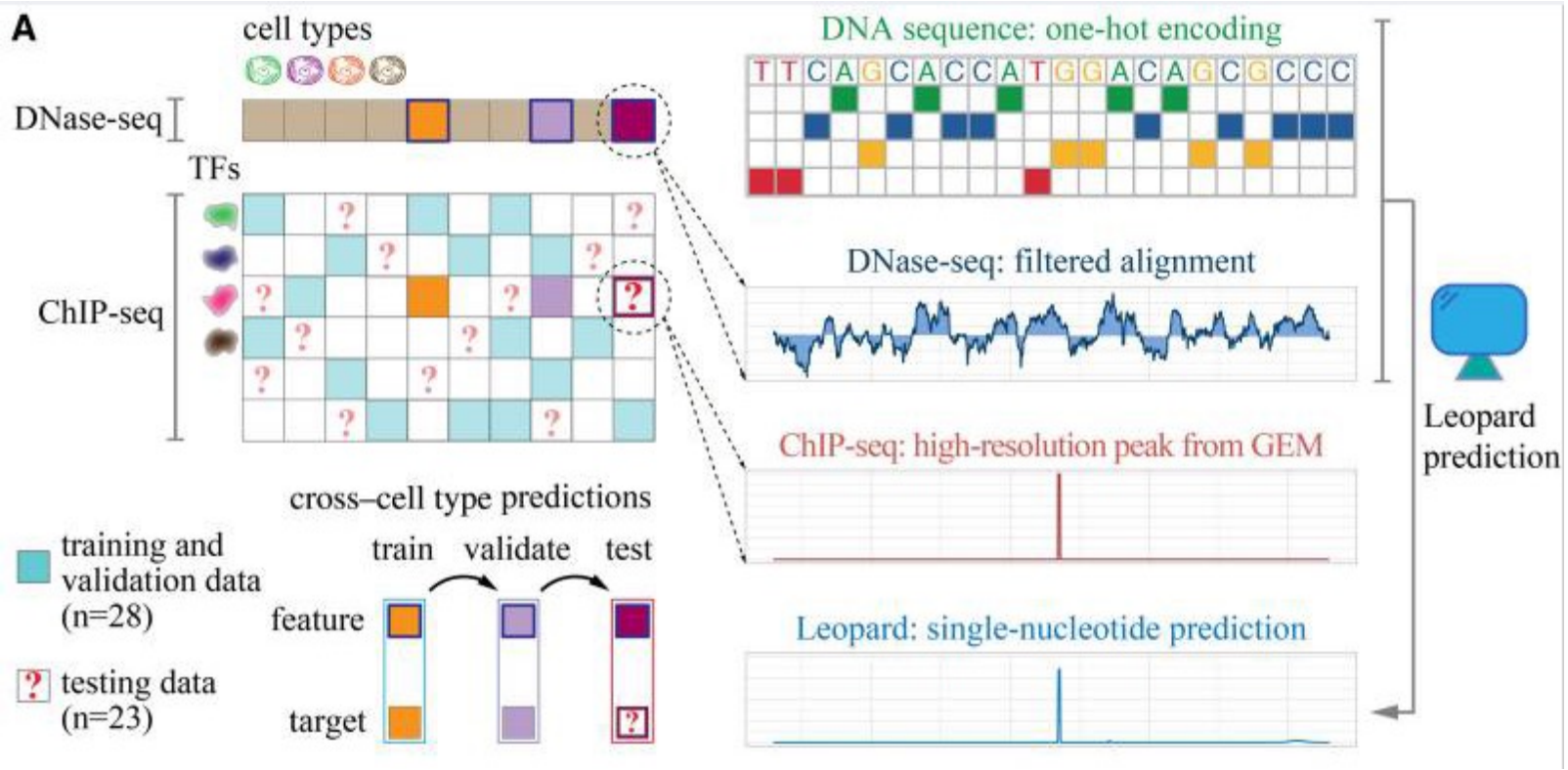
- Brief introduction to Transcription Factor
- How to build a model using deep neural network
- How to speedup your NN using Horovod
- How to submit a job in Gadi
- Q/A

Brief introduction to Transcription Factor



(Image from link)

How to build a model using deep neural network



(Image from link)

How to build a model using deep neural network

One hot Encoding DNA sequence

- **categorical** to **numbers**
- **Input Dimension: 4 x N**

	C	G	A	T	A	A	C	C	G	A	T	A	T
A	0	0	1	0	1	1	0	0	0	1	0	1	0
C	1	0	0	0	0	0	1	1	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	1	0	0	0	0	0	0	1	0	1

(Image from link)

