□ Why Selective Search?

Before R-CNN, methods like the **sliding window** looked at **every possible position and scale** in an image — very slow and redundant.
So, **Selective Search** came to **reduce** the number of regions to check while **keeping most of the real objects**.

- Sliding window → checks ~100,000+ windows
- Selective Search → generates ~2,000 **good quality object proposals**

□ Step-by-Step Working of Selective Search

Step 1: Input Image

Start with a normal image.
Selective Search aims to find regions that might contain an object.

Step 2: Initial Over-segmentation (using Superpixels)

- The algorithm divides the image into many small segments called **superpixels**.
- Common method: **Felzenszwalb and Huttenlocher's segmentation algorithm**.
- Each superpixel is a small region of similar color, texture, and intensity.

□ So, instead of looking at each pixel, we now look at these **small meaningful areas**.

Example:
An image of a dog might be divided into 100–500 small color-consistent patches — parts of fur, background grass, etc.

Step 3: Extract Region Features

For each superpixel region, calculate simple features like:

- **Color histogram** (dominant colors)
- **Texture histogram** (pattern information)
- **Size**
- **Shape (bounding box coordinates)**

Step 4: Region Merging (Hierarchical Grouping)

Now comes the core part.

The algorithm tries to **merge regions that look similar** step-by-step:

1. Compute **similarity** between every pair of neighboring regions.
2. Merge the most similar pair.
3. Recompute similarity between new region and its neighbors.
4. Repeat until the whole image becomes one big region.

Step 5: Similarity Measure

How does it decide what to merge?

A combined similarity score is computed using:

- **Color similarity** — how close color histograms are.
- **Texture similarity** — similar texture patterns.
- **Size similarity** — smaller regions are merged earlier to avoid bias toward large ones.
- **Fill similarity** — how well regions fit together spatially.

The final similarity = weighted sum of these four.

Step 6: Form Region Proposals

Each time two regions merge, the **bounding box** of the merged region is saved as a **region proposal**.

- At the end, all these merged boxes (at all hierarchy levels) are collected.
- Usually results in **~2000 region proposals per image**.

These boxes vary in **size, position, and aspect ratio**, covering most potential objects.

□ Step 7: Use in R-CNN

- Each region proposal (bounding box) is cropped and **warped** to a fixed size (e.g., 224×224).
- Then each one is fed into the **CNN** to extract features.
- A classifier (like SVM) decides **which object** (if any) is present.

□ Example Intuition

Imagine an image of a **cat on a couch**:

1. Segmentation → small color-consistent regions: cat fur, eyes, couch fabric, background wall.
2. Merging → cat fur patches combine into one region → forms a bounding box around the cat.
3. Other merges form couch box, background box, etc.
4. R-CNN will later test each box to see which contains the "cat".

□ Summary Table

Step	Process	Purpose
1	Over-segmentation	Divide image into small color-consistent regions
2	Feature extraction	Describe each region (color, texture, size)
3	Similarity computation	Find pairs of regions that look alike
4	Hierarchical merging	Combine regions step by step
5	Bounding box creation	Save boxes at each merge level
6	Output ~2000 boxes	Use them as inputs to CNN in R-CNN

□ Limitations of Selective Search

- **Slow** — must compute and merge thousands of regions.
- **Handcrafted** — not learned; it doesn't improve with training.
- **Produces redundant boxes** (many overlapping boxes for the same object).

That's exactly why **Faster R-CNN** later introduced the **Region Proposal Network (RPN)** — a deep-learning-based alternative to Selective Search.