Visual cues-based object detection and localization for pick and place robots

Lokesh Bansal

Table of contents

Introduction

Introduction

- Goals: Pick and place robot
  - The robot must be able to operate in an unstructured environment.
  - Object of interest must be inferred based on cues provided by humans.
  - The object in question can be unseen.
- Challenges
  - Object identification with visual cues
  - Recognition of the object in the environment
  - Localization for robot operation

Vision Layers

- Neural Networks
  - Convolutional neural layers to embed input space into spatially local feature space
  - Siamese network architecture used to compare feature extractions
  - Spatial attention mechanism for score map in input space
- Template Matching
  - Searching and finding a template image in a larger image
  - Comparisons occur by sampling in image space

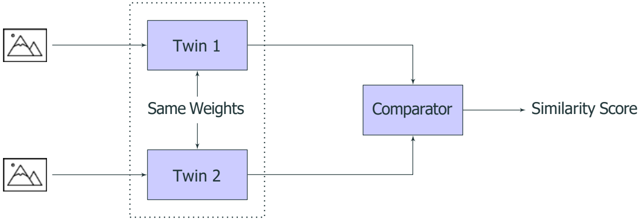
Literature

- Implementation
  - Pipelined Approach - Fine tuning, strong supervision
  - End-to-end architecture - weak supervision
- Methods
  - Object Recognition and Localization
    - Cues - Laser pointer, hand recognition [1], natural language
    - Perception - Neural networks, template matching, background subtraction [2]
  - One-Shot learning
    - Siamese networks [1]
    - Meta-Learning [4]

Building Blocks

Siamese Networks

- Siamese network consists of two identical neural network blocks.
- This architecture is used for comparing two objects.



Architecture of siamese network.

Siamese Networks

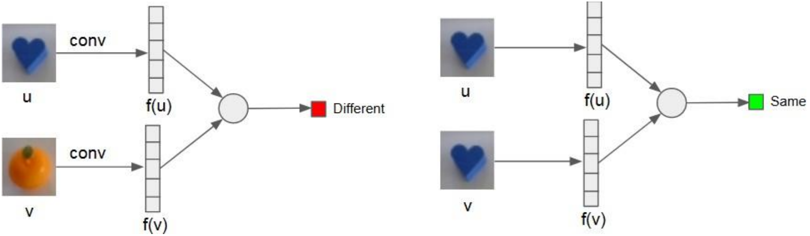
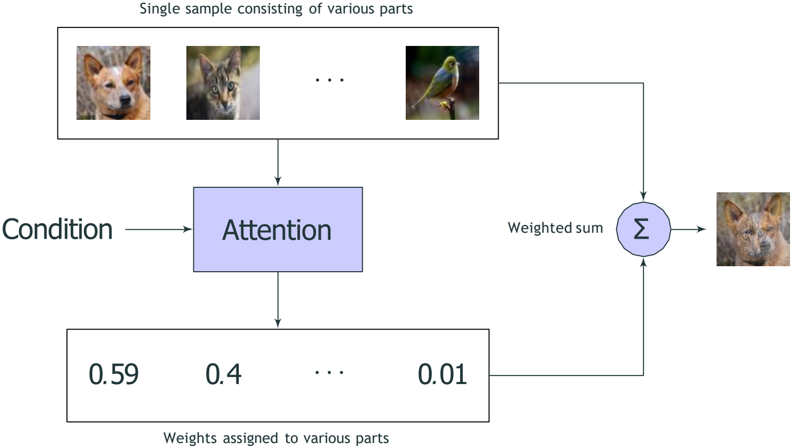


Illustration of a Siamese network comparing two images

Attention Mechanism

- In psychology, attention is selectively concentrating on some stimuli and ignoring other stimuli.
- Similarly, in deep learning, attention is giving high weightage to the features or parts of data that are useful.
- Attention mechanism got popularized in the context of machine translation[5].
- But attention mechanism can be used in any domain.

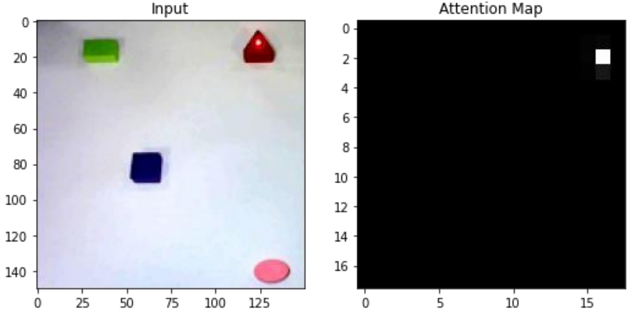
Attention Mechanism - General Form



General form of attention mechanism.

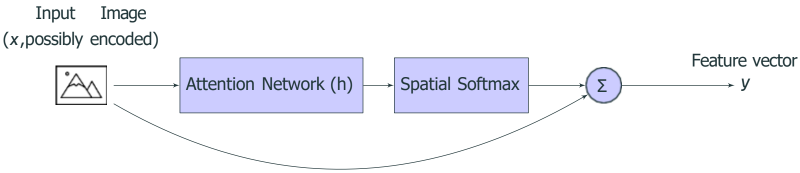
Spatial Attention

- Attend to the laser pointer.



