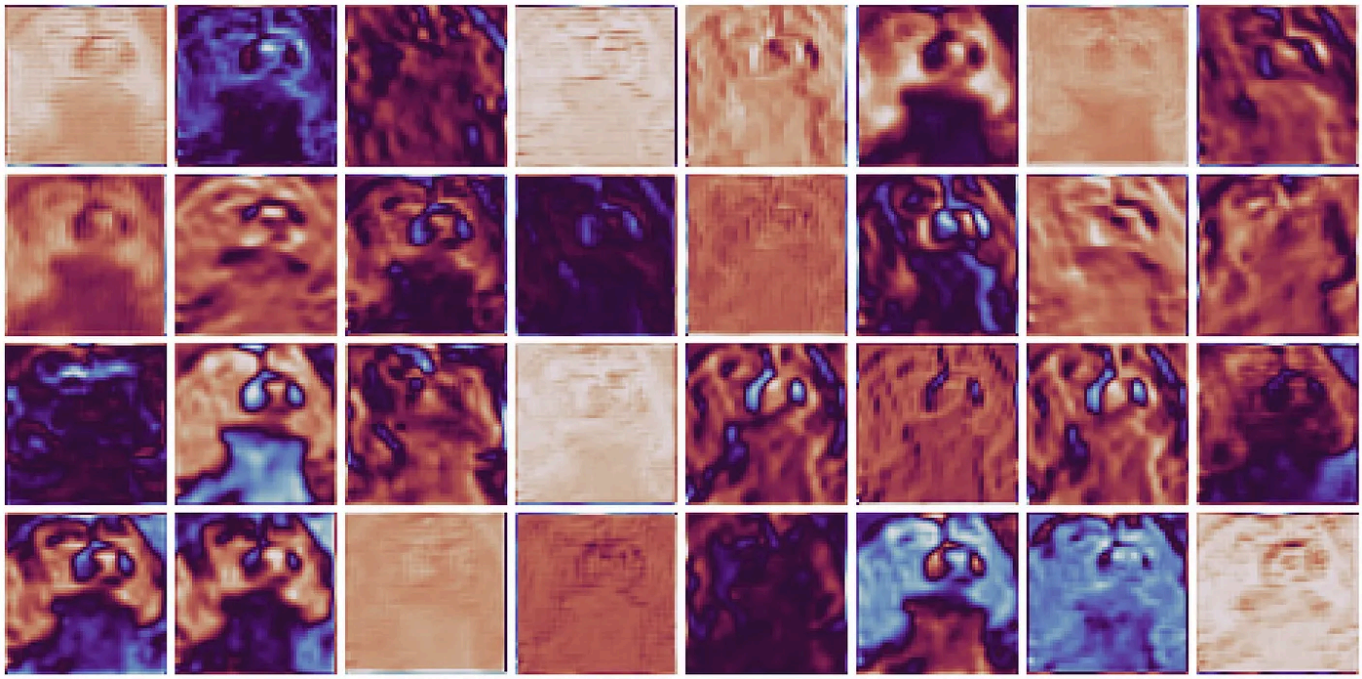
How to Visualize Layer Activations in PyTorch

Reza Kalantar · Follow

3 min read · Apr 15, 2024

This tutorial will demonstrate how to visualize layer activations in a pretrained ResNet model using the CIFAR-10 dataset in PyTorch. CIFAR-10 is a well-known dataset consisting of 60,000 32x32 color images categorized into 10 classes, making it a popular choice for developing and testing image recognition algorithms. Visualizing activations, the outputs of various layers within the model, is crucial for understanding how deep neural networks process visual information, which can help diagnose model behavior and inspire improvements.



