Fast RCNN

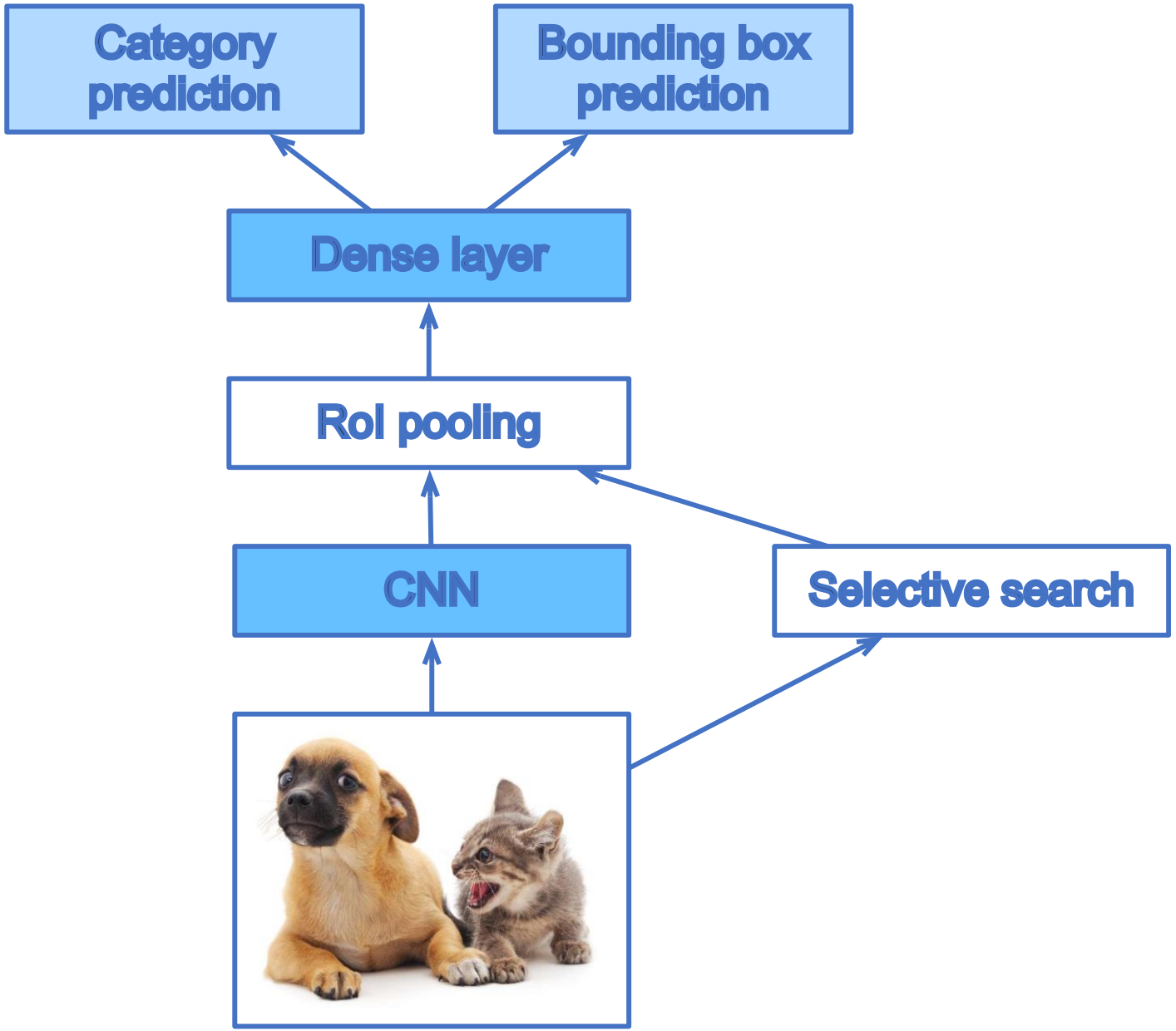
Fast R-CNN

- This paper was published one year after the original R-CNN paper and directly builds on top of it. The R-CNN paper was a major breakthrough in 2014, combining region proposals with a CNN. But it had some problems:
  - **It was slow**: You had to calculate a feature map (one CNN forward pass) for each region proposal.
  - **Hard to train**: Remember that in the R-CNN System we had 3 different parts (CNN, SVM, Bounding Box Regressor) that we had to train separately. This makes training very difficult.
  - **Large memory requirement**: You had to save every feature map of each region proposal. This needs a lot of memory
- So, how did the author try to solve those problems. The major thing introduced in this paper is the following:
  - *We only have one system, that we can train end-to-end.*
- We combine the three different parts that we had in the R-CNN system (CNN, SVM, Bounding Box Regressor) into **one** architecture.

Fast R-CNN

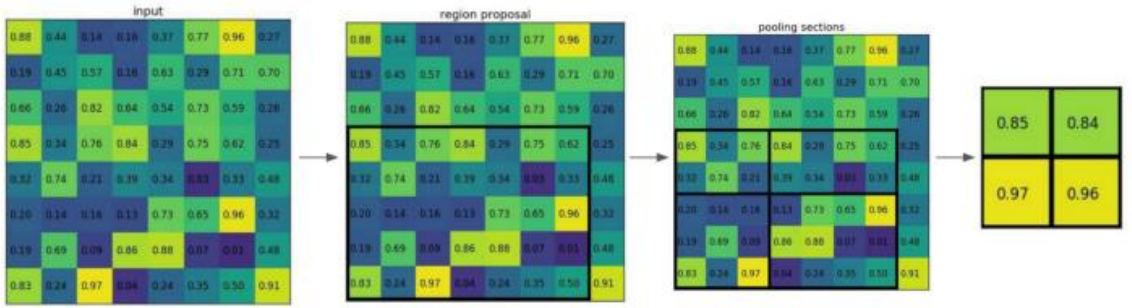
- The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN
- The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map
- From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer
- From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.
- The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it

fast R-CNN Architecture



Fast R-CNN - details

1. Process the whole image with the CNN. The result is a feature map of the image
2. For each region proposal extract the corresponding part from the feature map. We will call this the **region proposal feature map**.
3. We take the region proposal feature map from the feature map and resize it to a fixed size with the help of a pooling layer.
4. This pooling layer is called the Region of interest (RoI) pooling layer.

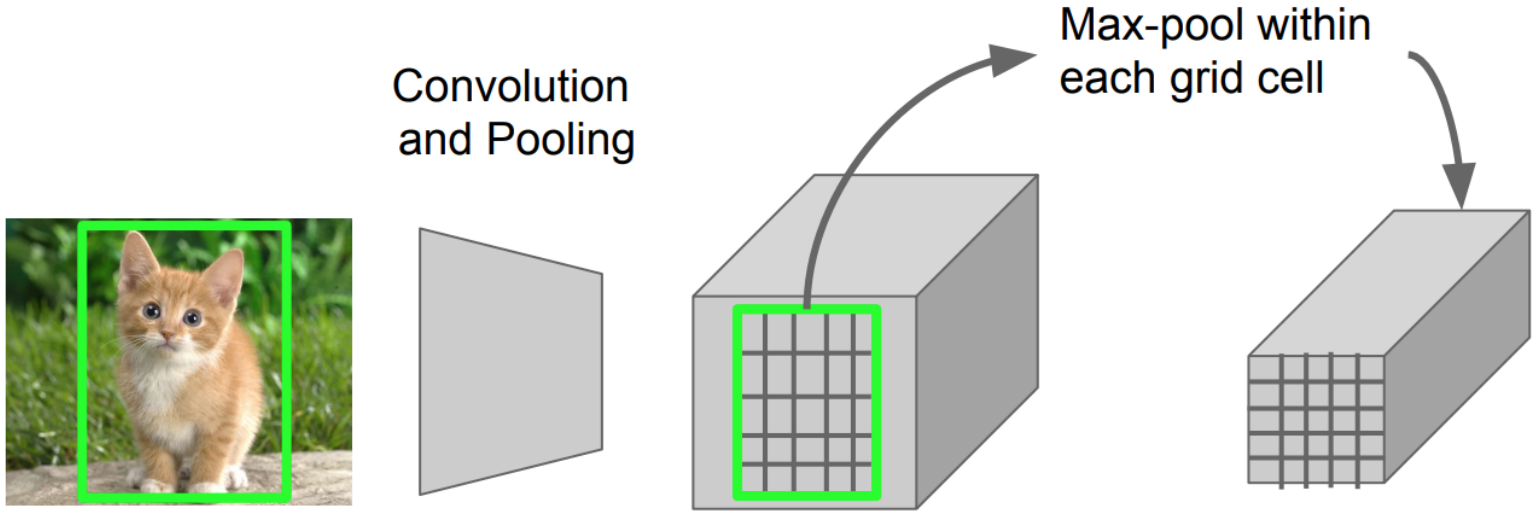


5. Then we flatten this fixed sized region proposal feature map. This is now a feature vector, that always has the same size.
6. This feature vector is now the input to the last part. These are fully connected layers that have 2 outputs. The first is the softmax classification layer, where we decide which object class we found. The second it the Bounding Box Regressor, where we output the bounding box coordinates for each object class.

- **Results:** The fast R-CNN trains the VGG16 network 9 times faster than R-CNN. But the amazing thing about this system is this:
- **The inference is 213 times faster and** achieves **a higher accuracy**

RoI Pooling

- It is a type of max pooling to convert features in the projected region of the image of any size, h x w, into a small fixed window, H x W
- The input region is divided into H x W grids, approximately every subwindow of size h/H x w/W. Then apply max-pooling in each grid.



Fast R-CNN Model Workflow

1. First, pre-train a convolutional neural network on image classification tasks.
2. Propose regions by selective search (~2k candidates per image).
3. Alter the pre-trained CNN:
   - Replace the last max pooling layer of the pre-trained CNN with a RoI pooling layer
   - The **RoI pooling layer outputs fixed-length feature vectors of region proposals**.
   - Sharing the CNN computation makes a lot of sense, as many region proposals of the same images are highly overlapped.
   - Replace the last fully connected layer and the last softmax layer (K classes) with a fully connected layer and softmax over K + 1 classes.
4. Finally the model branches into two output layers:
   - A softmax estimator of K + 1 classes (same as in R-CNN, +1 is the “background” class), outputting a discrete probability distribution per RoI.
   - A bounding-box regression model which predicts offsets relative to the original RoI for each of K classes.

Fast R-CNN : Bottlenecks

- Fast R-CNN is significantly faster in training and testing sessions over R-CNN
- When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals
- Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.