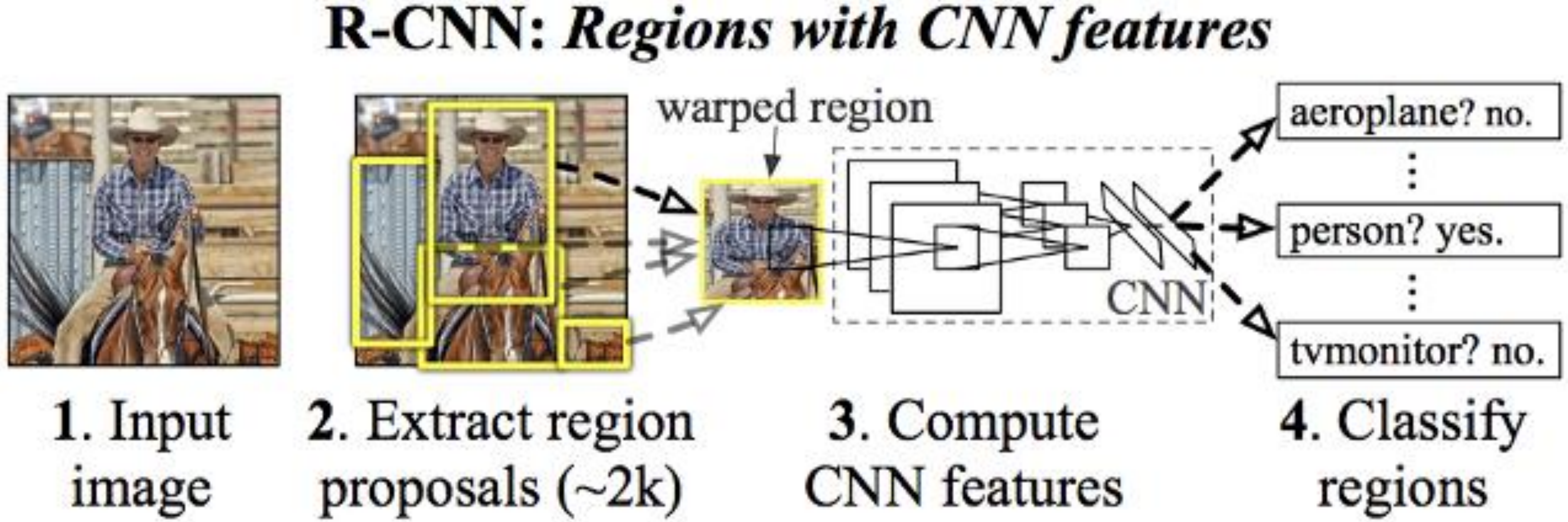
R-CNN

R-CNN


R-CNN: Regions with CNN features
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

R-CNN – Overview

- Ross Girshick et al. proposed a method where we use selective search to extract 2000 regions from the image and he called them region proposals
- These 2000 region proposals are generated using the **selective search** algorithm which is written below.
    1. Generate initial sub-segmentation, we generate many candidate regions
    2. Use greedy algorithm to recursively combine similar regions into larger ones
    3. Use the generated regions to produce the final candidate region proposals

