[Tensorflow Keras] 레이어를 구현해보자.

Robert Lee · 2021년 10월 5일

Deep Learning Keras tensorflow

목차

- 기본 레이어 개념을 이해하고 구현해본다.
  - 컨볼루션 `Conv Layer`
  - 플래튼 `Flatten Layer`
  - 폴링 `Pooling Layer`

환경

- 파이썬 `3.7`
- 텐서플로우 `2.6`
- 케라스 `2.6`

😳 기본 레이어 이해

우리는 앞서 딥러닝과 뉴럴 네트워크의 차이를 알아보았다.

그리고 뉴럴 네트워크 구현 최소 단위인 레이어 `Layer` 가 필요하다고 했다.

여기서는 `Conv Layer` `Pooling Layer` `Flatten Layer` 를 다뤄보도록 하겠다.

> “ 해당 개념은 링크의 동영상을 보는 것을 추천한다. ”

🏅 Conv Layer <링크>

우리가 이미지 분류를 잘하려면 특징 Feautre 을 잘 파악해야 할 것이다.

> “ 뉴럴 네트워크에서 이미지 특징을 어떻게 뽑아낼 수 있을까 ? ”

그에 대한 대답으로

> “ Conv Layer 의 필터를 사용하면 된다. ”

컨볼루션망 `CNN, Convolutional Neural Network` 을 들어본 경험이 있다면

이미지 분류를 잘하는 모델로 기억할 것이다.

> “ 그렇다. 이미지를 잘 분류하려면 Conv Layer 가 필요하다. ”

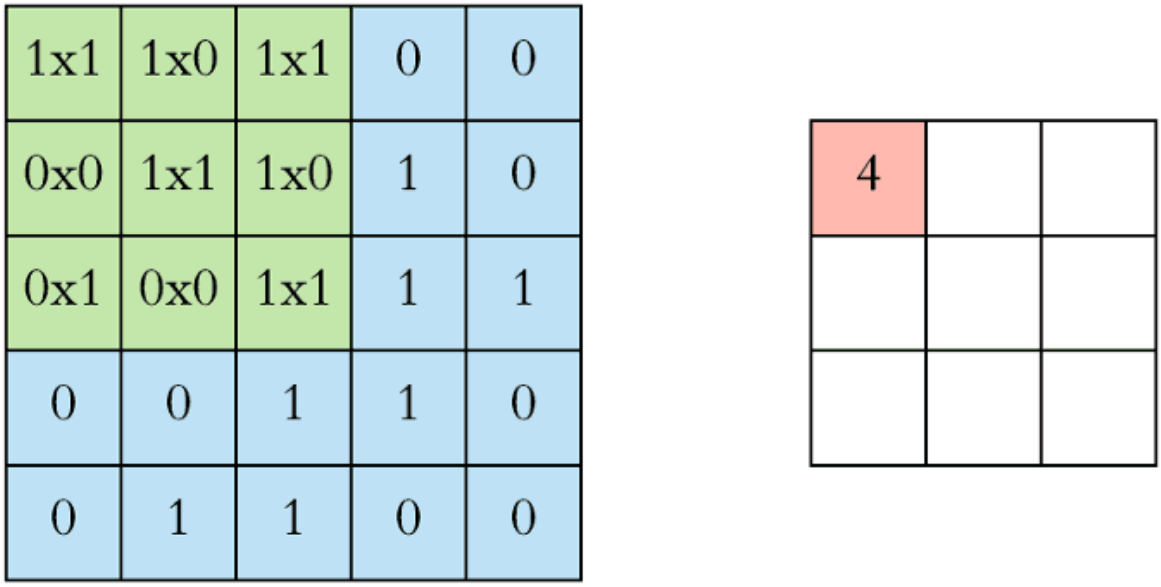
컨볼루션 `Conv Layer` 에는 필터 `Filter` 와 커널 `Kernel` 개념이 존재한다.

> “ 필터란 몇개의 특징으로 출력을 만들어 낼 것인가 ? 를 의미한다. ”

즉 해당 이미지를 판단하기 위해 가장 좋은 특징맵 Feature Map N 개를 찾아낸다.

