

Open in app ↗



Search

Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)

# A Walkthrough of Convolutional Neural Network — Hyperparameter Tuning



Suki Lau · Follow

Published in Towards Data Science

4 min read · Jul 11, 2017



Listen



Share



More

Among the diverse deep learning architecture, convolutional neural network (convnets for short) stands out for its unprecedented performance on computer vision. It is an artificial neural network inspired by the animal visual cortex and has been successfully applied to vision recognition tasks.



In this article, I will provide a walkthrough on convnet, how to tune hyperparameters and how to visualize the hidden convolutional layers.

## Motivation

First things first, what do we want our computer to do? When we see a cat running in backyard or sleeping on a couch, our minds subconsciously recognize it as a cat. We want our computer to do similar things for us, that is to take an image as input, figure out its unique features and label the image as an output. This is basically what convnet can do for us.

## What is convnet?

At its most basic, convnet is a special kind of neural networks which contains at least one convolutional layer. A typical convnet structure takes an image, pass it through a series of convolutional layers, nonlinear activation layers, pooling (downsampling) and a fully connected layer to output the classification labels.

A convnet differs from a regular neural network by the use of convolutional layer. In a regular neural network, we use the entire image to train the network. It works well for simple centered image (for example a centered handwritten digit image) but fails to recognize image with more complex variation (for example a running cat in backyard). Having more hidden layers to learn abstract features would help but it is rather impractical as we need far too many neurons to train and store in memory.

On the other hand, a convnet recognizes a object by first looking for low level features such as edges, lines and curves, and then building up more abstract features (for example what makes a cat a cat) through series of convolutional layers. During the learning process in the convolutional layer, the network learns individual parts of an object by shared filters (or feature detectors) on small regions of the image, and add them up to build abstract features. The use of shared filters largely reduce the actual parameter learning. A convnet also generalizes better for complex image recognition as learned filters can be reused to detect abstract features compositionally through convolutional layers.

## Hyperparameter tuning

Tuning hyperparameters for deep neural network is difficult as it is slow to train a deep neural network and there are numerous parameters to configure. In this part, we briefly survey the hyperparameters for convnet.

### Learning rate

Learning rate controls how much to update the weight in the optimization algorithm. We can use fixed learning rate, gradually decreasing learning rate, momentum based methods or adaptive learning rates, depending on our choice of optimizer such as SGD, Adam, Adagrad, AdaDelta or RMSProp.

### Number of epochs

Number of epochs is the the number of times the entire training set pass through the neural network. We should increase the number of epochs until we see a small gap between the test error and the training error.

### Batch size

Mini-batch is usually preferable in the learning process of convnet. A range of 16 to 128 is a good choice to test with. We should note that convnet is sensitive to batch size.

### Activation function

Activation function introduces non-linearity to the model. Usually, rectifier works well with convnet. Other alternatives are sigmoid, tanh and other activation functions depending on the task.

### **Number of hidden layers and units**

It is usually good to add more layers until the test error no longer improves. The trade off is that it is computationally expensive to train the network. Having a small amount of units may lead to underfitting while having more units are usually not harmful with appropriate regularization.

### **Weight initialization**

We should initialize the weights with small random numbers to prevent dead neurons, but not too small to avoid zero gradient. Uniform distribution usually works well.

### **Dropout for regularization**

Dropout is a preferable regularization technique to avoid overfitting in deep neural networks. The method simply drops out units in neural network according to the desired probability. A default value of 0.5 is a good choice to test with.

### **Grid search or randomized search**

Manually tuning hyperparameter is painful and also impractical. There are two generic approaches to sampling search candidates. Grid search exhaustively search all parameter combinations for given values. Random search sample a given number of candidates from a parameter space with a specified distribution.

To implement grid search more efficiently, it is better to start with coarse ranges of hyperparameter values in the initial stage. It is also helpful to perform coarse grid search for smaller number of epochs or smaller training set. The next stage would then perform a narrow search with more epochs or entire training set.

Although grid search is useful in many machine learning algorithms, it is not efficient in tuning hyperparameter for deep neural network as the number of parameters increases, computation grows exponentially. Random search has been found more efficient compared to grid search in hyperparameter tuning for deep neural network (see [Paper on “Random Search for Hyper-Parameter Optimization”](#)). It is also helpful to combine with some manual tuning on hyperparameters based on prior experience.