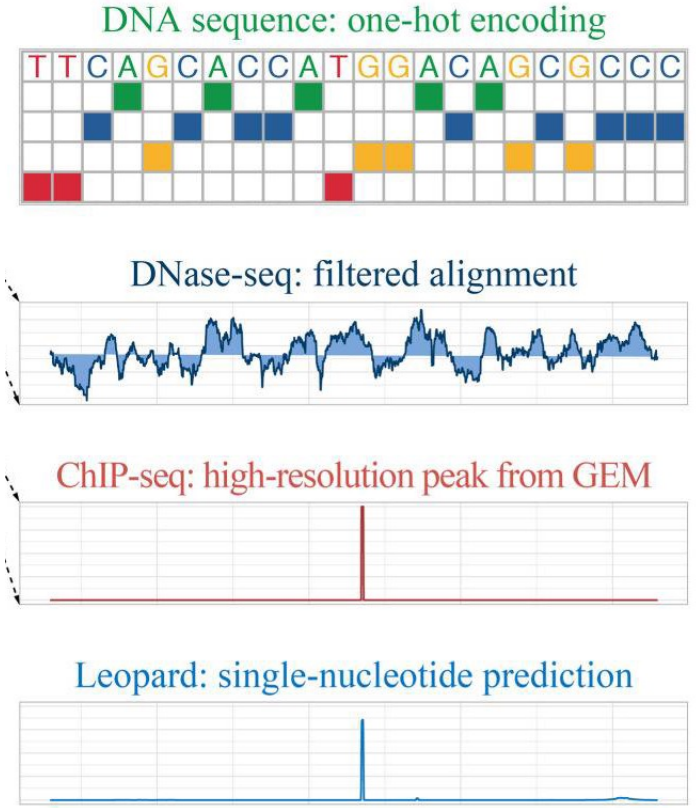
How to build a model using deep neural network

Input:

- Combine biological meaningful data
- DNase-seq: chromatin accessibility
- Input Dimension: 5 x N

Output:

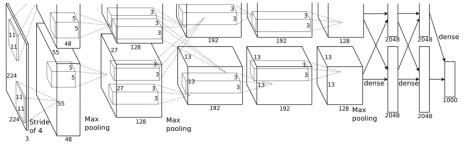
- ChIP-seq: Protein-DNA relationship
- Peak calling using GEM
- Output Dimension: 1 x N

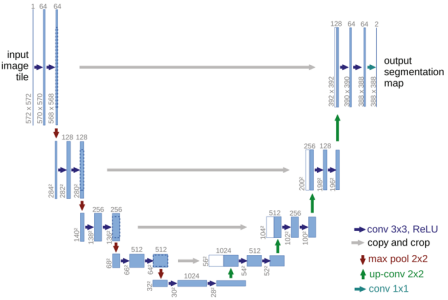


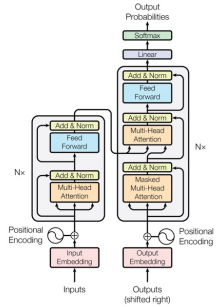
(Image from link)

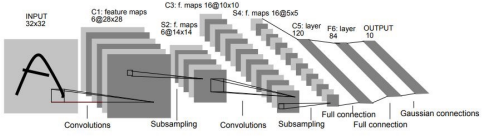
How to build a model using deep neural network

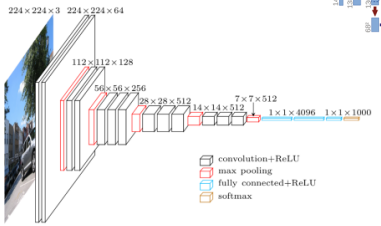
DL Development Timeline



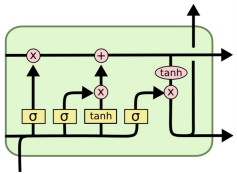








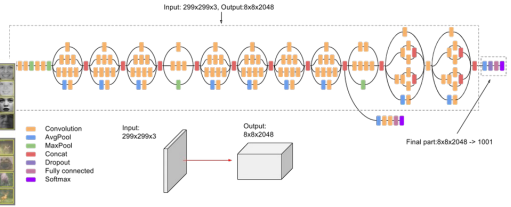
1986
RNNs



1989
LeNet

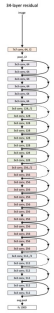
1997
LSTMs

2012
AlexNet



2014
VGG-16
GANs

2015
Inception-V1-3
UNet

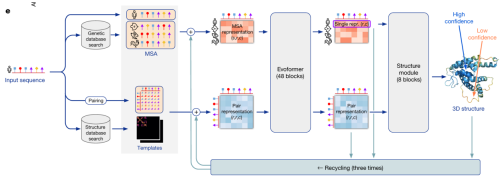


2016
ResNet-50

2017
Transformer

2018
BERT

2021
AlphaFold2
Enformer

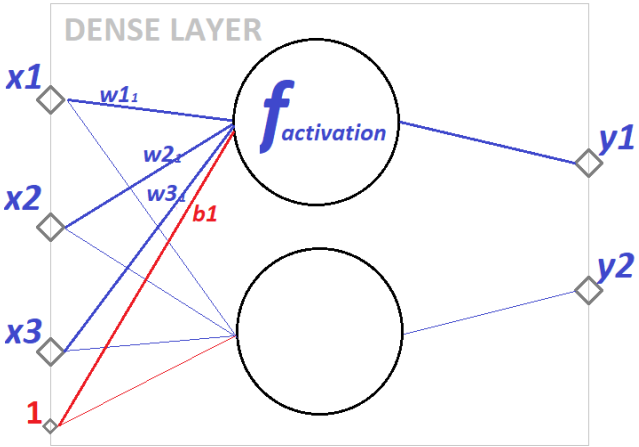


Building Block of NN

- DNN
- CNN
- RNN
- Transformer

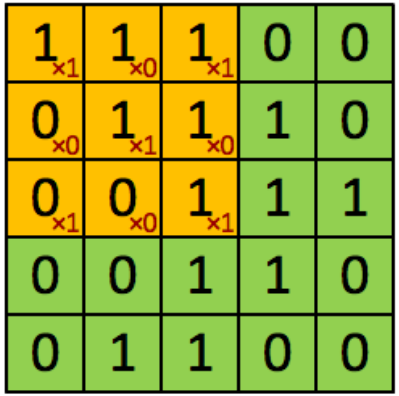
How to build a model using deep neural network