또한 특징맵을 한개를 만들어가는 과정에서 커널 Kernel 개념이 사용된다.

이미지 크기가 5 by 5 이고 커널 크기가 3 또는 (3, 3) 인 그림을 살펴보자.



출처 - jjeongil.tistory.com

고정된 커널 크기에 따라 곱셈 연산을 하고 이를 모두 더한값을 기록한다.

이제 tf.keras.layers.Conv2D() 이용하여 3 = (3, 3) 커널로 이루어진 32 개의 필터를 만들어보자.

```
import tensorflow as tf

input_layer = tf.keras.layers.Input(
    shape=(200, 200, 3), name='input_layer'
)

conv_layer = tf.keras.layers.Conv2D(
    filters=32, kernel_size=(3, 3), activation='relu', name='conv_layer'
)(input_layer)

output_layer = tf.keras.layers.Dense(
```

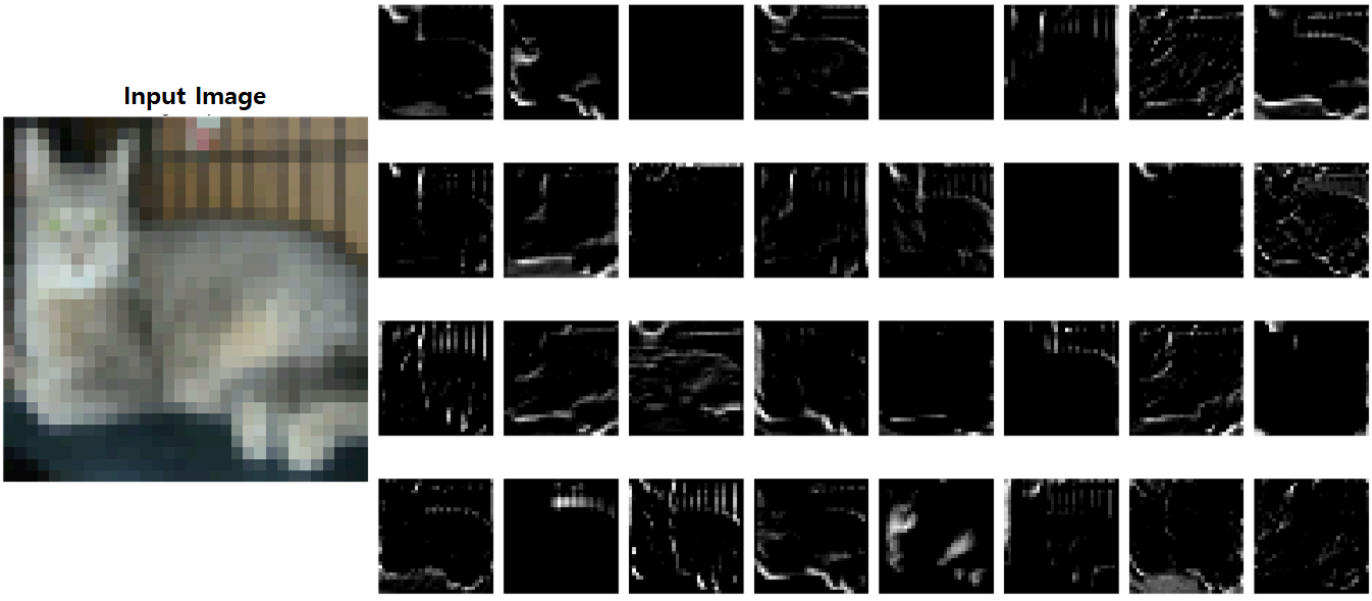
```
    units=2, activation='softmax', name='output_layer'
)(conv_layer)

model = tf.keras.models.Model(input_layer, output_layer)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 200, 200, 3)]     0
_________________________________________________________________
conv_layer (Conv2D)          (None, 198, 198, 32)      896
_________________________________________________________________
output_layer (Dense)         (None, 198, 198, 2)       66
=================================================================
Total params: 962
Trainable params: 962
Non-trainable params: 0
_________________________________________________________________
```

그리고 다음과 같은 특징맵 `Feature Map` 을 얻을 수 있을 것이다.

