Image Processing in Python: Algorithms, Tools, and Methods You Should Know

9 mins read Author Neetika Khandelwal Updated November 15th, 2021

Images define the world, each image has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as **Image Processing**.

It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field.

Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output.

Let’s get into it!

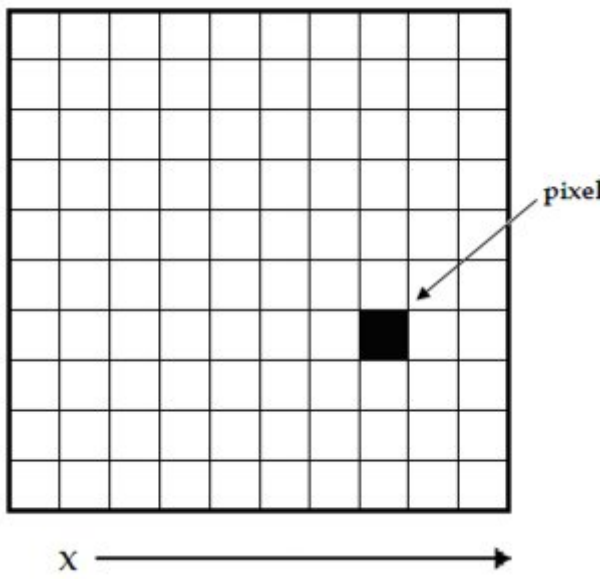
What is image processing?

As the name says, image processing means processing the image and this may include many different techniques until we reach our goal.

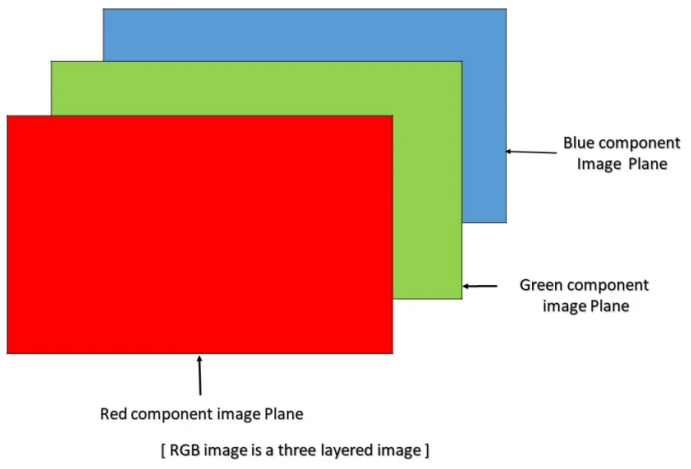
The final output can be either in the form of an image or a corresponding feature of that image. This can be used for further analysis and decision making.

But what is an image?

An image can be represented as a 2D function F(x,y) where x and y are spatial coordinates. The amplitude of F at a particular value of x,y is known as the intensity of an image at that point. If x,y, and the amplitude value is finite then we call it a digital image. It is an array of pixels arranged in columns and rows. Pixels are the elements of an image that contain information about intensity and color. An image can also be represented in 3D where x,y, and z become spatial coordinates. Pixels are arranged in the form of a matrix. This is known as an **RGB image**.



Source



Source

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

1. Morphological Image Processing

Morphological image processing tries to remove the imperfections from the binary images because binary regions produced by simple thresholding can be distorted by noise. It also helps in smoothing the image using opening and closing operations.

Morphological operations can be extended to grayscale images. It consists of non-linear operations related to the structure of features of an image. It depends on the related ordering of pixels but on their numerical values. This technique analyzes an image using a small template known as **structuring element** which is placed on different possible locations in the image and is compared with the corresponding neighbourhood pixels. A structuring element is a small matrix with 0 and 1 values.

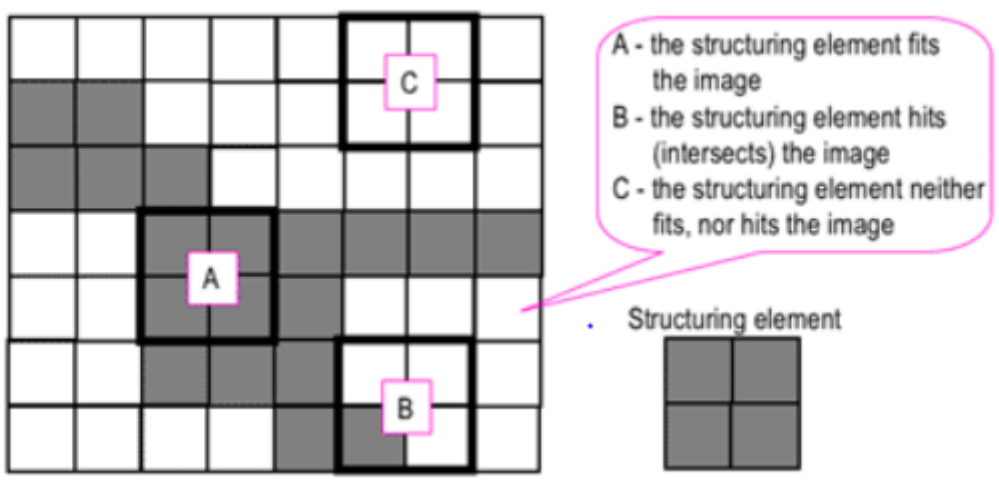
Let's see the two fundamental operations of morphological image processing, **Dilation and Erosion:**

- **dilation** operation adds pixels to the boundaries of the object in an image
- **erosion** operation removes the pixels from the object boundaries.