## **Visualization**

We can better understand how convnet learns features if we can visualize the convolutional layers. Two straight forward ways are to visualize the activations and the weights. The activations usually look more sparse and localized as training progresses. If the activations have dark pixels for many different inputs, it may indicate dead filters (zero activation map) due to high learning rates.

Well-trained networks usually have nice and smooth filters without any noisy patterns. Noisy patterns observed in the weights may indicate that network hasn't been trained long enough, or low regularization strength that may have led to overfitting.

### References:

- [Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville](#)
- [Practical recommendations for gradient-based training of deep architectures by Yoshua Bengio](#)
- [Random Search for Hyper-Parameter Optimization](#)
- [Convolutional Neural Networks: Architectures, Convolution / Pooling Layers](#)
- [Understanding and Visualizing Convolutional Neural Networks](#)
- [Understanding Neural Networks Through Deep Visualization](#)
- [Understanding Convolutions](#)
- [How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras](#)

[Machine Learning](#)[Neural Networks](#)[Deep Learning](#)[Data Science](#)[Follow](#)

## Written by Suki Lau

355 Followers · Writer for Towards Data Science

always a student

### More from Suki Lau and Towards Data Science



Suki Lau in Towards Data Science

### Image Augmentation for Deep Learning

Deep networks need large amount of training data to achieve good performance. To build a powerful image classifier using very little...

3 min read · Jul 11, 2017



382



4







Tim Sumner in Towards Data Science

## A New Coefficient of Correlation

What if you were told there exists a new way to measure the relationship between two variables just like correlation except possibly...

10 min read · Mar 31, 2024



3.4K



42



Youness Mansar in Towards Data Science

## Meet the NiceGUI: Your Soon-to-be Favorite Python UI Library

Build custom web apps easily and quickly

8 min read · Apr 17, 2024

 1.1K

 5







 Suki Lau in Towards Data Science




## Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning

When training deep neural networks, it is often useful to reduce learning rate as the training progresses. This can be done by using...