여기서 특징맵은 `200 - (3 - 1) by 200 - (3 - 1)` 의 크기로 `32` 개가 출력된다.

한장의 사진은 `(1, 200, 200, 3)` 크기로 컨볼루션에 입력되어 `(32, 198, 198, 3)` 크기로 출력된다.



출처 - ricardodeazambuja.com/deep_learning

2 Flatten Layer <링크>

먼저 플래튼 `Flatten` 의 사전적 의미는 다음과 같다.

> **flatten** ['flætn]
> 1. (동사) 납작 [반반] 해지다, 납작하게 [반반하게] 만들다
> 옥스퍼드 영한사전

플래튼을 사용하는 이유는 단순하다.

우리는 <강아지, 고양이> 이미지 분류 `Classification` 를 할 예정이라면

`1` 차원 배열로 출력해야 한다. <두개 값으로 나와야 한다.>

> “ 이미지 1 의 예측값 예시 = [강아지 확률, 고양이 확률] = [90 % , 10 %] ”

그러나 우리는 `3` 차원 이상 배열 `Tensor` 을 이용해 특징을 찾아가고 있다.

그러므로 플래튼이 없다면 `3` 차원으로 출력이 될 것이다.

다음 그림의 출력 `Output Shape` 을 살펴보자.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 200, 200, 3)]             0
_________________________________________________________________
conv_layer (Conv2D)          (None, 198, 198, 32)            896
_________________________________________________________________
output_layer (Dense)         (None, 198, 198, 2)              66
=================================================================
```

자 이제 텐서플로우 `tf.keras.layers.Flatten()` 을 통해 일자로 평평하게 만들어보자.

단순 `1` 차원 배열 `Array` 로 만드는 것이므로 파라미터는 필요 없다.

```
import tensorflow as tf

input_layer = tf.keras.layers.Input(
    shape=(200, 200, 3), name='input_layer'
)
```

```
conv_layer = tf.keras.layers.Conv2D(
    filters=32, kernel_size=(3, 3), activation='relu', name='conv_layer'
)(input_layer)

faltten_layer = tf.keras.layers.Flatten(name='flatten_layer')(conv_layer)

output_layer = tf.keras.layers.Dense(
    units=2, activation='softmax', name='output_layer'
)(faltten_layer)

