KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information

2022

스마트교통 빅데이터 분석

CNN을 이용한 교통 데이터 분석

2022.3.30.

이 홍 석 (hsyi@kisti.re.kr)

Day3-Lec1: 강의
CNN Model

학습 목표

❖ 목표

- ✓딥러닝 기반 이미지 분류하기 위해 지도학습과 이미지(특성) 및 레이블에 대하여 이해한다.
- ✓커널, 콘볼루션, 풀링을 이해하며 CNN 아키텍처에 대하여 이해한다.

❖ 배경

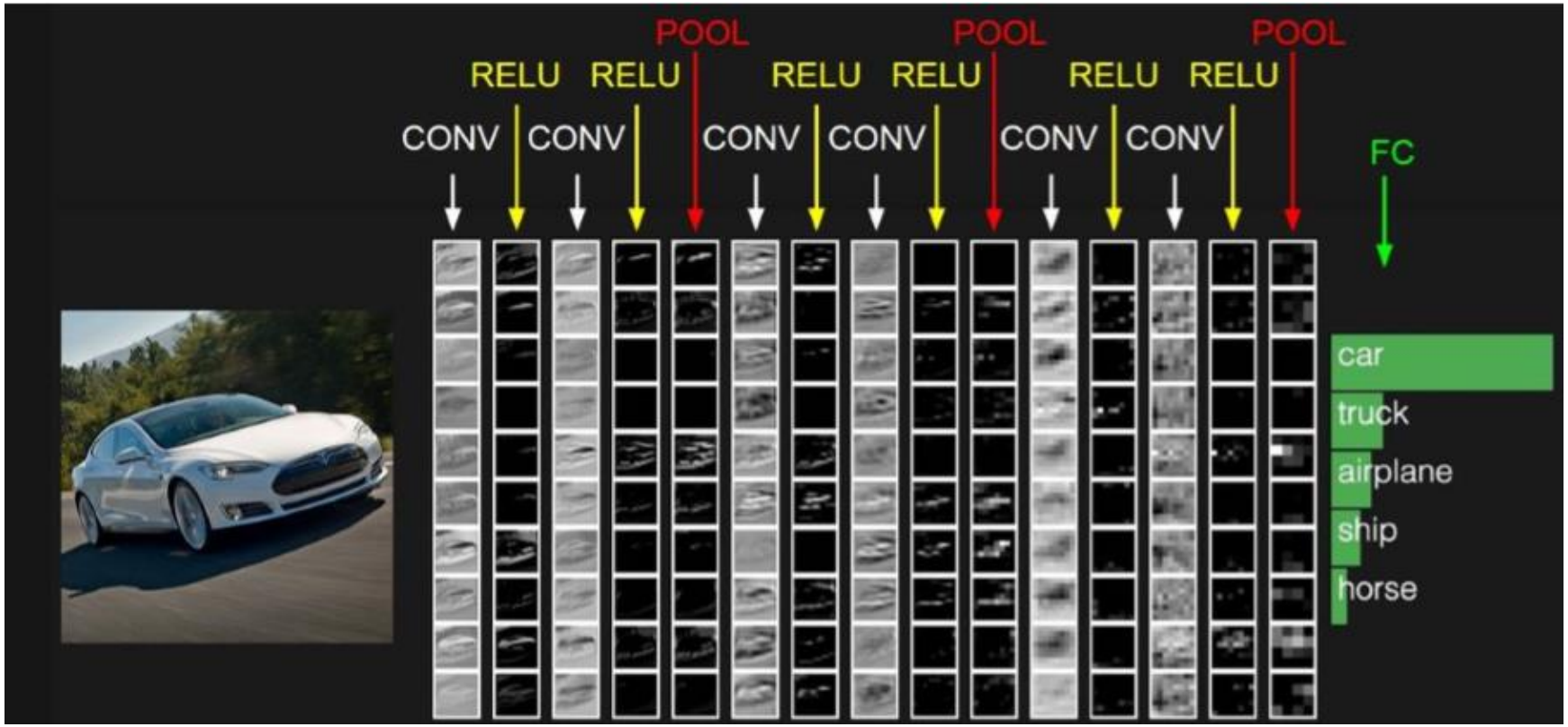
- ✓스마트 교통 데이터는 시계열 데이터와 CCTV 등 영상데이터가 대부분이다.
- ✓교통 이미지 데이터를 수집하고 분류하기 위해서 정확도가 높은 CNN 구현이 필요하다.
- ✓CNN 정확도를 이해하고 의미있는 연구결과를 얻기 위해서 필요한 연구 전략에 대하여 알아본다.

❖ 생각해볼 문제

- ✓CNN 정확도 95%가 주는 의미는 무엇이며, 실제 스마트 교통에 필요한 CNN 데이터와 분석 정확도는 얼마가 적당한가?
- ✓시계열 교통 데이터를 CNN을 처리 할 수 있을까?

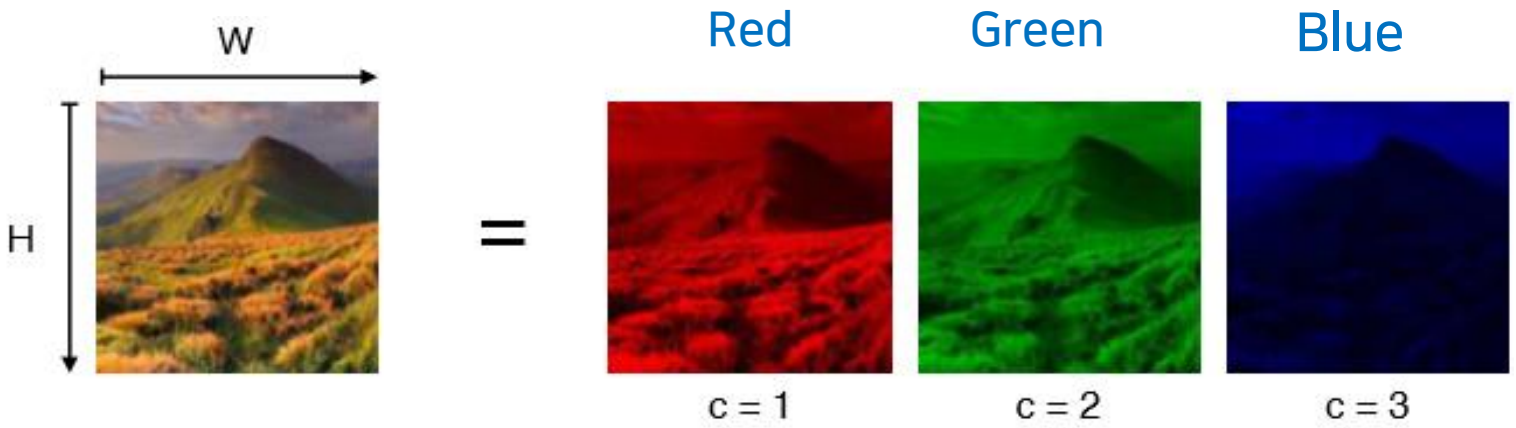
CNN의 특성 학습

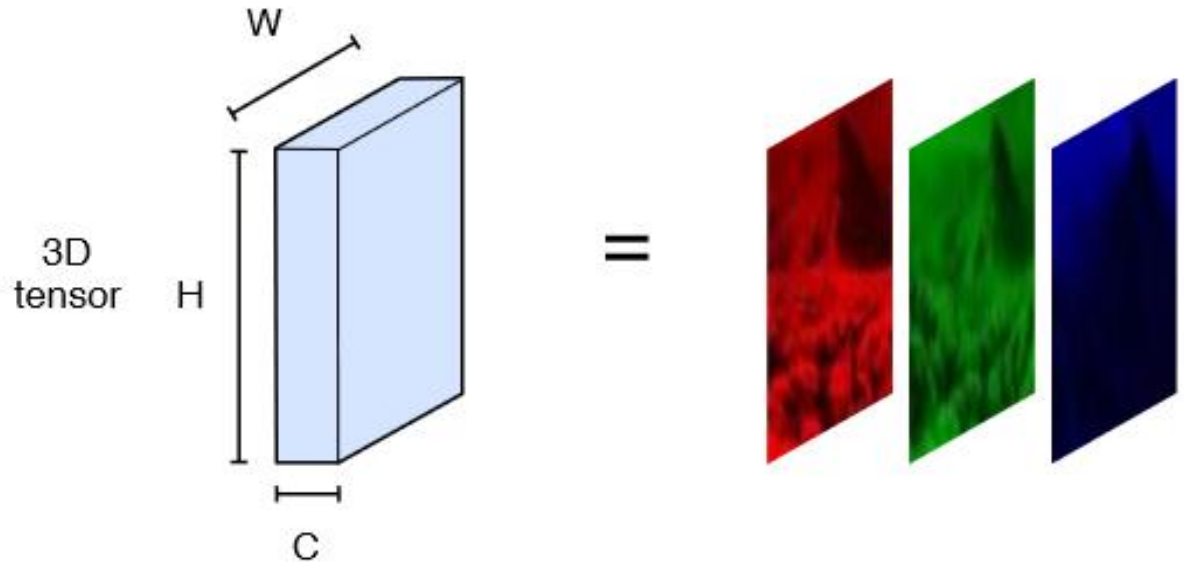
Input → Conv → ReLU → Pooling → ReLU → Conv → ReLU → Pooling → Fully Connected → Output



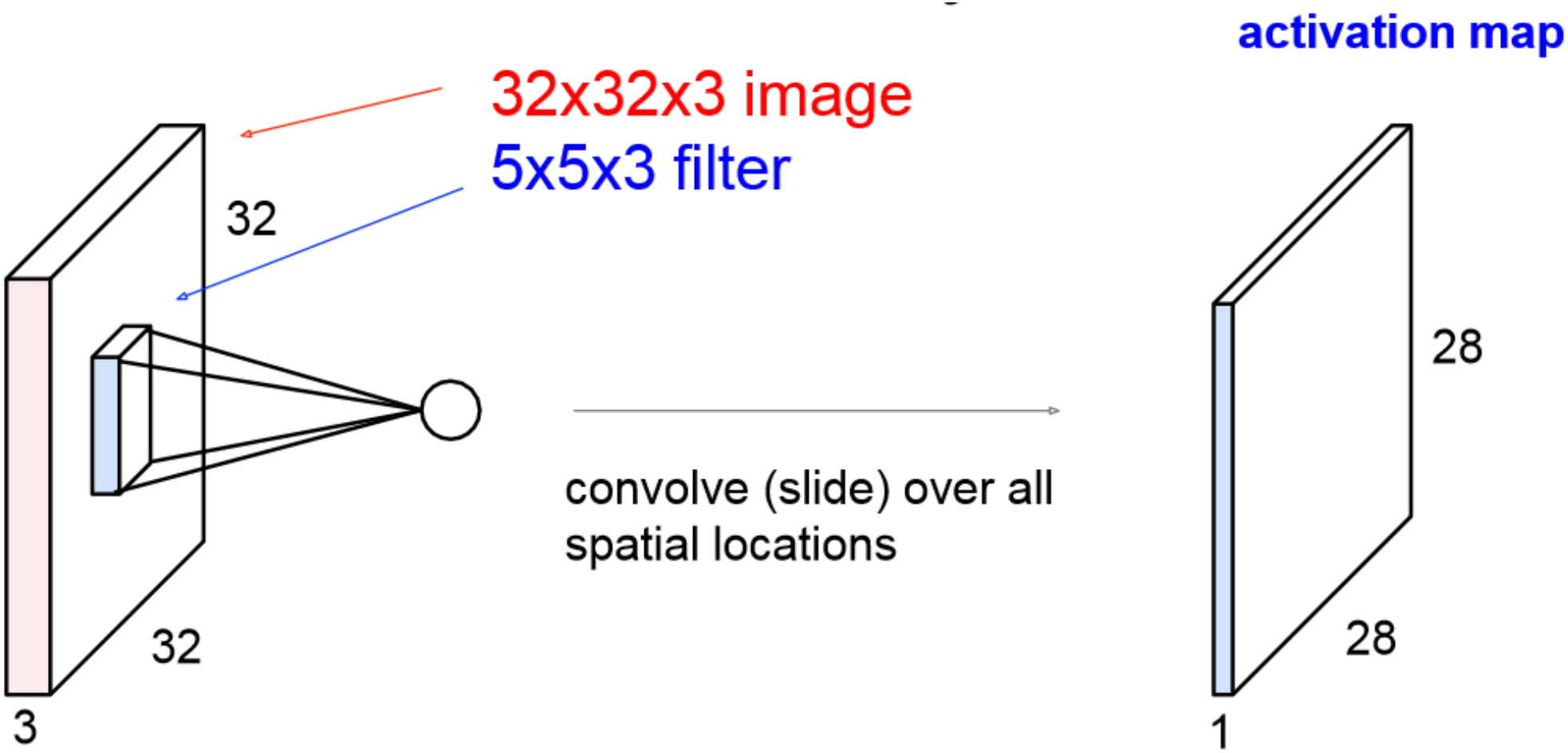
레이블링

칼러 이미지 데이터는 3차원으로 표현





Convolution Layer와 filter

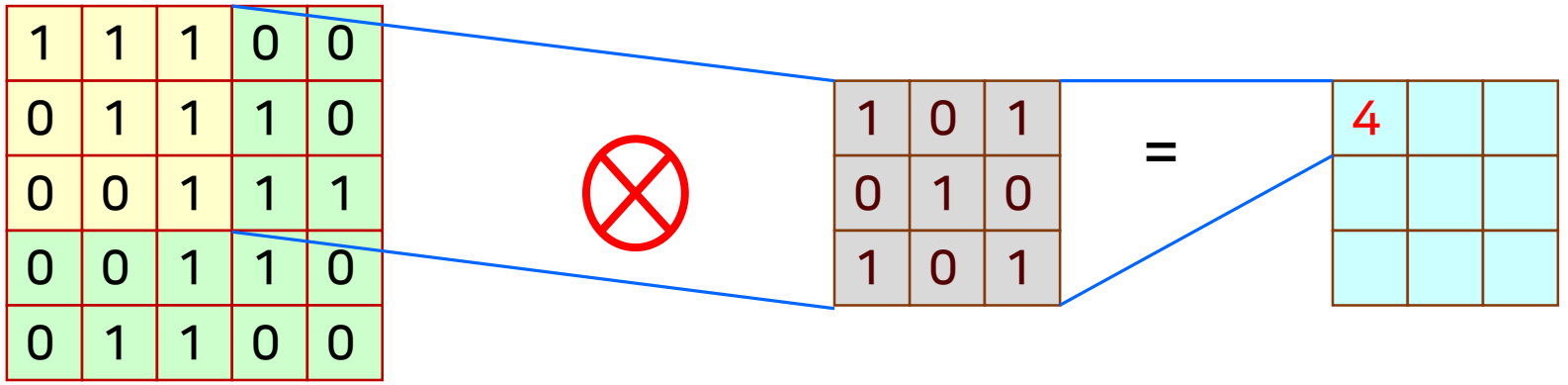


1. 콘볼루션(Convolution) 연산의 이해

Input → Conv → ReLU → Pooling → ReLU → Conv → ReLU → Pooling → Fully Connected → Output

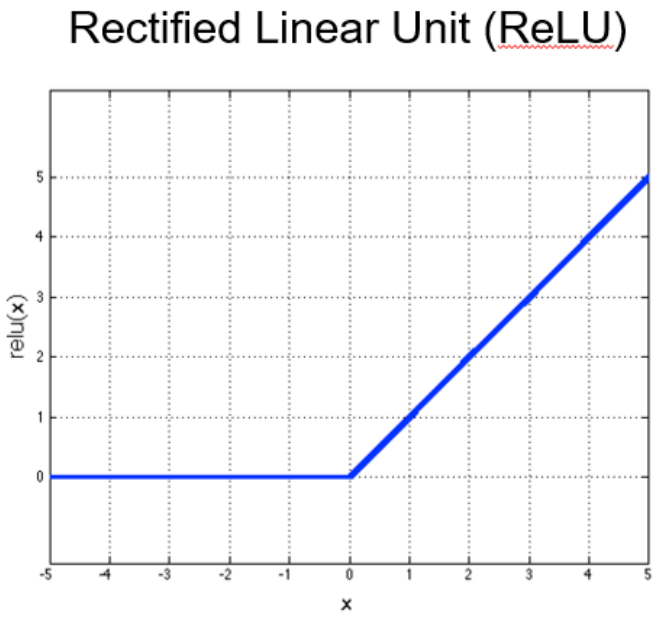
Convolution : 같은 배열끼리 곱하고 합하기

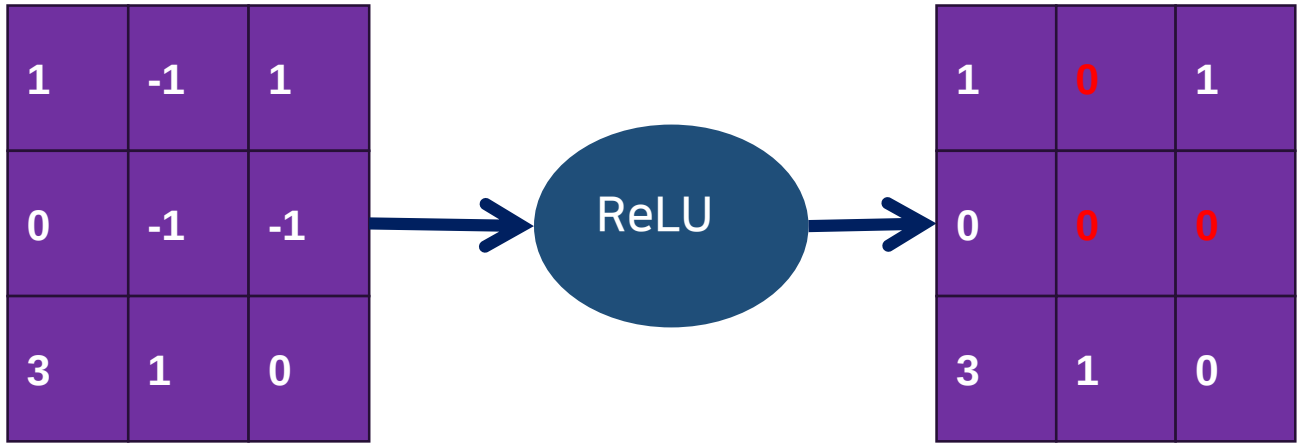
1x1 + 1x0 + 1x1 + 0x0 + 1x1 + 1x0 + 0x1 + 0x0 + 1x1 = 4



2. ReLU 활성함수 Linear chains

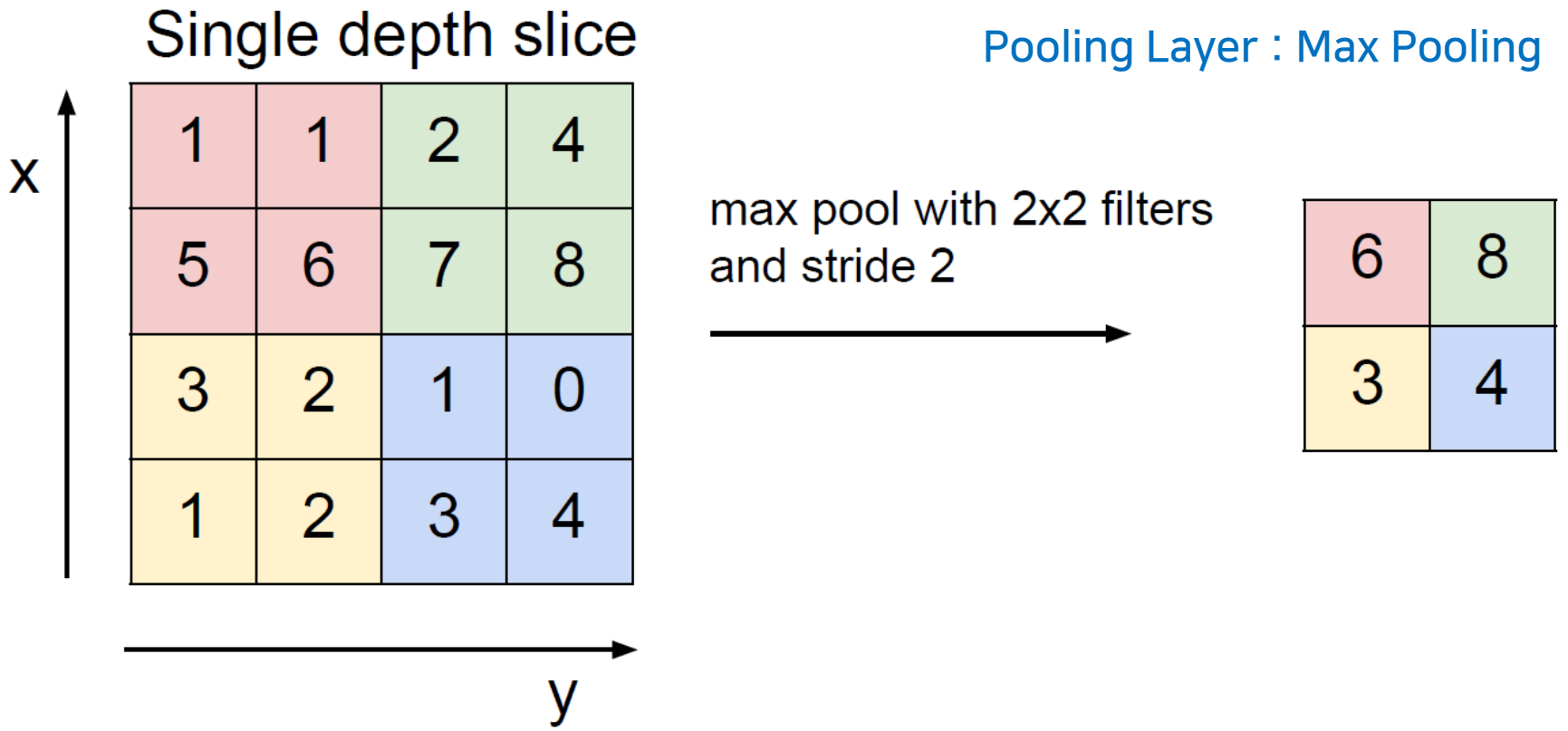
Input → Conv → ReLU → Pooling → ReLU → Conv → ReLU → Pooling → Fully Connected → Output





3. Pooling Layer와 ReLU 활성함수

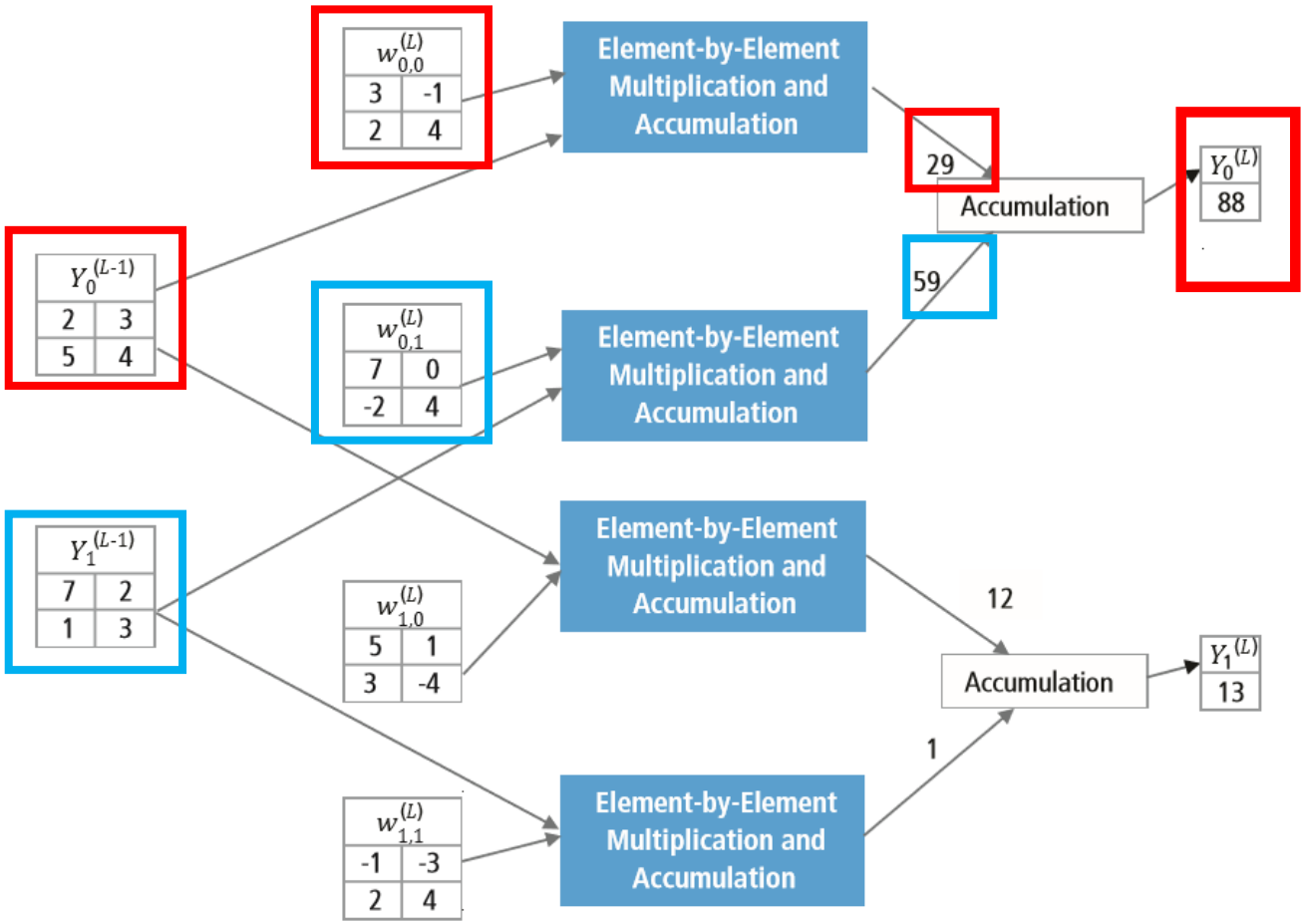
Input → Conv → ReLU → Pooling → ReLU → Conv → ReLU → Pooling → Fully Connected → Output



Pooling Layer : Max Pooling

4. Fully Connected Layer (예제)

Input → Conv → ReLU → Pooling → ReLU → Conv → ReLU → Pooling → Fully Connected → Output