Environment Setup

Ensure your environment is prepared with the necessary libraries:

```
pip install torch torchvision matplotlib
```

Load and Visualize CIFAR-10 Data

First, load the CIFAR-10 dataset and visualize some of the images. This helps in understanding the input that the model processes.

```
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt

# Transformations for the images
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

# Load CIFAR-10 dataset
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4, shuffle=True)

# Function to show images
def imshow(img):
    img = img.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    img = std * img + mean  # unnormalize
    plt.imshow(img)
    plt.show()

# Get some images
dataiter = iter(trainloader)
images, labels = next(dataiter)

# Display images
imshow(torchvision.utils.make_grid(images))
```

Sample images from the CIFAR-10 dataset

Load Pretrained Model and Set Up Hooks

Load a pretrained ResNet model from PyTorch's model zoo and set up hooks on specific layers to capture activations during the forward pass.

```
import torch
from torchvision.models import resnet18

# Load pretrained ResNet18
model = resnet18(pretrained=True)
model.eval()  # Set the model to evaluation mode

# Hook setup
activations = {}
def get_activation(name):
    def hook(model, input, output):
        activations[name] = output.detach()
    return hook

# Register hooks
```

```
model.layer1[0].conv1.register_forward_hook(get_activation('layer1_0_conv1'))
model.layer4[0].conv1.register_forward_hook(get_activation('layer4_0_conv1'))
```

Run Data Through the Model

Process the images through the model to capture activations.

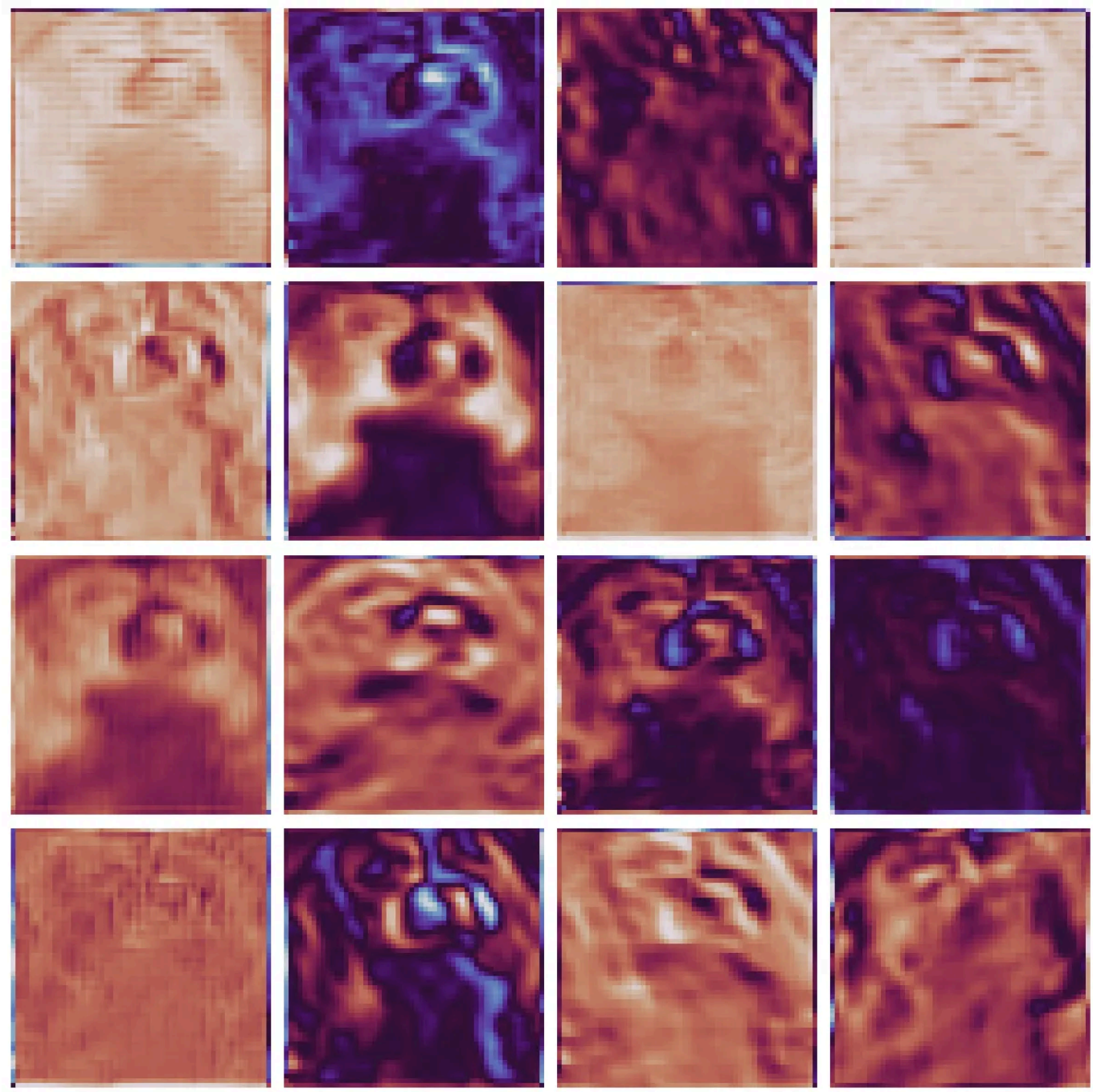
```
# Run the model
with torch.no_grad():
    output = model(images)
```