The number of pixels removed or added to the original image depends on the size of the structuring element.

At this point you may be thinking "what is a structuring element?" Let me explain:

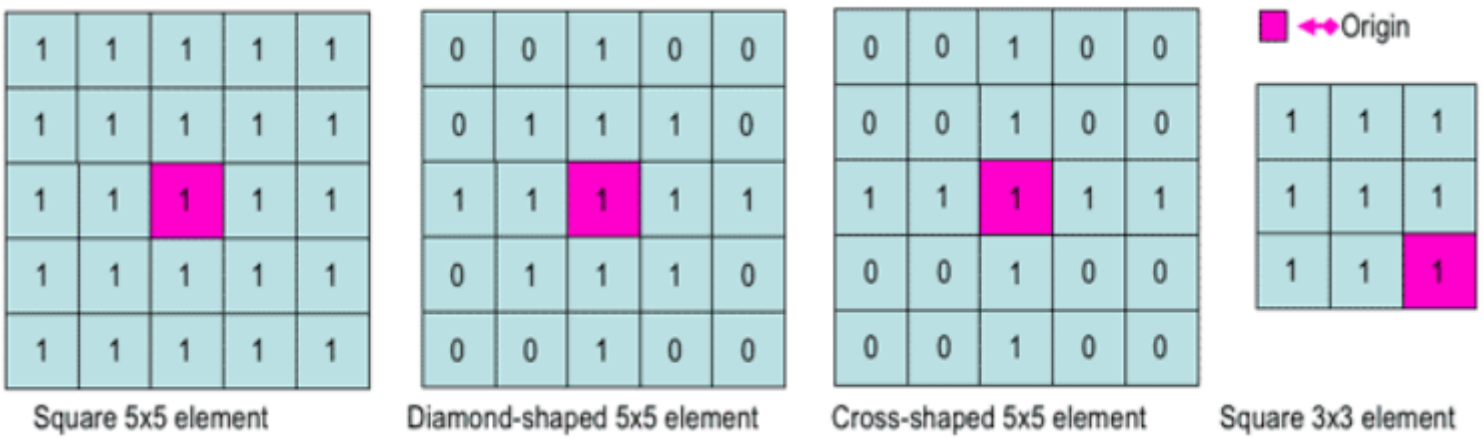
Structuring element is a matrix consisting of only 0's and 1's that can have any arbitrary shape and size. It is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels.



Source

The square structuring element 'A' fits in the object we want to select, the 'B' intersects the object and 'C' is out of the object.

The zero-one pattern defines the configuration of the structuring element. It's according to the shape of the object we want to select. The center of the structuring element identifies the pixel being processed.



Source

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

neptuneblog

Search all articles

How to Use Neptune ML Experiment Tracking ML Model Management

Dilation | *Source*

Erosion | *Source*

2. Gaussian Image Processing

Gaussian blur which is also known as gaussian smoothing, is the result of blurring an **image** by a **Gaussian** function.

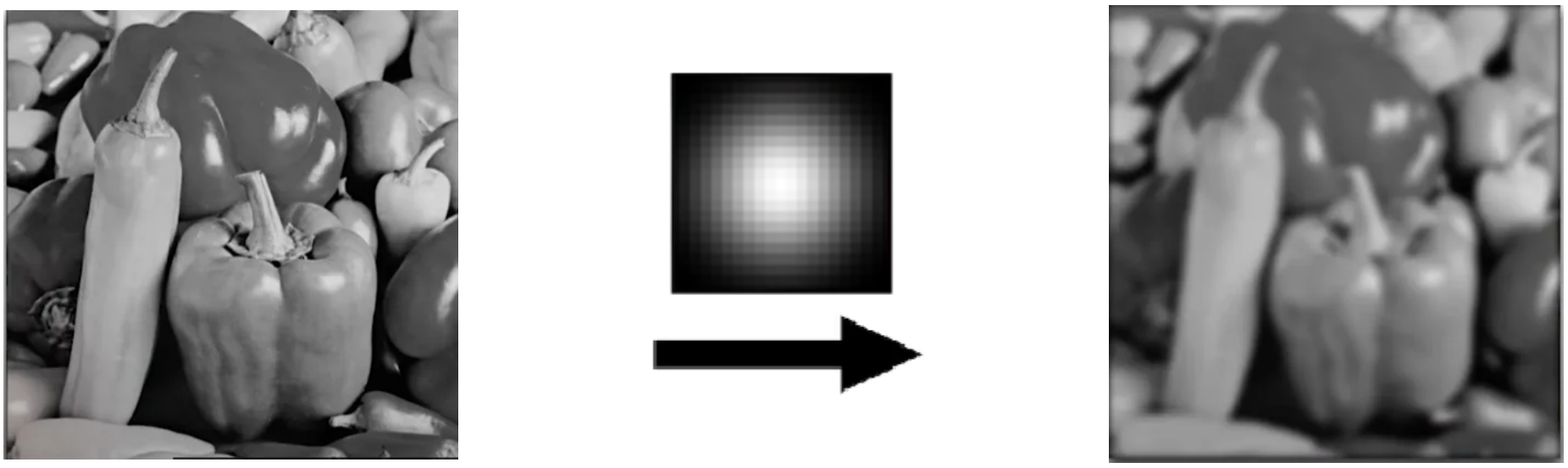
It is **used to reduce image noise and reduce details**. The visual effect of this blurring technique is similar to looking at an image through the translucent screen. It is sometimes used in computer vision for image enhancement at different scales or as a data augmentation technique in deep learning.