model = tf.keras.models.Model(input_layer, output_layer)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 200, 200, 3)]             0
_________________________________________________________________
flatten_layer (Flatten)      (None, 1254528)                   0
_________________________________________________________________
conv_layer (Conv2D)          (None, 198, 198, 32)            896
_________________________________________________________________
output_layer (Dense)         (None, 2)                 2509058
=================================================================
Total params: 2,509,954
Trainable params: 2,509,954
Non-trainable params: 0
_________________________________________________________________
```

즉 플래튼을 적용하면 정상적으로 <강아지, 고양이> `= [0.9, 0.1]` 또는

<강아지, 고양이, 호랑이> `= [0.9, 0.0, 0.1]` 출력이 가능하다.

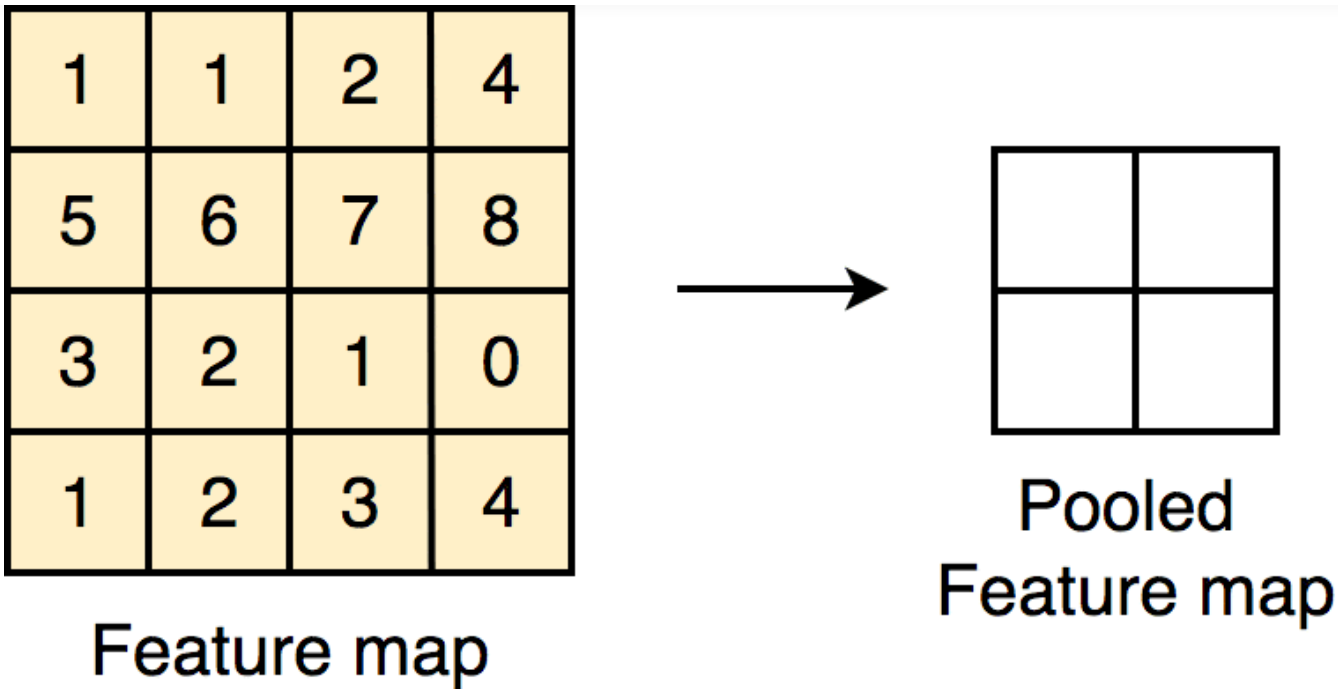
🥉 Pooling Layer <링크>

마지막으로 살펴볼 레이어는 풀링 `Pooling Layer` 이다.

풀링은 `Sub Sampling` 으로 불리며 이미지 데이터를 작은 크기로 줄여주는 역할을 한다.

풀링에는 대표적으로 `Max Pooling Layer` 와 `Average Pooling Layer` 있다.

다음 그림은 `Max Pooling Layer` 을 나타내며 풀링 크기 `Pooling Size` 가 `(2, 2)` 이다.



출처 - medium.com/parva.shah808

`Max Pooling` 풀링 크기에 마춰 해당 픽셀 중 가장 큰 값을 기록한다.

> “ 다시 말해 Max Pooling 을 사용하면 가장 두드러진 특징만을 기록할 수 있다. ”

> “ 그렇다면 Average Pooling 의 특징은 무엇일까 ? 고민해보고 이포스트 하단을 읽어보자. ”

그리고 문득 아래와 같은 질문을 할 수 있다.

> “ 이미지 데이터를 작은 크기로 만들 필요가 있을까 ? 왜 그래야만 하는거지 ? ”

우리는 컨볼루션 `Conv Layer` 를 바탕으로 이미지 특징을 찾아내고

플래튼 `Flatten Layer` 으로 일자로 만든 뒤에 확률 형태로 출력을 하였다.

풀링이 없는 뉴럴 네트워크를 구성해도 되지만

최적화 해야되는 파라미터 `Parameter` 개수가 많아질 것이다.

> “ 파라미터가 많다는 의미 오버피팅, 학습시간 등 문제를 야기한다는 뜻이다. ”

만약 풀링 `Pooling` 이 없다면

아래 그림과 같이 `2,509,954` 개의 파라미터를 찾아야 된다.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 200, 200, 3)]             0
_________________________________________________________________
conv_layer (Conv2D)          (None, 198, 198, 32)            896
_________________________________________________________________
flatten_layer (Flatten)      (None, 1254528)                   0
_________________________________________________________________
output_layer (Dense)         (None, 2)                 2509058
=================================================================
Total params: 2,509,954
Trainable params: 2,509,954
Non-trainable params: 0
_________________________________________________________________
```

그래서 우리는 파라미터를 줄이기 위해 `Pooling` 을 한다.

