İTÜ

09.01.2025

Istanbul Technical University
Computer Engineering Department
BLG 453E - Computer Vision - Fall 24/25
Assignment IV
Due: 27.01.2025, 23.59

Notes

- You should do your own work! . Cheating is highly discouraged.
- You **cannot** use built-in Python, Numpy, and OpenCV functions until otherwise stated.
- You should write your codes in Python.
- **A well-written report is required, and the codes will be expected to be well-commented**.
- For your questions, do not hesitate to reach Res. Asst. Ziya Ata Yazıcı (yaziciz21@itu.edu.tr) and Res. Asst. Tuğçe Temel (temel21@itu.edu.tr).

Q1 (20 pts) - Region-Growing

The region-growing algorithm is a fundamental image segmentation technique used in computer vision and image processing. It begins with one or more seed points, typically selected manually or automatically, and groups neighboring pixels based on predefined criteria such as intensity, color, or texture similarity. The algorithm expands the region iteratively by adding adjacent pixels that satisfy the similarity condition, stopping when no more pixels meet the criteria.

Develop a region-growing method for image segmentation. Select a **single** seed point and segment the **lion** in the provided *lion.jpg* image (Figure 1). Annotate the selected seed location with a red dot. Repeat the steps for three seed locations and showcase the resulting segmentation results. **Try to extract the lion as much as possible, removing the unrelated regions.**

BONUS: Showcase three different scenarios when **three** seed points are selected. Report the performance differences by reporting the differences between the created segmentation outputs.

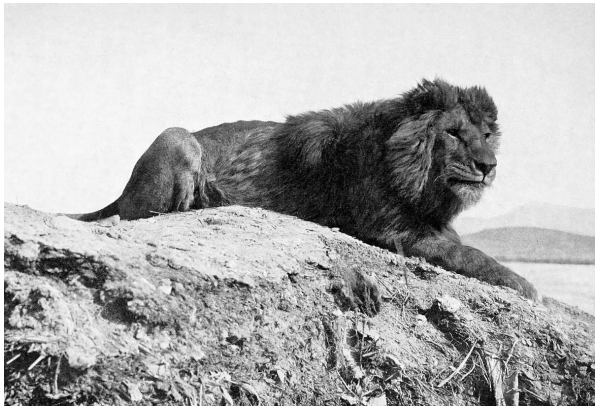

Figure 1: The Lion image to be segmented.

Q2 (20 pts) - Towards Synthetics 🤖

In this assignment, you will be experimenting with the State-of-the-Art diffusion models! Diffusion models are a type of deep learning model used to generate data, like images (Figure 2), video, or audio. They work by starting with random noise and gradually "denoising" it step-by-step until a clear output is produced. They are capable of generating hyperrealistic images and videos!

In this task, you will perform hands-on experiments on text-to-image, text-to-video, image-to-image, and image-to-video models supported on Hugging Face 🤗.

For your ease, you can use the following models from Hugging Face 🤗.

- Text-to-Image: "stable-diffusion-v1-5/stable-diffusion-v1-5"
- Text-to-Video: "cerspense/zeroscope_v2_576w"
- Image-to-Image: "stable-diffusion-v1-5/stable-diffusion-v1-5"
- Image-to-Video: "stabilityai/stable-video-diffusion-img2vid-xt"

Since diffusion models are best utilized in the graphical processing units (GPU), you are advised to use Google Colab (T4 GPU runtime). Depending on the number of frames and resolution you would like to generate, the processing time may take a few minutes for video generation.

If you would like to experiment with different/advanced models, feel free to use them (as long as you access them via free-to-use APIs! ChatGPT and Gemini (Imagen 3) models do not meet the requirement). The given models were selected due to their quick usability. Other models may require you to fill out a usage form and gain free access afterward.

Come up with interesting prompts and showcase your best work!

Figure 2: A generated image from the prompt of "A parrot driving a sports car".

Q3 (30 pts) - Optical Flow

Optical flow is a valuable technique for motion analysis in computer vision. It describes how pixels move between frames in a video. It produces a vector field where each vector shows how a pixel shifts from one frame to the next. This technique can be used for motion detection, object tracking, etc.

Its applications in robotics, augmented reality, and video processing highlight its importance in modern technology. For this part, we are interested in the robotic application.

