

[Open in app](#)

Medium

 Write

The U-Net : A Complete Guide

Alejandro Ito Aramendia · [Follow](#)

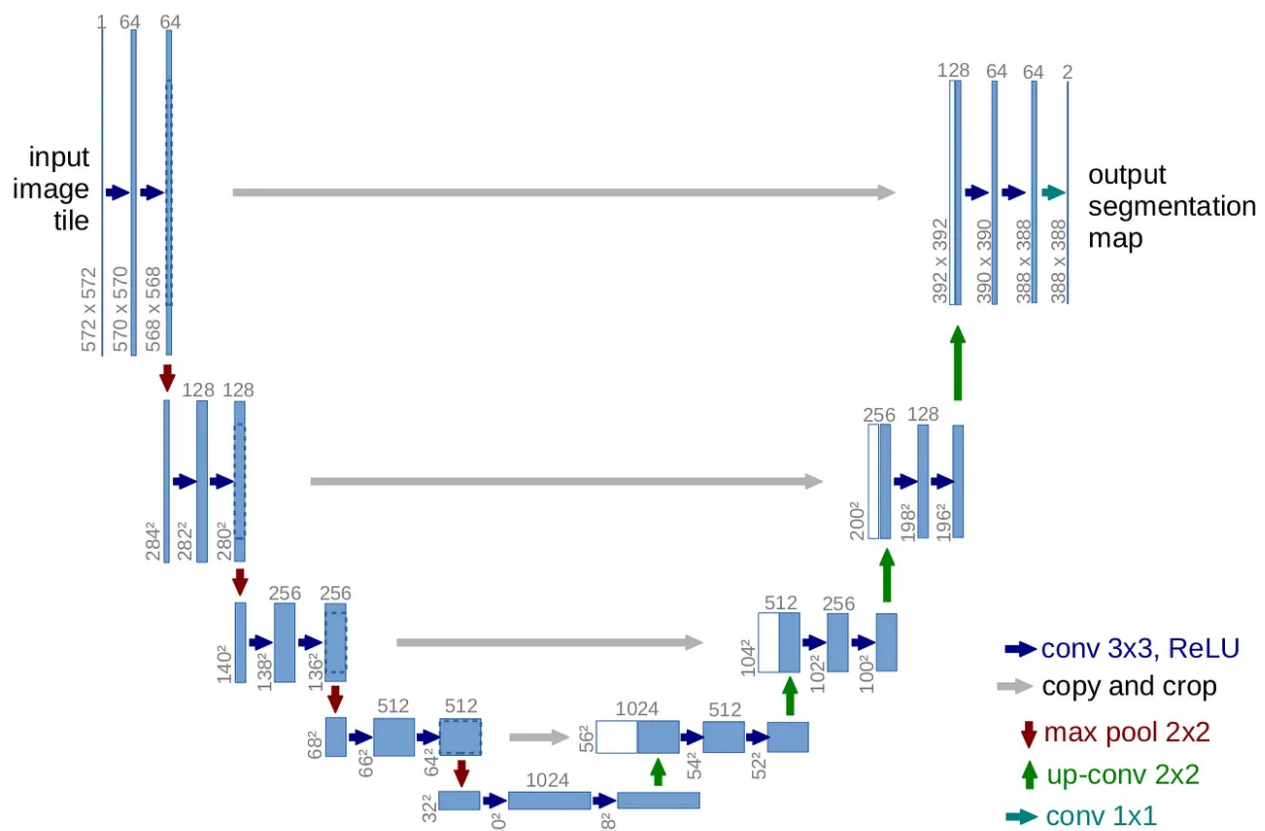
7 min read · Feb 1, 2024



284



2



The U-Net architecture.

Table of Contents

- [Introduction](#)
- [Contracting Path](#)
- [Expanding Path](#)
- [Up-Convolution and Channels](#)
- [Image Example](#)
- [Summary](#)

Introduction

The creation of the U-Net was a ground breaking discovery in the realm of **image segmentation**, a field focused on locating objects and boundaries within an image. This novel architecture proved to carry immense value in the analysis of biomedical images.

The U-Net is a special type of Convolutional Neural Network (CNN) and as a result, it is highly recommend to be familiar with them before delving into this article. If necessary please learn about CNNs [here](#).

The U-Net is composed of two main components: a **contracting path** and an **expanding path**.