The basic gaussian function looks like:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

In practice, it is best to take advantage of the Gaussian blur's separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass. Let's see an example to understand what gaussian filters do to an image.

If we have a filter which is normally distributed, and when its applied to an image, the results look like this:



Original Filter Result

Source

You can see that some of the edges have little less detail. The filter is giving more weight to the pixels at the center than the pixels

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

neptuneblog

Search all articles

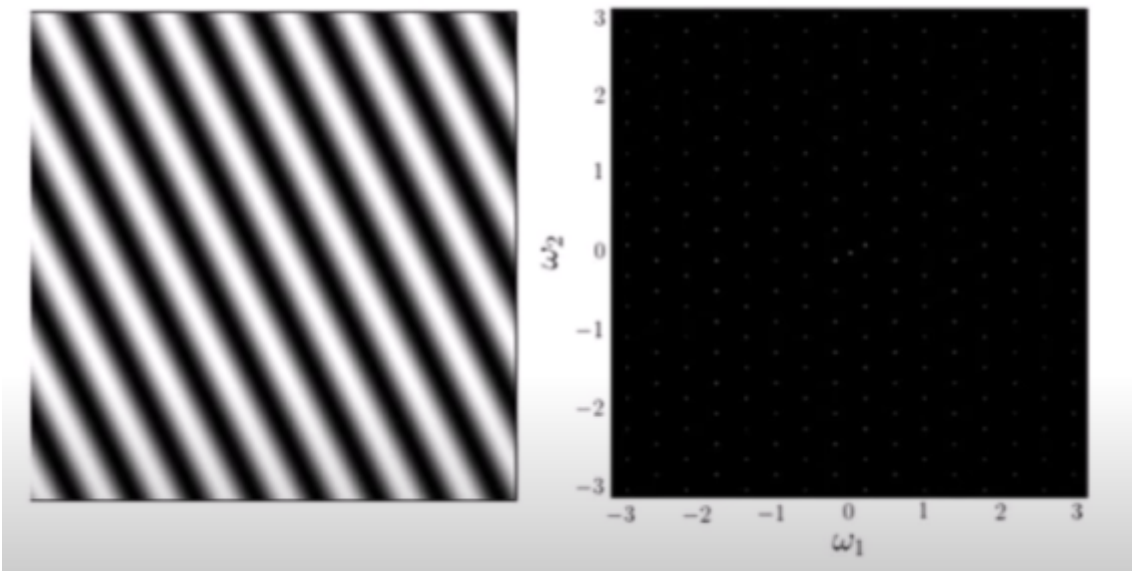
How to Use Neptune ML Experiment Tracking ML Model Management

Since we are talking about images, we will take discrete fourier transform into consideration.

Let's consider a sinusoid, it comprises of three things:

- Magnitude – related to contrast
- Spatial frequency – related to brightness
- Phase – related to color information

The image in the frequency domain looks like this:



Source

The formula for 2D discrete fourier transform is:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

In the above formula, f(x,y) denotes the image.

The inverse fourier transform converts the transform back to image. The formula for 2D inverse discrete fourier transform is:

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v)\, e^{j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

4. Edge Detection in image processing

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.

This could be very beneficial in extracting useful information from the image because most of the shape information is enclosed in the edges. Classic edge detection methods work by detecting discontinuities in the brightness.

It can rapidly react if some noise is detected in the image while detecting the variations of grey levels. Edges are defined as the local maxima of the gradient.

The most common edge detection algorithm is **sobel edge detection algorithm**. Sobel detection operator is made up of 3*3 convolutional kernels. A simple kernel Gx and a 90 degree rotated kernel Gy. Separate measurements are made by applying both the kernel separately to the image.