그리고 `Pooling Layer` 는 `conv_layer` 다음에 위치하는 것이 일반적이다.

특징을 찾은 이후에 풀링으로 이미지 크기를 줄인다.

이제 `tf.keras.layers.MaxPool2D()` 를 바탕으로 `Pooling Layer` 를 구현해보자.

```
import tensorflow as tf

input_layer = tf.keras.layers.Input(
    shape=(200, 200, 3), name='input_layer'
)

conv_layer = tf.keras.layers.Conv2D(
    filters=32, kernel_size=(3, 3), activation='relu', name='conv_layer'
)(input_layer)

pool_layer = tf.keras.layers.MaxPool2D(
    pool_size=(3, 3), name='pool_layer'
)(conv_layer)

faltten_layer = tf.keras.layers.Flatten(name='flatten_layer')(pool_layer)

output_layer = tf.keras.layers.Dense(
    units=2, activation='softmax', name='output_layer'
)(faltten_layer)
```

```
model = tf.keras.models.Model(input_layer, output_layer)

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 200, 200, 3)]             0
_________________________________________________________________
conv_layer (Conv2D)          (None, 198, 198, 32)            896
_________________________________________________________________
pool_layer (MaxPooling2D)    (None, 66, 66, 32)                0
_________________________________________________________________
flatten_layer (Flatten)      (None, 139392)                    0
_________________________________________________________________
output_layer (Dense)         (None, 2)                    278786
=================================================================
Total params: 279,682
Trainable params: 279,682
Non-trainable params: 0
_________________________________________________________________
```

위 결과와 같이 풀링 `Pooling Layer` 를 적용하면

`279,682` 개 파라미터를 최적화 시키면 된다.

만약 풀링이 없다면 `2,509,954` 파라미터를 찾아야 될 것이다.

추가로 FC 레이어 `Fully Connected Layer` 에 대해 알아보자.

> “ FC 레이어는 일렬로 펴진 층과 모든 노드가 연결된 구간을 이야기한다. ”

지금 우리가 만든 모델에도 FC 레이어가 존재한다.

일자로 평평하게 만든 플래튼 `Flatten Layer` 과 마지막 출력을 위한 `Dense Layer` 합친 구간을 말한다.

또한 뉴럴 네트워크 깊이에 따라 플래튼과 출력 사이에 한개 이상 `Dense Layer` 를 위치시킬 수 있다.

통상적으로 이 부분을 은닉층 `Hidden Layer` 이라 부르는 것 같다.

이제 우리는 레이어에 대해 간략하게 이해를 하였다.

그러나 막상 레이어를 사용 시에 고려해야할 점들이 존재한다.

예를 들어 다음과 같은 질문이 될 수 있다.

> “ Max Pooling 과 Average Pooling 중에 어떤것을 사용하여야 하는가 ? ”

다음 포스트에서는 몇가지 레이어 `Layer` 를 비교해 보고자 한다.

🍀 참고

- 오픈 튜토리얼
- 텐서플로우 튜토리얼
- 텐서플로우 API 문서

팔로우

Robert Lee

Hi. Hello World.

다음 포스트
[Tensorflow Keras] MaxPooling 과 Average Pooling 을 살펴보자.

이전 포스트
[Deep Learning] 딥러닝과 뉴럴 네트워크의 관계를 알아보자.

0개의 댓글

댓글을 작성하세요

댓글 작성

관심 있을 만한 포스트

AlexNet의 이해

참고AlexNet은 2012년에 개최된 ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 대회의 우승을 차지한 컨볼루션 신경망(CNN) 구조이다. CNN의 부흥에 아주 큰 역할을 한 구조라고 말할 수 있다. Alex...