Spatial attention. The red dot corresponds to the laser pointer. And the bright point on attention map shows the region where the network concentrates.

Spatial Attention - Working



Working of spatial attention.

Spatial Attention - Working

$$a_{i,j} = h(x_{i,j})$$ Scoremap

$$\alpha_{i,j} = softmax_{i,j}(a)$$ Attention map

$$y = \sum_{i,j} \alpha_{i,j} x_{i,j}$$ Feature vector

where

$$softmax_{i,j}(a) = \frac{e^{a_{i,j}}}{\sum_{i,j} e^{a_{i,j}}}$$

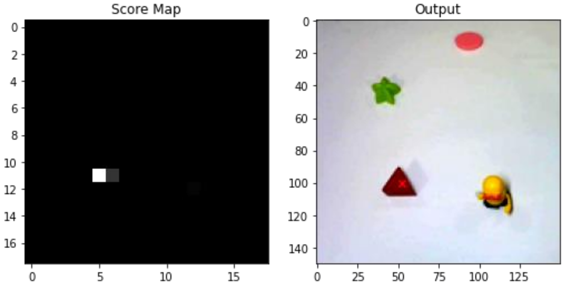
SoftArgMax

- Consider the problem of finding (x,y) position of an object in the image.
- We can directly regress the value of (x,y) but it will take a lot of time to train and it may not produce accurate output.
- Can we do better?

SoftArgMax

- We can do better by exploiting the fact that 2D CNN preserves spatial dependency.
- In softargmax, we will create a score map where high score corresponds to the presence of the object of interest.

SoftArgMax



Normalized score map. The output image is generated by plotting the predicted point on the input image.

SoftArgMax

- The softargmax regressor can be expressed as[1],

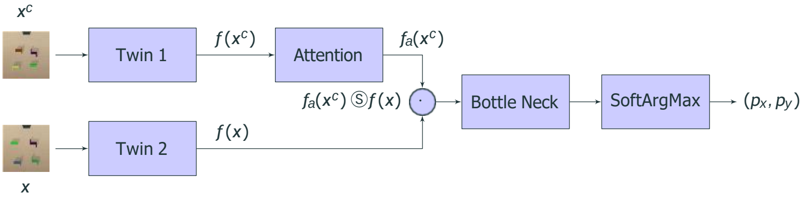
$$\beta_{i,j} = softmax_{i,j}(\varphi) \qquad \varphi \text{ is the scoremap}$$
$$p_x = \sum_{i,j} \beta_{i,j} \boldsymbol{i}$$
$$p_y = \sum_{i,j} \beta_{i,j} \boldsymbol{j}$$

- Why softargmax instead of argmax? - Softargmax is differentiable.

Network Architecture

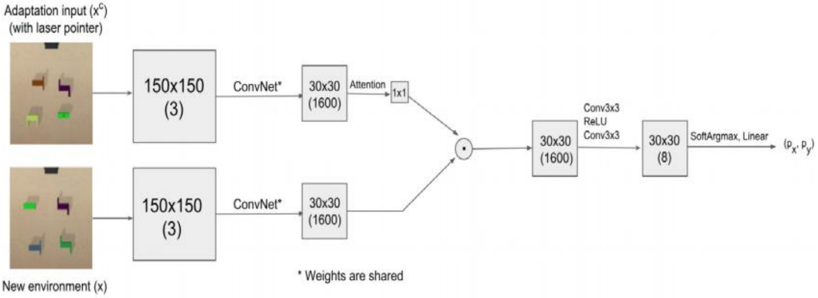
NETWORK ARCHITECTURE

- The overall architecture.



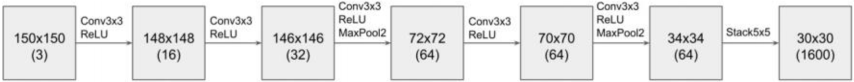
Network Architecture

- The network architecture used in the paper.



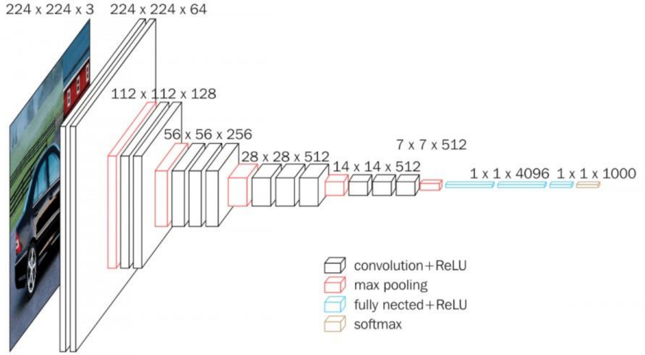
Network Architecture

- The network architecture(Siamese block) used in the paper.