DNN

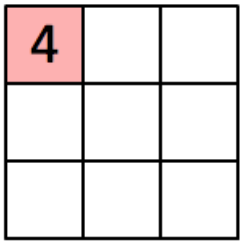


Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Convolution Neural Network



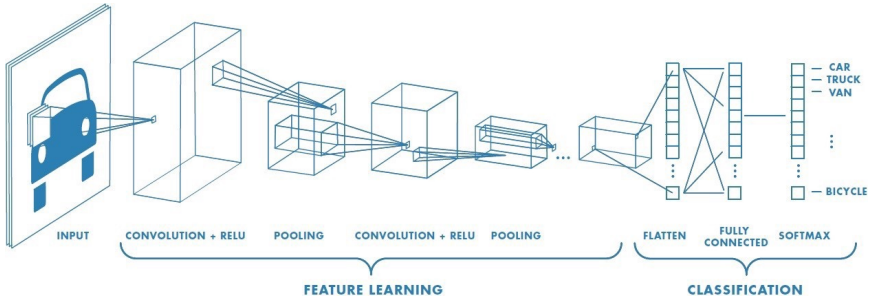
Image

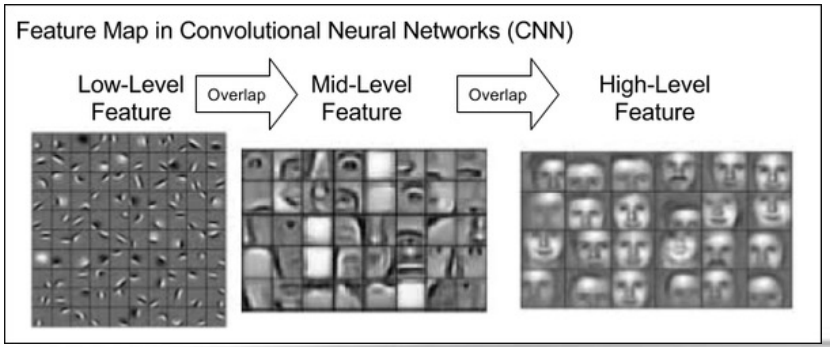


Convolved Feature

- **The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image.**

Convolution Neural Network





Why CNN achieves great success in computer vision.

- **Hierarchical feature representation:**
  - First layers detect low level features, i.e., color, gradients, etc.
  - The deeper layers detect high level features which are more abstract.

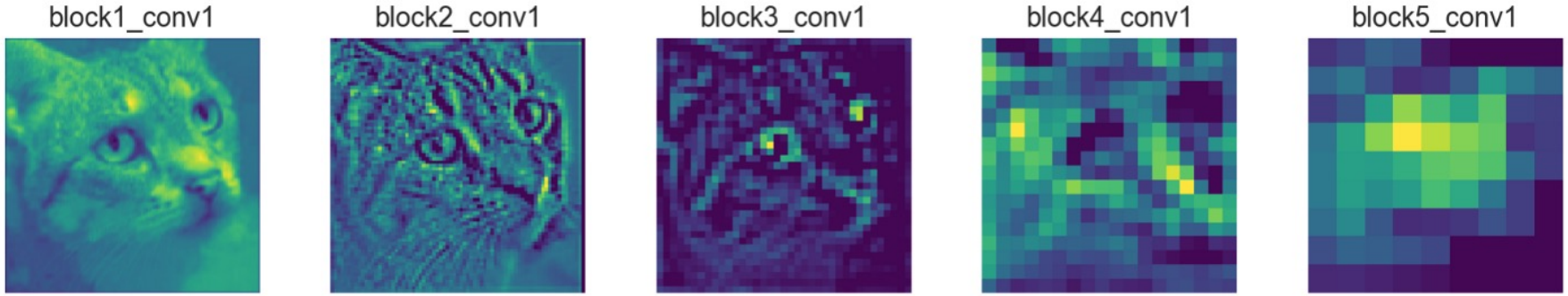
Convolution Neural Network

Why CNN achieves great success in computer vision.

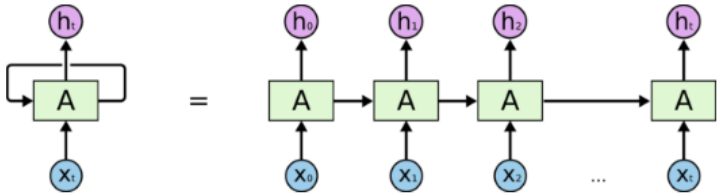
- **Locality: related inputs are close to each other. (i.e., color, lighting and texture)**

Will CNN still able to learn long range dependency in Genomic sequence?

