DAT301m Lab 2: Object Detection, Semantic Segmentation and Transfer Learning

Overview

In this assignment, you will work with object detection, semantic segmentation, and transfer learning using TensorFlow and Keras. You will learn to:

Learning Objectives

- Apply data preprocessing and augmentation techniques for detection and segmentation tasks.
- Implement object detection and segmentation architectures with and without using pre-trained models and fine-tuning.
- Leverage multitask transfer learning to perform detection and segmentation jointly.

Requirements: Jupyter Notebook or Python Scripts, using Tensorflow and Tensorflow Datasets.

The **Oxford-IIIT Pet Dataset Dataset** is a dataset of multiple different breeds of cats and dogs. It contains 37 categories with roughly 200 images for each class. The labels that are capitalized determines the cat breeds, and the labels that are lower-cased determined the dog breeds. The images have a large variations in scale, pose and lighting. All images have an associated ground truth annotation of breed, head ROI, and pixel level trimap segmentation.

This dataset can be directly downloaded using **Tensorflow Datasets** library. To do so, install `tensorflow_datasets` package if you are running locally (Colab and Kaggle notebooks have this library readily installed). Then, `import tensorflow_datasets as tfds` and simply load the dataset by calling `tfds.load('oxford_iiit_pet:4.0.0')`. There are other arguments to consider as well. Make use of them.

Note: We will use version 4.0.0 of this dataset, as only this version has the bounding boxes feature. A full list of features can be found here: https://www.tensorflow.org/datasets/catalog/oxford_iiit_pet

Task 1: Dataset Exploration and Preprocessing (2 points)

There are 3,680 images in `train` and 3,669 images in `test`. However, in the `test` split, the bounding boxes are empty. Therefore, to compare the true and predicted bounding boxes, we can randomly split 20% of the training data as the `val` split.

- Download the dataset and split the `train` into `train` and `val`.
- Implement proper data preprocessing, augmentation and feature extraction techniques (resizing, normalizing, one-hot encoding segmentation masks, transforming bounding box formats, alpha matting...) when needed.
- Analyze and visualize the dataset distribution and characteristics with interesting and insightful visualizations.

For Task 2 and Task 3, we will build a model with a pre-trained backbone. Some sample backbones include ResNet50, MobileNetV2, EfficientNet, YOLOv8, EfficientDet, U-Net, DeepLabV3... Include any custom detection or segmentation heads as needed.

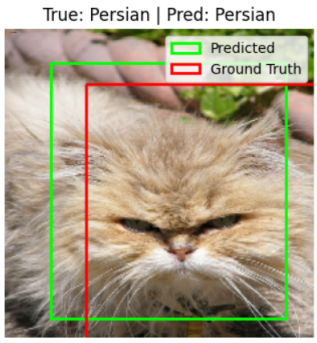
In each task, we will build 2 versions of the model with the same backbone and architecture:

- **Without pre-trained weights:** The entire model, including the backbone, will be trained from scratch using randomly initialized weights.
- **With pre-trained weights:** Apply transfer learning by initializing the backbone with weights pre-trained on a large dataset (such as ImageNet), allowing the model to leverage learned features and improve training efficiency.

Task 2: Object Detection (3 points)

First, we will build 2 models to detect the breeds by predicting bounding boxes around them.

- **Model Variants:** Implement 2 versions of the object detection models, with and without pre-trained weights. You may choose to freeze the backbone or fine-tune it depending on observed performance.
- **Training:** Implement a full training loop using an appropriate loss function and optimizer. Train your models on the `train` and `val` splits for a sufficiently large number of epochs (50–100) until convergence. Apply early stopping and learning rate scheduling to improve training stability. Plot the training and validation loss/accuracy curves for each model.
- **Performance Comparison:** Evaluate both models using key metrics such as:
  - Final validation accuracy (if using detection head in your model)
  - Final bounding box Mean Absolute Error (MAE)
  - Number of epochs until early stopping
  - Total training time
- **Visualization:** Show qualitative results by overlaying predicted and true bounding boxes on sample images from the `val` sets. Compute and display the Intersection over Union (IoU) score for these examples. Example visualization:



- **Test Evaluation:** Run inference on the `test` split and visualize sample predicted bounding boxes.

Answer the following questions (Q1) in your report: Which model performed better overall? Were there any signs of overfitting or underfitting in either model? How did you address them? How did the bounding box MAE and IoU values compare across models?

Task 3: Semantic Segmentation (3 points)

Next, we will build 2 models to perform semantic segmentation of the breed regions in images.

