Region of interest pooling explained

February 28, 2017 / 42 Comments / in Data science, Deep learning, Machine learning / by Tomasz Grel

Region of interest pooling (also known as RoI pooling) is an operation widely used in object detection tasks using convolutional neural networks. For example, to detect multiple cars and pedestrians in a single image. Its purpose is to perform max pooling on inputs of nonuniform sizes to obtain fixed-size feature maps (e.g. 7×7).

We've just released an open-source implementation of RoI pooling layer for TensorFlow (you can find it here). In this post, we're going to say a few words about this interesting neural network layer. But first, let's start with some background.

Two major tasks in computer vision are object classification and object detection. In the first case the system is supposed to correctly label the dominant object in an image. In the second case it should provide correct labels and locations for all objects in an image. Of course there are other

POPULAR POSTS

Don't waste the power. AI supply chain management
December 13, 2018 - 4:09 pm

Powering up demand forecasting with machine learning
December 6, 2018 - 2:41 pm

AI Monthly digest #3: Artificial Intelligence in science getting big
December 6, 2018 - 2:41 pm

CATEGORIES

- Big data & Spark
- Data science
- Deep learning
- Machine learning
- Neptune
- Reinforcement learning
- Seahorse

FOLLOW OUR NEWSLETTER

Your e-mail

Subscribe

You can modify your privacy settings and unsubscribe from our lists at any time (see our privacy policy).

interesting areas of computer vision, such as image segmentation, but today we're going to focus on detection. In this task we're usually supposed to draw bounding boxes around any object from a previously specified set of categories and assign a class to each of them. For example, let's say we're developing an algorithm for self-driving cars and we'd like to use a camera to detect other cars, pedestrians, cyclists, etc. — our dataset might look like this.

In this case we'd have to draw a box around every significant object and assign a class to it. This task is more challenging than classification tasks such as MNIST or CIFAR. On each frame of the video, there might be multiple objects, some of them overlapping, some poorly visible or occluded. Moreover, for such an algorithm, performance can be a key issue. In particular for autonomous driving we have to process tens of frames per second.

So how do we solve this problem?

Related: Playing Atari with deep reinforcement learning - deepsense.ai's approach

Typical architecture

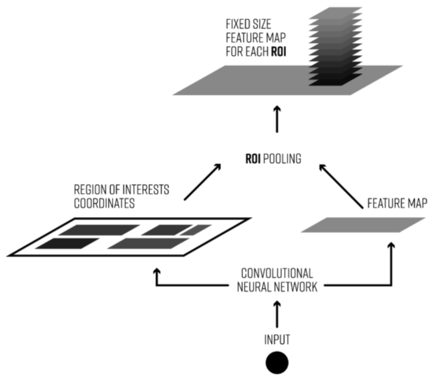
The object detection architecture we're going to be talking about today is broken down in two stages:

1. Region proposal: Given an input image find all possible places where objects can be located. The output of this stage should be a list of bounding boxes of likely positions of objects. These are often called region proposals or regions of interest. There are quite a few methods for this task, but we're not going to talk about them in this post.

ARCHIVES

Select Month

2. Final classification: for every region proposal [illegible] whether it belongs to one of the target classes or to the background. Here we could use a deep convolutional network.


FIXED SIZE FEATURE MAP FOR EACH ROI
ROI POOLING
REGION OF INTERESTS COORDINATES
FEATURE MAP
CONVOLUTIONAL NEURAL NETWORK
INPUT

Object detection pipeline with region of interest pooling

Usually in the proposal phase we have to generate a lot of regions of interest. Why? If an object is not detected during the first stage (region proposal), there’s no way to correctly classify it in the second phase. That’s why it’s extremely important for the region proposals to have a high recall. And that’s achieved by generating very large numbers of proposals (e.g., a few thousands per frame). Most of them will be classified as background in the second stage of the detection algorithm.

Some problems with this architecture are:

- Generating a large number of regions of interest can lead to performance problems. This would make real-time object detection difficult to implement.

- It's suboptimal in terms of processing speed.
- N
- You can't do end-to-end training, i.e., you can't train all the components of the system in one run (which would yield much better results)

That's where region of interest pooling comes into play.

