IVP Project Report

Aditya B

April 2025

Introduction

Project 1: WOSDETC2025 - IJCNN Drone vs Bird Detection

Overview

In this project, we focused on the task of detecting drones in aerial footage, distinguishing them from birds. This work was inspired by the Drone vs Bird Detection challenge organized as part of WOSDETC2025. The primary goal was to perform a comparative analysis by fine-tuning both YOLOv10 and YOLOv11 object detection models on a large collection of video frames annotated for drone appearances.

Dataset Preparation

The dataset provided was in the form of multiple video files. We developed a custom Python script (`extractFrames.py`) using `ffmpeg` to extract every frame from these videos and save them in directories named after each video. This was done to preserve traceability between the frames and their sources.

The dataset also included annotation files in a custom text-based format, describing bounding boxes for drones per frame. We wrote a conversion script (`convertAnnToCocoFormat.py`) to transform these into COCO format JSON files. Each annotation file was processed individually, resulting in one COCO file per video.

We then merged all individual COCO JSON files into a single combined annotation file using `combinedAnnotations.py`. From this, we generated a master YOLO dataset using `convertCocoToYolo.py`, which produced per-image YOLO annotations in `.txt` format.

Directory Structuring

Once the full set of images and labels was available, we organized them into the following directory structure, adhering to YOLO requirements:

```
dataset/
 images/
    train/
    val/
    test/
 labels/
     train/
     val/
     test/
```

We used `splitDataset.py` to split the full dataset into 70% training, 15% validation, and 15% testing, ensuring reproducibility with a fixed seed.

Model Fine-tuning and Comparative Analysis

We conducted a comparative analysis between YOLOv10 and YOLOv11 models, fine-tuning both to evaluate their performance on the drone detection task. The training was performed using the `trainer.py` script with the following parameters:

- Epochs: 50
- Batch Size: 16
- Image Size: 1024x1024
- Optimizer: SGD (automatically chosen)
- Losses: Box loss, classification loss, and DFL loss
- Device: CPU (Apple M3 chip)
- Workers: 12
- AMP (Automatic Mixed Precision): Enabled

The training configuration was specified in `video_data.yaml`, which defined the dataset paths and class names.

Sanity Checks and Quality Assurance

Before training, we implemented and ran various sanity checks using `sanityCheck.py`:

- Verified that every label had a corresponding image and vice versa
- Checked that frame counts matched the number of annotations extracted
- Validated that YOLO-format annotations were non-empty and well-formed
- Ensured proper data distribution across train/val/test splits

Challenges Encountered

- The dataset was not initially in COCO or YOLO format, requiring multiple conversion scripts
- Annotation image IDs sometimes mismatched frame filenames, requiring normalization
- Training on CPU was significantly slower and required optimization through reduced batch sizes
- Memory constraints on the M3 chip necessitated careful tuning of batch sizes and image dimensions
- Comparative analysis required maintaining separate training runs and configurations for each YOLO version

Outcome

By the end of this project, we had:

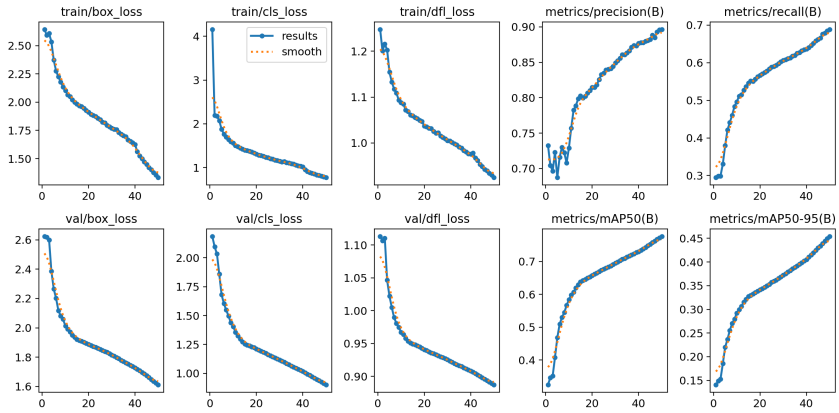
- A complete, clean YOLO-compatible dataset derived from raw video and custom annotations
- Trained YOLOv10 and YOLOv11 models fine-tuned specifically for drone detection
- A comparative analysis of both models' performance on the drone detection task
- An extensible training pipeline that can be reused for future variations of this task

This project demonstrated our ability to handle raw video datasets, convert annotations between formats, validate data integrity, and perform comparative analysis of state-of-the-art models under limited hardware constraints.

