UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

Group member: Dang Quang Minh - 22BI13276
Ly Tran Gia Minh- 22BI13288
Nguyen Hoang Tuan Anh - 22BI13023
Tran Thanh Nam - 22BI13328
Tran The Anh- 22BI13042

Study the YOLO for object detection in remote sensing images

Midterm Project of Introduction to Deep Learning

Major: ICT
Hanoi, 10/2024

I. Introduction

Humans are quick to identify objects, locations, and interactions when they see a picture. The human visual system is so fast and accurate that it allows us to perform complex tasks like driving with little conscious thought. Quick and accurate object recognition algorithms would pave the way for the creation of versatile, adaptable robotic systems and the capacity for computers to drive cars without specialized sensors or assistive devices to give users real-time scene information. In existing detection systems, classifiers are repurposed for detection purposes. To detect an object, these systems apply a classifier for the object and evaluate it at various sizes and locations in a test image. Models such as deformable parts (DPM).

II. Problem which we are solving

Object detection in optical remote sensing images (RSIs) is to determine if a given aerial or satellite image contains one or more objects belonging to the class of interest and locate the position of each predicted object in the image. The term ‘object’ used in this survey refers to its generalized form, including man-made objects (e.g. vehicles, ships, buildings, etc.) that have sharp boundaries and are independent of background environment, as well as landscape objects, such as land-use/land-cover (LULC) parcels that have vague boundaries and are parts of the background environment. As a fundamental problem in the field of aerial and satellite image analysis, object detection in optical RSIs plays an important role for a wide range of applications, such as environmental monitoring, geological hazard detection, LULC mapping, geographic information system (GIS) update, precision agriculture, urban planning, etc.Object detection in optical RSIs often suffers from several increasing challenges including the large variations in the visual appearance of objects caused by viewpoint variation, occlusion, background clutter,illumination, shadow, etc., the explosive growth of RSIs in quantity and quality, and the various requirements of new application areas. To address these challenges, the topic of geospatial object detection has been extensively studied since the 1980s. The low spatial resolution of earlier satellite images (such as Landsat) would not allow the detection of separate man-made or natural objects. Therefore, researchers mostly focused on extracting the region properties from these images. With the advances of remote sensing technology, the very high resolution (VHR) satellite and aerial images have been providing us with more detailed spatial and textural information. Aside from region properties,a greater range of man-made objects become recognizable and even can be separately identified than ever before because of the increased sub-meter resolution. This opens new prospects in the field of automatic detection of geospatial objects.

III. YOLO key components and processes:

1. Model Architecture

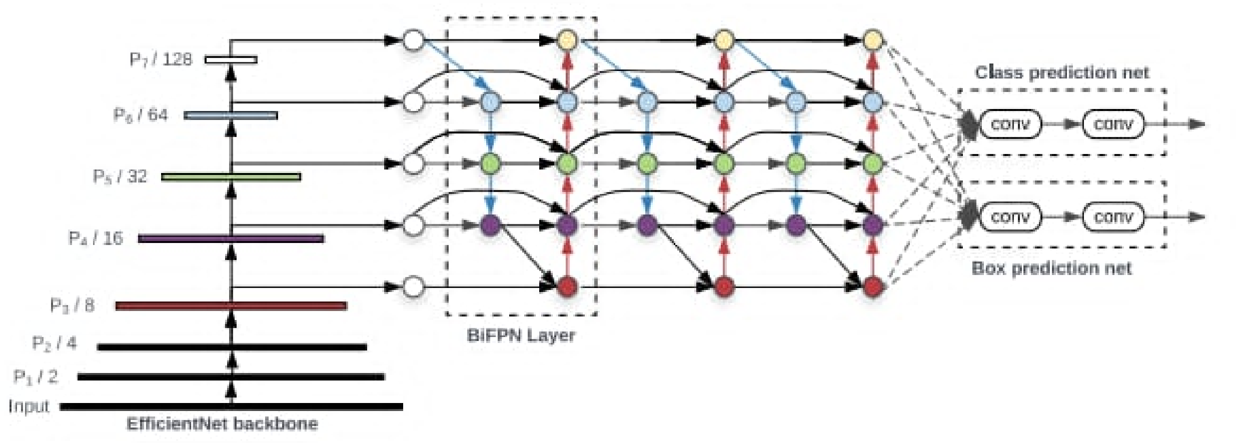
The YOLO model was the first object detector to connect the procedure of predicting bounding boxes with class labels in an end to end differentiable network.

The YOLO network consists of three main pieces.

