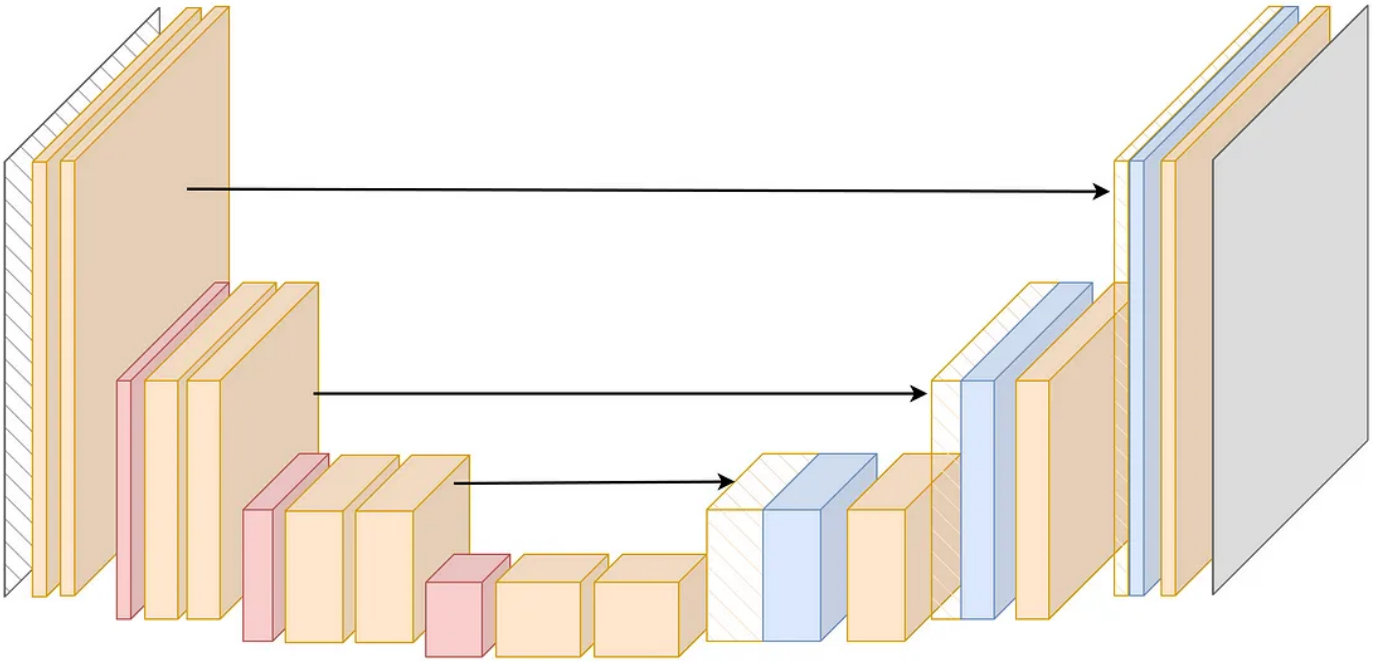
✨ Member-only story

U-Net Explained: Understanding its Image Segmentation Architecture

How skip connections allow CNNs to perform accurate semantic segmentation with less data

Conor O'Sullivan · Follow
Published in Towards Data Science
7 min read · Mar 8

Listen Share ••• More



(source: author)

U-Net is a popular deep-learning architecture for semantic segmentation. Originally developed for medical images, it had great success in this field. But, that was only the beginning! From satellite images to handwritten characters, the architecture has

improved performance on a range of data types. Yet, other CNN architectures can also do segmentation, so what makes U-Net so special?

To answer this, we will explore the U-Net architecture. We will compare it to CNNs used for classification and autoencoders. By doing so, we will understand how the **skip connections** are the key to U-Net's success. We will see how they allow the architecture to perform accurate segmentations with less data.

What is semantic segmentation?

We'll start by understanding what U-Net was developed for. **Image segmentation** or **semantic segmentation** is the task of assigning a class to each pixel in an image. Models are trained using segmentation maps as target variables. For example, see Figure 1. We have the original image and a binary segmentation map. The map separates the image into cell and non-cell pixels.

This biomedical image segmentation task is what U-Net was originally developed for. The defining factor of these datasets is the small number of training images. The example in Figure 1 comes from a dataset of only 35 images. With the help of image augmentation, UNet provided an 11% improvement in accuracy over the second-best approach.

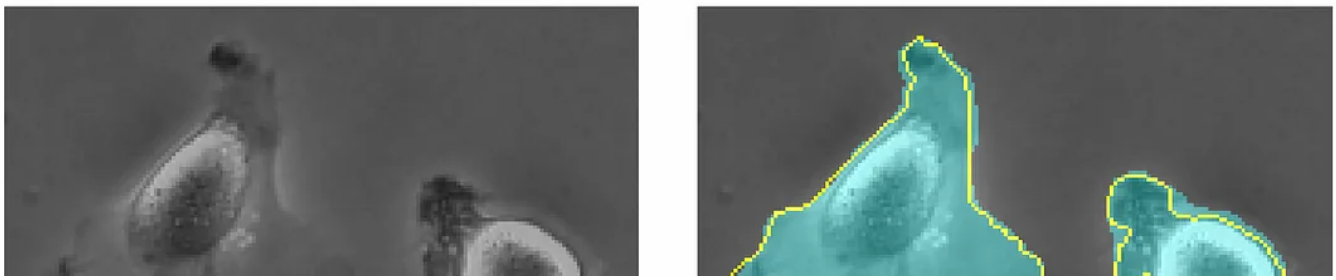


Figure 1: segmenting cells in medical images (source: O. Ronneberger, et. al.)

U-Net is also flexible. I have applied it in my own research to segment satellite images. As seen in Figure 2 we segment images of coastlines into 2 classes — land and water. The task is similar but the input is different to medical images. We have gone from a single greyscale image to using 12 spectral bands available in satellite images.

Figure 2: coastline water body segmentation (source: author) (dataset: SWED) (licence: Sentinel Data Legal Notice)

U-Net Architecture

So U-Net can achieve good results for a variety of segmentation tasks. To explain why we will look at the most important components of the architecture — the encoder, decoder and skip connections. We will see how these fit together to *extract* and *localize* features in images.

Encoder

For semantic segmentation, we care about *what* objects are in the image and *where* in the image they are. This is compared to **object detection** or **image classification.** Here we aim to predict one class for each image**.** That is we only care *if* an object is in an image. To make these predictions we can use an encoder.

You will find a version of an encoder in all CNN architectures. Its job is to create a compact representation of the input image. This is a lower-dimensional representation that contains only the most important information in the image. In other words, the encoder is used to *extract* features.