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

neptuneblog

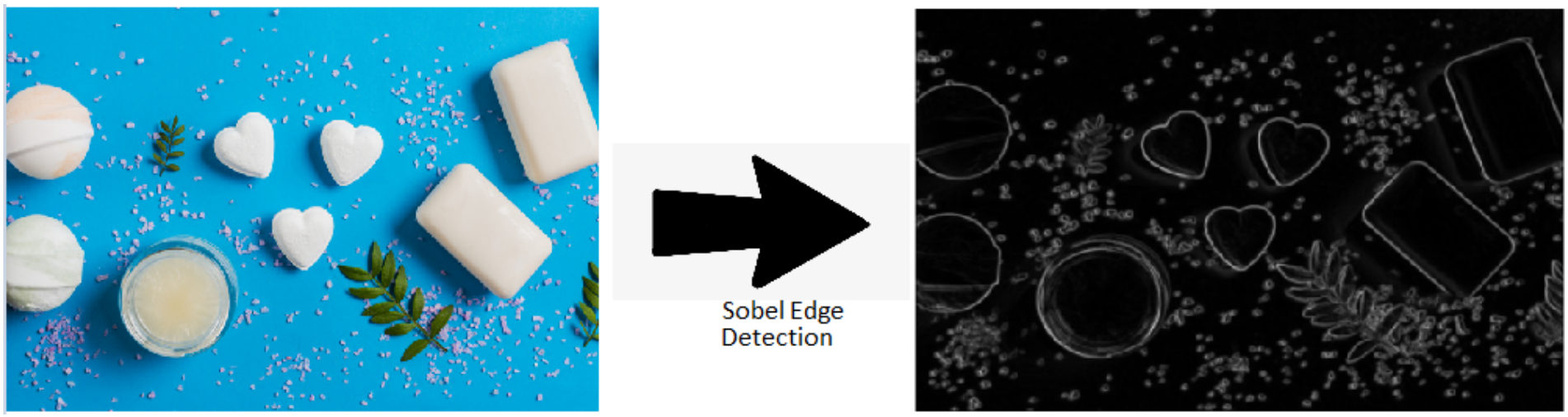
Search all articles

How to Use Neptune ML Experiment Tracking ML Model Management

* denotes the 2D signal processing convolution operation.

Resulting gradient can be calculated as:

$$G = sqrt(Gx^2 + Gy^2)$$



Source

5. Wavelet Image Processing

We saw a Fourier transform but it is only limited to the frequency. Wavelets take both time and frequency into the consideration. This transform is apt for non-stationary signals.

We know that edges are one of the important parts of the image, while applying the traditional filters it's been noticed that noise gets removed but image gets blurry. The wavelet transform is designed in such a way that we get good frequency resolution for low frequency components. Below is the 2D wavelet transform example:

Source

Image processing using Neural Networks

Neural Networks are multi-layered networks consisting of neurons or nodes. These neurons are the core processing units of the neural network. They are designed to act like human brains. They take in data, train themselves to recognize the patterns in the data and then predict the output.

A basic neural network has three layers:

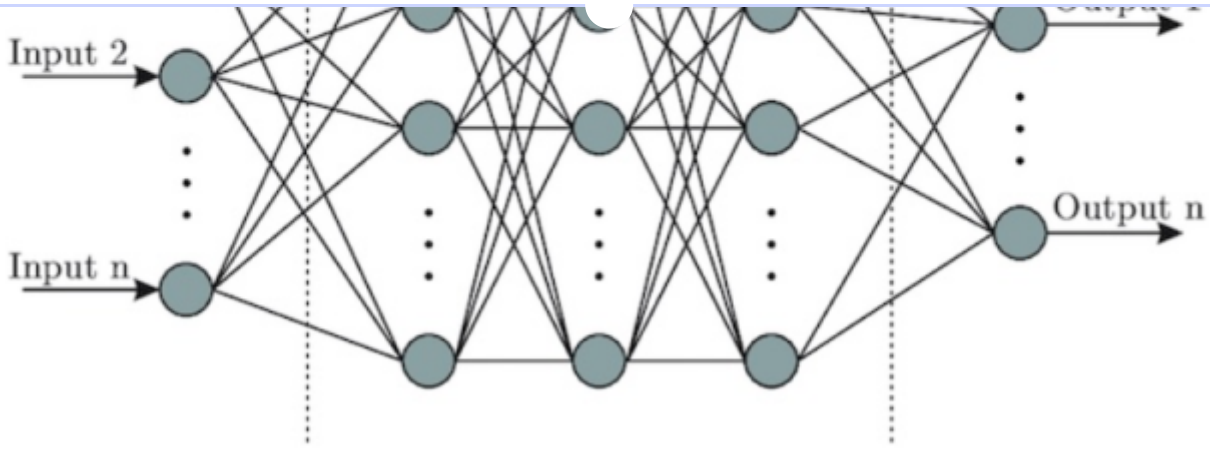
Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

neptuneblog

Search all articles

How to Use Neptune ML Experiment Tracking ML Model Management



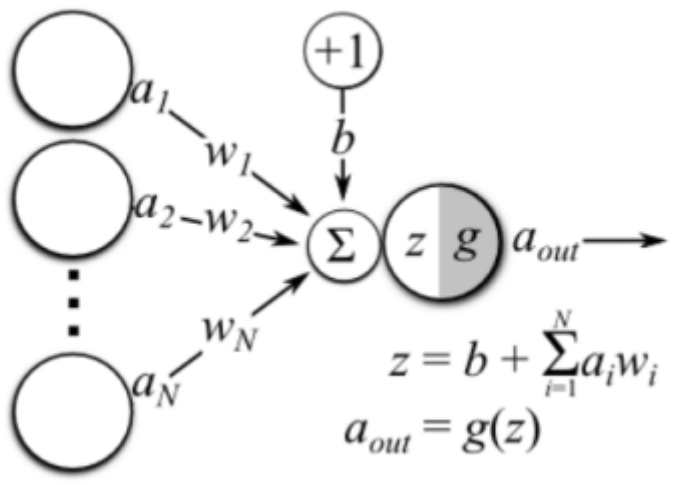
Basic neural network | Source

The input layers receive the input, the output layer predicts the output and the hidden layers do most of the calculations. The number of hidden layers can be modified according to the requirements. There should be atleast one hidden layer in a neural network.