1) 같은 행끼리 곱하고
2x3=6
3*(-1)=-3
5*2=10
4*4=16

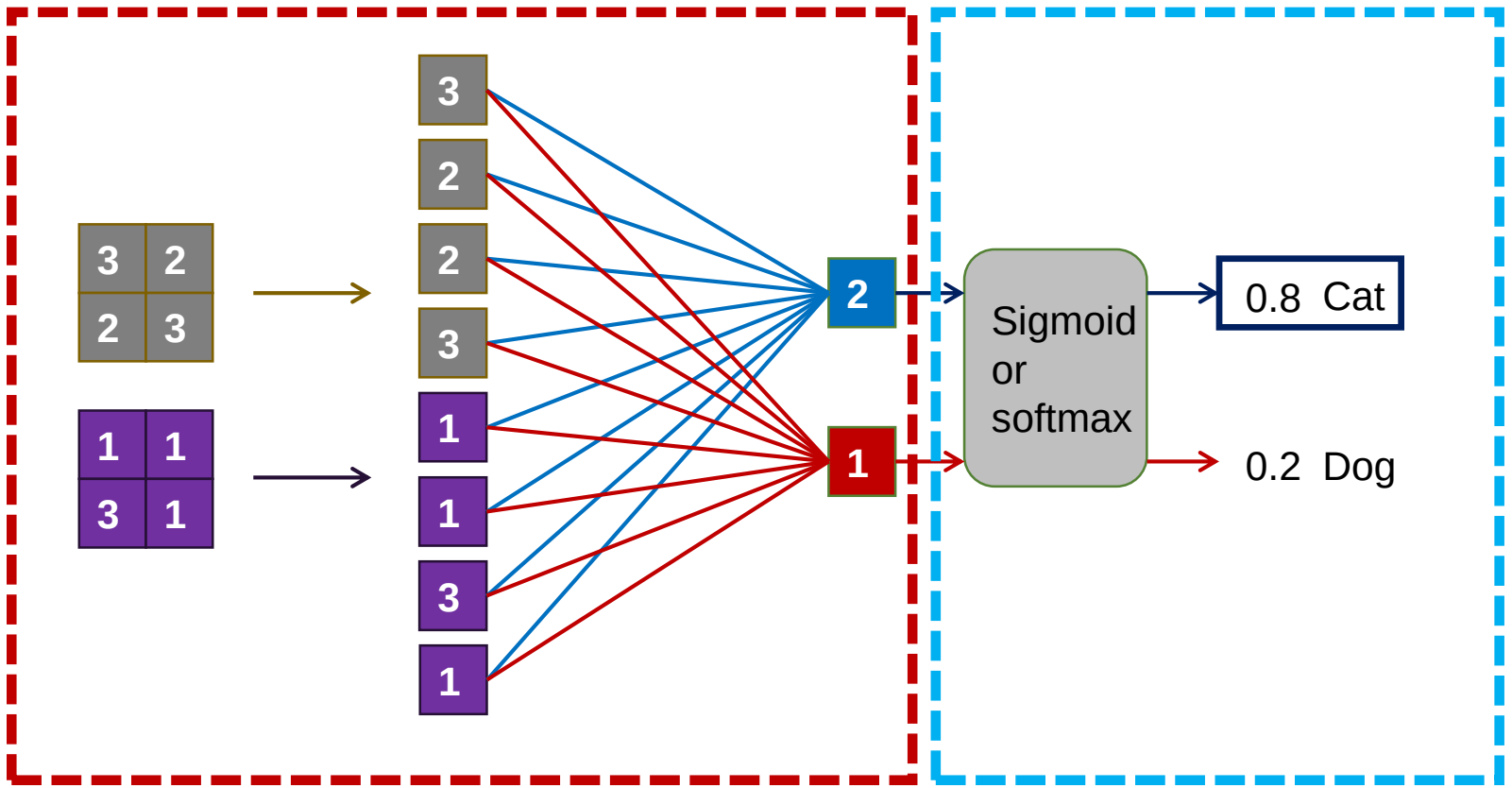
2) 모든 요소를 더하며
6-3+10+16=29

3) 7*7=49
2*0=0
1*(-2)=-2
3*4=12

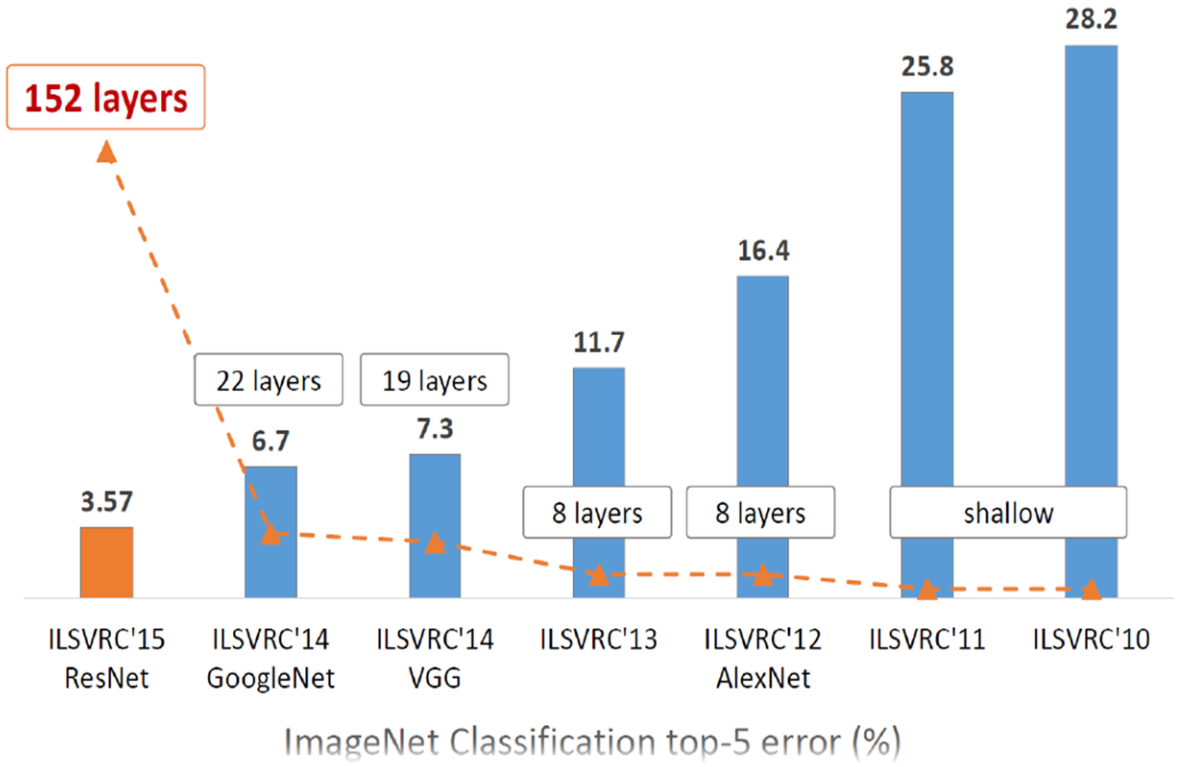
4) 49+0-2+12=59
5) 29+59=88

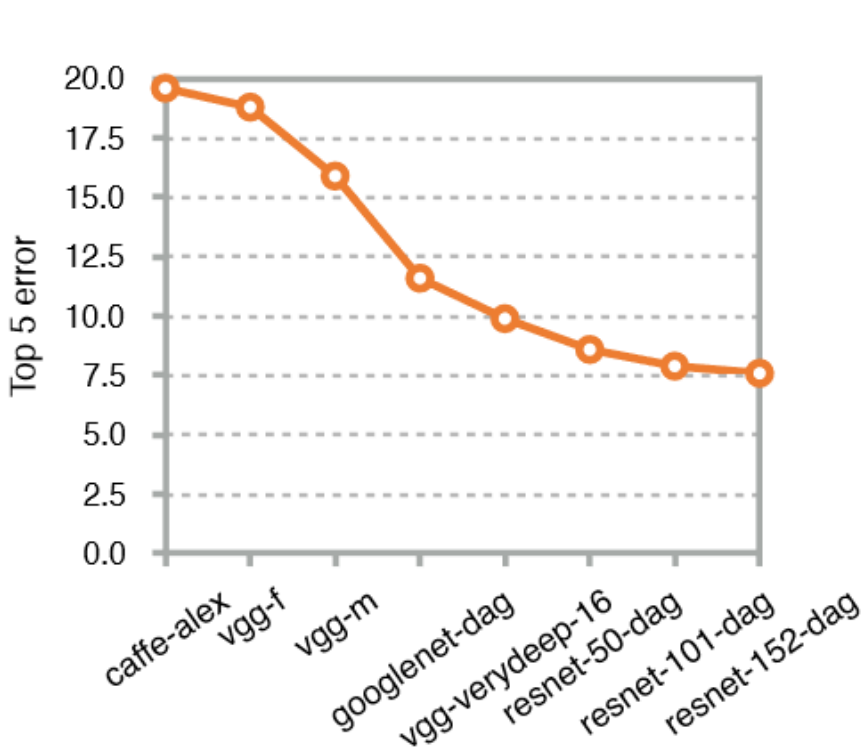
5. Fully-Connected Layer와 Output

Input → Conv → ReLU → Pooling → ReLU → Conv → ReLU → Pooling → Fully Connected → Output

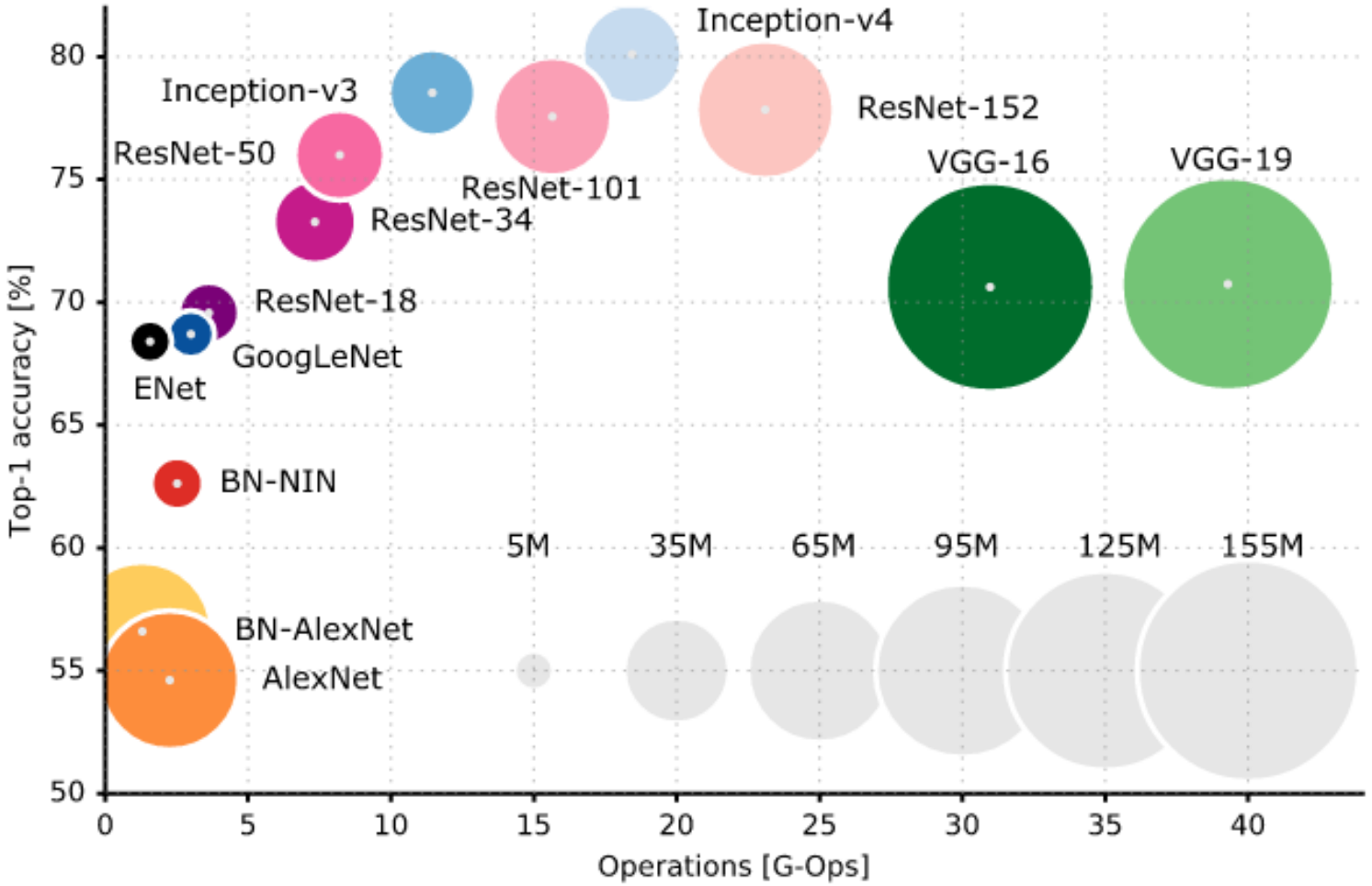


ImageNet 분류의 정확성 및 은닉층 개수 (Deep)





CNN Accuracy vs. efficiency

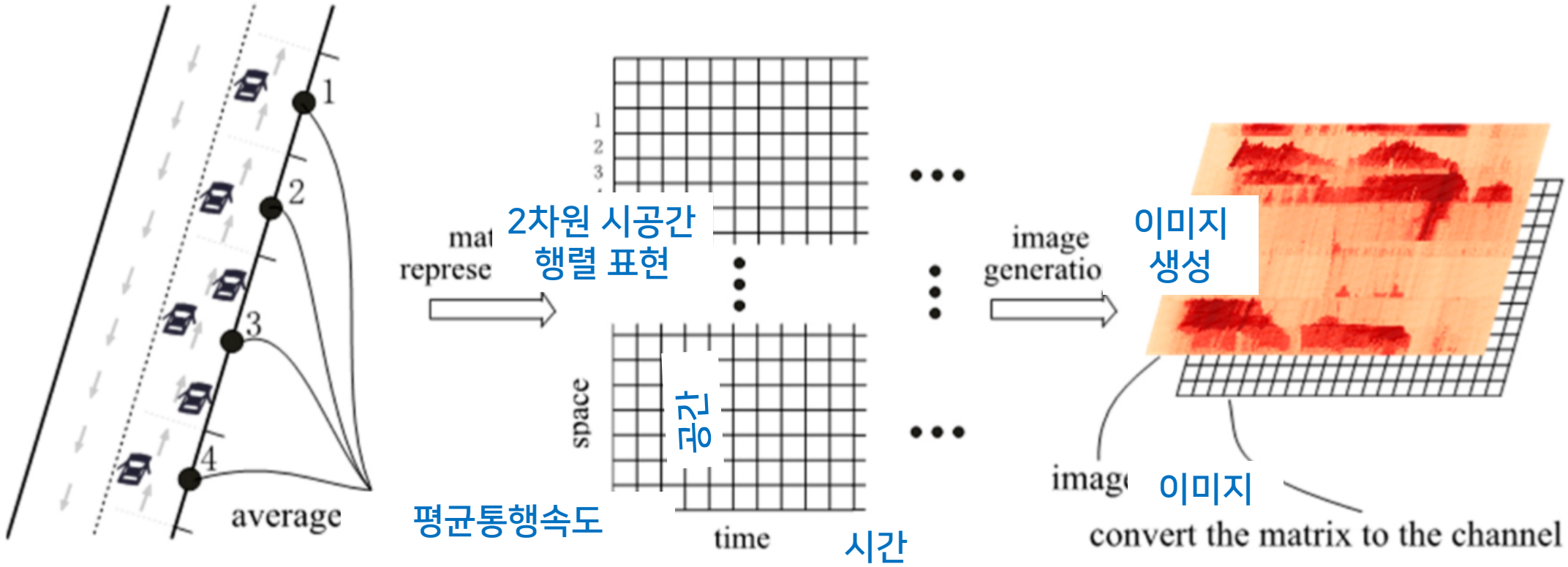


https://culurciello.github.io/tech/2016/06/04/nets.html

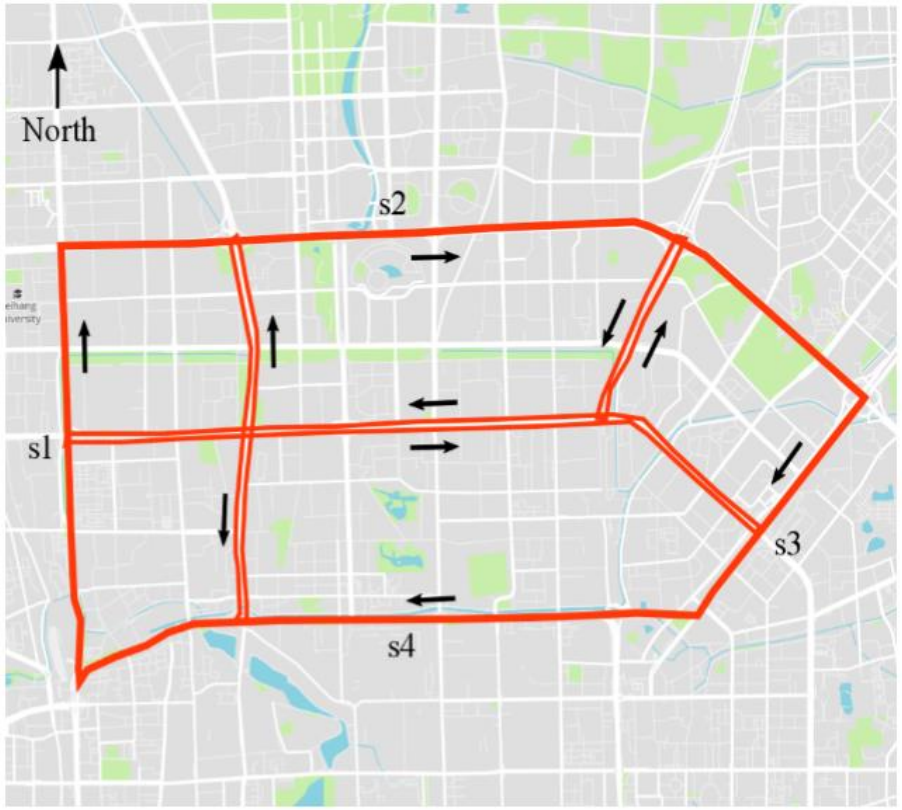
CNN 모델을 위한 교통 데이터 종류

Traffic-to-image conversion on a network

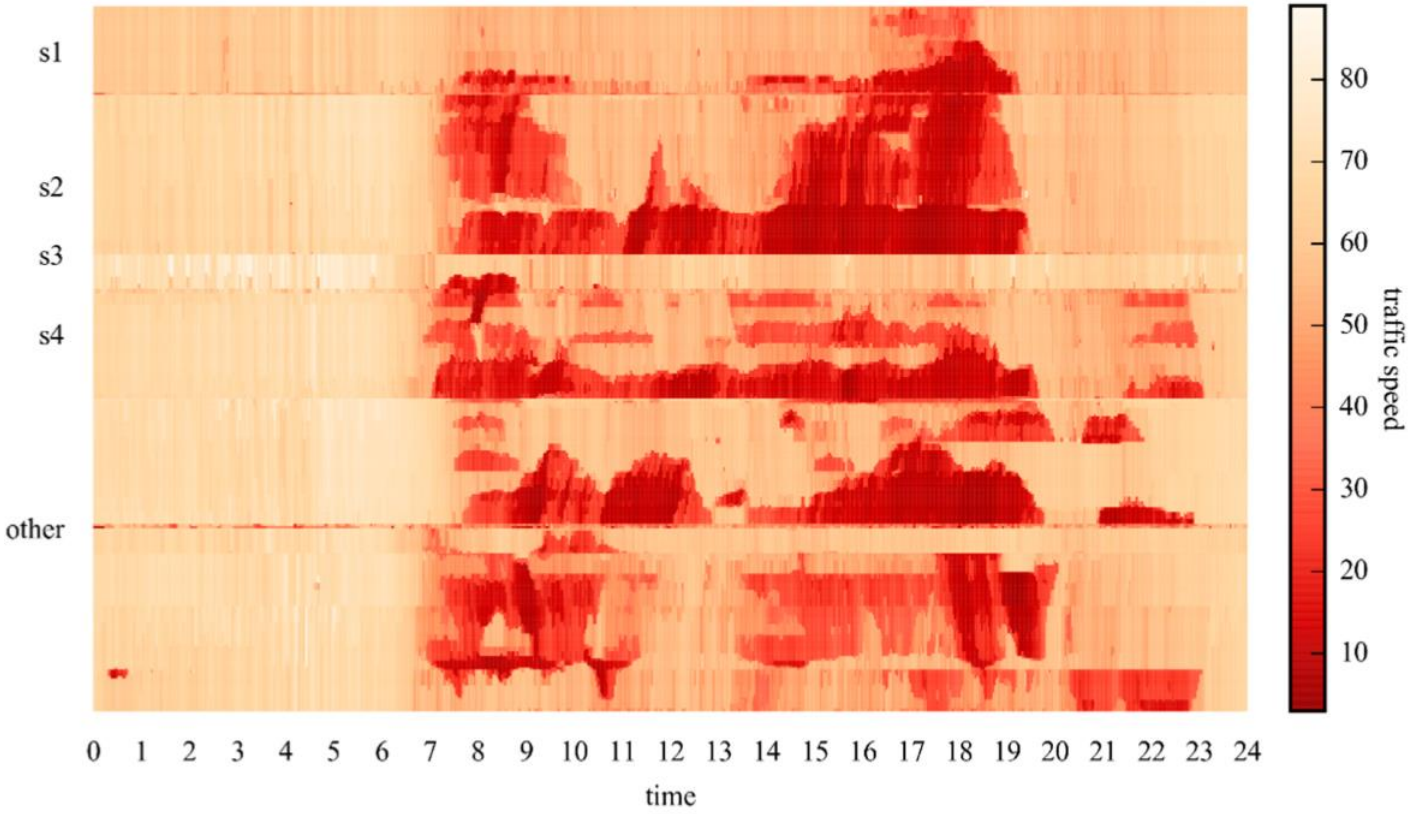
A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction



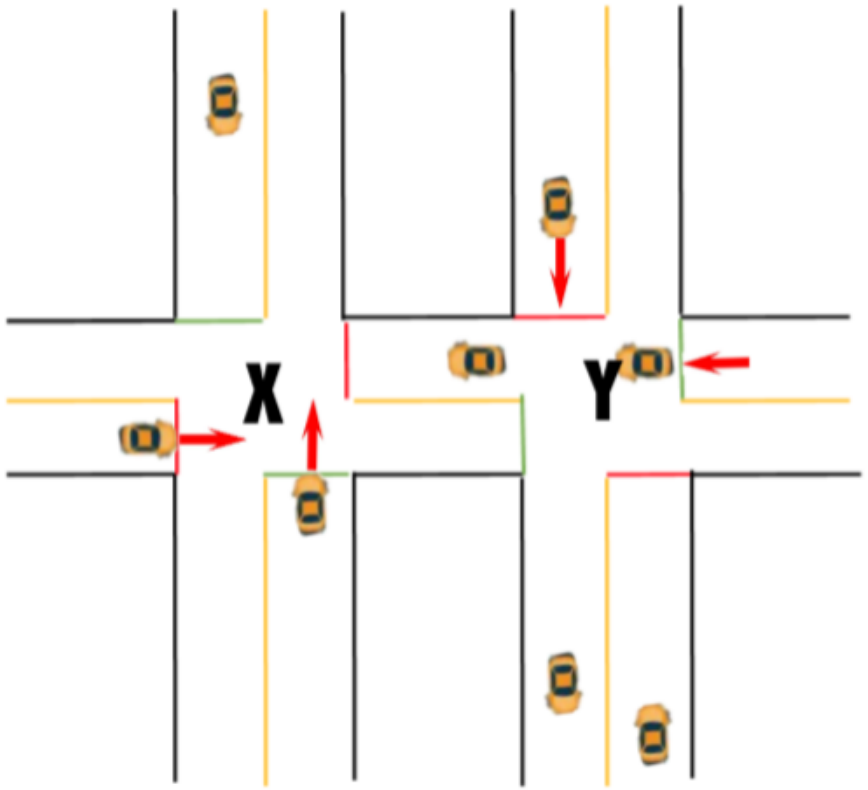
CNN 분석을 위한 교통 데이터 처리 (도심 교통 흐름)



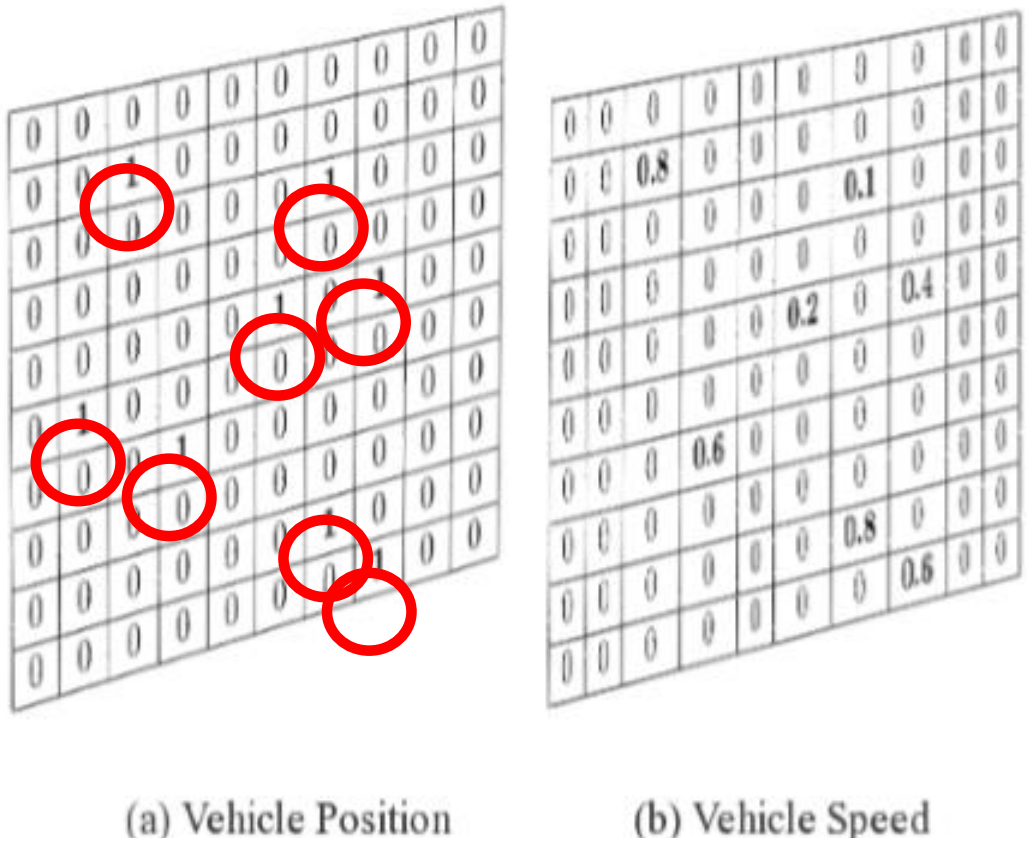
레이블링은 무엇인가?



CNN 분석을 위한 교통 데이터 처리 (교통 신호처리)



레이블링은 무엇인가?



(a) Vehicle Position

(b) Vehicle Speed

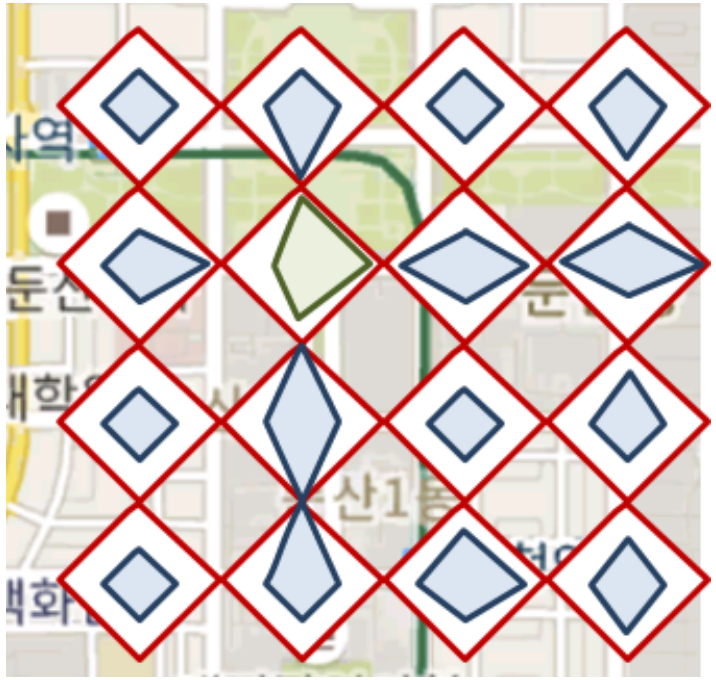
차량들 위치

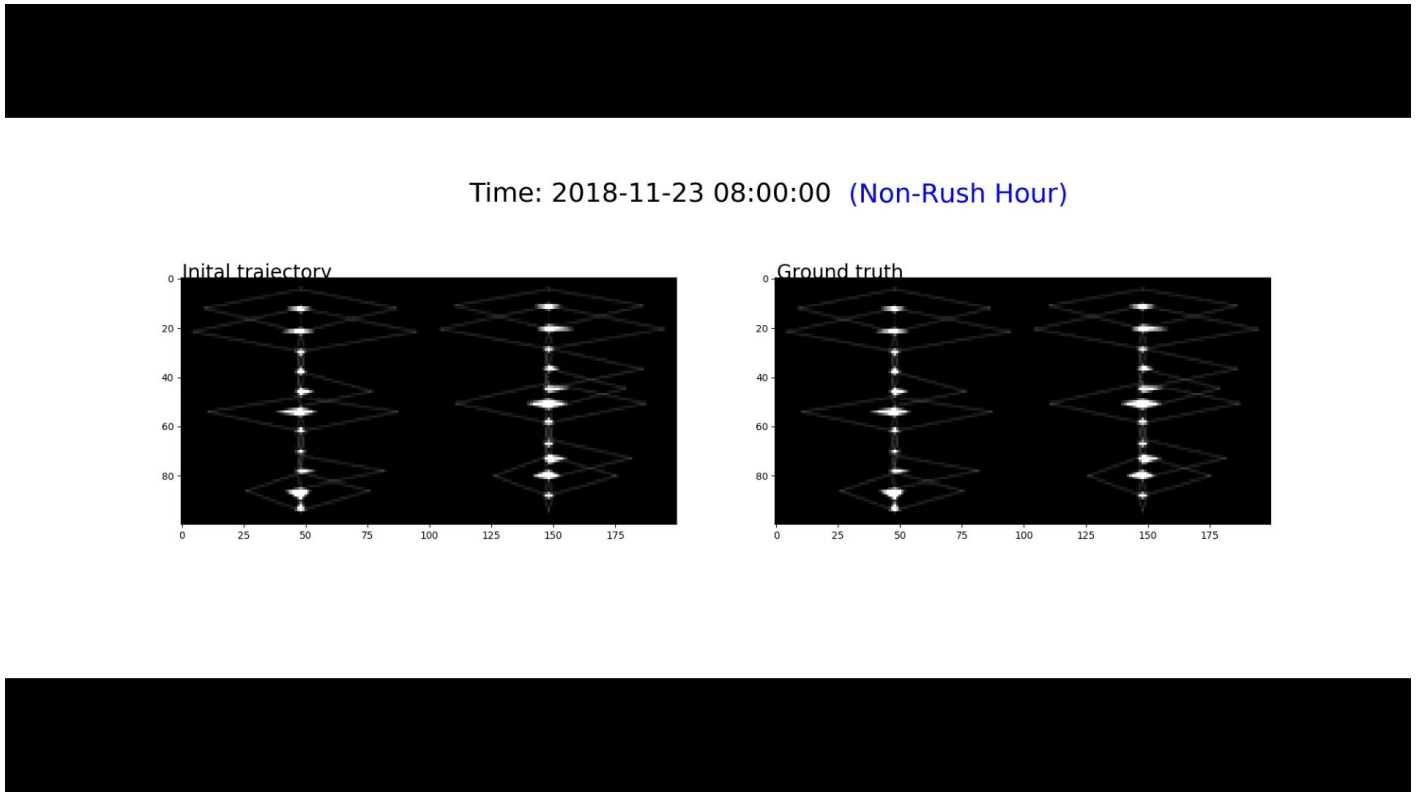
차량들 속도

CNN 분석을 위한 교통 데이터 처리 (RSE 데이터)

❖ 인공지능 : CNN의 Top 1 Accuracy

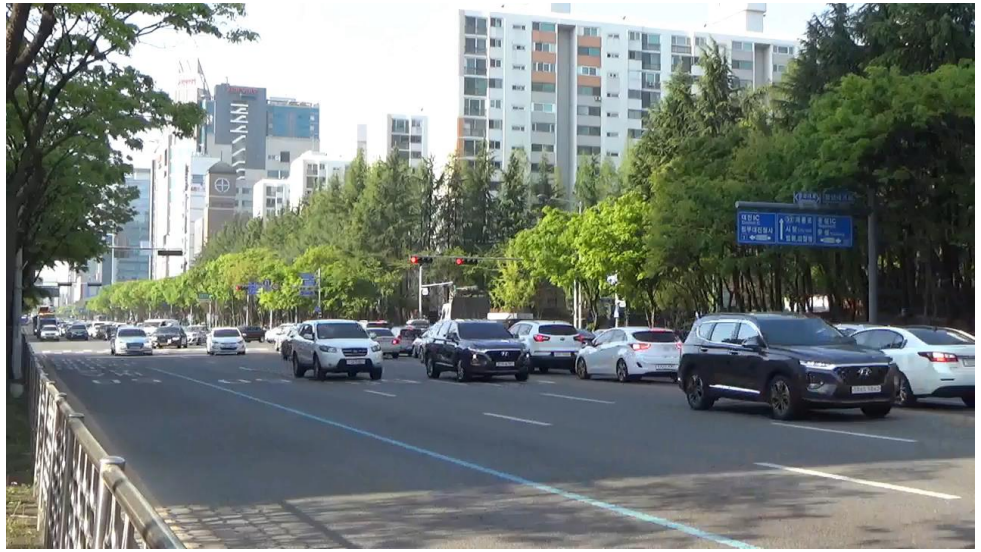
- 13개 교차로를 1개의 이미지(교통량)로 변환
- 딥러닝 CCN을 적용하여 교통 패턴 분류한 기술