- **Contracting path:** aims to decrease the spatial dimensions of the image, while also capturing relevant information about the image.
- **Expanding path:** aims to upsample the feature map and produce a relevant segmentation map using the patterns learnt in the contracting path.

As you may notice, the U-Net in fact resembles an **encoded-decoder** architecture, which coincidentally makes a **U shape**, hence the name.

Let's now get into more depth about each component.

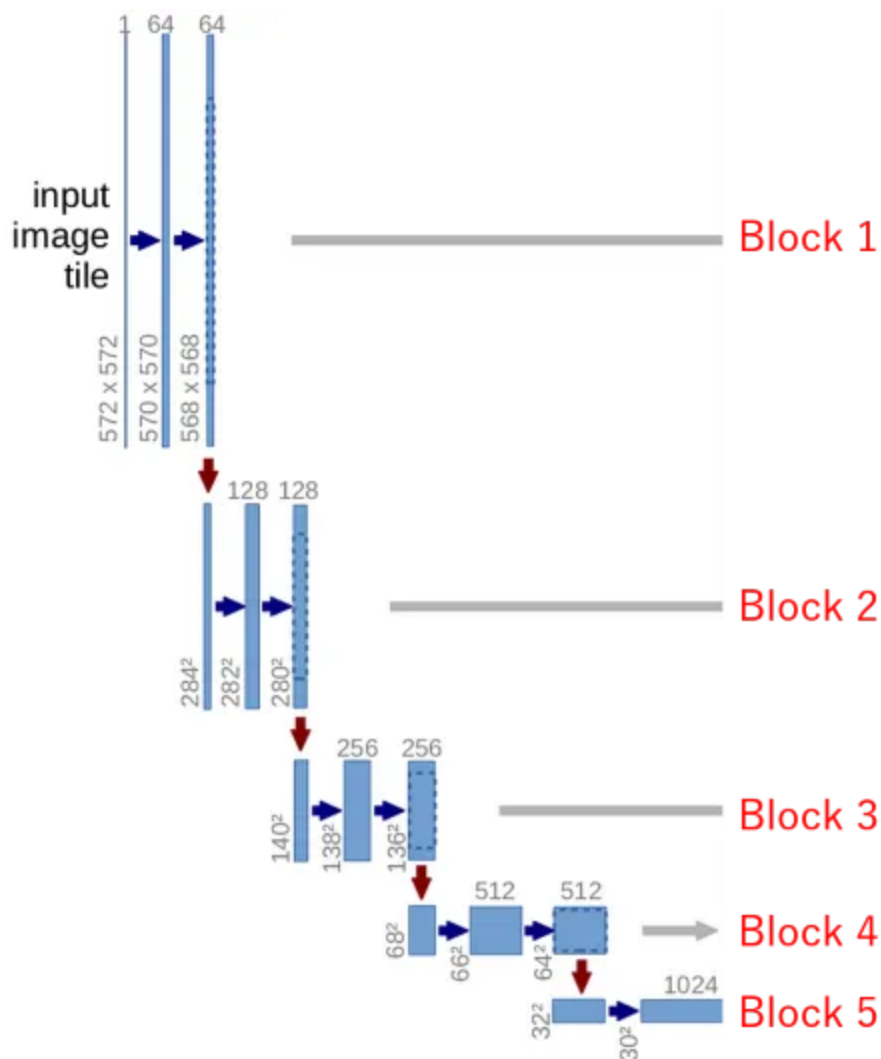
Note: While reading further on, you may wonder, *“How on earth is it possible to change the number of channels and what on earth is an up-convolution?!”*

Well, don't worry, I have covered this at the end. And if you already know this, then feel free to skip that section.

Contracting Path

The contracting path uses a combination of **convolution** and **pooling** layers to extract and capture features within an image, while at the same time, reducing its spatial dimensions.

Let's now explore each of the **5 blocks** in the **contracting path** down below.



The contracting path of the U-Net.

Block 1

1. An input image with dimensions 572^2 is fed into the U-Net. This input image consists of only **1 channel**, likely a **grayscale** channel.
2. Two **3×3 convolution** layers (unpadded) are then applied to the input image, each followed by a **ReLU** layer. At the same time the number of **channels** are increased to **64** in order to capture higher level features.
3. A **2×2 max pooling** layer with a **stride of 2** is then applied. This downsamples the feature map to **half** its size, 284^2 .