The basic working of the neural network is as follows:

1. Let's consider an image, each pixel is fed as input to each neuron of the first layer, neurons of one layer are connected to neurons of the next layer through channels.
2. Each of these channels is assigned a numerical value known as weight.
3. The inputs are multiplied by the corresponding weights and this weighted sum is then fed as input to the hidden layers.
4. The output from the hidden layers is passed through an activation function which will determine whether the particular neuron will be activated or not.
5. The activated neurons transmits data to the next hidden layers. In this manner, data is propagated through the network, this is known as Forward Propagation.
6. In the output layer, the neuron with the highest value predicts the output. These outputs are the probability values.
7. The predicted output is compared with the actual output to obtain the error. This information is then transferred back through the network, the process is known as Backpropagation.
8. Based on this information, the weights are adjusted. This cycle of forward and backward propagation is done several times on multiple inputs until the network predicts the output correctly in most of the cases.
9. This ends the training process of the neural network. The time taken to train the neural network may get high in some cases.

In the below image, **ai's** is the set of inputs, **wi's** are the weights, **z** is the output and **g** is any activation function.



Operations in a single neuron | Source

Here are some guidelines to prepare data for image processing.

- More data needs to be fed to the model to get the better results.
- Image dataset should be of high quality to get more clear information, but to process them you may require deeper neural networks.
- In many cases RGB images are converted to grayscale before feeding them into a neural network.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

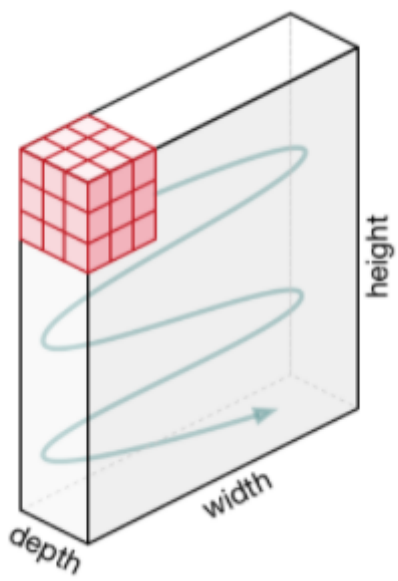
Got it!

neptuneblog

Search all articles

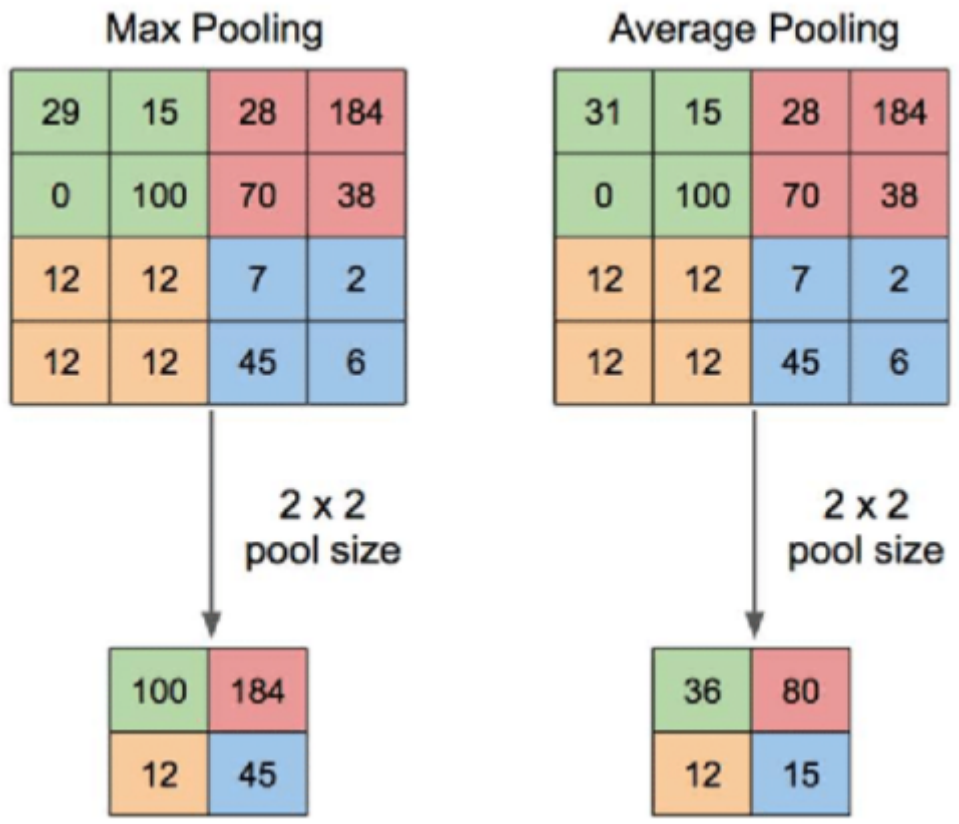
How to Use Neptune ML Experiment Tracking ML Model Management

element involved in carrying out the convolution operation in this la is called the **Kernel/Filter (matrix).** The kernel makes horizontal and vertical shifts based on the **stride rate** until the full image is traversed.