6 min read · Jul 29, 2017

 1.8K

 10

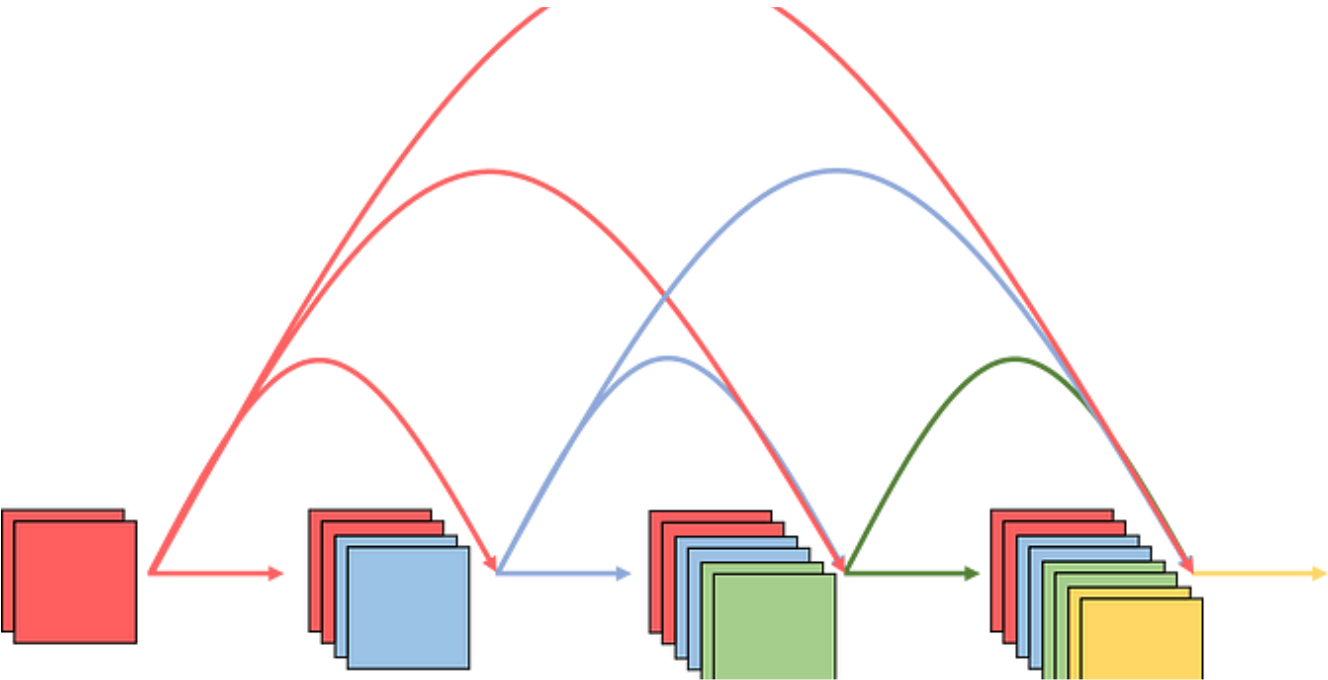





See all from Suki Lau

See all from Towards Data Science

Recommended from Medium



 Alejandro Ito Aramendia

DenseNet : A Complete Guide

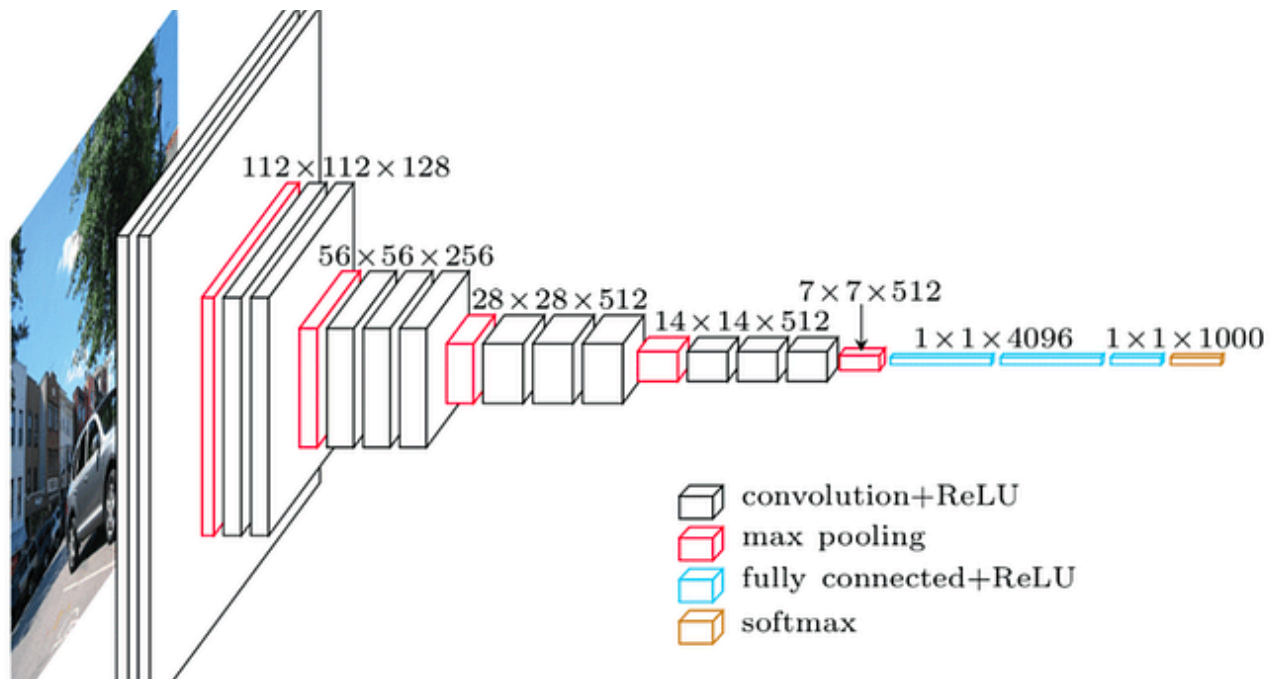
Extending the ResNet to improve performance.

5 min read · Mar 1, 2024

 85  1





Namita

## Convolutional Neural Networks

Before kickstarting into CNNs we must first know about an image. What really is an RGB image?

5 min read · Dec 12, 2023



65



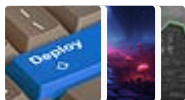
1



...