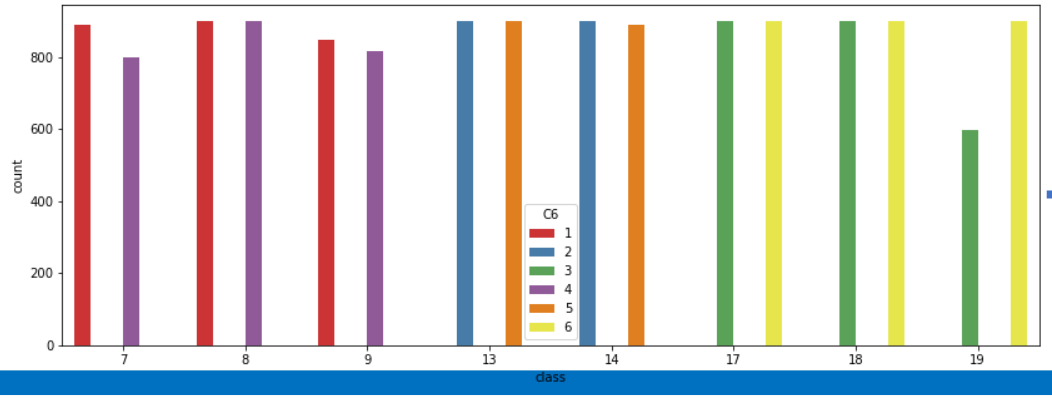




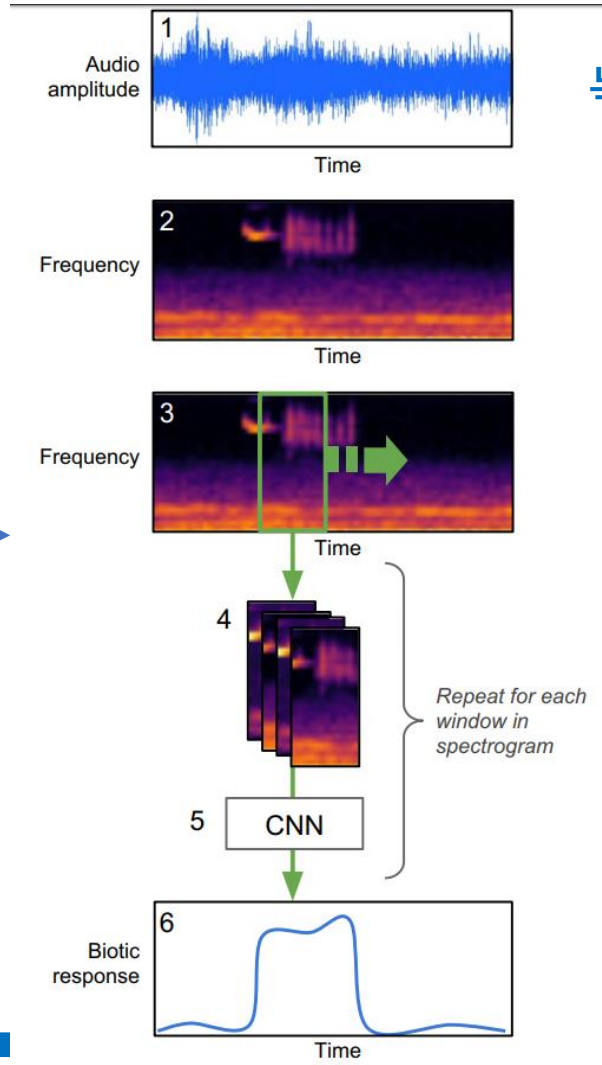
CNN 분석을 위한 교통 데이터 처리 (교통 소음)

- 소음 빅데이터 자체 수집 : DJ_TrafficSound14K (대덕대로)





- 딥러닝 CNN : Mel Spectrogram



녹음된 음원 추출

Day3-Lec2: 강의
Object Detection

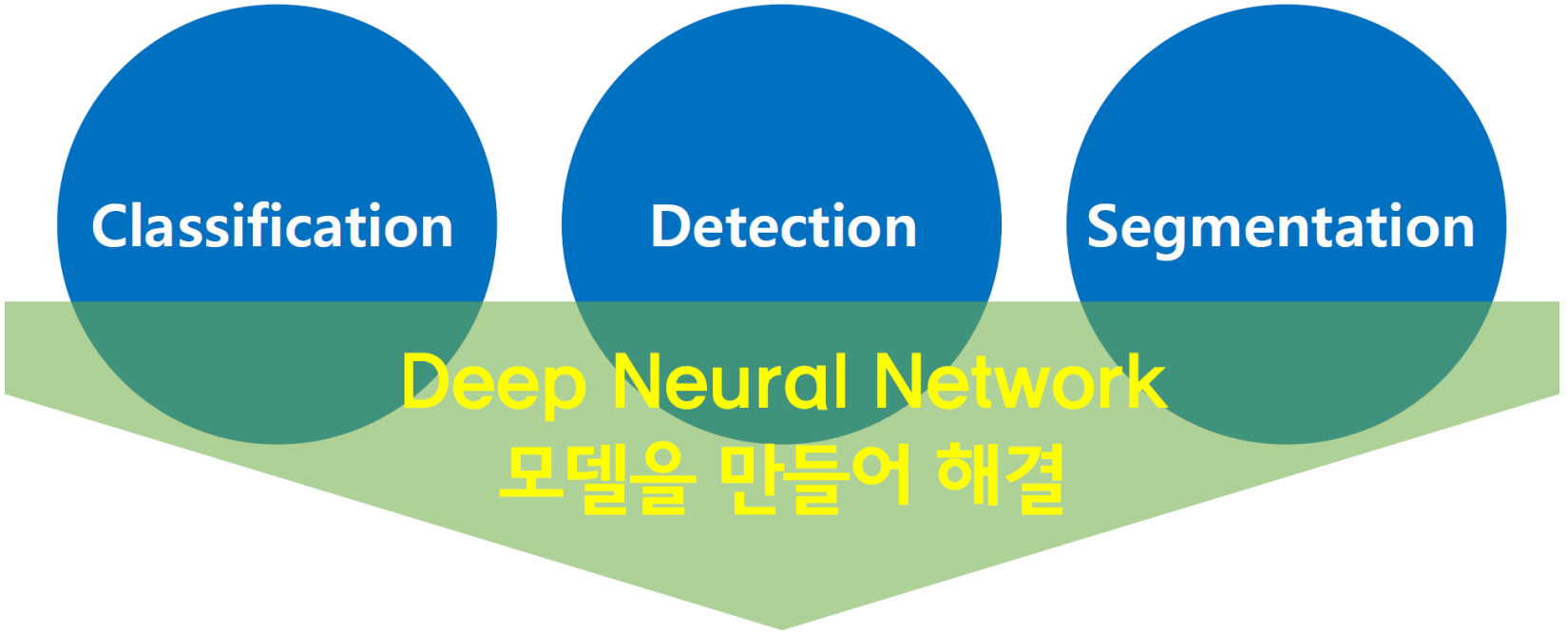
컴퓨터 비전 (Computer Vision)

❖ 컴퓨터에게 인간의 시각 능력을 갖게 하는 것

카메라를 통해서 본 사물 인식

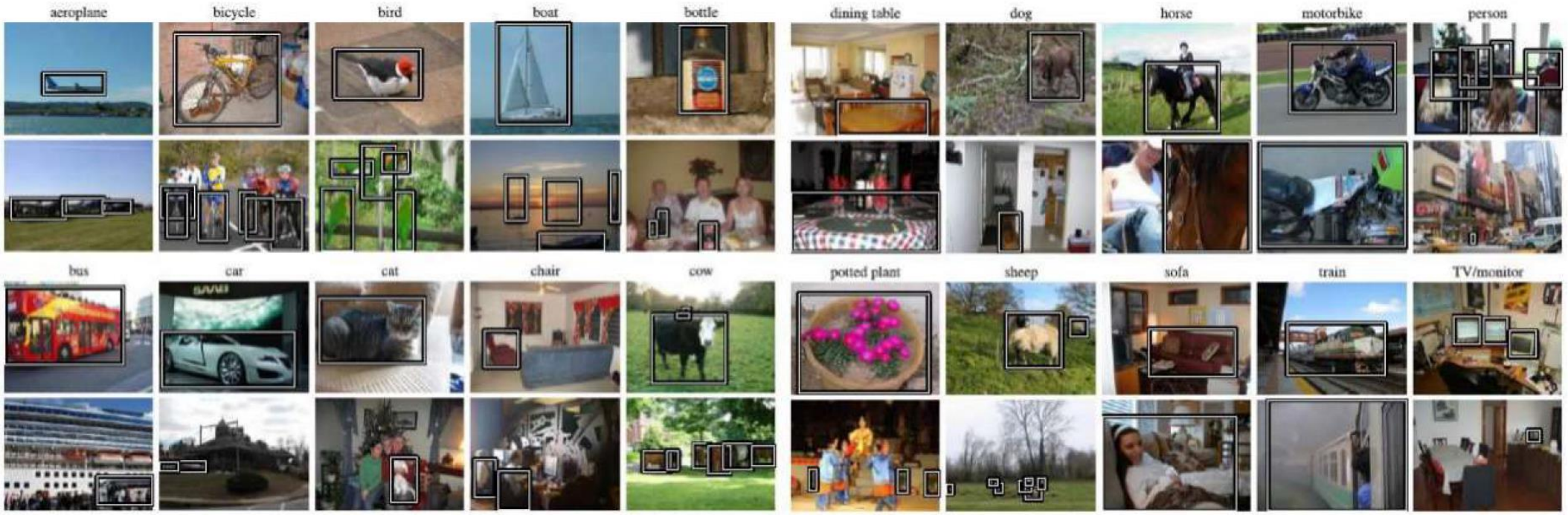
Image Recognition 문제

❖ Image Recogntition 문제



분류(Classifiction)

- 이미지 안에 어느 특정한 클래스가 포함되어 있는지 여부를 분류하는 모델을 만드는 것
  - ✓클래스(Class)란 분류대상이 되는 객체
- PASCAL VOC Challenge에서 다루는 Class는 20 가지임

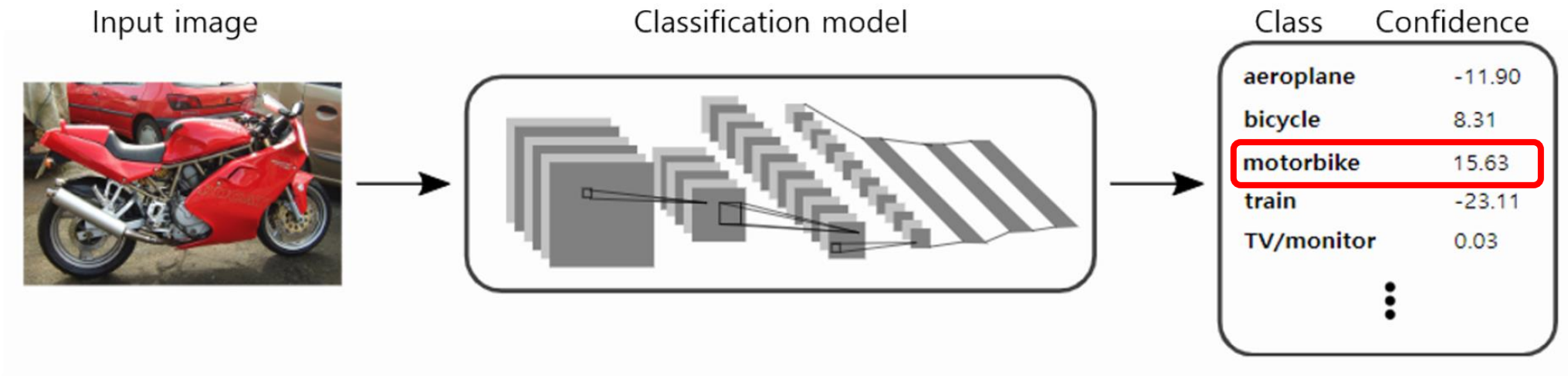


Classifiction_모델의 결과물

❖ 정의

✓존재할 가능성을 신뢰도 점수로 표현함

✓있다/없다(X)



오토바이 신뢰도

단일 사물 분류의 신뢰도 점수

- 20가지 클래스 중에서 신뢰도 점수가 가장 큰 클래스를 선정
  - ✓“주어진 이미지에서 고양이(cat)이 포함되어 있을 것이다 " 로 해석

Class	Confidence	Class	Confidence
aeroplane	-10.22	dining table	-5.98
bicycle	-1.92	dog	29.99
bird	-21.56	horse	8.89
boat	-18.77	motorbike	-20.01
bottle	0.56	person	7.67
bus	-1.09	potted plant	-1.23
car	1.11	sheep	15.23
cat	37.26	sofa	2.89
chair	5.60	train	-23.11
cow	7.27	TV/monitor	0.03

복수 사물의 신뢰도

❖ 각 클래스별 신뢰도 점수가 한계값 보다 크면 클래스를 선정함

✓각 클래스 마다 한계점(Threshold)을 미리 설정함.

✓“주어진 이미지에 cow와 person이 포함되어 있을 것이다 " 로 해석함.

Class(Threshold)	Confidence	Class(Threshold)	Confidence
aeroplane(13.72)	-1.99	dining table(7.76)	-10.11
bicycle(10.66)	2.21	dog(13.21)	-6.54
bird(17.58)	-3.14	horse(14.44)	13.21
boat(16.70)	-33.09	motorbike(8.97)	4.44
bottle(8.11)	-12.13	person(8.01)	11.33
bus(10.11)	5.72	potted plant(21.31)	-10.10
car(15.61)	9.22	sheep(17.17)	16.66
cat(15.09)	0.98	sofa(12.34)	-1.99
chair(11.22)	-8.90	train(16.06)	2.21
cow(9.98)	27.38	TV/monitor(11.99)	-29.34

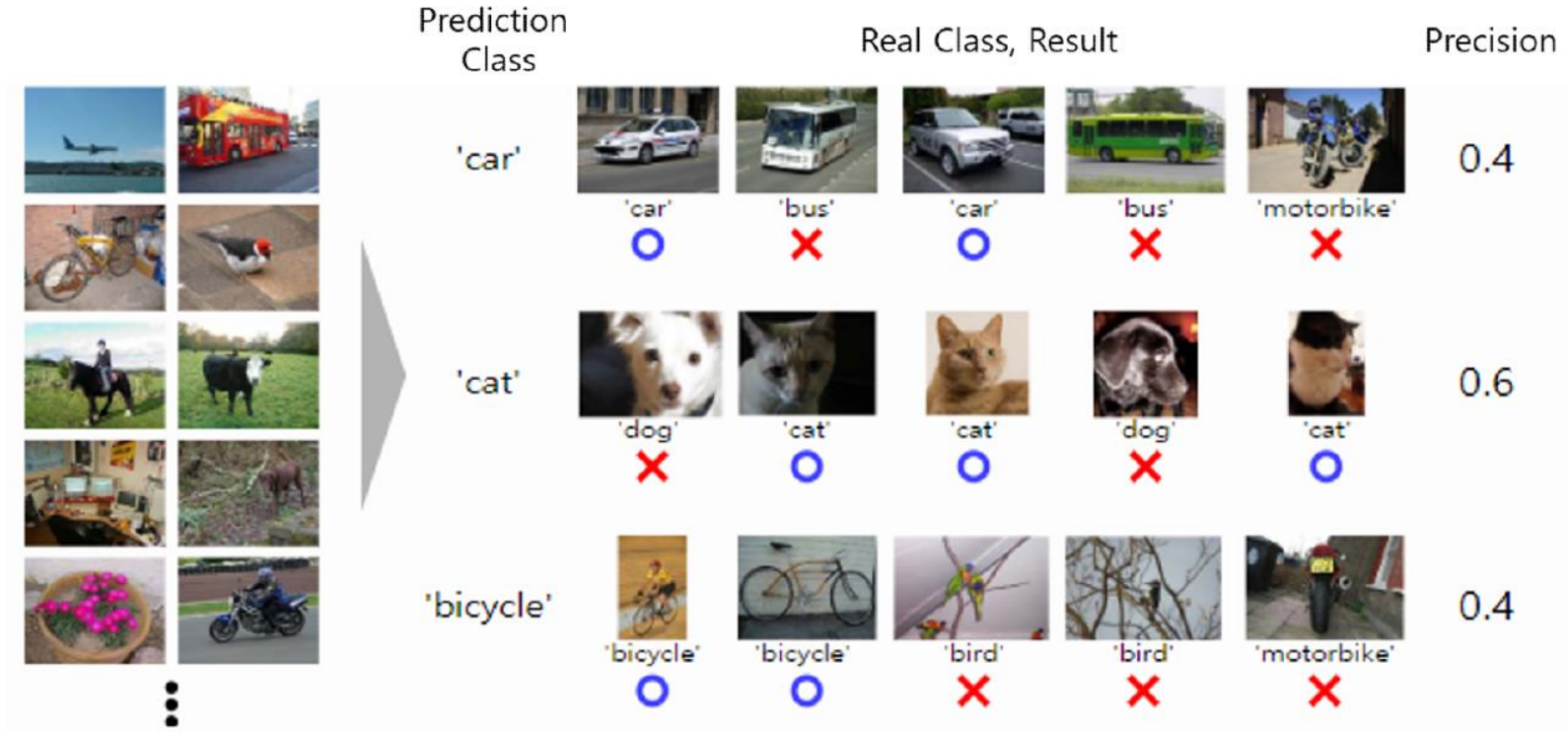
단일 사물의 성능 평가 (정확도)

$$\text{Accuracy} = \frac{\text{올바르게 분류한 이미지 수}}{\text{전체 이미지 수}} = \frac{7}{10} = 0.7(70\%)$$

	Prediction Class	Real Class	Result		Prediction Class	Real Class	Result
	'aeroplane'	'aeroplane'	O		'bus'	'bus'	O
	'car'	'bicycle'	X		'bird'	'bird'	O
	'horse'	'horse'	O		'cow'	'cow'	O
	'train'	'TV/monitor'	X		'dog'	'dog'	O
	'potted plant'	'potted plant'	O		'bicycle'	'motorbike'	X

복수 사물의 성능 평가(정밀도)

$$\text{Class c의 Precision} = \frac{\text{올바르게 분류한 } Class\ C \text{ 이미지 수}}{Class\ C \text{일 것으로 예측한 이미지 수}} \qquad \text{평균 Precision} = \frac{1}{C}\sum_{c=1}^{C}(Class\ C\text{의 } precision)$$

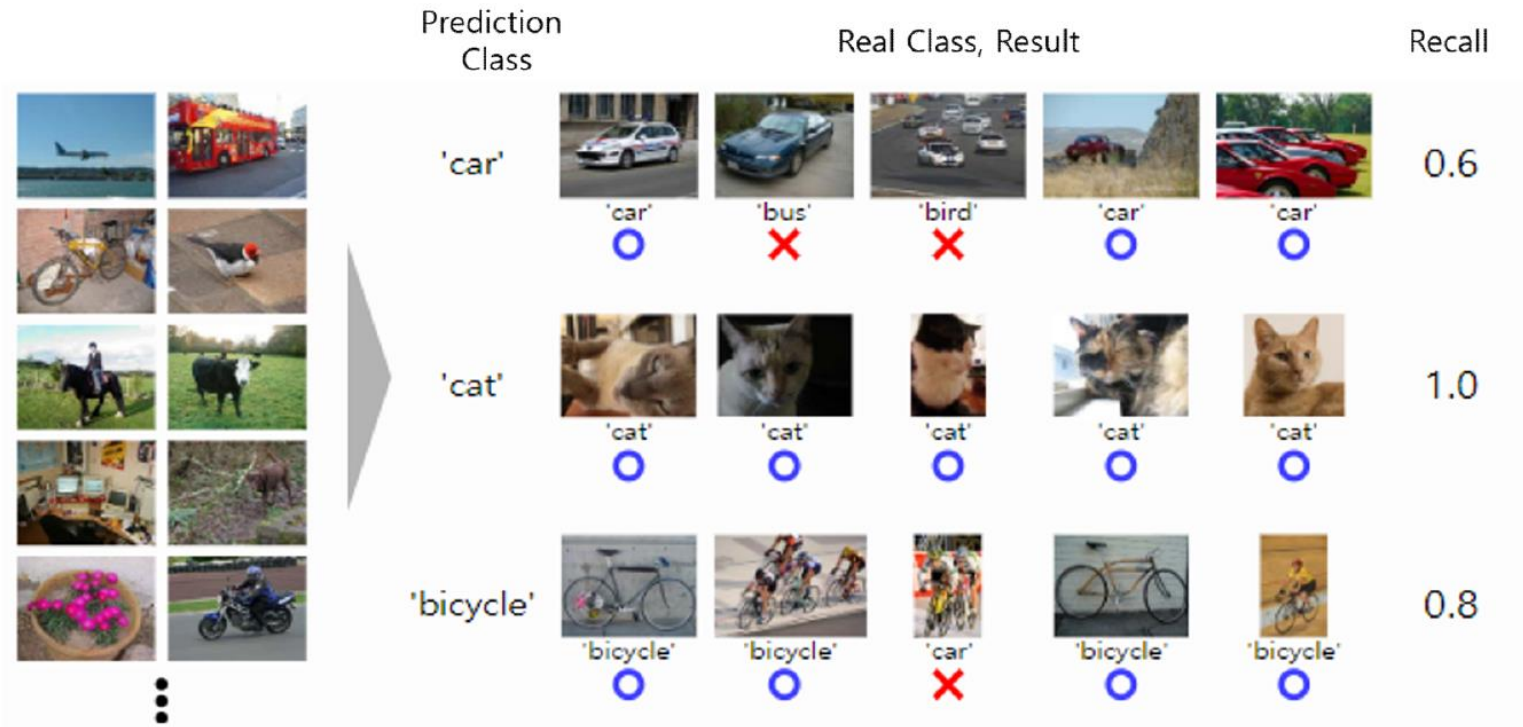


평균 정밀도: $(0.4 + 0.6 + 0.4)/3 = 0.47(47\%)$

복수 사물의 성능 평가 (재현율,Recall)

$$\text{Class c의 Recall} = \frac{\text{올바르게 분류한 } Class\ C \text{ 이미지 수}}{\text{전체 } Class\ C \text{ 이미지 수}}$$

$$\text{평균 Recall} = \frac{1}{C}\sum_{c=1}^{C}(Class\ C\text{의 } Recall)$$

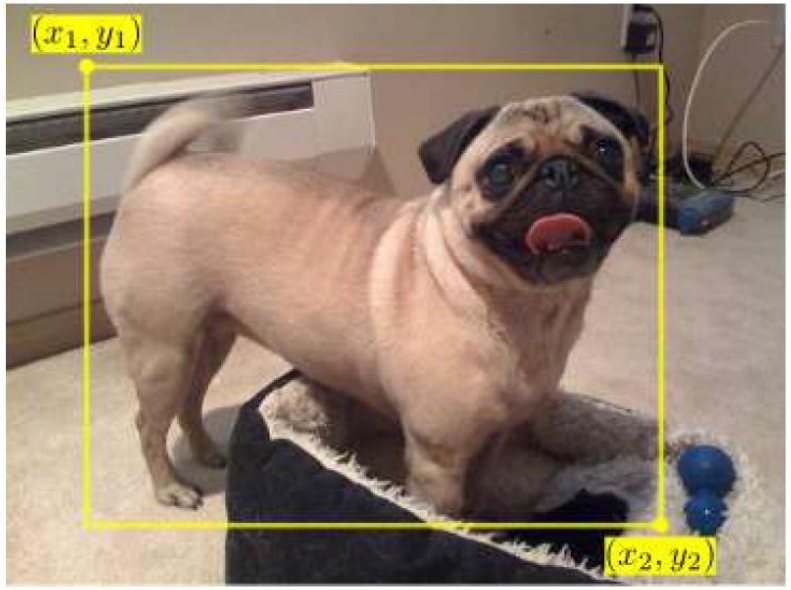


$(0.6 + 1.0 + 0.8)/3 = 0.8(80\%)$ 평균 재현율:

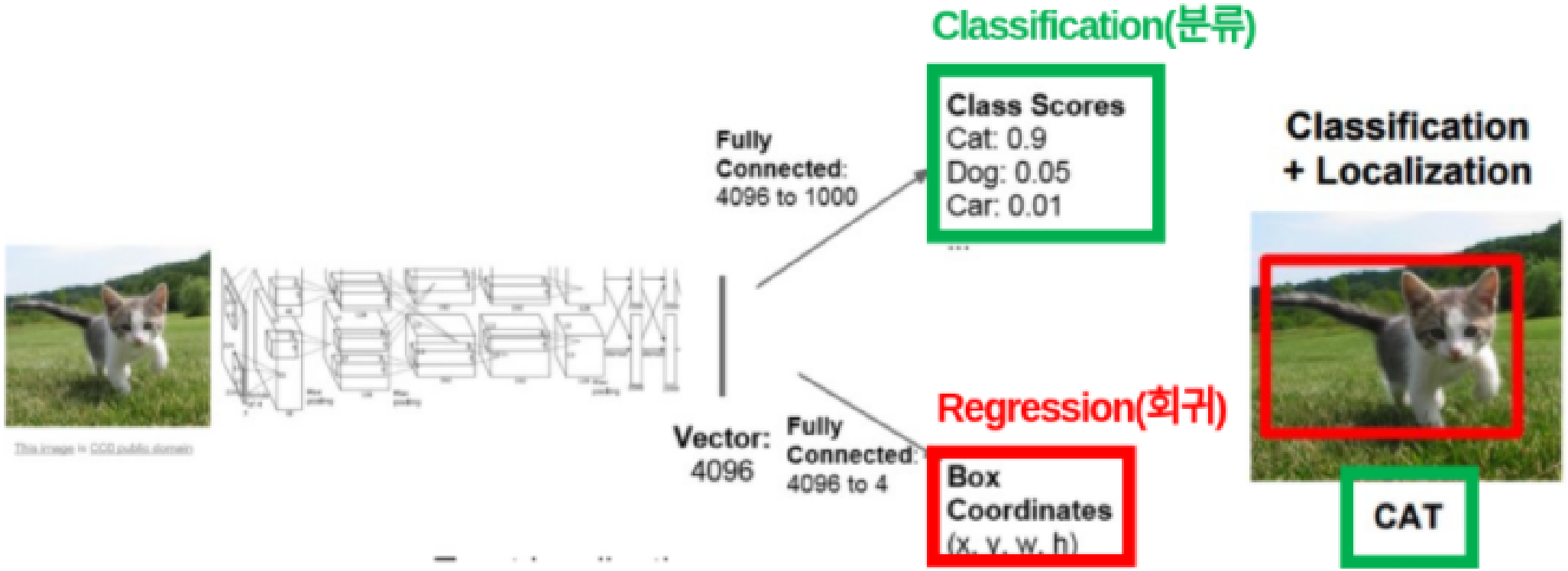
Object Detection (객체 검출)_정의

- 객체 검출은 입력 영상이 주어질 때, 영상 내에 존재하는 모든 클래스에 대하여 분류와 지역화를 수행하여 박스형태(Bounding Box)로 검출하는 모델을 만드는 것
  - ✓Localization(지역화)는 바운딩 박스를 찾는 회귀(Regression)이고
  - ✓Classification은 바운딩 박스 내 물체가 무엇인지 분류하는 문제이다.
  - ✓"주어진 이미지의 바운딩 박스 안에 dog가 포함되어 있을 가능성을 신뢰도 점수로 표현 " 한다.