Movement of the kernel | Source

- **Pooling Layer** (*POOL*): This layer is responsible for dimensionality reduction. It helps to decrease the computational power required to process the data. There are two types of Pooling: Max Pooling and Average Pooling. Max pooling returns the maximum value from the area covered by the kernel on the image. Average pooling returns the average of all the values in the part of the image covered by the kernel.



Pooling operation | Source

- **Fully Connected Layer** (*FC*): The **fully connected layer** (**FC**) operates on a flattened input where each input is connected to all neurons. If present, **FC layers** are usually found towards the end of **CNN** architectures.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

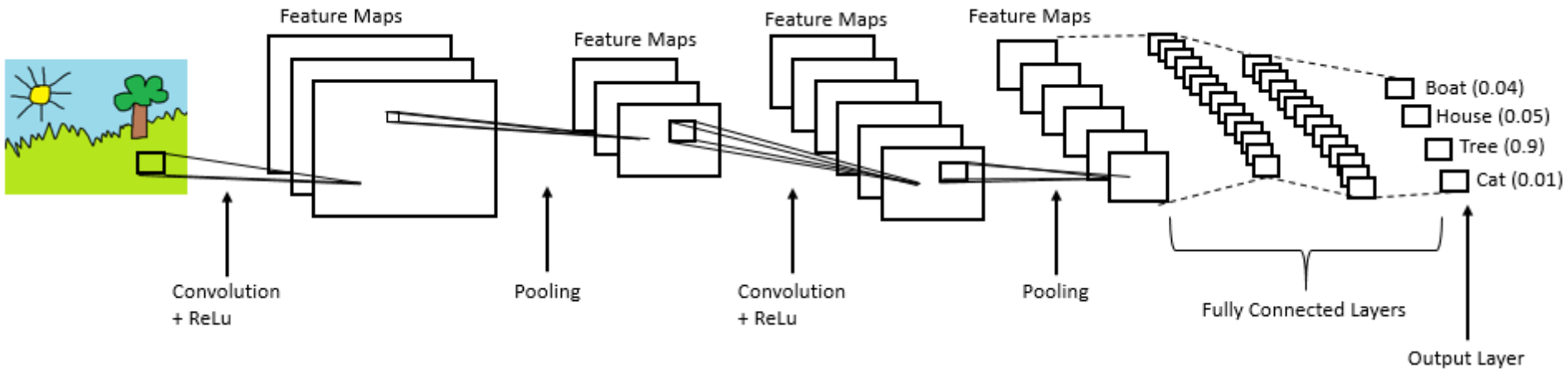
neptuneblog

Search all articles

How to Use Neptune ML Experiment Tracking ML Model Management

Fully connected layers | Source

CNN is mainly used in extracting features from the image with help of its layers. CNNs are widely used in image classification where each input image is passed through the series of layers to get a probabilistic value between 0 and 1.



Source

Generative Adversarial Networks

Generative models use an unsupervised learning approach (there are images but there are no labels provided).

GANs are composed of two models **Generator** and **Discriminator.** *Generator* learns to make fake images that look realistic so as to fool the discriminator and *Discriminator* learns to distinguish fake from real images (it tries not to get fooled).

Generator is not allowed to see the real images, so it may produce poor results in the starting phase while the discriminator is allowed to look at real images but they are jumbled with the fake ones produced by the generator which it has to classify as real or fake.

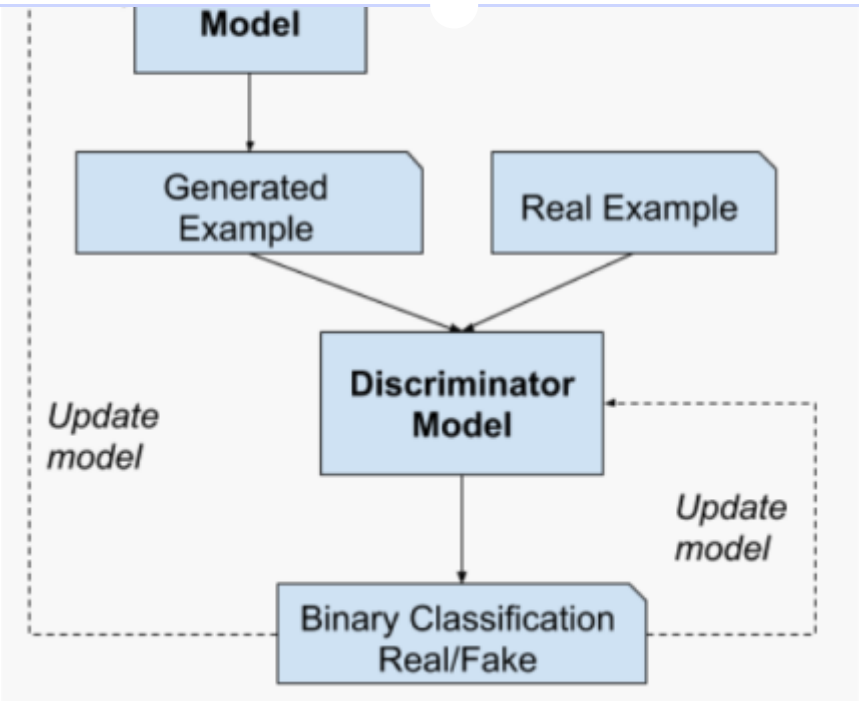
Some noise is fed as input to the generator so that it’s able to produce different examples every single time and not the same type image. Based on the scores predicted by the discriminator, the generator tries to improve its results, after a certain point of time, the generator will be able to produce images that will be harder to distinguish, at that point of time, the user gets satisfied with its results. Discriminator also improves itself as it gets more and more realistic images at each round from the generator.

