KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information

2023

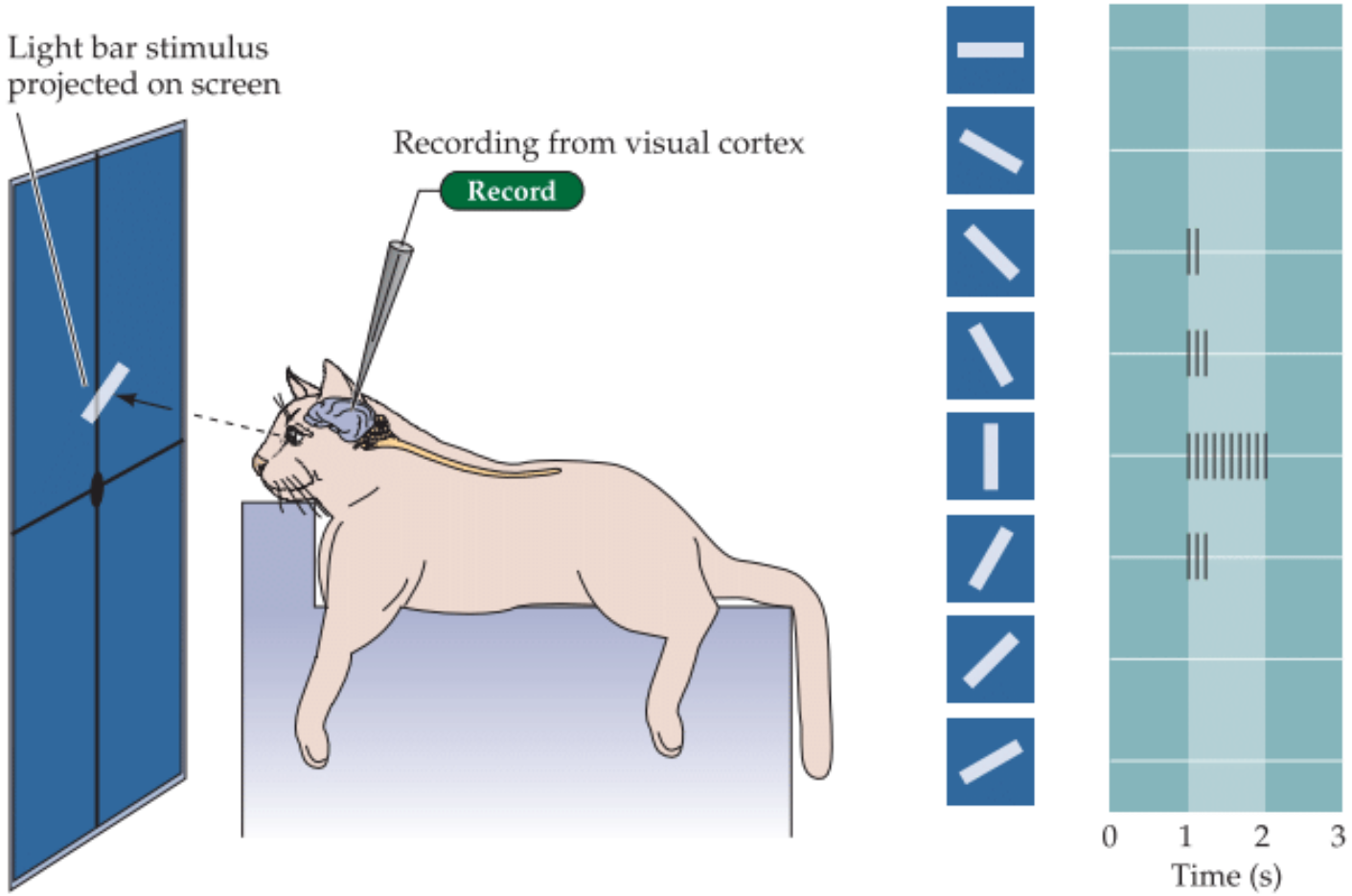
인공지능특론
(Advanced Artificial Intelligence)
15주 합성곱 신경망과 활용

Hongsuk Yi KISTI

합성곱 신경망
(Convolutional Neural Networks)

고양이의 시각피질

- Hubel과 Wiesel은 시각피질의 세포들을 여러 유형으로 구별 (1959)



시각피질

KISTI SINCE 1962

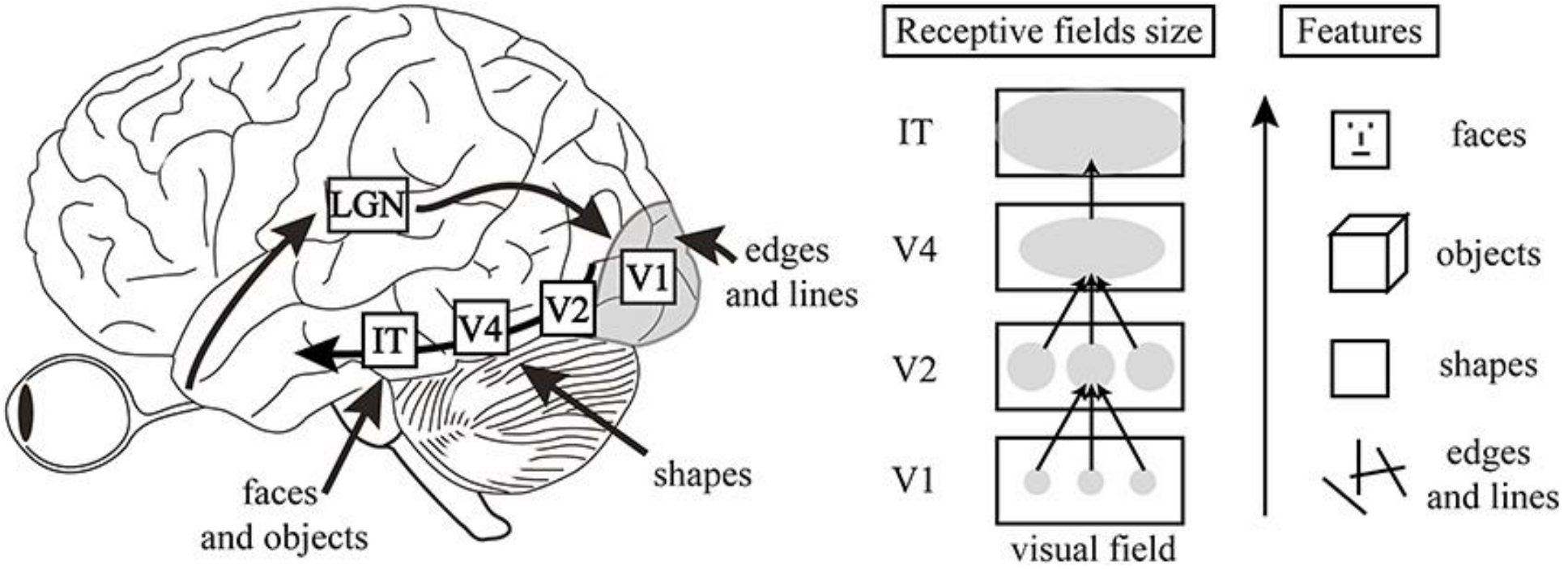
≡ 위키백과
우리 모두의 백과사전

시각피질(영어: visual cortex)은 V1으로 알려진 일차 시각피질(striate cortex)과 V2, V3, V4, V5로 알려진 영역(extrastriate cortex)을 통틀어 일컫는 말이다. 일차 시각피질은 브로드만의 뇌지도의 17번 영역과 해부학적으로 동일하다.

시각피질은 대뇌 반구 양쪽의 후두엽에 위치해 있다. 시각정보는 망막을 통해 시상을 거쳐 후두엽의 시각피질에 들어오기전 시각교차(optic chiasm)에 의해서 왼쪽 반구의 시각피질은 오른쪽 시야에서, 오른쪽 시각피질은 왼쪽 시야에서 신호를 받는다.[1]

사람 뇌의 시각피질(visual cortex)

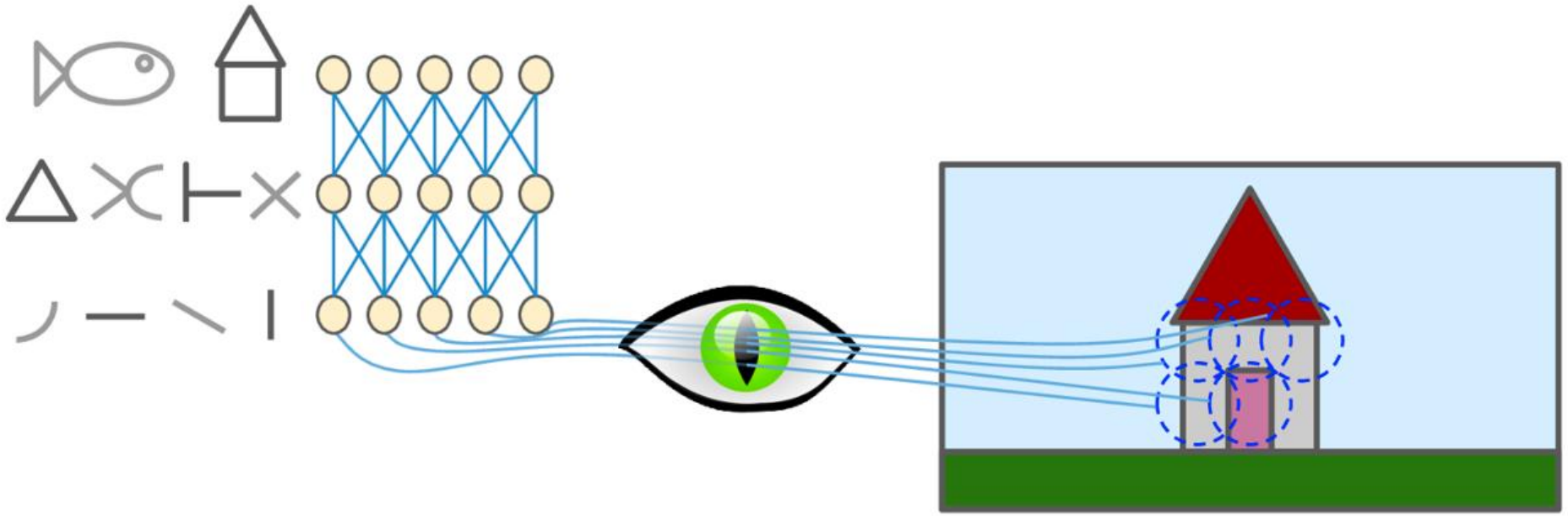
- Human visual cortex is hierarchical
  - ✓We want to learn higher level features by compositing lower-level features



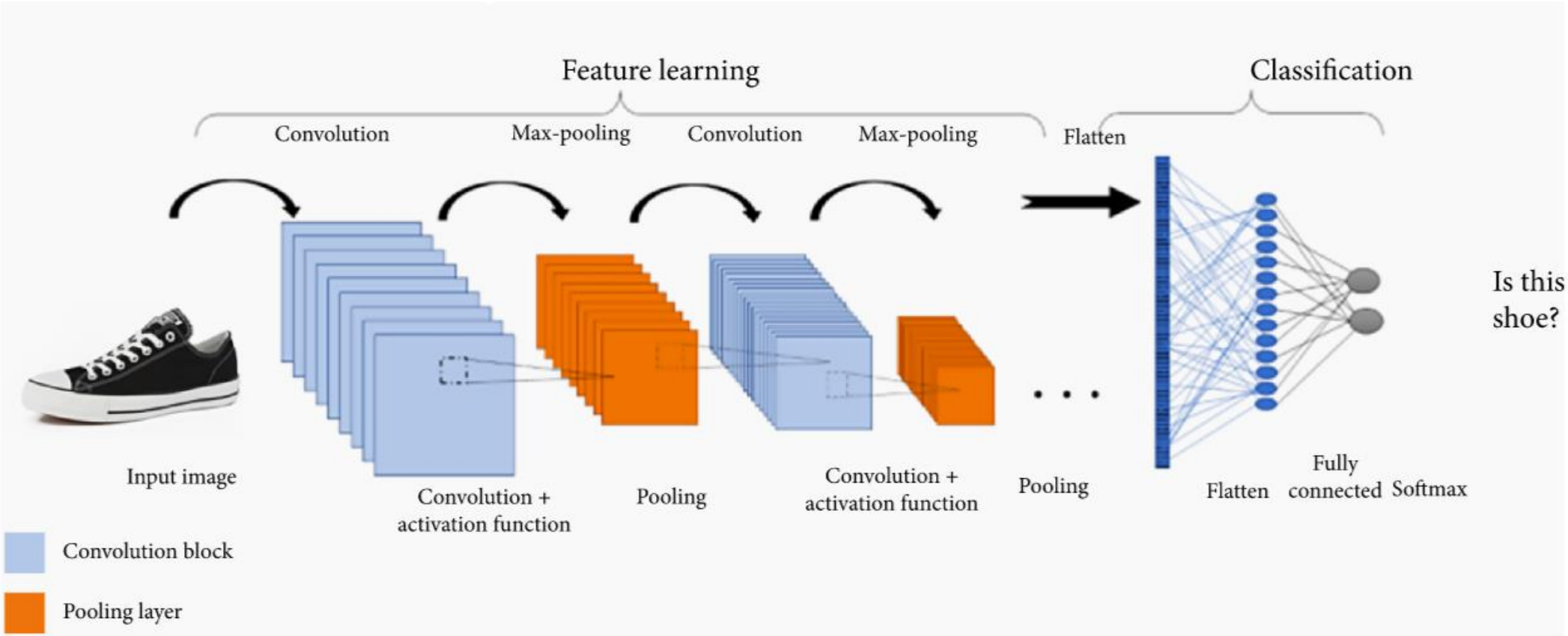
❖ 합성곱 신경망으로 발전

≡ 위키백과
우리 모두의 백과사전

합성곱 신경망은 뉴런 사이의 연결 패턴이 동물 시각 피질의 조직과 유사하다는 점에 영감을 받았다. 개별 피질 뉴런은 수용장(receptive field)으로 알려진 시야의 제한된 영역에서만 자극에 반응한다. 상이한 뉴런의 수용 필드는 전체 시야를 볼 수 있도록 부분적으로 중첩된다.[출처 필요]



CNNs for Image Classification



source : https://www.hindawi.com/journals/wcmc/2022/7549397/fig2/

Convolutional Neural Networks

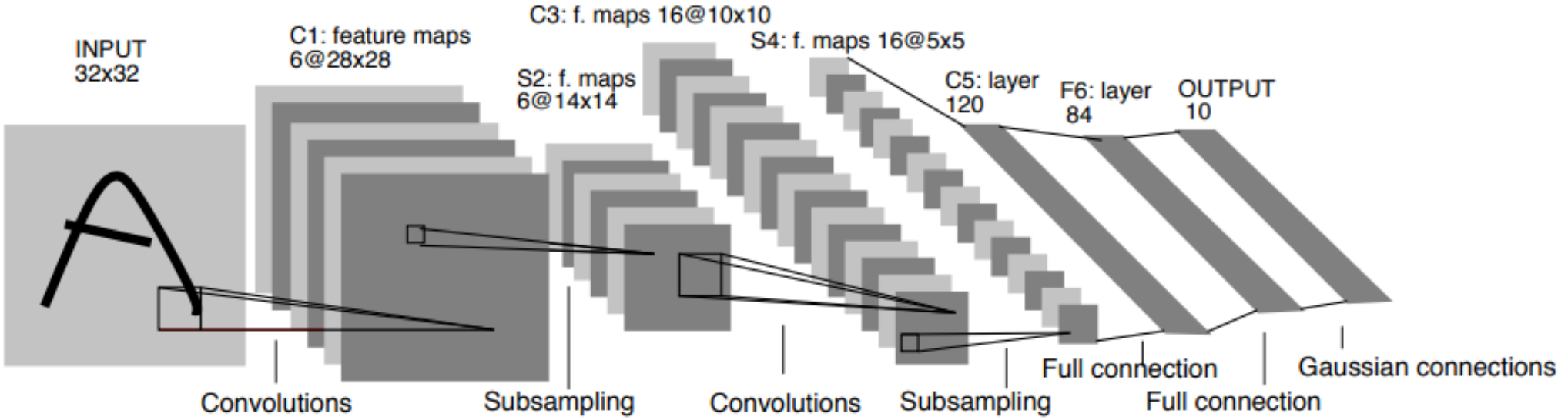
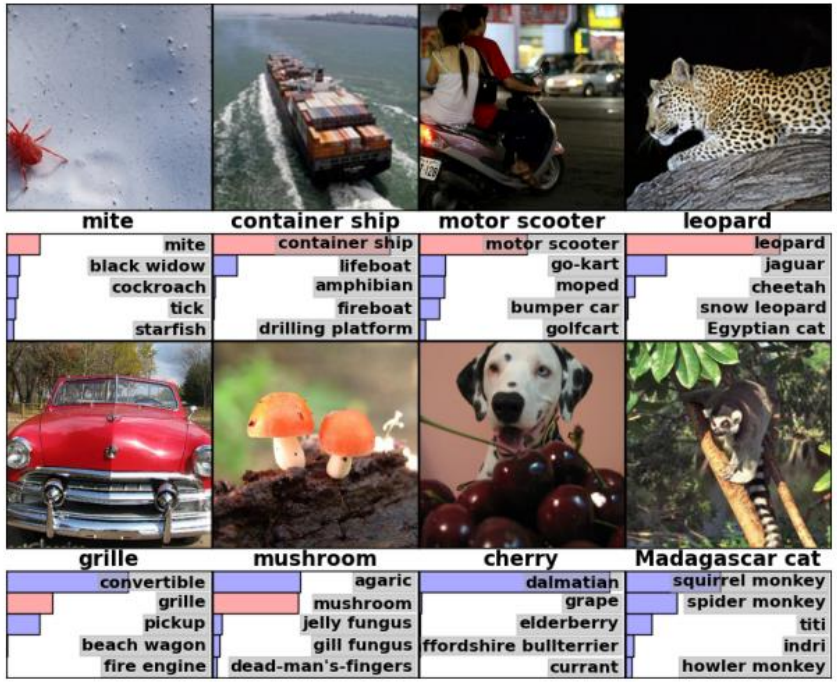