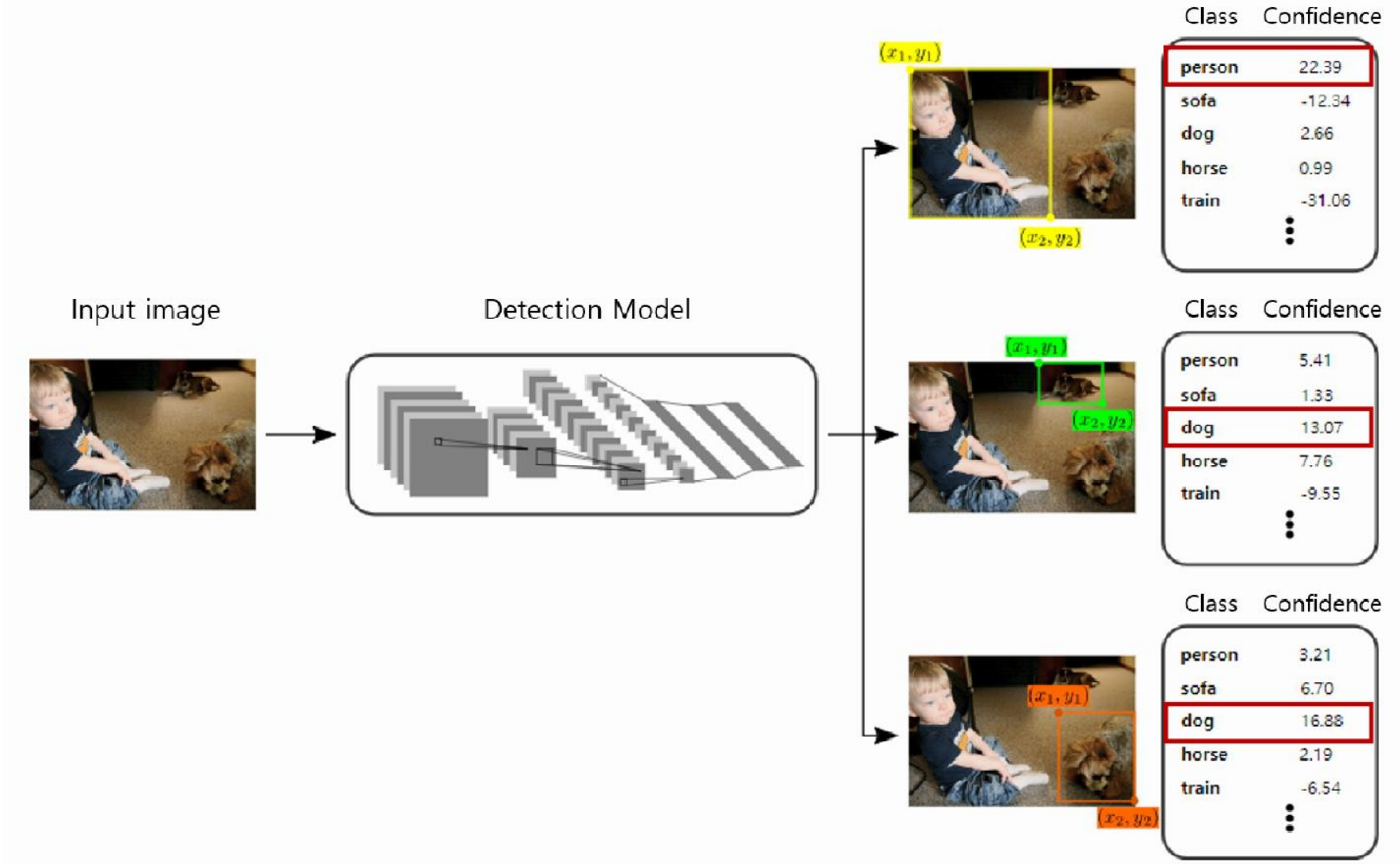
Object Detection = Classifiction + Localization



객체탐지_복수 객체 검출 모델의 결과물 예시



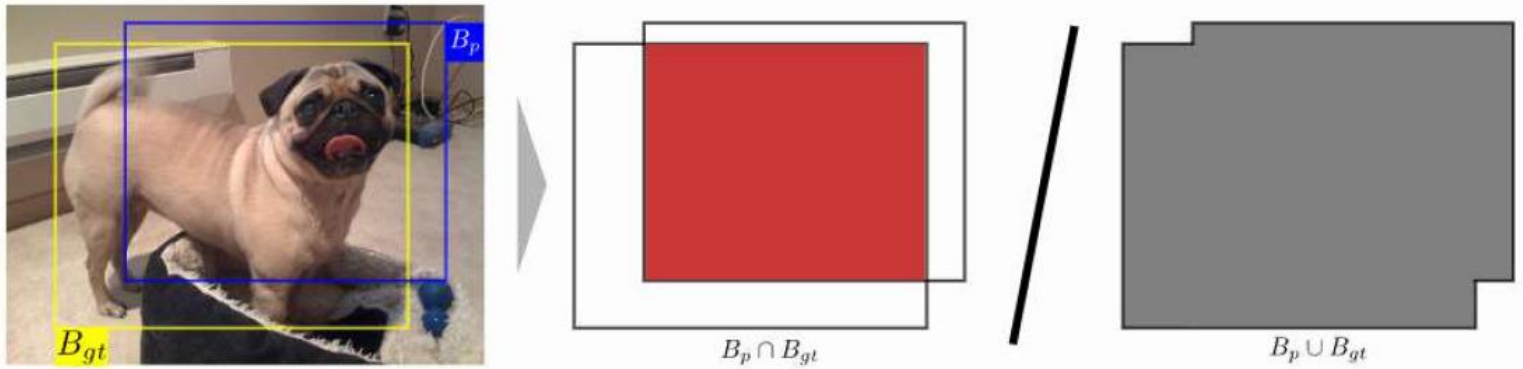
객체탐지_복수 객체 검출 모델의 결과물 예시



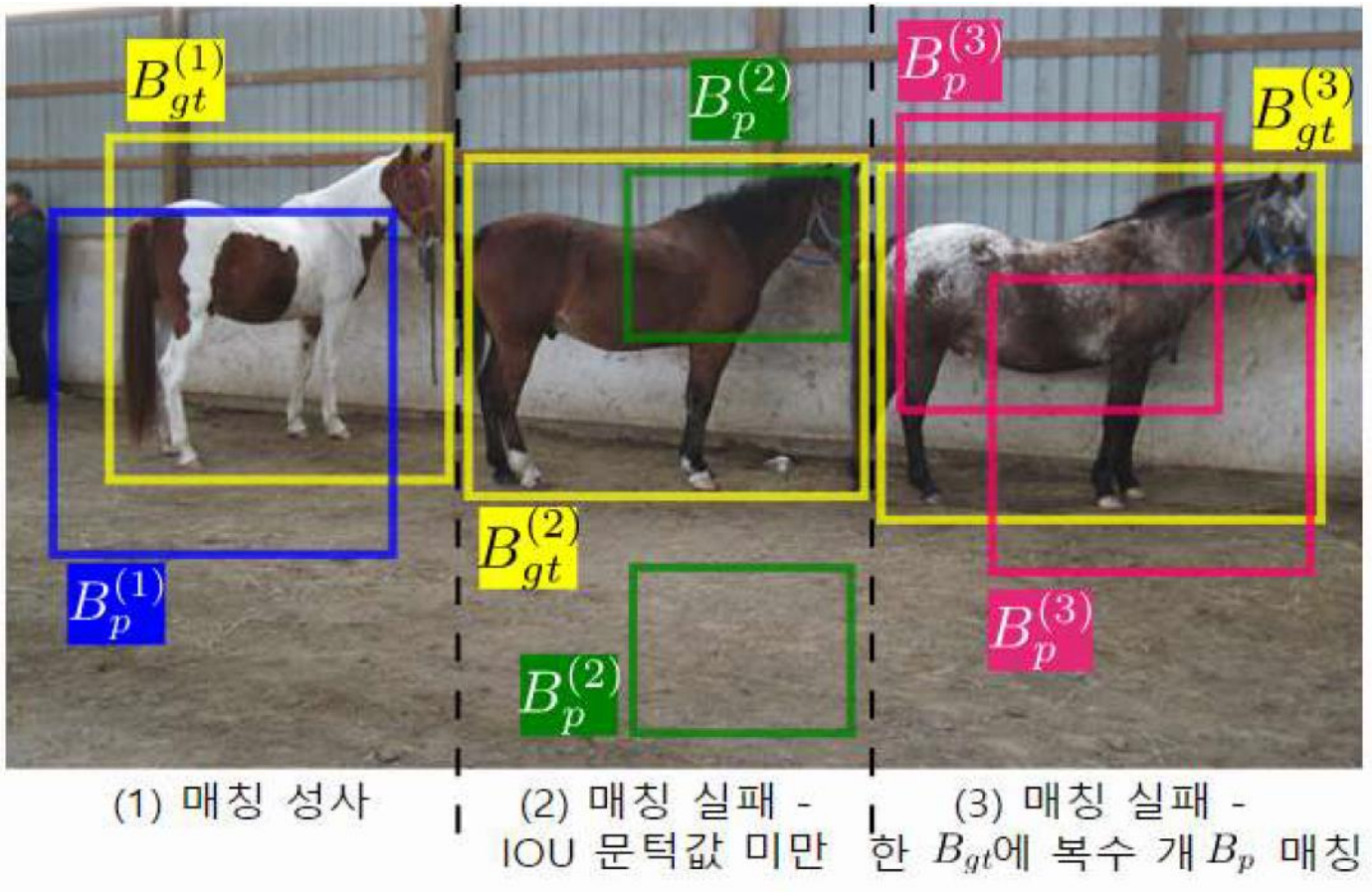
객체탐지_평가

- 객체 검출에서 바운딩 박스를 얼마나 잘 예측하였는지를 IoU라는 지표를 통해서 측정함.
- IOU(Intersection over Union)
  - ✓정답인 Ground Truth와 예측된 바운딩 박스가 얼마나 겹치는가를 측정함.
  - ✓겹친 영역의 비율이 50%(한계점)를 넘겼을 때만 두 바운딩 박스를 매칭
    - 매칭된 예측 바운딩 박스가 바운딩 박스 위치로 선정

$$B_p\text{와 }B_{gt}\text{의 IOU} = \frac{B_p \cap B_{gt}\text{영역 넓이}}{B_p \cup B_{gT}\text{영역 넓이}}$$

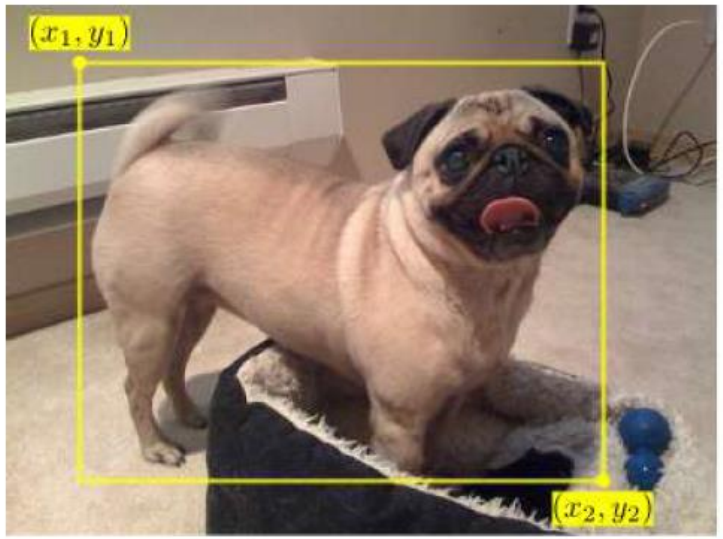


❖ Ground Truth와 예측 바운딩 박스 간의 매칭 성사 및 실패 사례



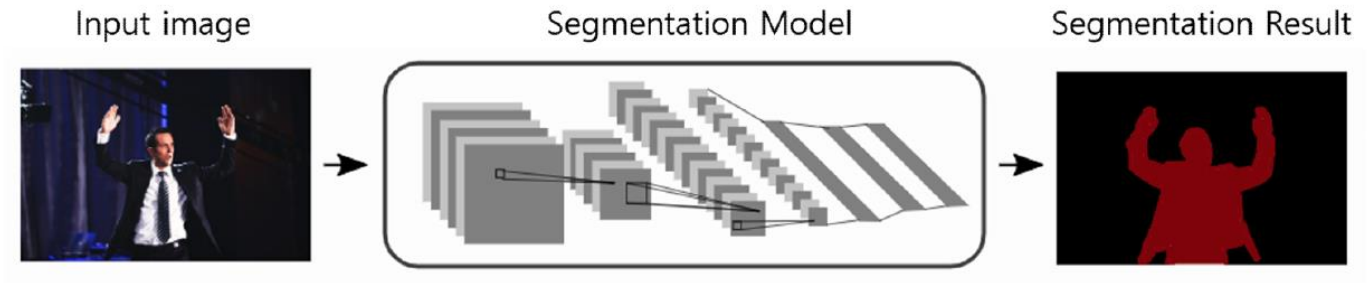
❖ 예시

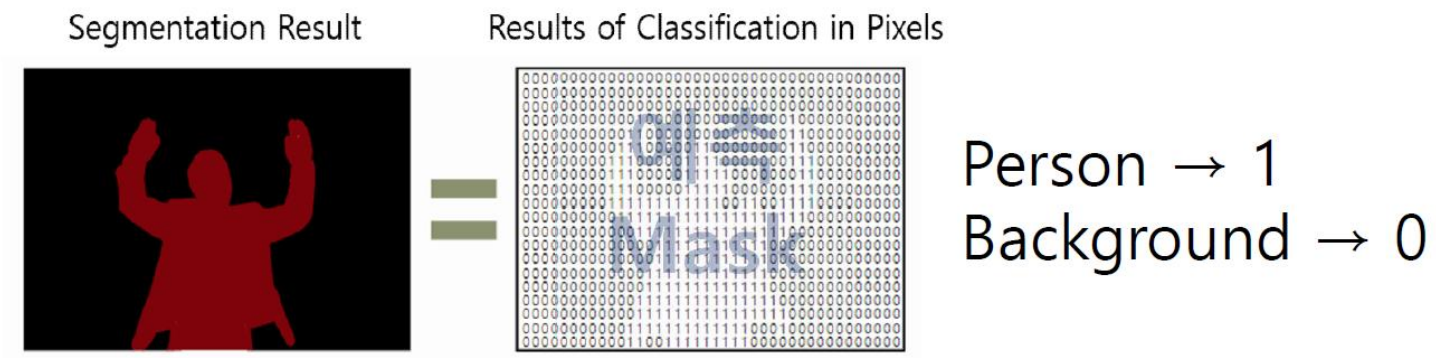
✓주어진 이미지에서 바운딩 박스 안에 dog가 포함되어 있을 가능성을 신뢰도 점수로 표현



Segmantation(분할)_정의

- 주어진 이미지 안에 어느 특정한 클래스가 포함되어 있는지?와 위치가 어디인지?를 픽셀 단위로 분할하는 모델을 만드는 것
  - ✓각 픽셀이 어떤 클래스에 해당 하는지를 곧바로 표시 : Dog->2, Cat->3 등으로 표시





Segmantation(분할)_종류

Input Image

Semantic Segmentation

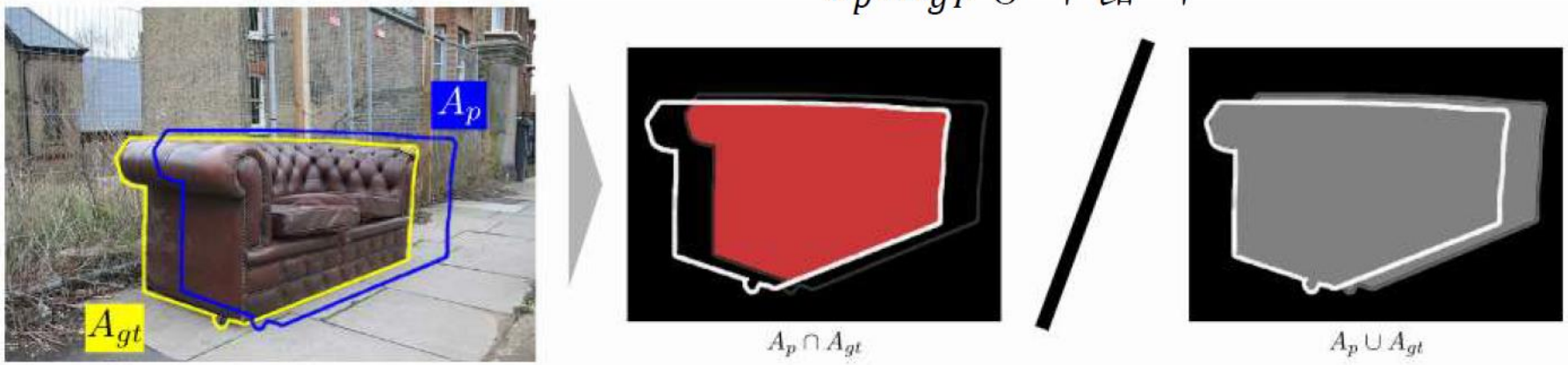
Instance Segmentation

	Semantic Segmentation	Instance Segmentation
분할 기본 단위	Class	Instance(object)
예측 Mask에 표기 방법	동일한 Class는 같은 색상으로 표기	동일한 Class라 하더라도 서로 다른 object라면 다른 색상으로 표기

Segmantation(분할)_모델의 사물 위치 표시 규정

- IOU: 예측 mask와 실제 mask가 얼마나 겹치는가?를 평가

$$A_p\text{와 } A_{gt}\text{의 IOU} = \frac{A_p \cap A_{gt}\text{ 영역 넓이}}{A_p \cup A_{gT}\text{ 영역 넓이}}$$

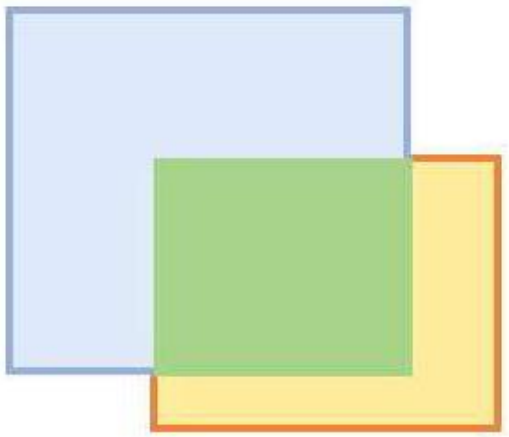


(mask상 흰색으로 표시된 픽셀들은 IOU 계산 시 고려 대상에서 제외)

Segmantation(분할)_모델의 성능 평가 척도

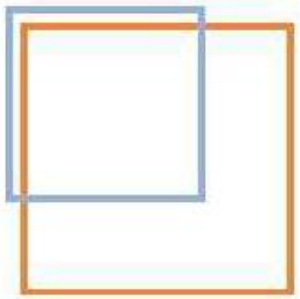
❖ 주어진 이미지에서 mask 안에 person이 포함되어 있을 가능성을 신뢰도 점수로 표현

❖ IoU란?(Intersection over Union)

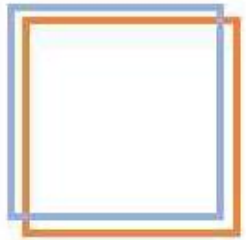


$$IoU = \frac{\text{교집합 면적}}{\text{합집합 면적}}$$

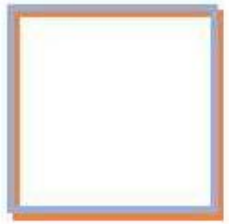
1. $0 \leq IoU \leq 1$
2. 클수록 일치도가 높음



0.4034

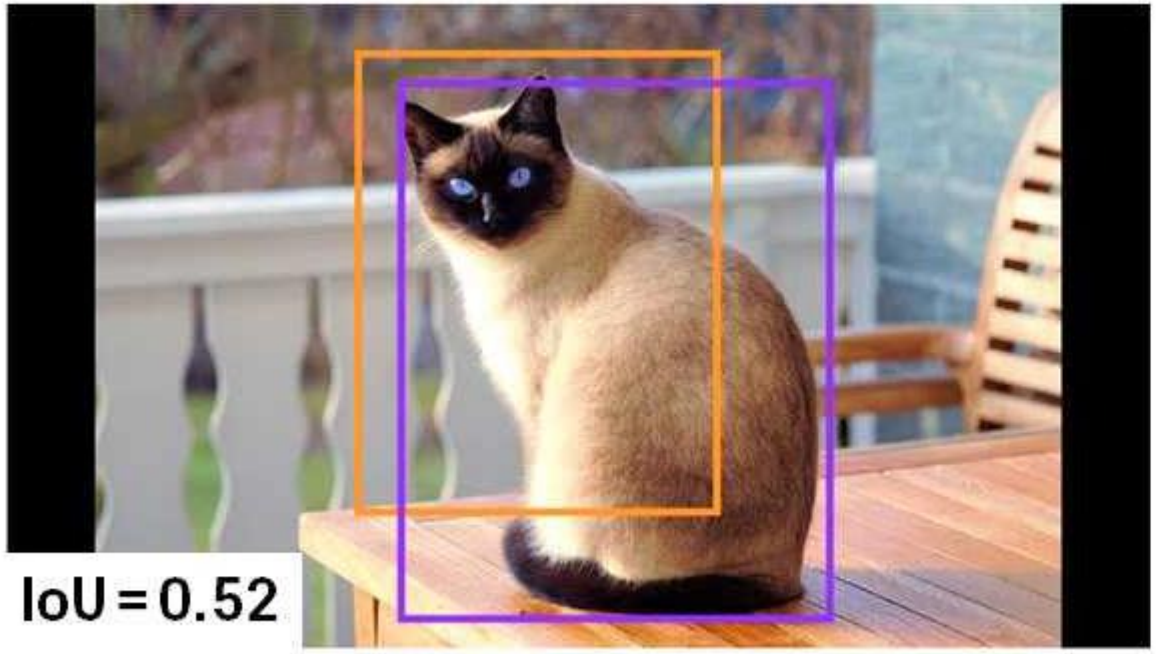


0.7330



0.9264

❖ 객체 탐지에서 IoU 기준



1 Pascal VOC

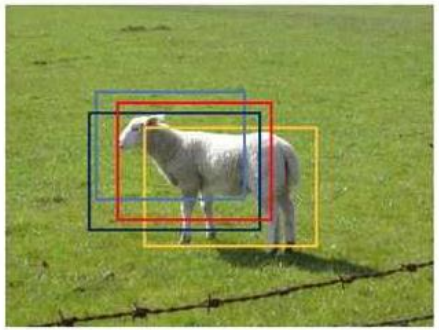
정답 IoU ≥ 0.5

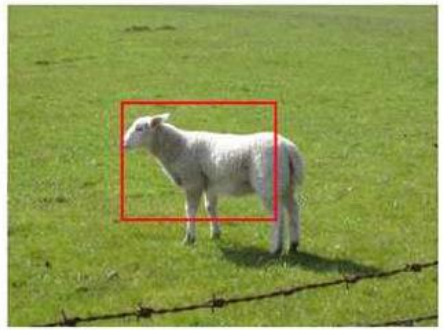
나머지는 오답

2 MS Coco

IoU 0.5부터 0.95까지
0.05 간격으로
측정하여 합산

- NMS(Non Max Suppression) : 중복된 객체의 제거에 사용
  - ✓IoU가 일정 이상 중첩된 box들 중에서 Confidence Score가 가장 높은 것만을 남김



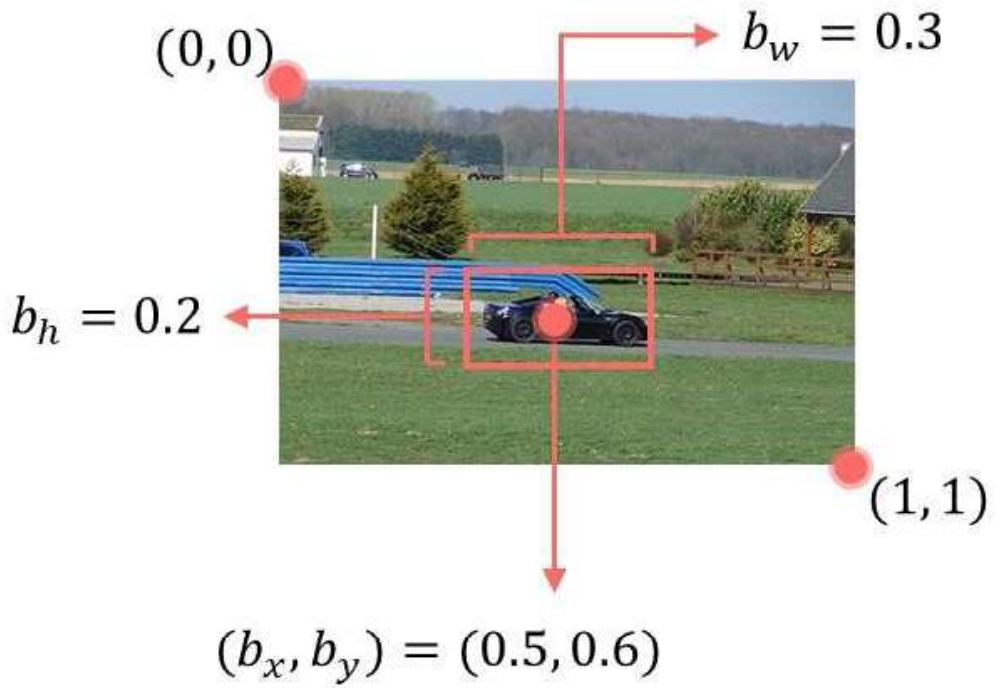


Confidence. Score: 0.5
Confidence. Score: 0.6
Confidence. Score: 0.9
Confidence. Score: 0.7

- 해당 ROI를 CNN 분류기를 통해 인식하고 Confidence Score가 threshold 이상을 결과로 출력

- 하지만, sliding window는 너무 느리므로 한번에 인식할 수 있는 다른 방안이 필요함

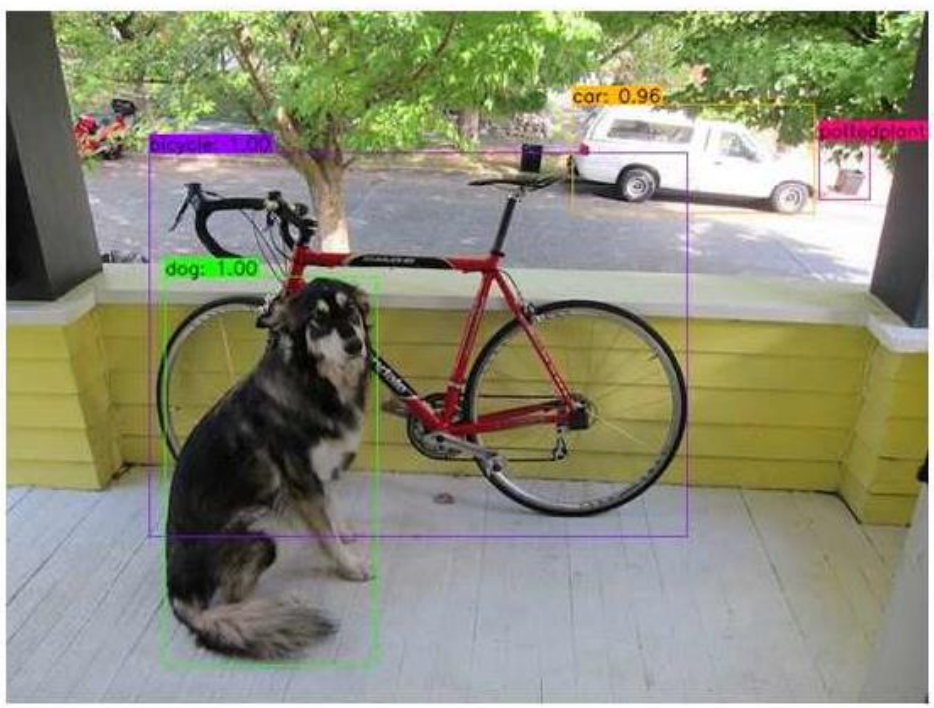
❖ Bounding box의 정의

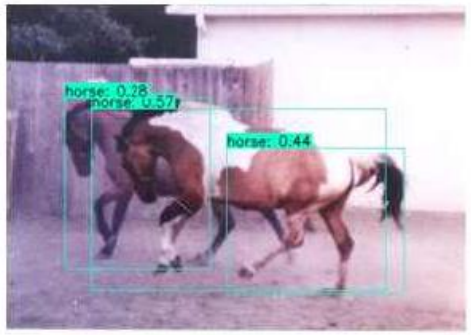


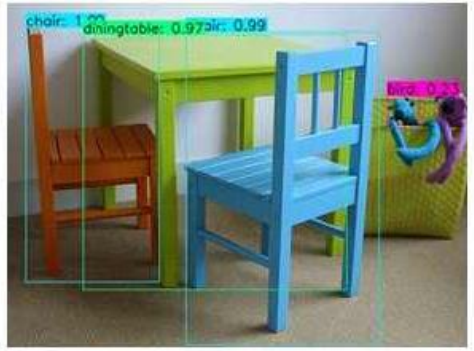
- b_x, b_y: 센터 위치
- b_w, b_h: width, height
- bounding box 출력 텐서

$$y = \begin{bmatrix} b_x \\ b_y \\ b_w \\ b_h \end{bmatrix}$$

❖ 객체탐지 실행 결과 : Pascal VOC 2012 학습 결과







❖ 객체탐지 실행 결과

1 거리 영상

Pascal VOC 2012로 학습

2 강아지 영상

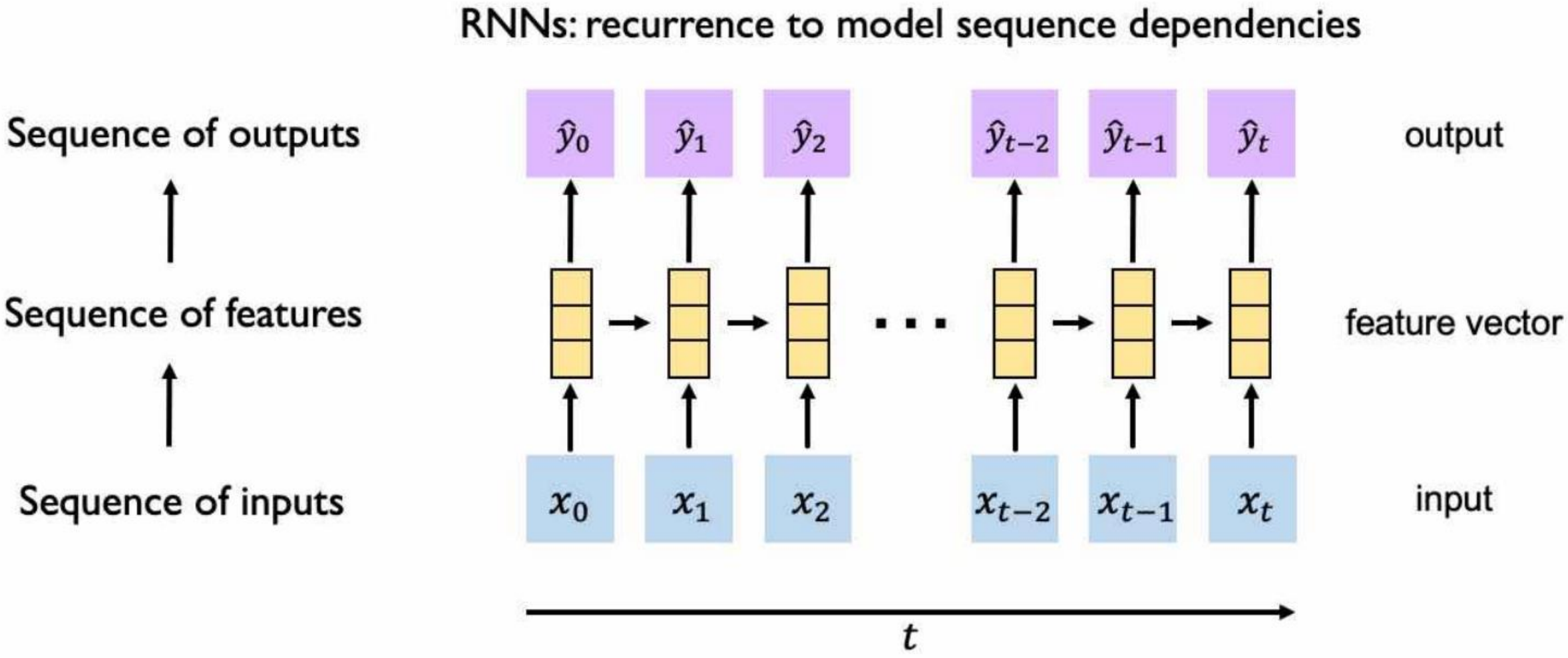
COCO로 학습

(YOLOv4의 사전 학습)

동영상 원본 출처 :
Youtube German Korb
- No copyright
Youtube CHANNEL PMI
- Create Comons Licence