Visualize Activations

Using the hooks set up in Step 3, visualize the activations to gain insight into what features each layer is extracting.

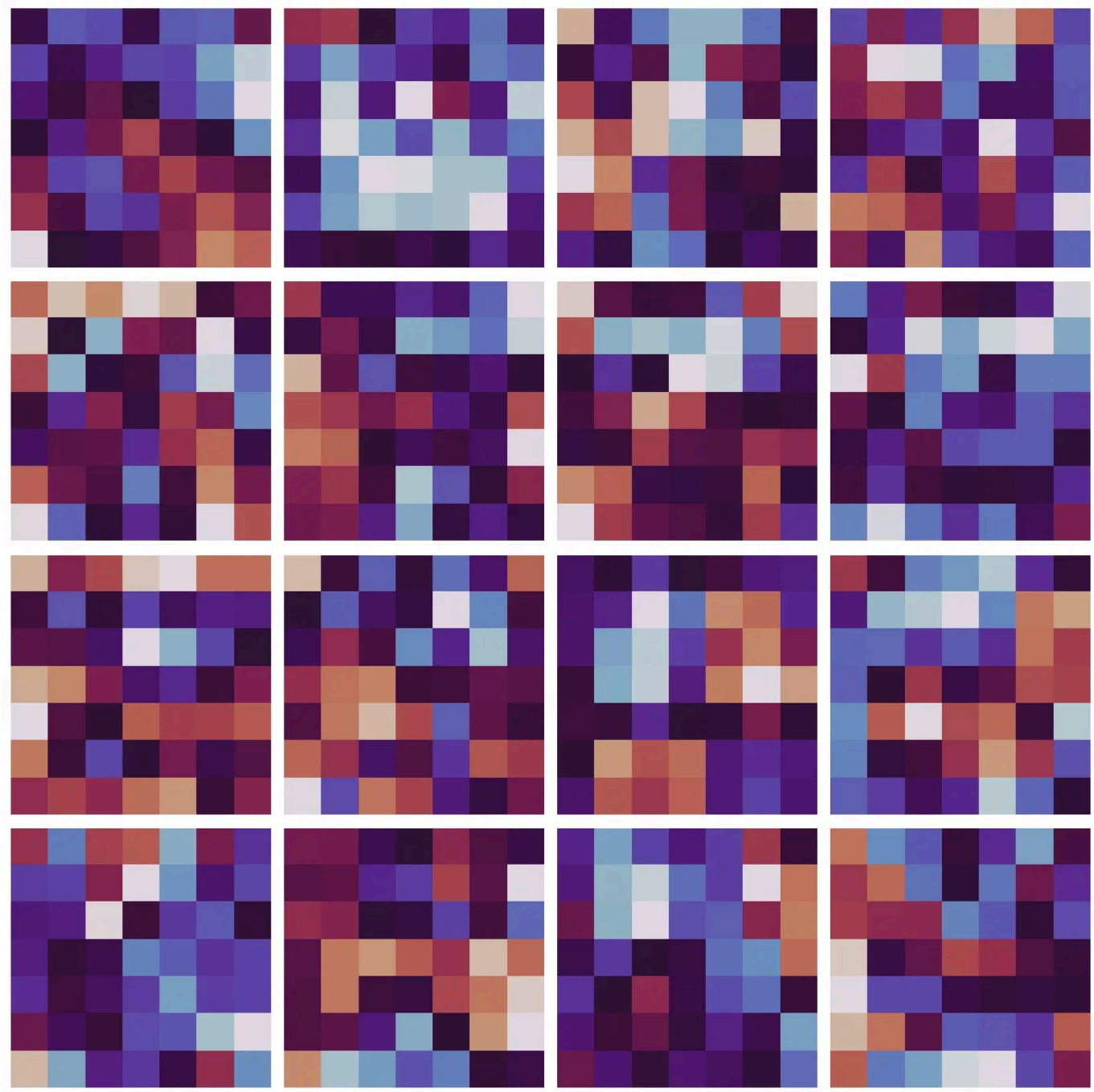
```
# Visualization function for activations
def plot_activations(layer, num_cols=4, num_activations=16):
    num_kernels = layer.shape[1]
    fig, axes = plt.subplots(nrows=(num_activations + num_cols - 1) // num_cols, ncols=num_cols, figsize
    for i, ax in enumerate(axes.flat):
        if i < num_kernels:
            ax.imshow(layer[0, i].cpu().numpy(), cmap='twilight')
            ax.axis('off')
    plt.tight_layout()
    plt.show()
```

```
# Display a subset of activations
plot_activations(activations['layer1_0_conv1'], num_cols=4, num_activations=16)
```