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

History of CNN and ILSVRC winners

- The ImageNet set that was used has ~1.2 million images and 1000 classes
- Accuracy is measured as top-5 performance:
  - ✓Correct prediction if the true label matches one of the top 5 predictions of the model



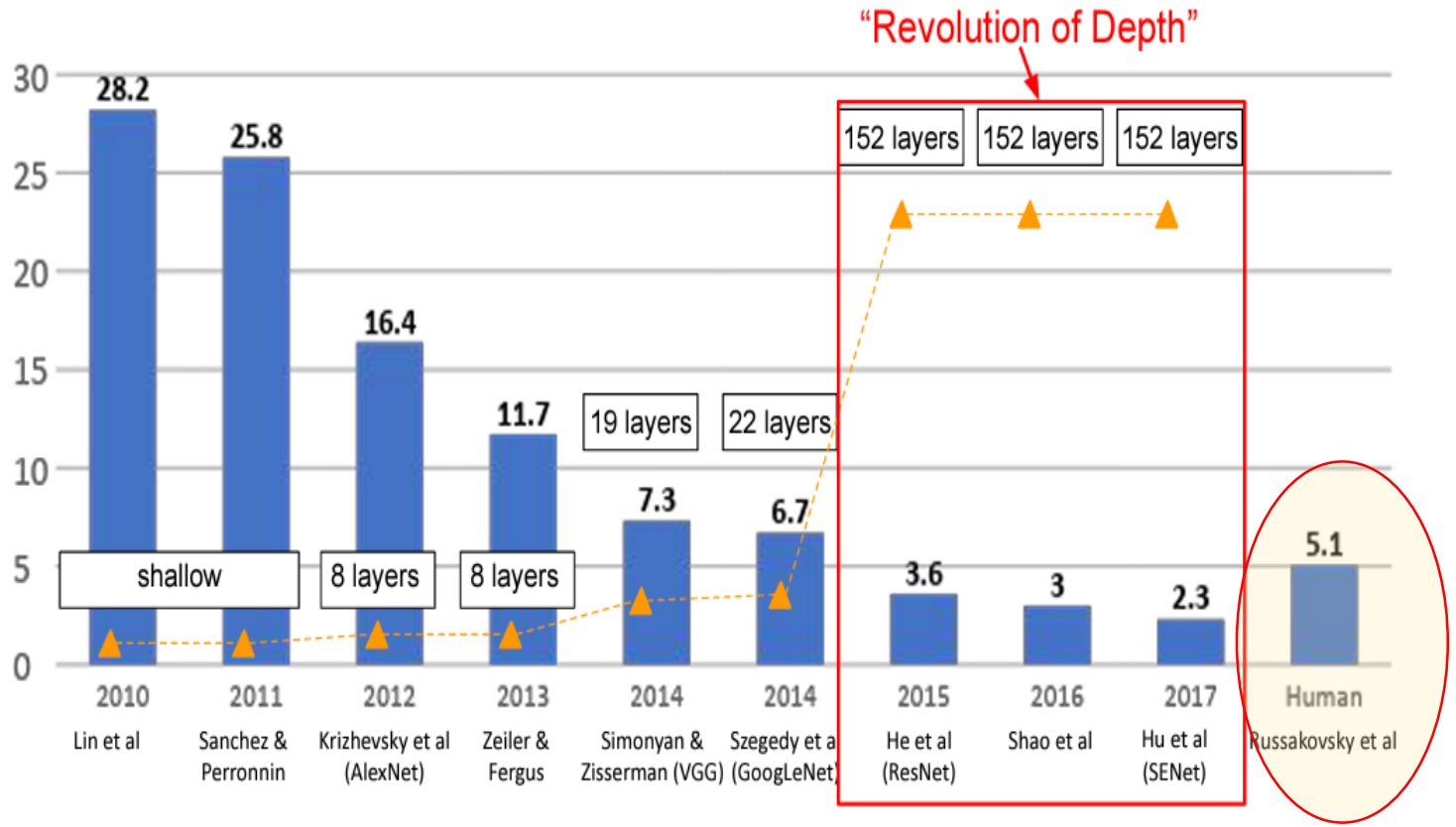
Note that the actual network inputs were still 224x224 images (random crops from downsampled 256x256 images)

224x224 is still a good/ reasonable size today (224*224*3 = 150,528 features)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Main Breakthrough for CNNs: AlexNet & ImageNet

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
  - ✓Human Error ~ 5.1%



source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

History of CNN

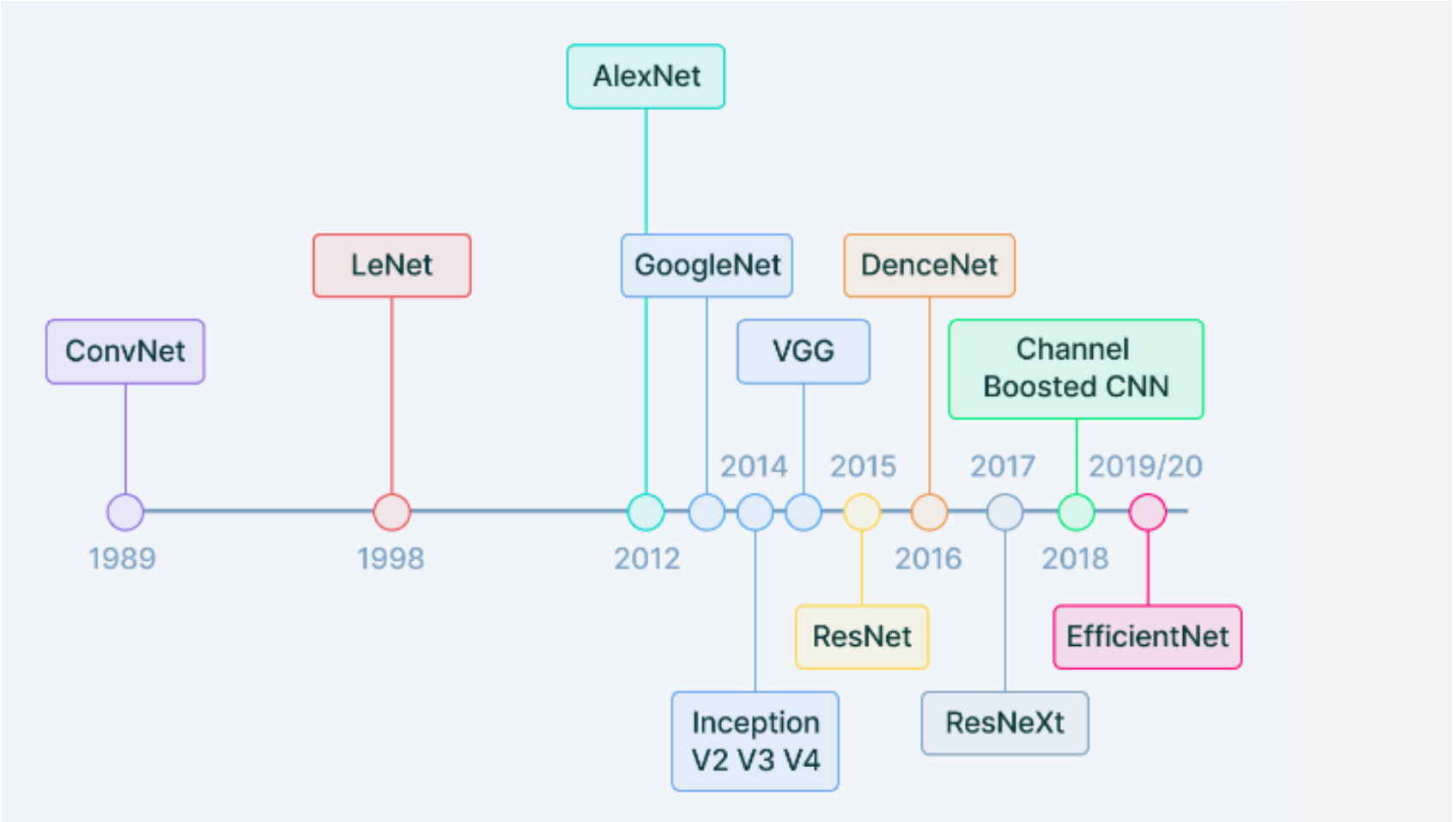
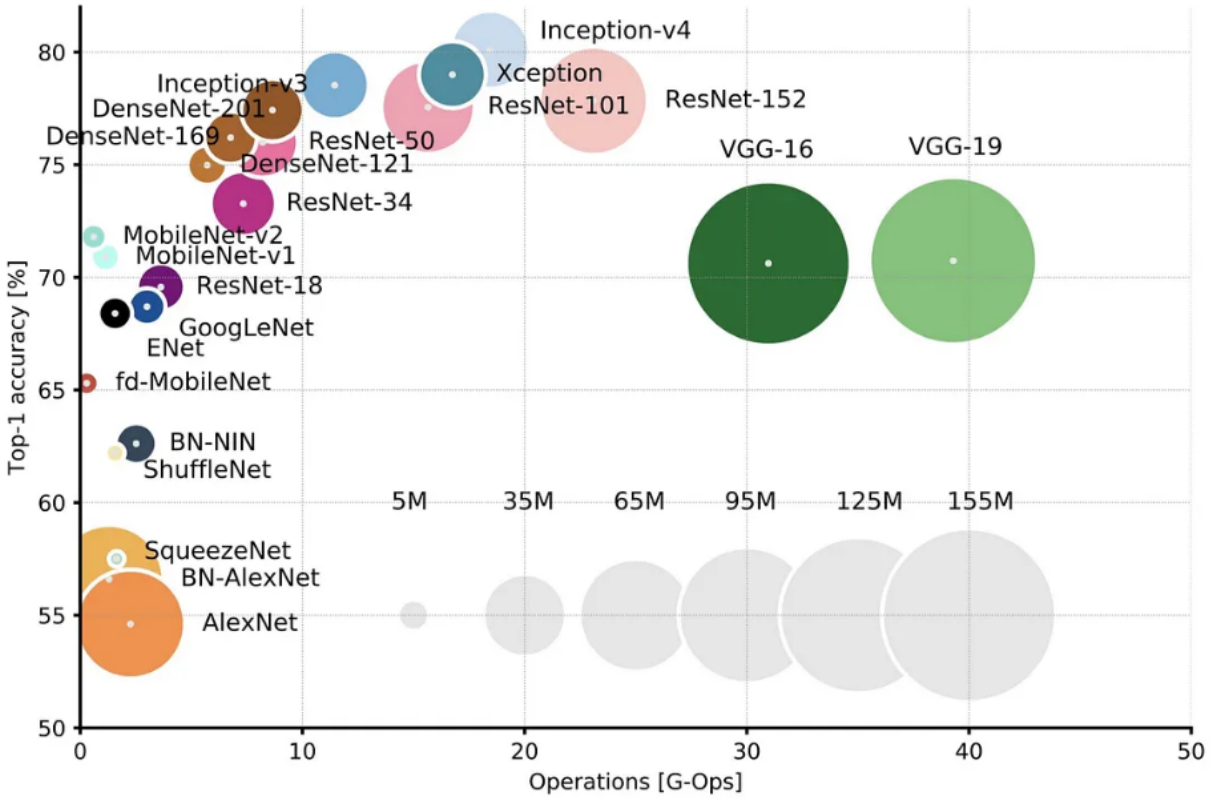


Figure source) https://www.v7labs.com/blog/convolutional-neural-networks-guide

Common CNN Architectures



Top1 vs. operations, size ∝ parameters. Top-1 one-crop accuracy versus amount of operations required for a single forward pass. See also here

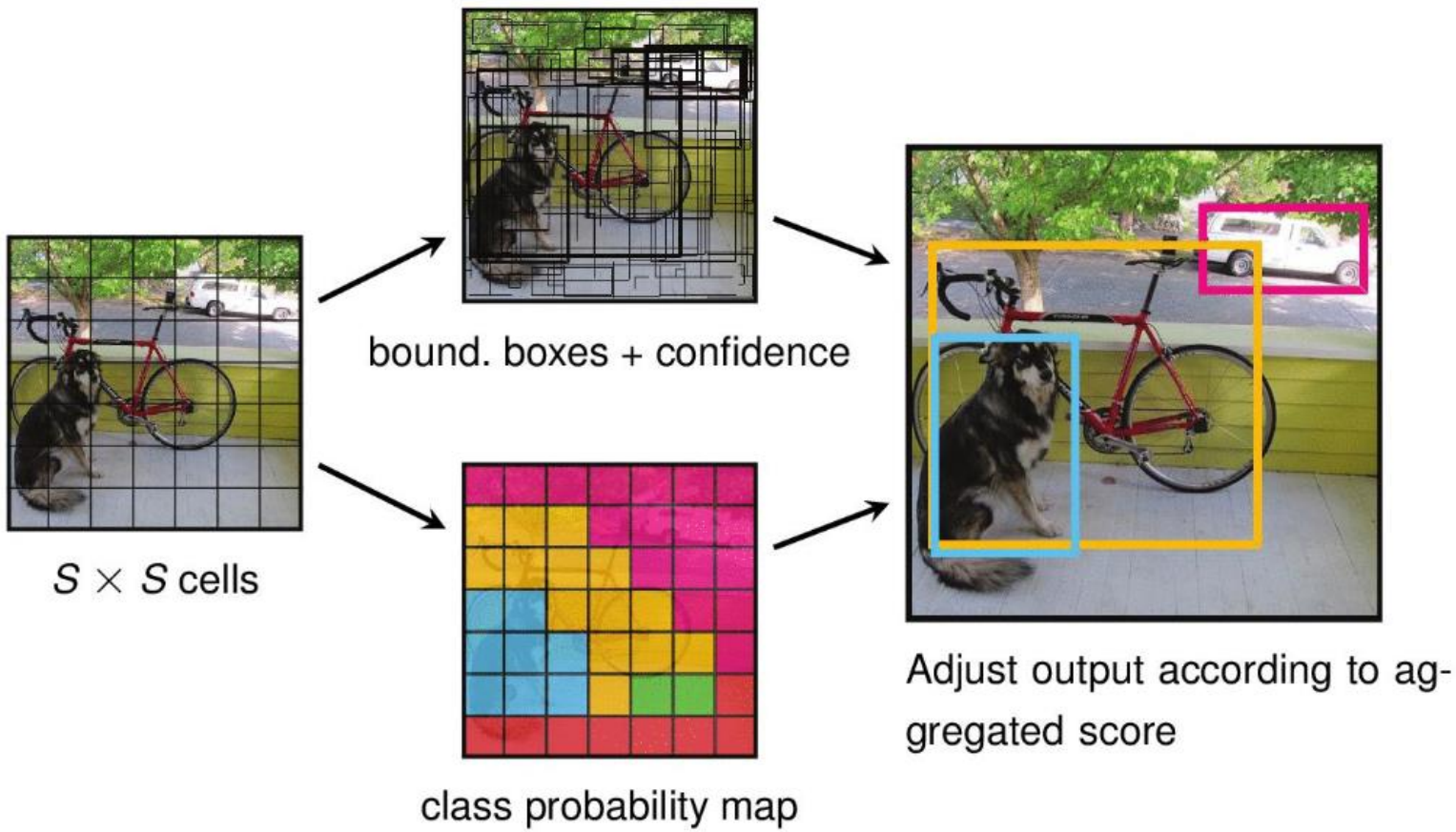
Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