Block 2

1. Just like in block 1, two **3x3 convolution** layers (unpadded) are applied to the **output of block 1**, each followed again by a **ReLU** layer. At each new block the number of feature **channels** are **doubled**, now to **128**.
2. Next a **2x2 max pooling** layer is again applied to the resulting feature map reducing the spatial dimensions by **half** to **140²**.

Block 3

The procedure used in **block 1** and **2** is the **same** as in **block 3**, so will not be repeated.

Block 4

Same as block 3.

Block 5

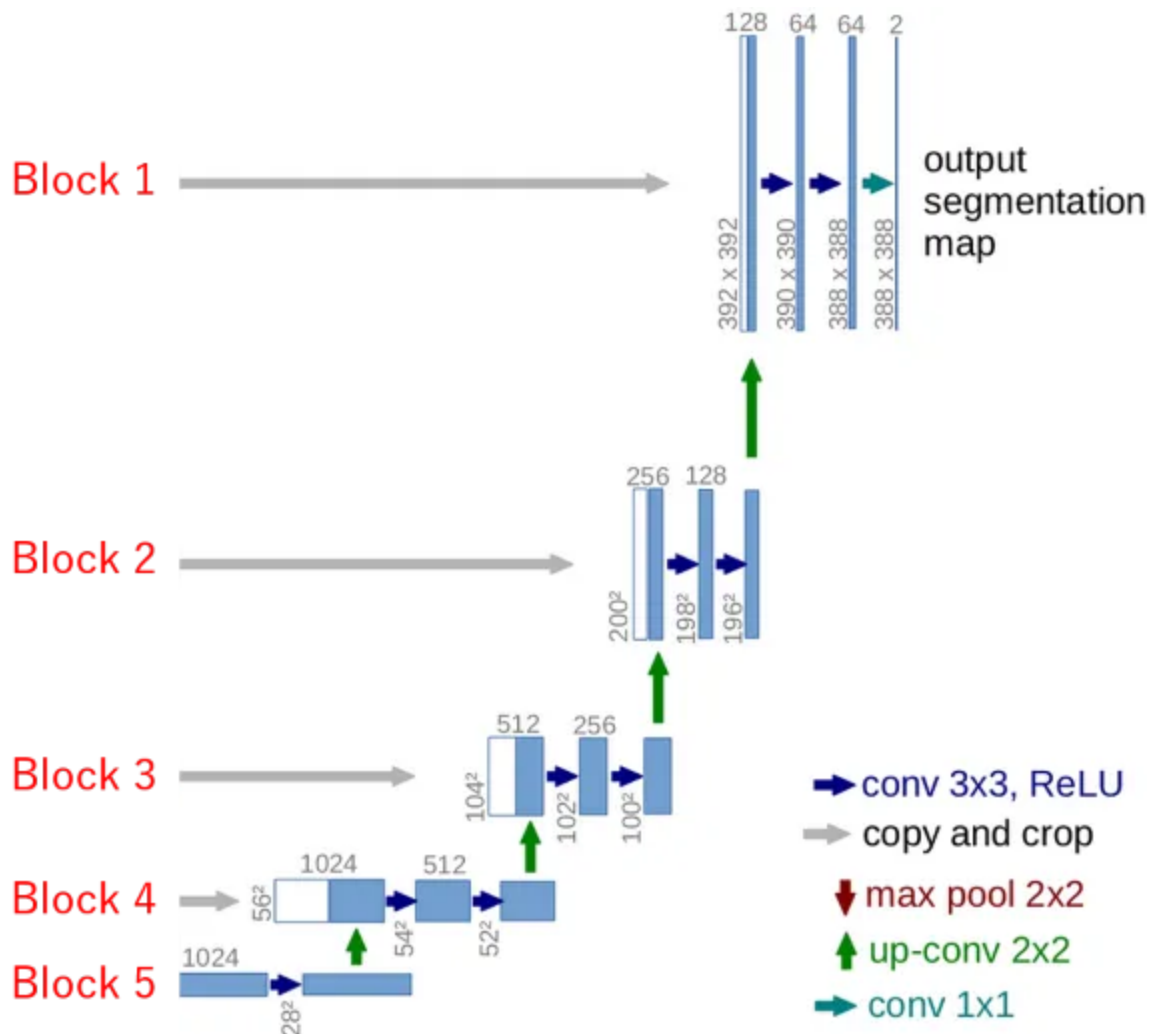
1. In the **final block** of the contracting path, the number of feature **channels** reach **1024** after being **doubled** at each block.
2. This block also contains two **3x3 convolution** layers (unpadded), which are each followed by a **ReLU** layer. However, for symmetry purposes, I have only included **one** layer and included the second layer in the expanding path.