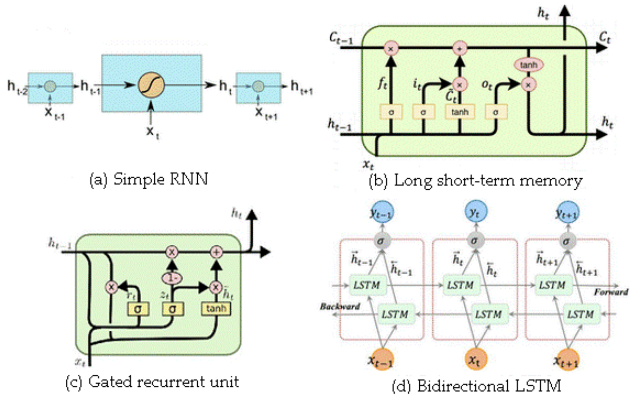
What if input genomic sequence has arbitrary length?



Recurrent Neural Network

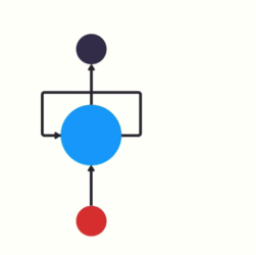


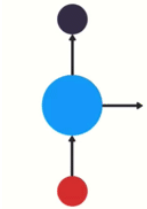
An unrolled recurrent neural network.



- **can use their internal memory to process arbitrary sequences of inputs.**
- **It can handle long range dependency**

Recurrent Neural Network





- **RNN and LSTM can't be trained in parallel.**
- **What if we have a very long input? i.e., 10240bp**

Transformer

Attention is all you need
A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc
… the number of **attention** heads and the **attention** key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head **attention** is 0.9 …
☆ Save 99 Cite Cited by 37308 Related articles All 35 versions »

