INFO 7375 - Neural Networks & AI

Home Work to Chapter - 17

Submitted By:

Abdul Haseeb Khan
NUID: 002844724
khan.abdulh@northeastern.edu

What are anchor boxes and how do they work?

Anchor boxes (also known as anchor priors or default boxes) are **pre-defined bounding boxes with specific sizes, aspect ratios, and positions** that serve as reference templates during object detection. They are placed at various positions across an image, often in a grid-like pattern, to capture objects of different scales and shapes.

How Anchor Boxes Work:

1. Generation and Placement
- Anchor boxes are generated at predefined locations across the image
- Multiple anchor boxes with different aspect ratios and scales are placed at each grid cell
- For example, you might have 2 anchor boxes per grid cell with different shapes (tall vs wide)

2. Training Process
- Each object in the training image is assigned to the grid cell containing the object's midpoint
- The object is then matched to the anchor box with the highest Intersection over Union (IoU)
- The model learns to predict:
    - **Offsets** to adjust the anchor box position and size to better fit the actual object
    - **Class probabilities** for what object is present
    - **Confidence scores** indicating whether an object is present

3. Prediction Output
- Instead of outputting 3×3×8 for a single prediction per grid cell
- With 2 anchor boxes, the output becomes 3×3×2×8 (or 3×3×16)
- Each anchor box gets its own set of predictions (Pc, bounding box coordinates, class probabilities)

4. Key Advantages
- **Efficiency**: Eliminates the need to scan the image with sliding windows at every position
- **Multiple scales**: Different sized anchor boxes can detect objects of various sizes
- **Aspect ratio handling**: Different shaped boxes (1:1, 1:2, 2:1) handle objects with varying proportions

The key insight is that anchor boxes serve as **templates that the model refines** rather than having to predict bounding boxes from scratch, making the object detection task more manageable and efficient.

What is bounding box prediction and how does it work?

Bounding box prediction is the process of determining rectangular regions that enclose objects of interest within an image. It involves predicting four coordinates (x_min, y_min, x_max, y_max) or (x_center, y_center, width, height) that define where objects are located.

1. Grid-Based Approach

- The image is divided into a grid (e.g., 3×3)
- Each grid cell is responsible for detecting objects whose center falls within that cell
- The model processes the entire image and outputs predictions for each grid cell

2. Output Structure

For each grid cell, the model predicts:

- **Pc** (Probability of object): Whether an object is present (0 or 1)
- Bounding box coordinates: (bx, by, bw, bh)
    - bx, by: Center position of the box
    - bw, bh: Width and height of the box
- **Class probabilities**: (c1, c2, c3, ...) for different object classes

3. With Anchor Boxes

The prediction process becomes more sophisticated:

Without Anchor Boxes:

- Output: 3×3×8 (for 3 classes)
- Each grid cell predicts one bounding box

With Anchor Boxes (e.g., 2 anchor boxes):

- Output: 3×3×16 (or 3×3×2×8)
- Each grid cell predicts adjustments for multiple anchor boxes
- The model predicts **offsets** to refine pre-defined anchor box positions

4. Training Process

- Ground truth bounding boxes are provided in the training data
- Objects are assigned to grid cells based on their center points
- When using anchor boxes, objects are matched to the anchor box with highest IoU
- The model learns to predict:
    - Whether an object is present
    - How to adjust the anchor box to fit the actual object
    - What class the object belongs to

5. Key Components of Prediction

For each bounding box, the model predicts:

- **Confidence score**: How confident the model is that an object exists
- **Location refinement**: Adjustments to position and size
- **Class predictions**: Probability distribution over possible classes

6. Offset Prediction

Rather than predicting absolute coordinates, modern models predict:

- **Offset values** that adjust reference boxes (anchor boxes)
- These offsets help fine-tune the bounding box position

- For example, if the predicted object's face is partially cut off, offset values help adjust the box boundaries

Describe R-CNN

R-CNN is a pioneering object detection architecture from 2014 that combines region proposals with convolutional neural networks to detect and classify objects in images. The system works in four stages: first, it takes an input image and uses the Selective Search algorithm to generate approximately 2000 region proposals (candidate areas likely to contain objects); second, each region proposal is warped to a fixed size and fed through a CNN that extracts a 4096-dimensional feature vector; third, these features are classified using an SVM to determine the object class; and finally, the system predicts offset values to refine the bounding box positions. While R-CNN was revolutionary in demonstrating that CNNs could be effectively used for object localization and achieved significant performance improvements over previous methods, it suffers from major limitations including extremely slow processing (taking ~47 seconds per test image), inefficient training that requires classifying 2000 regions per image, and the use of a fixed, non-learnable Selective Search algorithm that cannot improve during training. These limitations led to the development of faster variants like Fast R-CNN (which processes all regions together) and Faster R-CNN (which uses learnable convolutional networks for region proposals), but R-CNN remains historically significant as the foundation for modern region-based object detection systems.