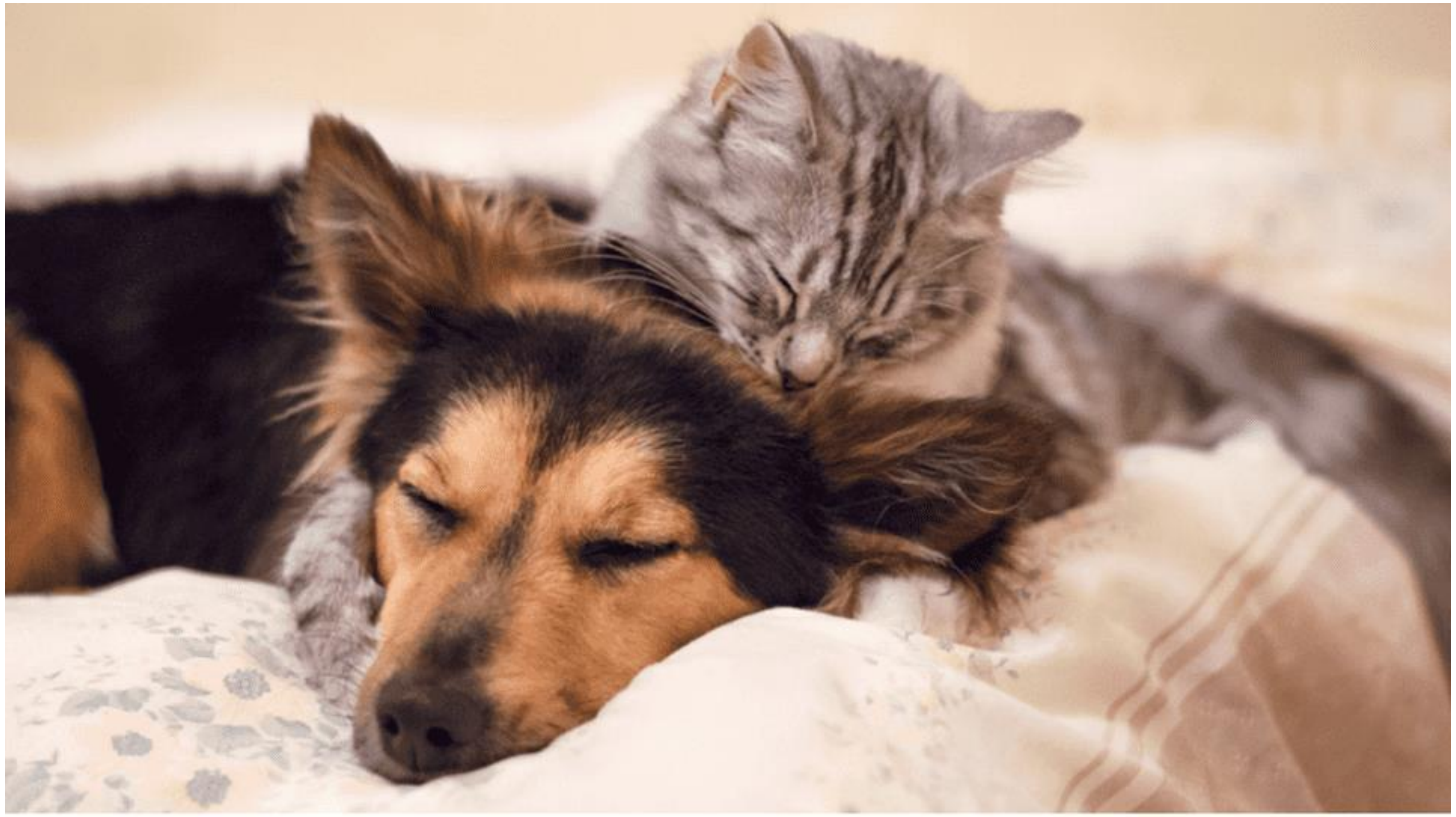
R-CNN - Overview

- These 2000 candidate region proposals **are warped into a square** and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output
- The **CNN** acts as a feature extractor and the output dense layer consists of the features extracted from the image
- The extracted features are fed into an **SVM** to classify the presence of the object within that candidate region proposal
- In addition to predicting the presence of an object within the region proposals, the algorithm also **predicts four values** which are offset values to increase the precision of the bounding box
- For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could’ve been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.