Visualize 16 activations from layer 'layer1_0_conv1'

```
plot_activations(activations['layer4_0_conv1'], num_cols=4, num_activations=16)
```



Visualize 16 activations from layer 'layer4_0_conv1'

Conclusion

Visualizing activations helps in understanding how various layers in a convolutional neural network respond to different features in the input images. This method provides insights into the internal workings of the model, highlighting which features are most influential for the classification task and where the model might be focusing incorrectly, thus informing potential improvements. By visualizing different layers, you can assess whether early layers capture basic features like edges and textures, while deeper layers capture more complex features. This knowledge is invaluable for diagnosing issues, tweaking layer architectures, and improving the overall model performance.

Thank you for reading! If you find my blogs interesting or would like to get in touch, reach out on here, Github or LinkedIn.

Pytorch | Machine Learning | Programming | Deep Learning | Technology

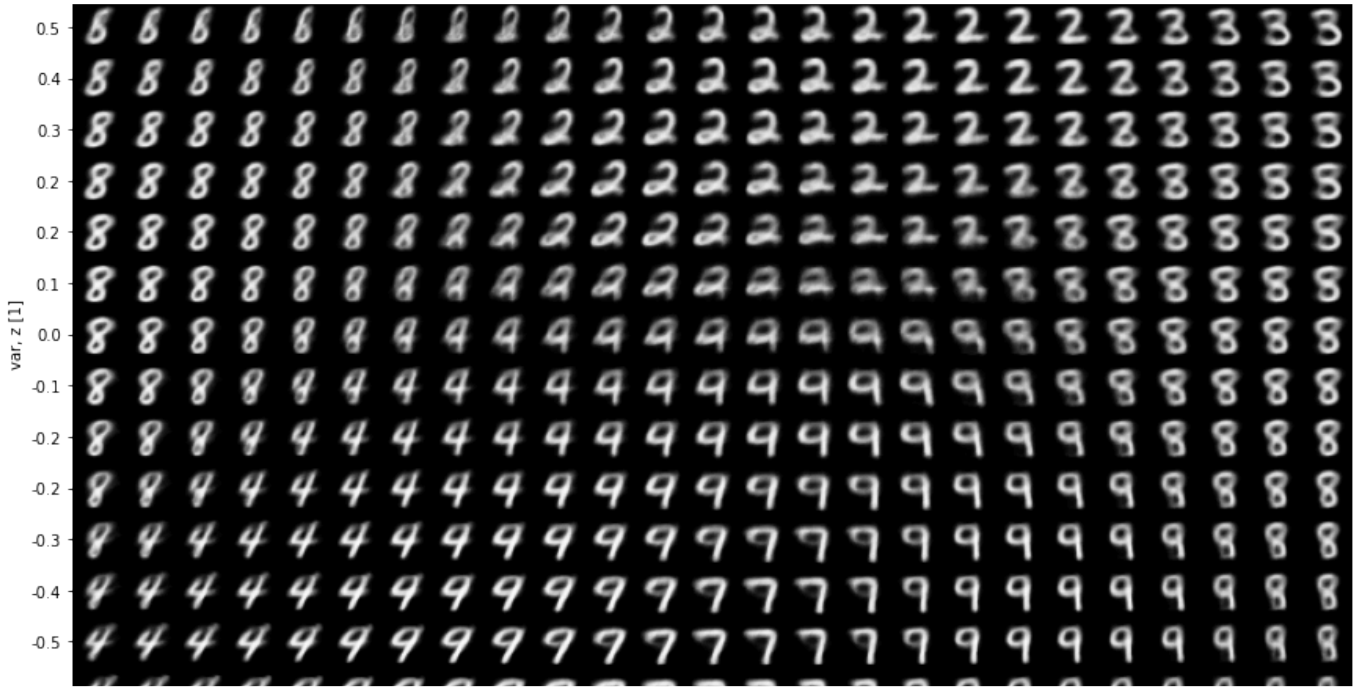
Follow

Written by Reza Kalantar

1.1K Followers

Medical AI Researcher by Profession • Scientist/Engineer by Trade • Investor by Instinct • Explorer by Nature • Procrastinator by Choice