Related: Optimize Spark with DISTRIBUTE BY & CLUSTER BY

Region of interest pooling — description

Region of interest pooling is a neural-net layer used for object detection tasks. It was first proposed by Ross Girshick in April 2015 (the article can be found here) and it achieves a significant speedup of both training and testing. It also maintains a high detection accuracy. The layer takes two inputs:

1. A fixed-size feature map obtained from a deep convolutional network with several convolutions and max pooling layers.
2. An N x 5 matrix of representing a list of regions of interest, where N is a number of RoIs. The first column represents the image index and the remaining four are the coordinates of the top left and bottom right corners of the region.

An image from the Pascal VOC dataset annotated with region proposals (the pink rectangles)

What does the RoI pooling actually do? For every region of interest from the input list, it takes a section of the input feature map that corresponds to it and scales it to some pre-defined size (e.g., 7×7). The scaling is done by:

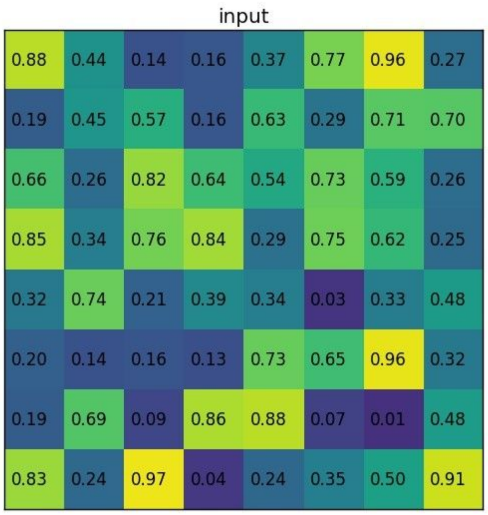
1. Dividing the region proposal into equal-sized sections (the number of which is the same as the dimension of the output)
2. Finding the largest value in each section
3. Copying these max values to the output buffer

The result is that from a list of rectangles with different sizes we can quickly get a list of corresponding feature maps with a fixed size. Note that the dimension of the RoI pooling output doesn't actually depend on the size of the input feature map nor on the size of the region proposals. It's determined solely by the number of sections we divide the proposal into. What's the benefit of RoI pooling? One of them is processing speed. If there are multiple region proposals of the same frame (and usually there'll be a lot of them), we can still use the same input feature map for all of them. Since computing the convolutions at early stages of processing is very expensive, this approach can save us a lot of time.

Related: Deep learning for satellite imagery via image segmentation

Region of interest pooling — example

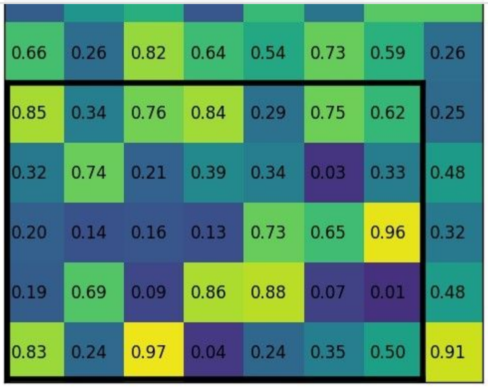
Let's consider a small example to see how it works. We're going to perform region of interest pooling on a single 8×8 feature map, one region of interest and an output size of 2×2. Our input feature map looks like this:



input

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

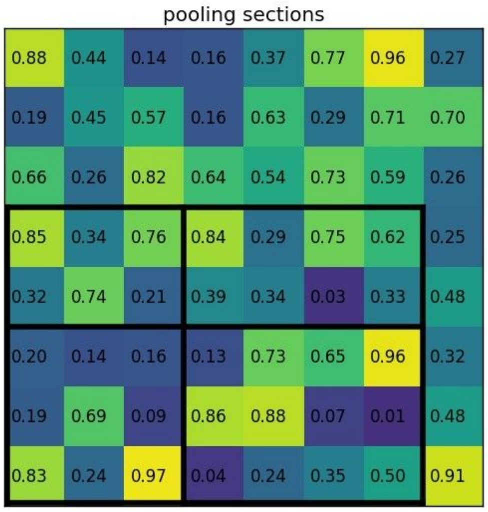
Let's say we also have a region proposal (top left, bottom right coordinates): (0, 3), (7, 8). In the picture it would look like this:



0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

Normally, there'd be multiple feature maps and multiple proposals for each of them, but we're keeping things simple for the example.