### Lists



#### Predictive Modeling w/ Python

20 stories · 1153 saves



#### Practical Guides to Machine Learning

10 stories · 1391 saves



#### Natural Language Processing

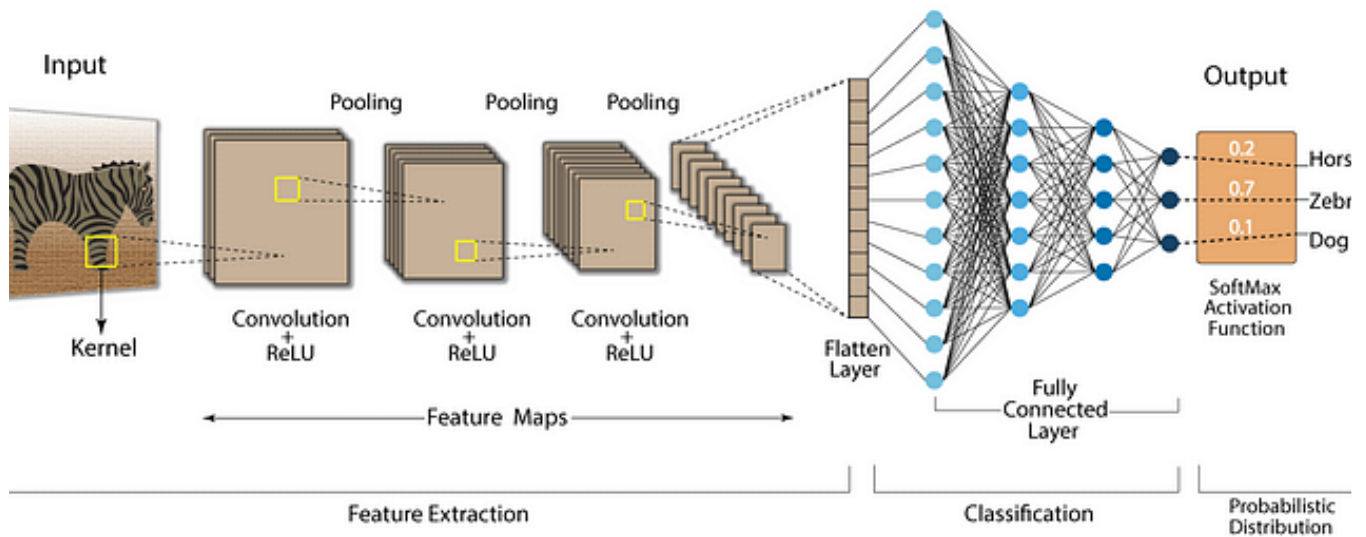
1429 stories · 924 saves



#### data science and AI

40 stories · 146 saves

## Convolution Neural Network (CNN)



Koushik

## Understanding Convolutional Neural Networks (CNNs) in Depth

Convolutional Neural Networks skillfully capturing and extracting patterns from data, revealing the hidden artistry within pixels.

