Vision Dashboard - A One-Stop CV Learning Tool

Sashrika Surya[1], Srijarko Roy[1]
[1]Department of Computer Science and Engineering
ss7133@srmist.edu.in, sr8962@srmist.edu.in
SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu

Abstract

Computer Vision (CV) is a growing field that attracts many beginners in the field of Machine Learning. According to research, visual information is mapped better in students' minds (Williams, 2009[1]) and helps them retain information for a longer duration. However, the traditional educational methodology involves teaching theoretical concepts utilizing text-based explanations and audio. This results in most students not being able to visualize or understand the significant CV techniques, and thus students are unsure about how to approach CV as a field. In addition, CV models tend to be computationally heavy, expensive, and difficult to run from a beginner's point of view, which discourages students from pursuing the field seriously. This paper presents a method of demonstrating CV algorithms using a Vision Dashboard, keeping the aforementioned issues in mind.

Our approach allows students to run various CV methods on any image compiled on a single dashboard. This helps students visualize techniques like Object Detection, Instance Segmentation, Semantic Segmentation, Style Transfer, Image Classification, Super Resolution, Denoising, Image generation using GANs, and Face Detection efficiently, serving as an effective teaching tool.

Keywords: Computer Vision, Detection, Segmentation, Teaching

Introduction

IBM defines Computer Vision as “a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information”. If AI enables computers to think, Computer Vision allows them to see, observe and understand. CV includes a wide variety of tasks such as Image Classification, Detection, Segmentation, Denoising, Style Transfer, etc. with several techniques and algorithms to perform each of them. With its visual aspect, not only do CV datasets tend to be large and require more memory to pre-process but CV models also tend to be heavy, computationally expensive and slow to run, making it tedious for students to learn, as well as for teachers to teach.

Computer Vision tasks are best understood when appropriately visualized, clearly indicating the similarities and essential differences between each of the tasks. However, the conventional teaching methods followed focus on developing theoretical knowledge through written documentation and audio, making it difficult for students to distinguish between the tasks, understand their working and imagine how the outcomes of each task would look. We propose a way of learning various Computer Vision concepts with the help of visual elements such as a Tree for classifying various tasks, resources including videos, blogs, discussions, and the like, a section for ‘Frequently Asked Questions’ to help students get through the most common doubts and issues, and most importantly a Visualizer to run the most widely used Computer Vision algorithms hands-on and visualize their outcomes. In our project, we have integrated all the aforementioned ideas in a single dashboard for students to work interactively and learn more practically, thus solving the problems faced while learning and teaching CV.

Method

Our project, VisionDash, consists of a dashboard providing the users with an option to overview various CV tasks, learn about different CV techniques utilizing the resources provided on the dashboard and supplement their knowledge with the SOTA implementations of each algorithm.

The algorithms provided are divided into broad categories: Image Classification, Detection, Segmentation, Denoising, Generative Adversarial Networks, and Style Transfer. These categories are further divided into the specific tasks of Image Classification, Object Detection and Face Detection, Instance Segmentation and Semantic Segmentation, Noise2Noise, Super Resolution GAN, and Fast Style Transfer.

Image Classification utilizes the EfficientNetB7 architecture from the Torchvision library[2]. This model is pre-trained on the ImageNet dataset. It identifies and classifies objects present in an image according to the classes present in the ImageNet library.

Face Detection is implemented using pre-trained weights of the MTCNN model from the 'facenet-pytorch'[3] library that draws bounding boxes around human faces present in an input image.

Object Detection is a model used to detect and draw bounding boxes around objects detected in an image. These objects are instances of COCO Category Names. Our implementation utilizes RetinaNet from the Torchvision library to implement Object Detection.

Instance Segmentation draws bounding boxes with class labels and a pixel to pixel segmentation mask around the objects detected in an input image using the Mask RCNN ResNet50 FPN model pre-trained on the MS COCO dataset.

Semantic Segmentation performs pixel to pixel segmentation of objects within an input image using the DeepLabV3 ResNet101 model pre-trained on a subset of COCO train2017, on the 20 categories present in the Pascal VOC dataset.

Noise2Noise is a model introduced in 2018 used to remove noise from images. Our implementation utilizes open source weights[4] for Noise2Noise, trained on a subset of MS COCO dataset. This model can remove two types of noise, gaussian and textual noise.

Style Transfer changes an input image so that it resembles the content of the content image and the texture/artistic style of the style image to reproduce it as a new artistic stylized image. Fast Style Transfer has been implemented using open source weights[5] trained for four different style images namely ‘candy’, ‘mosaic’, ‘rain princess’ and ‘udnie’.

SRGAN is a Generative Adversarial Network used explicitly for super-resolution i.e., improving the clarity of images. Our version of SRGAN is used to quadruple the resolution of an image by using pre-trained, open source weights[6].