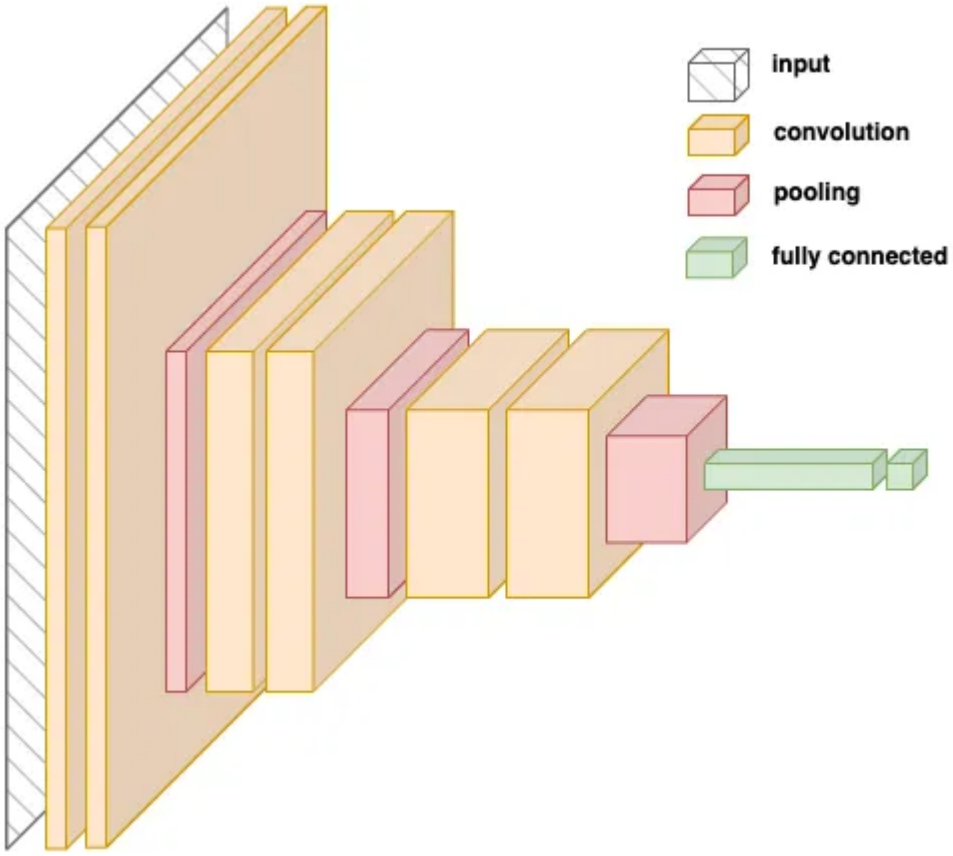


Figure 3: encoder used for image classification (source: author)

This is done using the convolutional layers and pooling layers. The convolution layer is a mapping or kernel that passes over each pixel in an image. This mapping is *learned* through the process of training the model. Then, using a *predefined* function, the pooling layers reduce the dimensionality of the output.

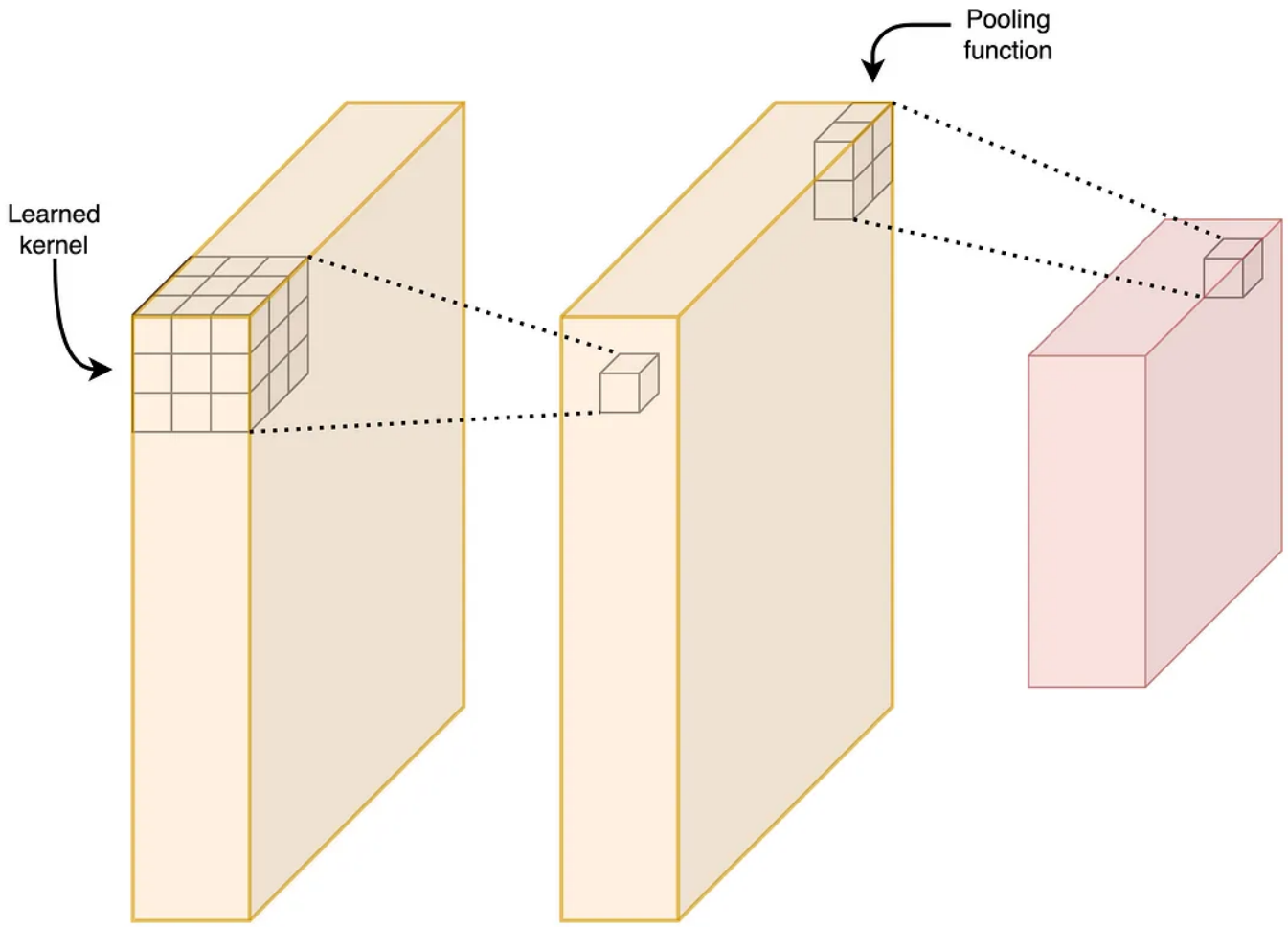


Figure 4: convolutional and pooling layers (source: author)

By combining multiple convolutional and pooling layers, we can extract more detailed information. We go from low-level details like edges and colours to high-level features like ears, teeth and eyes. The network will learn what features are important for classification and extract these to create a compact representation of an image.

One problem is this compact representation does not include the *location* of the features in an image. This is fine for image classification. To classify a dog, we only need to know *if* a tail, ear or fur is in the image. It does not matter *where* in the image these features come from. In comparison, for segmentation location is important.

Decoder

Another problem with the encoder is its output has a low dimension. If used for classification the final layer will have a few nodes — one for every class. For segmentation, our output will be an image with the same height and width as the input.