Day3-Lec3: 강의 Attention Model

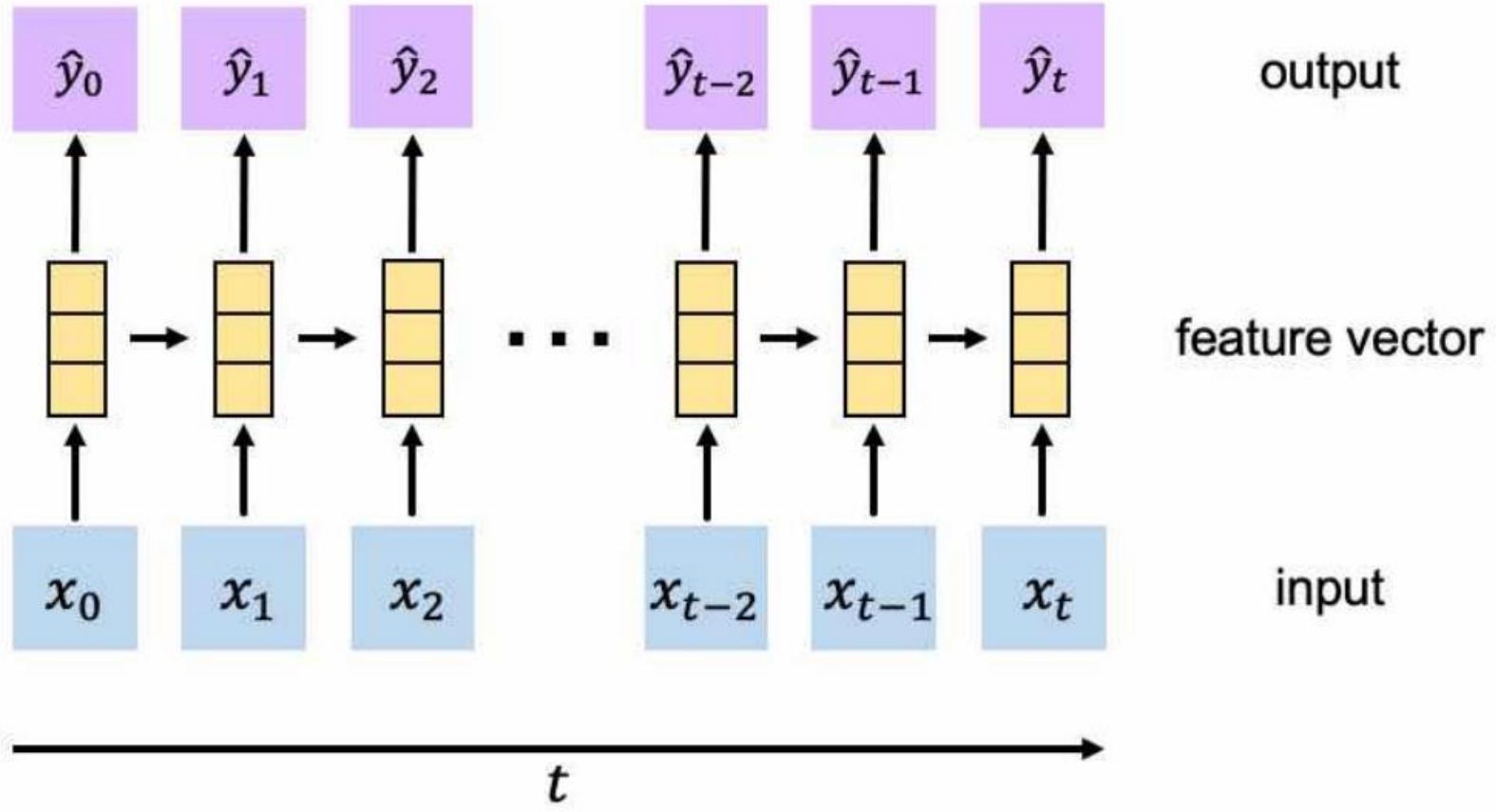
Goal of Sequence Modeling



Goal of Sequence Modeling

Limitations of RNNs

- Encoding bottleneck
- Slow, no parallelization
- Not long memory

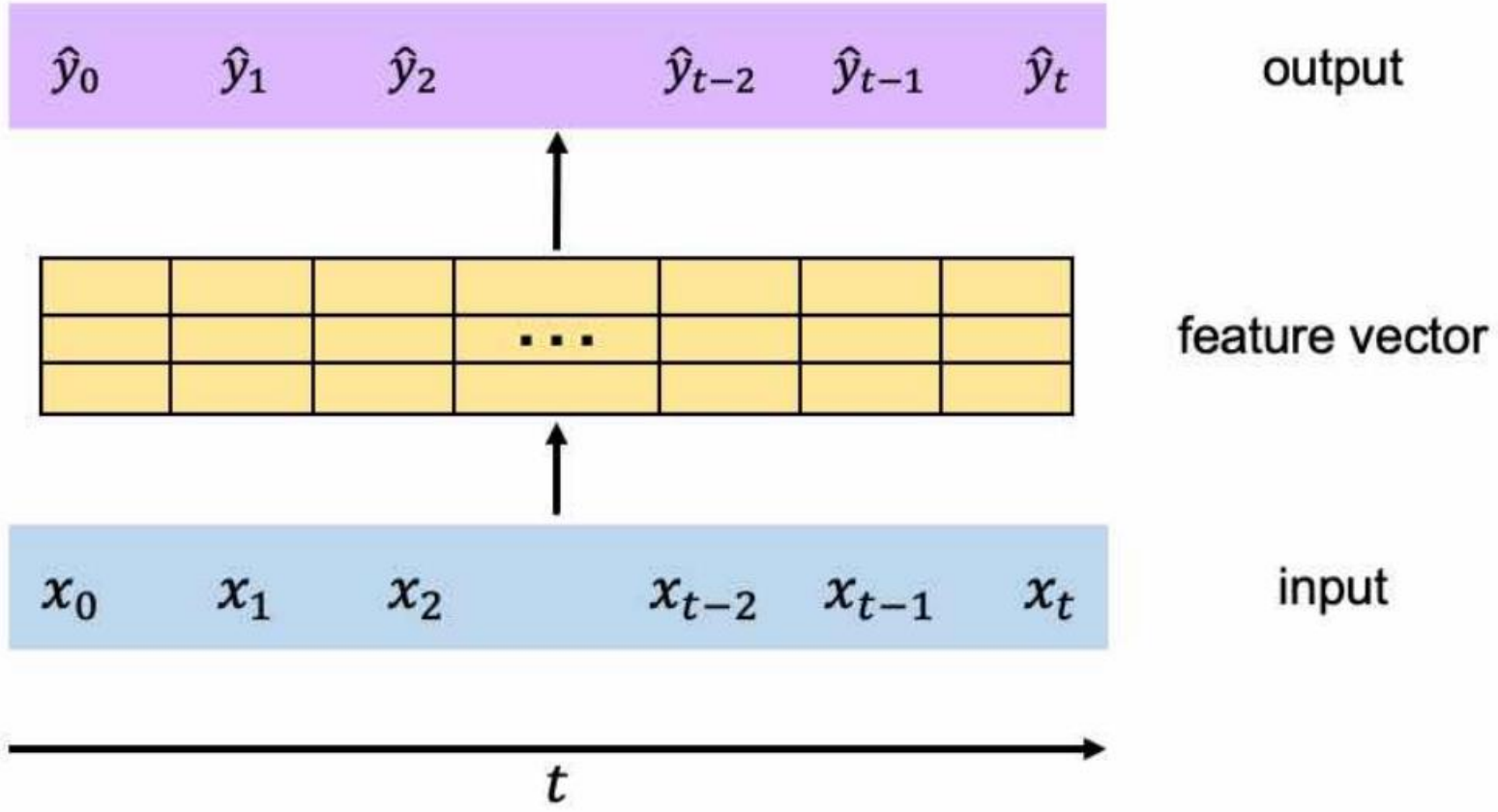


Goal of Sequence Modeling

Desired Capabilities

- Continuous stream
- Parallelization
- Long memory

Can we eliminate the need for recurrence entirely?



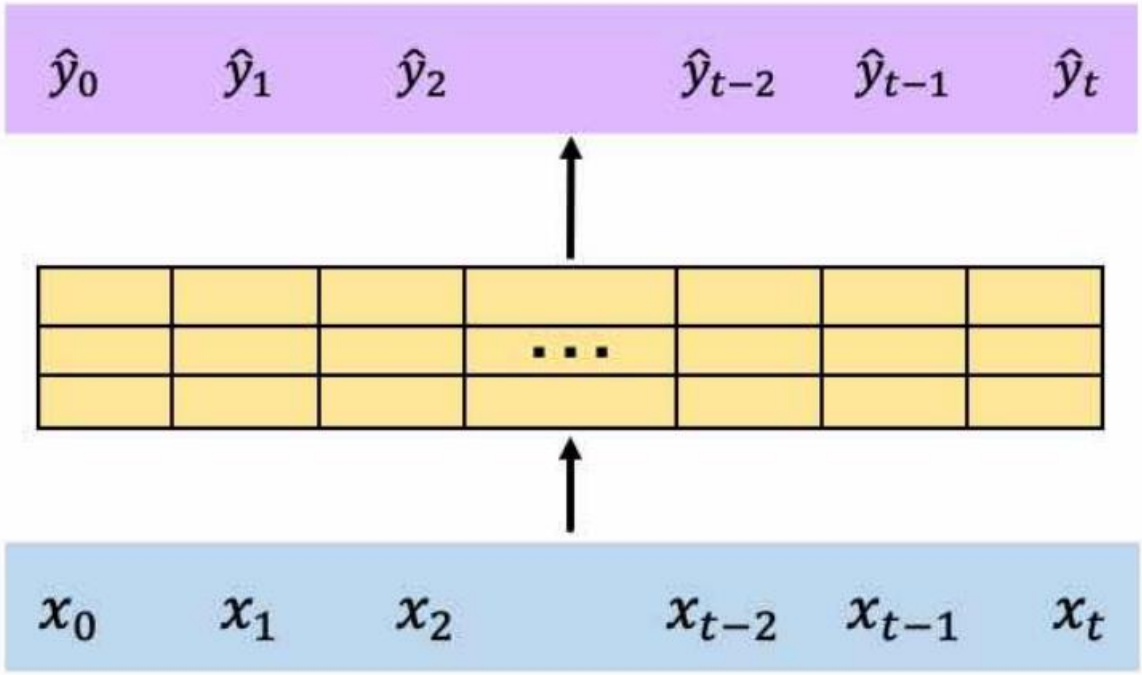
Goal of Sequence Modeling

Idea 1: Feed everything into dense network

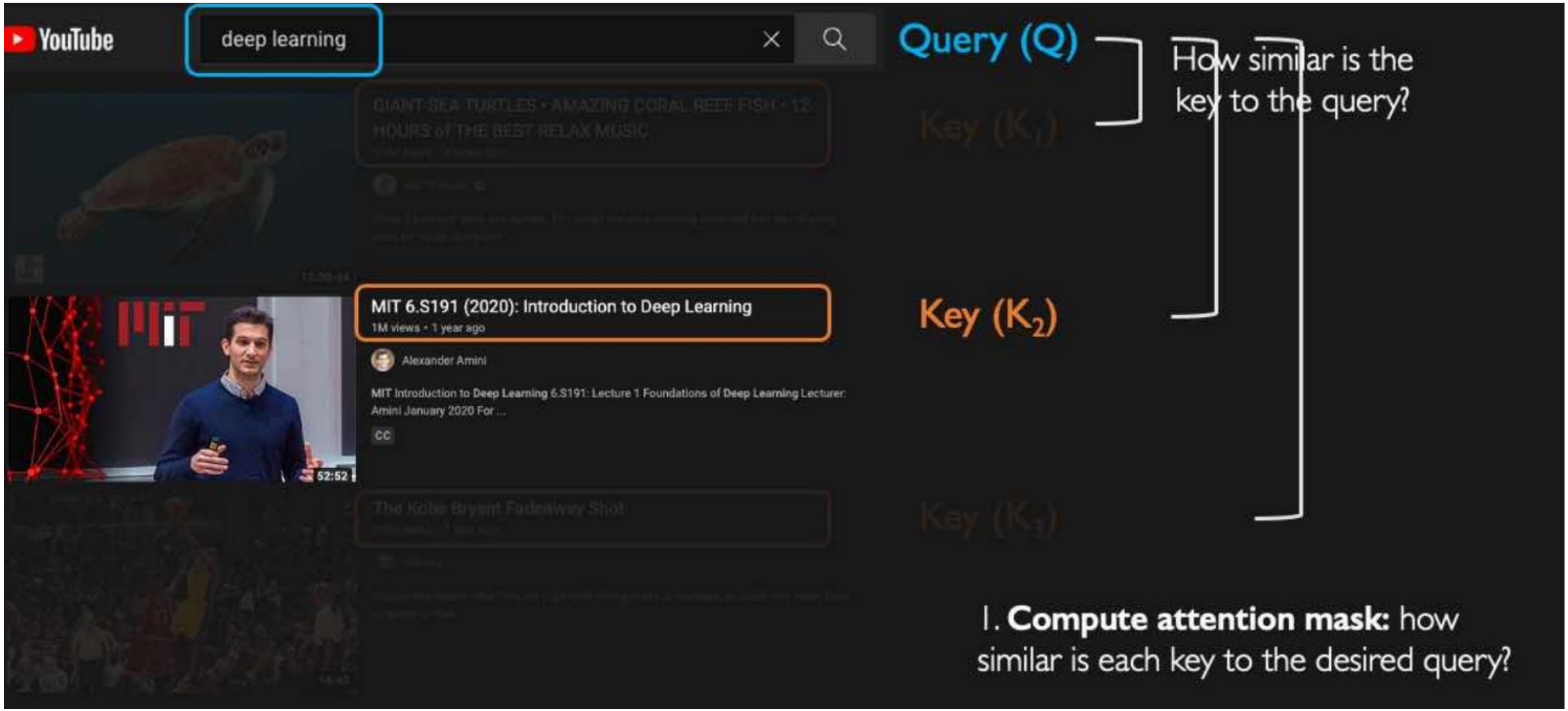
- ✓ No recurrence
- ✗ Not scalable
- ✗ No order
- ✗ No long memory

Idea: Identify and attend to what's important

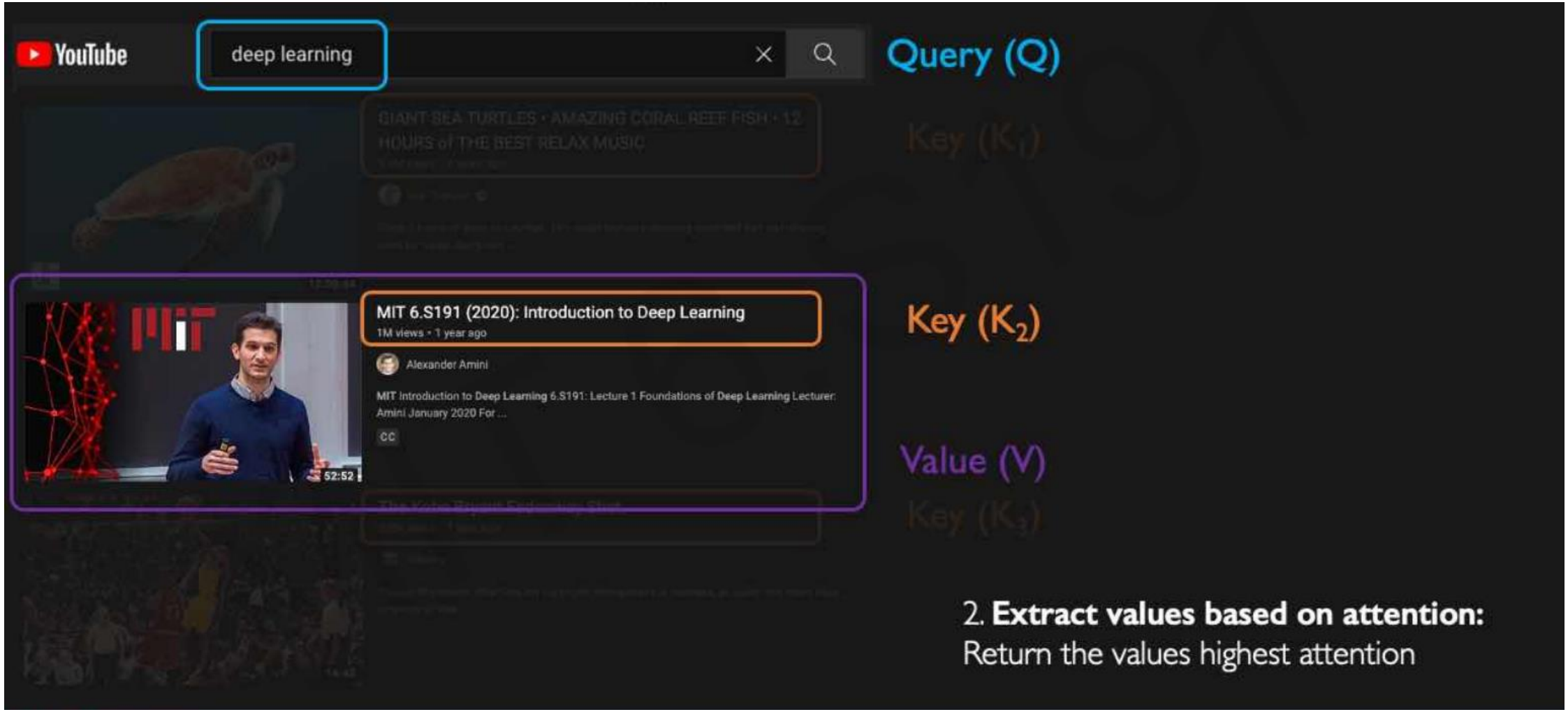
Can we eliminate the need for recurrence entirely?



Understanding Attention with Search

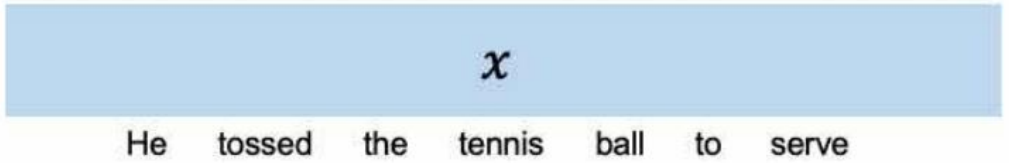


Understanding Attention with Search



Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.



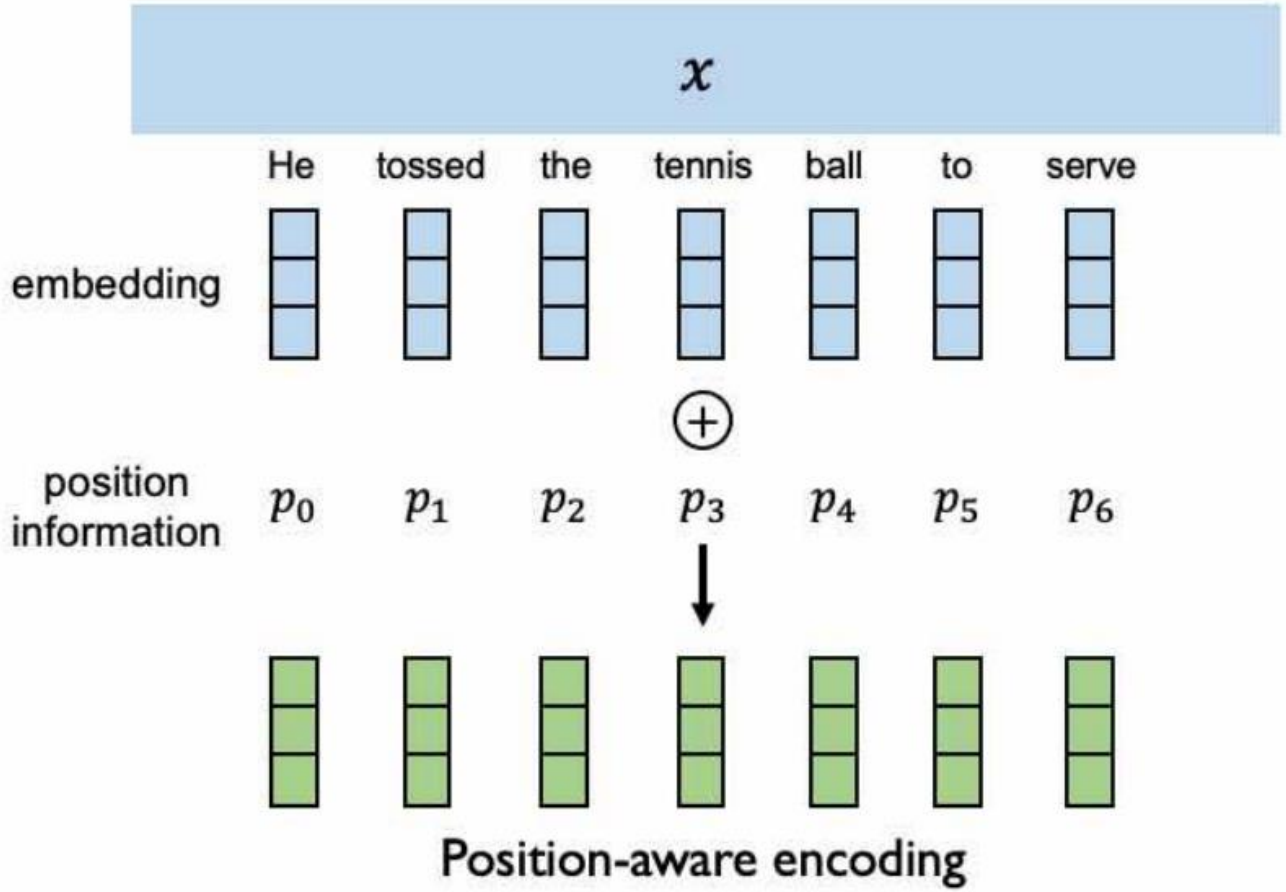
1. Encode **position** information
2. Extract query, key, value for search
3. Compute attention weighting
4. Extract features with high attention

Data is fed in all at once! Need to encode position information to understand order.

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract query, key, value for search
3. Compute attention weighting
4. Extract features with high attention

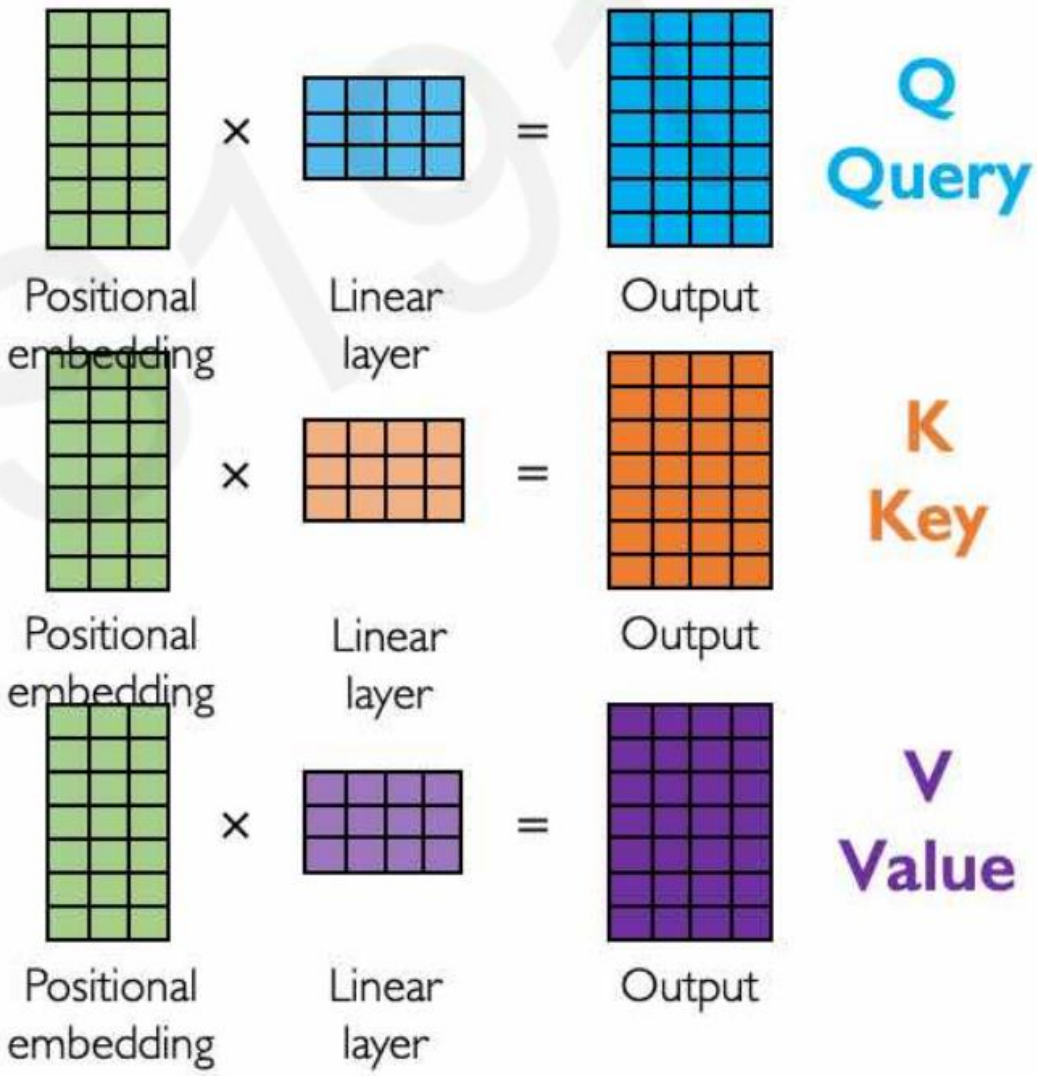


Data is fed in all at once! Need to encode position information to understand order.

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute attention weighting
4. Extract features with high attention



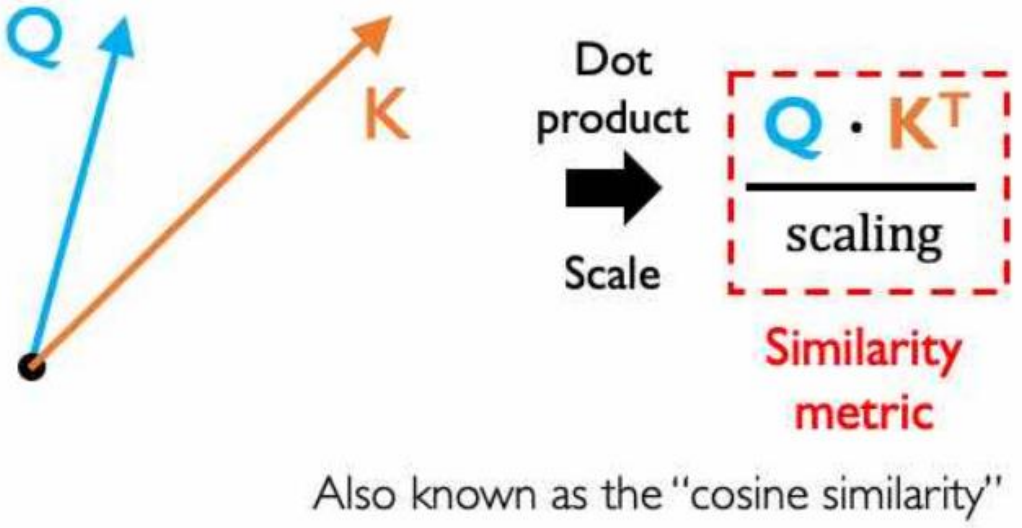
Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



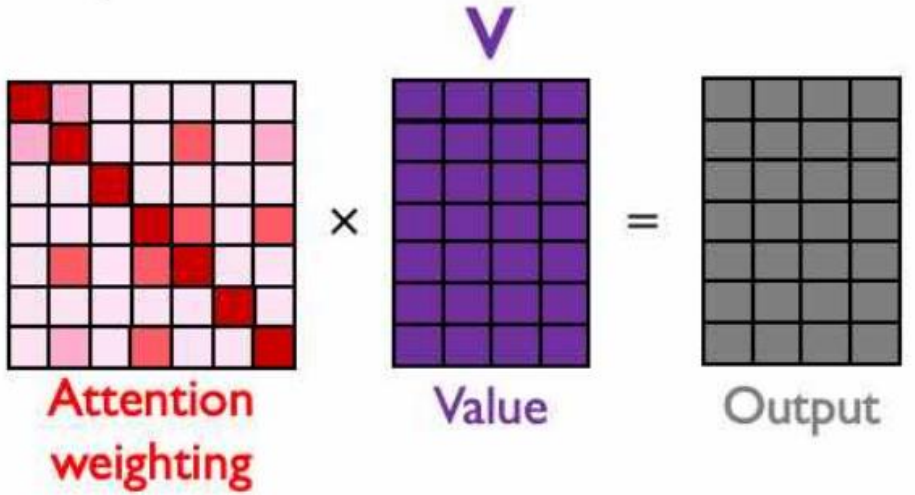
Also known as the "cosine similarity"

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Last step: self-attend to extract features



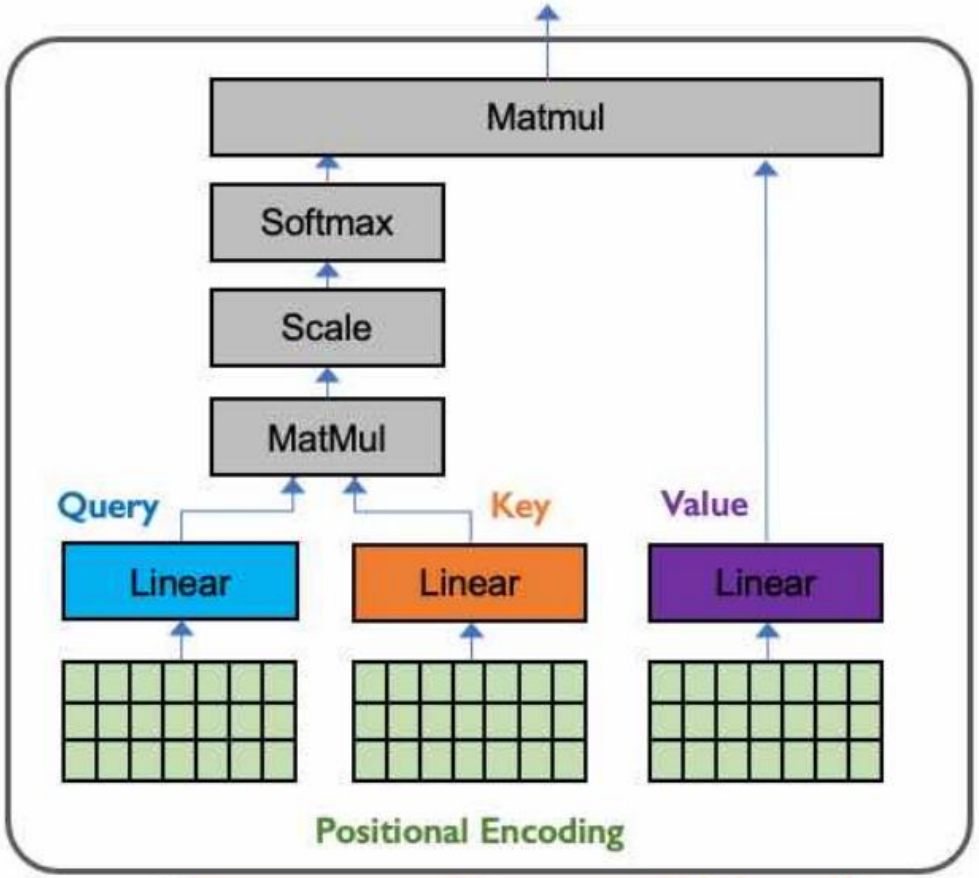
$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V = A(Q, K, V)$$

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

These operations form a self-attention head that can plug into a larger network. Each head attends to a different part of input.