Results and Visualizations

The following figures provide a visual summary of the model training and evaluation results:

- **Training Summary Graphs:**

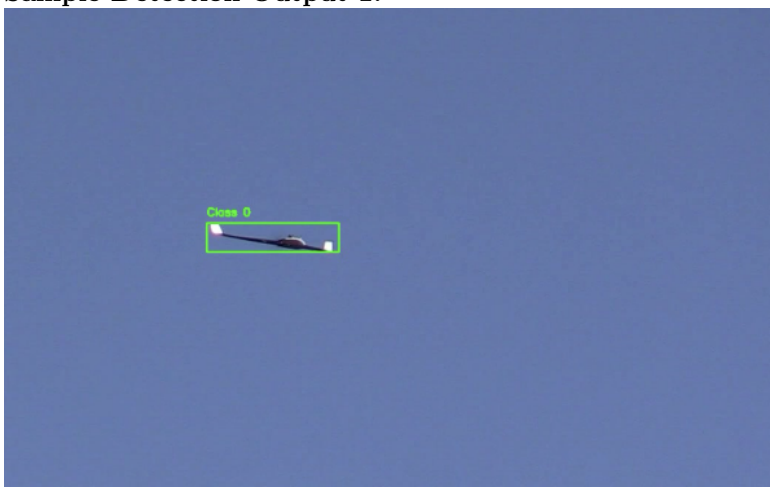


- **Final Epoch Metrics:**

```
35 epochs completed in 2.809 hours.
Optimizer stripped from runs/detect/drone_vs_bird_finetune3/weights/last.pt, 5.5MB
Optimizer stripped from runs/detect/drone_vs_bird_finetune3/weights/best.pt, 5.5MB

Validating runs/detect/drone_vs_bird_finetune3/weights/best.pt...
Ultralytics 8.3.62 🚀 Python-3.10.16 torch-2.5.1 CUDA:0 (NVIDIA RTX A5000, 24233MiB)
YOLO11n summary (fused): 238 layers, 2,582,347 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 267/267 [00:15<00:00, 17.40it/s]
                   all       8531      10272      0.898      0.688      0.777      0.454
Speed: 0.1ms preprocess, 0.5ms inference, 0.0ms loss, 0.3ms postprocess per image
Results saved to runs/detect/drone_vs_bird_finetune3
💡 Learn more at https://docs.ultralytics.com/modes/train
VS Code: view Ultralytics VS Code Extension ⚡ at https://docs.ultralytics.com/integrations/vscode
```

- **Sample Detection Output 1:**



Introduction

Project 2: NTIRE Challenge - Image Denoising

Overview

In this project, we participated in the NTIRE (New Trends in Image Restoration and Enhancement) Image Denoising Challenge. The primary objective was to effectively remove Gaussian noise ($\sigma = 50$) from 800 high-resolution images (2K). Our goal was to evaluate and compare several state-of-the-art methodologies including DnCNN, IPT, and IPTV2.

Dataset Preparation

The dataset consisted of 800 clean 2K resolution images. We artificially introduced Gaussian noise with $\sigma = 50$ to each image using a custom Python script employing the NumPy library. This step generated a noisy dataset suitable for evaluating denoising algorithms. The datasets were structured as follows:

```
dataset/
 clean_images/
 noisy_images/
 denoised_results/
     DnCNN/
     IPT/
     IPTV2/
```

Denoising Methodologies

We explored three different deep learning-based denoising techniques, each with its unique approach to handling noise:

1. DnCNN (Denoising Convolutional Neural Network):

DnCNN employs a deep convolutional neural network architecture specifically designed for image denoising through residual learning. Instead of directly learning the clean image, it learns to predict the noise component, which is then subtracted from the noisy image.

- **Residual Learning:** Rather than directly estimating the clean image, DnCNN predicts the residual noise map. This approach has proven more effective as it helps the network focus specifically on noise patterns.
- **Architecture Components:**
  - Conv + ReLU for the first layer
  - Conv + BN + ReLU for hidden layers
  - Conv for the last layer to reconstruct residual noise

- **Loss Function:** Uses MSE loss between predicted and actual noise patterns
- **Advantages:**
  - Efficient training and inference
  - Good performance on Gaussian noise
  - Relatively lightweight architecture
- **Limitations:**
  - May struggle with complex noise patterns
  - Limited receptive field due to CNN architecture

2. IPT (Image Processing Transformer):
IPT represents a paradigm shift in image denoising by leveraging transformer architecture, originally designed for natural language processing, to capture global dependencies in images.

- **Architecture Design:**
  - Multi-head self-attention layers for global context
  - Encoder-decoder architecture with skip connections
  - Positional embeddings to maintain spatial information
- **Key Features:**
  - Long-range dependency modeling
  - Multi-task capability (can handle various image restoration tasks)
  - Pre-trained models available from Huawei
- **Training Strategy:**
  - Uses both supervised and self-supervised learning
  - Incorporates perceptual loss along with L1/L2 losses
- **Advantages:**
  - Superior handling of complex noise patterns
  - Better preservation of global image structure

3. IPTV2 (Image Processing Transformer V2):
IPTV2 enhances the original IPT architecture by introducing specialized spatial-channel transformer blocks and a more sophisticated training regime.