Common Applications of CNNs

source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

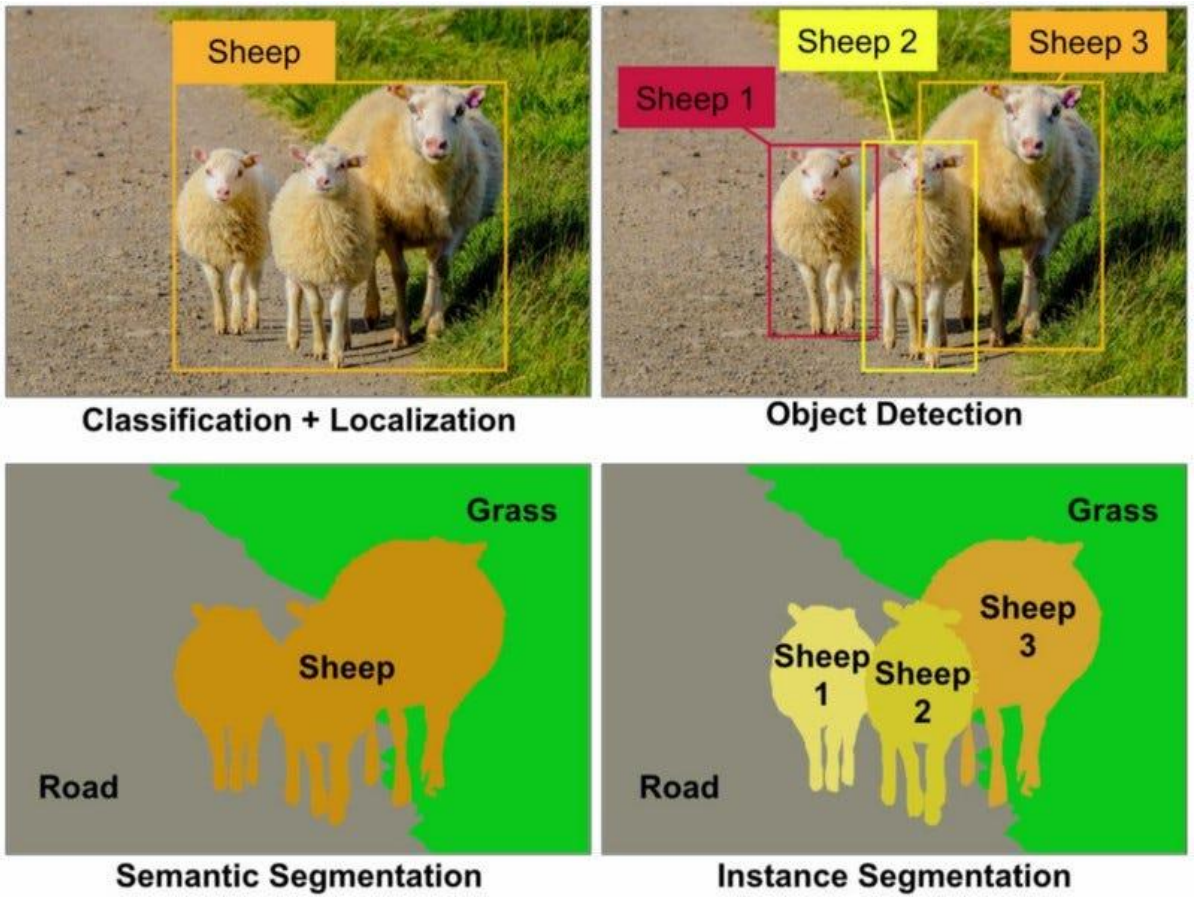
Object Detection and YOLO

You Only Look Once (YOLO) Algorithm



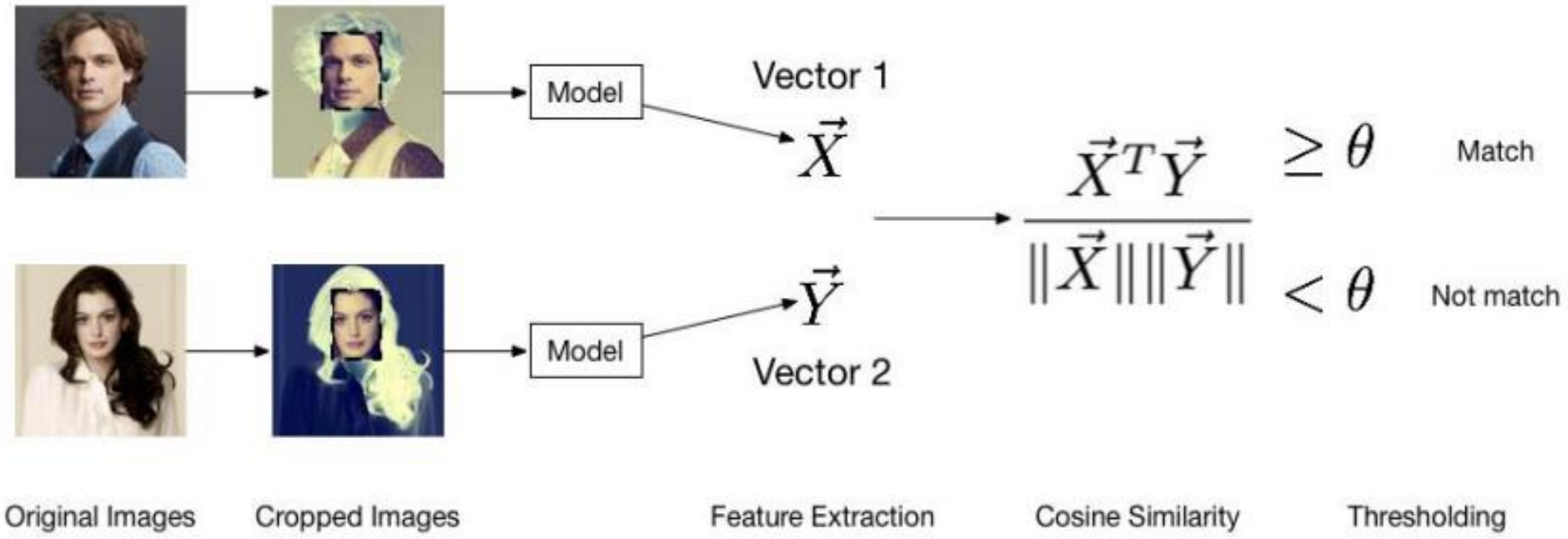
source : https://lme.tf.fau.de/lecture-notes/lecture-notes-dl/lecture-notes-in-deep-learning-segmentation-and-object-detection-part-4/

Object Segmentation



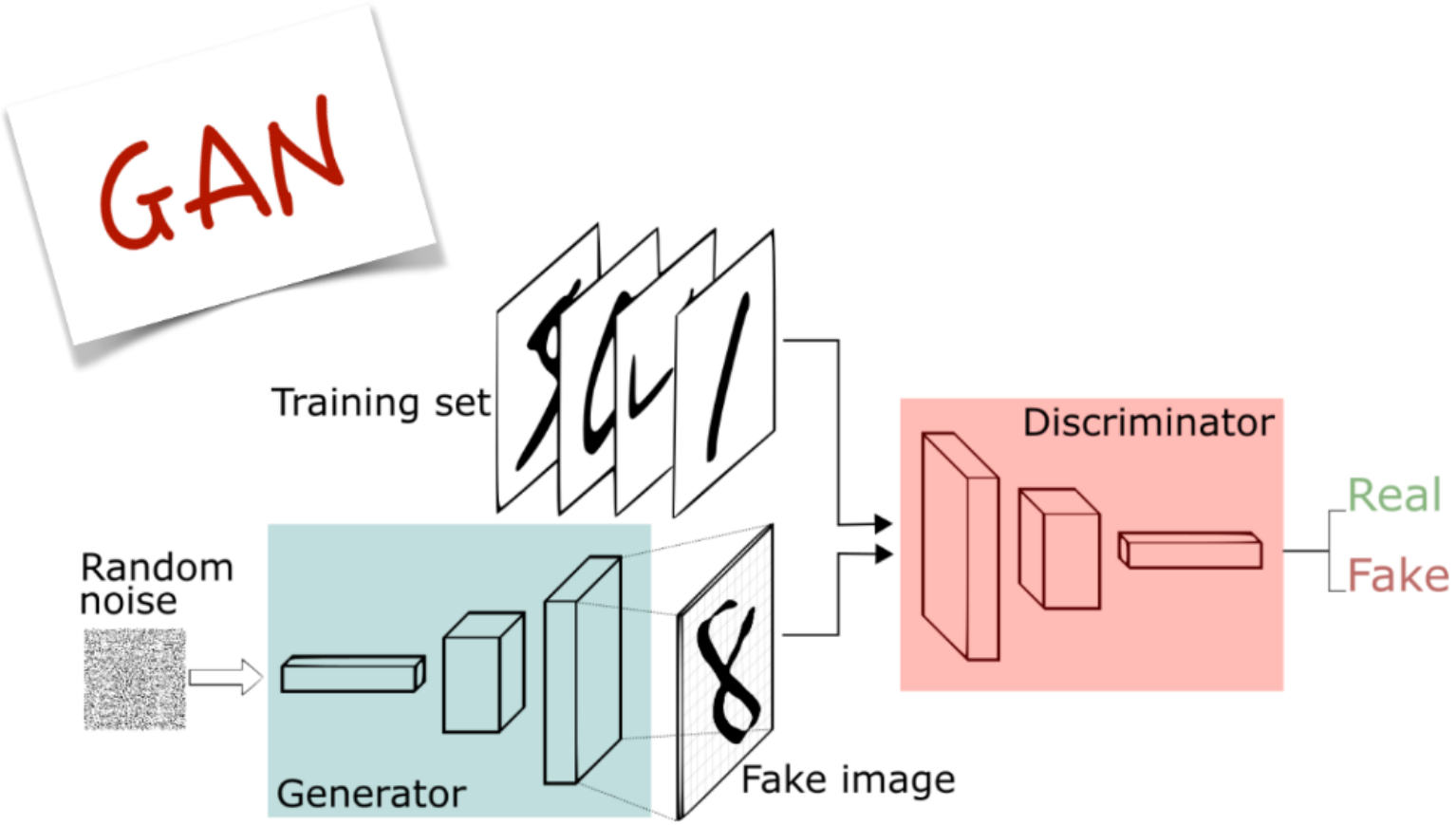
source https://nirmalamurali.medium.com/image-classification-vs-semantic-segmentation-vs-instance-segmentation-625c33a08d50

Face Recognition



source : https://arxiv.org/ftp/arxiv/papers/1910/1910.11563.pdf

Image Synthesis : Generative Adversarial Networks



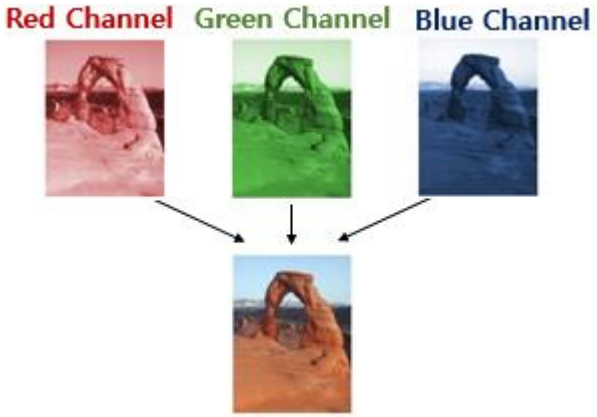
source : https://velog.io/@hwany/GAN

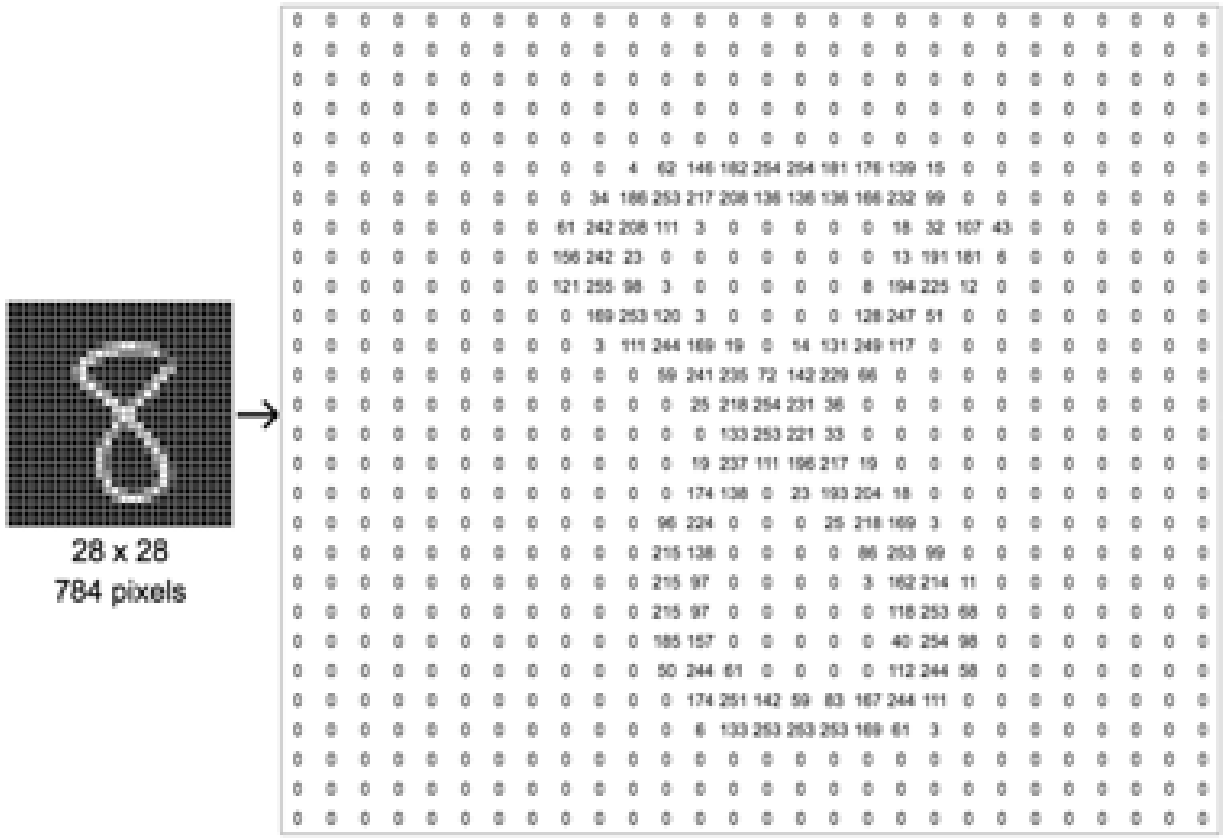
Why Computer Vision is (was) Hard: Challenges with Image Classification

source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

A computer sees an image as a matrix of numbers

- Any real-world image pixels
  - ✓Rows*columns*number of channels
  - ✓Example number 8: 28*28*1 -> 784
- Channel is Color
  - ✓Channel : RGB
  - ✓Value: 0~255





(source) https://wikidocs.net/64066

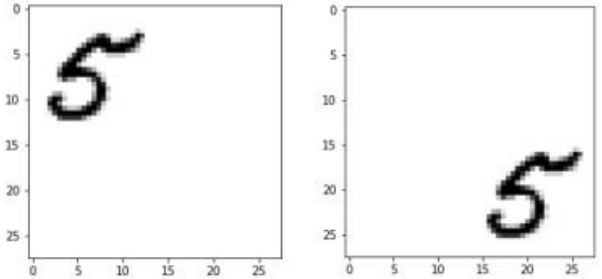
Why Image Classification is Hard

Different lighting, contrast, viewpoints, etc.

