

Calculating Parameters of Convolutional and Fully Connected Layers with Keras

Explain how to calculate the number of params and output shape of convolutional and pooling layers



Yan Ding · Follow

6 min read · Oct 15, 2020



285



1



When we build a model of deep learning, we always use a convolutional layer followed by a pooling layer and several fully-connected layers. It is necessary to know how many parameters in our model as well as the output shape of each layer. Let's first see [LeNet-5](#)[1] which a classic architecture of the convolutional neural network.

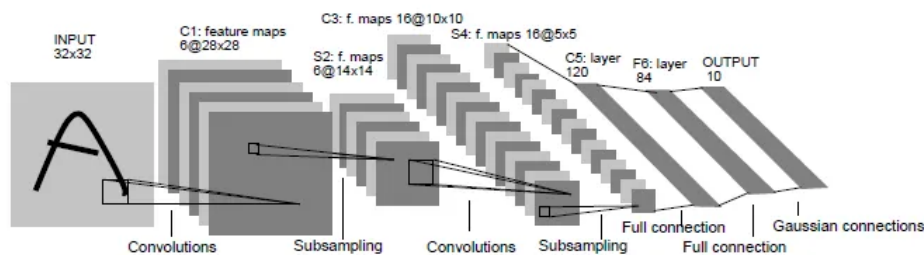


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Fig1. LeNet-5

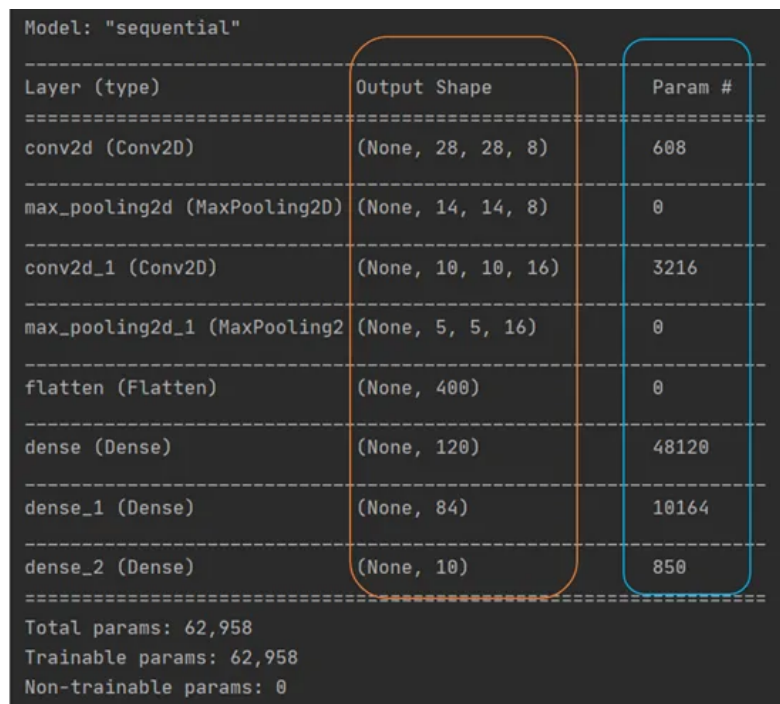
The input shape is (32,32,3). **The first layer** is the convolutional layer, the kernel size is (5,5), the number of filters is 8. Followed by a max-pooling layer with kernel size (2,2) and stride is 2. **The second layer** is another convolutional layer, the kernel size is (5,5), the number of filters is 16. Followed by a max-pooling layer with kernel size (2,2) and stride is 2. **The third layer** is a fully-connected layer with 120 units. **The fourth layer** is a

fully-connected layer with 84 units. **The output layer** is a softmax layer with 10 outputs.