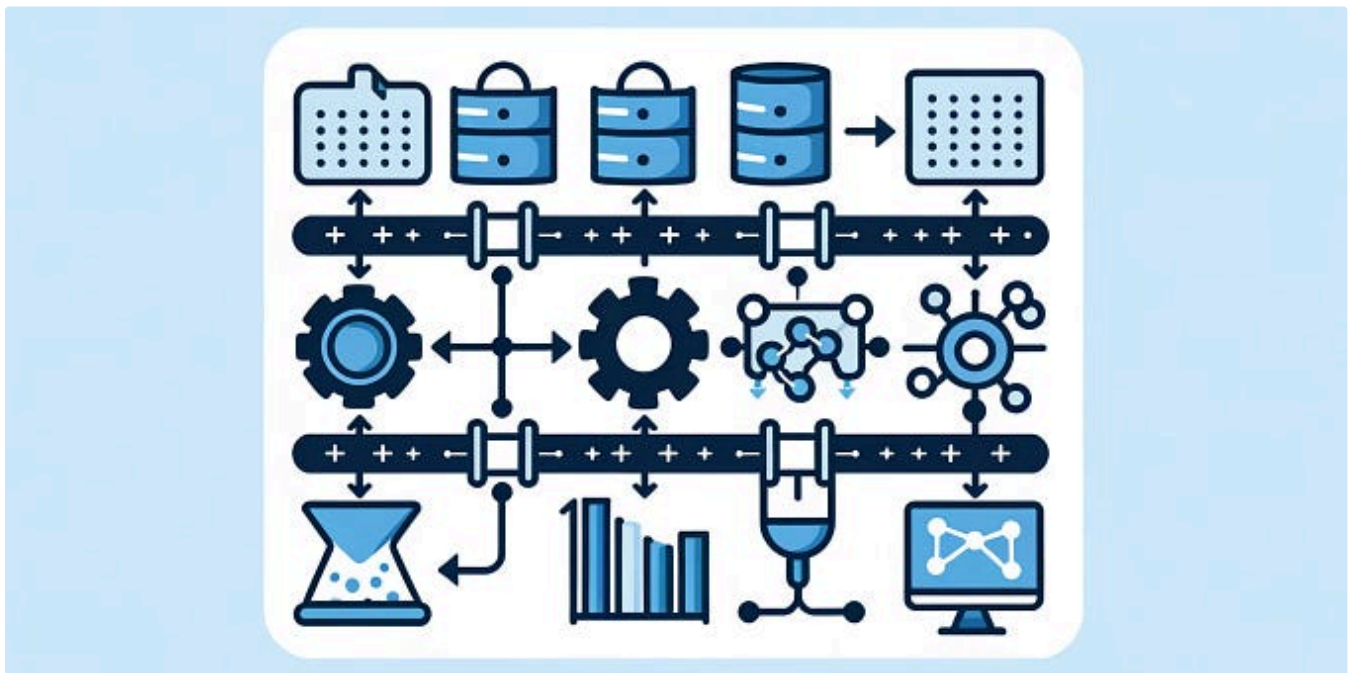
12 min read · Nov 28, 2023



600



6



Sandaruwan Herath in Data Science and Machine Learning

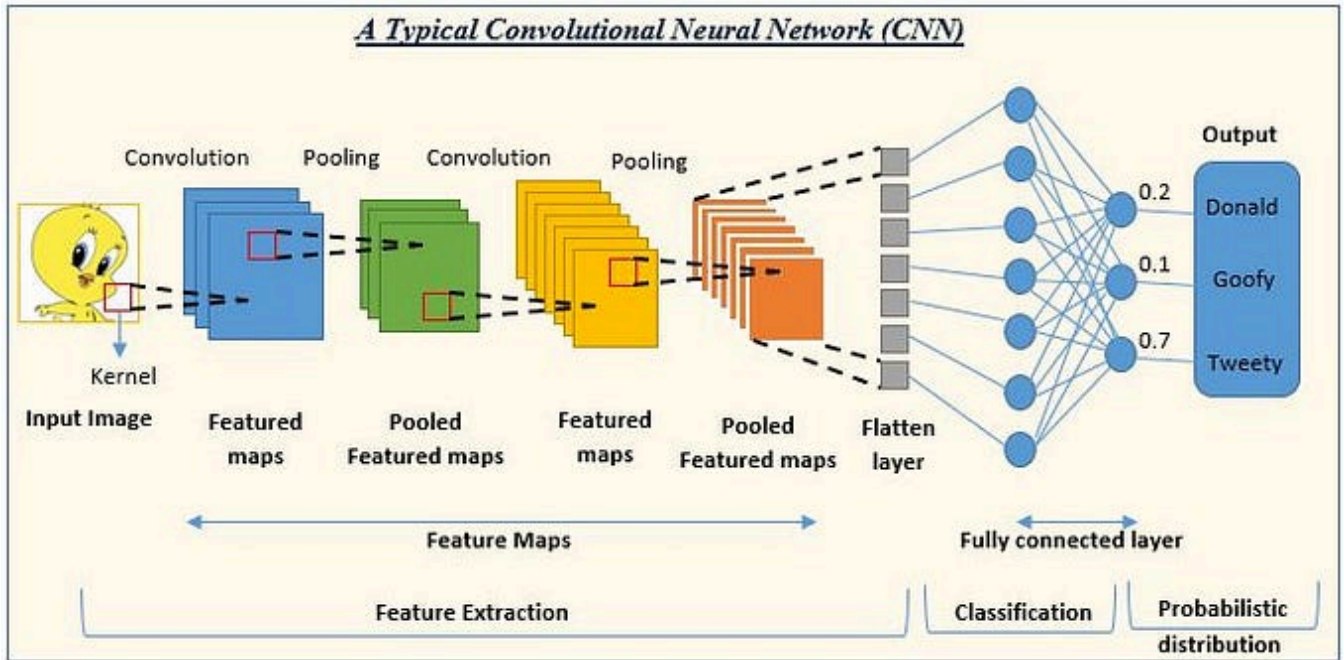
## EfficientNetB0 architecture—Stem Layer


