Member-only story

Building a Residual Network with PyTorch

The Moment When Networks Become Really Deep

Tim Cheng · Follow

Published in Towards Data Science

5 min read · Aug 29, 2021

Listen Share

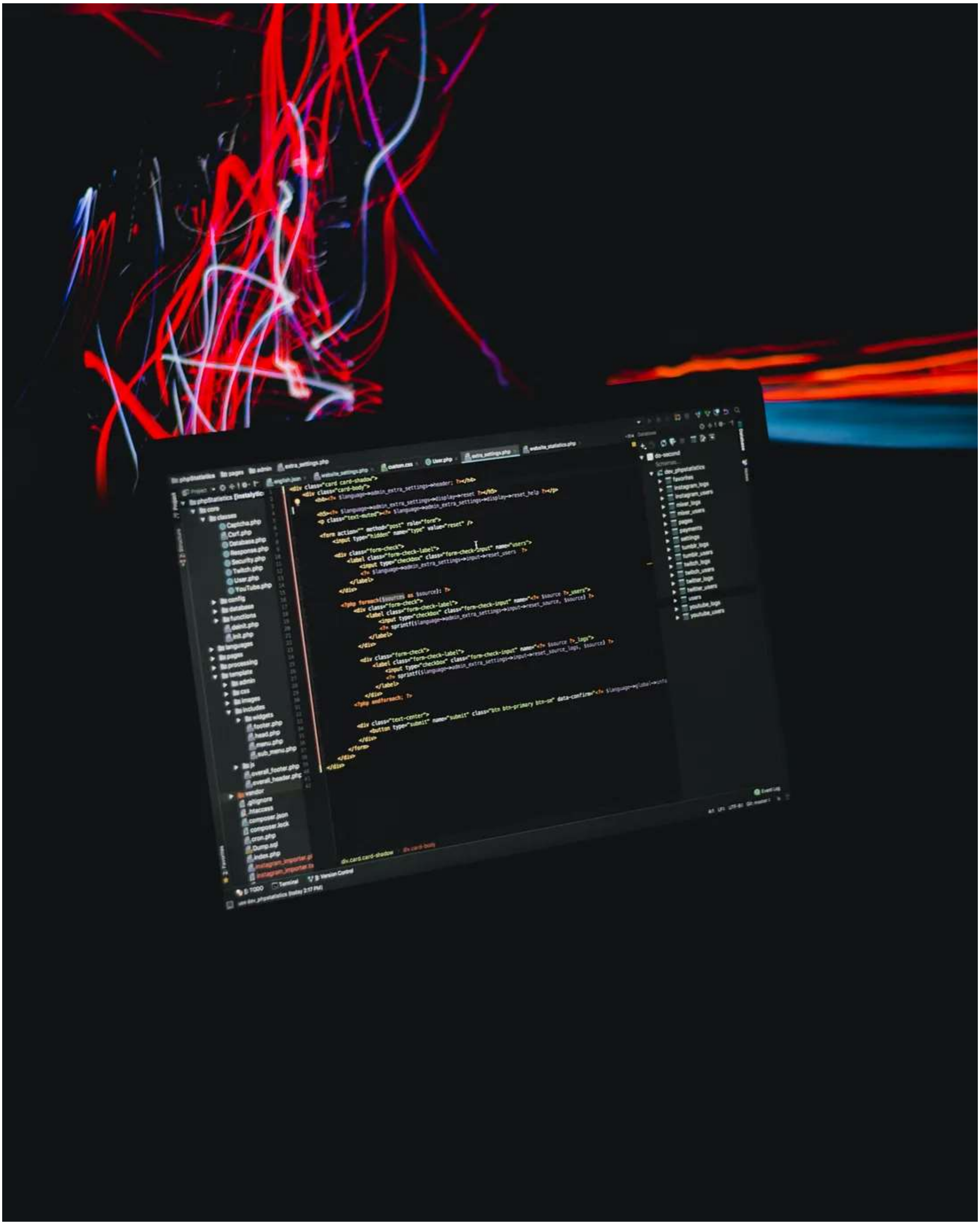


Image from Unsplash.