Backbone: A convolutional neural network that aggregates and forms image features at different granularities.

Neck: A series of layers to mix and combine image features to pass them forward to prediction.

Head: Consumes features from the neck and takes box and class prediction steps.



2. Input Preparation

* Image Preprocessing: a. Resizing Dimension Reduction: High-resolution images from remote sensing can be obtained. YOLO usually needs a set input size, such as 608 x 608 pixels or 416 x 416 pixels. Preserving Features: It's crucial to keep the image's essential elements intact when scaling. To get the desired size, padding (adding borders) may need to be applied after the image has been resized to its intended dimension.

b. The process of normalization

Scaling Pixel Values: To aid in model convergence during training, pixel values—which are typically in the range of 0-255—are normalized to a range, which is often 0-1.

Mean Subtraction: To help center the data, photographs may occasionally be standardized by subtracting the mean and dividing by the dataset's standard deviation.

c. Augmenting Data

Methods Employed: Rotation, flipping, scaling, cropping, and noise addition are a few examples of augmentation techniques. These introduce unpredictability into the training data, which improves the model's capacity to generalize.

Particular Points to Remember: Making ensuring augmentations don't alter objects of interest in remote sensing photos is crucial since misrepresentation could result in inaccurate training signals.

d. Dataset for input

This collection comprises 693 tripod-taken images of chess pieces that are left-tilted. Of them, there are 606 images in the training set (or approximately 87.47%), 58 images in the validation set (or approximately 8.37%), and 29 images in the test set (or approximately 4.19%). The bounding boxes in each image are identified by the titles of the black and white pieces, which are the king, queen, bishop, knight, rook, and pawn. comprising 2894 labels and 292 separate images that have all been scaled to 416 by 416 pixels. The dataset's ongoing picture process makes it ideal for machine learning applications.

3. Grid Division

There are S×SS \times SS×S grids throughout the image. For objects whose centers fall inside a grid cell, the cell's bounding boxes and class probabilities must be predicted.

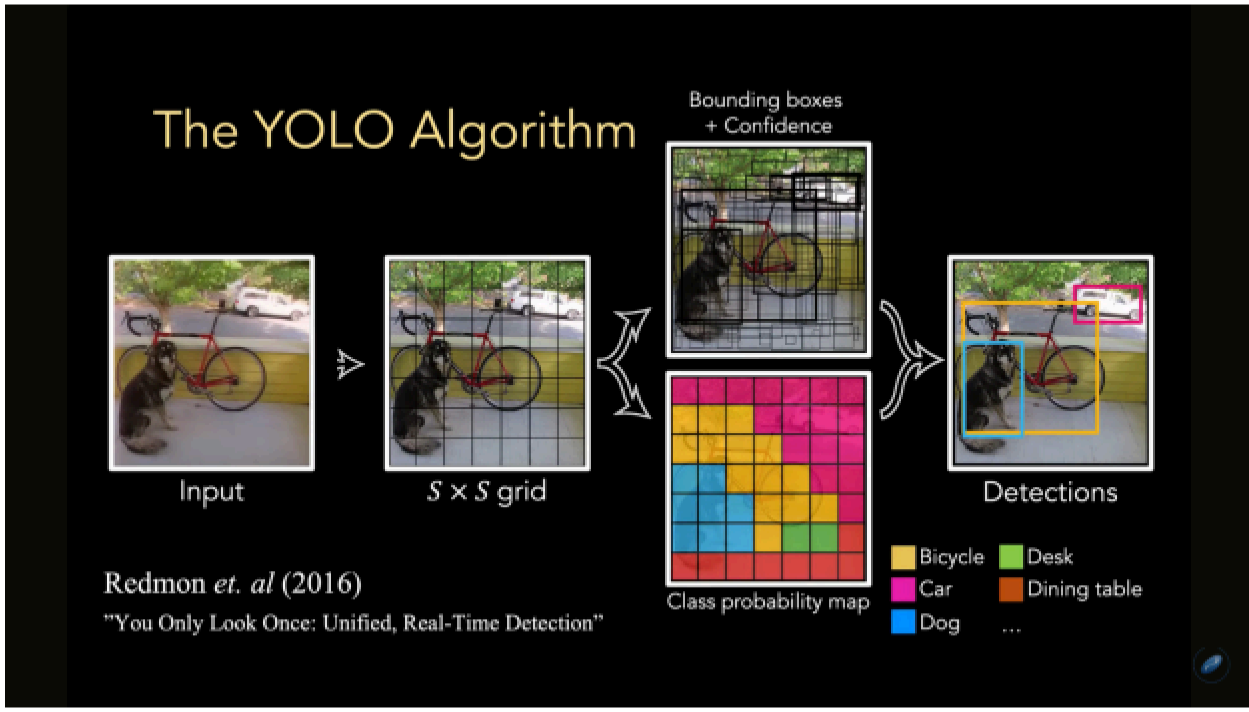
Grid Dimensions (S×SS \times SS×S)

S definition: The size of the grid The parameter SSS controls the number of grid cells into which the image is divided. SSS is typically set to 7, 13, or larger, depending on the supplied image's resolution.