What are advantages and disadvantages of R-CNN?

Advantages:

- **Accurate Object Detection**: R-CNN provides highly accurate object detection by leveraging region-based convolutional features, excelling in scenarios where precise object localization and recognition are crucial
- **Robustness to Object Variations**: Can handle objects with different sizes, orientations, and scales, making it suitable for real-world scenarios with diverse objects and complex backgrounds
- **Flexibility**: Versatile framework that can be adapted to various object detection tasks, including instance segmentation and object tracking - by modifying the final layers, R-CNN can be tailored to specific needs
- **Historical Significance**: Pioneered the successful combination of CNNs with region proposals, establishing the foundation for modern object detection architectures

Disadvantages:

- **Computational Complexity**: R-CNN is computationally intensive, involving extracting region proposals, applying a CNN to each proposal, and running extracted features through a classifier - this multi-stage process is slow and resource-demanding
- **Slow Inference**: Due to sequential processing of region proposals, R-CNN is relatively slow during inference (taking around 47 seconds per test image), making the latency unacceptable for real-time applications

- **Slow Training**: Takes a huge amount of time to train the network as it needs to classify 2000 region proposals per image
- **Overlapping Region Proposals**: May generate multiple region proposals that overlap significantly, leading to redundant computation and potentially affecting detection performance
- **Not End-to-End**: Unlike modern architectures like Faster R-CNN, R-CNN is not an end-to-end model - it involves separate modules for region proposal and classification, which can lead to suboptimal performance compared to models that optimize both tasks jointly
- **Fixed Region Proposal Algorithm**: The Selective Search algorithm is fixed with no learning happening at that stage, which could lead to the generation of bad candidate region proposals

What is semantic segmentation?

Semantic segmentation is a computer vision task where **the goal is to categorize each pixel in an image into a specific class or object**. Unlike object detection which draws bounding boxes around objects, semantic segmentation assigns a class label to every individual pixel in the image, producing a dense pixel-wise segmentation map.

Key Characteristics:

Per-Pixel Classification: Every pixel is treated independently, and the model predicts what class that pixel belongs to based on the input features at that pixel's location. This creates a complete segmentation where the entire image is divided into distinct categorical regions.

Practical Applications: For example, in autonomous driving, semantic segmentation helps identify different elements of the road scene by classifying pixels as:

- Road (pink)
- Sidewalk (cyan)
- Building (yellow)
- Vegetation (green)
- Vehicle (red)
- Fence (brown)
- Pole (blue)

The result is a color-coded map where each color represents a different class, allowing the system to understand exactly which parts of the image belong to which category. This is crucial for applications like autonomous vehicles that need to identify vehicles, pedestrians, traffic signs, pavement, and other road features to navigate safely.

Difference from Object Detection:

While object detection places rectangular bounding boxes around objects, semantic segmentation provides the exact shape and boundary of each object by classifying every pixel, offering much more precise spatial information about where different objects and regions are located in the image.

How does deep learning work for semantic segmentation?

Deep learning for semantic segmentation uses a specialized encoder-decoder architecture that first compresses the input image to extract features, then expands it back to produce pixel-wise predictions. In the encoding path, the image passes through standard convolutional layers that progressively reduce spatial dimensions while increasing feature depth, extracting high-level semantic features and contextual information. The decoding path then uses transposed convolutions (up-convolutions) to gradually increase the spatial resolution back to the original image size, with new convolutional layers symmetrically added with growing size and reduced depth. A crucial innovation is the use of skip connections, which pass high-resolution features from early encoding layers directly to corresponding decoding layers, preserving spatial details that would otherwise be lost during downsampling and combining precise localization with semantic understanding. U-Net exemplifies this approach with its characteristic U-shaped architecture: the contracting path uses repeated 3×3 convolutions followed by ReLU and 2×2 max pooling for downsampling (doubling feature channels at each step), while the expansive path uses 2×2 up-convolutions that halve feature channels, concatenating with cropped feature maps from the contracting path. The final 1×1 convolution maps each pixel's feature vector to class probabilities, producing a segmentation map where every pixel is assigned a class label. This architecture effectively combines what is in the image (through deep features) with where exactly it is located (through skip connections and precise upsampling), enabling accurate pixel-level classification that's essential for applications like autonomous driving where understanding the exact shape and boundary of objects is crucial.