We need a decoder. As seen in Figure 5, this is the section that starts after the conv4 block. The decoder will reconstruct an image from the compact representation. Like the encoder, it has convolutional blocks. Expect now we have *deconvolution* layers that *increase* the dimensionality of the image.

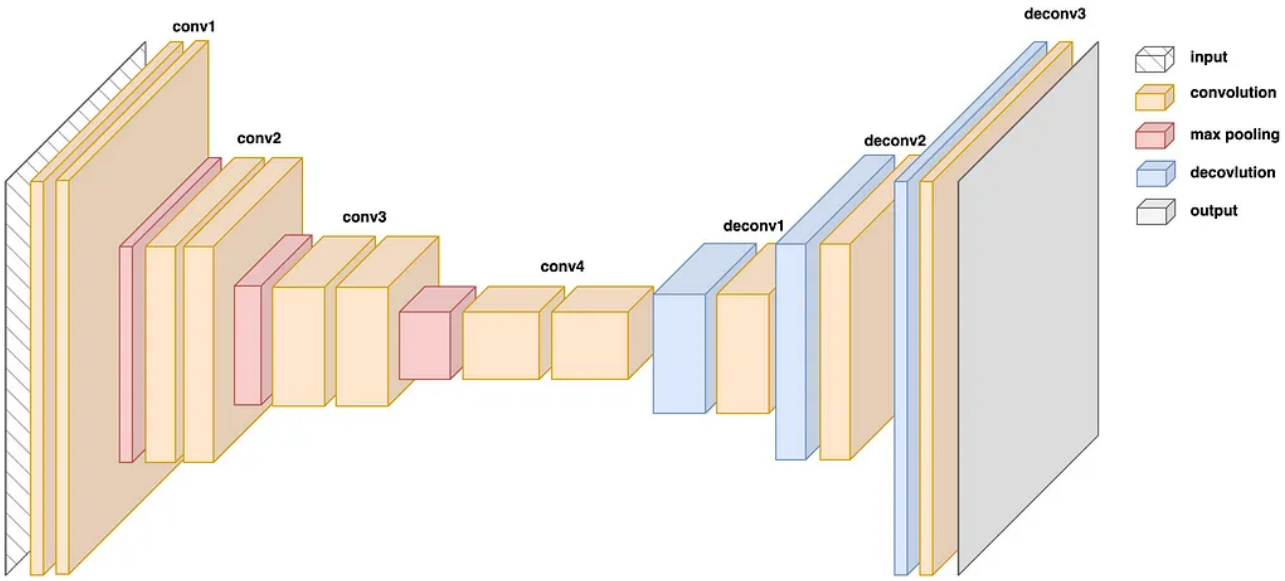


Figure 5: autoencoder architecture (source: author)

As mentioned, pooling layers will use a predefined method to *reduce* dimensionality. Such as max pooling, which takes the maximum pixel value in the cell window. In comparison, upsampled or deconvolution layers *increase* dimensionality using a function that is learned. That is the upsampling function is updated as the model is trained.

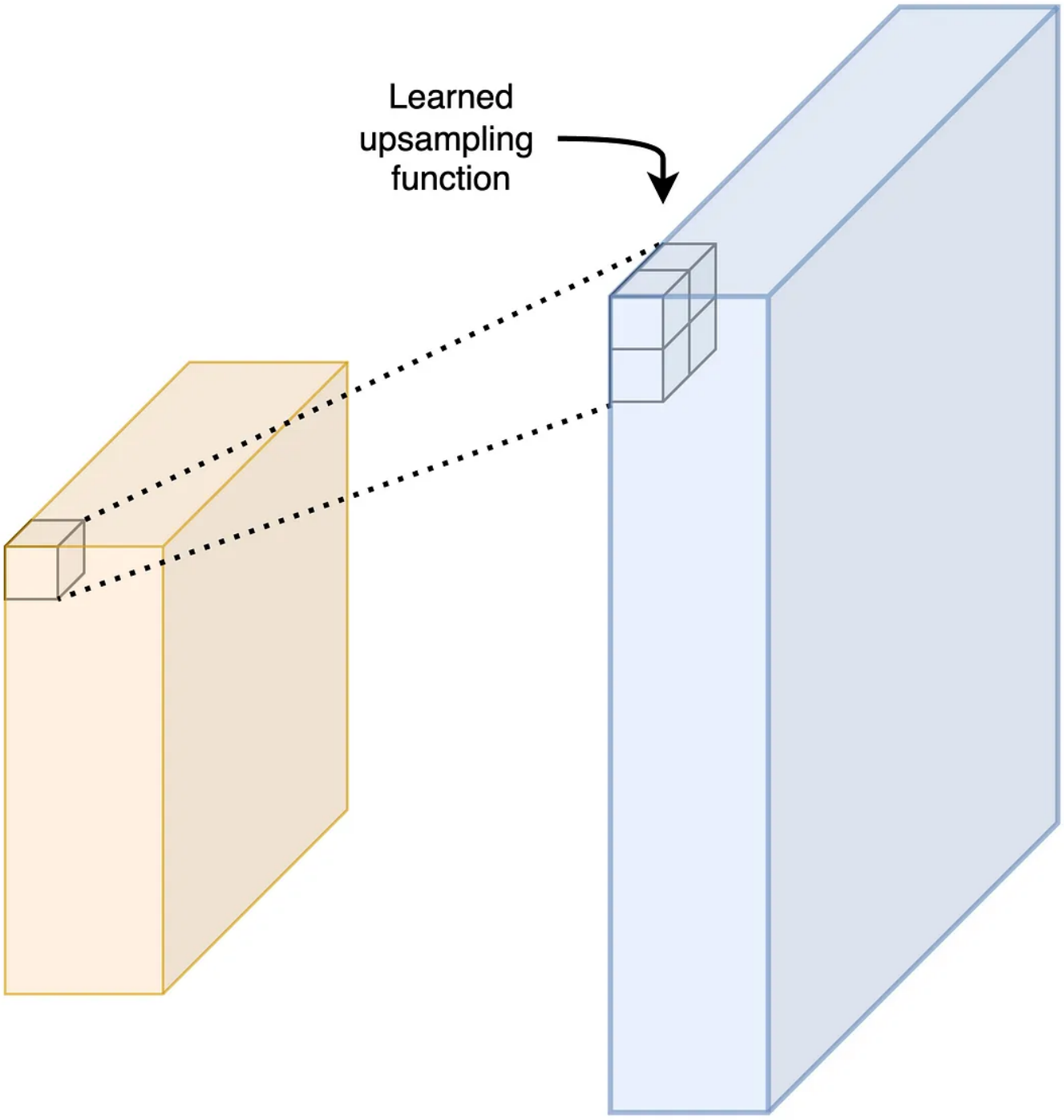


Figure 6: deconvolution using a learned upsampling function (source: author)

In an autoencoder, the input and output images will be the same. Here the goal of the decoder is to reconstruct the input as accurately as possible. We can then take the parameters from one of the lower dimensional layers (i.e conv4) as a compressed image. We can save or send the compressed image. The decoder can then be used to reconstruct the original input.