More from Reza Kalantar

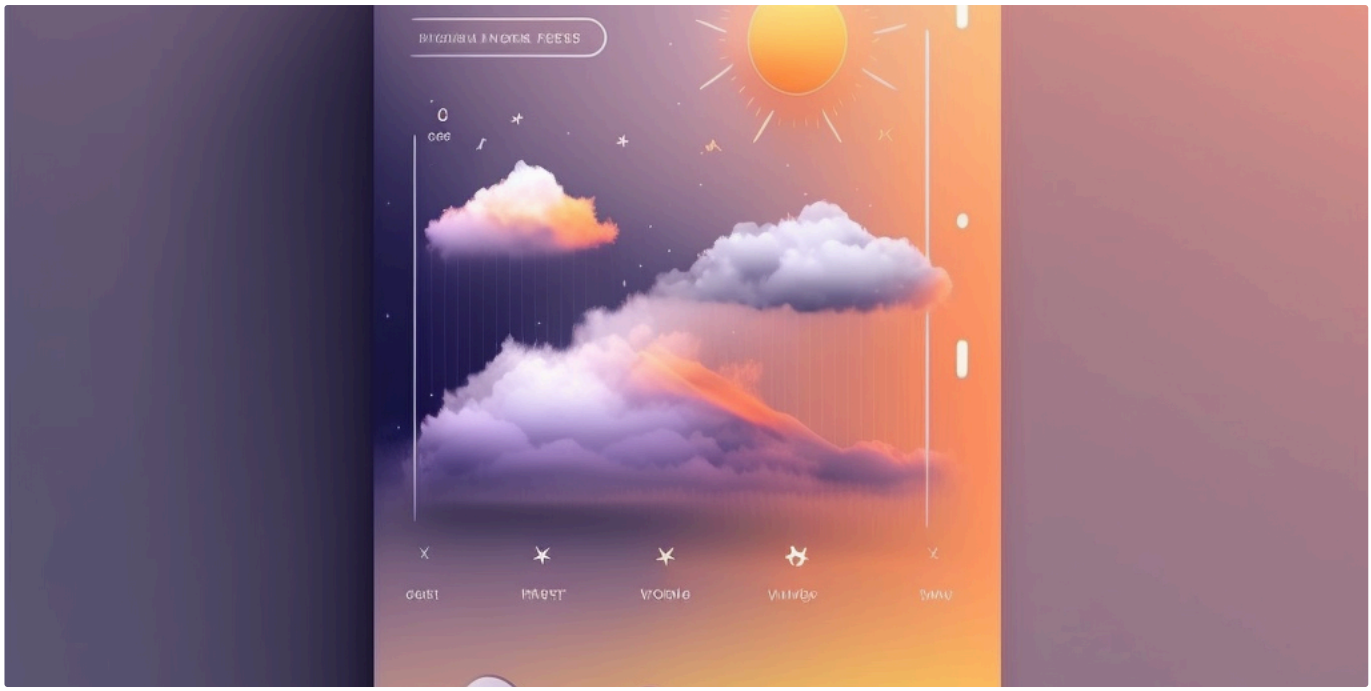


Reza Kalantar

Variational Auto-Encoder (VAE)—PyTorch Tutorial

Step-to-step guide to design a VAE, generate samples and visualize the latent space in PyTorch.

Nov 20, 2022 204 7



Reza Kalantar

How to Build a Simple Weather App in Python with OpenWeatherMap API

This tutorial will guide you through the process of building a simple weather app using Python and OpenWeatherMap API.

Mar 6, 2023 124 3

Reza Kalantar in Python in Plain English

How to Freeze Model Weights in PyTorch for Transfer Learning: Step-by-Step Tutorial

Transfer learning is a machine learning technique where a pre-trained model is adapted for a new, but similar problem. One of the key steps...

Sep 5, 2023 36

Reza Kalantar

Segment Anything Model (SAM) for Medical Image Segmentation

In the evolving landscape of artificial intelligence (AI), medical imaging stands as a field witnessing profound transformation. Riding...