What is transposed convolution?

Transposed convolution, also called up-convolution, is an operation that increases the spatial dimensions of feature maps, essentially performing the opposite of regular convolution which typically reduces spatial dimensions. It's a crucial component in semantic segmentation architectures where compressed feature representations need to be expanded back to the original image resolution for pixel-wise predictions.

The operation works by sliding a kernel across the input tensor and creating multiple intermediate results that are then summed together. Specifically, for an input tensor of size n_h × n_w and a kernel of size k_h × k_w, the kernel slides n_w times in each row and n_h times in each column, generating n_h×n_w intermediate tensors of size (n_h + k_h - 1) × (n_w + k_w - 1), each initialized with zeros. At each position, every element in the input tensor is multiplied by the entire kernel, and this product replaces a corresponding portion in the intermediate tensor. The position of this replacement corresponds to where the input element was located. Finally, all these intermediate results are summed element-wise to produce the output, which has larger spatial dimensions than the input.

The slide provides a concrete example showing a 2×2 input being transformed into a 3×3 output using a 2×2 kernel. In this example, each of the four input values (0, 1, 2, 3) is multiplied by the entire kernel, and these products are strategically positioned and summed to create the expanded output. This effectively "spreads" the input information across a larger spatial area, with the final output being [0,0,1], [0,4,6], [4,12,9].

In the context of semantic segmentation, transposed convolution serves a critical purpose: after the encoding path compresses the image to extract high-level semantic features, the decoding path uses transposed convolutions to progressively expand these features back to the original image size. Unlike simple interpolation methods, transposed convolution is learnable, meaning the network can optimize how features are upsampled for the specific segmentation task, maintaining both semantic information and spatial precision needed for accurate pixel-level classification.

Describe U-Net.

U-Net is a convolutional neural network architecture developed at the University of Freiburg for biomedical image segmentation, designed to work effectively with fewer training images while yielding more precise segmentation results. Named for its characteristic U-shaped architecture, the network consists of a contracting path (left side) that captures context and an expansive path (right side) that enables precise localization.

The contracting path follows a typical convolutional network architecture, consisting of repeated application of two 3×3 unpadded convolutions, each followed by a ReLU activation and a 2×2 max pooling operation with stride 2 for downsampling. At each downsampling step, the number of feature channels is doubled, progressively reducing spatial dimensions while increasing feature depth to extract high-level semantic information. This encoding process compresses the image from its original resolution down to a compact feature representation at the bottom of the "U."

The expansive path performs symmetric upsampling through a series of up-convolutions (transposed convolutions). Every step consists of an upsampling of the feature map using a 2×2 up-convolution that halves the number of feature channels, followed by a concatenation with the correspondingly cropped feature map from the contracting path, and two 3×3 convolutions each followed by ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer, a 1×1 convolution maps each 64-component feature vector to the desired number of classes, with the network having 23 convolutional layers in total.

A defining characteristic of U-Net is its use of skip connections (shown as gray arrows in the architecture diagram) that directly connect layers from the contracting path to corresponding layers in the expansive path. These connections pass high-resolution features from early layers to later layers, preserving spatial information that would otherwise be lost during downsampling. This combination of deep features (for understanding what is in the image) with shallow features (for understanding where exactly it is) enables precise localization crucial for segmentation tasks.

U-Net's efficiency and effectiveness have made it widely influential beyond biomedical imaging. The architecture processes a 512×512 image in less than a second on modern GPUs, making it practical for real-world applications. Interestingly, the U-Net architecture has also been employed in diffusion models for iterative image denoising, demonstrating its versatility beyond its original segmentation purpose. Its ability to produce accurate segmentations with limited training data, combined with its elegant architectural design that balances context capture with precise localization, has established U-Net as a foundational architecture in the field of semantic segmentation.