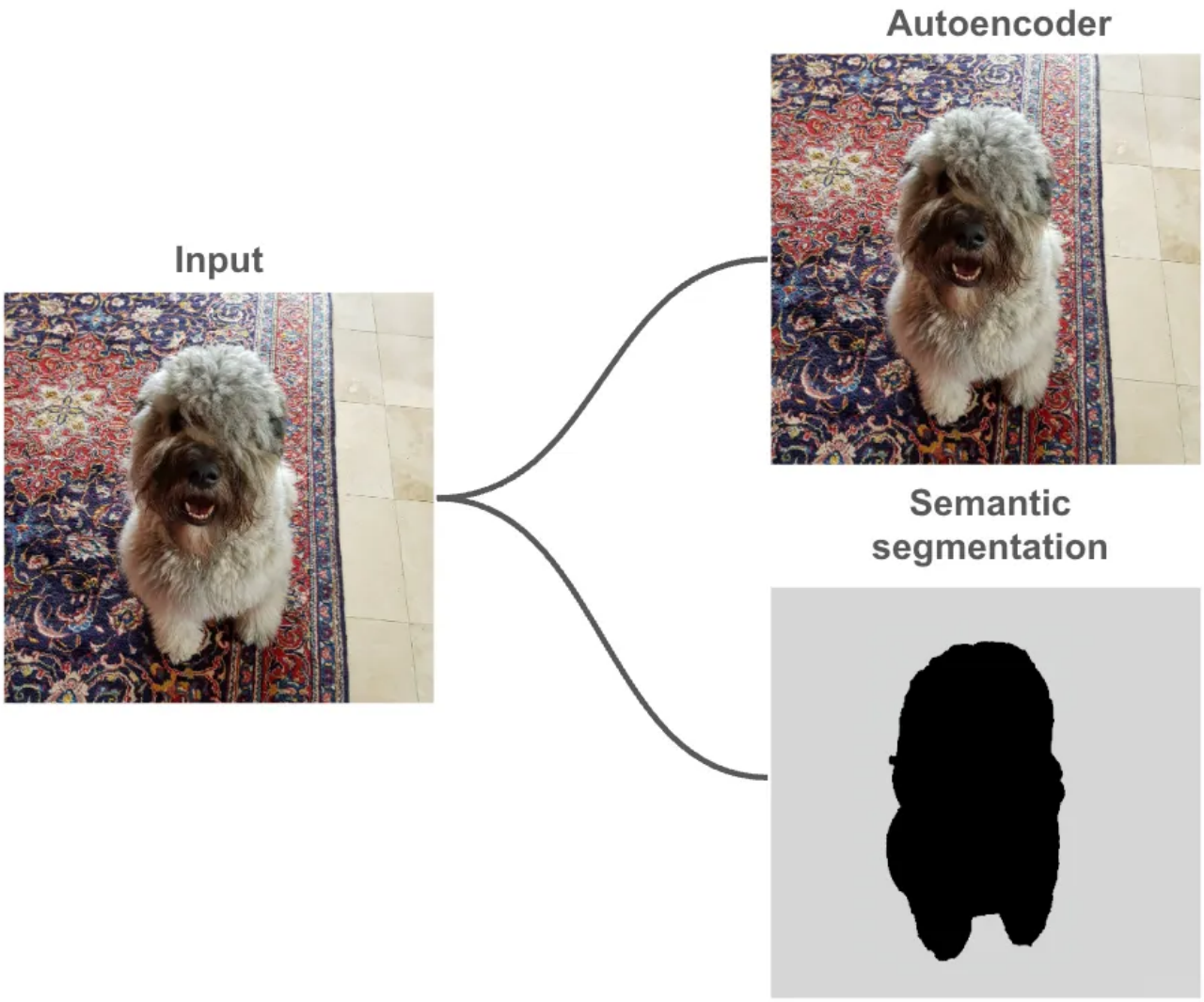


Figure 7: the output of an autoencoder vs semantic segmentation model (featuring my dog, Guinness) (source: author)

At this point, you may be asking if the encoder and decoder are enough. This architecture can learn a mapping from image to image. So surely, it can learn a mapping to the simpler output required for segmentation.

The decoder is able to pass the important features to the encoder. The problem is the location of the features is still lost. To get around this, we need an *immense* amount of data to train an autoencoder. This is the only way the decoder can learn to accurately reconstruct the images from compressed representations. With skip connections, we can reduce this data requirement.

Skip connections

The important point is that for autoencoders the encoder and decoder *must* be separate. Otherwise, it defeats the entire point of image compression. For semantic

segmentation, we do not have this restriction.

In a U-Net, skip connections are used to pass information from earlier convolutional layers to the deconvolution layers. Critically, what is passed is the *location* of the feature extracted by convolutional layers. That is the skip connections tell the network *where* in the image the features come from.

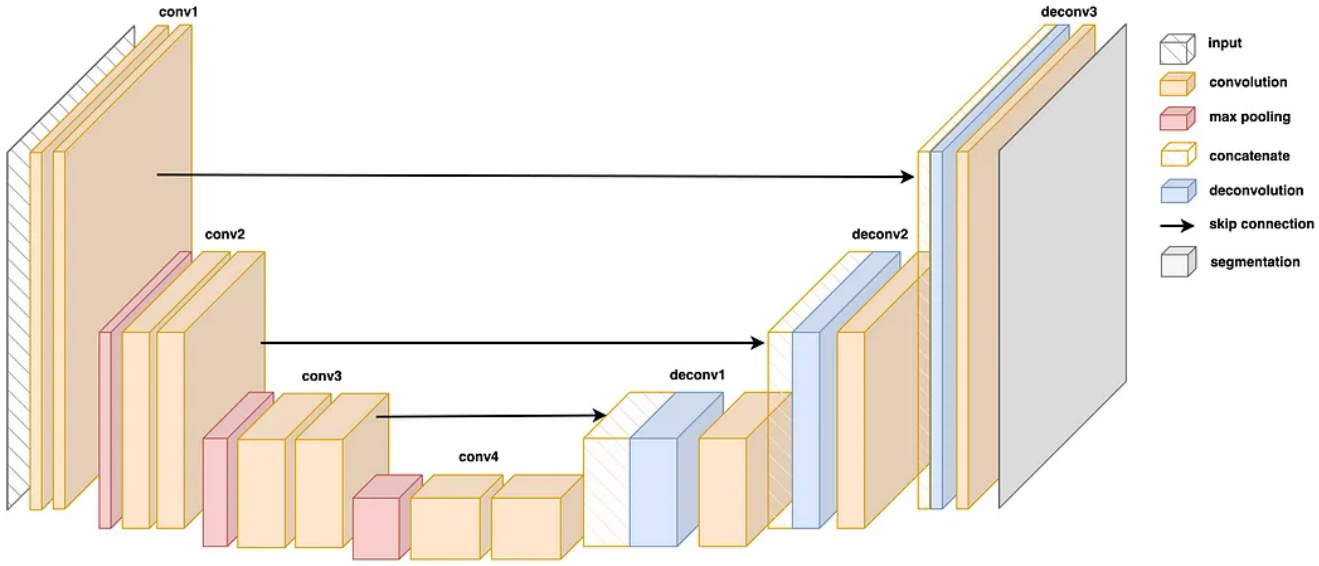


Figure 8: U-Net architecture (source: author)

This is done by concatenating the last layer in the convolutional block and the first layer of the opposite deconvolutional block. The U-Net is symmetrical — the dimensions of the opposite layers will be the same. As seen in Figure 9, this makes it easy to combine the layers into a single tensor. Convolution is then done as usual by running the kernel over the single concatenated tensor.