May 30, 2023 215 1

See all from Reza Kalantar

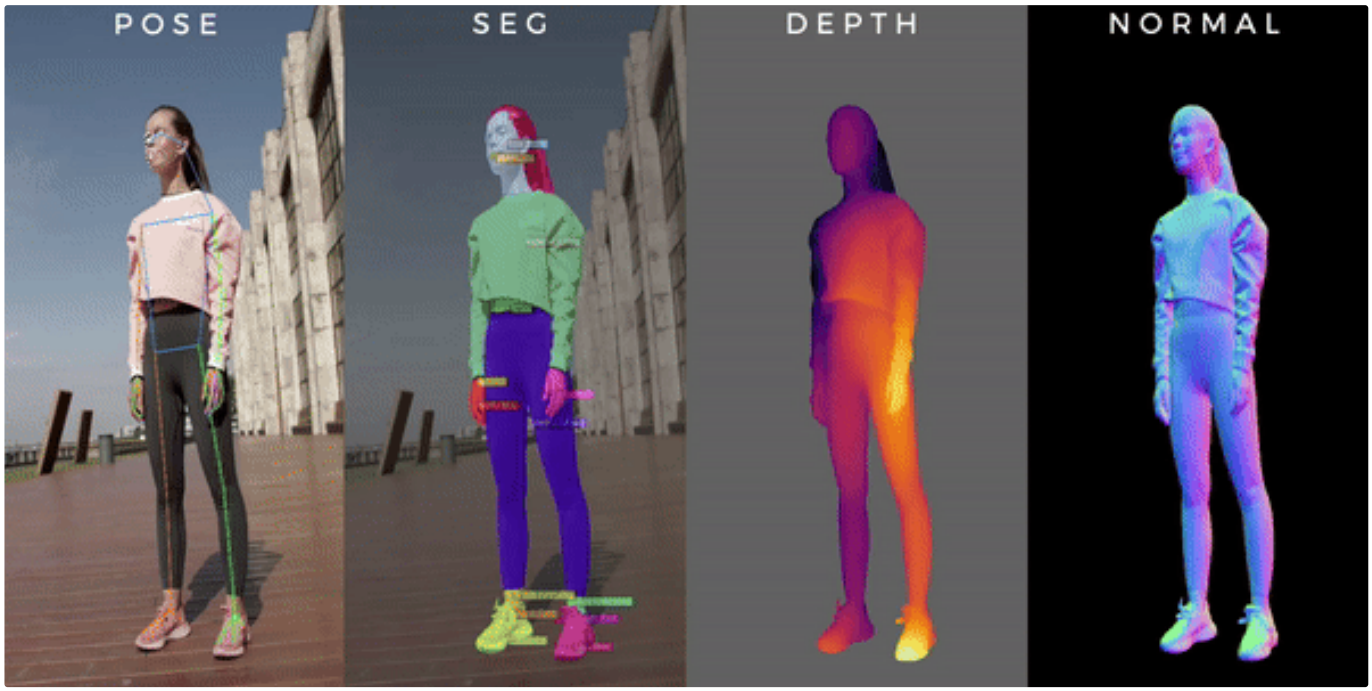
Recommended from Medium

Deepak N R in Python in Plain English

YOLOv8 vs YOLOv11: A Comparison

YOLO models have been state-of-the-art in computer vision for real-time object detection, and segmentation tasks. YOLO11 represents the...

Oct 10 192 1



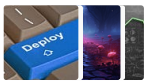