[PDF] neurips.cc

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

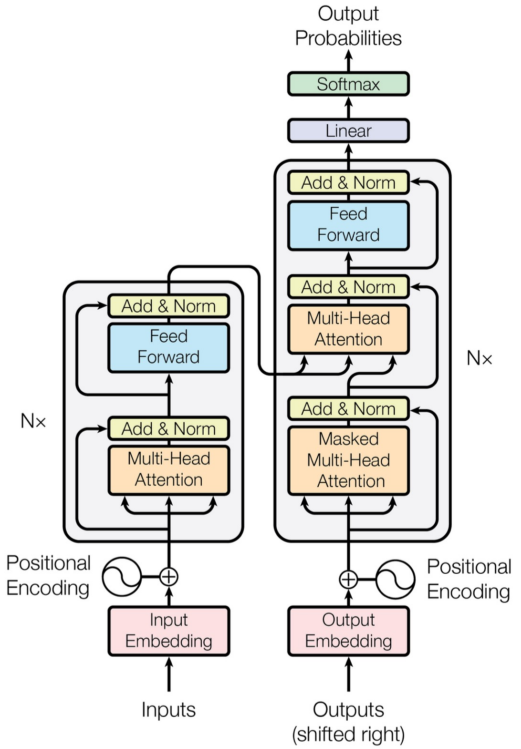
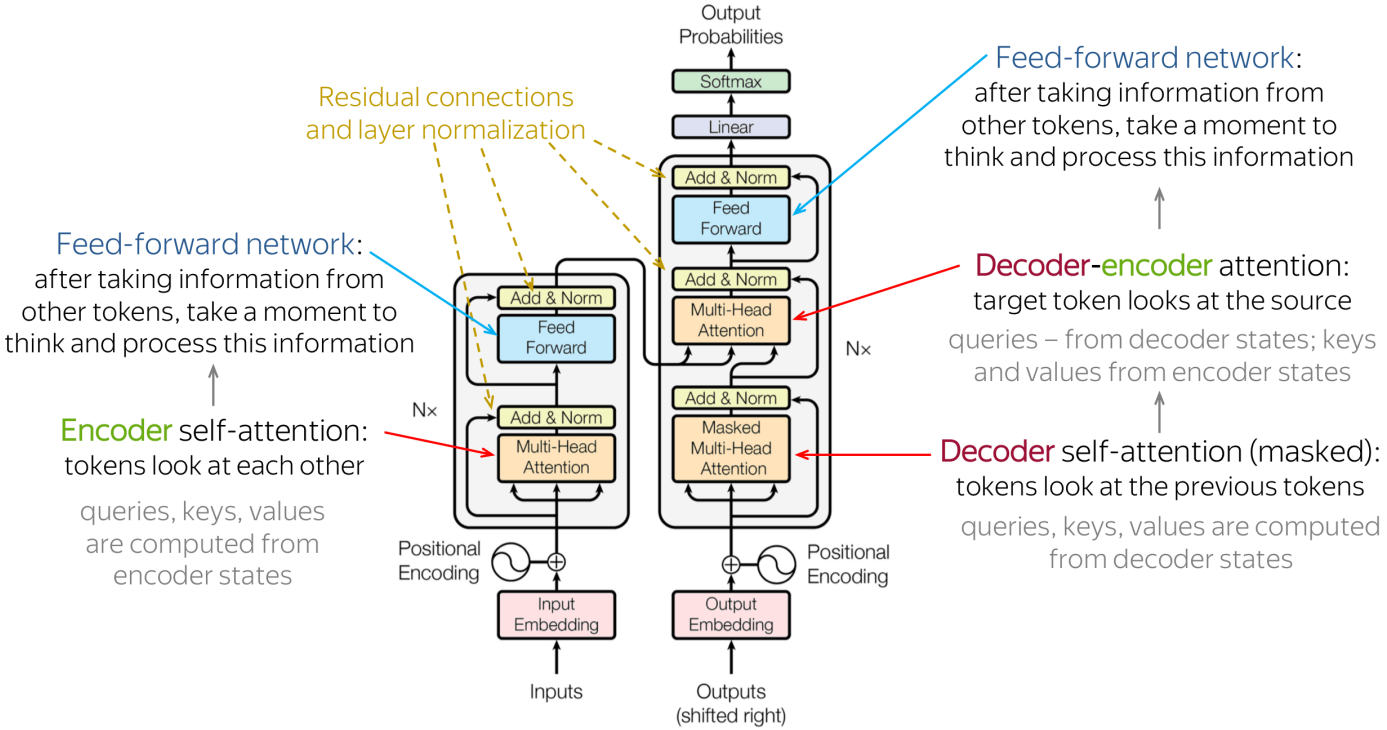
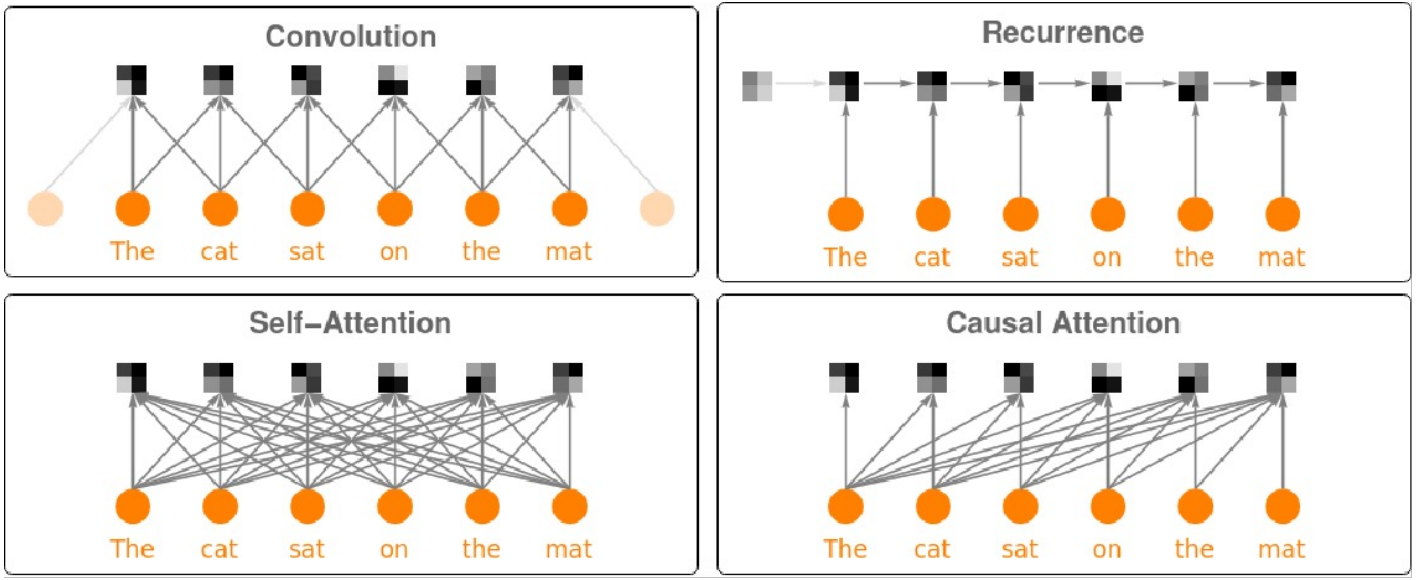


Figure 1: The Transformer - model architecture.