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V$$

Self-Attention Applied

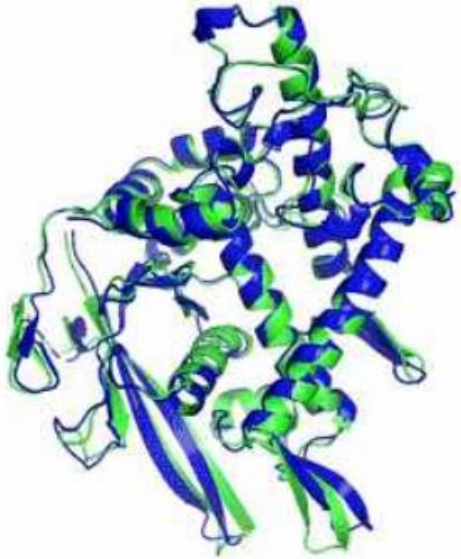
Language Processing

An armchair in the shape of an avocado

BERT, GPT-3

Devlin et al., *NAACL* 2019
Brown et al., *NeurIPS* 2020

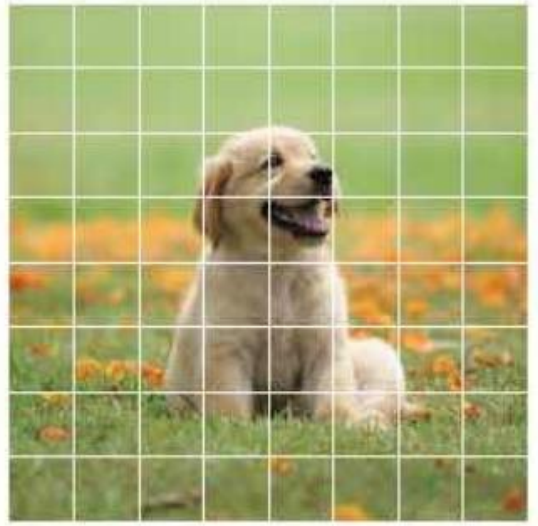
Biological Sequences



AlphaFold2

Jumper et al., *Nature* 2021

Computer Vision



Vision Transformers

Dosovitskiy et al., *ICLR* 2020

Day3-Lec4: 실습(회귀) 시간의존 LSTM 예측하기

(B) 시계열을 고려한 회귀를 해보자

문제를 간단히 하기 위해서 속도(Speed)만을 고려하자

-과거 데이터 패턴인 기억(Memory)의 길이를 고려하자.

- windows (Look_back)으로 데이터를 새롭게 구성하자.
- Many-to-One 문제를 고려한다. (Many-to-Many)도 가능함

In [56]:
```
df.head()
```

Out[56]:

Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
2017-04-02 00:00:00	43	34	9	0	50.3	1.90
2017-04-02 00:05:00	45	32	13	0	58.9	1.84
2017-04-02 00:10:00	46	34	12	0	50.6	1.87
2017-04-02 00:15:00	45	36	9	0	50.9	1.72
2017-04-02 00:20:00	27	13	13	1	62.2	1.12

B1: 시계열 패턴을 고려한 RNN 예측

In [62]:
```
spd=df.reset_index()['Speed']
spd.head()
plt.plot(spd)

# windows = 30
windows = 12
X, y = [], []

for i in range(len(spd)-windows-1):
    X.append(spd[i:(i+windows)])
    y.append(spd[(i+windows)])

X = np.array(X)
y = np.array(y)
X.shape
```

In [82]:
```
X[:3]
```

Out[82]:
```
array([[50.3, 58.9, 50.6, 50.9, 62.2, 63.5, 73.6, 52.7, 59.9, 59.1, 54.5,
        50.6],
       [58.9, 50.6, 50.9, 62.2, 63.5, 73.6, 52.7, 59.9, 59.1, 54.5, 50.6,
        50.8],
       [50.6, 50.9, 62.2, 63.5, 73.6, 52.7, 59.9, 59.1, 54.5, 50.6, 50.8,
        65. ]])
```

In [83]:
```
y[:3]
```

Out[83]:
```
array([50.8, 65. , 58.1])
```

B1: 시계열 패턴을 고려한 RNN 예측

In [84]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X = scaler.fit_transform(X)
pd.DataFrame(X).head()
```

Out[84]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.523507	0.632783	0.527319	0.531131	0.674714	0.691233	0.819568	0.554003	0.645489	0.635324	0.576874	0.527319
1	0.632783	0.527319	0.531131	0.674714	0.691233	0.819568	0.554003	0.645489	0.635324	0.576874	0.527319	0.529860
2	0.527319	0.531131	0.674714	0.691233	0.819568	0.554003	0.645489	0.635324	0.576874	0.527319	0.529860	0.710292
3	0.531131	0.674714	0.691233	0.819568	0.554003	0.645489	0.635324	0.576874	0.527319	0.529860	0.710292	0.622618
4	0.674714	0.691233	0.819568	0.554003	0.645489	0.635324	0.576874	0.527319	0.529860	0.710292	0.622618	0.637865

Prediction

In [85]:
```
#now lets split data in test train pairs

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle=False)
Acc = []
```

LSTM

In [86]:
```
X_train_ = X_train.reshape(X_train.shape[0],X_train.shape[1],1)
X_test_ = X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

In [87]:
```
print(X_train_.shape)
print(X_test_.shape)
```

B1: 시계열 패턴을 고려한 RNN 예측

In [111]:

```
from tensorflow.keras.layers import LSTM
from keras import metrics

def reg_lstm():
    model = Sequential()

    model.add(LSTM(20, activation = 'relu', return_sequences=True,
                   input_shape=(X_train.shape[1],1)))
    #model.add(LSTM(20, activation = 'relu', return_sequences=True))
    model.add(LSTM(20, activation = 'relu'))
    model.add(Dense(1))

    model.compile(loss='mean_squared_error', optimizer='adam',
                  metrics =[metrics.mae])

    return model
```

B1: 시계열 패턴을 고려한 RNN 예측

In [112]:

```
# Model Training

model_B1 = reg_lstm()
model_B1.summary()

history = model_B1.fit(X_train_, y_train, epochs=20, batch_size=16, validation_split=0.2)

model_B1.evaluate(X_test_, y_test)

# Prediction
y_pred_B1 = model_B1.predict(X_test_)

y_pred_B1

pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_B1.flatten()})
pred_df.head()
```

B1: 시계열 패턴을 고려한 RNN 예측

In [119]:
```
# Measure the Accuracy Score

from sklearn.metrics import r2_score

print("Accuracy score of the predictions: {0}".format(r2_score(y_test, y_pred_B1)))
Acc.append(r2_score(y_test, y_pred_B1))
```

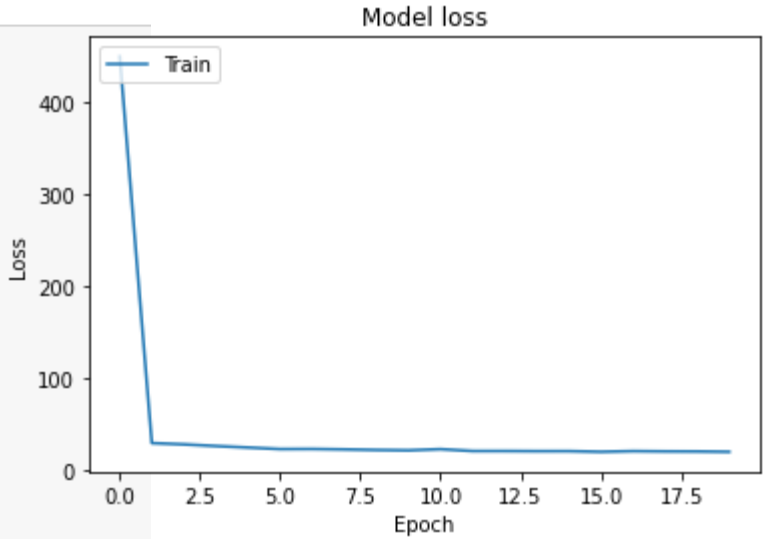
```
Accuracy score of the predictions: 0.6592988682742402
```

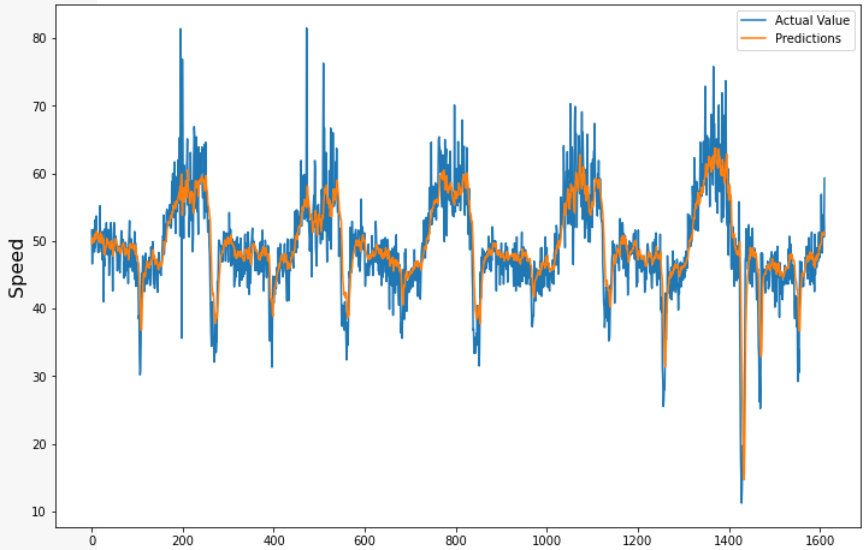
In [120]:
```
plt.figure(figsize=(12,8))
plt.ylabel('Speed', fontsize=16)
plt.plot(pred_df)
plt.legend(['Actual Value', 'Predictions'])
plt.show()
```

B1: 시계열 패턴을 고려한 RNN 예측

In [121]:

```
# 7 훈련 과정 시각화 (손실)
plt.plot(history.history['loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Visualising the results
plt.figure(figsize=(14,9))
plt.plot(y_test, color = 'red', label = 'Real VDS Data')
plt.plot(y_pred_B1, color = 'blue', label = 'Predicted VDS Data')
plt.title('Traffic Speed Prediction')
plt.xlabel('Time')
plt.ylabel('VDS (Speed)')
plt.legend()
plt.show()
```





B2: 시계열 패턴을 고려한 DNN 예측

In [123]:

```
# Model Training
model_B2 = reg_dnn(inp_dim=X_train.shape[1])
history = model_B2.fit(X_train, y_train, epochs=20, validation_split=0.2)

# Prediction
y_pred_B2 = model_B2.predict(X_test)

# Measure the Accuracy Score
from sklearn.metrics import r2_score
print("Accuracy score of the predictions: {0}".format(r2_score(y_test, y_pred_B2)))
Acc.append(r2_score(y_test, y_pred_B2))

pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_B2.flatten()})
pred_df.head()
```

B3: 시계열 패턴을 고려한 1D-CNN

3. CNN

In [128]:
```
# Model Training

model_B3 = reg_cnn()
model_B3.fit(X_train_, y_train, epochs=20, validation_split=0.2)

# Prediction

y_pred_B3 = model_B3.predict(X_test_)

pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_B3.flatten()})
pred_df.head()
```

```
Epoch 1/20
161/161 [==============================] - 3s 12ms/step - loss: 275.1259 - val_loss: 20.1644
Epoch 2/20
161/161 [==============================] - 2s 10ms/step - loss: 22.2775 - val_loss: 19.6817
```

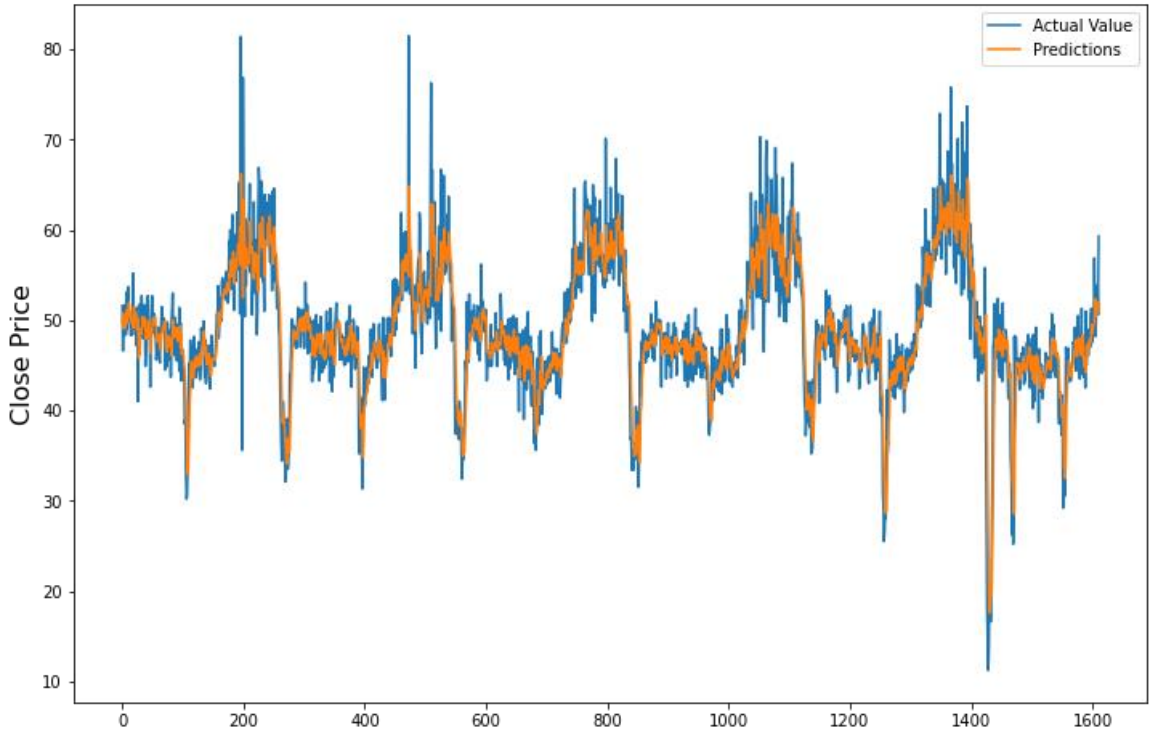
B3: 시계열 패턴을 고려한 1D-CNN

In [131]:
```
# Measure the Accuracy Score

from sklearn.metrics import r2_score

print("Accuracy score of the predictions: {0}".format(r2_score(y_test, y_pred_B3)))
Acc.append(r2_score(y_test, y_pred_B3))
```

```
Accuracy score of the predictions: 0.7063216813941307
```

B3: 시계열 패턴을 고려한 1D-CNN

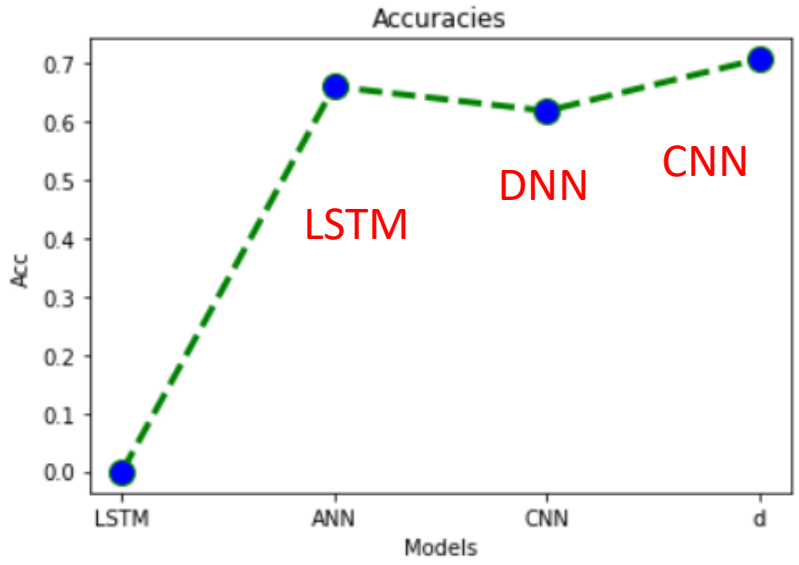
In [132]:
```
plt.figure(figsize=(12,8))
plt.ylabel('Speed', fontsize=16)
plt.plot(pred_df)
plt.legend(['Actual Value', 'Predictions'])
plt.show()
```



B3: 시계열 패턴을 고려한 1D-CNN

In [135]:

```
plt.plot(range(4), Acc, color='green', linestyle='dashed', linewidth = 3,
         marker='o', markerfacecolor='blue', markersize=12)
plt.ylabel('Acc')
plt.xlabel('Models')
plt.title("Accuracies")
plt.xticks(range(4), ['LSTM', 'ANN', 'CNN','d'])
plt.show()
```



Day3-Lec5: 강의
교통을 위한 트랜스포머 모델

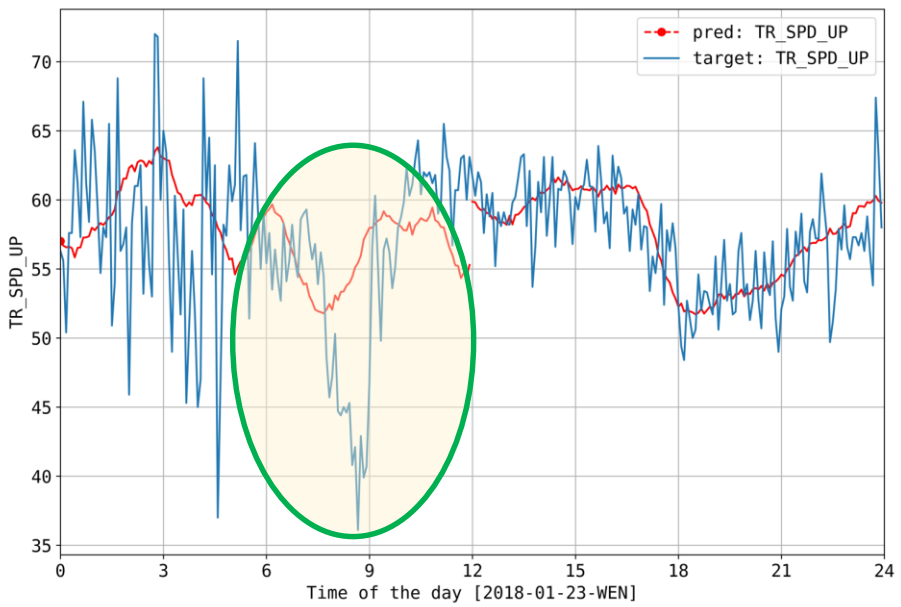
대상 도로 : 대학로 (유성구청네거리-구성삼거리)

❖ 약 1.4km 왕복 4차로, RSE 3개, 영상검지기 5개(현재 3개, 향후 2개)

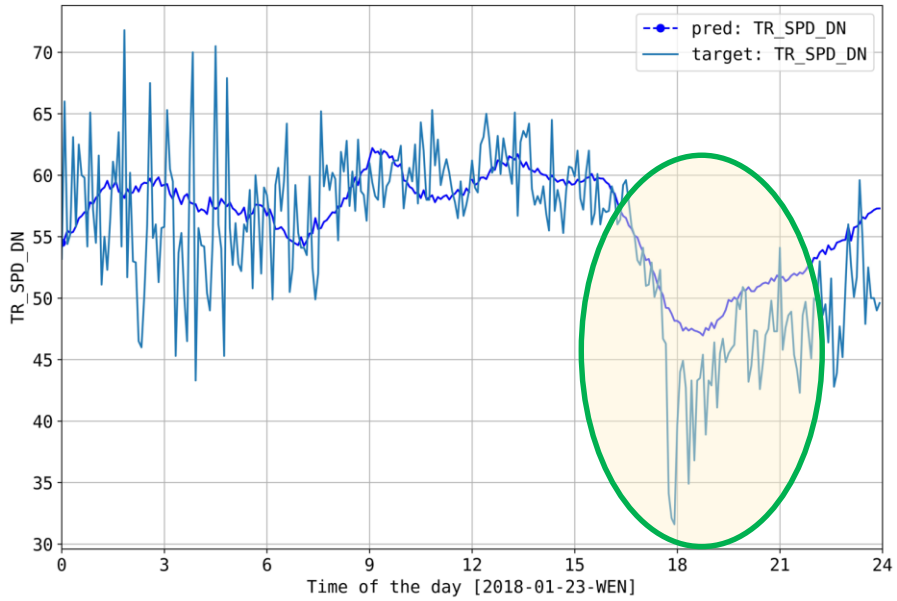
LSTM을 이용한 교통 흐름 예측 및 새로운 모델의 필요

LSTM 예측은 RNN에 비하여 성능은 좋은나, Long-Sequence 장기예측에는 성능저하가 발생한다.

상행 12시간예측 (2018.01.23.,수요일)



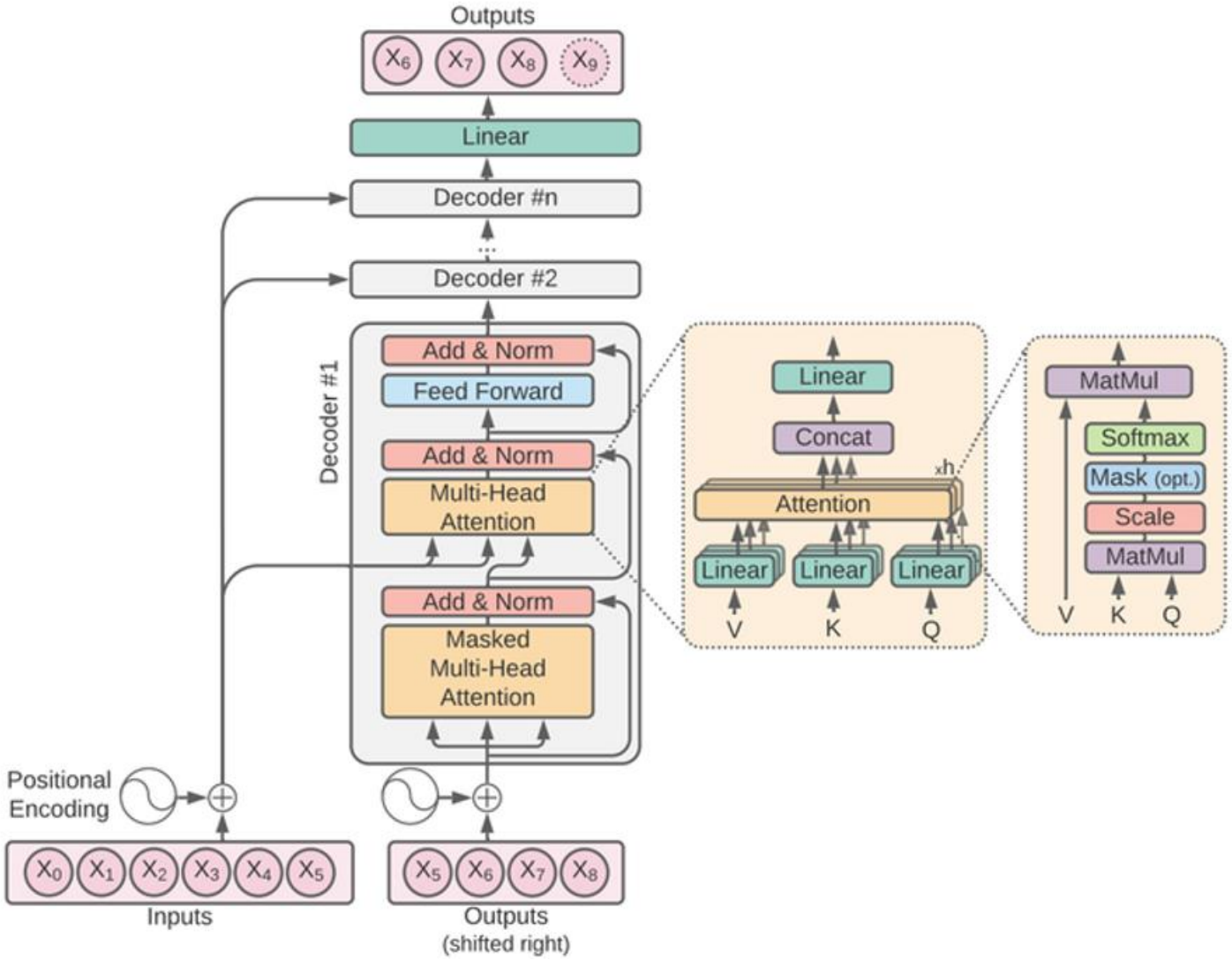
하행 12시간예측 (2018.01.23.,수요일)



LSTM 장기예측(12시간)과 예측 정확은 재고할 필요가 있다. → 새로운 모델이 필요하다.

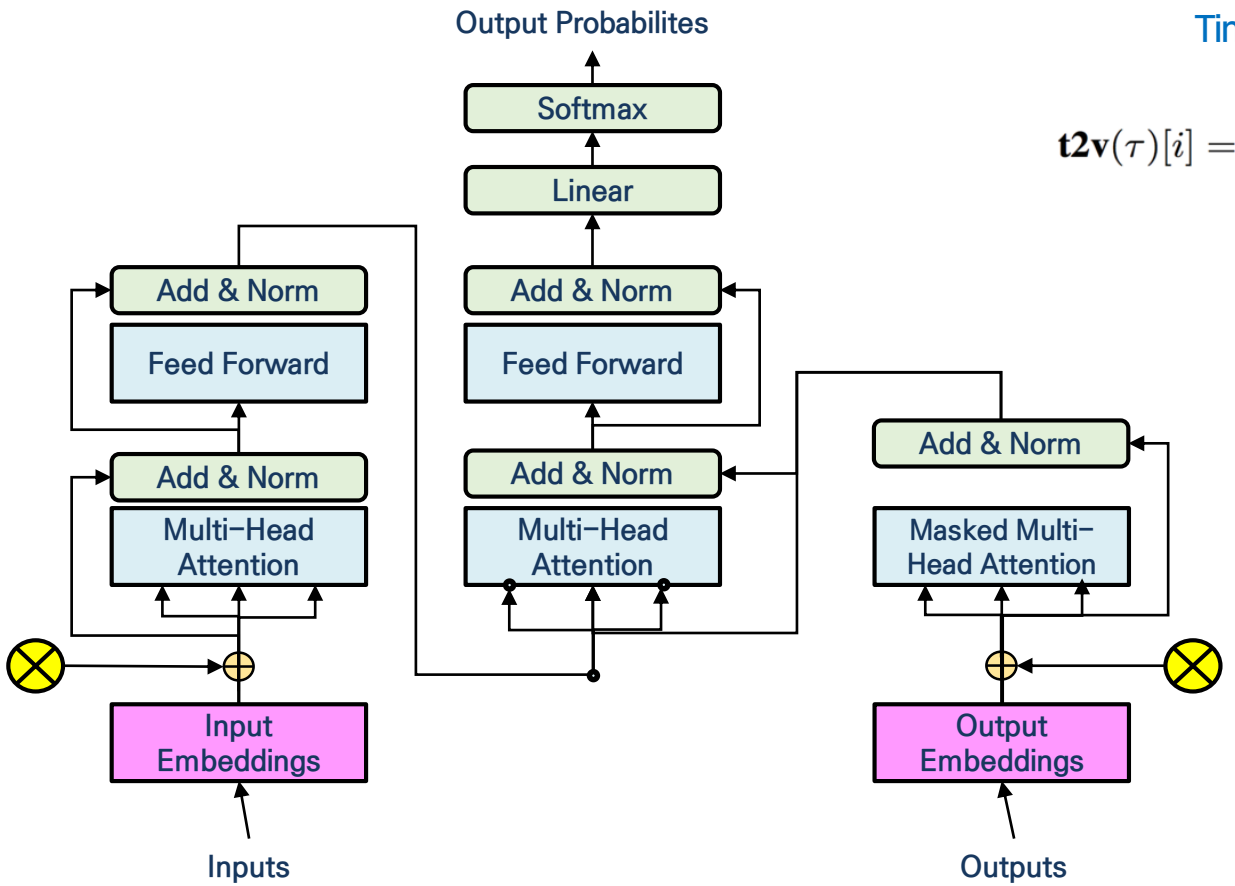
☆ LSTM 장기 예측(12시간) 결과는 상대적으로 정확도가 현저히 떨어진다.

멀티 헤드 어텐션 메커니즘



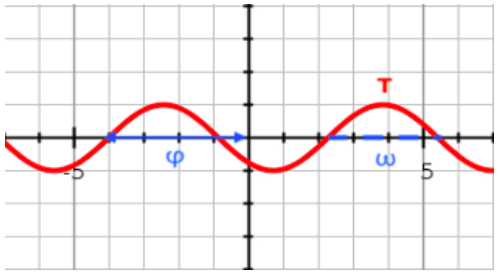
교통에 적합한 Position Encoding : Time2Vec

자연어 처리 트랜스포머 모델의 포지셔널 인코딩을 교통에 적합하게 Time2Vec 개발이 핵심이다.