- **Architectural Innovations:**
  - U-shaped encoder-decoder with dual-path transformers

  - Separate spatial and channel attention mechanisms
  - Progressive upsampling with feature fusion

- **Advanced Loss Functions:**
  - L1 loss for pixel-wise accuracy
  - MSE loss for overall image quality
  - SOBEL loss for edge preservation
  - Perceptual loss using VGG features

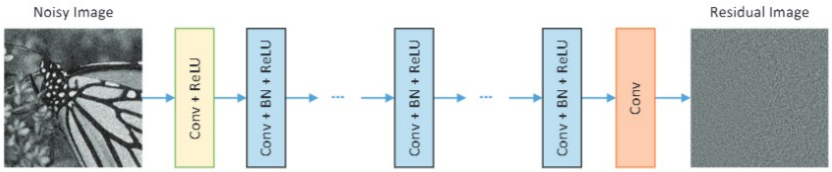
- **Training Methodology:**
  - Progressive resolution training strategy
  - Curriculum learning for noise levels
  - Cosine similarity-based attention mechanisms

- **Key Advantages:**
  - Better handling of high-frequency details
  - Improved convergence during training
  - Superior PSNR and SSIM metrics

Model Architectures

Placeholder diagrams for each model architecture:

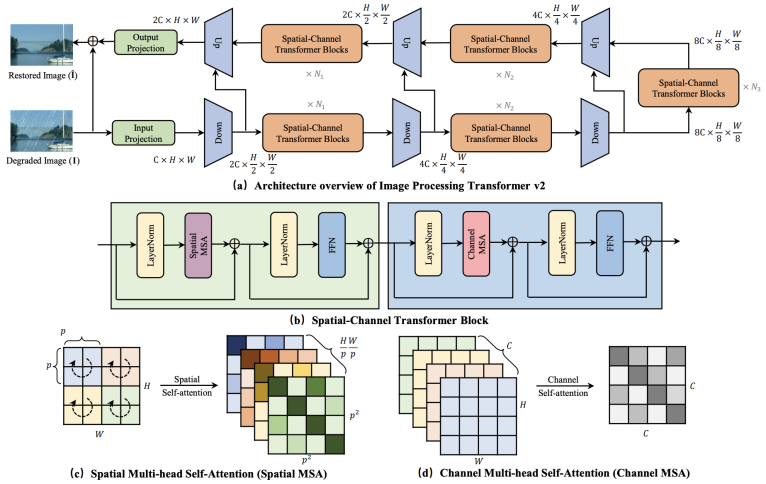
- **DnCNN Model Architecture:**



- **IPT Model Architecture:**

ipt_architecture.png

- **IPTV2 Model Architecture:**



(a) Architecture overview of Image Processing Transformer v2

(b) Spatial-Channel Transformer Block

(c) Spatial Multi-head Self-Attention (Spatial MSA)

(d) Channel Multi-head Self-Attention (Channel MSA)

Comparative Analysis

We performed a detailed comparative analysis of DnCNN, IPT, and IPTV2 based on the PSNR metric. IPTV2 consistently achieved the highest PSNR values due to its advanced transformer-based spatial-channel interaction. DnCNN provided a strong baseline performance and significantly lower computational overhead compared to transformer-based approaches.

Challenges Encountered

- **Dataset Challenges:**
  - Diverse noise patterns in real-world scenarios vs. synthetic Gaussian noise
  - Limited availability of paired clean-noisy image datasets
  - Ensuring balanced representation of different image types and noise levels
- **Model Selection and Training:**
  - Trade-off between model complexity and performance
  - Difficulty in preserving fine details while removing noise
  - Balancing multiple loss terms for optimal denoising
  - Challenge of preventing over-smoothing in transformer-based models
- **Evaluation Metrics:**
  - PSNR/SSIM metrics not always correlating with perceptual quality
  - Need for better evaluation metrics for edge preservation
  - Difficulty in quantifying texture preservation

Outcome

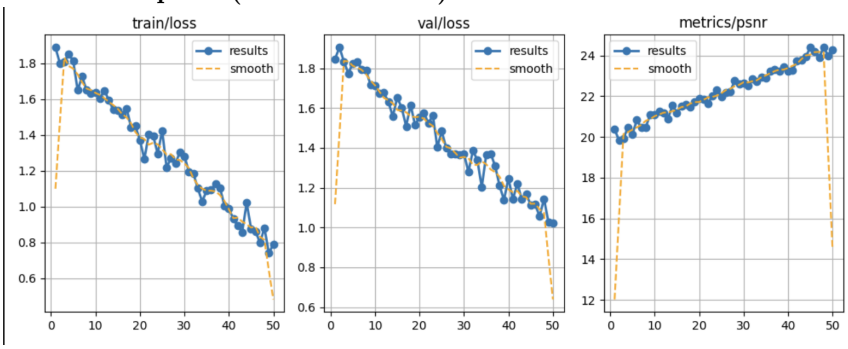
By the end of this project, we had:

- Successfully denoised the 800-image dataset using state-of-the-art denoising methodologies.
- Conducted a thorough comparative analysis between DnCNN, IPT, and IPTV2.
- Demonstrated superior performance of IPTV2 in image denoising tasks.

Results and Visualizations

The following figures illustrate performance metrics and loss convergence:

- **PSNR vs Epoch (IPTV2 models):**



- **Sample Denoised Image Comparison:**

denoised_comparison.png