After complex features and patterns have been extracted, the feature map moves on to the expanding path.

Expanding Path

The expanding path uses both **convolution** and **up-convolution** operations to combine learnt features and upsample the input feature map until it generates a segmentation map.

Much like with the contracting path, each block will be discussed below.



The expanding path of the U-Net.

Before we read further: Skip connections are used to send images directly from the contracting path to the expanding path without them having to go through all the blocks. This allows for both high and low level features to be preserved and learnt, reducing any information loss that occurs during the contracting path.

Block 5

1. Continuing on from the contracting path, a second 3x3 convolution (unpadded) is applied with a ReLU layer after it.

2. Then a **2x2 convolution** (up-convolution) layer is applied, upsampling the spatial dimensions **twofold** and also **halving** the number of channels to **512**.

Block 4

1. Using **skip connections**, the corresponding feature map from the contracting path is then **concatenated**, doubling the feature **channels** to **1024**. Note that this concatenation must be **cropped** to match the expanding path's dimensions.
2. Two **3x3 convolution** layers (unpadded) are applied, each with a **ReLU** layer following, reducing the **channels** to **512**.
3. After, a **2x2 convolution** (up-convolution) layer is applied, upsampling the spatial dimensions **twofold** and also **halving** the number of channels to **256**.

Block 3

The procedure used in **block 5** and **4** is the **same** as in **block 3**, so will not be repeated.

Block 2

Same as block 3.

Block 1

1. In the **final block** of the expanding path, there are **128 channels** after **concatenating** the skip connection.
2. Next, two **3x3 convolution** layers (unpadded) are applied on the feature map, with **ReLU** layers inbetween reducing the number of feature **channels** to **64**.

3. Finally, a **1x1 convolution** layer, followed by an **activation** layer (sigmoid for binary classification) is used to reduce the number of **channels** to the desired number of classes. In this case, 2 classes, as **binary classification** is often used in medical imaging.

After upsampling the feature map in the expanding path, a segmentation map should be generated, with each pixel classified individually.

Up-Convolutions and Channels

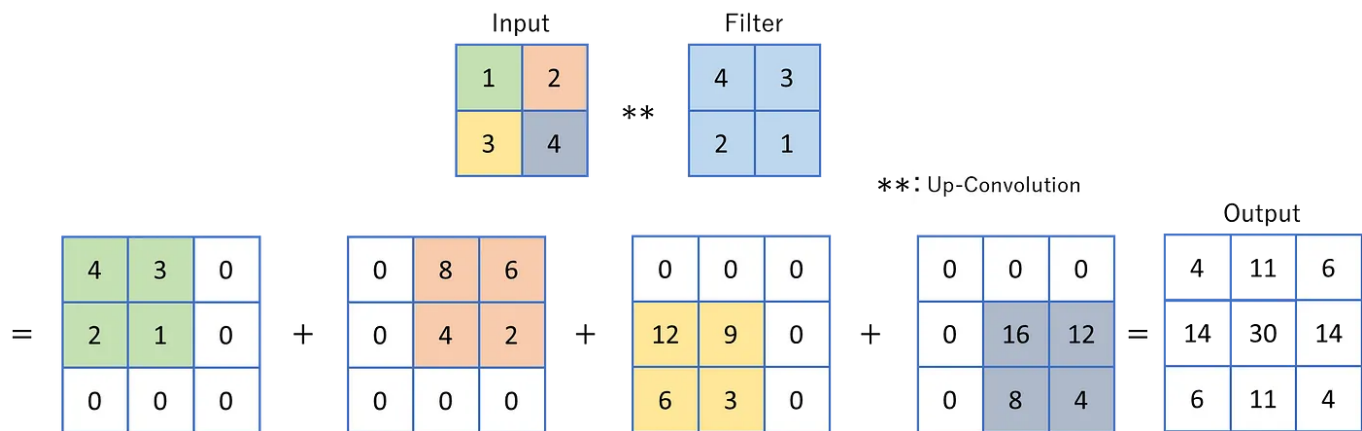
In this section I would like to discuss what up-convolutions are and how changing the number of feature channels is possible. **Convolutions**, **pooling**, **strides** and **padding** were discussed in my previous CNN article and therefore, I have chosen not to cover them again. If necessary, please recap these concepts [here](#).