AI Papers Academy

Sapiens by Meta AI: Foundation for Human Vision Models

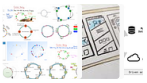
In this post we dive into Sapiens, a new family of computer vision models by Meta AI that show remarkable advancement in human-centric...

Aug 27 103

Lists



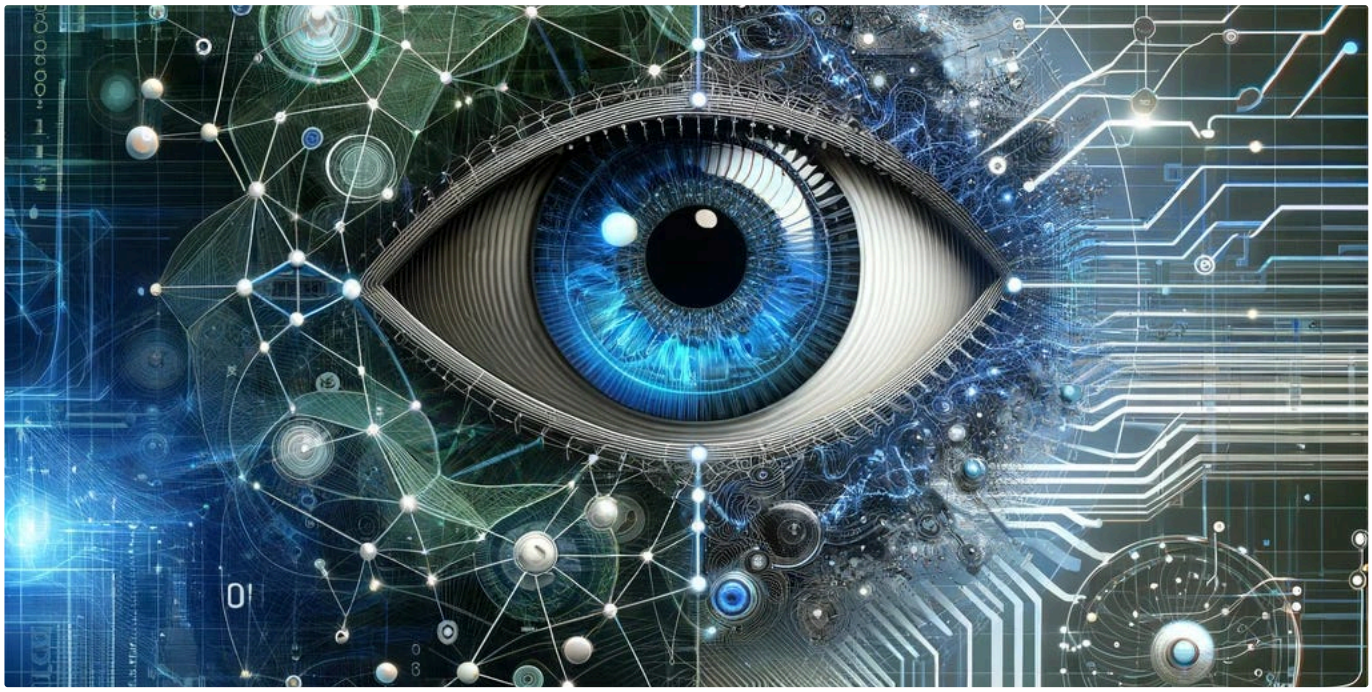
Predictive Modeling w/ Python
20 stories · 1610 saves



