Raktim Bora

Follow 85 Followers About

Object tracking with Siamese networks and detectron2

Learning image correlations to track objects in videos in real-time

Raktim Bora Mar 24, 2020 · 7 min read ★

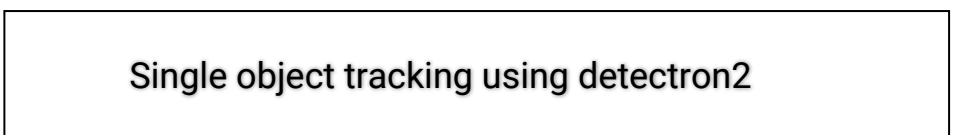
In this post, we will discuss the central concepts behind some of the state of the art papers in ***Single*** Object Tracking (SOT) namely **SiamFC**, **SiamRPN** and **SiamRPN++** . Putting words to action, we will also develop a Single Object tracker from scratch (without reinventing wheels as much as possible!) using the **detectron2** *object detection* framework in PyTorch.

If you are a "show me the code" sorta person, the **code lives here**. If you prefer to indulge me for a few minutes, grab a cup of coffee and read on!

borarak/detectron2-sot

This repository demonstrates construction of a Single Object Tracker (SOT) using the detectron2 framework. We mainly...

github.com



What is Single Object tracking?

The video above demonstrates single object tracking where we are interested in tracking the black pawn throughout the video. Object tracking is the ability to track an entity in a class agnostic manner, across multiple time steps (video frames) and often online, i.e without having previously seen an instance of the entity being tracked (online actually, is **one-shot** here). This generally involves identifying the object to track once at the beginning of a video. In the video above, we identify the black pawn once by simply drawing a bounding box around it in the first frame of the video. This is the only guidance we provide to the model about our object of interest. No training was done for black pawns specifically (or on any chess pieces for that matter) and the model learns to identify and track the object of interest (black pawn) and differentiate it from other instances of the same class (like the white pawn).

Siamese networks

Ok, I'm a big fan of Siamese networks. So far I have used Siamese networks to create applications to do anomaly detection, one-shot/few shot image recognition, CBIR (Content based image retrieval) etc. It's a powerful weapon that i believe every deep vision scientist should have in their arsenal!

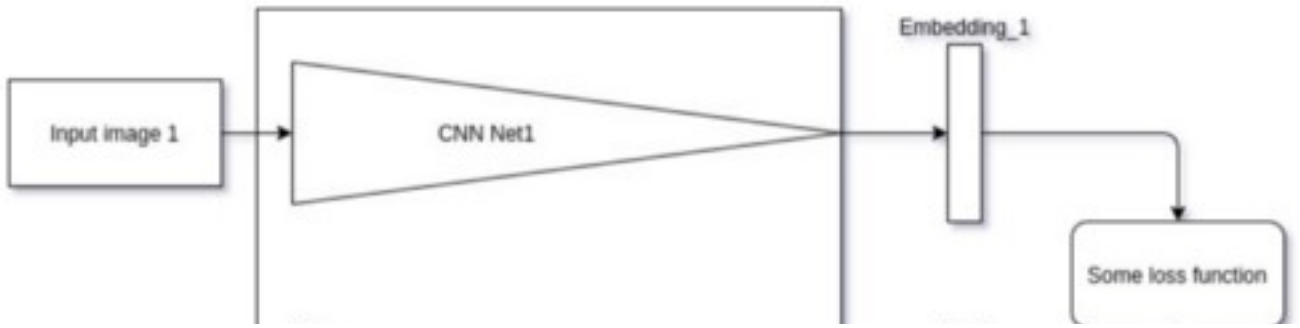


Fig 1. Typical network structure of a Siamese network

Siamese networks get their name from the fact that there are two twin neural networks in play that share the parameter space between them, as show in the Fig. 1. At their heart, Siamese networks create a high dimensional embedding representation of their input features. Train this embedding network with a differentiating criteria (e.g a L2 distance, contrastive loss etc.) and you get a powerful recipe that can measure similarity or dis-similarity between the inputs fed to the twin network.