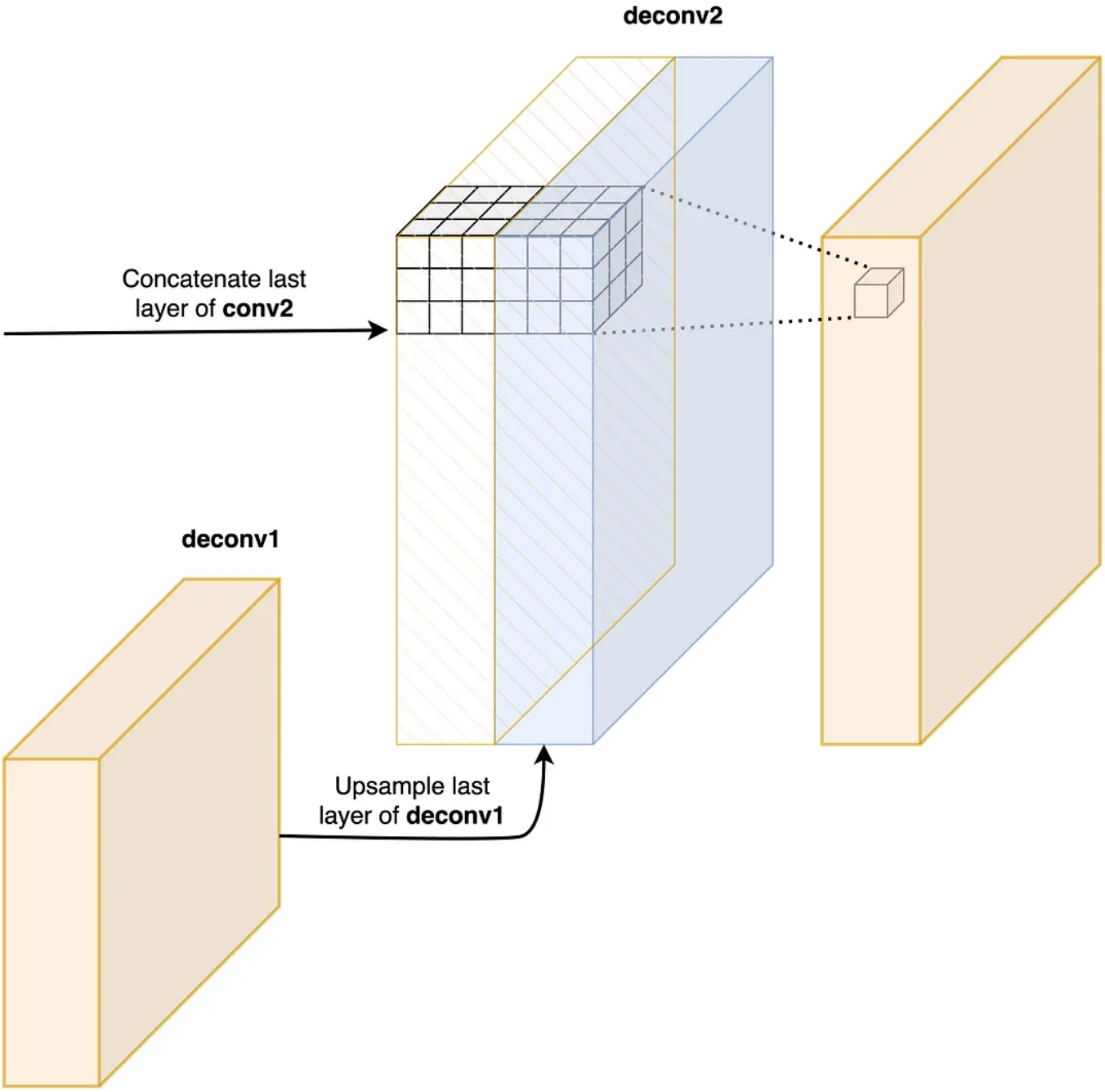


Figure 9: concatenating layers (source: author)

This concatenation is at the heart of the U-Net. It combines two important pieces of information:

- **Feature extraction** — features are passed from the previous layer to the upsampled layer (blue)
- **Feature localization** — the location of the feature is passed from the opposite convolution layer (shaded orange)

By combining this information, we can improve the performance of semantic models and reduce the amount of data required to train the network.

We've glossed over a few details such as the activation functions, the number of layers and the dimensionality of the layers. These can all be taken as hyperparameters in the U-Net. To tackle specific segmentation problems, adaptions have been made to the original architecture. The key to the success of all of these is the skip connections.

I hope you enjoyed this article! You can support me by becoming one of my **referred members :)**

| Twitter | YouTube | Newsletter — sign up for **FREE** access to a Python SHAP course

References

Olaf Ronneberger, Philipp Fischer, Thomas Brox, **U-Net: Convolutional Networks for Biomedical Image Segmentation** (2015)
https://arxiv.org/abs/1505.04597

Bharath K, **U-Net Architecture For Image Segmentation** (2021),
https://blog.paperspace.com/unet-architecture-image-segmentation/

Jeremy Zhang, **UNet — Line by Line Explanation** (2019),
https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5

Heet Sankesara, **UNet — Introducing Symmetry in Segmentation**
https://towardsdatascience.com/u-net-b229b32b4a71