2022년 2월 22일 · 0개의 댓글

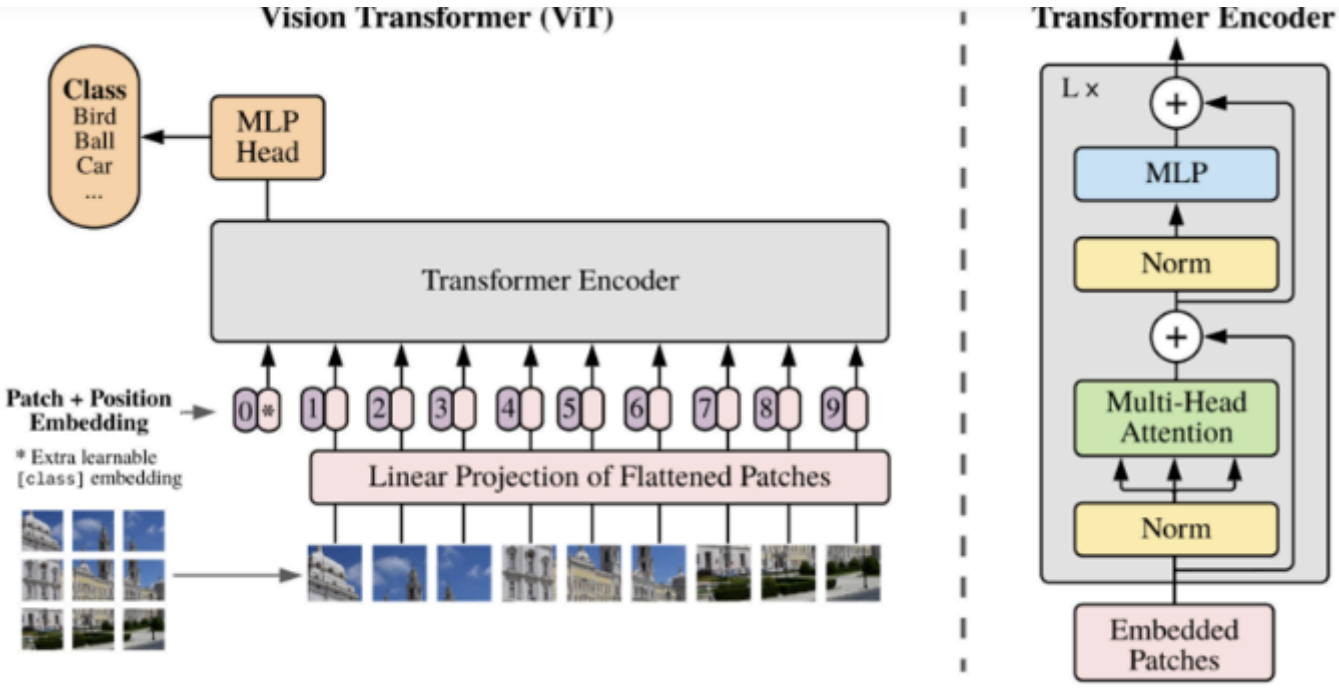
by **박재한** 4

LeNet-5의 이해

참고 1. LeNet-5란 무엇인가? LeNet는 이미지 분류용 CNN 중에 조상격으로 CNN초기에 CNN의 기본 구조를 잘 정립하였다. LeNet의 다양한 버전들(LeNet-1, LeNet-2,...)이 있으나 최종 버전은 LeNet-5이다. LeNet은 CNN을 처...