Now let's get into it.

Up-Convolution

An up-convolution, also known as a deconvolution or transpose convolution, is a method used to upsample images and recover spatial information.

Let's look at the example below and briefly discuss what's happening.



An example of up-convolution with stride 1.

The best way to perform up-convolutions is to **expand** and **duplicate** each element **from the input** feature map to the **same size** as the filter. This process **up-samples** the input. The filter is then applied over each of these expanded regions.

For example, the expanded **green input** above is initially just composed of **four 1s**. Likewise, the expanded **red, yellow and grey regions** are initially filled with just **2s, 3s and 4s**, respectively. Next, the **filter** is applied over each of these regions and the results are summed to form the output feature map.

In the U-Net described above, the spatial dimensions were doubled, which means that a **2x2 filter** was used with a **stride of 2**.

Changing the Amount of Channels

Throughout the U-Net, the number of feature channels are constantly changing. How do convolution operations affect this?

Well, the convolutions itself do not directly affect the number of channels present. It is in fact, determined by the **number of filters** used in the convolution layer. If **64 filters** are applied over the input, with each

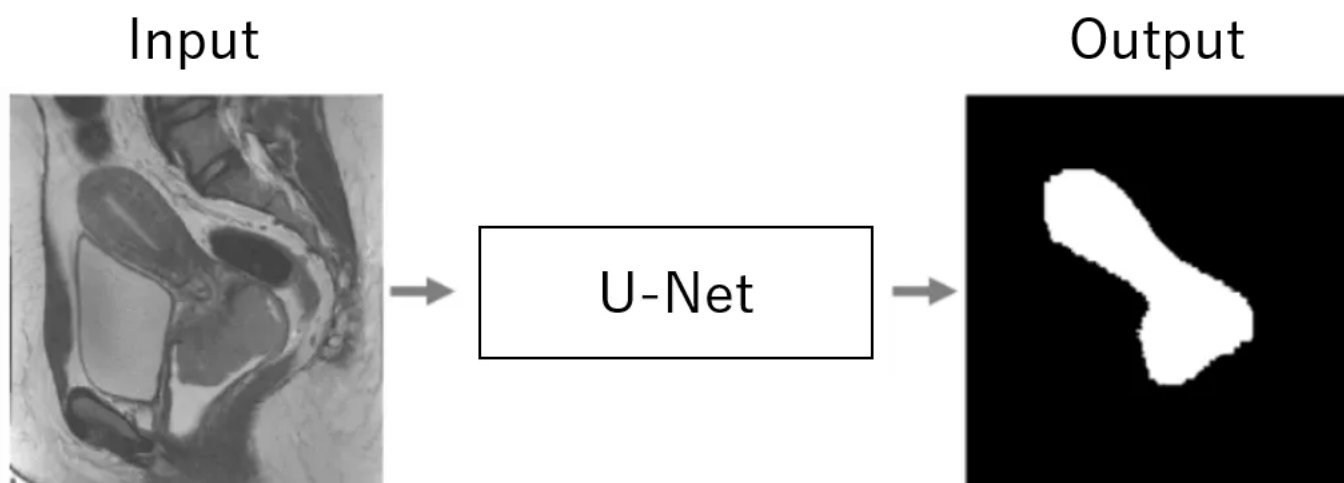
attempting to extract a different feature, **64 feature maps** will also be generated.

This may seem obvious to some, but was something that stumped me while learning this.

Image Example

U-Nets are often used in medical imaging. They play crucial roles in **detecting** and **locating** tumors, cysts and other abnormalities.

Below is a possible example of what an input and output of a U-Net may look like.



A possible input and output of a U-Net. In this case, a grayscale input and a single channel binary output.

A medical grayscale image of a uterus was used as an input and fed into a U-Net. After having being processed in the U-Net, each pixel was classified into one of two classes: **tumor** or **not-tumor**. This segmentation map can be seen in the output image.