All of the papers SiamFC, SiamRPN and SiamRPN leverage Siamese networks and hence the prefix — 'Siam' in their names.

SiamFC

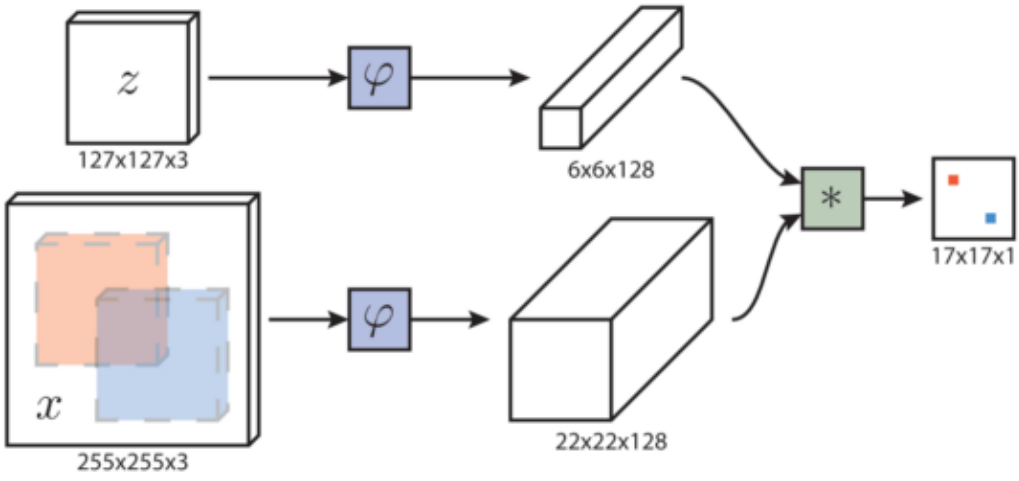


Fig 2. SiamFC network architecture, image taken from paper [1]

Given a **search image — *x***, SiamFC attempts to locate a smaller **template image — z** within x. This is done by first producing learned embeddings for both these images and then cross-correlating (**the * operator**) these embeddings. The resultant is a spatial score map containing confidences of presence of the template signal within the search signal. This is in essence equivalent to doing template matching in OpenCV but instead

SiamRPN

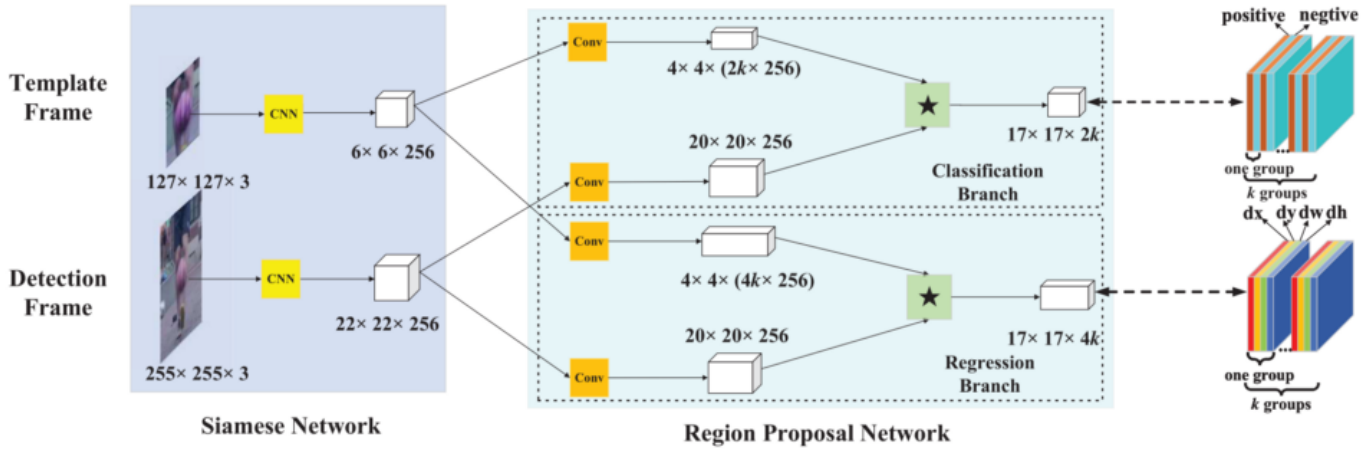


Fig 3. SiamRPN network architecture, image from paper [2]

SiamRPN builds on the idea of cross-correlation from SiamFC. A base **AlexNet** extractor is used as usual to produce search and template image embeddings but now instead of directly cross-correlating these embeddings, they are fed to a Region Proposal Network (RPN). The RPN then generates class and location filters from the template embeddings and subsequently cross-correlates them with the corresponding class and location feature maps from the search image.

If you are not familiar with Regional Proposal Networks, they are responsible for "proposing regions" in an image that are likely to contain an object. This usually involves regressing multiple bounding boxes with "objectness logits " (object or not-object scores) on the input image. "k" anchor boxes at each spatial position of a higher level feature map are used to produce 2k objectness logits (k boxes * 2 logits/box) and 4k localisation values (k boxes * 4 dims of a bounding box). RPN networks are generally class agnostic, i.e that are trained to only identify the presence or absence of an object, but in the case of binary classification/detection they actually serve the purpose of a fully functional object detector!

A detailed explanation of Anchor boxes and Region Proposal networks is beyond the scope of this post and for the interested, may i direct you to the seminal papers **FastRCNN** and **FasterRCNN** (the later introduces RPNs) that laid the foundations for

In SiamRPN the template embedding is fed to a RPN to produce a classification filter (4*4*2k*256 in above image) and a localization filter (4*4*4k*256 in above image) which are then cross-co-related with the corresponding RPN classification and localisation feature maps (20 * 20 * 256) from the input image in a “grouped” convolution manner. Similar to typical object detection, NMS suppression is used to arrive at the final bounding boxes containing our object. One of the key divergences from traditional object detection is the use of anchor boxes of only a single scale(multiple aspect ratios are maintained), as the input images are already cropped using the same scale prior to feeding the embedding network.

SiamRPN++

SiamRPN++ as the name might suggest are improvements on top of SiamRPN. The authors of SiamRPN++ explore some of the shortcomings of SiamRPN and propose measures that allow Siamese based trackers to achieve some of the best SOTA scores among contemporary single object trackers, at the same achieving impressive real-time speeds of upto 35fps.

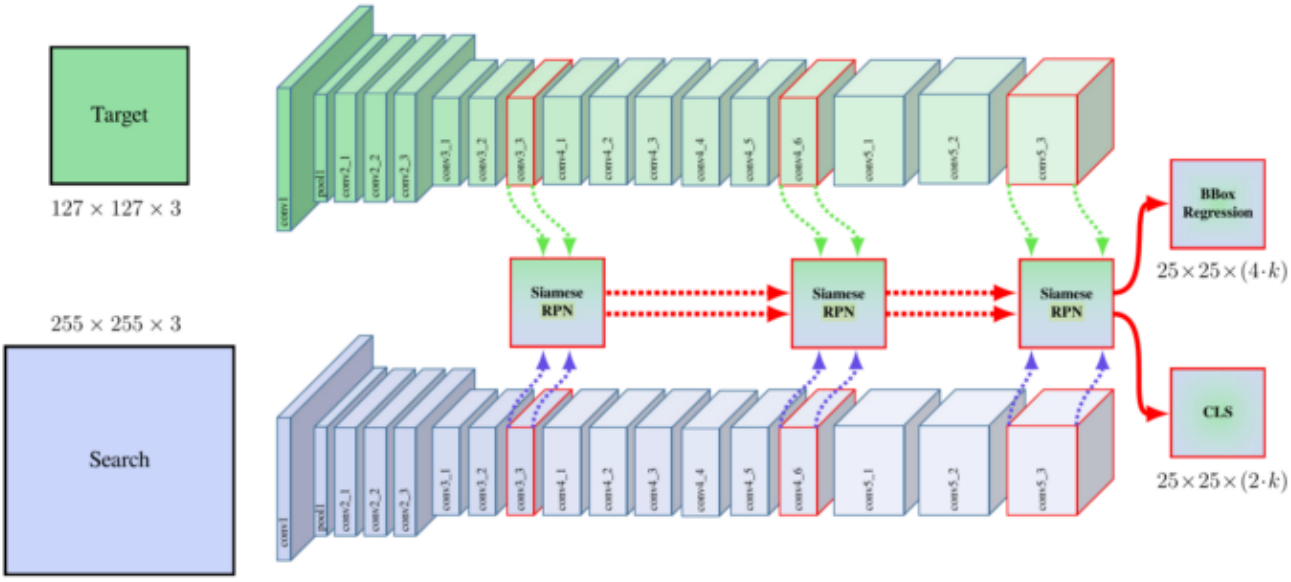


Fig 4. SiamRPN++ network architecture, image from paper [3]

A ResNet backbone replaces the original Alexnet backbone in SiamRPN. Noticing that padding destroys the spatial in-variance in SiamRPN it is done away with it in SiamRPN++.

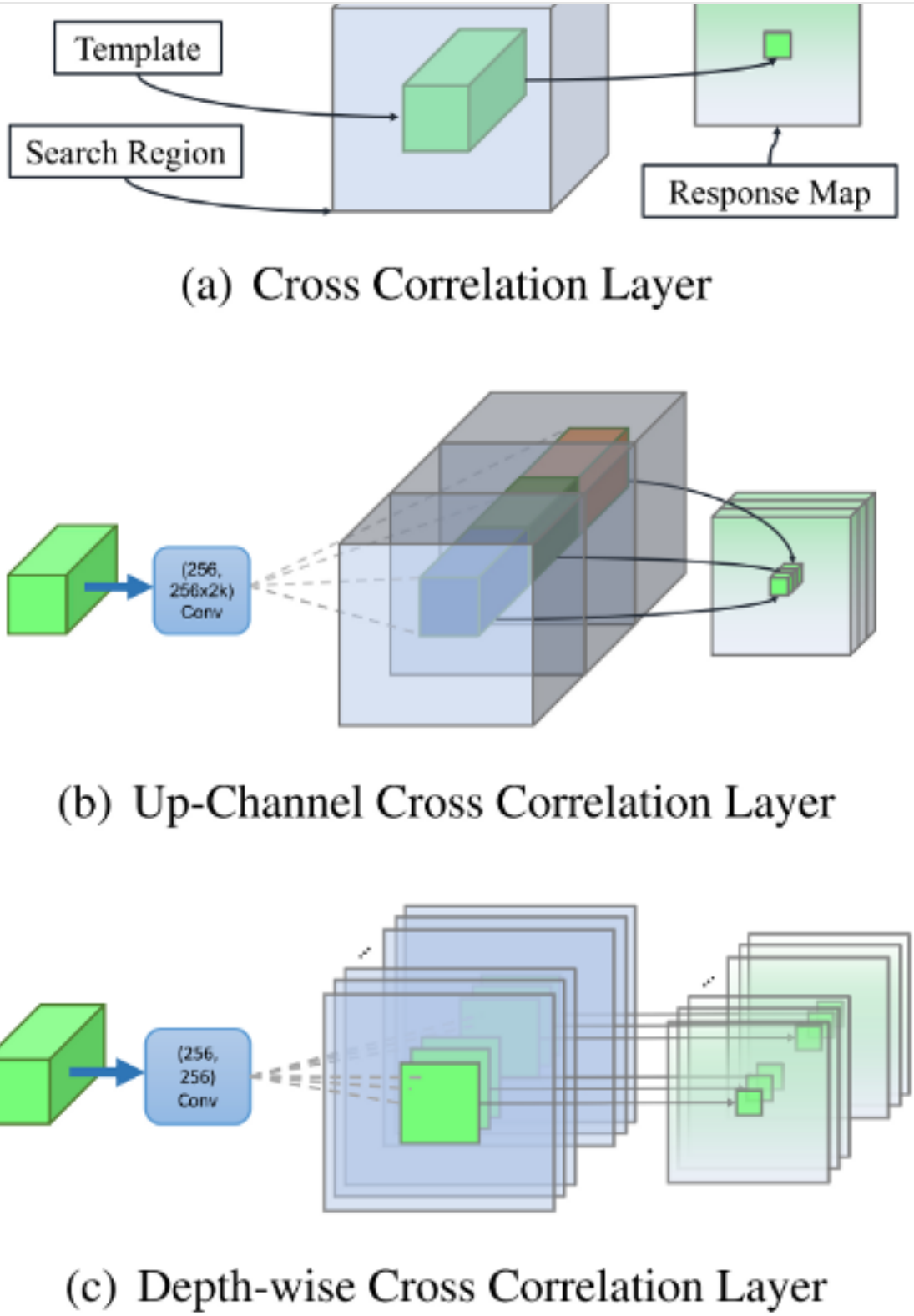


Fig.5 a,b,c show cross-correlation operation in SiamFC, SiamRPN and SiamRPN++ respectively. Image from SiamRPN++ paper [3]

SiamRPN's two Siamese branches are highly imbalanced in terms of parameters which degrades results. To improve this, the previous grouped cross-correlation operation (Fig.b in left image) is replaced with a depthwise cross-correlation operation (Fig. c in

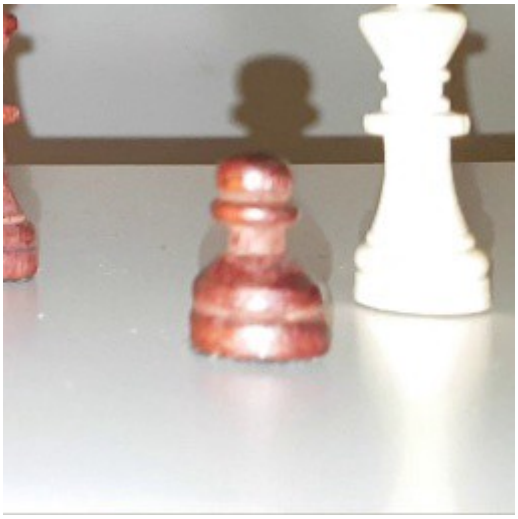
characteristics compared to SiamRPN's UpChannel (or the "grouped" style) cross-correlation.

Finally, multiple RPN blocks are used to produce proposals at different stages of the network as seen in Fig.4 (sort of a pyramid of RPNs!) and the results are fused to produce the objectness logits and bounding box regressions.

Dataset and training

I have used the COCO2017 dataset for training while other datasets like Youtube bounding box dataset, VOT2016 etc are also suitable. The dataset is prepared so that search and template images are always centered around the ground truth bounding box. Search images are cropped to 255 * 255 and template images are 127 * 127 pixels. Some context area is added around the actual bounding box before cropping and all remainder pixels are turned off.

Sample template crop



Sample search crop

Left images show examples of a search crop and template crop. Please refer the **code** for details of dataset preparation.

template. Losses for RPN network is binary cross entropy for class logits and smooth L1 is used for localization loss (similar to FasterRCNN). Training for around 14–15 epochs (4 hours on a Nvidia 2080) already produces a good enough model although training longer is likely to improve results.

Tracking

In tracking mode, the object of interest is highlighted using a bounding box in the first frame of the video. Search and template crops are done similar to training. An additional update step centers the search crop for the current frame using the bounding box prediction from the previous frame. The initial template is left unchanged throughout the video.

Conclusion

In this post, we dived in the world of single object tracking using Siamese networks discussing some of the SOTA approaches. Additionally, the companion code repository builds a object tracker from scratch using PyTorch's detectron2 framework, so do check it out!

Thank you for reading, I hope you enjoyed the post! Consider leaving claps or comments if this post was helpful, that's the only i know if my efforts are reaching you :)

References

1. Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. S. (2016). *SiamFC: Fully-Convolutional Siamese Networks for Object Tracking*.

2. Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). SiamRPN: High Performance Visual Tracking with Siamese Region Proposal Network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2018.00935

3. Li, B., Wu, W., Wang, Q., Zhang, F., Xing NLPR, J., & Yan SenseTime Research, J. (n.d.). *SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks*. Retrieved March 1, 2020, from https://lb1100.github.io/SiamRPN++.

4. https://github.com/STVIR/pysot