2022년 2월 21일 · 0개의 댓글

by **박재한** 1



Vision Transformer(ViT) 논문 리뷰

ViT(비전 트랜스포머) 논문 읽기

2023년 4월 9일 · 2개의 댓글

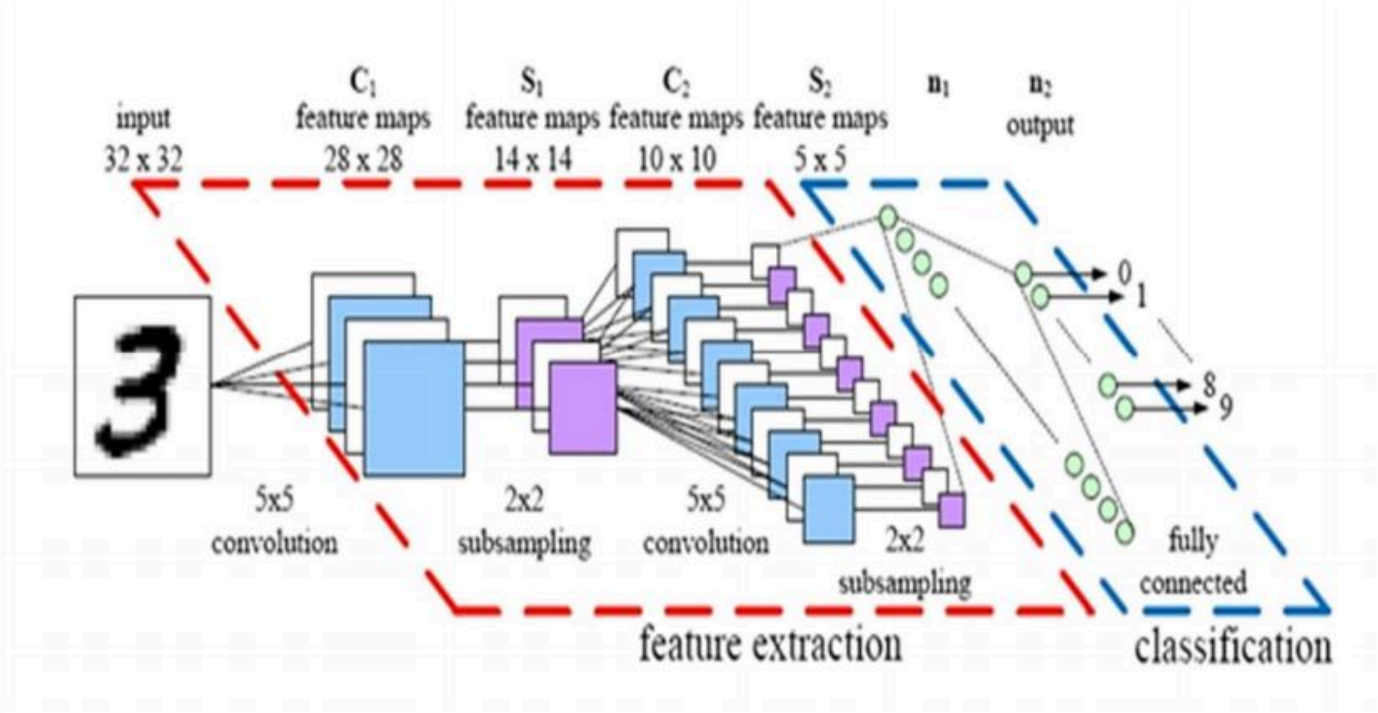
by **정예슬** 2

UNet의 이해

참고1참고2이미지 세그멘테이션(image segmentation)은 이미지의 모든 픽셀이 어떤 카테고리(예를 들면 자동차, 사람, 도로 등)에 속하는지 분류하는 것을 말한다.이미지 전체에 대해 단일 카테고리를 예측하는 이미지 분류(image classification)...

2022년 6월 27일 · 0개의 댓글

by **박재한** 1



[AI] CNN(Convolutional Neural Network) 개념정리

CNN 학습

2020년 8월 10일 · 0개의 댓글

by **개발Velog**

4

이미지 세그멘테이션(Image Segmentation)

참고1참고2참고3Neural Network를 이용한 이미지 처리에는 다음과 같이 분류할 수 있다. 다음 그림을 예제로 들어 이미지 처리의 종류를 알기 쉽게 설명하였다.Imgur1) Image Classification : 사진의 이미지 객체가 어떤 것인지 구분하는 것(...