Effect of Size: A bigger SSS (e.g., 13x13) offers more granularity, which is helpful for spotting smaller objects, but a smaller SSS (e.g., 7x7) makes fewer predictions and may lose detail.

b. Accountability of Cells

Bounding Box Prediction: The task assigned to each grid cell is to estimate the number of



bounding boxes.

4. Bounding Box Prediction

Bounding boxes are predicted by each grid cell in a predetermined number. The output for each box includes the class probabilities, confidence score, and coordinates (x, y, width, and height).

The accuracy and probability that a box contains an object are both reflected in the confidence score.

a. Coordinates of Bounding Box

Format: Multiple bounding boxes are predicted by each grid cell, which are commonly shown as:

The box's center coordinates in relation to the grid cell are (x, y). Usually, these coordinates are normalized to lie in the interval 0–1.

(w,h) The box's width and height, which are normally standardized to the size of the full image as well.

Compute:

The center coordinates are modified in accordance with the grid position of the cell:

where σ is the sigmoid and cx, cy are the cell's coordinates

5. Training on Remote Sensing Data

Dataset Preparation: Training requires annotated datasets with bounding boxes and class labels specific to remote sensing contexts, like buildings, roads, or vegetation.

Data Collection:

Gather high-quality remote sensing images from sources like satellites, aerial drones, or UAVs (Unmanned Aerial Vehicles).

Ensure that the dataset covers various geographical areas, seasonal variations, and times of day to enhance generalization.

Annotation:

Manually annotate images to create bounding boxes around objects of interest (e.g., buildings, vehicles, roads).

Use tools like LabelIng, VGG Image Annotator, or other specialized annotation software.

Ensure annotations are consistent and accurately reflect the object classes and locations.

Dataset Split:

Divide the dataset into training, validation, and test sets. A common split might be 70% training, 15% validation, and 15% testing.

The validation set is crucial for tuning hyperparameters, while the test set is reserved for final performance evaluation.

Data Augmentation

Techniques:

Geometric Transformations: Random rotations, flips, scaling, and translations to create variations of images.

Color Augmentation: Adjusting brightness, contrast, and saturation to simulate different lighting conditions.

Cropping: Randomly cropping sections of images to focus on different parts and increase diversity.

Noise Injection: Adding Gaussian noise or other types of noise to make the model robust against variations.

Hyperparameter Tuning

Learning Rate:

- Select an appropriate learning rate to ensure stable convergence during training. You may start with a common value like 0.001 and adjust based on performance.

Batch Size:

- Choose a batch size that fits into memory constraints while providing stable gradient estimates. Smaller batch sizes can offer more variability in training, while larger sizes provide smoother gradients.

Epochs:

- Determine the number of training epochs. Monitor training and validation losses to avoid overfitting.

Anchor Boxes:

- Define anchor boxes suitable for the specific objects in the dataset. The sizes and aspect ratios of these boxes should be representative of the target objects (e.g., buildings may have a different aspect ratio than vehicles).

Training the Model

Model Architecture:

Select the appropriate YOLO version (YOLOv3, YOLOv4, YOLOv5, etc.) based on your application needs, computational resources, and desired accuracy.

Transfer Learning:

Utilize transfer learning by starting with pre-trained weights on a similar dataset (like COCO or VOC) and fine-tuning on your remote sensing data. This can significantly reduce training time and improve performance.

Training Process:

Use a suitable loss function (e.g., mean squared error for bounding box coordinates and cross-entropy for class probabilities).

Implement techniques like early stopping to halt training when validation performance begins to degrade.

Monitoring Training:

Track performance metrics like loss, precision, recall, and mAP during training. Visualization tools like TensorBoard can be helpful for monitoring.

Evaluation on Validation Set

Validation Metrics:

- After training, evaluate the model on the validation set to adjust hyperparameters and improve performance.

- Analyze metrics such as mAP, precision, and recall to ensure the model is effectively detecting and classifying objects.