By dividing it into (2×2) sections (because the output size is 2×2) we get:

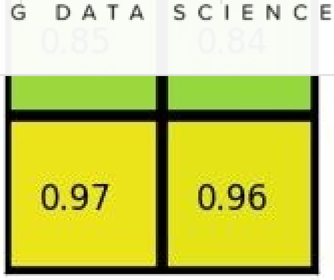


pooling sections

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

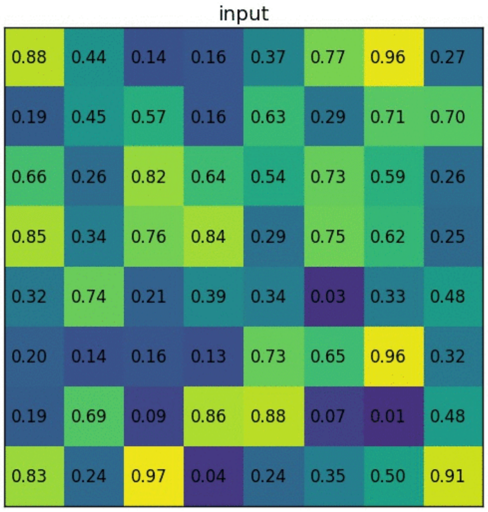
Notice that the size of the region of interest doesn't have to be perfectly divisible by the number of pooling sections (in this case our RoI is 7×5 and we have 2×2 pooling sections).

The max values in each of the sections are:



0.85	0.84
0.97	0.96

And that's the output from the Region of Interest pooling layer. Here's our example presented in form of a nice animation:



input

0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

What are the most important things to remember about RoI Pooling?

- It's used for object detection tasks
- It allows us to reuse the feature map from the convolutional network
- It can significantly speed up both train and test time
- It allows to train object detection systems in an end-to-end manner

If you need an opensource implementation of RoI pooling in TensorFlow you can find our version here.

In the next post, we're going to show you some examples on how to use region of interest pooling with Neptune and TensorFlow.

References:

- Girshick, Ross. “Fast r-cnn.” Proceedings of the IEEE International Conference on Computer Vision. 2015.
- Girshick, Ross, et al. “Rich feature hierarchies for accurate object detection and semantic segmentation.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
- Sermanet, Pierre, et al. “Overfeat: Integrated recognition, localization and detection using convolutional networks.” arXiv preprint arXiv:1312.6229. 2013.

Related Posts

Crime forecasting – ‘Minority Report’ realized

Don’t waste the power. AI supply chain management

Powering up demand forecasting with machine learning

Share this entry

42

REPLIES

Jere
March 1, 2017 at 12:22 pm

Quite informative. Waiting for your next post

Reply

Rui
March 31, 2017 at 3:26 pm

typo: “Let’s say we also have a region proposal (top left, bottom right coordinates): (5,0), (10,7)”, should be (0, 7)

Reply

Filip Novotny
April 15, 2017 at 4:35 pm

Hello,

a very good explanation of RoI

However I am confused by this:
In this case we'd have to *draw a box around every significant object* and assign a class to it. This task is more challenging […]
So how do we solve this problem?

From what I understand, you say that Fast(er) R-CNN allow us to predict locations of objects without any ground truth boxes: just run the thing and candidates class-agnostic boxes will pop out.

However, this is a quote from the Fast r-cnn paper:
"Third, the network is modified to take two data inputs: a
list of images and a *list of RoIs* in those images."

There seems to be a need for ground truth boxes. Am I missing something or is there a mistake in the article?

Thanks,

F.

Reply

Tomasz Grel
April 24, 2017 at 8:22 am

Hello,

You definitely need ground truth boxes to train the model and at test time, the model is going to predict both the bounding box and the class label. So by "draw a box around every significant object" I meant that the boxes are generated at test time, at train time you have to provide the ground truth boxes and labels.

Reply

Jamie
April 21, 2017 at 2:17 am

Many thanks for the condensed explanation ☺

Reply

Krupa
June 2, 2017 at 7:46 am

I think I'm missing something here. How is the mapping from the input ROI to the feature map ROI done?

Reply

Tomasz Grel
June 2, 2017 at 1:04 pm

Usually the coordinates resulting from the Region Proposal Network are not compatible with the input feature map to the ROI pooling. This is mitigated by careful post processing of those coordinates.

For the details you can read our blog post about ROI pooling with a complete and working example here: https://blog.deepsense.ai/region-of-interest-pooling-in-tensorflow-example/

Reply

Alex Wang
August 24, 2017 at 12:16 pm

Hi:
I am confused by this:
'For every region of interest from the input list, it takes a section of the input feature map that corresponds to it'

for example:
a image of size (299,299), and a region of interest (cat , 50,40, 30, 30), how can I find the region in feature map corresponding to this ROI?
Thanks.