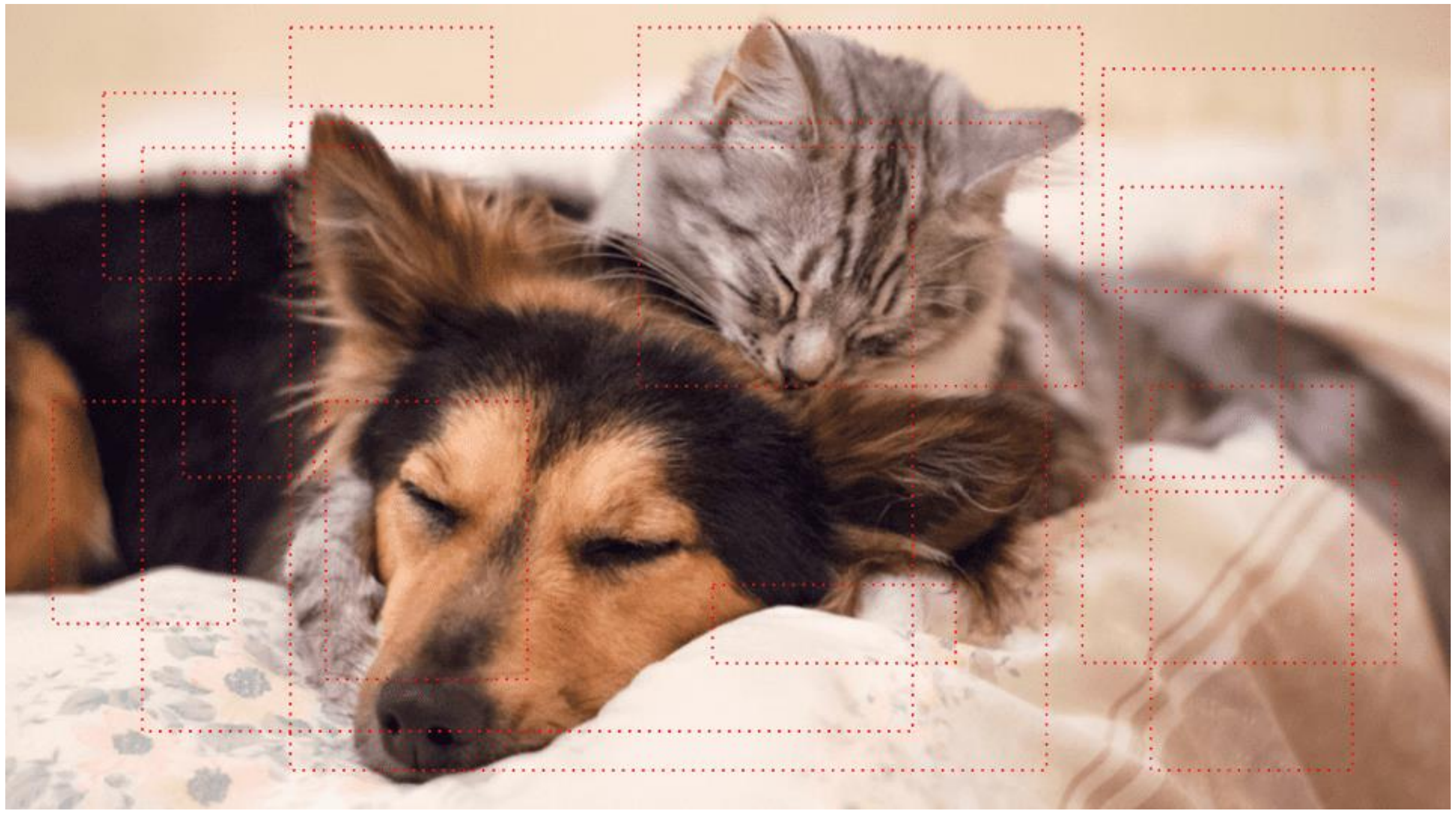
Pictorial flow of R-CNN

https://www.mihaileric.com/posts/object-detection-with-rcnn

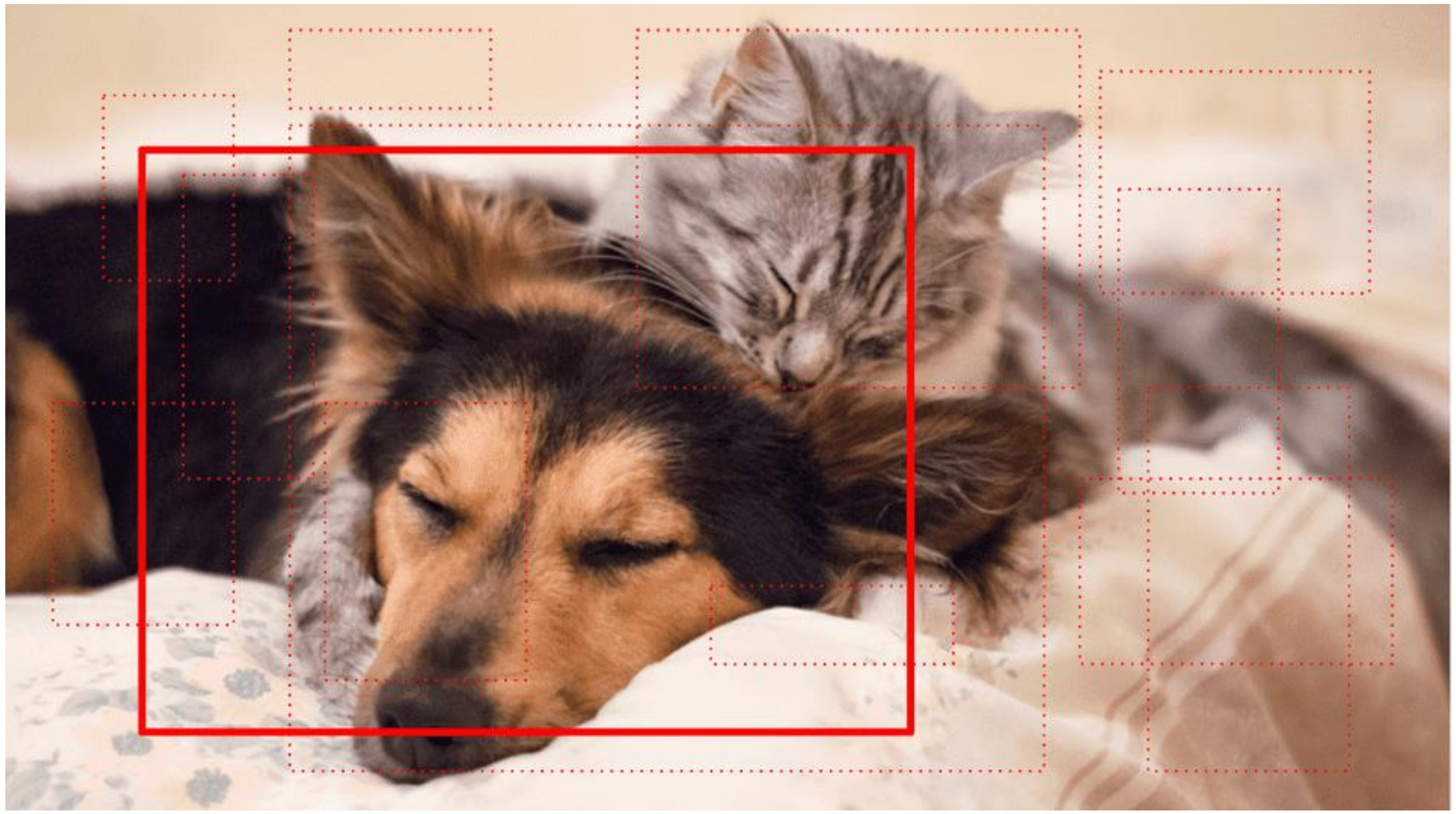
Let’s say we wanted to run an R-CNN object detection on the following image