❖ Attention 도입으로 시계열성을 없어지고, 대신에 Time2Vec 임베딩을 이용하여 시공간적인 정보 유지

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$



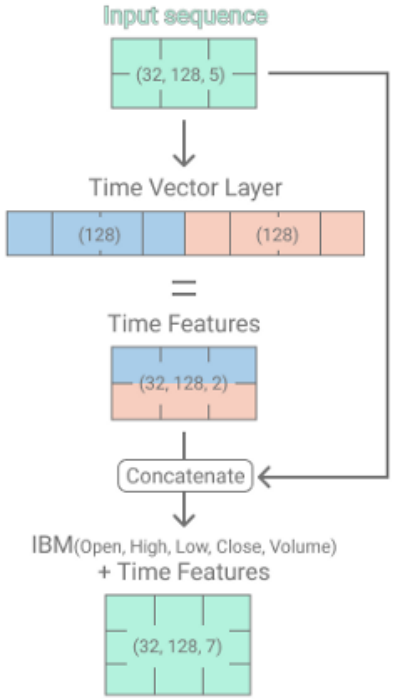
주기성: 삼각함수(sin)를 사용

비주기성: 선형함수

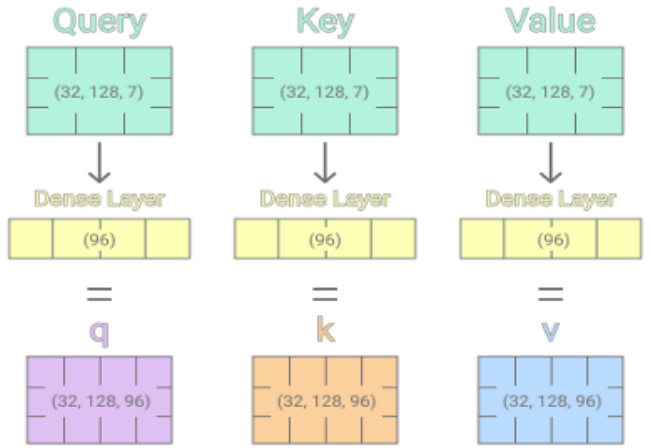
어텐션의 Query, Key, Value 전략

Time2Vec

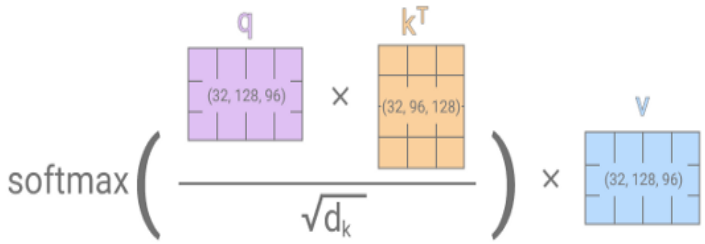
- 5개 VDS 17, VDS 18, VDS 19, VDS 20, VDS 21
- 트랜스포머 모델 입력크기(input-size)
  - batch_size: 32, sequence_length: 128, feature : 5



Single Head Attention : Query, Key, Value



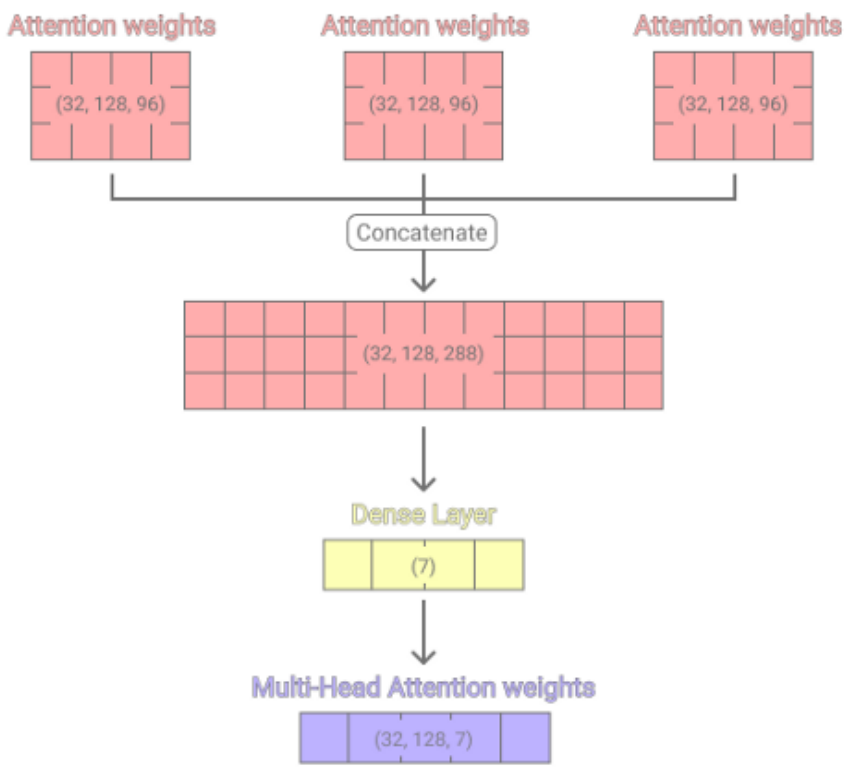
- Attention Weight는 Softmax를 사용함. Q*K*V



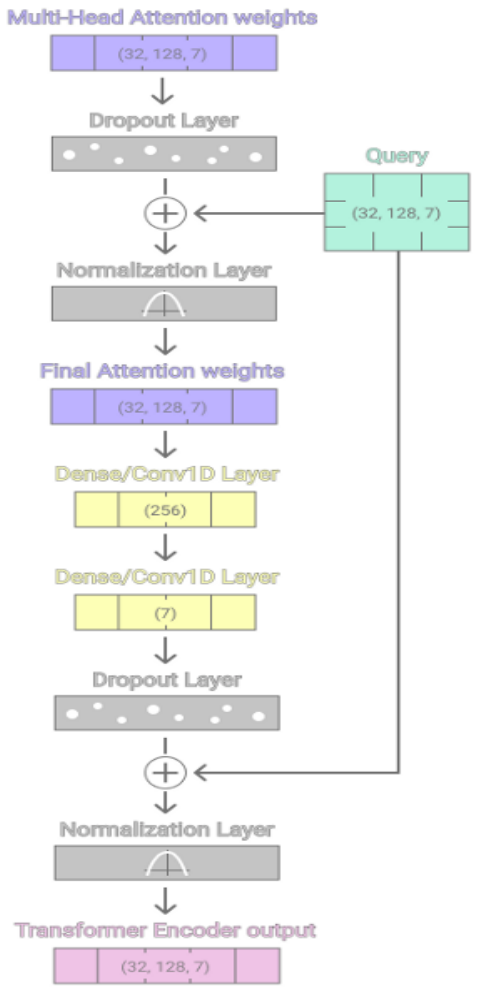
인코더-디코더 아키텍처 기반 Transformer 교통 모델

Mulit-Head Attention으로 병렬처리가 가능함

❖ VDS 17, VDS 18 데이터 구조 (train, test, validation)

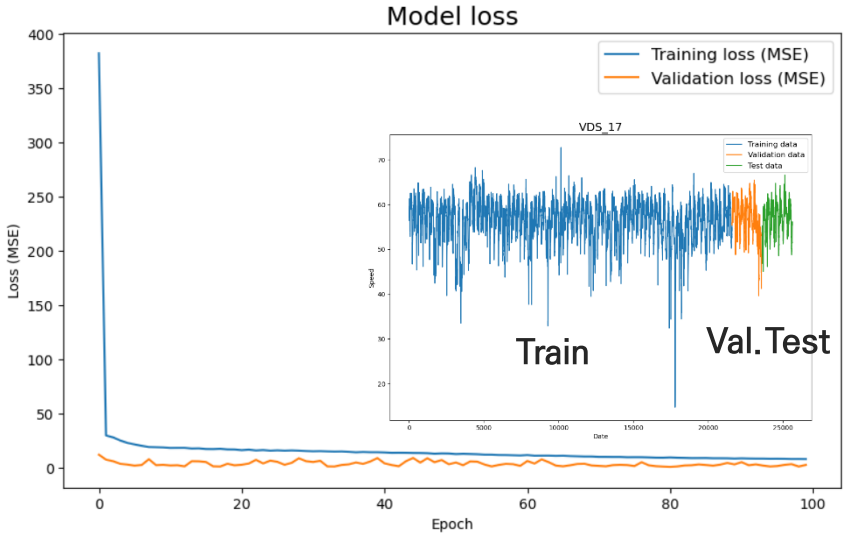


Transformer Encoder 출력이되 Decoder에 사용됨



트랜스포머 VDS 데이터로 장기 예측

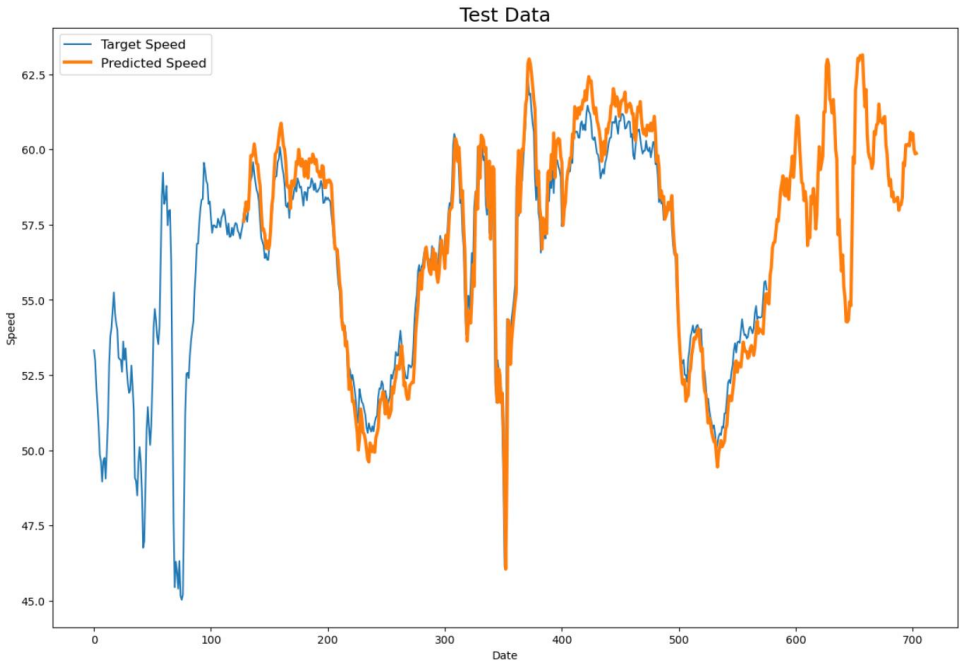
대전시 차량검지기(VDS) 데이터 105개



- 대학로 관련 5개 위치 데이터 90일
  - 데이터 개수: 25,920개
  - 90일*24*12 (5분 단위)

장기예측 성능 : 2016개 예측(1주일, 288*7)

- TransformerVDS: Transformer+TimeEmbedding



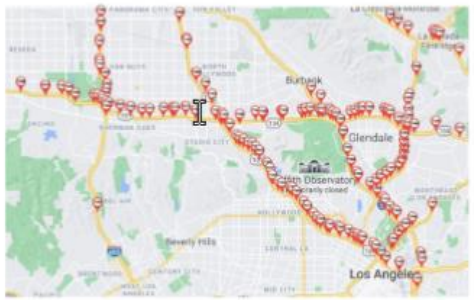
- TransformerVDS 결과와 대전 UVDS 데이터에 적용하고 있으며, SCI 논문에 투고할 예정

Dynamic Spatial Transformer WaveNet Network

KISTI SINCE 1962

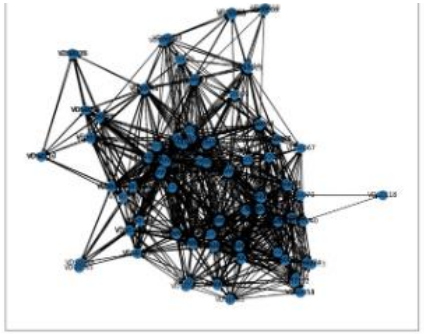
UVDS: 대전 VDS 데이터 기반 시공간 그래프 신경망 구축
Spatial-temporal Graph neural Network

- Metr-LA : 대표적 교통 공개 데이터 (미국, LA지역 4개월)
- UVDS 공개 : (시공간) 대전시 차량검지기 데이터 (3개월)

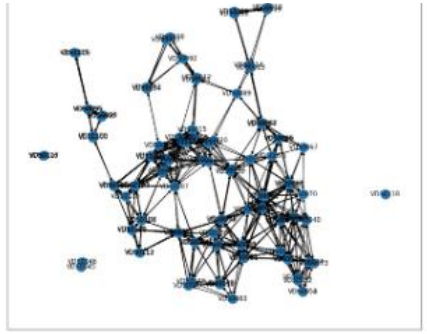


a) METR-LA

b) UVDS



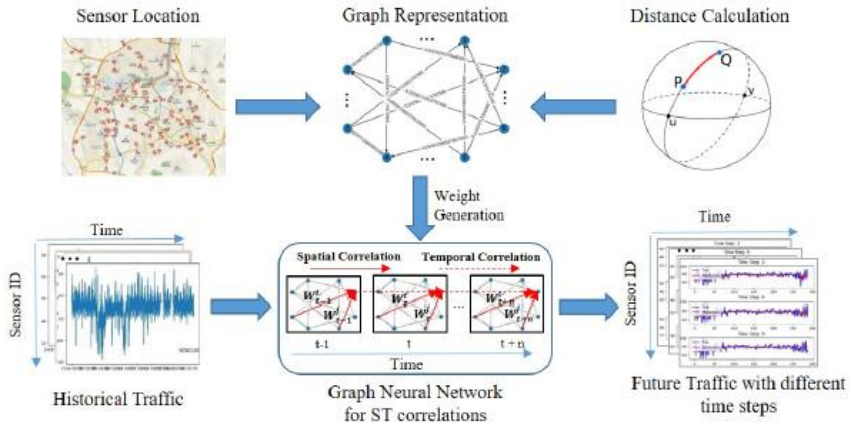
a) β = 0.1



b) β = 0.5

대전시 VDS 데이터 기반 새로운 UVDS 데이터 공개
UVDS: A New Dataset for Trac Forecasting with Spatial-Temporal Correlation

- General Framework for Traffic Forecasting using Spatial-temporal Correlaitons



Dynamic Spatial Transformer WaveNet Network

UVDS 데이터 기반 동적 시공간-트랜스포머 웨이브넷 신경망 개발 및 UVDS 데이터 성능 테스트
DSTWN 모델 : Dynamic Spatial Transformer WaveNet Network로 Spatial-Temporal Graph 신경망 보다 우수성 검증

❖ 기존 Spatial-Temporal Graph Network과 비교 성능 향상 목표

❖ 개발한 DSTWN 알고리즘 성능 벤치마크 : Metr-LA 데이터

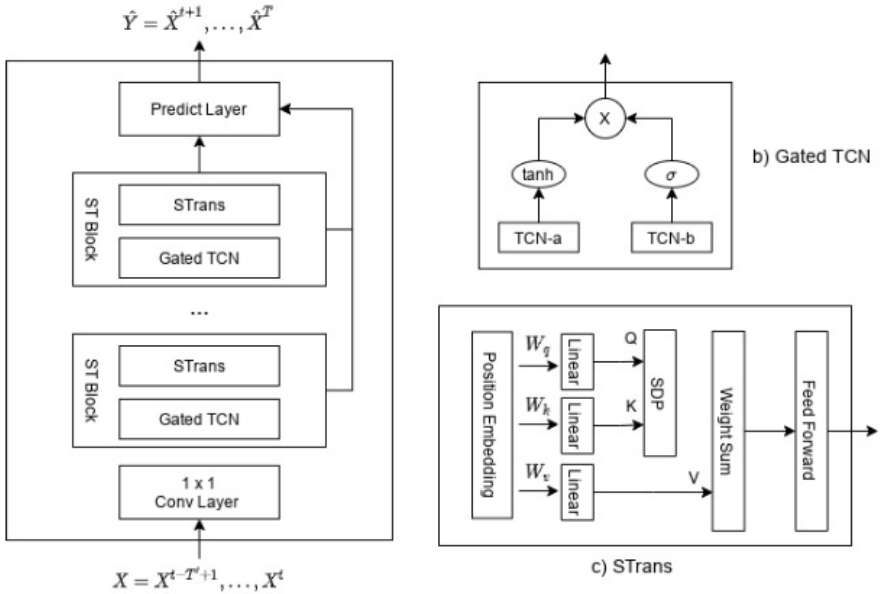


Fig. 3. The DSTWN Architecture

Table 2. Results on METR-LA dataset.

Step	Metrics	GraphWN	MTGNN	STWaNet	STTN	DSTWN
1	MAE	2.2408	2.2441	2.2791	2.4080	2.3220
	RMSE	3.8636	3.9185	3.9793	4.2339	4.0704
	MAPE	0.0540	0.0551	0.0550	0.0588	0.0567
3	MAE	2.7127	2.6762	2.7457	2.9160	2.8758
	RMSE	5.1690	5.1428	5.3162	5.6556	5.6285
	MAPE	0.0695	0.0686	0.0711	0.0778	0.0759
6	MAE	3.0974	3.0605	3.0947	3.3819	3.2909
	RMSE	6.1839	6.2002	6.2781	6.8651	6.7740
	MAPE	0.0847	0.0816	0.0838	0.0958	0.0900
9	MAE	3.3617	3.3100	3.3239	3.6961	3.5461
	RMSE	6.8279	6.8380	6.8639	7.5749	7.3837
	MAPE	0.0950	0.0912	0.0924	0.1085	0.0984
12	MAE	3.5760	3.4937	3.5036	3.9533	3.7446
	RMSE	7.2883	7.2421	7.2761	8.1262	7.8246
	MAPE	0.1035	0.0982	0.0993	0.1181	0.1047
Training		45.6927	62.8770	54.5744	77.5496	133.2374
Inference		1.4630	1.6825	1.5830	6.6945	3.8088

딥러닝을 이용한 교통흐름 예측 연구

KISTI SINCE 1962

❖ 대전시 데이터웨어하우스에서 차량검지기(VDS)와 RSE 데이터 수집 및 분석함

- ✓KISTI 정문 앞 도로 차량검지기(VDS 17데이터)는 오전과 출근과 오후 퇴근에 일부 속도가 떨어지는 경향이 있음.
- ✓대학로 인근 RSE 데이터는 전체적으로 속도가 너무 낮게 측정되었음.

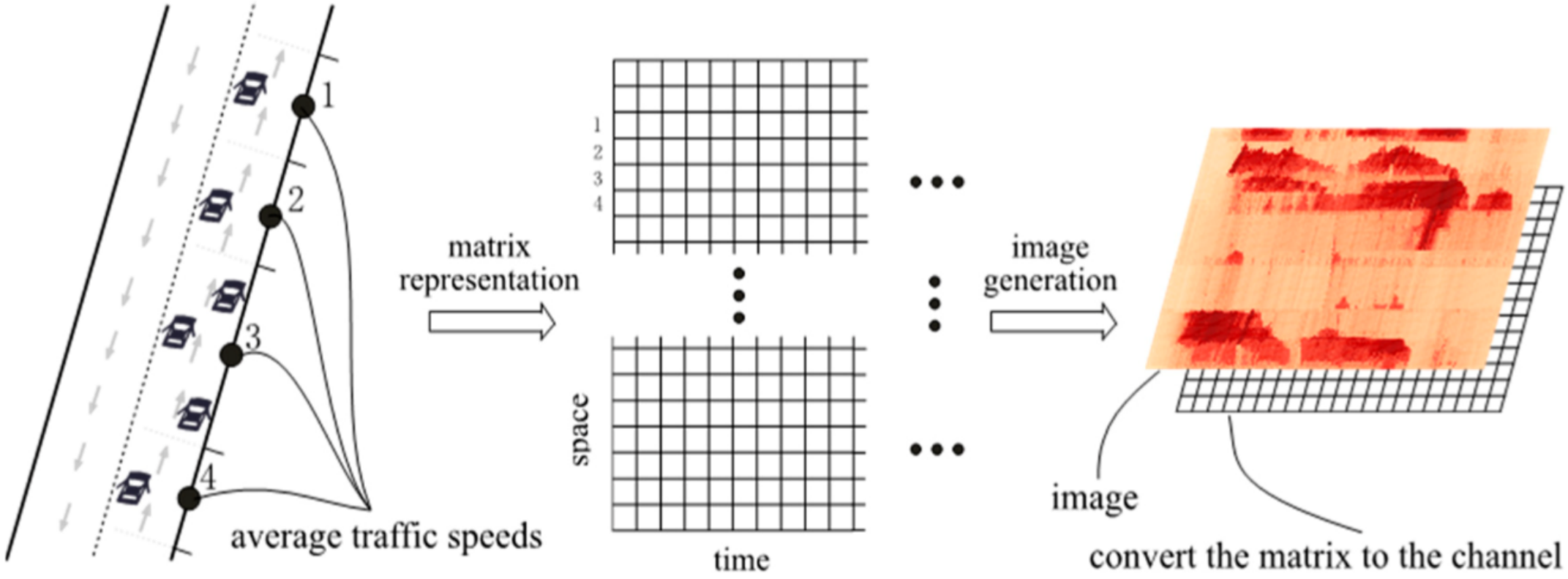
❖ (딥러닝 기반 교통 흐름 예측)

- ✓RNN 기반 LSTM을 양방향 장단기로 교통 흐름 예측하였고 교통 혼잡은 없는 것으로 예측됨
- ✓Transformer_VDS 모델 개발로 장기 교통흐름 예측 정확도가 향상됨
- ✓Spatial-Temporal Graph 데이터 기반 Dynamic Spatial Transformer WaveNet 모델 개발 및 성능 테스트 함

Day3-Lec6: 강의 Deep Reinforcement Learning

Transportation network speed

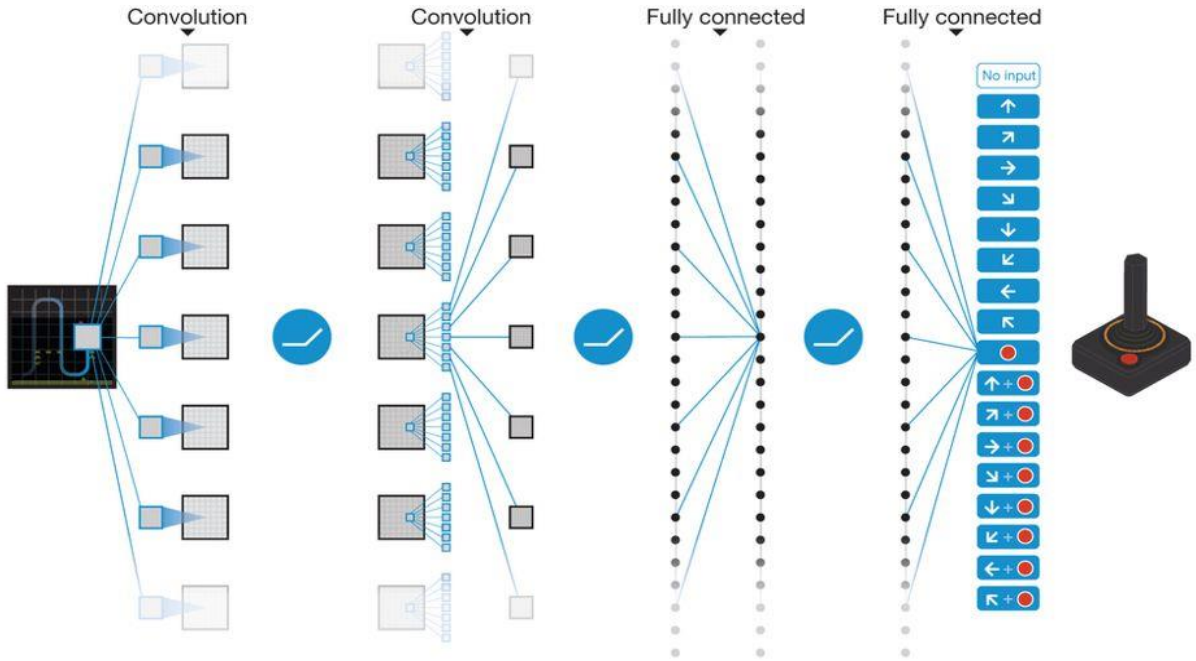
A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction



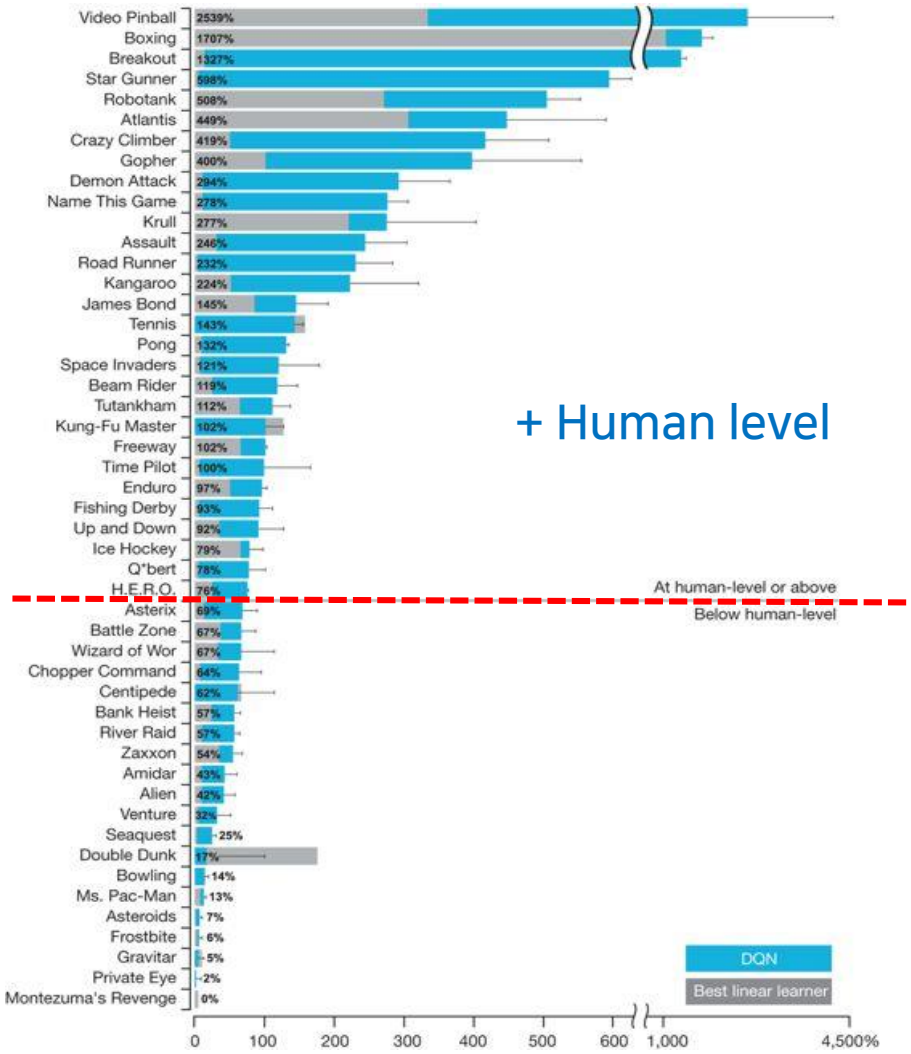
Sensors 2017, 17, x; doi:

딥러닝 강화학습은 인간 수준 넘어서야 함

Deep Reinforcement Learning



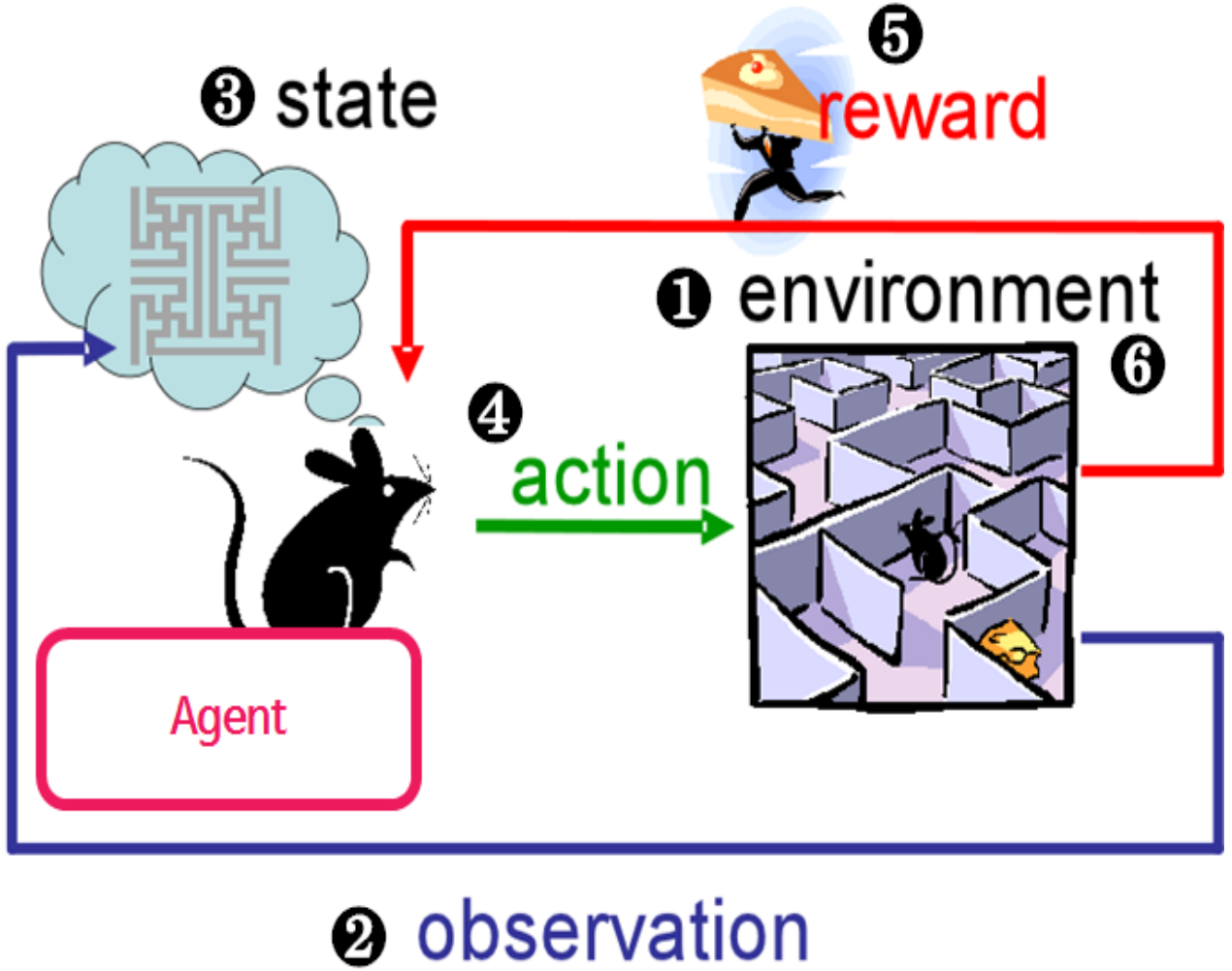
구글 딥마인드, Nature 2015



+ Human level

강화학습 이해

- 에이전트(Agent)
- 환경(Environment)
  - 상태(State)
  - 행동(Action)
  - 보상(Reward)
  - 정책(Policy)