Autonomous driving, face detections, and numerous computer applications owe their success to deep neural networks. Many may not realize, however, that the blossom of computer vision advancements was due to a specific type of architecture: residual networks. In fact, state-of-the-art results that led to this AIdominant world were only made possible with the invention of residual blocks — a concept so simple yet elegant that led to the leap in creating truly 'deep' networks.

This article dives into the intuition behind a residual network and an implementation in PyTorch to train ResNets on image classification tasks.

Before Getting 'Deep', What is the Degradation Problem?

Theoretically, a deeper network with more variables is a better function for approximating difficult tasks such as image understanding. However, empirical tests have shown that traditional deeper networks are much harder to train, performing even worse than shallower networks. We refer to this as the *degradation* problem.

This phenomenon is unintuitive, as supposedly if we have two networks equal number of layers, with the second one adding *x* layers upfront, the worst case scenario should be the first *x* layers outputting an identical mapping to the original input, and thus having equal performances.

The conjecture is that poorer performance was caused by identical mapping to the original input being forgotten along the way, and thus networks (e.g., VGG-16) were bounded to around 10–20 layers at most during the early 2010s.

Residual Architecture

A residual network is a simple and straightforward approach that targets the aforementioned *degradation* problem by creating a shortcut, termed skip-connection, to feed the original input and combine it with the output features after a few stacked layers of the network.