Error Analysis:

- Conduct an error analysis to identify common misclassifications or missed detections. This can inform further refinements in annotation or training strategies.

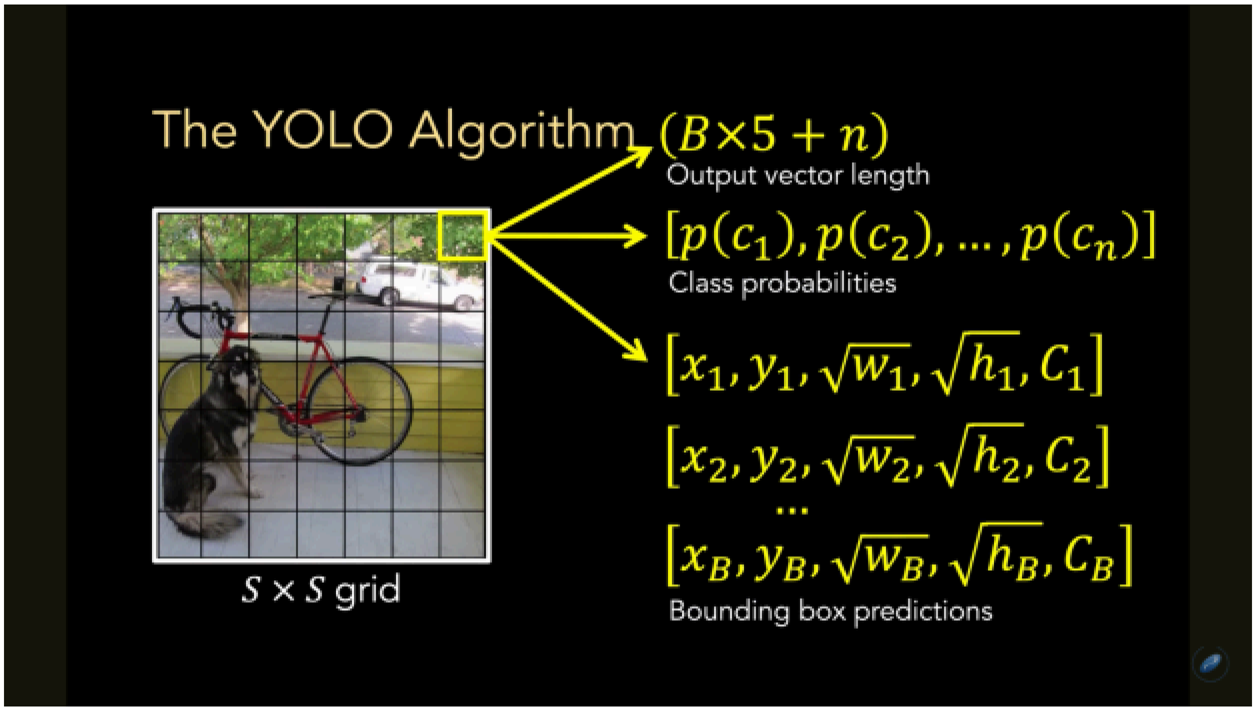
Final Testing

Testing on Unseen Data:

- After training, test the model on the test set to evaluate its performance on completely unseen data.
- Assess the generalization capability of the model in real-world scenarios.

Deployment Considerations:

- If the model performs well, consider deployment in a production environment or integrate it into a GIS (Geographic Information System) for real-time analysis.


The YOLO Algorithm
$(B\times 5+n)$
Output vector length
$[p(c_1), p(c_2), \ldots, p(c_n)]$
Class probabilities
$[x_1, y_1, \sqrt{w_1}, \sqrt{h_1}, C_1]$
$[x_2, y_2, \sqrt{w_2}, \sqrt{h_2}, C_2]$
...
$[x_B, y_B, \sqrt{w_B}, \sqrt{h_B}, C_B]$
Bounding box predictions
$S\times S$ grid

6. Result

After three epochs of training, the YOLOv5 model was able to identify different chess pieces with an overall precision of 0.144 and a recall of 0.489 for all classes. The mean average precision (mAP50) of the model at the IoU threshold of 0.50 was found to be 0.142, whereas the mAP50-95, which takes a broader range of IoU thresholds into account, was 0.0592. The white pawn (mAP50: 0.306) and white king (mAP50: 0.317) demonstrated the best detection accuracy among the individual chess pieces. Additionally, black pieces played fairly well, including the black king (mAP50: 0.163) and black pawn (mAP50: 0.284). With 157 layers and more than 7 million parameters, the model summary is stored in yolov5/runs/train/exp for future validation.