Popular types of GANs are Deep Convolutional GANs(DCGANs), Conditional GANs(cGANs), StyleGANs, CycleGAN, DiscoGAN, GauGAN and so on.

GANs are great for image generation and manipulation. Some applications of GANs include : Face Aging, Photo Blending, Super Resolution, Photo Inpainting, Clothing Translation.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!



Source

Image processing tools

1. OpenCV

It stands for Open Source Computer Vision Library. This library consists of around 2000+ optimised algorithms that are useful for computer vision and machine learning. There are several ways you can use opencv in image processing, a few are listed below:

- Converting images from one color space to another i.e. like between BGR and HSV, BGR and gray etc.
- Performing thresholding on images, like, simple thresholding, adaptive thresholding etc.
- Smoothing of images, like, applying custom filters to images and blurring of images.
- Performing morphological operations on images.
- Building image pyramids.
- Extracting foreground from images using GrabCut algorithm.
- Image segmentation using watershed algorithm.

Refer to this link for more details.

2. Scikit-image

It is an open-source library used for image preprocessing. It makes use of machine learning with built-in functions and can perform complex operations on images with just a few functions.

It works with numpy arrays and is a fairly simple library even for those who are new to python. Some operations that can be done using scikit image are :

- To implement thresholding operations use **try_all_threshold()** method on the image. It will use seven global thresholding algorithms. This is in the **filters** module.
- To implement edge detection use **sobel()** method in the **filters** module. This method requires a 2D grayscale image as an input, so we need to convert the image to grayscale.
- To implement gaussian smoothing use **gaussian()** method in the **filters** module.
- To apply histogram equalization, use **exposure** module, to apply normal histogram equalization to the original image, use **equalize_hist()** method and to apply adaptive equalization, use **equalize_adapthist()** method.
- To rotate the image use **rotate()** function under the **transform** module.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

it can help you perform several operations on images like rotating, [illegible]izing, cropping, grayscaling etc. Let's go through some of those operations

To carry out manipulation operations there is a module in this library called **Image.**

- To load an image use the **open()** method.
- To display an image use **show()** method.
- To know the file format use **format** attribute
- To know the size of the image use **size** attribute
- To know about the pixel format use **mode** attribute.
- To save the image file after desired processing, use **save()** method. Pillow saves the image file in *png* format.
- To resize the image use **resize()** method that takes two arguments as width and height.
- To crop the image, use **crop()** method that takes one argument as a box tuple that defines position and size of the cropped region.
- To rotate the image use **rotate()** method that takes one argument as an integer or float number representing the degree of rotation.
- To flip the image use **transform()** method that take one argument among the following: Image.FLIP_LEFT_RIGHT, Image.FLIP_TOP_BOTTOM, Image.ROTATE_90, Image.ROTATE_180, Image.ROTATE_270.

READ ALSO

Essential Pil (Pillow) Image Tutorial (for Machine Learning People)

4. NumPy

With this library you can also perform simple image techniques, such as flipping images, extracting features, and analyzing them.

Images can be represented by numpy multi-dimensional arrays and so their type is **NdArrays**. A color image is a numpy array with 3 dimensions. By slicing the multi-dimensional array the RGB channels can be separated.

Below are some of the operations that can be performed using NumPy on the image (image is loaded in a variable named **test_img** using imread).

- To flip the image in a vertical direction, use **np.flipud(test_img).**
- To flip the image in a horizontal direction, use **np.fliplr(test_img).**
- To reverse the image, use **test_img[::-1]** (the image after storing it as the numpy array is named as <img_name>).
- To add filter to the image you can do this:

Example: **np.where(test_img > 150, 255, 0)**, this says that in this picture if you find anything with 150, then replace it with 255, else 0.

- You can also display the RGB channels separately. It can be done using this code snippet:

To obtain a red channel, do **test_img[:,:,0]**, to obtain a green channel, do **test_img[:,:,1]** and to obtain a blue channel, do **test_img[:,:,2].**

5. Mahotas

It is a computer vision and image processing library and has more than 100 functions. Many of its algorithms are implemented in C++. Mahotas is an independent module in itself i.e. it has minimal dependencies.