Summary

To conclude this article, let's summarise what we have learnt.

The U-Net is an architecture that consists of **23 total layers**. Using a combination of **convolution**, **up-convolution**, **pooling** and **skip connections**, the U-Net is able to extract and capture complex features, while also keeping and reconstructing **spatial** information. This allows for the **localisation** of features in an image, thus producing accurate **segmentation** maps. This is especially useful in medical image analysis where accurately locating and detecting abnormalities is vital.

Thank you for getting this far, if you have any questions do not hesitate to ask.

References

[1] Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv:1505.04597.

[Machine Learning](#)[Data Science](#)[AI](#)[Medical](#)[Image Processing](#)

Written by **Alejandro Ito Aramendia**

Follow

172 Followers · 7 Following

Learning and writing.

Responses (2)



What are your thoughts?

Respond



Deepankar Singh he
2 months ago



Very nice article! One thing you should have added would be about how the U shape helps in changing the receptive field of the U-net model!



4

Reply



Sajid Hussain
about 2 months ago



The procedure used in block 1 and 2 is the same as in block 3, so will not be repeated.



Reply

More from Alejandro Ito Aramendia



 Alejandro Ito Aramendia

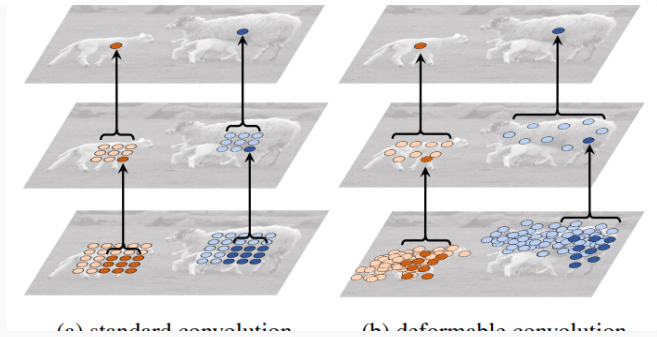
Attention Is All You Need : A Complete Guide to Transformers

Everything You Need to Know

Jan 1, 2024

 604

 7



 Alejandro Ito Aramendia

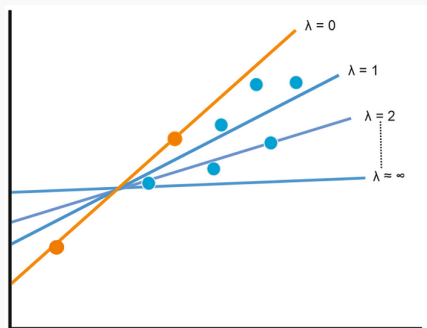
Deformable Convolutional Networks (DCNs) : A Complete...

Everything You Need to Know

Feb 1, 2024

 304

 3



 Alejandro Ito Aramendia

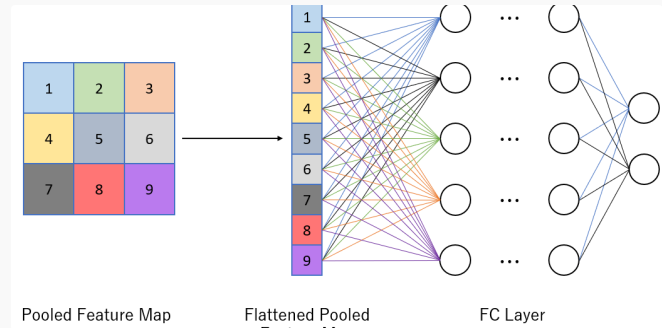
L1 and L2 Regularization (Part 1): A Complete Guide