General Coding Knowledge
20 stories · 1672 saves

Practical Guides to Machine Learning
10 stories · 1963 saves

Coding & Development
11 stories · 860 saves

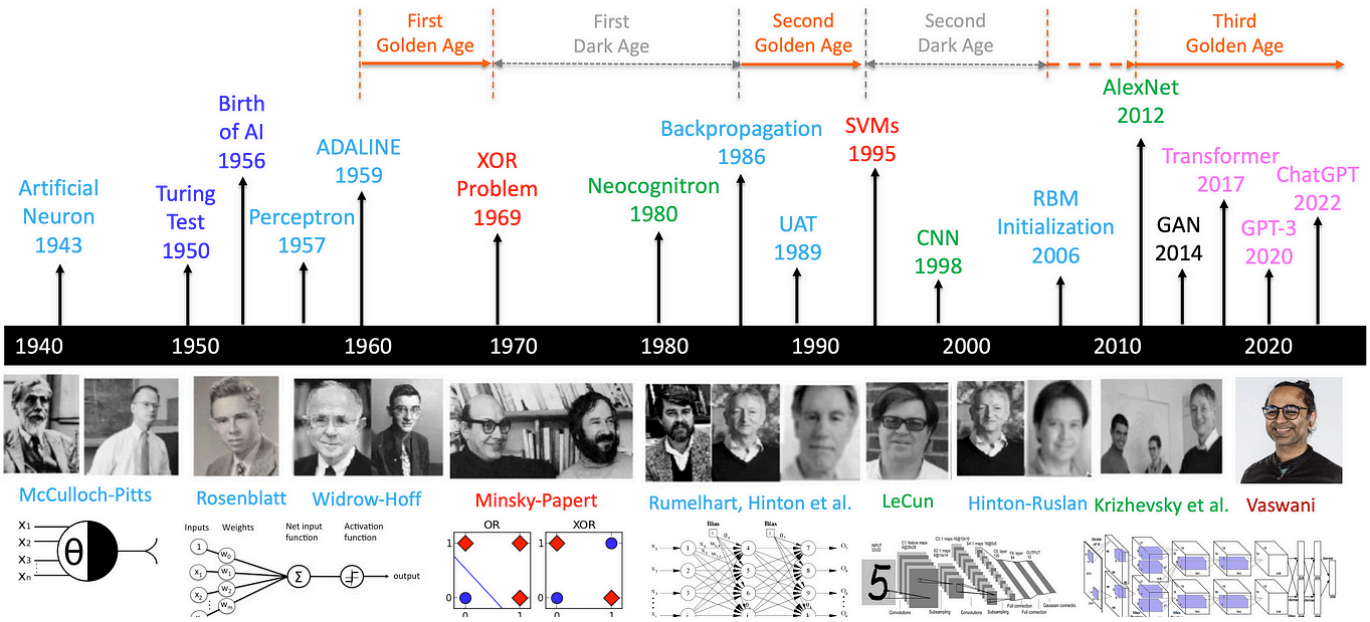


Ivan Drokin

Can KANs do Computer Vision?

An brief exploration of KANs for Computer Vision

Jun 11 133 1



LM Po

A Brief History of AI with Deep Learning

Artificial intelligence (AI) and deep learning have seen remarkable progress over the past several decades, transforming fields like...

Sep 1 242 5

Andrea Rosales

Liquid Neuronal Networks using Pytorch

Liquid Neuronal Networks (LNNs) were originally developed by the Computer Science and Artificial Intelligence Laboratory at MIT (CSAIL)...

May 23 245 1

U Vamsi Krishna

What Are Invertible Neural Networks and Why Do They Matter?

While looking at the fancy generative network concept, I wondered whether there were any neural networks that could be inverted. How may...

May 26 10 1

See more recommendations