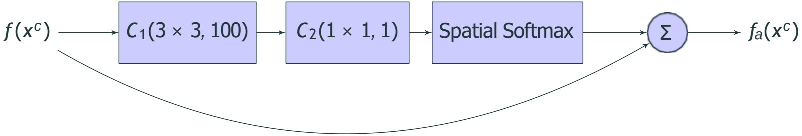
NETWORK ARCHITECTURE

- We have used an VGG16 for the Siamese Block.



NETWORK ARCHITECTURE

- Attention Network



Attention network we used. C_1 and C_2 are 2D convolutional layers. The values inside the bracket represents kernel size and number of output channels.

NETWORK ARCHITECTURE

- Bottle Neck Layers

$$f_a(x^c) \circledS f(x) \longrightarrow \boxed{C_1(3 \times 3, 100)} \longrightarrow \boxed{C_2(1 \times 1, 1)} \longrightarrow \varphi(x^c, x)$$

The bottle neck layers. C_1 and C_2 are 2D convolutional layers. The values inside the brackets represents kernel size and number of output channels.

Network Architecture

- The network architecture of the VGG network.

Model: "model_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808

block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

Experiments and Results

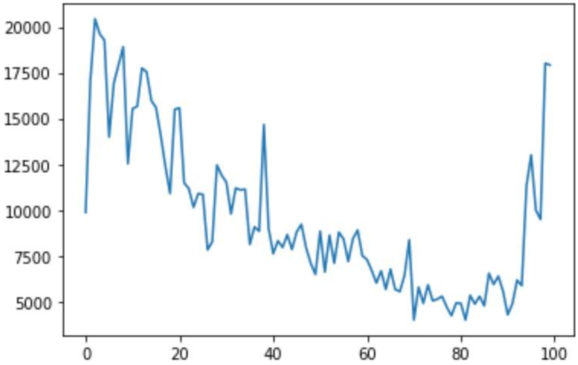
Experiments - Setup and Results

We performed two experiments in our Project.

- Object localization using Template Matching technique in pybullet.
- One shot object localization using Siamese Networks and MotoMini Robot with a custom gripper to pick the object.

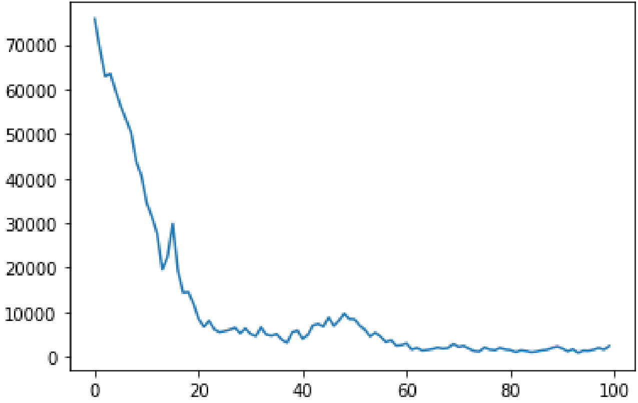
One shot Object Localization - Neural Network

When we use the Siamese network architecture given in the reference paper and train the complete model using MSE loss we observe that the loss is not minimized even after training for 100 epochs.

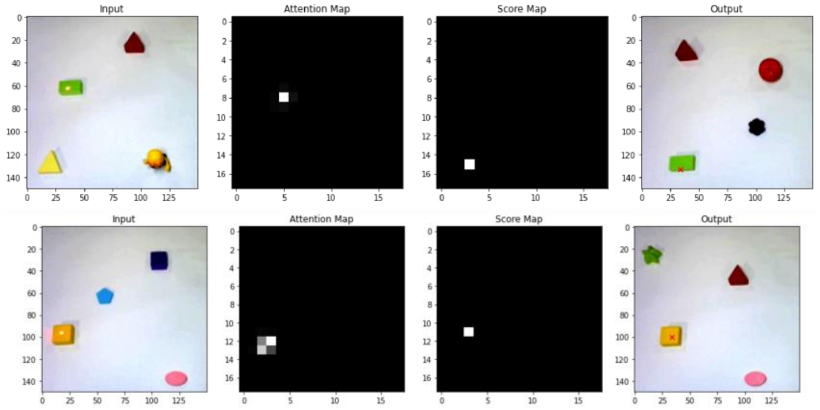


One shot Object Localization - VGG Net

To obtain faster convergence we used a VGG net that is pretrained on Imagenet dataset as twin network. However we only initialize the weights of the VGG with the pretrained model but still update the weights during training as that of general NN.



Output of the trained model



Output of our model on a sample test cases.

References

1 Sagar Venkatesh Gubbi and Bharadwaj Amrutur.
One-shot object localization using learnt visual cues via siamese networks.
In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2019, Macau.*, 2019.

2 Gary Bradski and Adrian Kaehler.
Learning OpenCV: Computer vision with the OpenCV library.
" O'Reilly Media, Inc.", 2008.

3 Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel.
End-to-end training of deep visuomotor policies, 2016.

References

4 Chelsea Finn, Pieter Abbeel, and Sergey Levine. **Model-agnostic meta-learning for fast adaptation of deep networks, 2017.**

5 Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. **Neural machine translation by jointly learning to align and translate, 2016.**

Thanks!