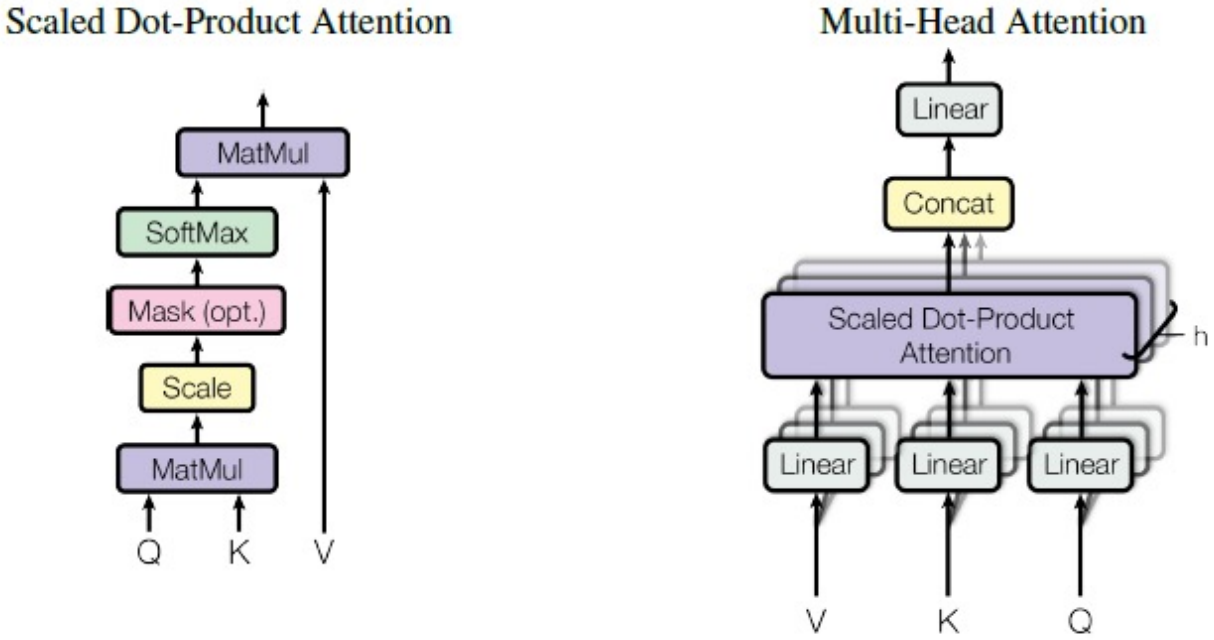
Transformer



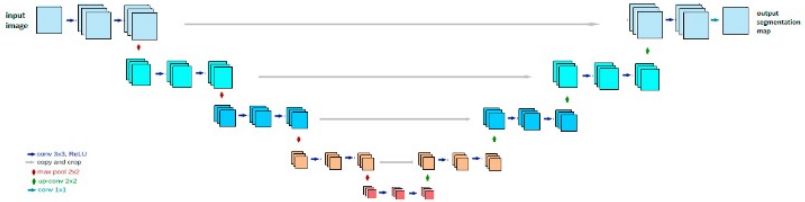
Contribution between input and output

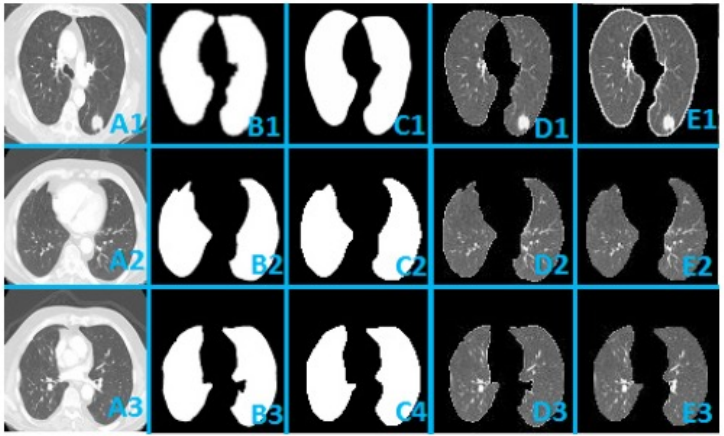


Transformer – Attention Mechanism



UNet Applications

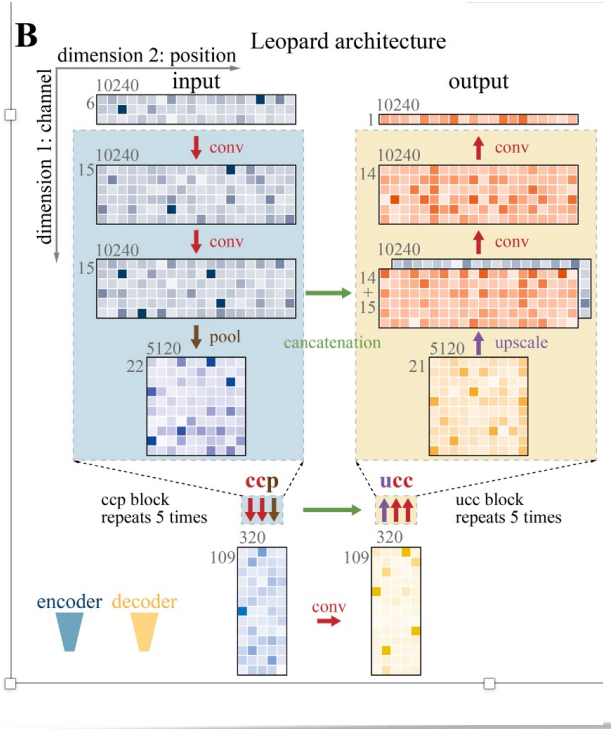




- **Lung CT image**

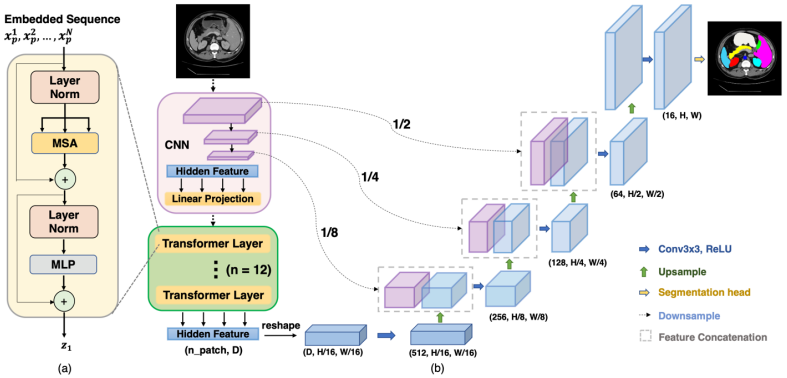
- **Computed Tomography**
- **Malignant lung tumor detection**
- **Limited training examples**

