Mini-Challenge Proposal DLBS

 Where is Santa? 

Autum Semester 2025/26

Sofia Alcaide
Katarina Fatur

1 Problem Statement

Can we find Santa? We're feeling a little Grinch-y and want to stop Santa before he delivers gifts! Joking aside, this object detection challenge will explore the following questions:

- How does YOLOv8 perform on a smaller, mixed-style dataset where the task is to detect Santa in images? The dataset will include both real-life and cartoon images.
- Can a pretrained model achieve an IoU above 0.5? We aim to see if the model can correctly detect Santa Claus figure, ensuring high recall on unseen test images.
- How can we handle challenges like inter-class similarity (confounding red/white features) and intra-class variability (varied poses and styles)?
- Where will the model struggle?
    - Inter-class confusion: Will it detect other red/white objects as Santa?
    - Intra-class variability: How well will the model handle different poses, unusual angles, and partial crops of Santa?
    - Domain shift: Will the model's performance drop when detecting cartoon Santas compared to real-life Santas?

Hypotheses:

H_1: A pretrained YOLOv8 model, when fine-tuned on a small, mixed-style Santa dataset, will achieve at least an IoU ≥ 0.5 and recall ≥ 0.7 on unseen test images containing real-life and cartoon Santas.

H_2: The main challenge will come from high intra-class variability (different styles, poses, viewpoints) and inter-class similarity (red/white objects resembling Santa).

2 Data

The dataset consists of ~600 annotated images, which is on the small side for object detection.

To teach variability of the Santa concept, we include a variety of styles: real-life photos, illustrations, Santa in different positions or partially occluded, with reindeer, gifts, sleigh. Santa has strong color cues (red & white). To reduce false positives we need to include hard negative images. These include confusing features for the model like men with beards and red suits that are NOT Santa, and Christmas decorations.

Quick inspection of data revealed some noisy labelling and duplicate images. Some images also repeat in test and train sets, but all these issues can be due to the small dataset size resolved manually.

3 Methods

In a pretrained YOLOv8 most features (edges, colors, shapes) are already learned, so we fine-tune only the final layers (detection head) on our Santa dataset. As evaluation metrics go, since we're detecting a single class (Santa), using IoU ≥ 0.5 is essential to evaluate if the model is detecting correctly. mAP at 0.5 provides an overall evaluation of detection performance. Also, since it's more important not to miss any Santa, recall should be prioritized.

We experiment with the effect of data size, hyperparameters (learning rate variations, batch size and number of epochs), and also test for robustness (how the model performs on blurred or occluded images).

Workflow:

1. **Data inspection, annotation, cleaning, preprocessing**: we check for missing, incorrect, or overlapping bounding boxes, adapt dataset for YOLO format, we standardize images (500x500px), apply normalization, split train/val/test
2. **Data augmentation**: we apply transformations (flips, rotations, scaling, contrast changes, random color jitter, random crop, mixup, cutout, mosaic …) to expand the dataset and make it robust to variations
3. **Baseline model**: a small ConvNet for object detection to compare its performance to YOLOv8
4. **YOLOv8**: use pre-trained weights to speed up convergence
5. **Memorization/overfitting**: take 20 images, freeze most layers and train the final detection head only with no regularization. No dropout. Train until 100% training metrics. Monitor training loss, IoU, mAP. If the model overfits, it means it learns.
6. **Regularization** & **Optimization**: we monitor the validation loss and recall after each epoch. Based on these, we adjust the learning rate, dropout rate, and L2 strength. WandB tracks these metrics in real-time.
7. **Monitoring**: we use WandB to visualize training and validation loss to ensure the model is learning. We track mAP, precision, and recall per epoch. We monitor predictions at regular intervals to see how it's performing on the validation set (visualize bounding boxes on random test images).