Machine Learning Data Science Image Segmentation Convolutional Network

Getting Started

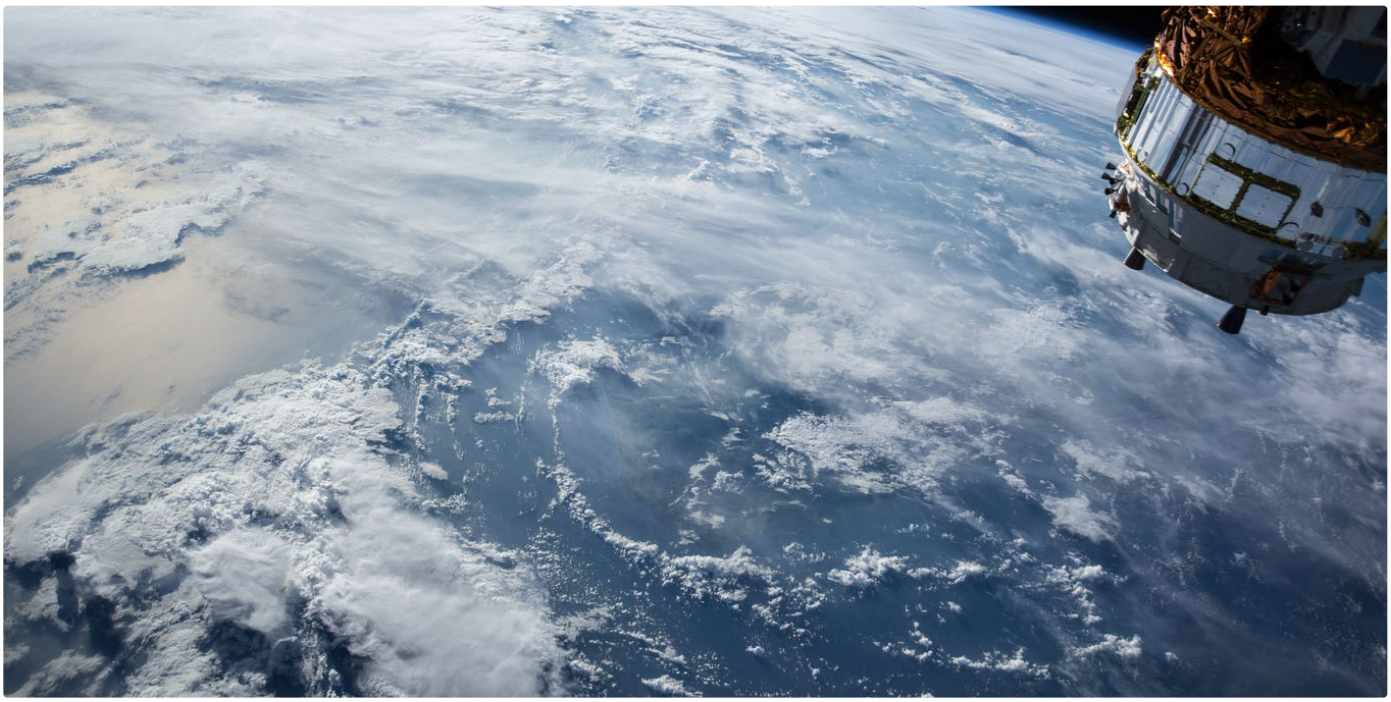
Follow

Written by Conor O'Sullivan

3.4K Followers · Writer for Towards Data Science

PhD Student | Writer | Houseplant Addict | Follow me for articles on IML, XAI, Algorithm Fairness and Remote Sensing | New article every 2 weeks!

More from Conor O'Sullivan and Towards Data Science



Conor O'Sullivan in Towards Data Science

Downloading Landsat Satellite Images with Python

Streamline Landsat scene downloads with the landsatxplore Python package

✨ · 6 min read · May 9

89

Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

2.9K 39

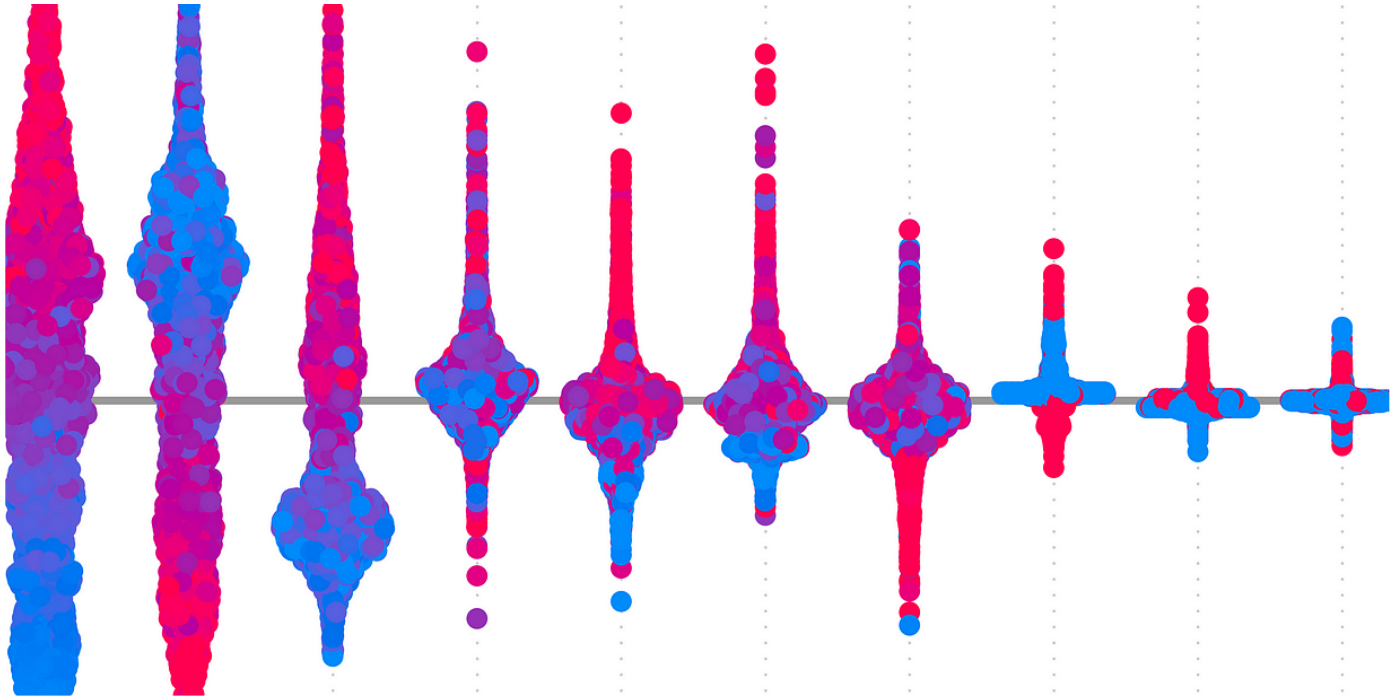
 Leonie Monigatti in Towards Data Science

Getting Started with LangChain: A Beginner's Guide to Building LLM-Powered Applications

A LangChain tutorial to build anything with large language models in Python

✦ · 12 min read · Apr 25

 2K  16 



 Conor O'Sullivan in Towards Data Science

Introduction to SHAP with Python

How to create and interpret SHAP plots: waterfall, force, mean SHAP, beeswarm and dependence