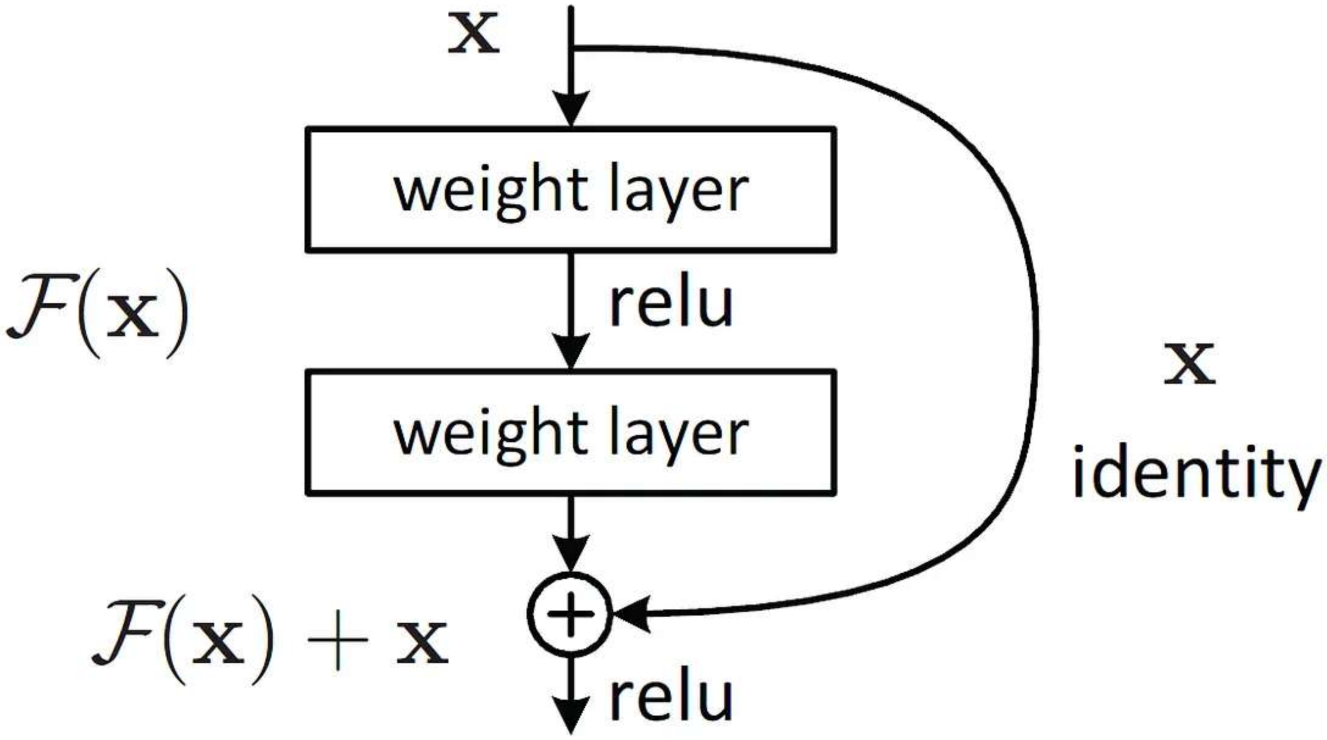


Figure 1. A simple residual block. Source: https://arxiv.org/abs/1512.03385

Formally, as shown in Figure 1., given the inputs into the stacked layers as x, and the layers of networks as a function F, the output y is as the following:

$$y = F(x) + x$$

When the dimensions of $F(x)$ and x doesn't match, one can simply perform a linear projection during the skip-connection to change the dimension of x.

We refer to the entire pipeline above as one residual block, and we can have multiple residual blocks to construct a much deeper network without the original *degradation* issue.

Computing Environment

```python
"""
The following is an import of PyTorch libraries.
"""
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
from torchvision import datasets, transforms
from torchvision.utils import save_image
import matplotlib.pyplot as plt
import numpy as np
import random
```

importLibraries.py hosted with ❤ by GitHub view raw

Dataset

To showcase the ability of residual networks, we perform testing on two datasets: the simpler MNIST dataset comprising 60000 images of handwritten digits from 0 to 9, and a more complicated dataset of CIFAR-10.

Often during testing, one may refer to more than one dataset, whether for research purposes or just to see which model generalises better. It is therefore very convenient when all the datasets are being organised into one platform. Luckily, a young startup named Graviti offers to host many of the infamous datasets on their platform. One can simply download them directly to perform further training and testing.