Reply

Tomasz Grel
August 28, 2017 at 8:17 am

In our example we used the ROI format of top left and bottom right corner of the rectangle, so the ROI (50,40,30,30) makes little sense (it should rather be (30,30,50,40) The corresponding ROI is the section of the feature map between the points (30,30)

Reply

llp
October 26, 2017 at 6:43 am

The ground truth boxes are based on images, but the rois are based on feature maps, how did they correspond to each other? I'm confused about this. Thank you.

Reply

Tomasz Grel
October 26, 2017 at 8:11 am

In general you might need to postprocess the boxes so that they match the feature maps. If you need a reference you can find an example along with TensorFlow code in one of our other posts: https://blog.deepsense.ai/region-of-interest-pooling-in-tensorflow-example/

Reply

Krzysztof
July 20, 2018 at 2:27 pm

As far as I understand their code, they are using VGG without last pooling layer, so they transform image (224 x 224 x 3) into (14 x 14 x 512) – hight ... 16. They map RoIs coordinates just by dividing them by 16.

I have the feeling that original authors did the same, as this step isn't explained in the paper and dividing by some factor is most obvious way of doing it.

Reply

Sai

December 8, 2017 at 4:29 pm

Great article. Can you elaborate a bit in why the pooling sections the way they are? Is there any intuition for it

Reply

Tomasz Grel

December 8, 2017 at 4:32 pm

The intuition is that you want to partition the image in roughly equal parts, but the size of the output is fixed and the input image can have any size.

Reply

Sai

December 8, 2017 at 10:44 pm

Thanks for the reply! My confusion is about different possible permutations of 4 pooling rectangles. The output will differ based on the 4 rectangles we choose right?

Reply

Sai

December 8, 2017 at 10:44 pm

Thanks for the timely reply! My confusion is about different possible permutations of 4 pooling rectangles. The output will differ based on the 4 rectangles we choose right?

Reply

Tomasz Grel

December 11, 2017 at 8:03 am

Yes, you're right it would be different, but we don't really care about it. It's perfectly sufficient to just pick one deterministic method and stick with it.

Reply

Ryan

I think there's a typo in case there's confusion for beginners.
"Let's say we also have a region proposal (top left, bottom right coordinates): (0, 3), (7, 8). In the picture it would look like this:"
if 1-index is used, top left coordinate would be (1,3) instead of (0,3) according to the picture used

Reply

Ryan
December 21, 2017 at 3:26 am

I think there's a typo in case there's confusion for beginners.
"Let's say we also have a region proposal (top left, bottom right coordinates): (0, 3), (7, 8). In the picture it would look like this:"
if 1-index is used, top left coordinate would be (1,3) instead of (0,3) according to the picture used

Reply

sunya
February 2, 2018 at 12:03 pm

what happens if the rois get too small, say (1*1) in the feature map?

Reply

sunya

February 3, 2018 at 1:48 pm

tensorflow.python.framework.errors_ /home/sunya1989/ml/lib/python3.5/s packages/roi_pooling/roi_pooling.so: undefined symbol: _ZTIN10tensorflow8OpKernelE'

Reply

Tomasz Grel
February 5, 2018 at 9:03 am

This looks like a low-level C++ issue. Please check that you're using the right gcc version and the right TensorFlow version.

Reply

Tomasz Grel
February 5, 2018 at 9:00 am

I don't think I've tested this scenario so I'm not sure. If you test this and notice incorrect behavior please create an issue on our github page.

Reply

AVi
March 18, 2018 at 11:00 am

did you understand that, what will happen if rois get too small? for example if we want output from ro pooling as 2×2 but our roi get smaller than that (1×1).

Reply

sunya
February 2, 2018 at 12:03 pm

what happens if the rois get too small, say (1*1) in the feature map?

Reply

sunya
February 2, 2018 at 12:18 pm

does it perform rescale? (transform rois to the feature map)

Reply

Tomasz Grel
February 5, 2018 at 9:02 am

It just performs the region of interest pooling, nothing else.