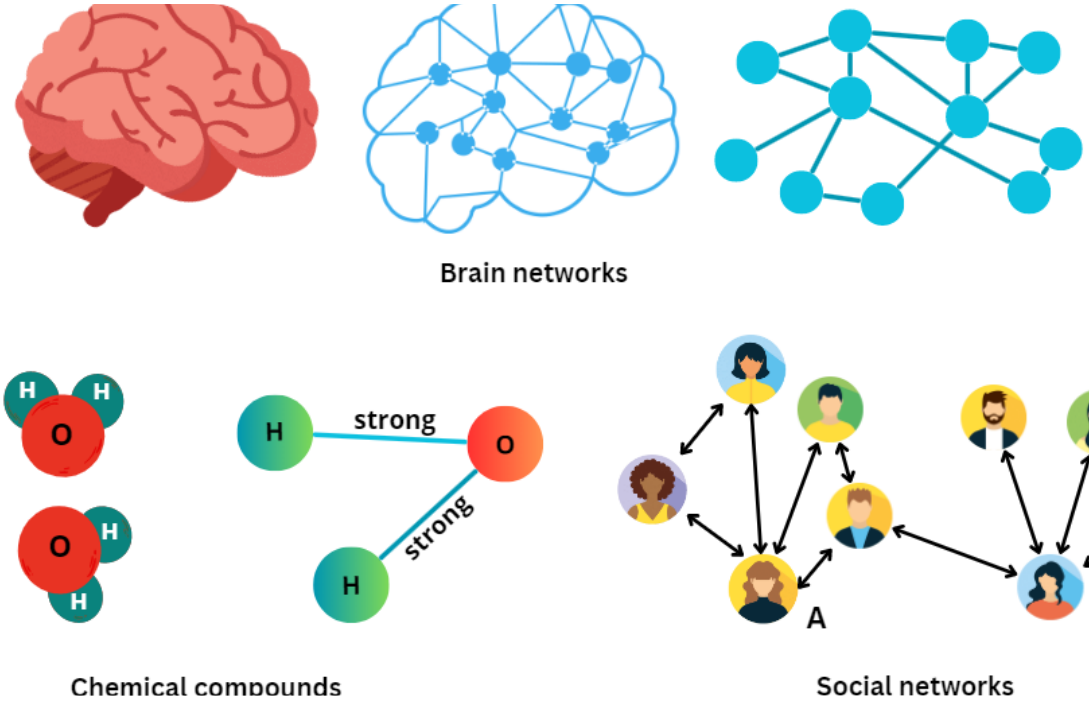
✨ · 11 min read · Dec 19, 2021

699 8

See all from Conor O'Sullivan

See all from Towards Data Science

Recommended from Medium



Bscarleth Gtz

Introduction to Graph Neural Networks: An Illustrated Guide

Hi Everyone! This post starts with the basics of graphs and moves forward until covering the General Framework of Graph neural networks...

14 min read · May 17

30

Edgardo Solano Carrillo in Towards Data Science

Boosting image generation by intersecting GANs with Diffusion models

An efficient recipe for stable image-to-image translation

8 min read · May 9

Lists

What is ChatGPT?

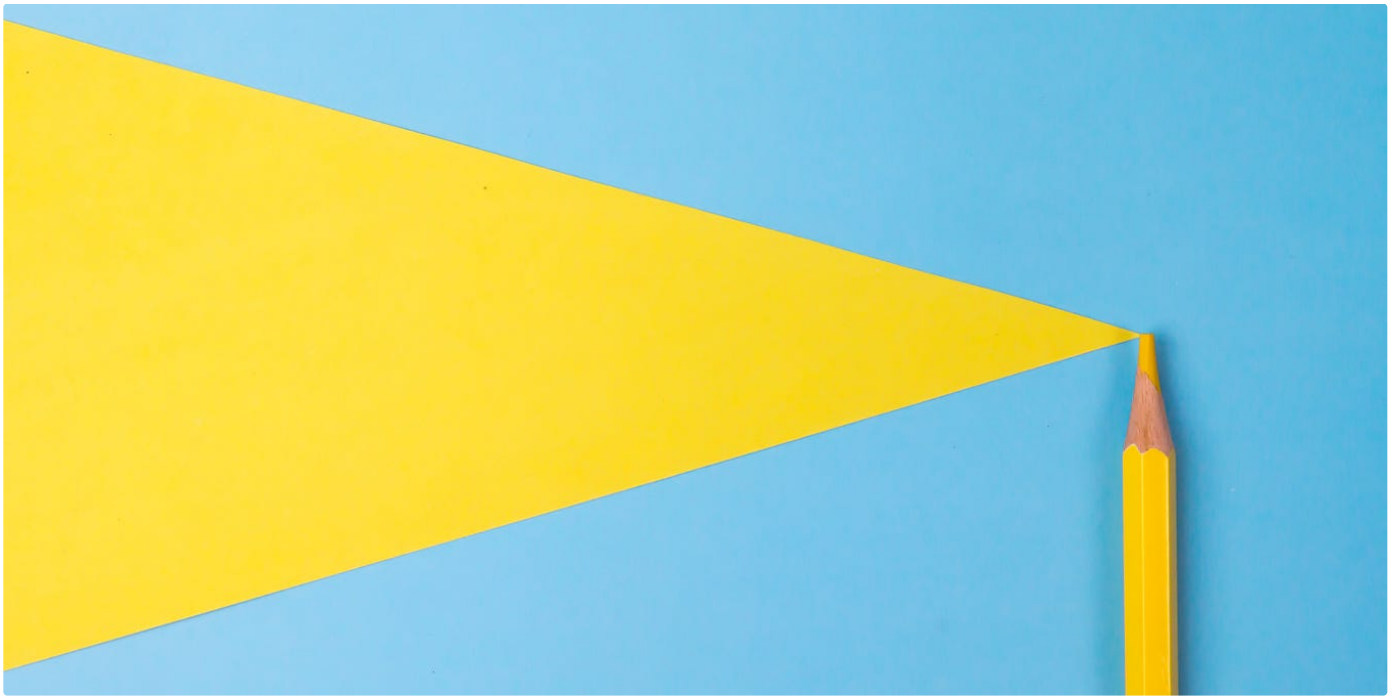
9 stories · 52 saves

Stories to Help You Level-Up at Work

19 stories · 44 saves

Staff Picks

323 stories · 81 saves



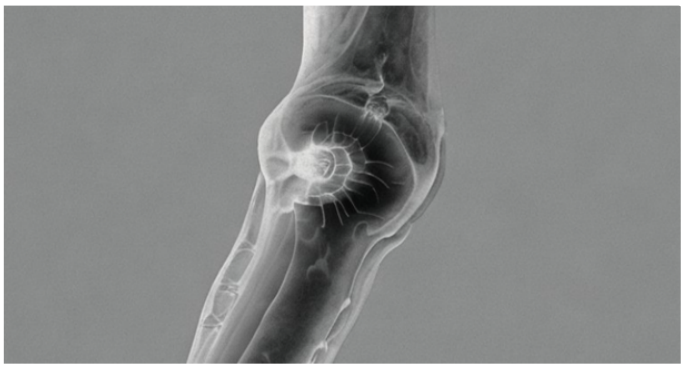
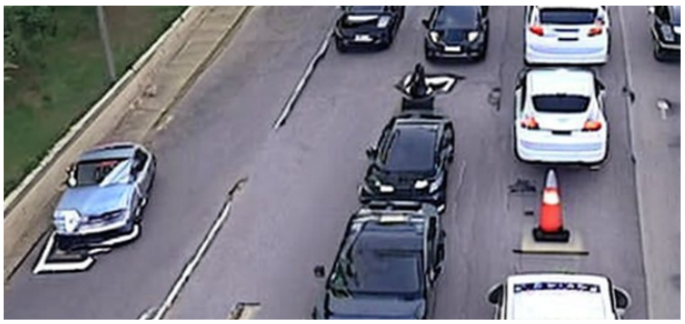
Diego Lopez Yse