Hardware Requirements

It is preferable to train neural networks on GPUs, as they increase the training speed significantly. However, if only CPUs are available, you may still test the program. In our case, a simpler residual-block-based network with only a few network should be runnable in both types of devices, whereas more complicated models such as ResNet-152 will be more suitable to be ran on GPUs. To allow your program to determine the hardware itself, simply use the following:

```python
"""
Determine if any GPUs are available
"""
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

torchDevice.py hosted with ❤ by GitHub view raw

Building a Residual Block

This section provides a tutorial on PyTorch for the simplest type of residual block one can create on a convolutional neural network with the dimension of the input and output being identical.

One may create that using the PyTorch *nn.Module* as the following:

Using Pre-Existing ResNet Models

Some networks that utilizes the residual architecture have already been proven successful under big dataset like ImageNet. Torchvision offers the checkpoints and architectures of networks surch as ResNet-34, ResNet-50, and ResNet-152 pre-built inside their library. One can simply retrieve their models through the following:

It is important, however, to update the final layer of ResNet if you are finetuning the network to a dataset that is not ImageNet, as the one-hot vector in the end is equal to the number of classes of a dataset.

Results

We train our networks for 50 epochs and can easily achieve around 99% on the MNIST dataset and 90% on the CIFAR-10 dataset for both ResNet-34 and ResNet-152.