Currently, it depends only on C++ compilers for numerical computations, there is no need for NumPy module, the compiler does all its work.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

- Convolution. (https://mahotas.readthedocs.io/en/latest/api.html)
- Spline interpolation (https://mahotas.readthedocs.io/en/latest/api.html)
- SLIC superpixels. (https://www.pyimagesearch.com/2014/07/28/a-slic-superpixel-tutorial-using-python/)

Let's look at some of the operations that could be done using Mahotas:

- To read an image use **imread()** method.
- To calculate the mean of the image use the **mean()** method.
- Eccentricity of an image measures the shortest length of the paths from a given vertex v to reach any other vertex w of a connected graph. To find the eccentricity of an image, use the **eccentricity()** method under the **features** module.
- For dilation and erosion on the image use, **dilate()** and **erode()** method under **morph** module.
- To find the local maxima of the image use **locmax()** method.

Summary

In this article, I briefly explained about classical image processing that can be done using **Morphological filtering, Gaussian filter, Fourier transform and Wavelet transform**.

All these can be performed using various image processing libraries like **OpenCV, Mahotas, PIL, scikit-learn.**

I also discussed popular neural networks like **CNN and GANs** that are used for computer vision.

Deep learning is changing the world with its broadway terminologies and advances in the field of image processing. Researchers are coming up with better techniques to fine tune the whole image processing field, so the learning does not stop here. Keep advancing.

Neetika Khandelwal

A self-motivated and enthusiastic computer science engineer with the following skills: C++, Python, Machine Learning, NLP, and solving real-world problems. I am a quick learner and passionate about learning new technologies. I have experience in developing technical articles in fields like Data Science, C++, Python.

Follow me on

READ NEXT

ML Experiment Tracking: What It Is, Why It Matters, and How to Implement It

10 mins read | Author Jakub Czakon | Updated July 14th, 2021

Let me share a story that I've heard too many times.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

some model parameters and dataset versions…

…after a few weeks, we weren't even sure what we have actually tried and we needed to re-run pretty much everything"

– unfortunate ML researcher.

And the truth is, when you develop ML models you will run a lot of experiments.

Those experiments may:

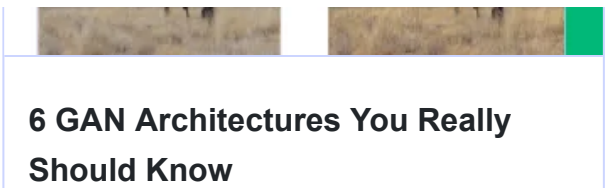
- use different models and model hyperparameters
- use different training or evaluation data,
- run different code (including this small change that you wanted to test quickly)
- run the same code in a different environment (not knowing which PyTorch or Tensorflow version was installed)

And as a result, they can produce completely different evaluation metrics.

Keeping track of all that information can very quickly become really hard. Especially if you want to organize and compare those experiments and feel confident that you know which setup produced the best result.

This is where ML experiment tracking comes in.

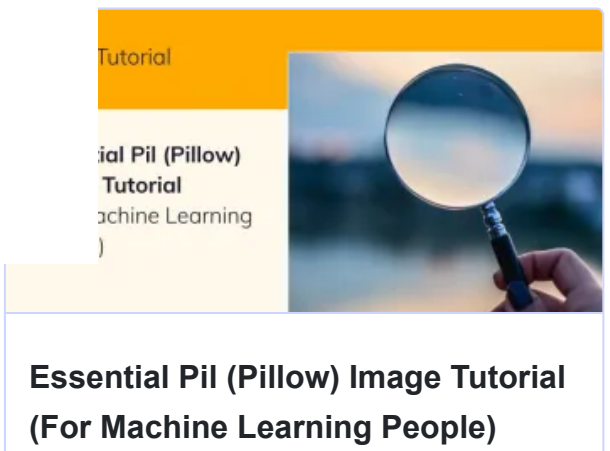
Continue reading ->



6 GAN Architectures You Really Should Know

by Shibsankar Das, April 2nd, 2020

Read more



Essential Pil (Pillow) Image Tutorial (For Machine Learning People)

by Derrick Mwiti, October 13th, 2020

Read more

Image Processing Techniques That You Can Use in Machine Learning Projects

by Gopal Singh Panwar, October 15th, 2020

Read more

Graph Neural Network and Some of

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!

neptuneblog

Search all articles

How to Use Neptune ML Experiment Tracking ML Model Management

Top MLOps articles from our blog in your inbox every month.

Type your email here Get Newsletter

GDPR compliant. Privacy policy.

Neptune is a metadata store for MLOps, built for research and production teams that run a lot of experiments.

Product

Overview

Experiment Tracking

Model Registry

Pricing

Roadmap

Service Status

Legal

Terms of service

Privacy policy

Resources

Neptune Blog

Neptune Docs

Neptune Integrations

ML Experiment Tracking

ML Model Management

MLOps

ML Project Management

Competitor Comparison

ML Experiment Tracking Tools

Best MLflow Alternatives

Best TensorBoard Alternatives

Best Kubeflow Alternatives

Other Alternatives

Company

About us

Careers

Copyright © 2021 Neptune Labs. All rights reserved.

Neptune.ai uses cookies to ensure you get the best experience on this website. By continuing you agree to our use of cookies. Learn more

Got it!