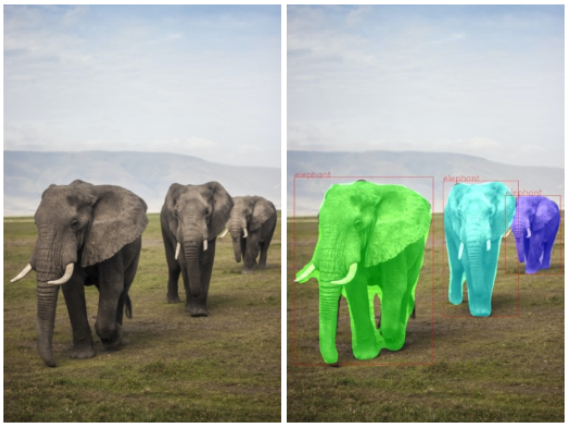
These models have been integrated into a single dashboard built using Streamlit[7] which turns data scripts into shareable web apps. Different widgets have been used to make the user interface as interactive and visually impactful as possible.

As a part of VisionDash, we provide resources to serve as a self-study tool for each CV technique implemented. This knowledge can be further augmented by the FAQs provided for each section.

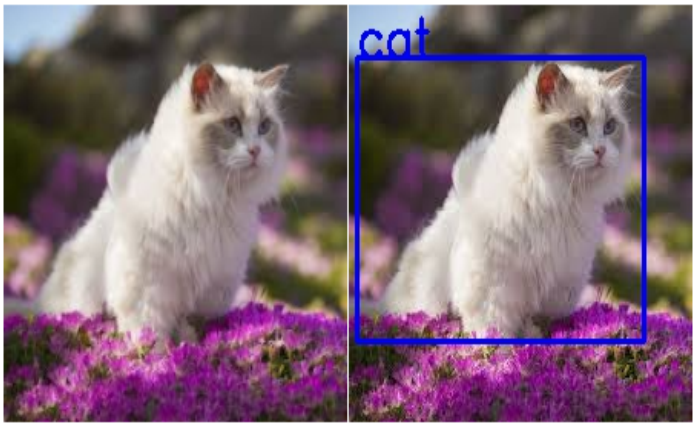
Results and Discussion

Deploying heavy ML models presents a problem due to the time and space required to run such models. This paper utilized Torchvision for most models, which helped reduce space complexity and let the application run smoothly. For the models not present in Torchvision like Noise2Noise, Style Transfer, and SRGAN, the model weights are stored on Google Drive and are downloaded only when required, thus optimizing for space.

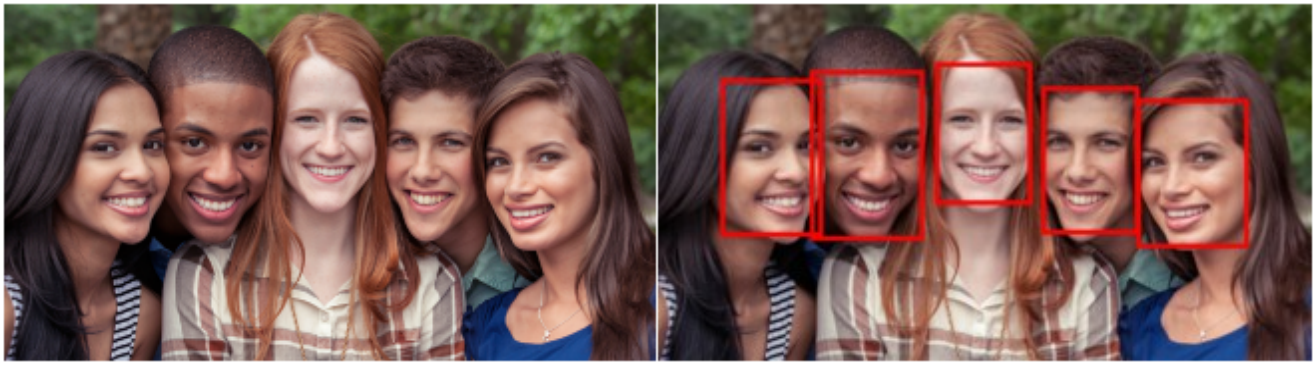
Hence, our dashboard can visualize all algorithms presented easily.



Instance Segmentation using Mask RCNN



Object Detection using RetinaNet



Face Detection using MTCNN

Conclusion

This paper presents a novel method for a learning technique specifically designed to teach and help visualize all primary CV techniques to newcomers in the field of Machine Learning. Our dashboard presents an easy and intuitive User Interface (UI) that interacts with students, provides them with resources, and showcases demos for all algorithms presented in this paper.

References

1. The Role of Visual Learning in Improving Students' High-Order Thinking Skills - Williams 2009
2. Torchvision Documentation - https://pytorch.org/vision/stable/models.html
3. Facenet-pytorch Documentation - https://github.com/timesler/facenet-pytorch
4. Noise2noise Documentation - https://github.com/joeylitalien/noise2noise-pytorch
5. Fast Style Transfer Documentation - https://github.com/pytorch/examples
6. SRGAN Documentation - https://github.com/twhui/SRGAN-pyTorch
7. Streamlit Documentation - https://docs.streamlit.io/
8. Streamlit Carousel Documentation - https://github.com/DenizDogan92/Streamlit-Image-Carousel