Your Guide to Autoencoders

A brief introduction to Autoencoders: uses and architectures

8 min read · 5 days ago

383 5

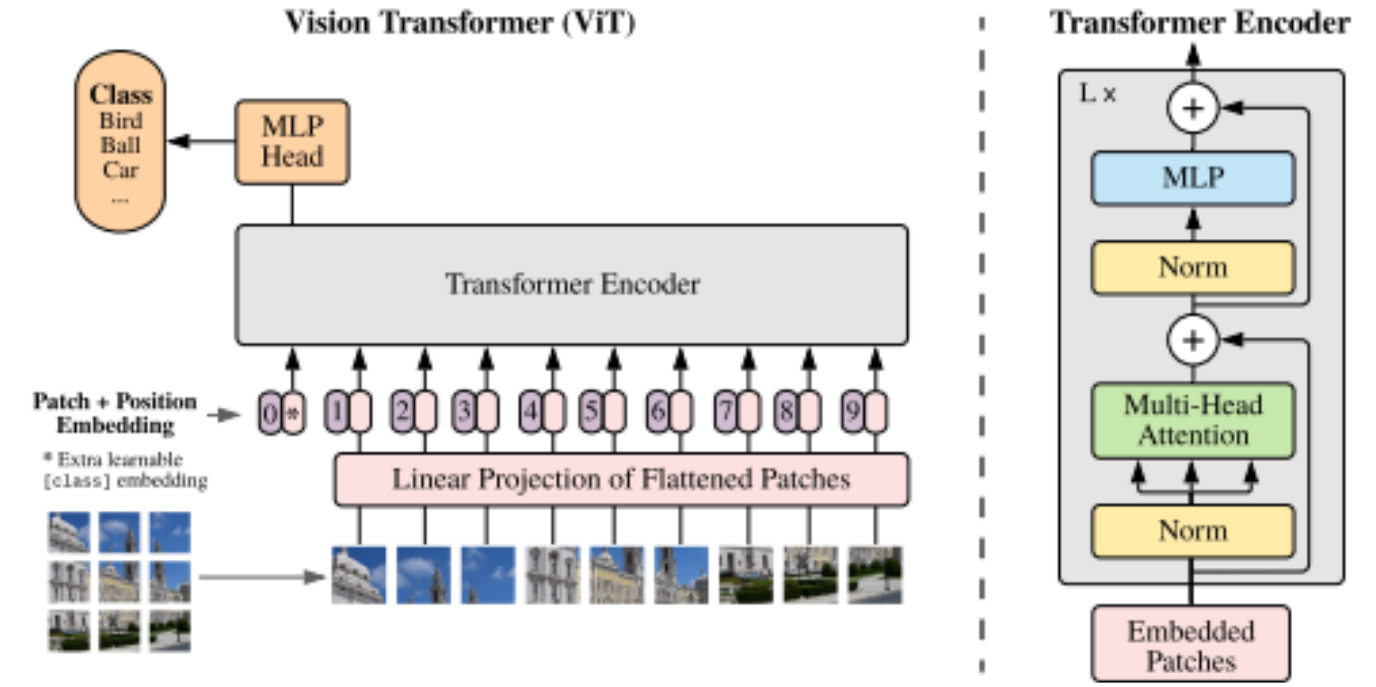

Chiron Bang

Data augmentation using Diffusion Models: Case of Medical Imaging

Artificial Intelligence and particularly machine learning and deep learning have taken the world by storm. Nowadays it is used in many...

10 min read · Dec 6, 2022

10



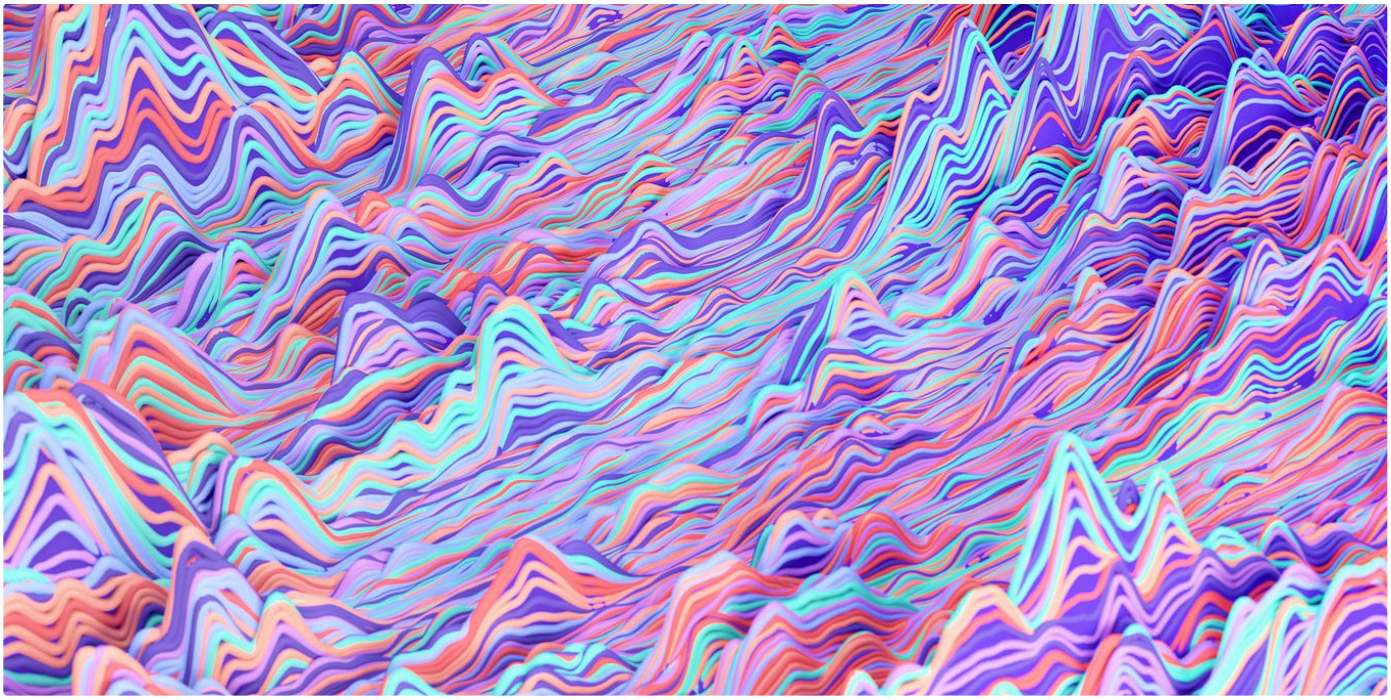
Vikram Pande

CNNs and Vision Transformers: Analysis and Comparison

Exploring the effectiveness of Vision Transformers and Convolutional Neural Networks (CNNs) in image classification tasks.

6 min read · Apr 29

80 3



Gonzalo Recio

AI Trends of May 2023 You Need to Know

Summary of latest AI trends for May 2023

8 min read · 5 days ago

 72  3