Image Source:
twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

Image Source: https://www.123rf.com/photo_76714328_side-view-of-tabby-cat-face-over-white.html

Or even simple translation

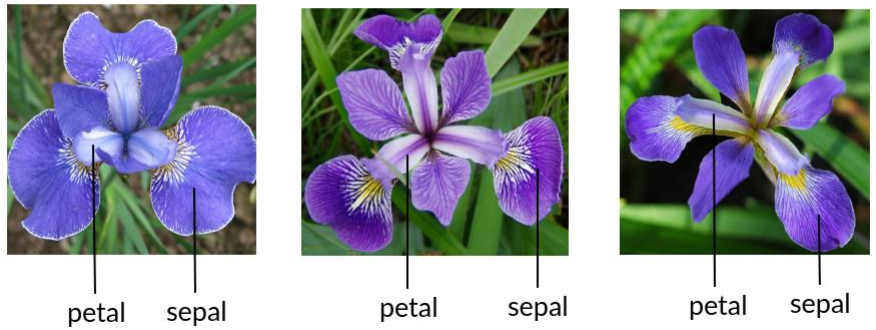


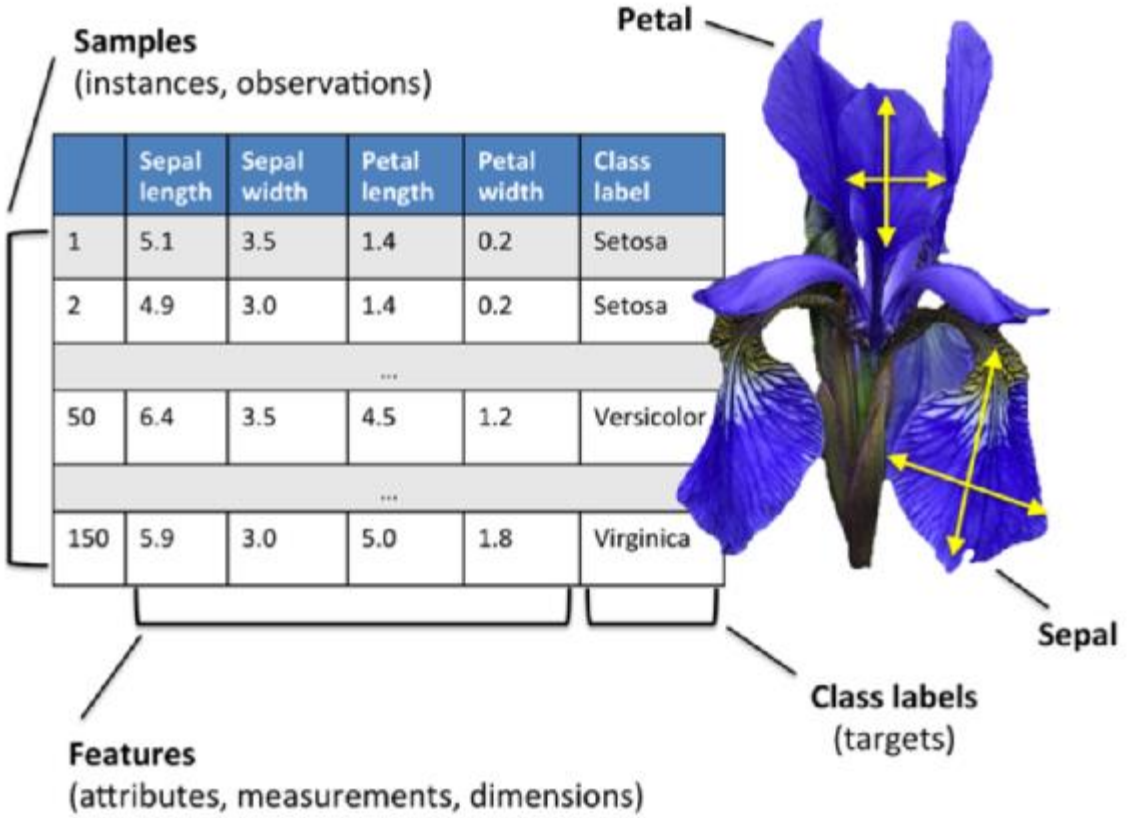
This is hard for traditional methods like multi-layer perceptrons, because the prediction is basically based on a sum of pixel intensities

source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

Traditional Approaches

- Use hand-engineered features of Iris flower data set



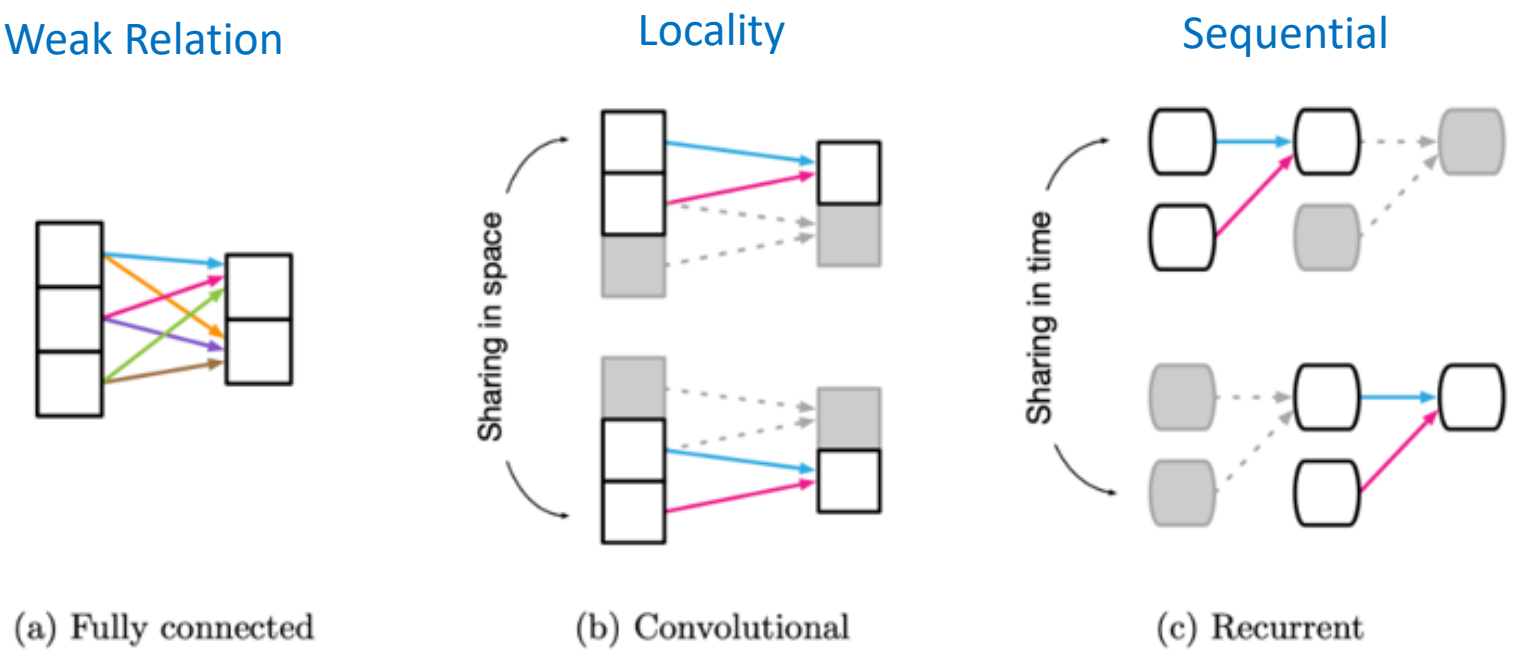


	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

source : https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_features_extraction.php

❖ Definition of Relational inductive biases

Every machine learning model requires some type of architecture design and possibly some initial assumptions about the data we want to analyze. **Generally, every building block and every belief that we make about the data is a form of inductive bias**



source : https://www.baeldung.com/cs/ml-inductive-bias

Relational inductive biases in deep learning

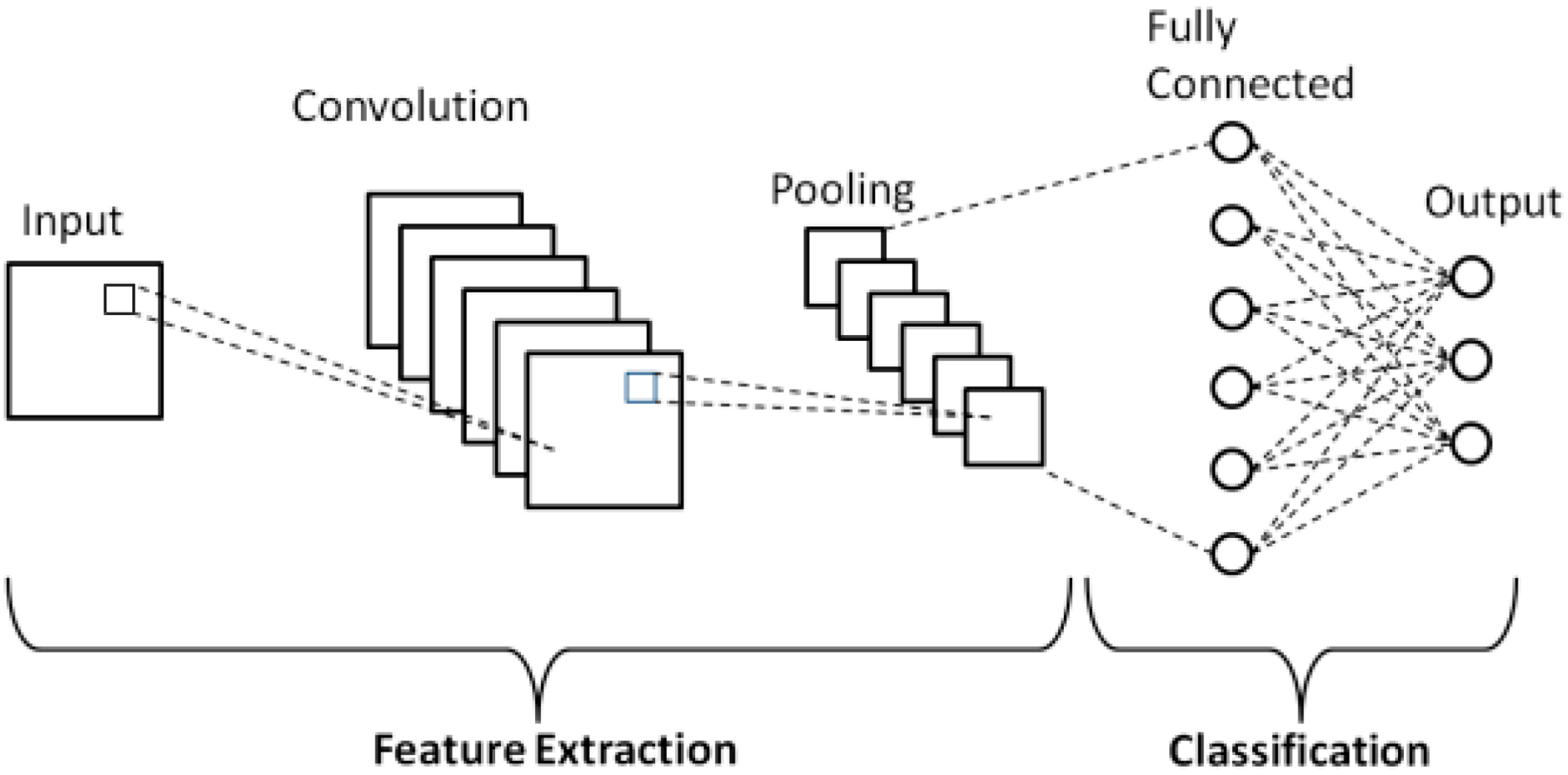
Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

source : https://enfow.github.io/paper-review/graph-neural-network/2021/01/11/relational_inductive_biases_deep_learning_and_graph_netowrks/

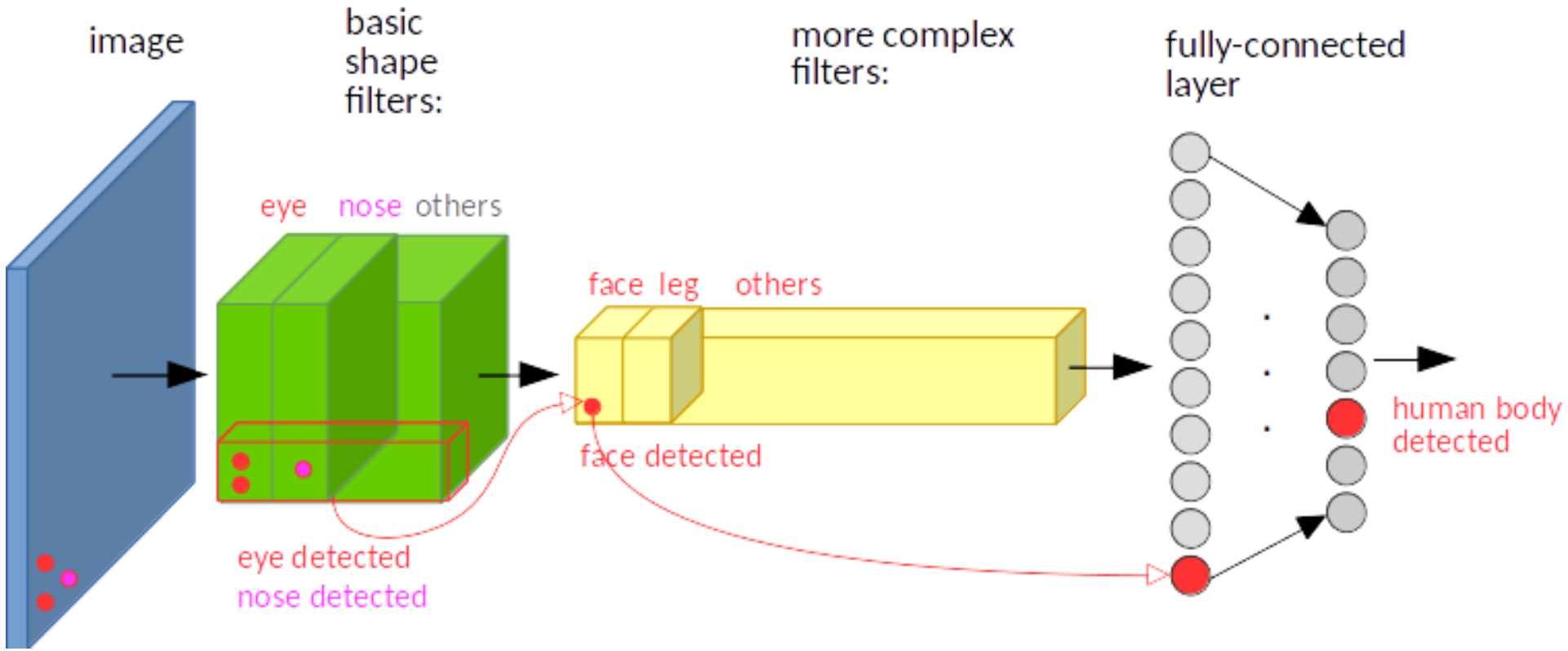
Convolutional Neural Network Basics

Schematic diagram of a basic CNN architecture

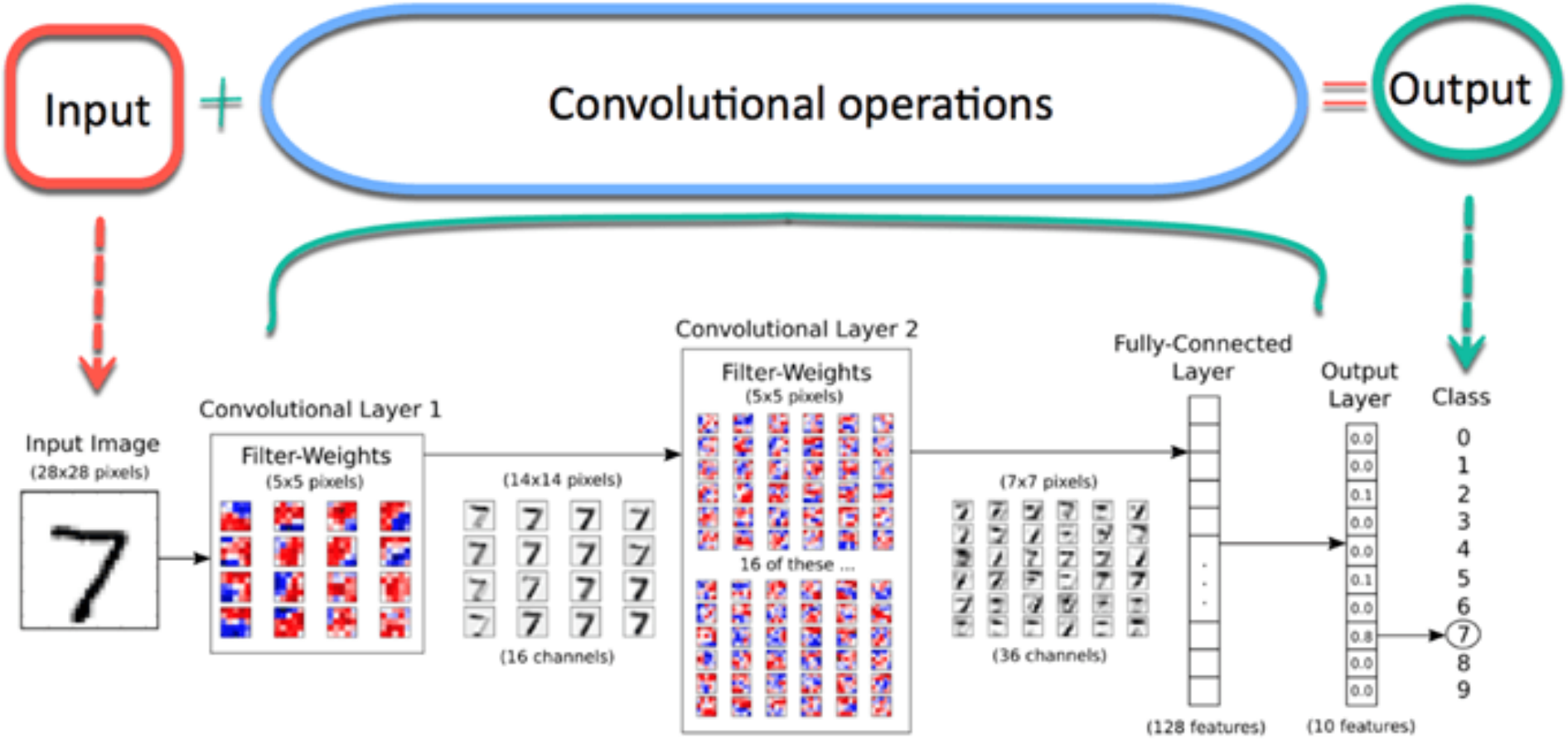


source : https://www.mdpi.com/2076-3417/9/21/4500

CNN Feature map



Architecture of a Convolutional Neural Network (CNN)



source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

What a CNN Can See

- Which patterns from the training set activate the feature map?

