Project 3 Guidelines: Implementation of Neural Networks for Object Detection

This project is designed to extend neural network knowledge from **image classification** to **object detection**, emphasizing transfer learning, architectural reasoning, and independent research. Students are expected to go beyond course material, investigate unfamiliar concepts, and justify all design decisions.

Part 1: Report on Classification Model Backbone (10 Points)

Backbone Architecture Selection

Aggregate the IDs of all group members and use the final digit of the digit-sum to select the backbone architecture:

- 0–3: ResNet18
- 4–6: VGG16
- 7–9: MobileNet V3

Important: An incorrect digit-sum calculation will result in a 10-point deduction.

Example: id1 = 12345678, id2 = 87654321 Digit sum = 72 → 7 + 2 = 9 → MobileNet V3

Requirements

1. **ID Calculation Documentation** Clearly document the full digit summation process and the resulting backbone choice.
2. **Inference Demonstration** Run inference using the pretrained backbone on several random images and present the classification results.
3. **Architecture Analysis Report** Write a detailed report (minimum two pages) analyzing:
    - The overall architecture of the selected model
    - Key building blocks not covered in course lectures
    - Design motivations behind these components
    - Observed strengths and limitations from inference results

   Use original research papers and additional resources.

Part 2: Single Class, Single Object Per Frame (40 Points)

Objective

Develop a **basic object detection model** for a dataset containing **a single object class**, where **each image contains exactly one object**. This part introduces the transition from classification to detection through transfer learning.

Background: Classification Backbone as a Detector

Pretrained classification models such as ResNet, VGG, and MobileNet are trained on large-scale datasets (e.g. ImageNet) and learn rich hierarchical feature representations. In this part, you will leverage such a model as a **backbone**, reusing its learned features while adapting the network to predict object location instead of a class label. This approach reduces training cost and improves generalization when data is limited.

Requirements

1. **Model Construction** Build an object detection model using:
   - The selected pretrained backbone
   - Only basic PyTorch building blocks (e.g. Conv2d, Linear, ReLU, pooling)

   You must explain how the classification model is adapted for detection.
2. **Bounding Box Representation** Research and implement an axis-aligned bounding box (AABB) representation suitable for a single-object scenario.
3. **Loss Functions and Metrics** Research and select appropriate loss function(s) and evaluation metric(s) for this task. Justify your choices theoretically and practically.
4. **Dataset Selection and Preparation** Select a dataset suitable for single-class, single-object detection (e.g. from Roboflow Public Datasets). Justify the dataset choice and describe how it is split into training, validation, and test sets.
5. **Model Training and Analysis** Train the model and provide a detailed analysis of:

- Training configuration (learning rate, batch size, etc.)
- Overfitting prevention strategies
- Training and validation behavior over time
- Adjustments made during experimentation

6. **Inference on External Video** Run inference on a video **not included in the dataset** and analyze results.

   Important:

   - The video must be non-trivial (background variation, motion, scale changes).
   - Easy videos (static object, uniform background) will result in point deductions.
   - Partial or unstable detection will also reduce the score.

Part 3: Multi-Class, Multi-Object Detection with Fixed Capacity (50 Points)

Objective

Extend the detector to handle **multiple object classes** and **multiple objects per frame**, while maintaining a **fixed upper bound on the number of detectable objects**. This part evaluates your ability to design, justify, and analyze a more complex detection system.

Setup Constraints

- Maximum number of objects per image: **3**
- Minimum number of classes: **2**

Requirements

1. **Dataset Extension** Use or construct a dataset that includes:

   - Multiple object classes
   - Images containing multiple objects in the same frame

   Justify the dataset choice and any preprocessing or annotation adaptations.

2. **Model Architecture Design** Modify your model to support multiple objects and multiple classes under the fixed-capacity constraint.

   You must clearly explain:

   - How predictions are structured
   - How multiple objects are represented
   - What assumptions your design makes
   - What limitations follow from those assumptions

3. **Loss Function and Training Adaptation** Adapt your loss functions and training procedure to support multi-object, multi-class detection. Explain how your approach handles unused predictions and avoids degenerate solutions.

4. **Evaluation Metrics** Select and justify appropriate evaluation metrics for this scenario (e.g. IoU, class-wise accuracy, detection performance across frames).

5. **Training and Validation Analysis** Train the model and analyze:

   - Learning dynamics
   - Stability
   - Failure cases
   - Differences compared to the single-object setup

6. **Inference on External Video** Run inference on a new external video containing:

   - Multiple objects
   - At least two classes **in the same frame**

   Visualize and critically analyze the results.

   Note: Failure to demonstrate multiple classes in the same frame will result in point deductions.

Common Requirements for Parts 2 and 3

The following apply to **both Part 2 and Part 3**:

1. **Dataset Splitting** Data must be split into training, validation, and test sets. The external video must not appear in any split.

2. **Data Augmentation** Implement data augmentation suitable for object detection. Bounding boxes must be handled correctly. All augmentations must be justified.

3. **TensorBoard Monitoring** Use TensorBoard to monitor training and validation progress. Select meaningful quantities and explain their importance.
4. **External Video Evaluation** External video inference is mandatory. Weak or trivial demonstrations will result in significant point deductions.
5. **Discussion and Limitations** Discuss failure cases, limitations, and possible future improvements.

Submission Guidelines

- Groups of up to two members.
- Submit a `.zip` file named:

```
PROJ3_NAME1_ID1_NAME2_ID2.zip
```

The zip must include:

- A comprehensive project report
- Source code in `.py` format
- Output videos or a YouTube link (visible at the beginning and end of the report)

Good Luck! Yoni Chechik