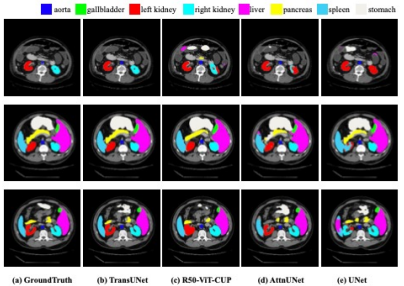
Transcription Factor Binding Prediction



- **With input of 10240 nt DNA sequence**
- **Output of binding site also in single nt resolution**
- **Combine extra information of DNase**
- **Many to many**

UNet Applications

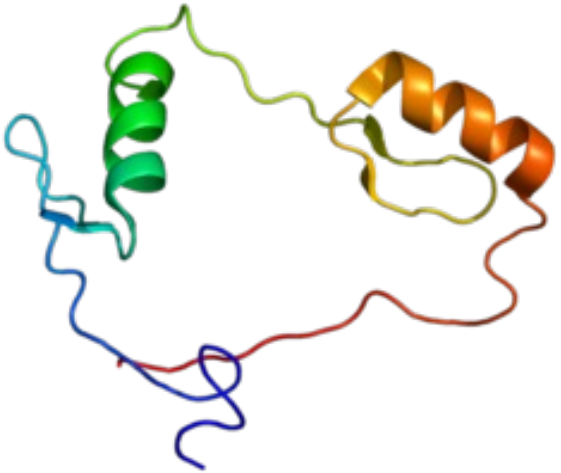




- **TransUNet**
  - Using CNN to extract features
  - Using Transformer to further encode the features
  - ViT as backbone

For simplicity

- **We only focus on a single TF: CTCF**
- **We have preprocessed the data and saved in tf.data.Dataset**
- **We have prepared a simple CNN model**



How to build a model using deep neural network

For simplicity

- We only focus on a single TF: CTCF
- **We have preprocessed the data and saved in tf.data.Dataset**
- We have prepared a simple CNN model

```
train_dataset = tf.data.experimental.load(os.path.join(data_dir + train_data_name))
val_dataset = tf.data.experimental.load(os.path.join(data_dir + val_data_name))
test_dataset = tf.data.experimental.load(os.path.join(data_dir + test_data_name))
```

How to build a model using deep neural network

For simplicity

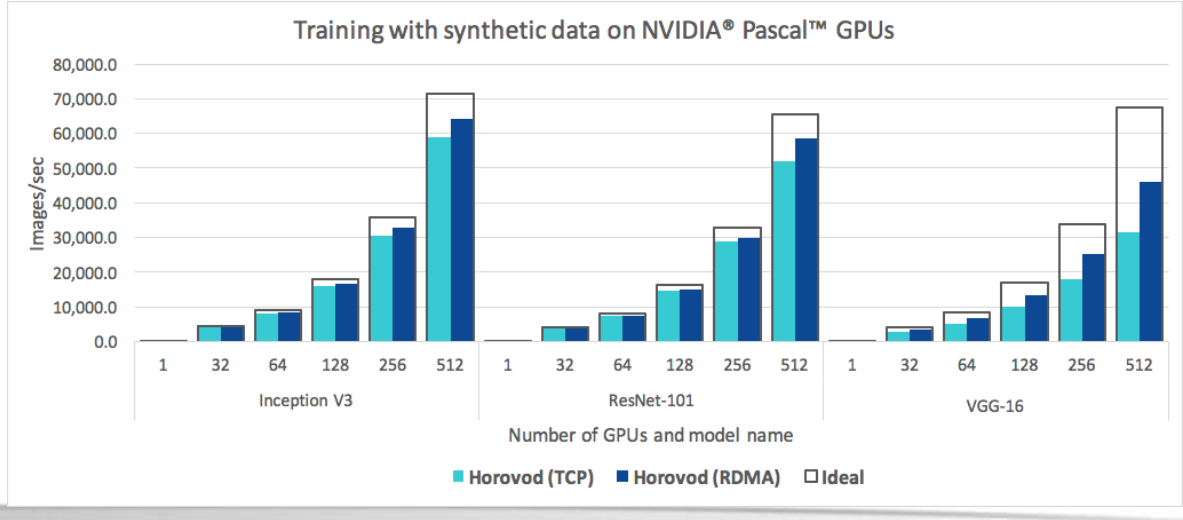
- We only focus on a single TF: CTCF
- We have preprocessed the data and saved in tf.data.Dataset
- **We have prepared a simple CNN model**

```
def cnn_model(max_len, vocab_size):
    model = Sequential([
        InputLayer(input_shape=(max_len, vocab_size)),
        Conv1D(32, 17, padding='same', activation='relu'),
        Conv1D(64, 11, padding='same', activation='relu'),
        Conv1D(128, 5, padding='same', activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    return model
```

HOROVOD

- **Distributed NN training framework**
- **Key Advantages:**
  - support multiple frameworks
  - minimal modification
  - fast



(Image from link)

How to speedup your NN using Horovod

- **Step 1:** . Start by importing Horovod and initializing it.


```
1  import horovod.tensorflow.keras as hvd
2
3  # Initialize Horovod
4  hvd.init()
```

