Region of interest pooling explained

February 28, 2017 / 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 [https://github.com/deepsense-ai/roi-pooling]). 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 interesting areas of computer vision, such as image segmentation, but today we're going to focus on

---

START YOUR SEARCH HERE

Search

Search →

NEWSLETTER SUBSCRIPTION

Your e-mail

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

This site is protected by reCAPTCHA and the Google privacy policy and terms of service apply.

Subscribe

THE NEWEST AI MONTHLY DIGEST

AI Monthly Digest 20 – TL;DR May 12, 2020

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. [https://www.youtube.com/watch?v=KXpZ6B1YB_k]

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 [http://yann.lecun.com/exdb/mnist/] or CIFAR [https://www.cs.toronto.edu/~kriz/cifar.html]. 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
[https://deepsense.ai/playing-atari-with-deep-reinforcement-learning-deepsense-ais-approach/]
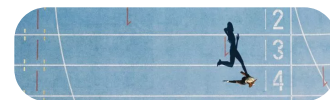
Typical architecture

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.

2. Final classification: for every region proposal from the previous stage, decide whether it belongs to one of the target classes or to the background. Here we could use a deep convolutional network.
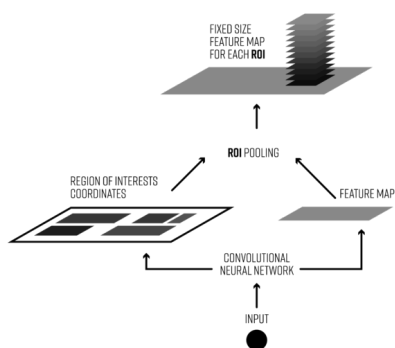
Object detection pipeline with region of interest pooling

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. More on this later.

- 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 [https://deepsense.ai/optimize-spark-with-distribute-by-and-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 [https://deepsense.ai/wp-content/uploads/2017/02/1504.08083.pdf]) 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.
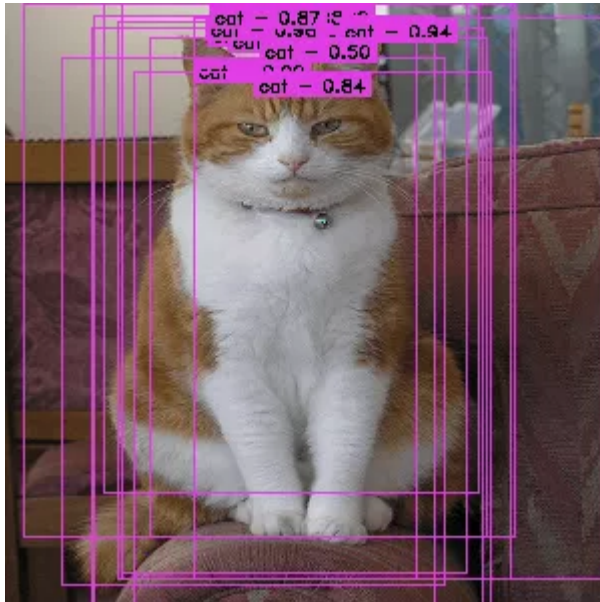
[https://deepsense.ai/wp-content/uploads/2017/02/region_proposal_cat.png]
An image from the Pascal VOC dataset annotated with region proposals (the pink rectangles)

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 object proposals on the 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 **[https://deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation/]**
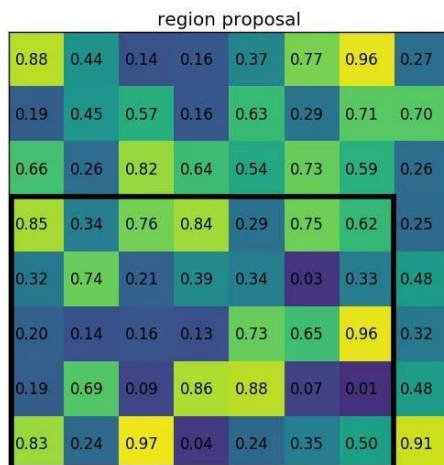
Region of interest pooling — example

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

[https://deepsense.ai/wp-content/uploads/2017/02/1.jpg]
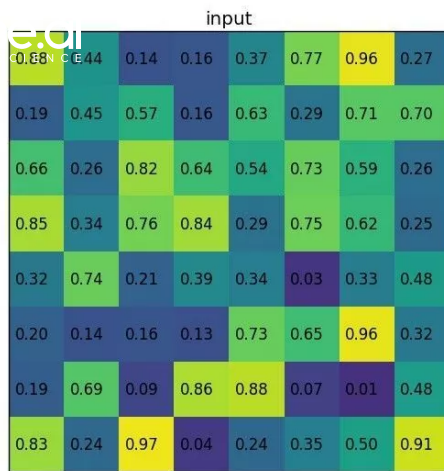
Let's say we also have a region proposal (top left, bottom right coordinates): (0, 3), (7, 8). In the picture it would look like this:

region proposal

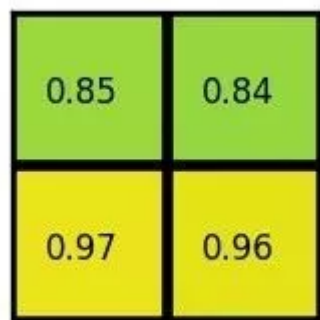[https://deepsense.ai/wp-content/uploads/2017/02/2.jpg]

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

[https://deepsense.ai/wp-
content/uploads/2017/02/3.jpg]
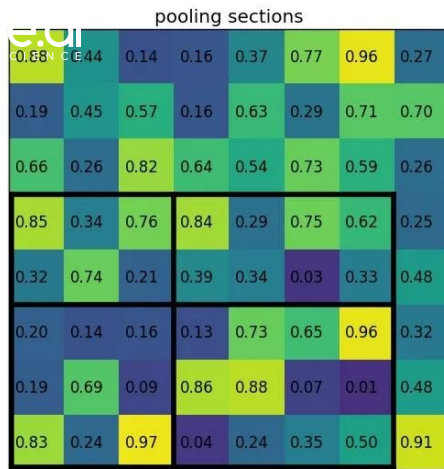
Notice that the size of the region of interest doesn't
have to be perfectly divisible by the number
of pooling sections (in this case our RoI is
7×5 and we have 2×2 pooling sections).
The max values in each of the sections are:

[https://deepsense.ai/wp-
content/uploads/2017/02/output.jpg]

And that's the output from the Region of Interest
pooling layer. Here's our example presented in form
of a nice animation:
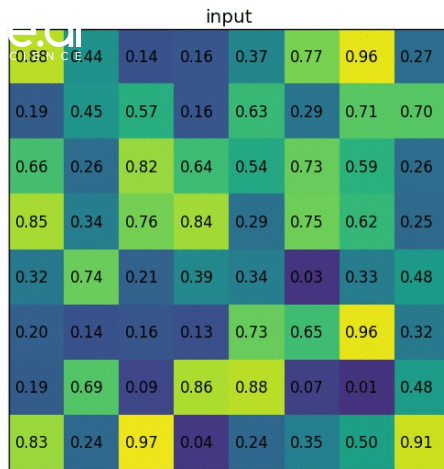
[https://deepsense.ai/wp-content/uploads/2017/02/roi_pooling-1.gif]

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 [https://github.com/deepsense-io/roi-pooling] .
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.

Share this entry

f 𝕏 🗨 in reddit ✉

Industries

Retail

Manufacturing

Financial Services

IT Operations

TMT and Other

Knowledge base

Blog

Press center

deepsense.ai

Our story

Management

Scientific Advisory Board

Careers

Support

Terms of service

Privacy policy

Contact us

Join our community

© deepsense.ai 2014-2021

This site uses cookies. By continuing to browse the site, you are agreeing to our use of cookies.

OK   Learn more

Cookie and Privacy Settings

How we use cookies

We may request cookies to be set on your device. We use cookies to let us know when you visit our websites, how you interact with us, to enrich your user experience, and to customize your relationship with our website.

Click on the different category headings to find out more. You can also change some of your preferences. Note that blocking some types of cookies may impact your experience on our websites and the services we are able to offer.

Essential Website Cookies

These cookies are strictly necessary to provide you with services available through our website and to use some of its features.

Because these cookies are strictly necessary to deliver the website, refuseing them will have impact how our site functions. You always can block or delete cookies by changing your browser settings and force blocking all cookies on this website. But this will always prompt you to accept/refuse cookies when revisiting our site.

We fully respect if you want to refuse cookies but to avoid asking you again and again kindly allow us to store a cookie for that. You are free to opt out

any time or opt in for other cookies to get a better experience. If you refuse cookies we will remove all set cookies in our domain.

We provide you with a list of stored cookies on your computer in our domain so you can check what we stored. Due to security reasons we are not able to show or modify cookies from other domains. You can check these in your browser security settings.

Other external services

We also use different external services like Google Webfonts, Google Maps, and external Video providers. Since these providers may collect personal data like your IP address we allow you to block them here. Please be aware that this might heavily reduce the functionality and appearance of our site. Changes will take effect once you reload the page.

Google Webfont Settings:

Google Map Settings:

Google reCaptcha Settings:

Vimeo and Youtube video embeds:

Privacy Policy

You can read about our cookies and privacy settings in detail on our Privacy Policy Page.

Accept settings   Hide notification only