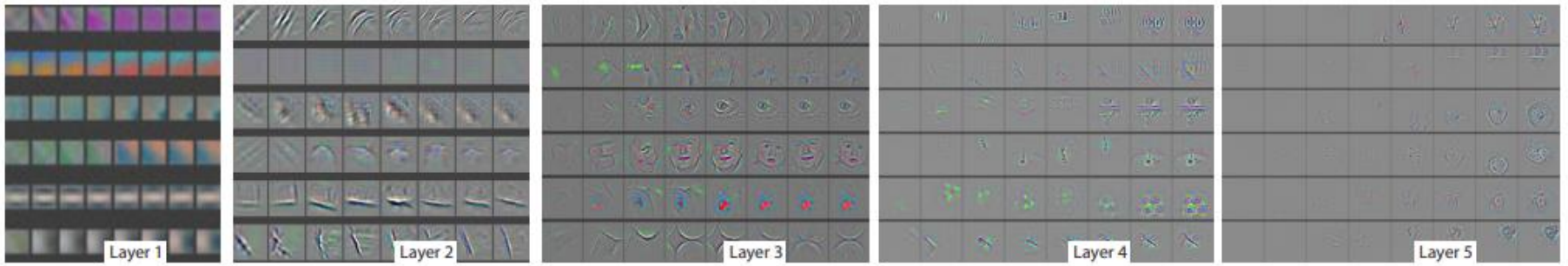


Fig. 4. Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, we show a randomly chosen subset of features at epochs [1,2,5,10,20,30,40,64]. The visualization shows the strongest activation (across all training examples) for a given feature map, projected down to pixel space using our deconvnet approach. Color contrast is artificially enhanced and the figure is best viewed in electronic form.

Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

Main Concepts Behind Convolutional Neural Networks

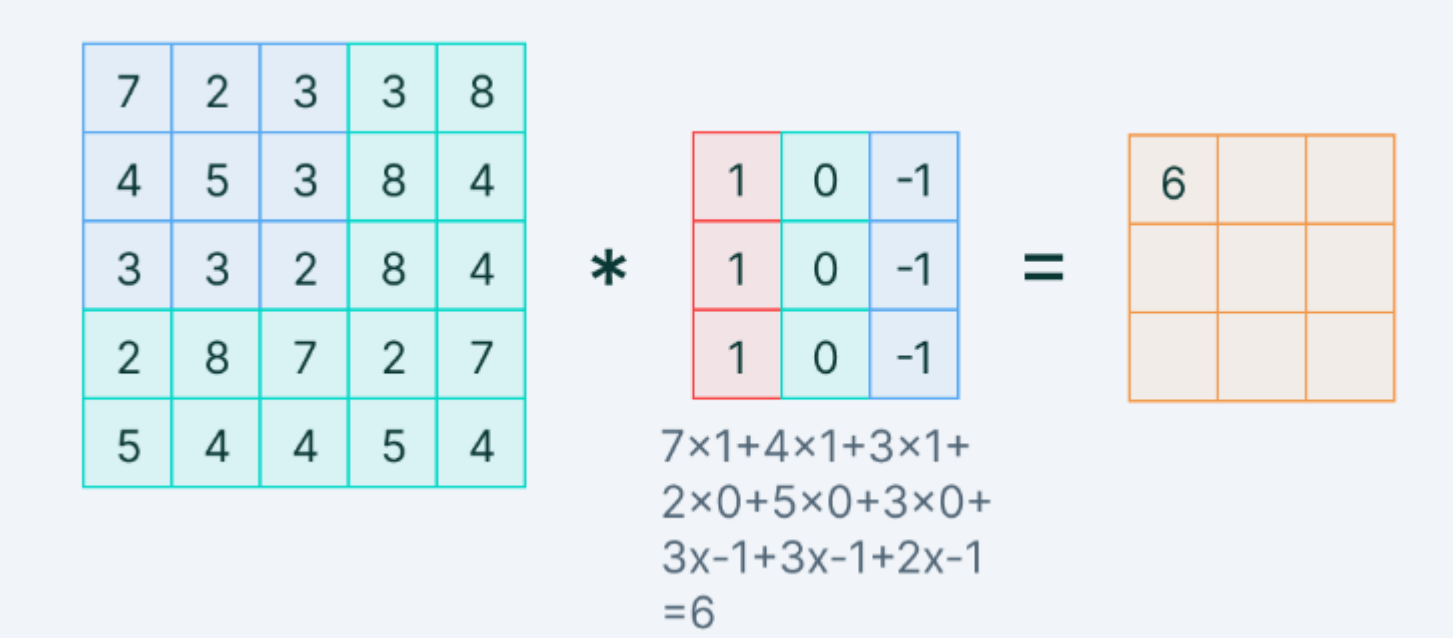
- **Local connectivity** refers to images represented in a matrix of pixel values
  - ✓we connect each neuron to only a patch of input data
  - ✓A single element in the feature map is connected to only a small patch of pixels.
- Parameter-sharing:
  - ✓The same weights are used for different patches of the input image.
- Many layers:
  - ✓Combining extracted local patterns to global patterns

source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

CNN works by comparing images piece by piece

❖ Convolution Operation

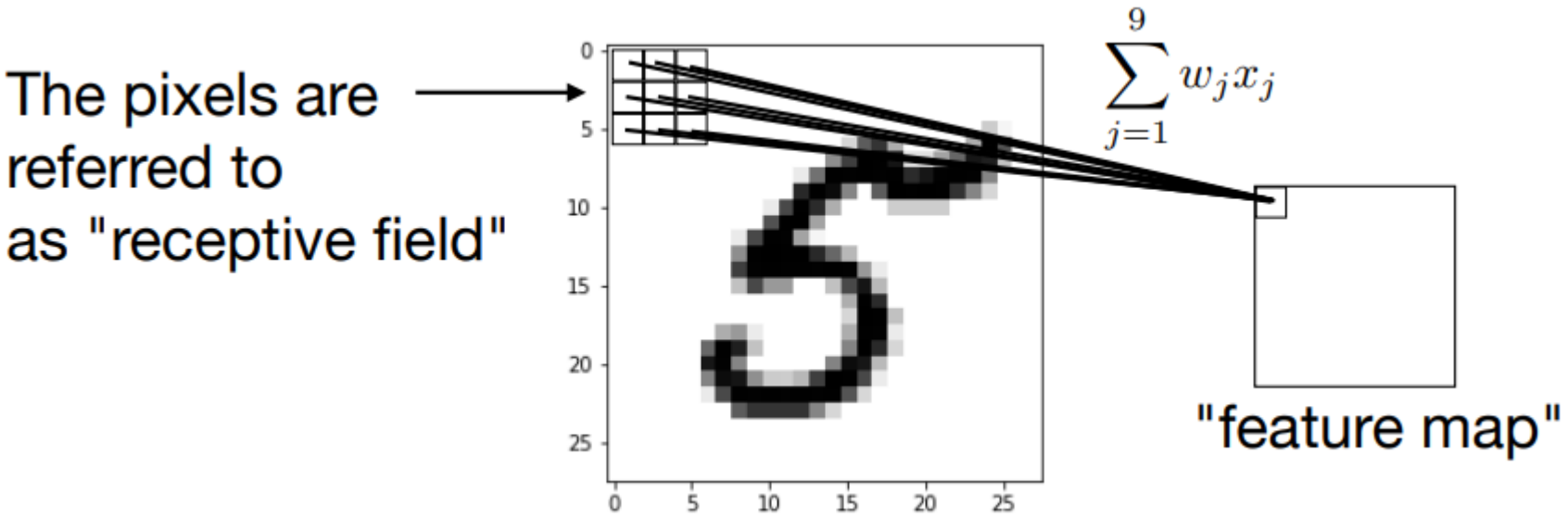
- ✓ Filters are spatially small along width and height
- ✓ Element-wise multiplication between the filter-sized patch of the input image
- ✓ And filter is done, which is then summed.



(source) https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html

Weight Sharing

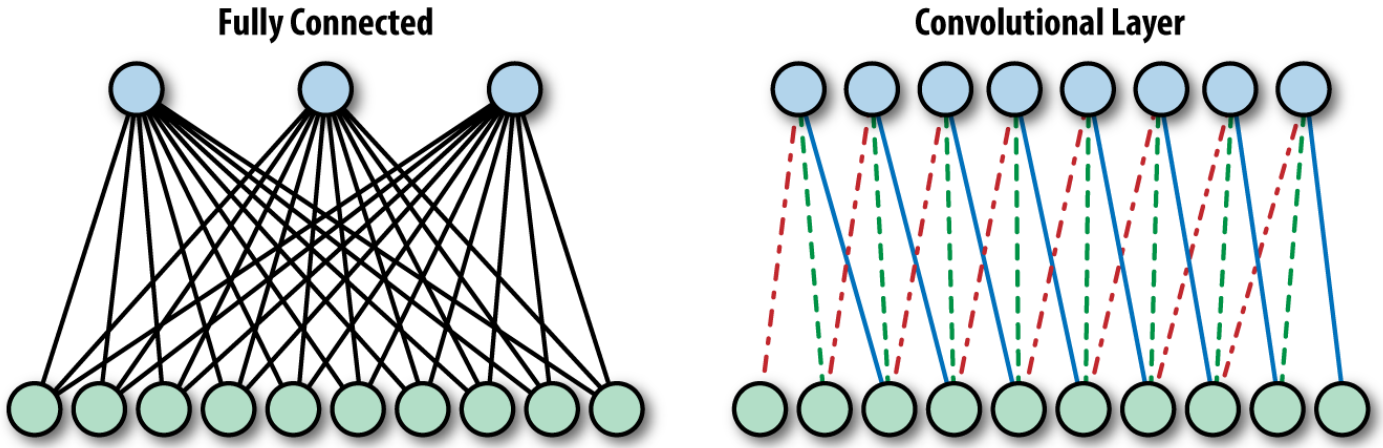
- A "feature detector" (filter, kernel) slides over the inputs to generate a feature map



source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

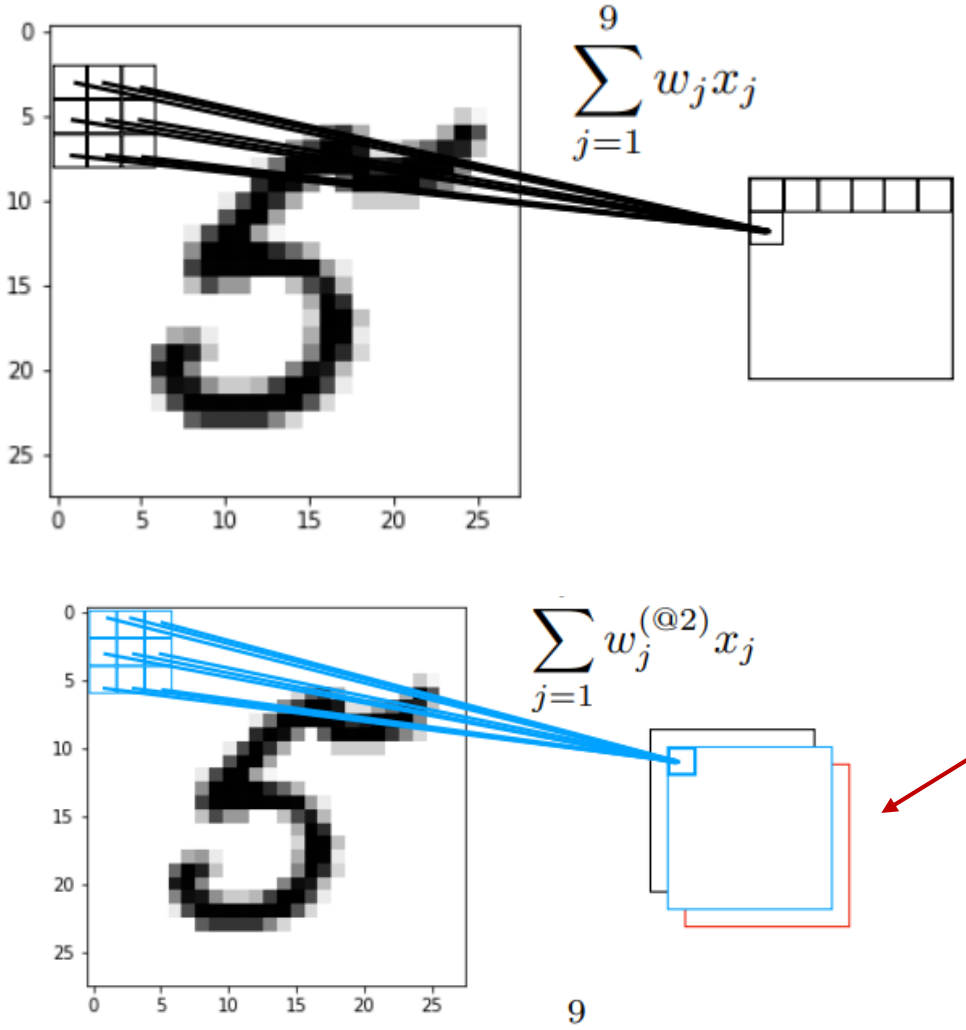
Fully-Connected layer vs Convnet

- Fully-connected layer dealing with such a huge amount of parameters requires many neurons
  - ✓ it may lead to overfitting
- CNN involves very few neurons with fewer parameters to scan an entire image