EfficientNet’s layers are based on a compound scaling method that uniformly scales the depth, width, and resolution of the network, with...


11 min read · Jan 14, 2024

 28 

 ...




 Priyanshu singh



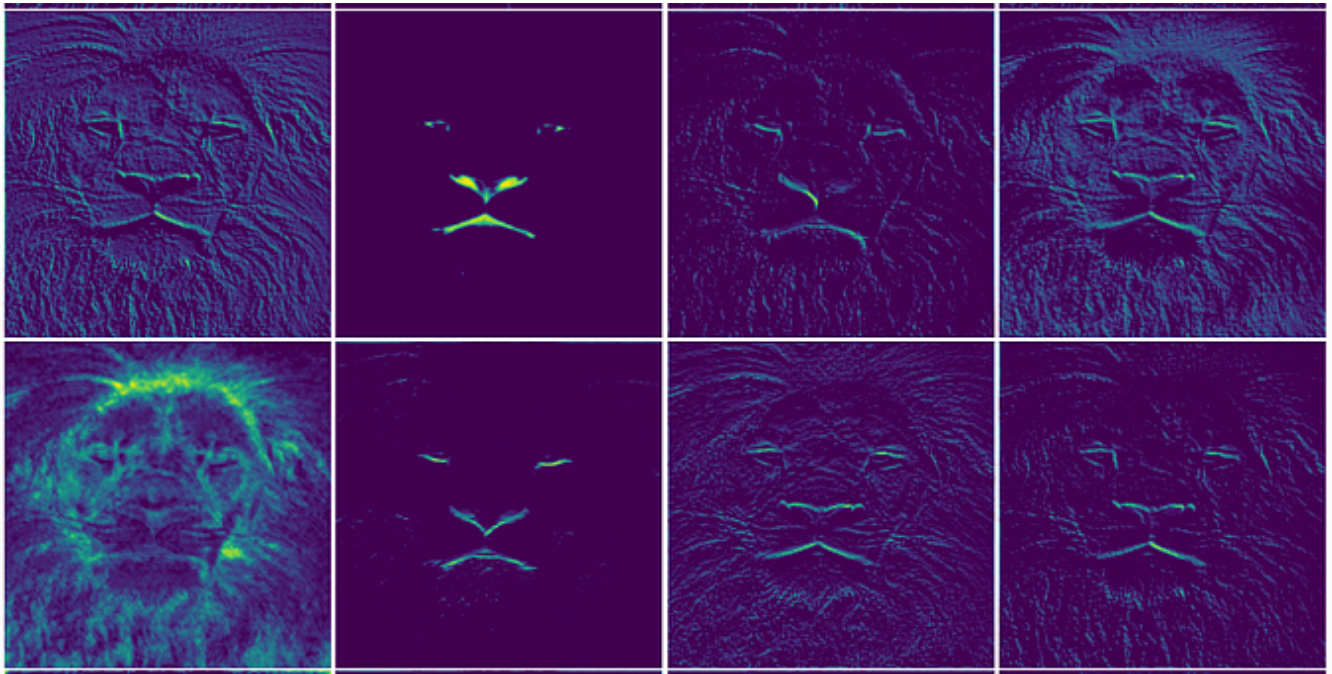
## A Brief Dive into Classic Convolutional Neural Networks: LeNet, AlexNet, VGG, ResNet, InceptionNet...

Introduction

2 min read · Nov 22, 2023

 58 

 ...



Rodrigo Silva in Towards Data Science

## Exploring Feature Extraction with CNNs

Using a Convolutional Neural Network to check specialization in feature extraction

8 min read · Nov 26, 2023



371



1



See more recommendations