Now let's build this model in Keras.

```
from tensorflow.keras import Sequential
from tensorflow.keras import layers
model = Sequential()
model.add(layers.Conv2D(8, (5,5), activation='relu', input_shape=(32,32,3)))
model.add(layers.MaxPooling2D((2,2), strides=2))
model.add(layers.Conv2D(16, (5,5), activation='relu'))
model.add(layers.MaxPooling2D((2,2), strides=2))
model.add(layers.Flatten())
model.add(layers.Dense(120, activation='relu'))
model.add(layers.Dense(84, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

We can see the summary of the model as follows:



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 8)	608
max_pooling2d (MaxPooling2D)	(None, 14, 14, 8)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	3216
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten (Flatten)	(None, 400)	0
dense (Dense)	(None, 120)	48120
dense_1 (Dense)	(None, 84)	10164
dense_2 (Dense)	(None, 10)	850
Total params: 62,958		
Trainable params: 62,958		
Non-trainable params: 0		

Fig2. Model summary

1 Calculating the output shape of Conv layers

Let's first see the orange box which is the output shape of each layer. Before we dive in, there is an equation for calculating the output of convolutional layers as follows:

$$output = \frac{input - kernel_size + 2 * padding}{stride} + 1$$

The input shape is (32,32,3), kernel size of first Conv Layer is (5,5), with no padding, the stride is 1, so the output size is (32-5)+1=28. And the number of filters is 8. So the output shape of the first Conv layer is (28,28,8). Followed by a max-pooling layer, the method of calculating pooling layer is as same as the Conv layer. The kernel size of max-pooling layer is (2,2) and stride is 2, so output size is (28-2)/2 +1 = 14. After pooling, the output shape is (14,14,8). You can try calculating the second Conv layer and pooling layer on your own. We skip to the output of the second max-pooling layer and have the output shape as (5,5,16). Before feed into the fully-connected layer, we need first flatten this output. So we got the vector of 5*5*16=400. Next, we need to know the number of params in each layer.

Top highlight

2 Calculating number of Params

2.1 Fully Connected Layer

Remember how to calculate the number of params of a simple fully connected neural network as follows:

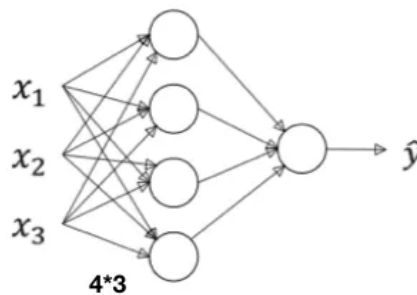


Fig3. A simple fully connected neural network

For one training example, the input is [x1,x2,x3] which has 3 dimensions(e.g. for house pricing prediction problem, input has [squares, number of bedrooms, number of bathrooms]). The first hidden layer has 4 units. Recap how to calculate the first-layer unit (suppose the activation function is the sigmoid function) as follows:

$$a = \sigma(Wx + b)$$

So the dimension of W is $(4, 3)$, and the number of param W is $4*3$, and the dimension of b is $(4, 1)$. The total params of the first hidden layer are $4*3+4=16$. More generally, we can arrive at the dimension of W and b as follows:

$$W^{[L]} : (n^{[L]}, n^{[L-1]})$$

$$b^{[L]} : (n^{[L]}, 1)$$

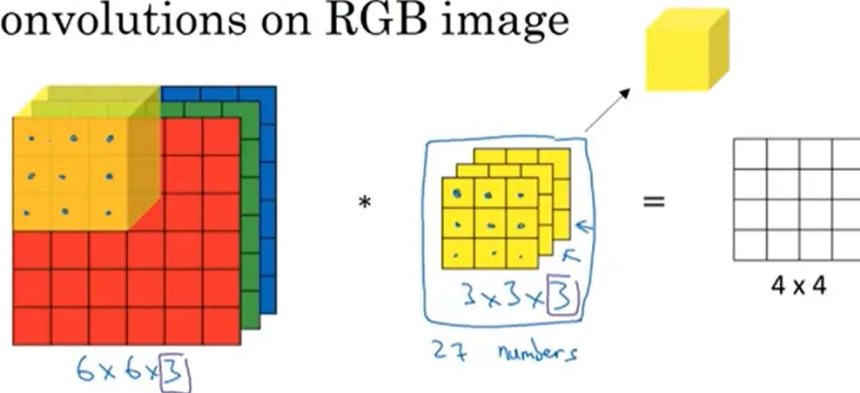
L is the L layer. $n^{[L]}$ is the number of units in the L layer. So the number of params for the L layer is:

$$\text{params} = n^{[L]} * n^{[L-1]} + n^{[L]} \quad (\text{layer } L)$$

2.2 Convolutional layer

The calculation of params of convolutional layers is different especially for volume. Suppose we have an image with size of $(32, 32, 3)$, and the kernel size of $(3, 3)$, the shape of params should be $(3, 3, 3)$ which is a cube as follows:

Convolutions on RGB image



Coursera: Week 1 "Convolutions Over Volume", Course 3 "Convolutional Neural Networks" of Deep learning Specialization[2]

The yellow cube contains all params for one filter. And don't forget about the bias b . Each cube has one bias. So the number of params for one filter is $3*3*3 + 1 = 28$. If there are 2 filters in first layer, the total number of params is $28*2 = 56$. More generally, we can arrive at:

$$one\ cube = k * k * n[l - 1] + 1$$

$$total\ params = one\ cube * filter_{num} = (k * k * n[l - 1] + 1) * n[l]$$

k is the kernel size, n[L] is the number of filters in layer L and n[L-1] is the number of filters in layer L-1 which is also the number of channels of the cube.

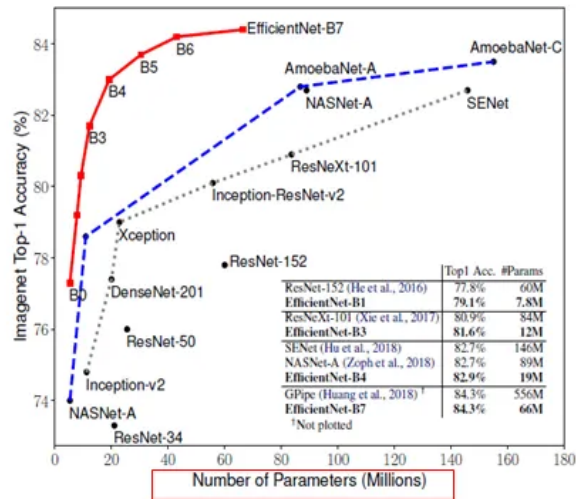
Now Let's see our example. The blue box in Fig2 shows the number of params of each layer. Input shape is (32, 32, 3). The kernel size of the first Conv layer is (5,5) and the number of filters is 8. The number of one filter is $5*5*3 + 1=76$. There are 8 cubes, so the total number is $76*8=608$.

The pooling layer has no params. The second Conv layer has (5,5) kernel size and 16 filters. Remember the cube has 8 channels which is also the number of filters of last layer. So the number of params is $(5*5*8+1)*16 = 3216$.

Flatten the output of the second max-pooling layer and get the vector with 400 units. Flatten also has no params. The third layer is a fully-connected layer with 120 units. So the number of params is $400*120+120=48120$. It can be calculated in the same way for the fourth layer and get $120*84+84=10164$. The number of params of the output layer is $84*10+10=850$. Now we have got all numbers of params of this model.

3 Summary

Having a good knowledge of the output dimensions of each layer and params can help to better understand the construction of the model. Furthermore, it can also help you to know how many updates each iteration does when training the model. Looking at popular models such as EfficientNet[3], ResNet-50, Xception, Inception, and BERT [4], LayoutLM[5], it is necessary to look at the model size rather than only accuracy. Because the model size affects the speed of inference as well as the computing source it would consume.



EfficientNet:Rethinking Model Scaling for Convolutional Neural Networks

Modality	Model	Precision	Recall	F1	#Parameters
Text only	BERT _{BASE}	0.9099	0.9099	0.9099	110M
	RoBERTa _{BASE}	0.9107	0.9107	0.9107	125M
	BERT _{LARGE}	0.9200	0.9200	0.9200	340M
	RoBERTa _{LARGE}	0.9280	0.9280	0.9280	355M
Text + Layout MVLM	LayoutLM _{BASE} (500K, 6 epochs)	0.9388	0.9388	0.9388	113M
	LayoutLM _{BASE} (1M, 6 epochs)	0.9380	0.9380	0.9380	113M
	LayoutLM _{BASE} (2M, 6 epochs)	0.9431	0.9431	0.9431	113M
	LayoutLM _{BASE} (11M, 2 epochs)	0.9438	0.9438	0.9438	113M
Text + Layout MVLM+MDC	LayoutLM _{BASE} (1M, 6 epochs)	0.9402	0.9402	0.9402	113M
	LayoutLM _{BASE} (11M, 1 epoch)	0.9460	0.9460	0.9460	113M