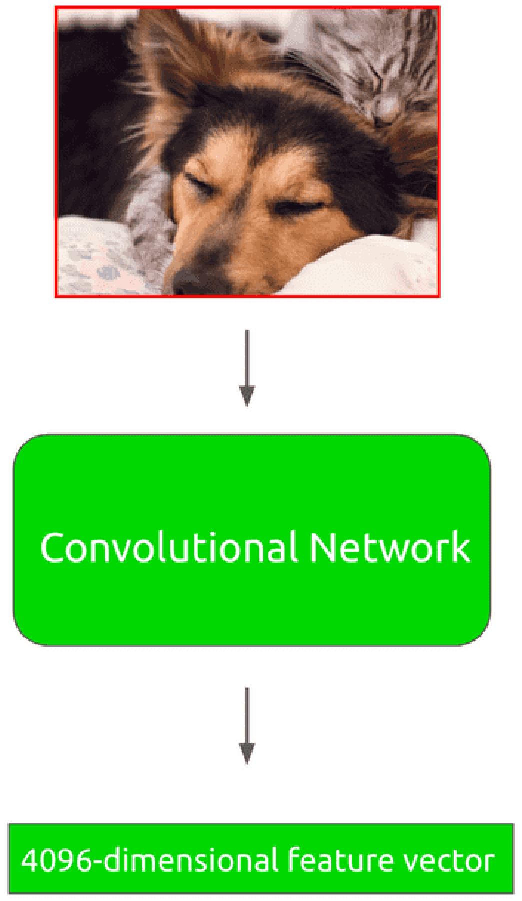
Selective search extracts around 2000 region proposals per image, which would look as follows



Now R-CNN runs each proposal through a high-capacity convolutional neural network to extract a learned feature representation, which ideally holds meaningful information about the objects in the image.
For example, the algorithm would take a proposal such as the following