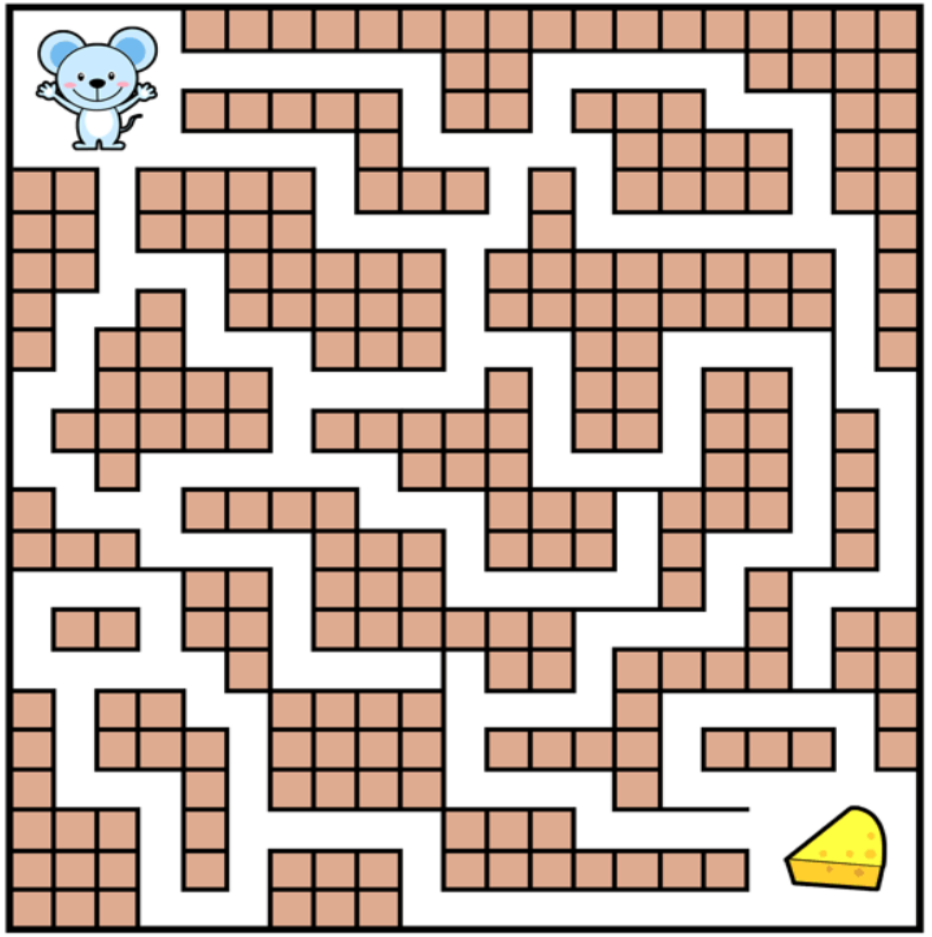
❖ 마르코프 의사 결정 과정 MDP (Markov Decision Processe)

$$\mathcal{H}_t \rightarrow S_t \rightarrow \mathcal{H}_{t+1}$$

시간(t)는 현재의 시간, 상태(S)는 현재의 상태, 메모리(H)는 현재의 기억

Markov Decision Process (MDP) $\sim \{S, A, P, R, \gamma\}$

강화학습 이해하기. (실습)



① 환경(E) : 미로 (설계자는 답을 알고 있다)
② 에이전트 : 쥐

③ 행동(A) : 위, 아래, 왼쪽, 오른쪽
④ 상태(S) : 시간(t)에서 에이전트 주위 정보
⑤ 보상(R) : 치즈 (미로 탈출 수 얻음)
⑥ MPD를 적용해보자.
⑦ 정책(P) : 보상을 최대화 할 수 있는 행동들

⑧ 매 순간 Q-에게 물어보자. Q(s,a)는 신경망으로 근사하여 계산. Deep Q-Network (DQN)

큐(Q)에게 상담을 해보자.

심화강화학습의 기본 알고리즘
DQN (Deep Q Network) : 불연속적인 (Descrete Action) 행동을 제어할 수 있는 알고리즘

큐(Q)는 미로의 구조를 알고 있다고 가정!

보상을 최대!

바로 직전 행동(t)

상태(위치, 좌표)

$$\hat{Q}(s,a) \Leftarrow r + \max_{a'} \hat{Q}(s',a')$$

새로운 행동(t+1)을 알려줌

Q Learning (Watkings89)

- Learn Q-function $Q(s_t, a_t) = E[R_t]$
- Bellman equation $\hat{Q}(s, a) = r + \gamma \max_{a' \in A} \hat{Q}(s', a')$

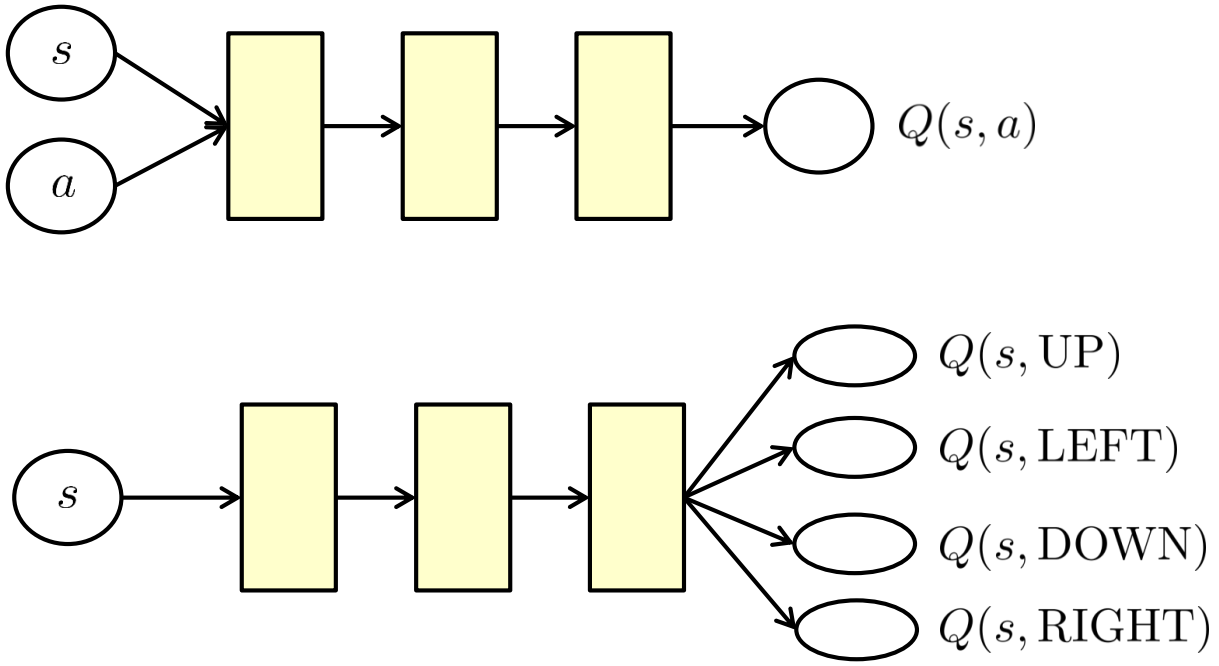
- Estimate optimal Q using learning rate

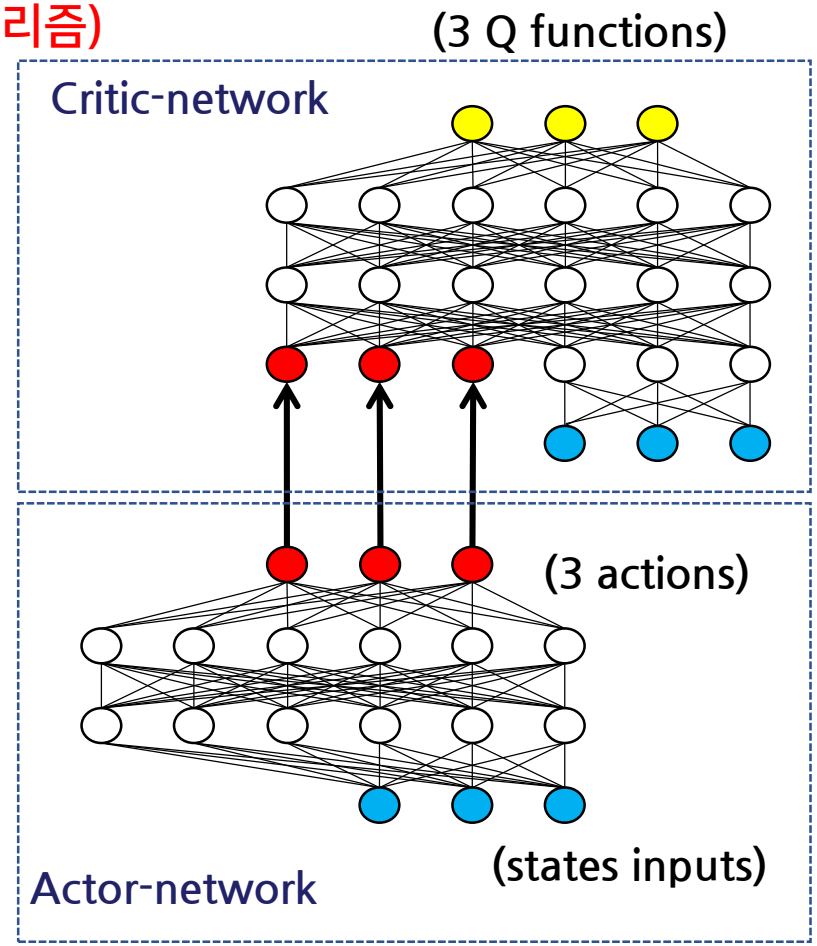
$$\hat{Q}(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha [r + \gamma \max_{a' \in A} \hat{Q}(s', a')]$$

- Optimal policy

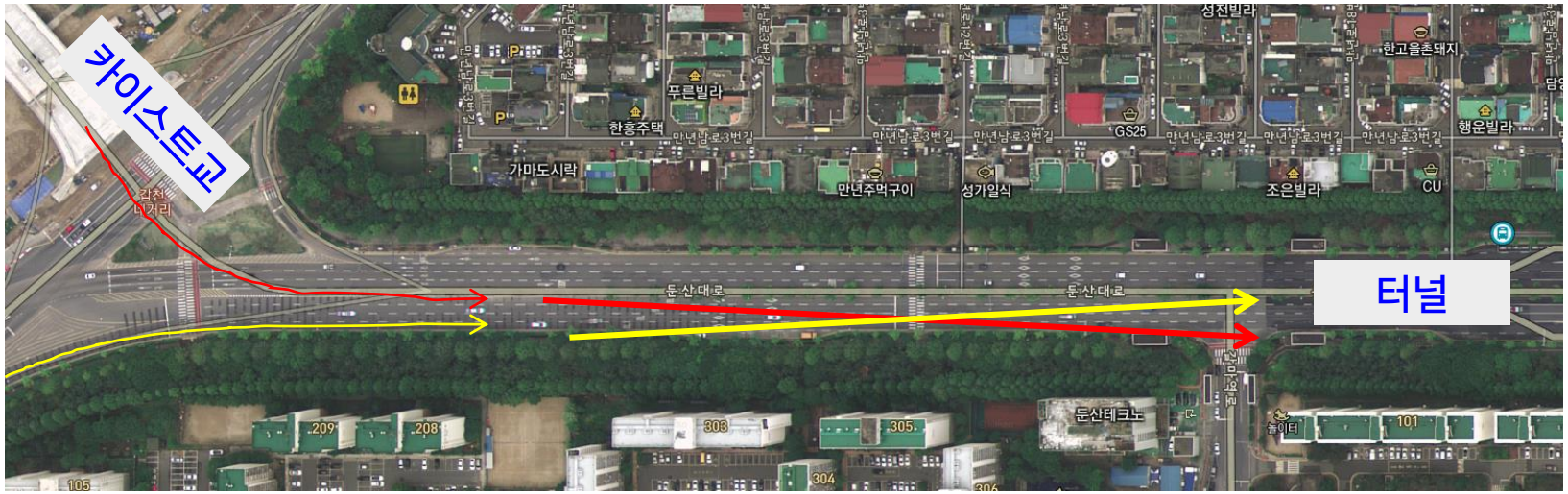
$$\pi^*(s) = \operatorname{argmax}_{a \in A} Q(s, a)$$

심화강화학습의 기본 알고리즘 : 연속적인 행동 제어
Actor-Critic 알고리즘 기반의 DDPG (심화 불연속적인 정책 경사 알고리즘)

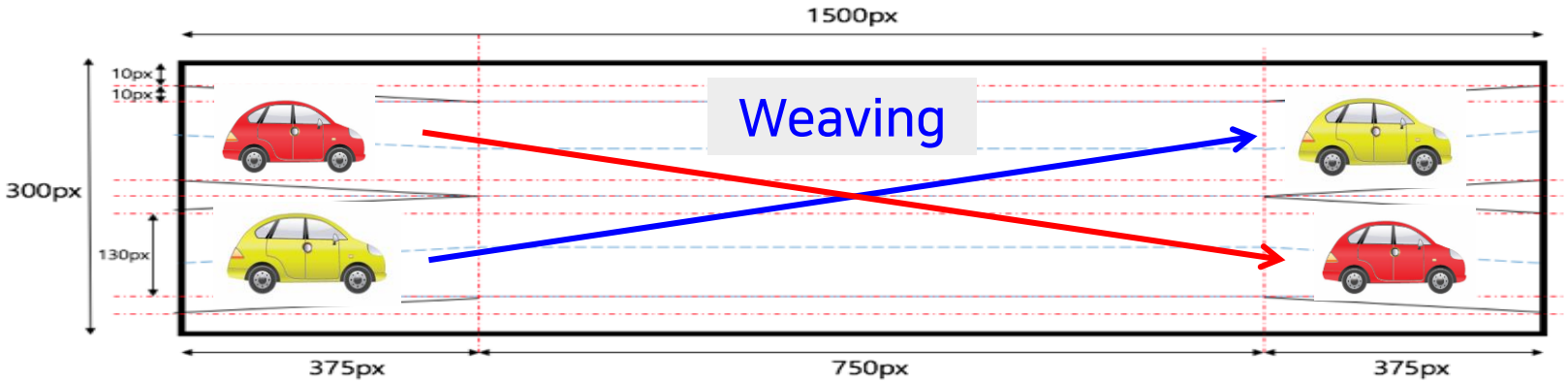




Weaving 구간 (카이스트교) : 차선 변경이 빈번히 발생하는 위험도로 구간



강화학습을 위한 환경(Environment) 설정



강화학습 멀티에이전트 : 환경, 상태, 행동

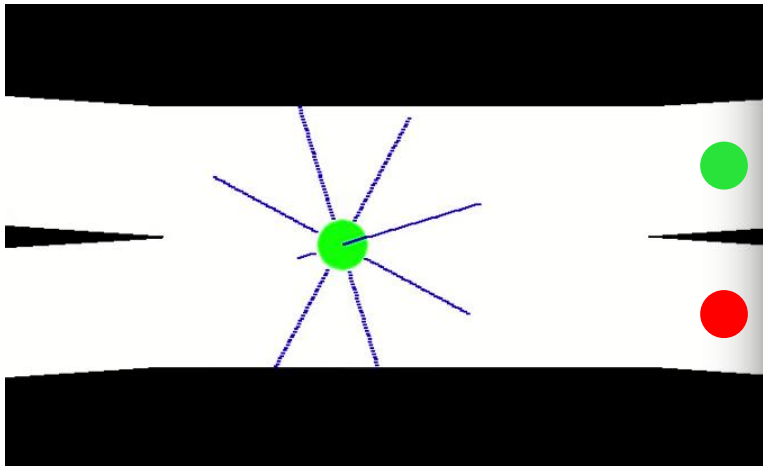
➢ 3-Actions

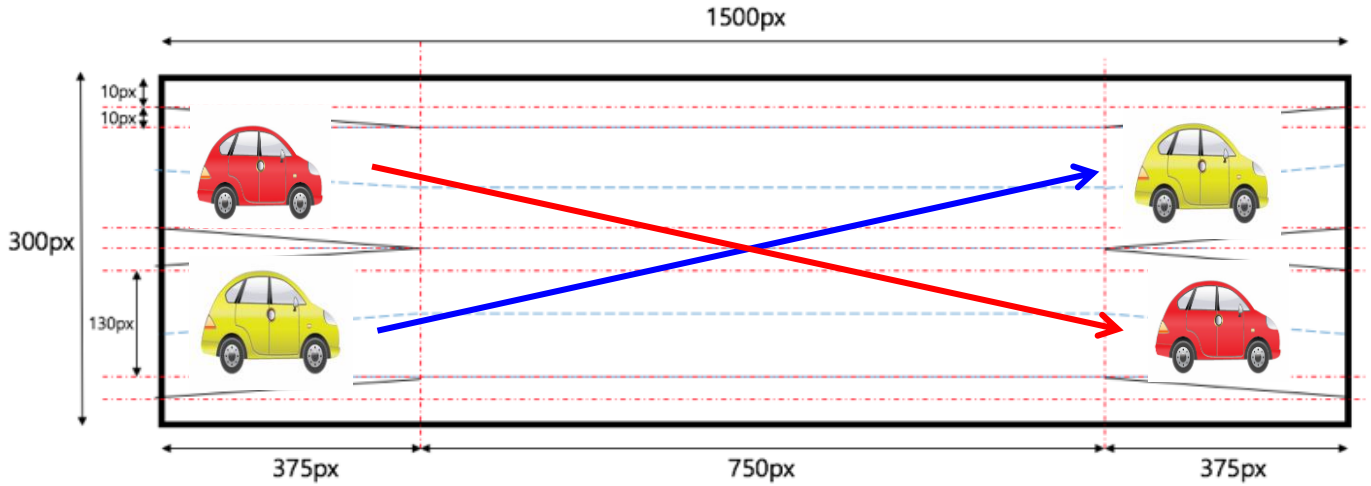
- Steering ∈ [-1, 1] (tanh)
- Acceleration ∈ [0, 1] (sigmoid)
- Brake ∈ [0, 1] (sigmoid)

➢ 12-States

- car body positions : x, y to ∈ [-1,1]
- car angle(radian) ∈ [-0.78,0.78]
- car velocity ∈ [0, 1]; car colors : 0 or 1
- 7 Sonar sensors [-90˚~+90˚] : ∈[0,1]

➢ Environment



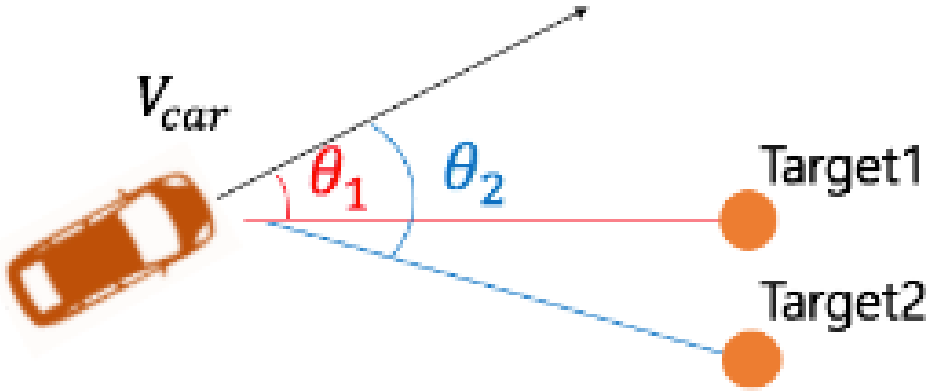


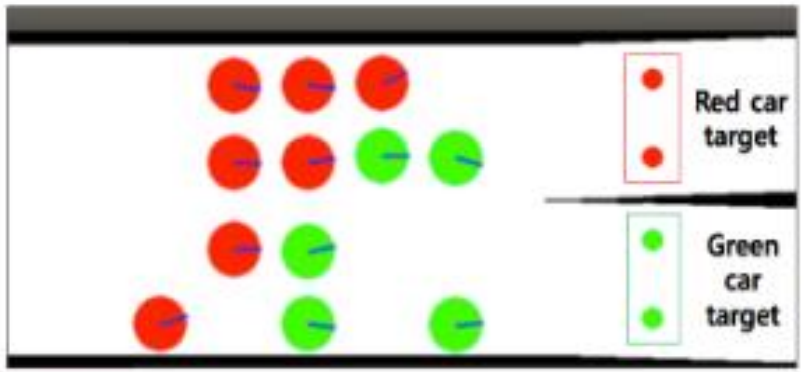
보상(Reward) 설계: 가장 주요함

- **Driving**

$$R = \max(V_{car}\cos(\theta_1), V_{car}\cos(\theta_2))$$

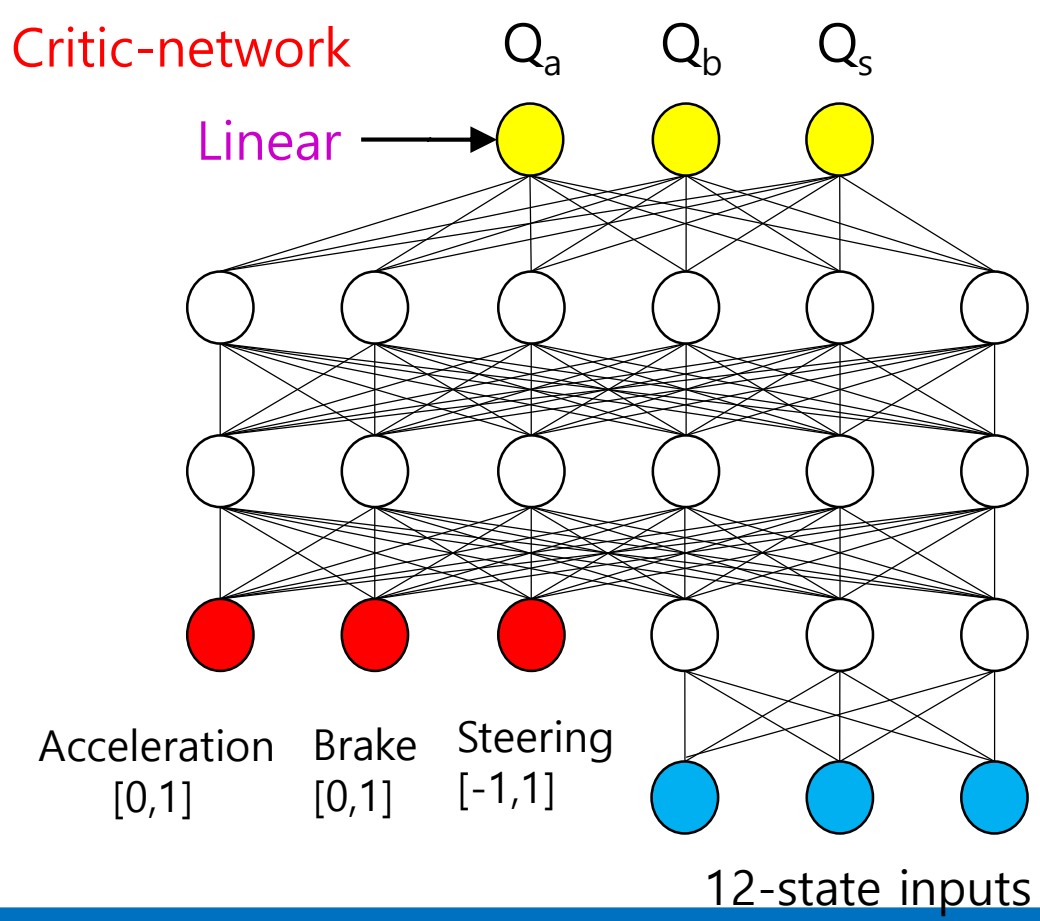
- **Got to Target**
  - ✓ +500 : colors are matched
  - ✓ -500 : colors are mis-matched
- **Collision**
  - ✓ -600 : when a agent makes collision

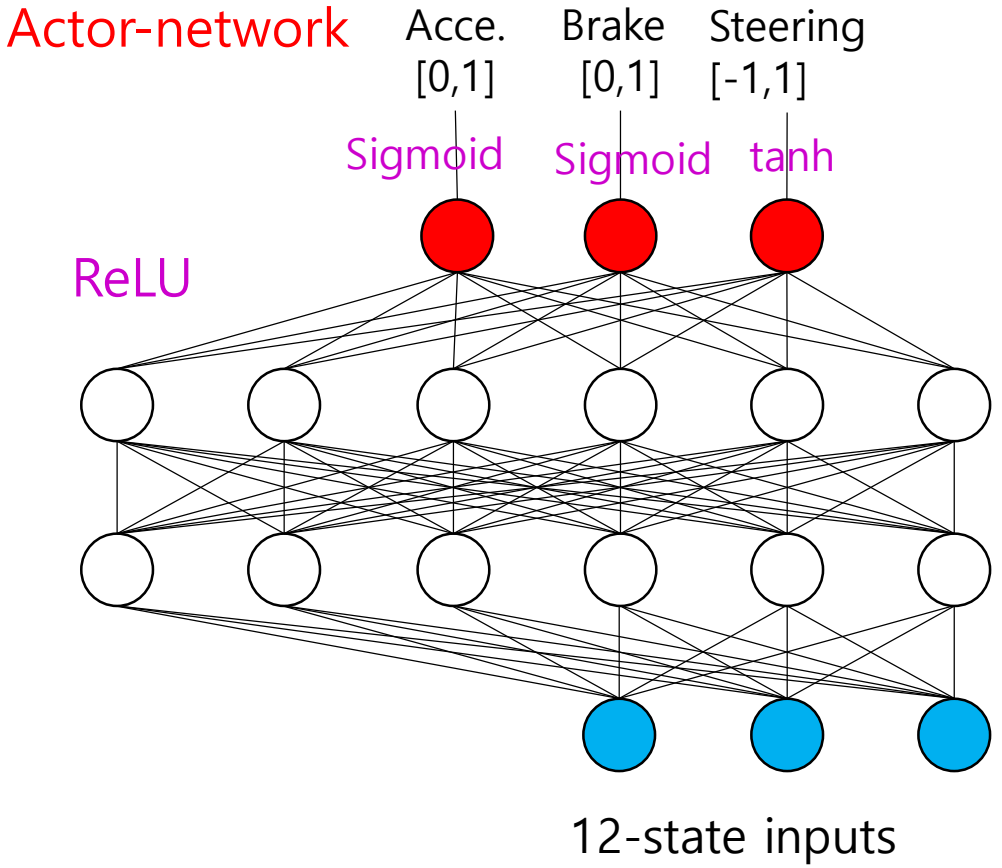




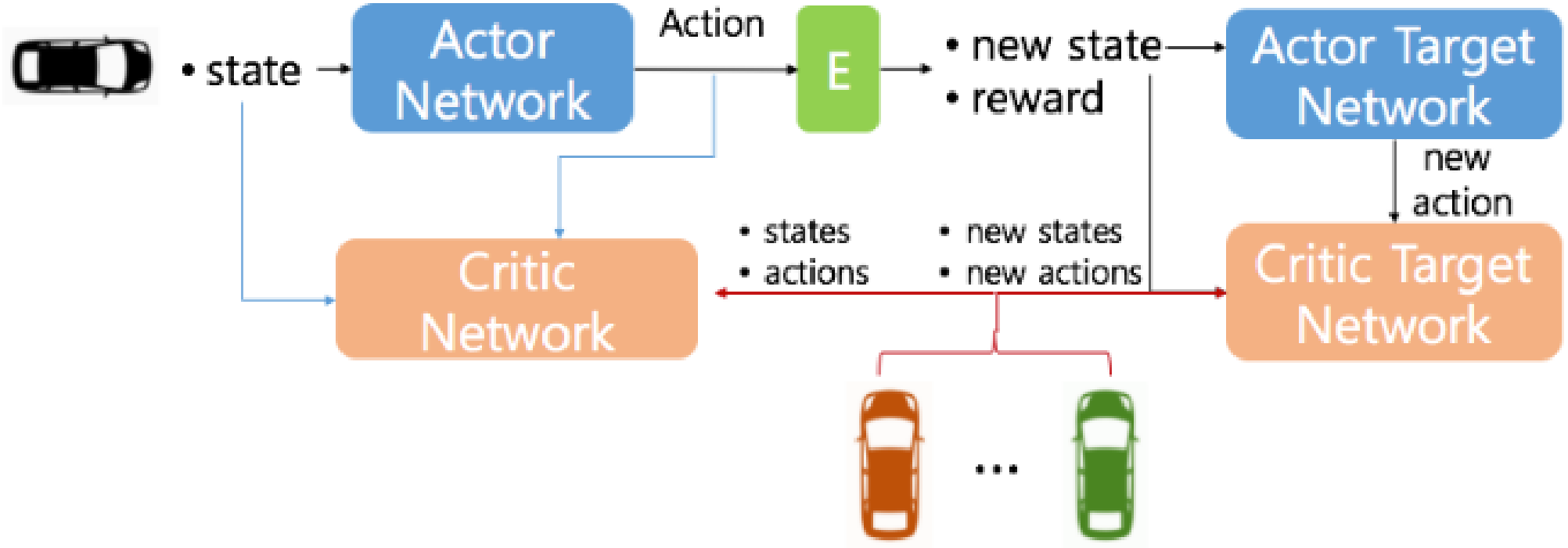
Actor-Critic Neural network Architecture

The critic model estimates Q-function values while the actor model selects the most optimal actions for each state based on those estimates.





- So we decide to apply this mechanism to our double merge scenario
  - ✓ All vehicles make their own reward to the maximum by sharing the same network parameters.



KISTI SINCE 1962

2022 TRUST KISTI

Korea Institute of Science and Technology Information