How to speedup your NN using Horovod

- **Step 2:** . Pin GPU

```
1 # Pin GPU to be used to process local rank (one GPU per proce
  ss)
2 gpus = tf.config.experimental.list_physical_devices('GPU')
3 for gpu in gpus:
4     tf.config.experimental.set_memory_growth(gpu, True)
5 if gpus:
6     tf.config.experimental.set_visible_devices(gpus[hvd.local
  _rank()], 'GPU')
```

How to speedup your NN using Horovod

- **Step 3:** Scale the batch size and learning rate


```
1  # Scale the learning rate based on the number of GPUs
2  opt = tf.optimizers.Adam(0.001 * hvd.size())
```

How to speedup your NN using Horovod

- **Step 4:** Wrap the optimizer


```
# Wrap the optimizer with Horovod DistributedOptimizer.
opt = hvd.DistributedOptimizer(opt)
```

How to speedup your NN using Horovod

- **Step 5:** Broadcast the initial variable states


```
# Horovod: broadcast initial variable states from rank 0 to all other processes.
# This is necessary to ensure consistent initialization of all workers when
# training is started with random weights or restored from a checkpoint.
hvd.callbacks.BroadcastGlobalVariablesCallback(0),
```

How to speedup your NN using Horovod

- **Step 6:** Save checkpoints only on one worker


```
# Horovod: save checkpoints only on worker 0 to prevent other workers from corrupting them.
if hvd.rank() == 0:
    callbacks.append(tf.keras.callbacks.ModelCheckpoint(checkpoint_dir)
```

How to submit job to Gadi

Gadi

- Gadi contains a total of 155,000 CPU cores, 567 Terabytes of memory and 640 GPUs.
- Rank 57 in latest Top500 HPC Lists
- PBS job system
- A detailed introduction will be held on 1st of July

How to submit job to Gadi

#PBS key steps:

- Define your project
- Define your queue
- Define your wall time
- Define your resource

```
1  #!/bin/bash
2  #PBS -N Leopard_NCI
3  #PBS -P il82
4  #PBS -r y
5  #PBS -q gpuvolta
6  #PBS -l storage=gdata/ik06
7  #PBS -l walltime=01:30:00
8  #PBS -l ncpus=24
9  #PBS -l ngpus=2
10 #PBS -l mem=100GB
11 #PBS -M ke.ding@anu.edu.au
12 #PBS -m e
```

How to submit job to Gadi

#PBS key steps:

- Define your project
- Define your queue
- Define your wall time
- Define your resource

```
1  #!/bin/bash
2  #PBS -N Leopard_NCI
3  #PBS -P il82
4  #PBS -r y
5  #PBS -q gpuvolta
6  #PBS -l storage=gdata/ik06
7  #PBS -l walltime=01:30:00
8  #PBS -l ncpus=24
9  #PBS -l ngpus=2
10 #PBS -l mem=100GB
11 #PBS -M ke.ding@anu.edu.au
12 #PBS -m e
```

How to submit job to Gadi

#PBS key steps:

- Define your project
- Define your queue
- Define your wall time
- Define your resource

```
1  #!/bin/bash
2  #PBS -N Leopard_NCI
3  #PBS -P il82
4  #PBS -r y
5  #PBS -q gpuvolta
6  #PBS -l storage=gdata/ik06
7  #PBS -l walltime=01:30:00
8  #PBS -l ncpus=24
9  #PBS -l ngpus=2
10 #PBS -l mem=100GB
11 #PBS -M ke.ding@anu.edu.au
12 #PBS -m e
```

How to submit job to Gadi

#PBS key steps:

- Define your project
- Define your queue
- Define your wall time
- Define your resource

```
1  #!/bin/bash
2  #PBS -N Leopard_NCI
3  #PBS -P il82
4  #PBS -r y
5  #PBS -q gpuvolta
6  #PBS -l storage=gdata/ik06
7  #PBS -l walltime=01:30:00
8  #PBS -l ncpus=24
9  #PBS -l ngpus=2
10 #PBS -l mem=100GB
11 #PBS -M ke.ding@anu.edu.au
12 #PBS -m e
```

Load modules

```
1 #load modules for gpu support
2 module load cuda
3 module load cudnn
4 module load nccl
5 module load openmpi
```

Load conda environment

```
1  # setup conda environment
2  # -- change the path to your own conda direc
   tory
3  source /g/data/ik06/stark/anaconda3/etc/prof
   ile.d/conda.sh
4  conda init bash
5  conda activate deep_tf
```

Run your script

```
1  # run the bechmark over 2 GPUs
2  # -- change the path to your own
3  source /g/data/ik06/stark/NCI_Leopard/multi_gpus_train.sh
```

Marking Criterion

Submission:

1. One page description for:
   - your NN model's architecture
   - modifications to improve training speed
2. Whole project folder exclude data (for reproductivity)
3. Saved model (using *tf.keras.Model.save()*)
4. Performance summary (*evaluation_metrics.csv)*

Marking Criterion

Marking:

- Report: 20%
- Model performance: 60%
  - *pr_auc, dice_coef, binary_iou*
- Model training speed: 20%
  - *training_time*

**model training time should be evaluated on Gadi*

Demo: running the job on Gadi

Q/A