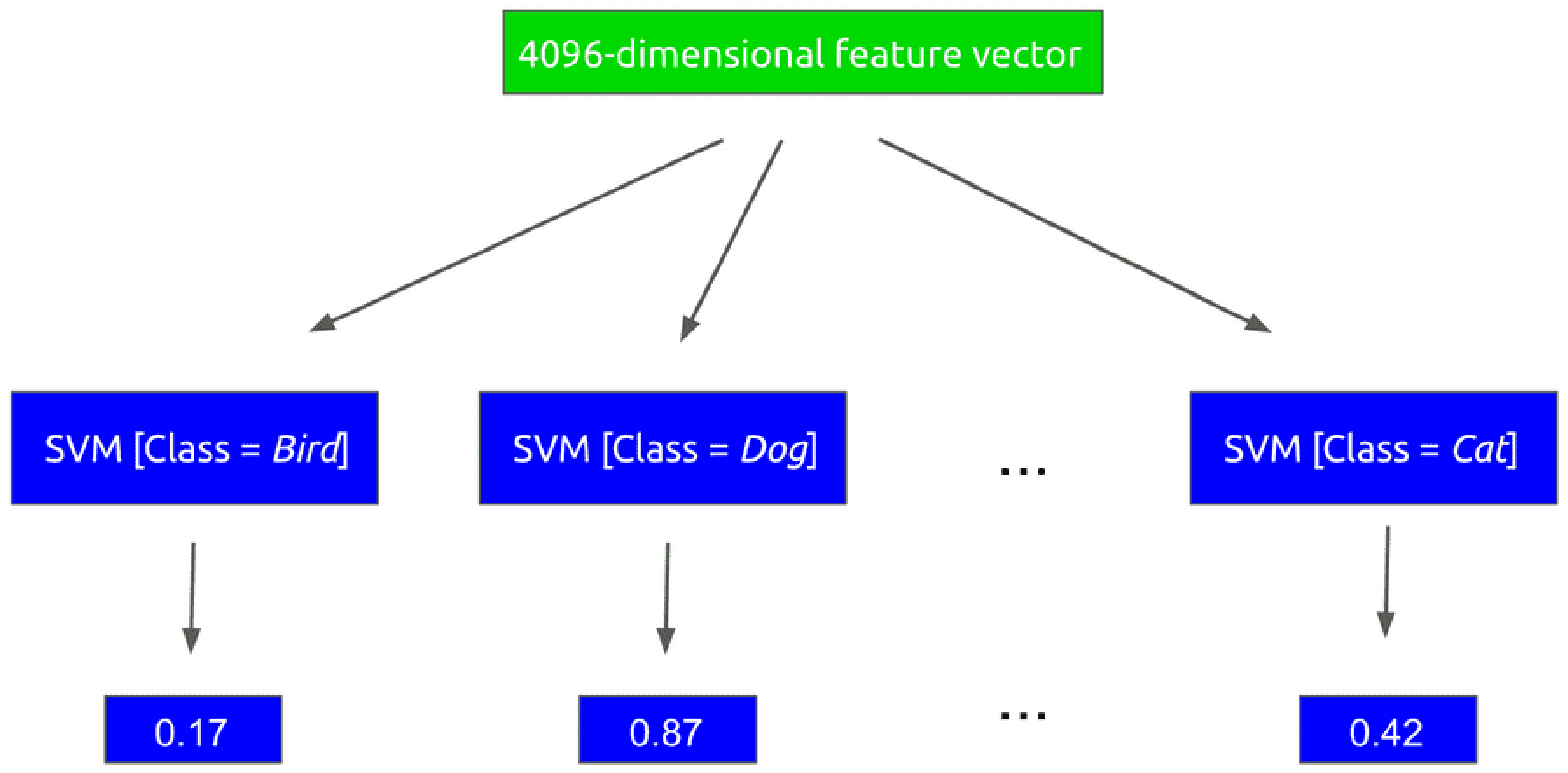


This proposal would be forward-propagated through a convolutional neural network to generate a 4096-dimensional feature vector



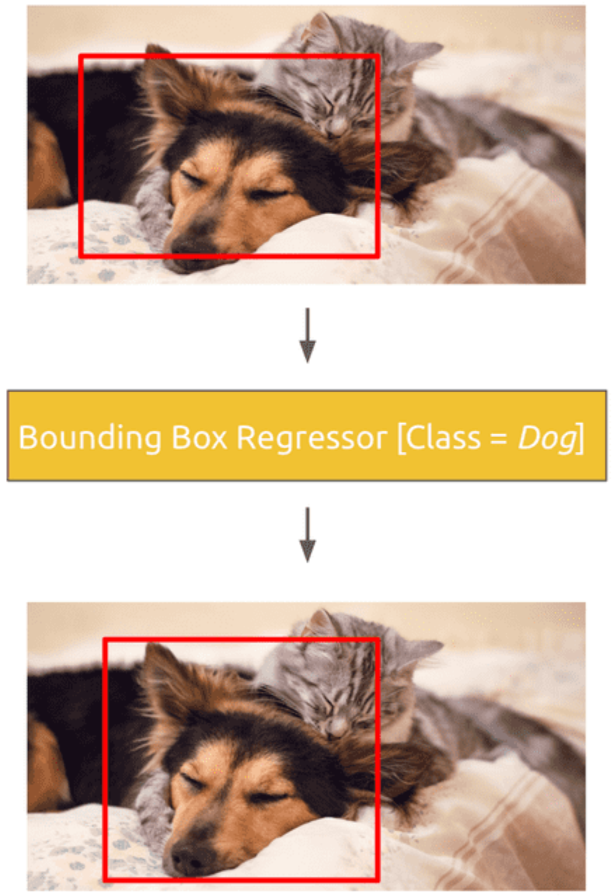
Afterwards, this feature vector is run through a collection of linear support vector machines (SVM for short), where each SVM is designed to classify for a single object class. In other words, there is an SVM trained to detect *boat*, another one for *parrot*, etc.

Each SVM outputs a score for the given class, indicating the likelihood that the region proposal contains that class. Note, that the exact classes that can be identified depend on the dataset used for training

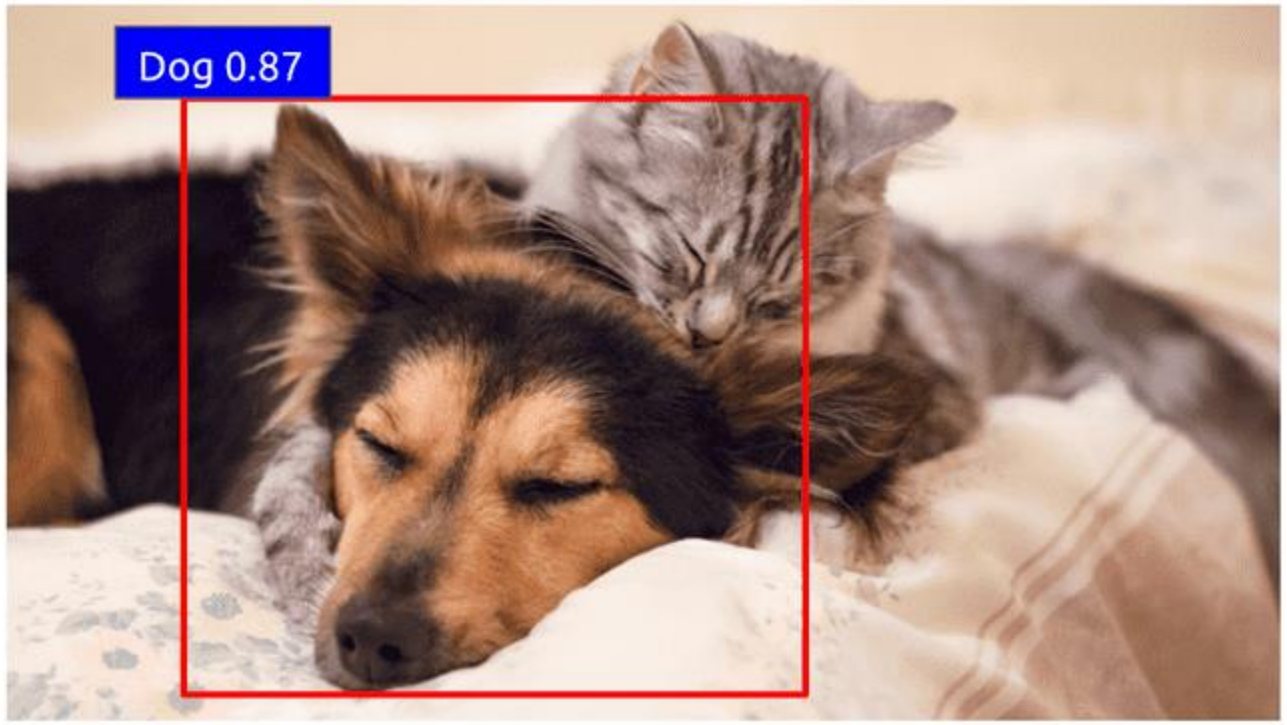


R-CNN will then label the region proposal with the class corresponding to the highest-scoring SVM, in this case *dog*

After R-CNN has scored each region proposal with a class-specific SVM, the bounding box for the proposal is refined using a class-specific bounding-box regressor. This could adjust the region in the following way



When the process is finished, the detected object for the region could look as follows

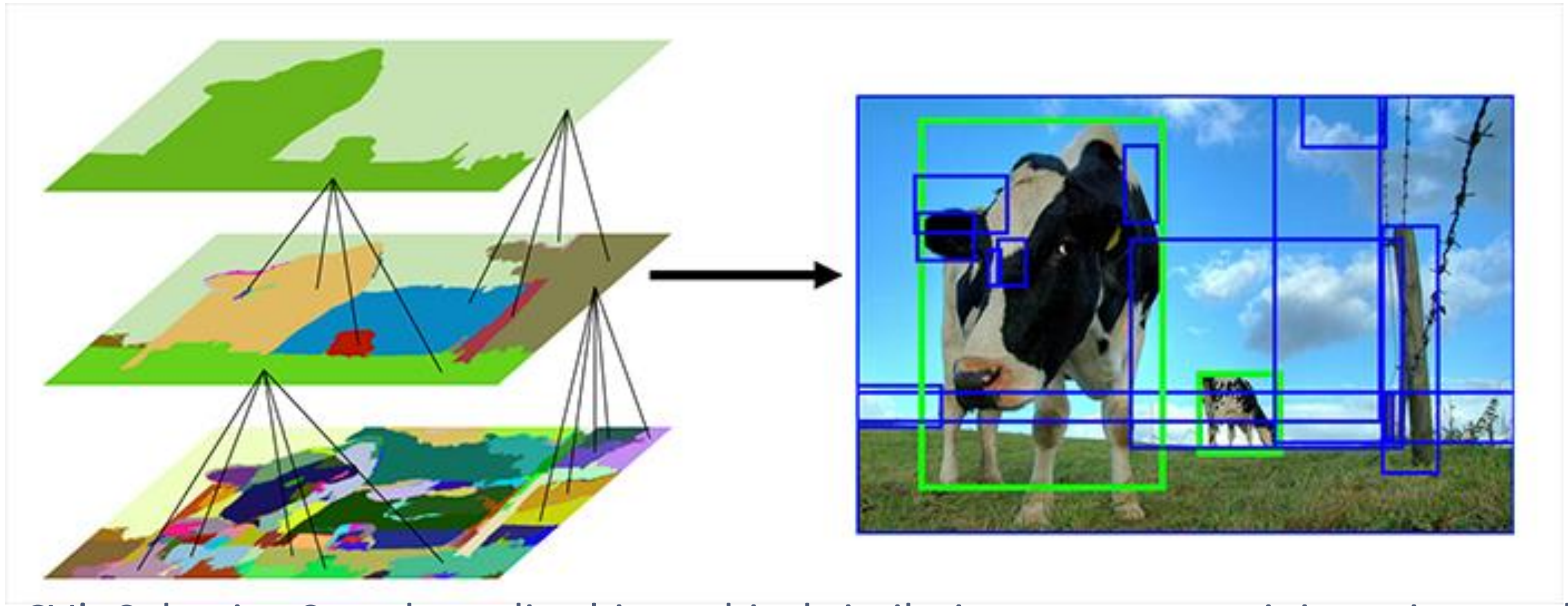


Note, this running example was only for a single region proposal. Running the algorithm on the other regions should detect that there is also a cat in the image

Selective Search

Selective Search

- The results of Selective Search applying these hierarchical similarity measures



OpenCV's Selective Search applies hierarchical similarity measures to join regions and eventually form the final set of proposals for where objects could be present

- On the bottom layer of the pyramid, we can see the original over-segmentation/superpixel generation from the Felzenszwalb method.
- In the middle layer, we can see regions being joined together, eventually forming the final set of proposals (top)
- Selective search merges superpixels in a hierarchical fashion based on five key similarity measures: **Colour similarity, Texture similarity, Size similarity, shape similarity/compatibility, final meta similarity measure** – which is a linear combination of the four similarities

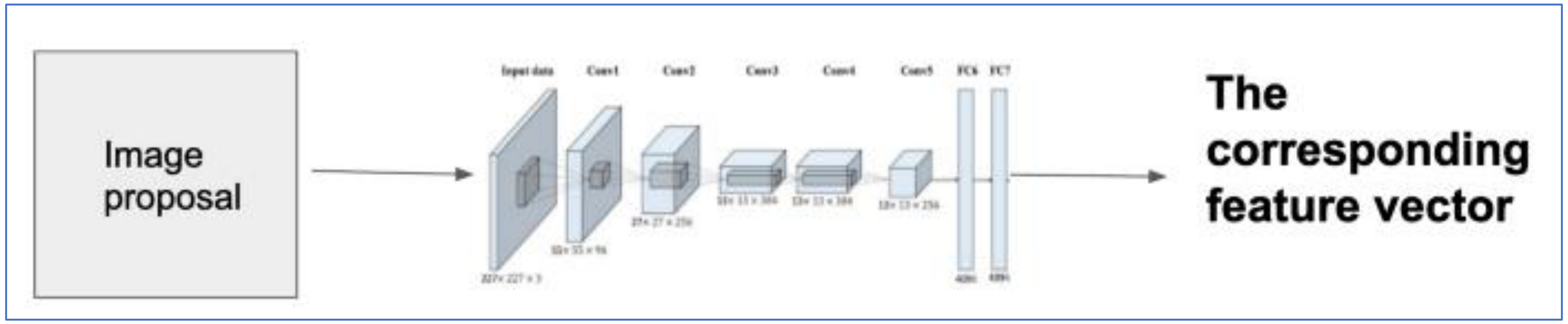
R-CNN Steps

R-CNN: Step 1 – Region proposals

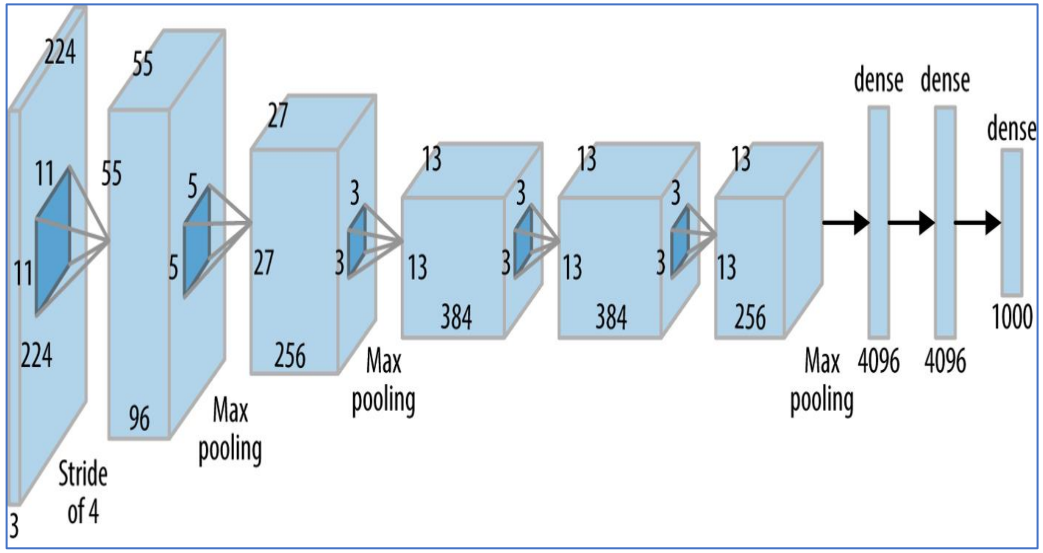
- R-CNN, using **selective search**, identifies a manageable number of bounding-box object region candidates
- *Region proposals are just smaller parts of the original image, that we think could contain the objects we are searching for.*
- There are different region proposal algorithms we can choose from. These are “normal” algorithms that work out of the box. We don’t have to train them or anything
- *The R-CNN is agnostic to the region proposal method.*
- This will create nearly 2000 different regions we will have to look at.

R-CNN – Step 2: Feature vector for each Region proposal

- In the next step, we take each region proposal and we will create a feature vector representing this image in a much smaller dimension using a Convolutional Neural Network (CNN)



- They used the AlexNet as a feature extractor
- The important thing to keep in mind is that the input to the AlexNet is always the same (227, 227, 3). The image proposals have different shapes though. Many of them are smaller or larger than the required size. So we will need to **resize every region proposal**.
- One fundamental issue with this R-CNN system. You can't train the whole system in one go (This will be solved by the fast R-CNN system)
- Rather, you will need to train every part independently. That means that the AlexNet was trained before on a classification task
- After the training, they removed the last softmax layer. Now the last layer is the fully connected 4096-dimensional one. **This means that our features are 4096 dimensional.**



R-CNN Step 3: SVM for feature classification

- We created feature vectors from the image proposals. Now we will need to classify those feature vectors
- The features and labeled category of each proposed region are combined as an example to train multiple support vector machines (SVM) for object classification.
- Here, each support vector machine is used to determine whether an example belongs to a certain category.
  - How we trained those different SVM's? Well, we train them on feature vectors created by the AlexNet. That means, that we have to wait until we fully trained the CNN before we can train the SVM. The training is not parallelizable
  - Because we know when training what feature vector represented which class we can easily train the different SVM's in a supervised-learning way

R-CNN Step 4: Bounding box

- Now we have image proposals that are classified on every object class. How do we bring them all back to the image?
- We use something called greedy non-maximum suppression.
- We combine each region if there is an overlap we take the proposal with the higher score (calculated by the SVM). We do this step for each object class independently
- After this ends we only keep regions with a score higher than 0.5.

R-CNN Step 4: Bounding box regressor

- The features and labeled bounding box of each proposed region are combined as an example to train a regression model for ground-truth bounding box prediction.
- When you are training the Bounding Box Regressor your input is the center, width and height in pixels of the region proposal and the label is the ground truth bounding box. The goal as stated in the paper is:
- *Our goal is to learn a transformation that maps a proposed box P to a ground-truth box G*

Problems with R-CNN

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented in real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals
- One fundamental issue with this R-CNN system. You can’t train the whole system in one go (This will be solved by the fast R-CNN system)
- Rather, you will need to train every part independently. That means that the AlexNet (CNN) was trained before on a classification task