- **Model Variants:** Implement 2 versions of the semantic segmentation models, with and without pre-trained weights. You may choose to freeze the backbone or fine-tune it depending on observed performance.
- **Training:** Implement a full training loop using an appropriate loss function and optimizer. Train your models on the `train` and `val` splits for a sufficiently large number of epochs (50–100) until convergence. Apply early stopping and learning rate scheduling to improve training stability. Plot the training and validation loss/accuracy curves for each model.
- **Performance Comparison:** Evaluate both models using key metrics such as:
  - Final validation accuracy
  - Number of epochs until early stopping
  - Total training time
- **Visualization:** Display example results, including:
  - Original image
  - Ground truth segmentation mask
  - Predicted segmentation mask

  Example visualization:

- **Test Evaluation:** Run inference on the `test` split and visualize predicted bounding boxes.

Answer the following questions (Q2) in your report: Which model performed better overall? Were there any signs of overfitting or underfitting in either model? How did you address them? If you tried multiple loss functions/optimizers, which one worked best?

Task 4 (Advanced): Multitask Transfer Learning (2 points)

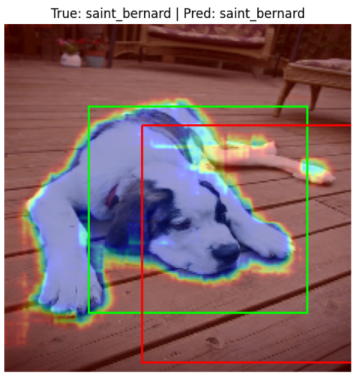
Finally, we will build a multitask model that performs both object detection and segmentation simultaneously. The goal is to leverage a shared backbone to improve efficiency and potentially enhance performance on both tasks.

- **Model Architecture:** Design a multitask model consisting of:
  - A shared backbone network.
  - A detection head that predicts bounding boxes.
  - A segmentation head that outputs segmentation masks.
- **Training:** Implement a joint training loop that optimizes both tasks simultaneously. Use an appropriate loss function for each task (e.g., Smooth L1 or GIoU loss for detection, cross-entropy or Dice loss for segmentation), and combine them using a weighted sum (if needed):

  $$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{det}} + \beta \cdot \mathcal{L}_{\text{seg}}$$

  Plot the training and validation loss/accuracy curves.
- **Evaluation:** Compare the multitask model's performance to the single-task transfer learning models from Tasks 2 and 3.
- **Visualization:** Show qualitative results by displaying predicted and true bounding boxes, as well as the predicted and true segmentation mask on sample images from the `val` sets.

  Example visualization (You can choose different ways to visualize the results):



- **Test Evaluation:** Run inference on the `test` split and visualize sample predicted bounding boxes and segmentation masks.

Answer the following questions (Q3) in your report: Did the multitask model achieve comparable or better performance than the individual models? How did sharing the backbone affect training speed and convergence? Would you recommend using multitask learning in this context? Why or why not?

What to submit:

1. Your well-documented Jupyter notebooks or Python scripts containing all implementations.
2. A report (2-6 pages) detailing:
    - Your approach to each task, with proper justification.
    - Analysis of results with supporting visualizations.

Your Grades will be based on completing the required tasks and submitting a report. It is mandatory and directly influences the points awarded for each task. **Failure to submit a report can result in a maximum of 30% deduction.** You can zip all files and submit a single .zip or .rar file. If you reach maximum file size when submitting your work, you can either submit a Google Drive or GitHub repo link.

Here's a breakdown of each part:

- **Task 1: Dataset Exploration and Preprocessing (2 points)**
    - Preprocessing, Augmentation, Feature Extraction – 1 pt
    - Dataset visualization and distribution analysis – 1 pt
- **Task 2: Object Detection (3 points)**
    - Model Variants – 0.5 pt
    - Training and Evaluation – 1 pt
    - Visualization and Test Evaluation – 1 pt
    - Q1 – 0.5 pt
- **Task 3: Semantic Segmentation (3 points)**
    - Model Variants – 0.5 pt
    - Training and Evaluation – 1 pt
    - Visualization and Test Evaluation – 1 pt
    - Q2 – 0.5 pt
- **Task 4 (Advanced): Multitask Transfer Learning**
    - Model Architecture – 0.5 pt
    - Training and Evaluation – 0.5 pt
    - Visualization and Test Evaluation – 0.5 pt
    - Q3 – 0.5 pt