Based on the results of the original paper by He et al., we can also see that Residual architectures perform significantly better on the ImageNet dataset than VGG and a network with equal number of layers but no residual architecture.

These results can be retrieved directly from the paper here.

Conclusion

The creation of the residual architecture by He et al. is arguably one of the greatest inventions in recent neural network developments for computer vision. Almost all networks today, even networks beyond convolutional networks, have shadows resembling the concept of it for better and deeper networks.

The simple yet elegant approach have created numerous possibilities to push the front-edge of machine's understanding of the human world.

Thank you for making it this far 🙏 ! I will be posting more on different areas of computer vision/deep learning. Make sure to check out my other articles on computer vision methods! If you are interested in the Graviti platform, feel free to join the discord channel too!

Data Science Artificial Intelligence Machine Learning Computer Science

Neural Networks

Follow

Written by Tim Cheng

357 Followers · Writer for Towards Data Science

Oxford CS | Top Writer in AI | Posting on Deep Learning and Vision

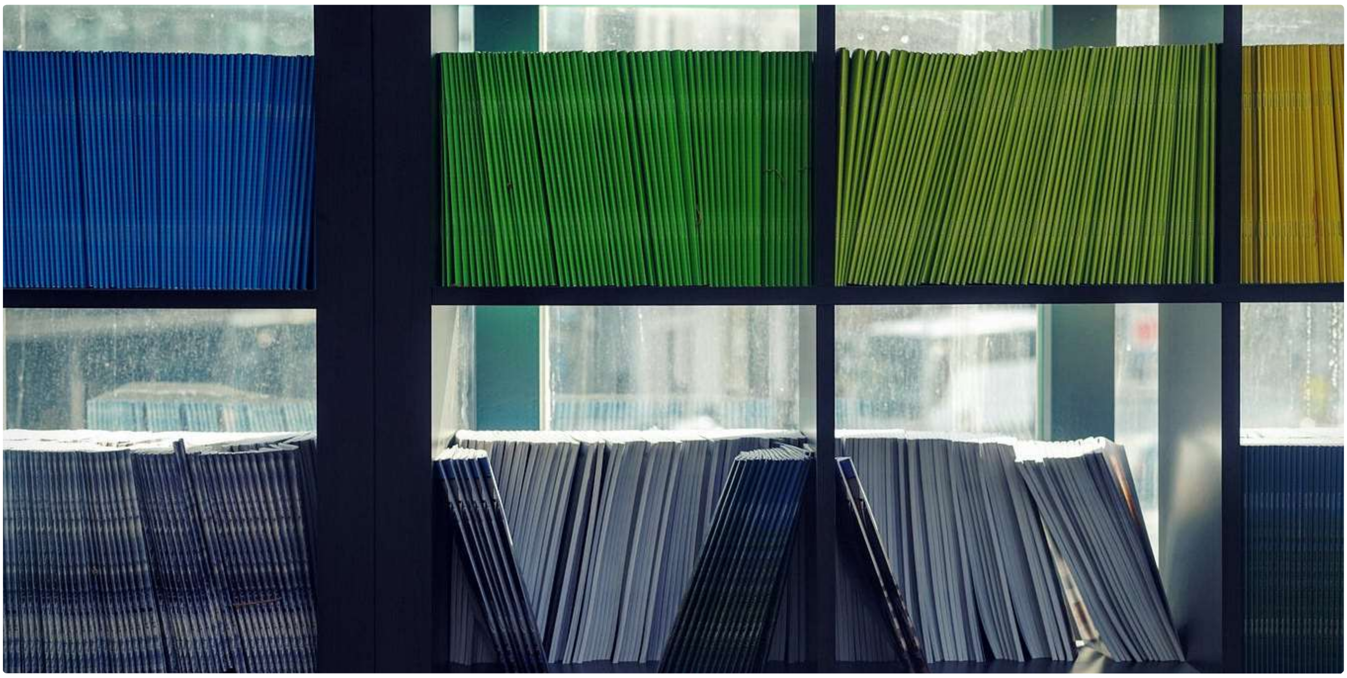
More from Tim Cheng and Towards Data Science

Tim Cheng in Towards Data Science

A Very Basic Overview of Neural Radiance Fields (NeRF)

Can they one day replace photos?

· 4 min read · Aug 1, 2022



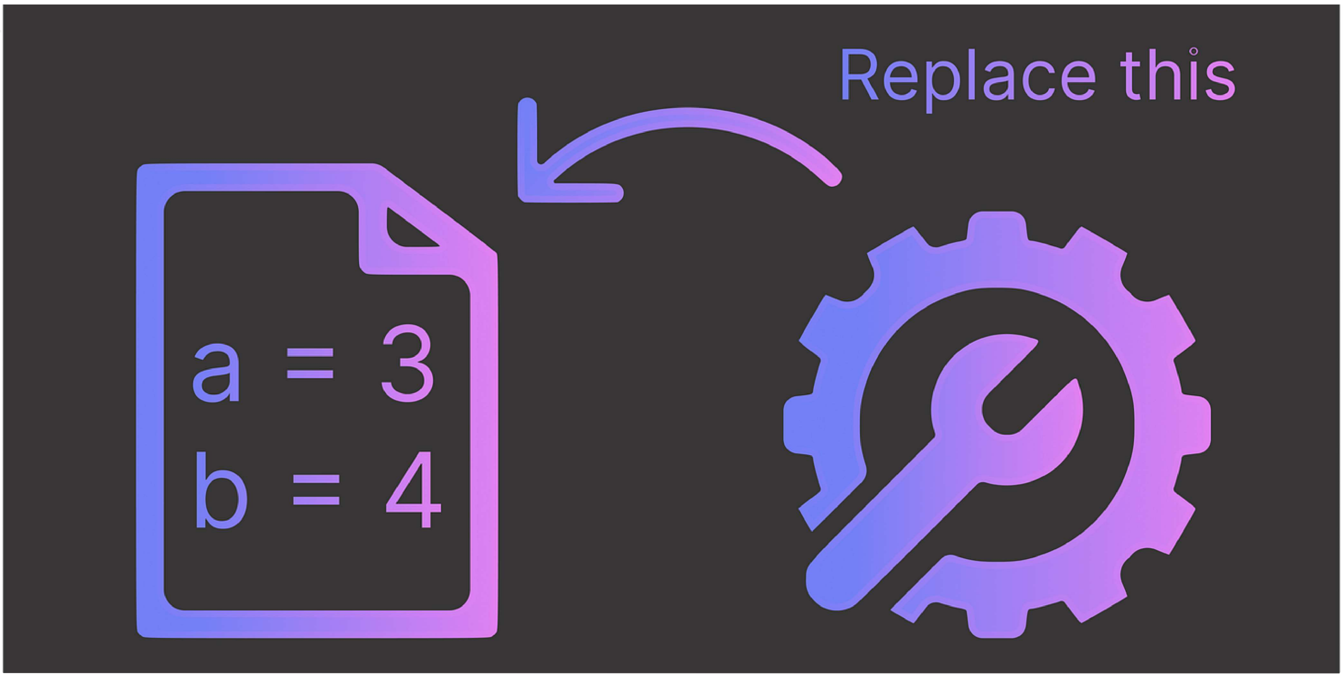
Jacob Marks, Ph.D. in Towards Data Science

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

4K 48



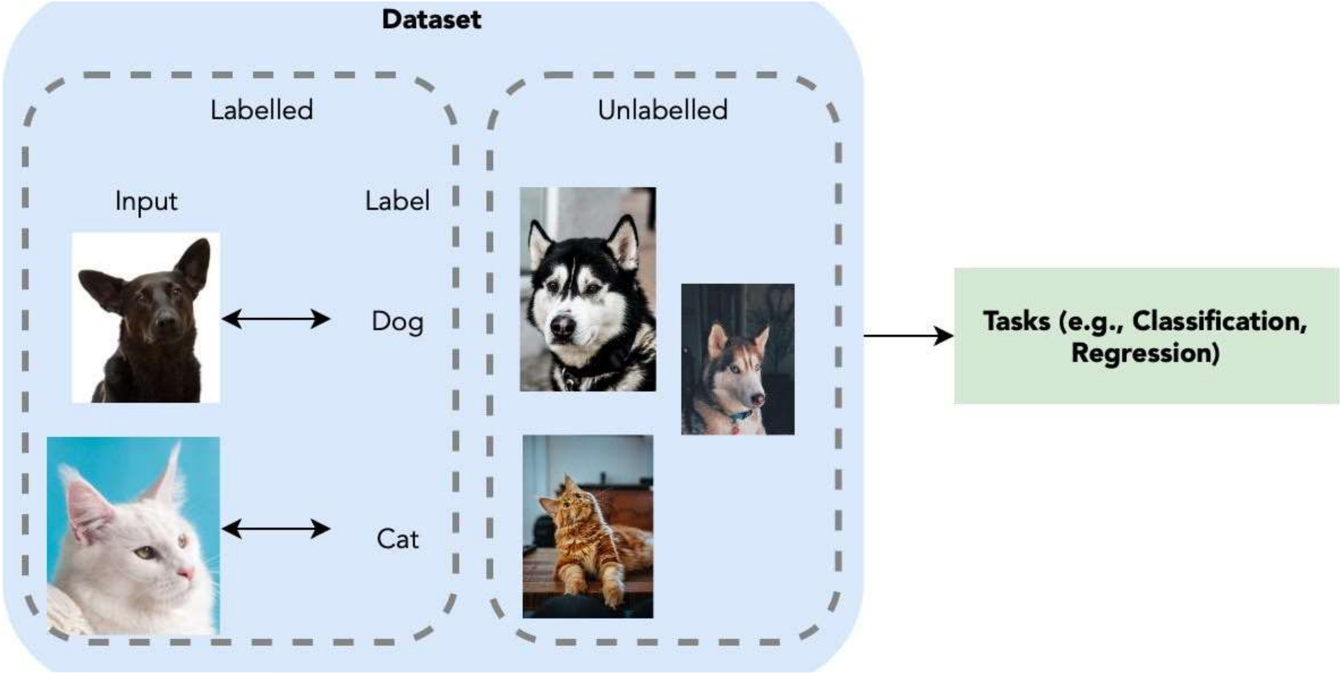
Khuyen Tran in Towards Data Science

Stop Hard Coding in a Data Science Project—Use Config Files Instead

And How to Efficiently Interact with Config Files in Python

· 6 min read · May 26

1.6K 19



Tim Cheng in Towards Data Science

Supervised, Semi-Supervised, Unsupervised, and Self-Supervised Learning

Demystifying each learning task

· 6 min read · Nov 26, 2021

304 2

See all from Tim Cheng

See all from Towards Data Science

Recommended from Medium

Leonie Monigatti in Towards Data Science

A Visual Guide to Learning Rate Schedulers in PyTorch

LR decay and annealing strategies for Deep Learning in Python

· 9 min read · Dec 7, 2022

1.1K 6

Will Badr in Towards Data Science

The Secret to Improved NLP: An In-Depth Look at the nn.Embedding Layer in PyTorch

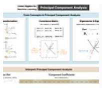
Dissecting the `nn.Embedding` layer in PyTorch and a complete guide on how it works

· 8 min read · Jan 25

Lists

Predictive Modeling w/ Python

18 stories · 10 saves



Practical Guides to Machine Learning

10 stories · 24 saves

Natural Language Processing

345 stories · 7 saves

ChatGPT prompts

17 stories · 5 saves

Steins

Diffusion Model Clearly Explained!

How does AI artwork work? Understanding the tech behind the rise of AI-generated art.

· 7 min read · Dec 26, 2022

1.2K 5

Steins

Stable Diffusion Clearly Explained!

How does Stable Diffusion paint an AI artwork? Understanding the tech behind the rise of AI-generated art.

· 5 min read · Jan 2

1.2K 6

Rukshan Pramoditha in Data Science 365

Determining the Right Batch Size for a Neural Network to Get Better and Faster Results

Guidelines for choosing the right batch size to maintain optimal training speed and accuracy while saving computer resources

· 4 min read · Sep 27, 2022

The PyCoach in Artificial Corner

You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of ChatGPT Users

Master ChatGPT by learning prompt engineering.