(source) https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html

CNN: Weight Sharing

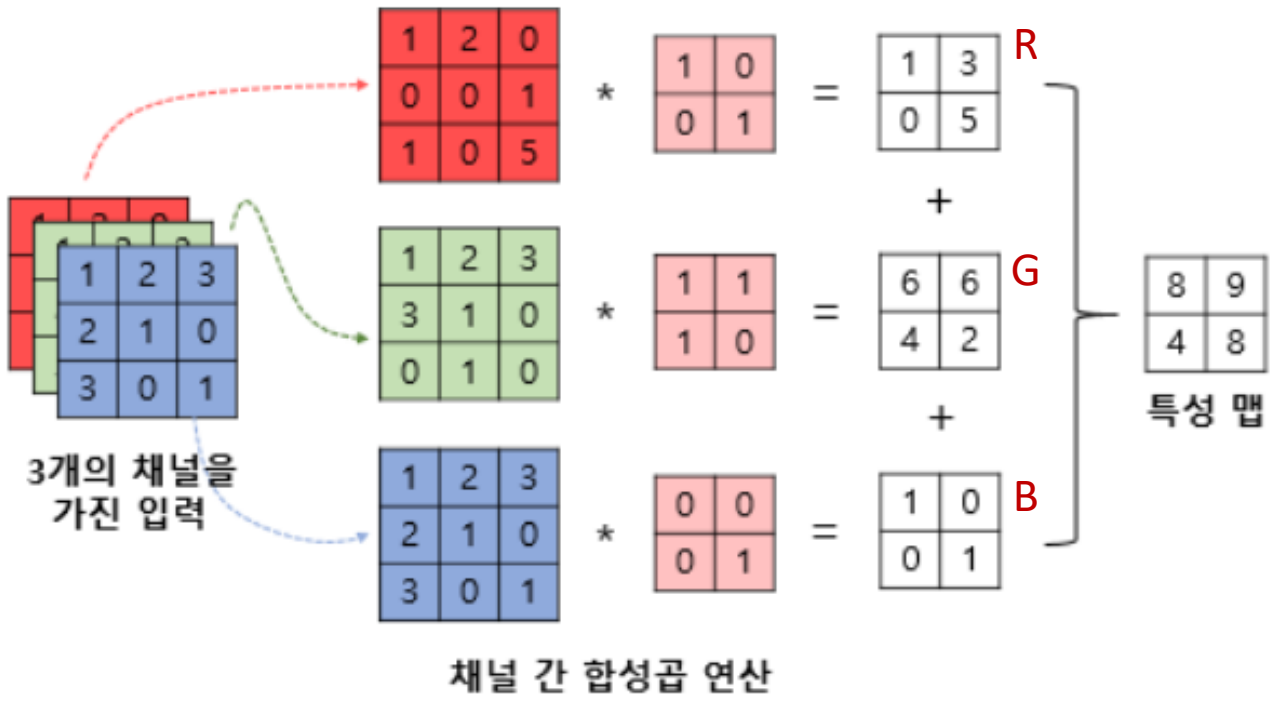


Multiple "feature detectors" (kernels) are used to create multiple feature maps

source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

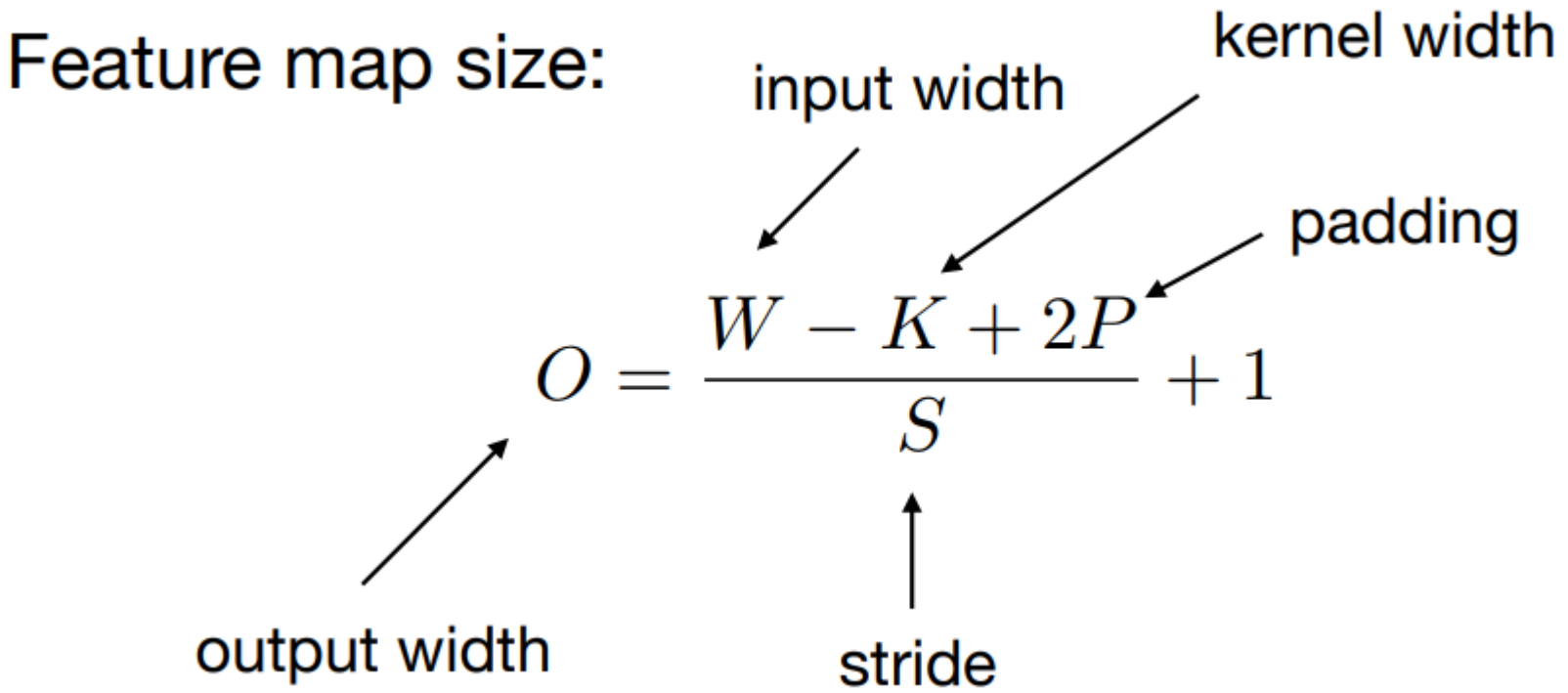
Convolutions with Color Channels

- For multiple channels, the feature map does not contain RGB meanings



(source) https://wikidocs.net/64066

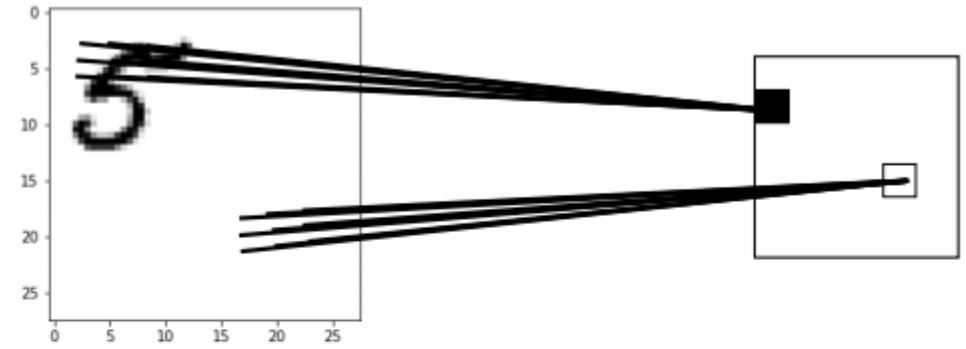
Feature map size: Size Before and After Convolutions

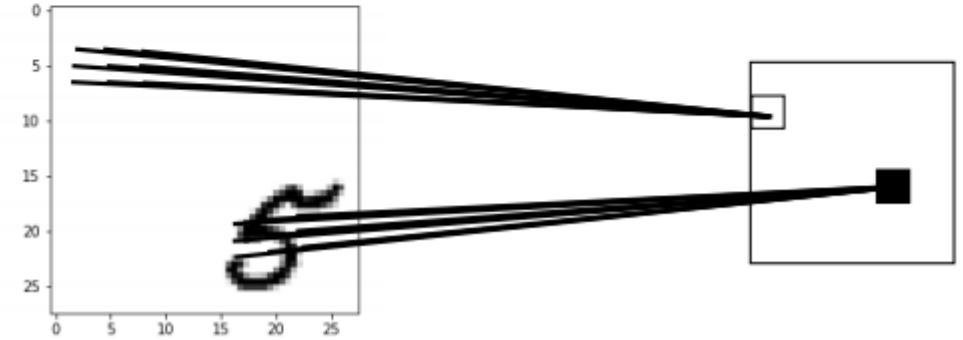


source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

CNNs and Translation/Rotation/Scale Invariance

- Note that CNNs are not really invariant to scale, rotation, translation, etc.
  - ✓The activations are still dependent on the location, etc.





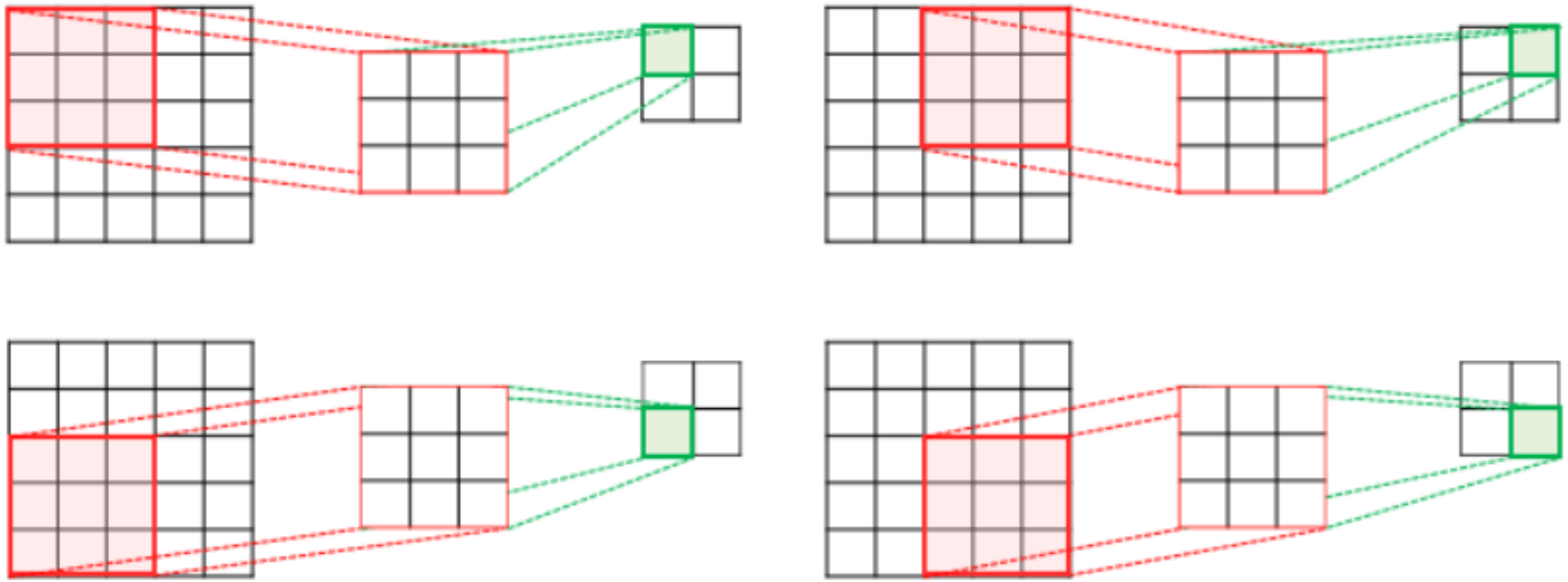
source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

Three hyperparameters in CNN

- The depth
  - ✓the number of filters we use to look for different features in the image.
- Stride
  - ✓The number of pixels we slide while matching the filter with the input image patch
- Zero-padding
  - ✓padding zeros around the border of the input image

Strided Convolution

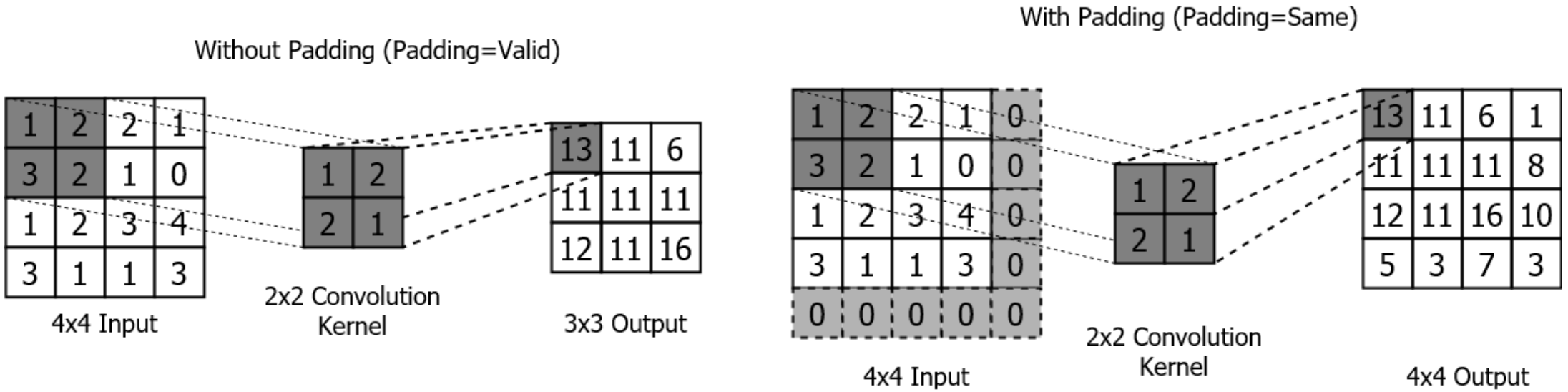
- Straid is range of movement : Stride
  - ✓5 × 5 input (spatially) assume 3x3 filter applied with stride, and finally 2x2 size feature map



(source) https://wikidocs.net/64066

Padding basically extends the area of an image

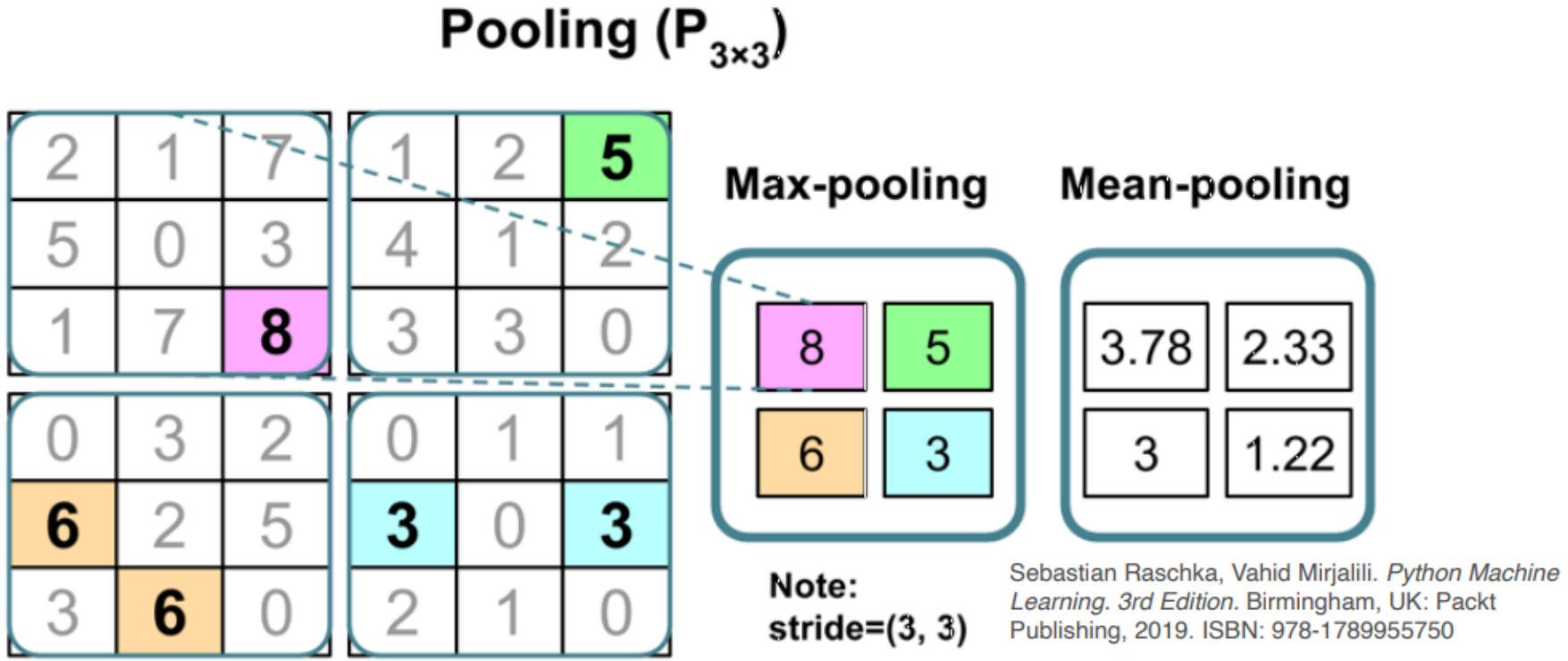
- Zero padding fills zero
  - ✓ padding="valid", padding="same"



(source) https://livebook.manning.com/book/tensorflow-in-action/chapter-4/v-1/

Pooling Layers Can Help With Local Invariance

- Downside: Information is lost.
  - ✓ May not matter for classification, but applications where relative position is important (like face recognition)



source : https://pages.stat.wisc.edu/~sraschka/teaching/stat453-ss2021/

1D CNN으로
교통흐름 예측하기 위한 전략

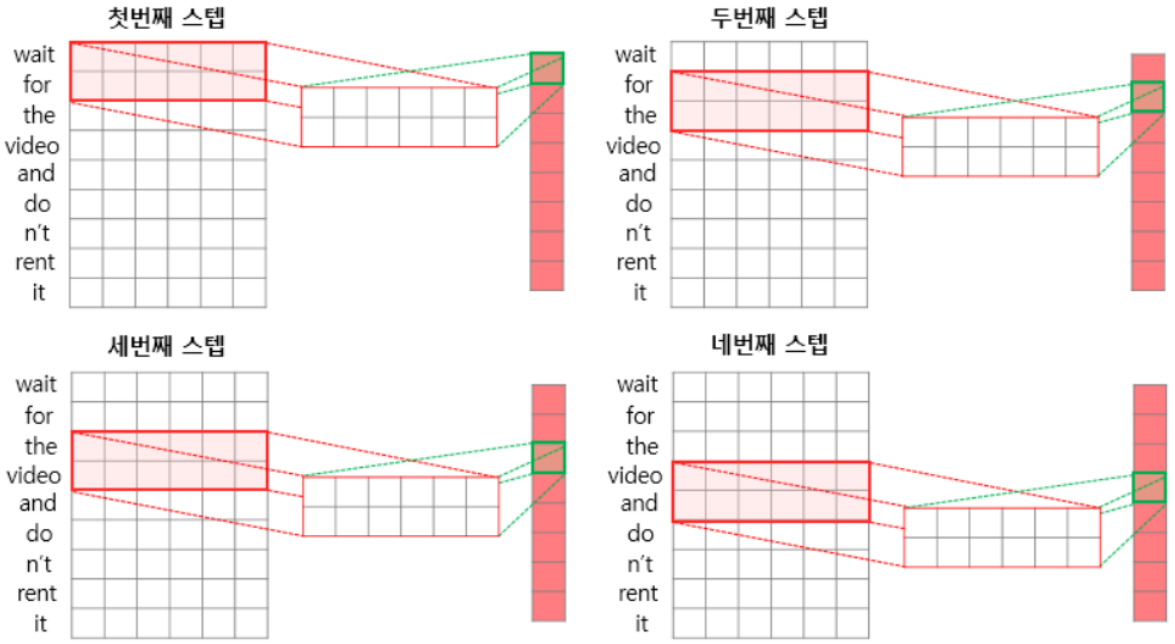
1차원 CNN

❖ 1차원 CNN 방법은 2차원과 달리

✓커널의 한 너비는 특성(feature)의 길이와 같이 둔다

✓커널의 높이가 1차원 CNN의 커널 크기가 된다.

