Project 3 Guidelines: Implementation of Neural Networks for Object Detection

This project is structured to integrate neural networks in object detection. The objective is to facilitate practical comprehension and application of neural network concepts, from classification to object detection tasks.

Part 1: Report on Classification Model Backbone (20 Points)

Backbone Architecture Selection

- Aggregate the IDs of all group members, and utilize the last digit to select your backbone architecture:
  - 0-3: ResNet18
  - 4-6: VGG16
  - 7-9: MobileNet V3

Requirements

1. Document the process of summing the IDs and selecting the architecture. A 10 point deduction will apply for incorrect ID sum calculations.
2. Present inference results from the pre-trained network to demonstrate its current capabilities.
3. Conduct a detailed analysis (minimum two pages) on the chosen architecture and inference results, focusing on unfamiliar blocks or components not covered in course lectures. Utilize original research papers and additional resources for this analysis.

Part 2: Object Detection Model Implementation (80 Points)

In this section, you are tasked with developing an object detection system, leveraging a preselected classification model as the foundational backbone. This endeavor takes us beyond the confines of our course syllabus, venturing into the application of transfer learning from established classification frameworks to the nuanced domain of object detection. A crucial aspect of this project involves your engagement with independent research to comprehend and integrate each essential component of an object detection model.

What is a backbone?

Before delving into the specific tasks, it's imperative to understand the significance of utilizing a pretrained classification model as the backbone for your object detection system. Pretrained classification models, such as ResNet, VGG, or MobileNet, have been extensively trained on vast datasets like ImageNet, enabling them to develop a rich, hierarchical representation of visual features. This foundational knowledge facilitates the model's ability to discern complex patterns and features in new images, a capability that is immensely beneficial for the task of object detection. By employing these pretrained networks, you capitalize on their learned representations, effectively shortcutting the need for training a model from scratch. This approach not only saves considerable computational resources and time but also enhances the model's detection capabilities, especially when dealing with limited data for the specific task at hand. The

transfer of learned features from classification to detection tasks is a testament to the versatility and efficiency of deep learning models, allowing for sophisticated applications such as accurate and robust object detection.

Model Construction and Evaluation

1. **Architecture Development**: Construct an object detection architecture utilizing the selected classification backbone. This phase includes several critical steps, each necessitating prior investigation and detailed justification:

    - **Axis-Aligned Bounding Boxes (AABB) for Multiple Labels**: Investigate AABB implementation in object detection, focusing on detecting at least two distinct labels, each with its respective AABB. Provide a mathematical rationale for using AABB in object detection and justify its applicability for your model, especially for handling multiple labels.

    - **Loss Functions and Metrics**: Explore various loss functions and metrics relevant to object detection, particularly those that can effectively manage multiple labels and their AABBs. Select options that best suit the specifics of your dataset and model objectives, justifying your choices based on the theoretical advantages they provide for enhancing model accuracy and reliability.

    - **Data Partitioning**: Examine strategies for splitting your dataset into training, validation, and testing sets. Justify your chosen method in terms of its impact on the model's ability to detect multiple labels and AABBs accurately. Discuss how this strategy will influence model generalization and performance evaluation.

    - **Data Augmentation**: Investigate and implement data augmentation techniques suitable for multi-label object detection, using `albumentations` or `torchvision`. Detail the augmentations chosen and discuss their expected impact on model robustness and generalization.

    - **TensorBoard Monitoring**: Use `tensorboard` to monitor training and validation progress, focusing on metrics relevant to multiple label detection and AABB accuracy. Discuss the importance of these visualizations in optimizing your model and ensuring balanced performance across all labels.

2. **Dataset Selection**: Select an appropriate dataset for object detection (a good place for it can be Roboflow Public Datasets). Choose a dataset that meets the project's requirement of detecting at least two labels and their AABBs. Justify your selection based on the dataset's relevance, complexity, and ability to challenge and validate your model's effectiveness.

3. **Model Training**: After constructing your model and selecting a dataset, proceed to train your object detection system. This stage is crucial for fine-tuning the model's parameters to effectively recognize and localize the objects of interest within the dataset. Provide a detailed account of your training process, including:

    - Configuration of training parameters (e.g., learning rate, batch size).
    - Strategies employed to avoid overfitting and ensure generalization.
    - Analysis of training and validation loss curves to monitor the learning process.
    - Adjustments made based on the performance metrics observed during training.

4. **Inference on External Video**: Test your model on a video not included in the training set to evaluate its real-world applicability. This step is essential for assessing the model's generalization capabilities, particularly in detecting multiple labels and accurately computing their AABBs. Document your methodology, outcomes, and any challenges encountered, offering insights into the model's operational performance.

Each section of your report should meticulously document your research, decision-making process, and implementation approach. This comprehensive methodology will not only showcase your technical expertise but also your capacity to critically engage with novel information and apply it to practical scenarios.

Part 3: Oriented Bounding Box Implementation (10 Bonus Points)

This bonus section introduces the implementation of Oriented Bounding Boxes (OBBs) to your object detection model, using the selected classification backbone. OBBs provide a more accurate localization by allowing bounding boxes to rotate, closely fitting the object's actual orientation and shape. This advanced feature is crucial for objects with varied poses and angles, enhancing detection precision beyond the capabilities of Axis-Aligned Bounding Boxes (AABBs).

Implementation Overview

- **Study OBB Concepts**: Briefly research the fundamentals of OBBs, focusing on their geometric properties and how they offer a more precise encapsulation of objects by including orientation.
- **Model Adaptation**: Similar to Part 2, modify your detection model to predict OBB parameters: center coordinates, dimensions, and rotation angle. This involves adjusting the architecture to output these additional details, akin to how you structured your model for AABB predictions.
- **Loss Function and Metrics**: Adapt your loss functions and evaluation metrics to accommodate the complexities of OBBs, especially for accurately predicting and evaluating the orientation of objects.
- **Training and Evaluation**: Train your model on an appropriate dataset, ensuring it includes orientation annotations. The training process and evaluation should mirror the approach taken in Part 2, with adjustments made for the unique aspects of OBB prediction.

This concise task aims to enhance your model's detection capabilities by incorporating orientation, a critical step for achieving higher precision in object detection.

Submission Guidelines

- Form groups of up to two members.
- Compile results in a `.zip` file named `PROJ3_NAME1_ID1_NAME2_ID2.zip`, containing:
    - A comprehensive summary of the project, including methodologies, assumptions, successes, and limitations of the algorithm.
    - Source code in `.py` format.
    - Output videos in a supported format.
    - Include the final result video either as a file within the `.zip` or provide a YouTube link in the report (ensure visibility at both the beginning and end of the report).

Deadline

- Submit by the day preceding the start of the next semester.

Good Luck! Yoni Chechik