Reply

sunya
February 2, 2018 at 12:18 pm

does it perform rescale? (transform rois to the feature map)

Reply

sunya
February 3, 2018 at 1:48 pm

can't work it returns error [illegible] framework.errors. /home/sunya1989/ml/lib/python3.5/site-packages/roi_pooling/roi_pooling.so: undefined symbol: _ZTIN10tensorflow8OpKernelE'

Reply

Tomasz Grel

February 5, 2018 at 9:02 am

It just performs the region of interest pooling, nothing else.

Reply

Tomasz Grel

February 5, 2018 at 9:03 am

This looks like a low-level C issue. Please check that you're using the right gcc version and the right TensorFlow version.

Reply

Tomasz Grel

February 5, 2018 at 9:00 am

I don't think I've tested this scenario so I'm not sure. If you test this and notice incorrect behavior please create an issue on our github page.

Reply

did you understand that, what will happen if rois get too small? for example if we want output from ro pooling as 2×2 but our roi get smaller than that (1×1). Do we neglect this roi?

Reply

lufficc
March 7, 2018 at 8:57 am

what if region proposal size is smaller than fixed-size feature maps size??

Reply

Tomasz Grel
March 12, 2018 at 11:45 am

This situation is perfectly normal. Usually proposals will cover only a small patch of the image where the interesting object is located. The purpose of the RoI Pooling layer is to cut this interesting patch out of the feature map and feed it to the rest of the network for further classification.

Reply

lufficc
March [illegible] 2018 at [illegible] am

what if region proposal size is smaller than fixed-size feature maps size??

Reply

Tomasz Grel
March 12, 2018 at 11:45 am

This situation is perfectly normal. Usually proposals will cover only a small patch of the image where the interesting object is located. The purpose of the RoI Pooling layer is to cut this interesting patch out of the feature map and feed it to the rest of the network for further classification.

Reply

Neeraj Sajjan
June 25, 2018 at 11:37 am

In the code of the original implementation[https://github.com/rbgirs rcnn/blob/master/matlab/fast_rcnn_im_de and many other implementations[eg: https://github.com/yhenon/keras-frcnn/blob/master/keras_frcnn/RoiPooling the approach to roi pooling is different from what you have presented here.

In the implementation presented here,

If say we have roi size [7,5] and [illegible] then we get the first block size to be floor([7/2,5/2]),i.e,[3,2]. Rest of the boxes will have their sizes adjusted to be [4,2],[3,3] and [4,3].

But according to the original implementation and in several others, they use the same [3,2] as size for all boxes. This leads to ignoring some of the activations in roi proposal.

Am i correct in my above observation and if so, shouldnt your implementation be better than the original?

Reply

Tomasz Grel
June 29, 2018 at 8:29 am

I would indeed assume that ignoring parts of the feature map activations may result in lower accuracy.

Reply

Krzysztof
July 20, 2018 at 8:44 am

There is one part of RoI pooling, which isn't clear for me – you've wrote:
"What does the RoI pooling actually do? For every region of interest from

the input list, it takes a section of the [illegible] corresponds to it.

How do we map coordinates of RoI on original image to coordinates in last layer of ConvNet? For the nets proposed in paper (AlexNet, VGG) the problem is how to map coordinates from (224, 224, 3) to (7, 7, 512). It's not trivial at all. Even if we omit last pooling layer in convNet, then it's still (14, 14, 256).

Reply

Tomasz Grel
July 20, 2018 at 8:52 am

Great question. This is not trivial, but no matter what backbone you're using (VGG, Resnet, FPP) you should be able to postprocess the RoI coordinates to map them into original image coordinates. Most papers treat this step as a technical detail so they don't really talk about it. If you really want to dive deep into this my advice would be to read the original Fast-RCNN or Faster-RCNN code in Caffe or some other high-quality implementation.

Reply

Sahar

thank you for the post,

i want to concatenate the pooling result of multiple convolutional feature maps to generate the final pooling features for detection tasks. Specifically, features from multiple lower-levelconvolution layers are ROI-pooled and those resulting features are then concatenated. can i do this ?

Reply

Leave a Reply

Want to join the discussion?
Feel free to contribute!

Name *

Email *

□ I agree to allow deepsense.ai sp. z o.o. to process my personal data for the communication purposes relating to the services offered by deepsense.ai.

You can modify your privacy settings and unsubscribe from our lists at any time (see our privacy policy).

Post Comment

SERVICES

Start with AI

AI solutions

AI services

Team augmentation

Machine learning training

PRODUCTS

Neptune

Seahorse

ABOUT US

Blog

Company

Management

Scientific advisory board

Careers

Press center

CONTACT

contact@deepsense.a