Text + Layout MVLM	LayoutLM _{LARGE} (1M, 6 epochs)	0.9416	0.9416	0.9416	343M
	LayoutLM _{LARGE} (11M, 1 epoch)	0.9524	0.9524	0.9524	343M
Text + Layout + Image MVLM	LayoutLM _{BASE} (1M, 6 epochs)	0.9416	0.9416	0.9416	160M
	LayoutLM _{BASE} (11M, 2 epochs)	0.9467	0.9467	0.9467	160M
Baseline	Ranking 1 st in SROIE	0.9402	0.9402	0.9402	-

LayoutLM:Pre-training of Text and Layout for Document Image Understanding

Reference

[1] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient-Based Learning Applied to Document Recognition.” PROC. OF THE IEEE, November 1998.

[2] Andrew Ng, week 1 of “Convolutional Neural Networks” Course in “Deep Learning Specialization”, Coursera.

[3] Mingxing Tan, Quoc V. Le, “EfficientNet:Rethinking Model Scaling for Convolutional Neural Networks”. May 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, May 2019.

[5] Yiheng Xu, Minghao Li, “LayoutLM: Pre-training of Text and Layout for Document Image Understanding”. Dec 2019.

[Convolutional Network](#)[Pooling](#)[Fcn](#)[Deep Learning](#)[Lenet](#)

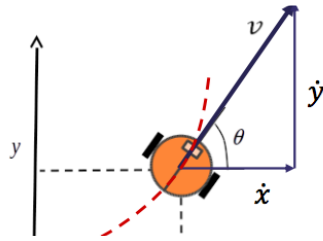
Written by Yan Ding

219 Followers

Deep Learning Engineer || Kaggle Expert <https://shuffleai.blog/>
<https://www.linkedin.com/in/dingyan89/> <https://www.kaggle.com/dingyan>

[Follow](#)

More from Yan Ding

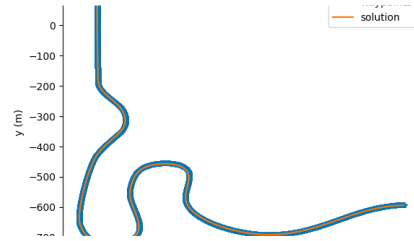


Yan Ding

Simple Understanding of Kinematic Bicycle Model

1. Introduction

Feb 15, 2020 197 5

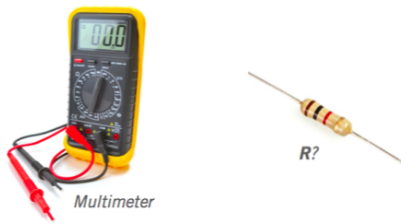


Yan Ding

Three Methods of Vehicle Lateral Control: Pure Pursuit, Stanley and

Introduction

Mar 6, 2020 344 9



Yan Ding

Least Squares, Recursive Least Squares, Kalman Filters, and

We will cover basic ideas of least squares, weighted least squares. Meanwhile, we will

Mar 20, 2020 295



Yan Ding

How to solve a simple control system problem with Laplace

Why we need Laplace Transform?

Feb 25, 2020 71 2

[See all from Yan Ding](#)

Recommended from Medium



 Jo Wang

Deep Learning Part 2—Neural Network and the critical Activation

Neural Network Structure




 Jun 28  1  



 Vyacheslav Efimov in Towards Data Science

Understanding Deep Learning Optimizers: Momentum, AdaGrad,

Gain intuition behind acceleration training techniques in neural networks

 Dec 30, 2023  452  4  

Lists



Staff Picks
763 stories · 1433 saves



Stories to Help You Level-Up at Work
19 stories · 862 saves




Self-Improvement 101
20 stories · 2995 saves



Productivity 101
20 stories · 2545 saves

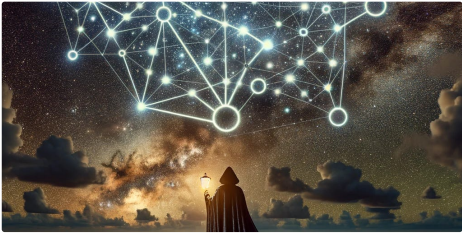


 Mounica Kommajosyula

Getting Started with Hyperparameters: A Beginner’s

In machine learning, there is a lot of talk about building powerful models, feeding them data,

★ Nov 4  ...



 Jorgecardete in The Deep Hub

Convolutional Neural Networks: A Comprehensive Guide

Exploring the power of CNNs in image analysis

Feb 7  2.6K  38  ...

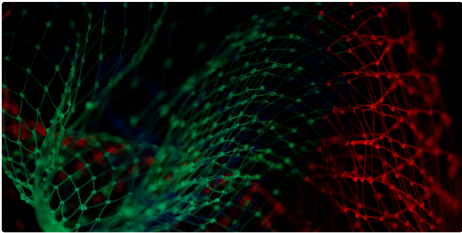


 Deepak N R in Python in Plain English

YOLOv8 vs YOLOv11: A Comparison

YOLO models have been state-of-the-art in computer vision for real-time object

★ Oct 9  194  3  ...



 Sanjithkumar

Optimization Algorithms In Deep Learning

Gradient Descent and its limitations

Jul 6  3  ...

See more recommendations