Improving estimation methods to better model data

Apr 1, 2024

 7

 3



 Alejandro Ito Aramendia

Convolutional Neural Networks (CNNs) : A Complete Guide

Everything You Need to Know

Jan 1, 2024

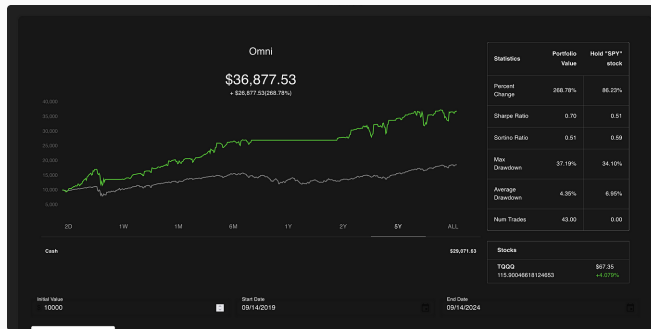
 208

 2



See all from Alejandro Ito Aramendia

Recommended from Medium

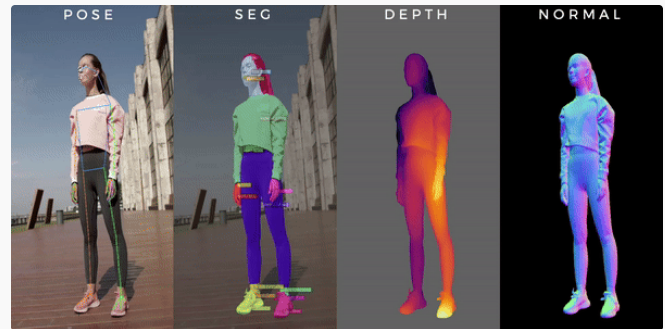


 In DataDrivenInvestor by Austin Starks

I used OpenAI's o1 model to develop a trading strategy. It is...

It literally took one try. I was shocked.

★ Sep 15, 2024 🖱 8.1K 💬 197 📌 ...



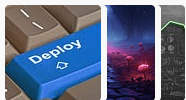
 AI Papers Academy

Sapiens by Meta AI: Foundation for Human Vision Models

In this post we dive into Sapiens, a new family of computer vision models by Meta AI that...

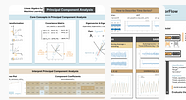
★ Aug 26, 2024 🖱 130 📌 ...

Lists



Predictive Modeling w/ Python

20 stories · 1749 saves



Practical Guides to Machine Learning

10 stories · 2121 saves



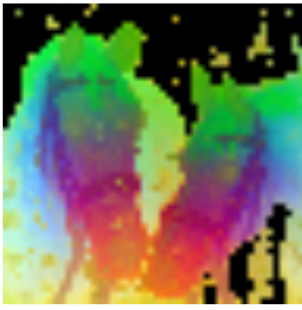
Natural Language Processing

1881 stories · 1515 saves



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 533 saves



Javier



DINOv2 Visualization

A guide to visualize DINOv2 embeddings



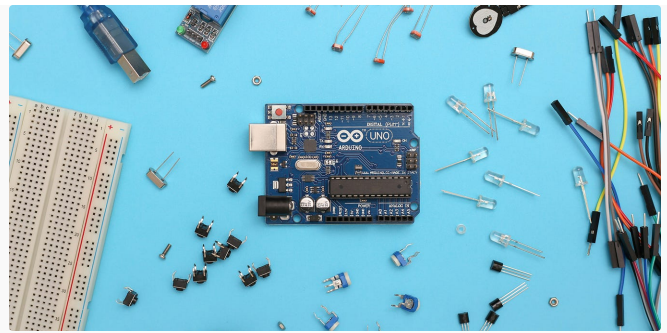
Jul 10, 2024



97



1



In Towards Data Science by Skylar Jean Callis

Vision Transformers, Explained

A Full Walk-Through of Vision Transformers in PyTorch

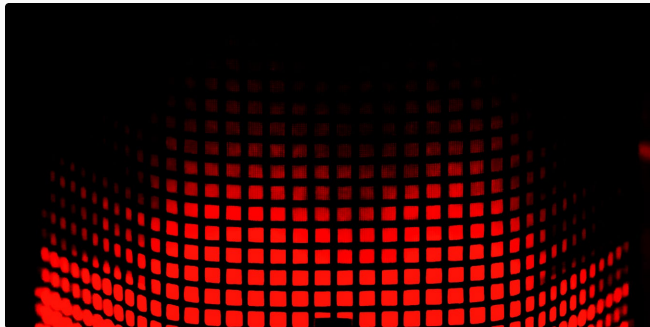
Feb 27, 2024



1.2K



11



In Cubed by Manish Gupta

Understanding Diffusion Models

An Illustrated Dive into Their Process and Design



Nov 30, 2024



4



1



In The Deep Hub by Jorgecardete

Convolutional Neural Networks: A Comprehensive Guide

Exploring the power of CNNs in image analysis

Feb 7, 2024



2.7K



38

[See more recommendations](#)