2022년 6월 27일 · 0개의 댓글

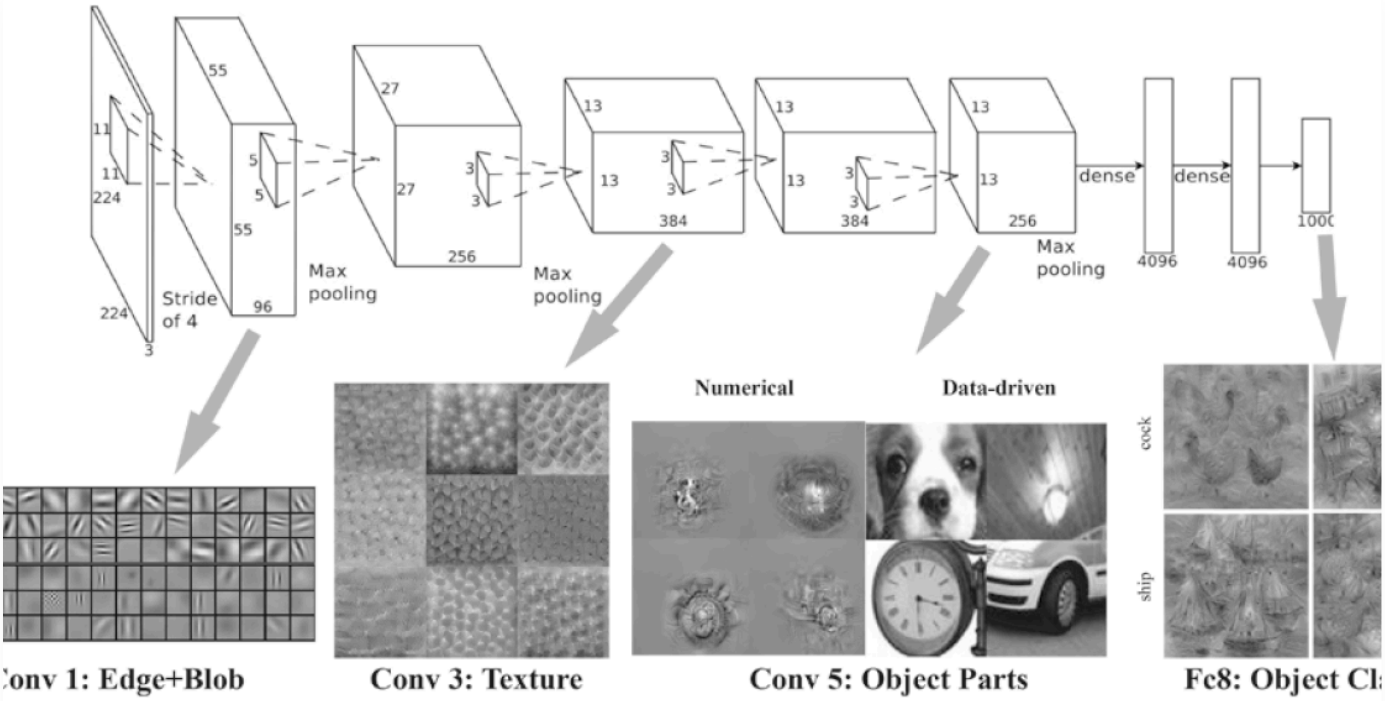
by **박재한** 1

[Tensorflow Keras] MaxPooling 과 Average Pooling 을 살펴보자.

레이어 개념 비교 설명: Max Pooling Average Pooling Global Average Pooling우리는 앞서 이미지 분류를 위한 기본적인 레이어에 대해 살펴보았다.여기서 우리는 더 깊은 이해를 위해 자료를 찾을 것이고폴링 Pool Layer 에는 ...

2021년 10월 5일 · 0개의 댓글

by **Robert Lee** 2



밑바닥부터 시작하는 딥러닝 - 7장 CNN

Chapter 7. 합성곱 신경망(CNN) 7.1 전체 구조 CNN에서는 새로운 합성곱 계층과 풀링 계층이 추가된다. CNN 계층은 conc-relu-pooling 흐름으로 연결된다. 여기서 중요한 것은 출력에 가까운 층에서는 affine-relu 구성을 사용할 수 있다. 7.2 합성곱 계층 CNN에서는 패딩, 스트라이드 등 CNN 고유의 용어가 등...

2020년 1월 28일 · 0개의 댓글

by **DSC W/S** 1