- 아래 예제는 : the kernel size is 2



시계열 데이터 유형

1)
시계열 숫자 (10개의 look_back, 1개는 숫자)
X_train.shape = (10,000, 3, 1)
커널은 k=(3,1)

2)
시계열 이미지 (10개의 look_back, 3개는 RGB)
X_train.shape = (10,000, 10, 3)

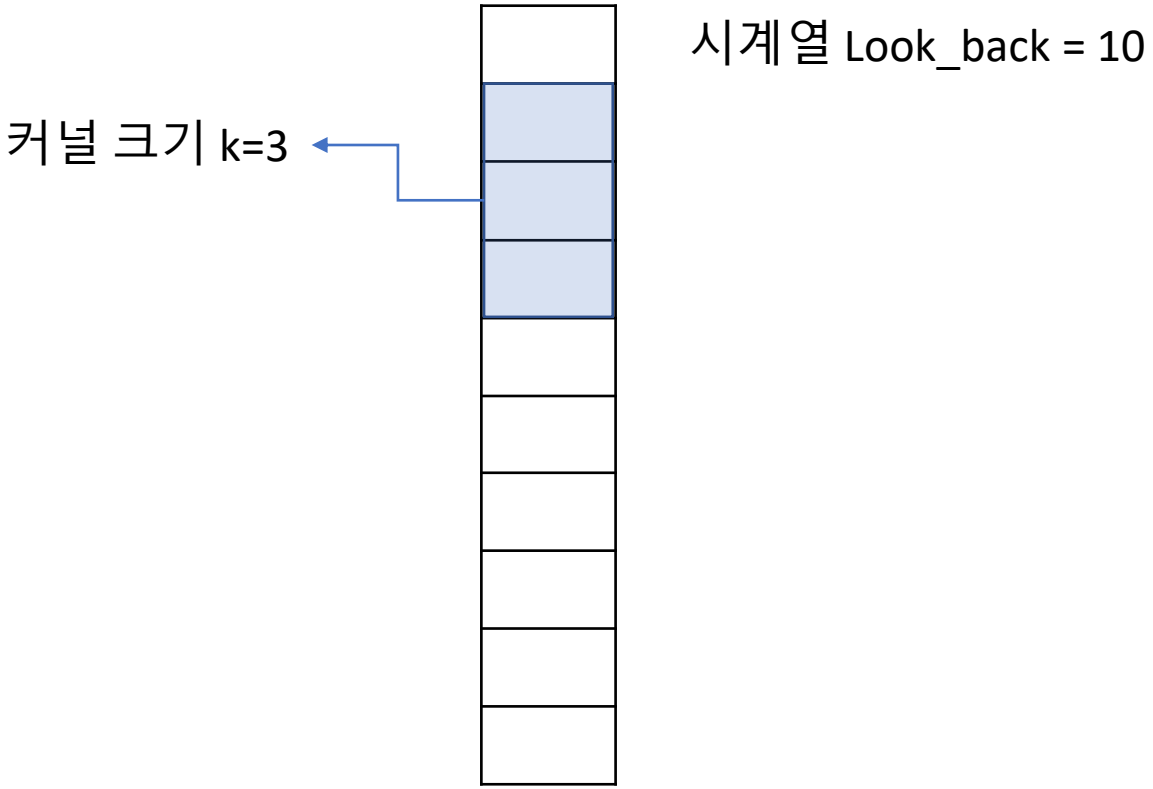
3)
시계열 문장 (10개의 look_back, 512는 임베딩차원)
X_train.shape = (10,000, 10, 512)

시계열 데이터 유형

- 차량검지기 데이터의 경우
  - ✓시계열 숫자
    - 10개의 look_back
  - ✓(예제)

X_train.shape = (배치, 시계열길이,특성)

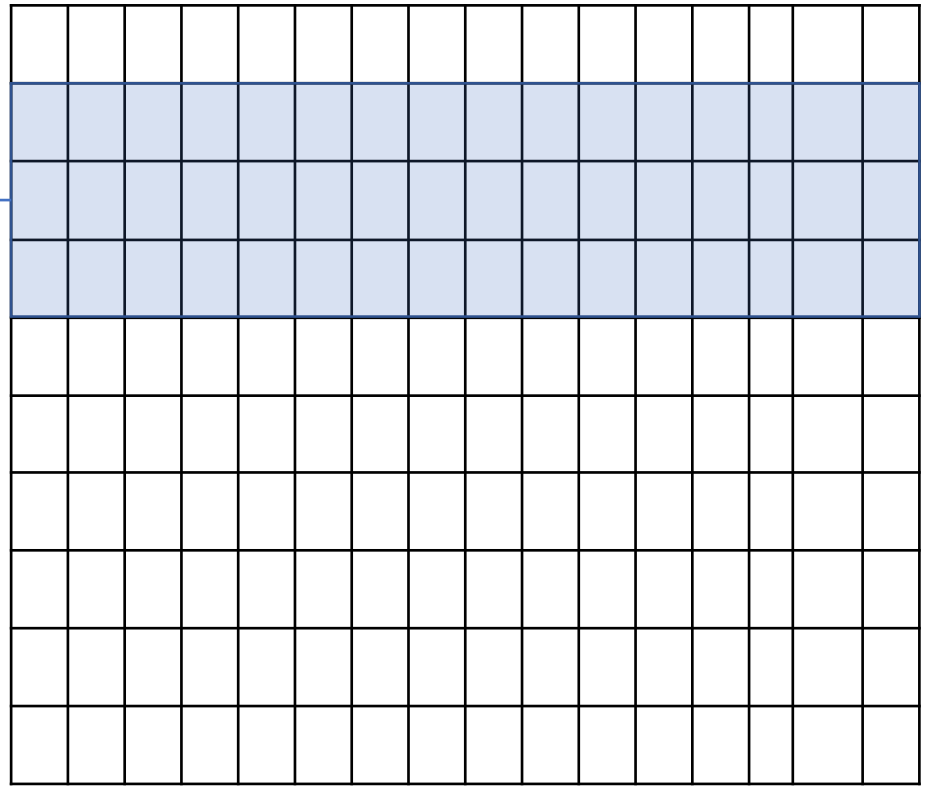
특성(feature) = 1



시계열 데이터 유형

- 텍스트 데이터의 경우
  - ✓시계열 숫자
    - 10개의 look_back
  - ✓단어 임베딩 길이
    - 보통 256

X_train.shape = (배치, 시계열길이,특성)
X_train.shape=(10,000, 10, 256)

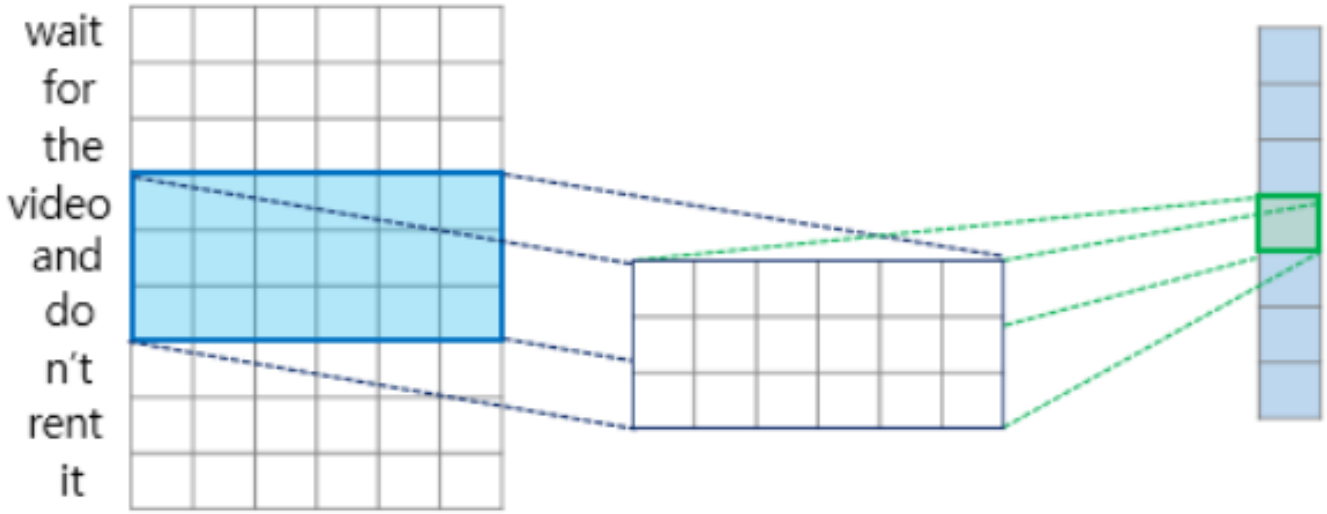


시계열 Look_back = 10

특성(feature) = 256

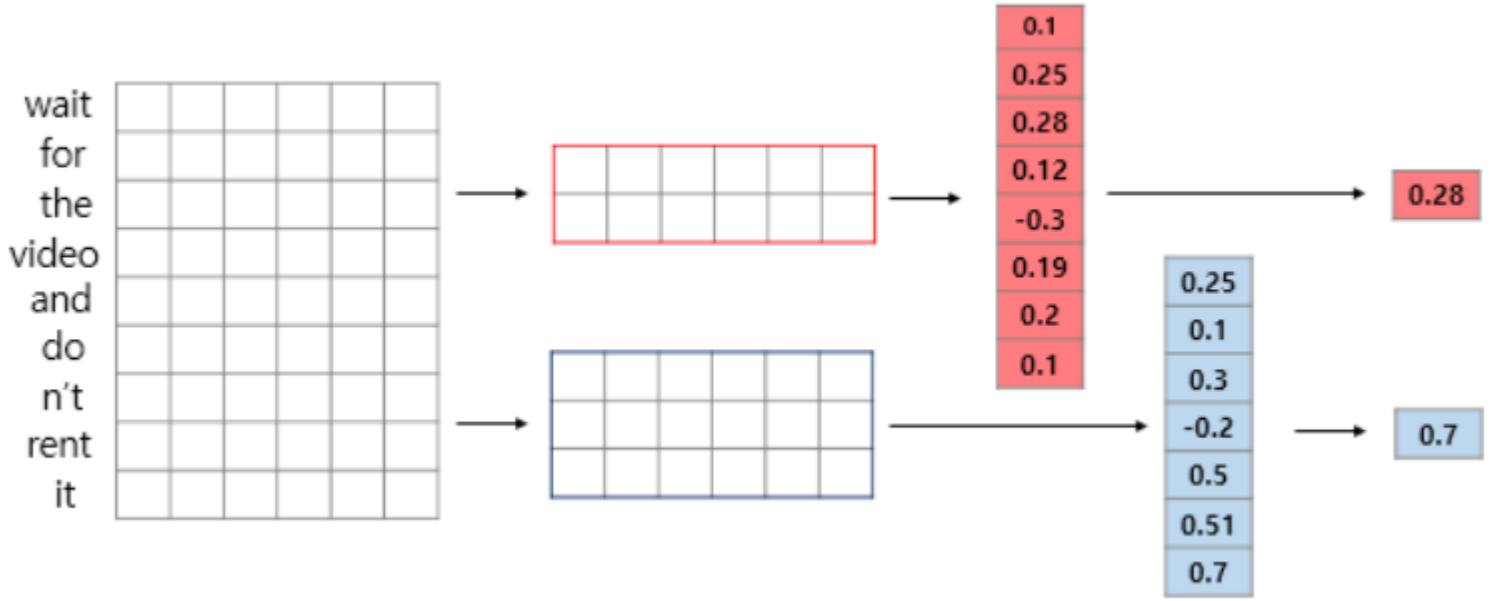
(예제) 1차원 CNN 단어

- kernel size = 3
  - ✓자연어 처리에서 참조되는 단어 묶음의 크기는 커널의 크기에 따라 다릅

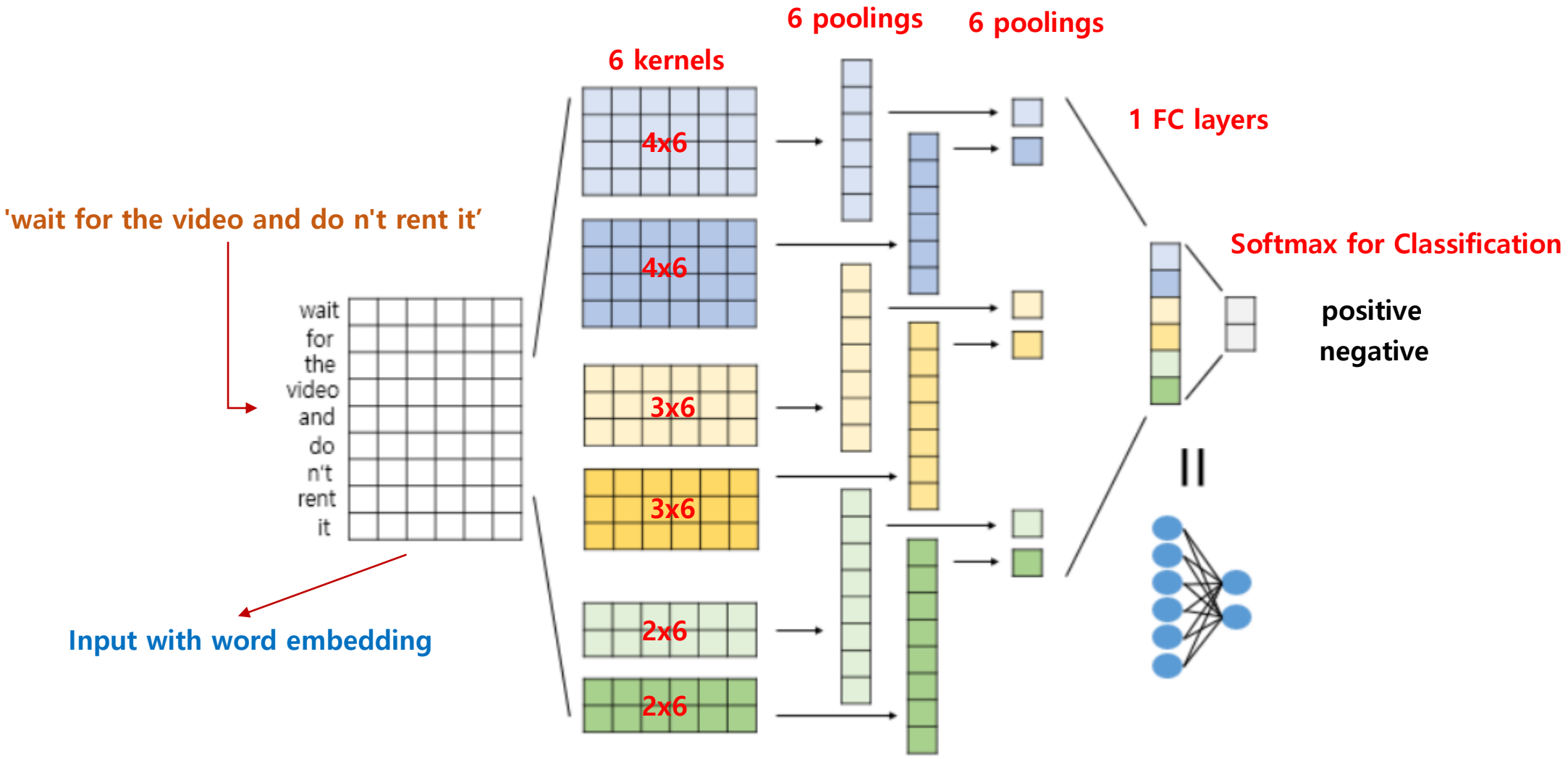


1D CNN에서 Max Pooling

- Convolutional layer
  - ✓(Convolution operation + activation function) followed by pooling layer on 1D CNN
  - ✓max-pooling operation if the kernel size is 2 and 3,



1D CNN Architecture : 문장 예시



CNN으로 교통흐름 예측 실습

- VDS 데이터, CONV1D

입력 데이터 다운로드

In [1]:
```
import tensorflow as tf
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv('https://raw.githubusercontent.com/hongsukyi/Lectures/main/vds/vds.csv')
df.head(2)
```

Out[3]:

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
0	2017-04-02 0:00	43	34	9	0	50.3	1.90
1	2017-04-02 0:05	45	32	13	0	58.9	1.84

속도를 예측해보자

```
In [4]: df_uni = df.iloc[:,5:6].values

In [5]: df_uni

Out[5]: array([[50.3],
               [58.9],
               [50.6],
               ...,
               [50.6],
               [59.3],
               [52.5]])

In [6]: print(df_uni.shape)

        (8064, 1)
```

정규화 : Speed

In [7]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range = (0,1))
scaled_df = scaler.fit_transform(df_uni)

scaled_df
```

Out[7]:
```
array([[0.52350699],
       [0.63278272],
       [0.52731893],
       ...,
       [0.52731893],
       [0.63786531],
       [0.55146125]])
```

In [8]:
```
look_back = 12*24
```

RNN 처럼 Conv1d를 위한 데이터 구조:

In [8]:
```
look_back = 12*24
```

In [9]:
```
X = []
y = []
for i in range(len(scaled_df)-look_back-1):
    X.append(scaled_df[i:(i+look_back)])
    y.append(scaled_df[(i+look_back)])

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

In [10]:
```
print(X.shape, y.shape)
```

```
(7775, 288, 1) (7775, 1)
```

데이터 나누기

In [11]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, shuffle=False)

print(X_train.shape, y_train.shape)
print(X_test.shape,  y_test.shape)
```

```
(6220, 288, 1) (6220, 1)
(1555, 288, 1) (1555, 1)
```

In [12]:
```
neuron_1 = 32
neuron_2 = 16
```

간단한 Conv1D 모델

In [14]:

```
def model_conv1d():
    model = Sequential([
        Conv1D(neuron_1 , kernel_size=(6,), padding='same',
               activation='relu', input_shape = (X_train.shape[1],1)),
        Flatten(),
        Dense(neuron_2 , activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss= 'mse', metrics=['mae'])
    return model
```

모델과 파라미터 개수

In [16]:

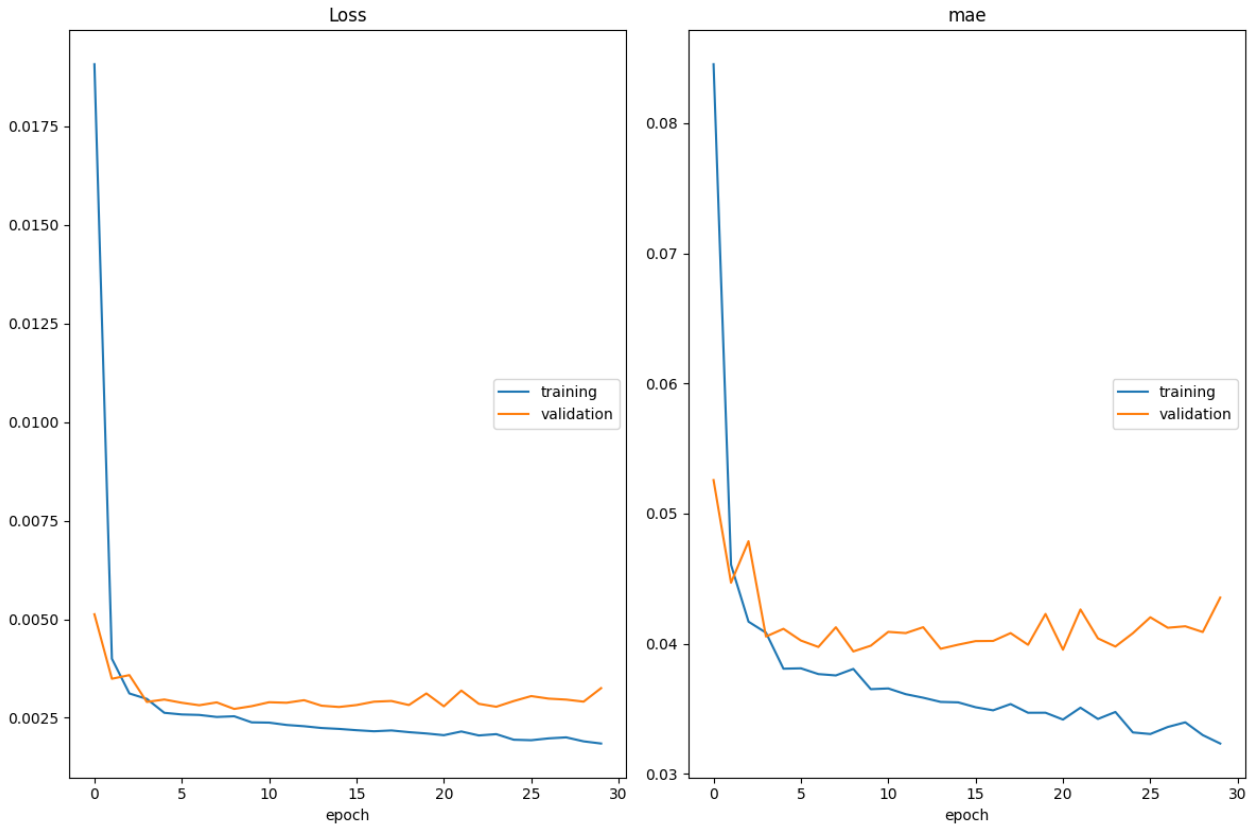
```
model = model_conv1d()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 288, 32)           224

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 16)                147472

 dense_1 (Dense)             (None, 1)                 17

=================================================================
Total params: 147,713
Trainable params: 147,713
Non-trainable params: 0
_________________________________________________________________
```

실시간 훈련 과정 가시화

In [17]:
```
from livelossplot import PlotLossesKeras
```

In [18]:
```
history = model.fit(X_train,y_train, epochs=30, validation_split=0.1, batch_size=32,callbacks=[PlotLossesKeras()])
```



테스트 데이터로 MSE 결과

```
Loss
        training                (min:    0.002, max:    0.019, cur:    0.002)
        validation              (min:    0.003, max:    0.005, cur:    0.003)
mae
        training                (min:    0.032, max:    0.085, cur:    0.032)
        validation              (min:    0.039, max:    0.053, cur:    0.044)
175/175 [==============================] - 6s 33ms/step - loss: 0.0018 - mae: 0.0323 - val_loss: 0.0033 - val_mae: 0.0435
```

In [19]:

```
mse, mae = model.evaluate(X_test,y_test, verbose=2)
print("테스트 세트의 평균 제곱 오차: {:5.4f}".format(mse))
```

```
49/49 - 1s - loss: 0.0031 - mae: 0.0409 - 1s/epoch - 25ms/step
테스트 세트의 평균 제곱 오차: 0.0031
```

정규화된 속도 예측

In [21]:
```
# Prediction
y_pred = model.predict(X_test)
```

```
49/49 [==============================] - 2s 28ms/step
```

In [24]:
```
pred_df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
pred_df.head()
```

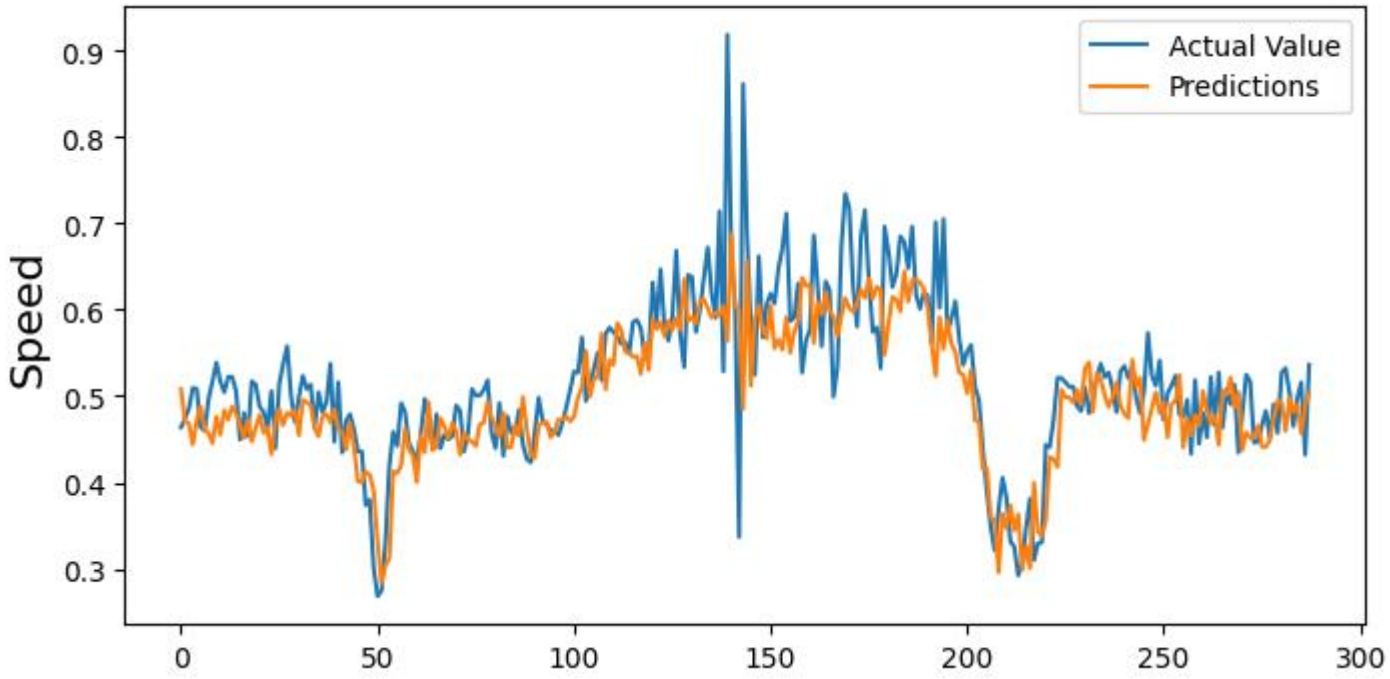
Out[24]:

	Actual	Predicted
0	0.463787	0.508119
1	0.473952	0.470623
2	0.485388	0.468415
3	0.509530	0.443892
4	0.508259	0.472212

실제와 예측을 비교 (정규화 값 사용)

In [25]:

```
plt.figure(figsize=(8,4))
plt.ylabel('Speed', fontsize=16)
plt.plot(pred_df[:288])
plt.legend(['Actual Value', 'Predictions'])
plt.show()
```

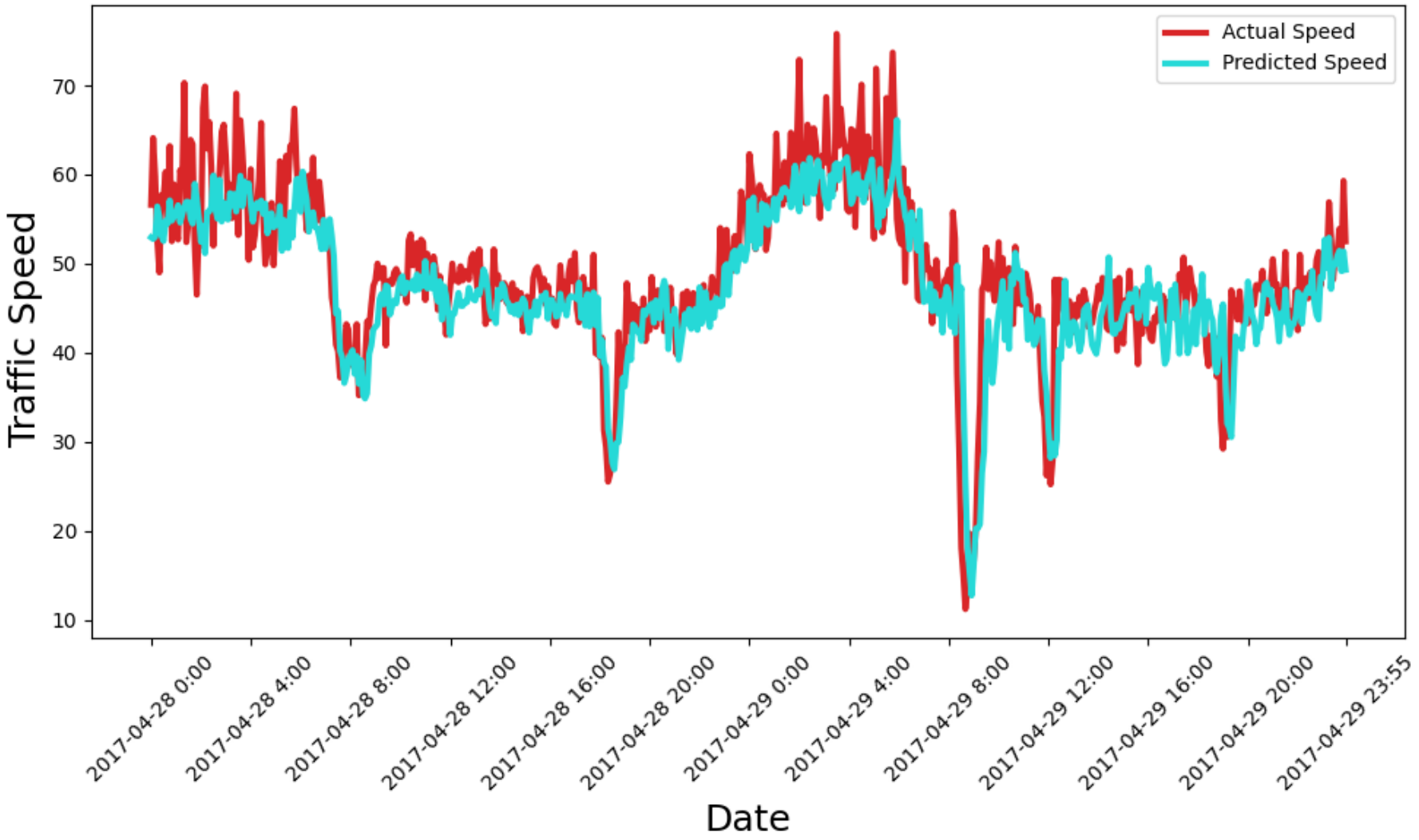


실제값과 비교를 위한 데이터 환원

In [26]:
```
y_pred = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred)
```

In [29]:
```
plt.figure(figsize=(10,6))
plt.plot(df["Date"].iloc[-576:], df_uni[-576:], linewidth= 3,  color='#d92628', label='Actual Speed')
plt.plot(y_pred[-576:], color='#26D9D7', linewidth= 3, label='Predicted Speed')
plt.xlabel("Date", size="18")
plt.ylabel("Traffic Speed ", size="18")
plt.xticks([0, 48, 96, 144,192, 240, 288, 336, 384, 432, 480, 528,575],rotation=45)
plt.tight_layout()
plt.legend()
```

대전시 유성구 대학로 앞 도로에서 차량 속도(km/h) 예측



원래 값으로 환산 : 속도

```
# Prediction
y_pred = model.predict(X_test, verbose=0)
print(y_pred.shape,y_test.shape)
```

```
(1555, 1) (1555, 1)
```

```
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform(y_test)
```

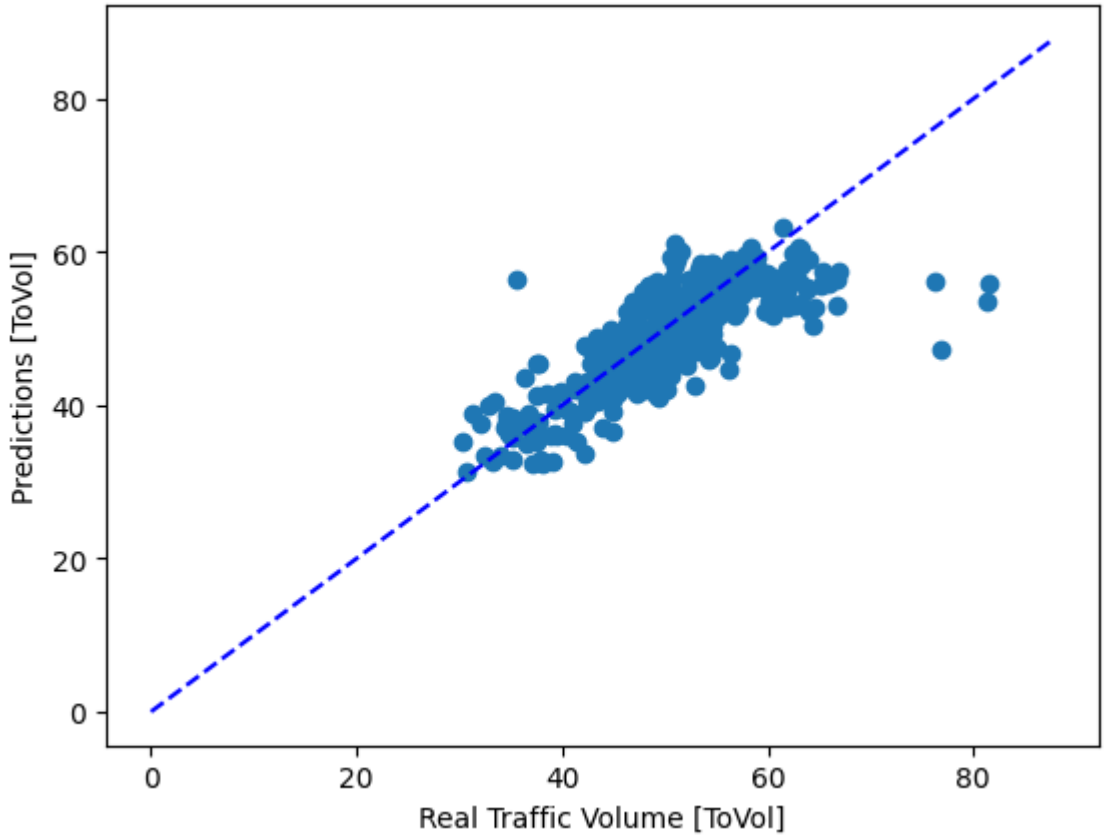
```
pred_df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
pred_df.head()
```

0] :

	Actual	Predicted
0	45.6	49.088932
1	46.4	46.138062

실제와 예측

```
plt.scatter(y_test.flatten()[:576], y_pred.flatten()[:576])
plt.xlabel('Real Traffic Volume [ToVol]')
plt.ylabel('Predictions [ToVol]')
plt.plot([0, max_speed], [0, max_speed],'b--')
```



숙제

❖ CNN 모델 최적화

✓ 속도(Speed)와 교통량에 대하여 아래 표를 작성하세요

- Conv1d의 kernel_size 크기를 최적화 해보세요.

✓Look_back, 뉴런, 커널 사이즈 모두 변경 가능 (옵션)

- 가장 작은 MSE 값을 제출함

항목	Look_back(5시간)	뉴런1=뉴런2	Kernel_size	MSE
Speed	60	32	3	?
Speed	60	32	?	?
Speed	60	32	?	?
Speed	60	32	?	?
Speed	?	?	?	?

```
look_back=144,  MSE=0.002782
```

KISTI SINCE 1962

2023 TRUST KISTI

Korea Institute of Science and Technology Information