Figure 3: The BAXTER

The Baxter is a chef-robot (Figure 3). Baxter records the preparing-cooking stages using the head camera. Example records are in Figure 4. In this part, you are expected to track the Baxter's arm using optical flow over the video recordings.

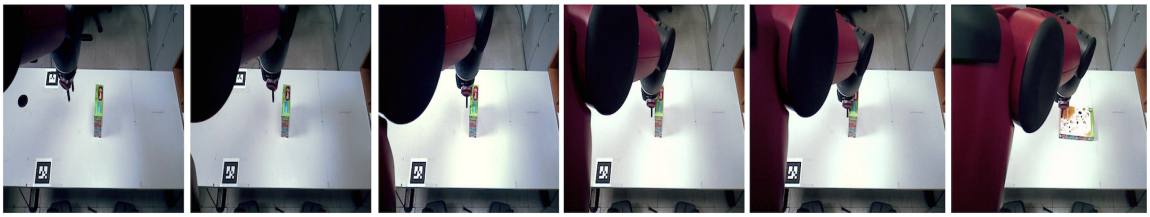

Figure 4: Sequential frames of the record.

If you can complete all the steps correctly, your outputs will look like Figure 5. **Use the opticalflow.py file to complete this task.**

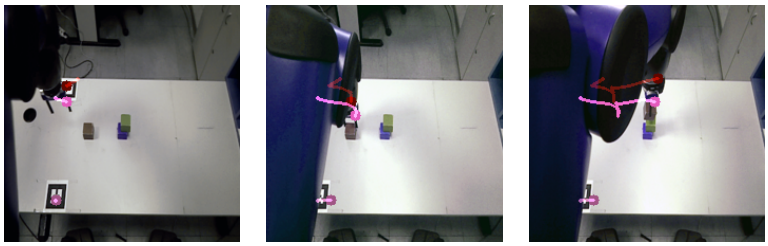

Figure 5: Result of the optical flow algorithm.

Q4 (30 pts) Principal Component Analysis

The objective of this part is to understand and apply Principal Component Analysis (PCA) and transfer learning using Python. PCA is a dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional space while preserving as much variance as possible. This method allows us to visualize a high-dimensional space in 2D or 3D. However, the performance of the dimensionality reduction depends on the quality of the feature vectors that are extracted from the data (i.e., images). As one can extract the features by simply **concatenating the RGB pixels** to create a vector, other approaches could be **using the pre-trained image encoders** to extract more meaningful features.

Off-the-shelf image encoders are usually provided with pre-trained weights that could be used for feature extraction. However, fine-tuning a model on a new dataset via transfer learning may result in a better representation capability. Transfer learning is a machine learning technique where a model trained on one task is reused for a different but related task. Instead of training a model from scratch, we start with a pre-trained model and fine-tune it for a new task or dataset. This approach is especially useful when the new task has limited data.

In this task, you are expected to create a feature space of a dataset of **fruits and vegetables**[1] (Figure 6). You are expected to perform two main tasks for this question: **1) Basic feature extraction and PCA visualization, 2) Feature extraction via a fine-tuned image encoder model and PCA visualization.**

Figure 6: Sample images from the given dataset.

For the first task 1), you should create a basic vector representation for each image in the **test split**. This could be performed by concatenating each pixel intensity in an image as Equation 1, where I_{concat} is the vector to be used for each image.

$$I_{\text{concat}} = \text{concat}(I_1, I_2, \ldots, I_n), \quad \text{where } I_i \in R^{w \times h \times 3} \tag{1}$$

For the second task 2), you should create more meaningful features via a fine-tuned model. For this model, you should apply transfer learning by following the instructions below. For your ease, we prepared a transfer learning file named **transferlearning.py**. To complete the steps, you are expected to use the given script.

- Complete the provided **transferlearning.py** file.
- Train the pre-trained ResNet-50 model using transfer learning on the **train split** of the given dataset for a classification task.
- Monitor the validation accuracy on the **validation split** and save the best-performing model during training.
- Extract the image features from the **test split** via the fine-tuned model. Image features can be retrieved from the first layer before the classifier layer (fc layer).
- Apply PCA on the extracted features to reduce dimensionality (tasks 1 and 2).
- Visualize the PCA results and cluster the feature groups. Show a few sample images from each cluster and analyze the results of both tasks (1 and 2).

[1] https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition