Where's Waldo?
A Deep Learning approach to Template Matching

Thomas Hossler
Department of Geological Sciences
Stanford University
thossler@stanford.edu

Abstract

We propose a new approach to Template Matching using Convolutional Neural Networks (CNN). Two CNNs are simultaneously trained on the template image and the query image. The spatial information of the template image is carried to the query image through an Hypernetwork. The network can accurately locate templates and equal classic computer vision methods, such as features based techniques. This Deep Learning approach is promising and will be further developed.

1. Introduction

Template matching is a common computer vision challenge where an algorithm is trying to find similarities between two or more different images. The task is complex due to the potential deformations (scaling, rotation, illumination) of the template in the image and has been historically performed by classic computer vision algorithm [5, 8, 1, 3].

In this paper, a Deep Learning approach to the template matching challenge is presented. The long-term objective is to train an algorithm to recognize retail store product in a shopping cart. The first step, presented in this paper, is the creation of a Deep Learning architecture that can match and potentially outperform classic computer vision approaches. Using CNNs trained on the template image and the query image, the algorithm identifies products on shelves. Using this approach, we are hoping to carry enough spatial information to overcome the inherent difficulties of the template matching challenge.

The template matching problem can be formulated as follow: given a query image and a template, the algorithm outputs the location(s) of the template on the query image, as presented Figure 1. One advantage of our approach is the format of the output, which will give a class score for each pixel (part of the template or not).

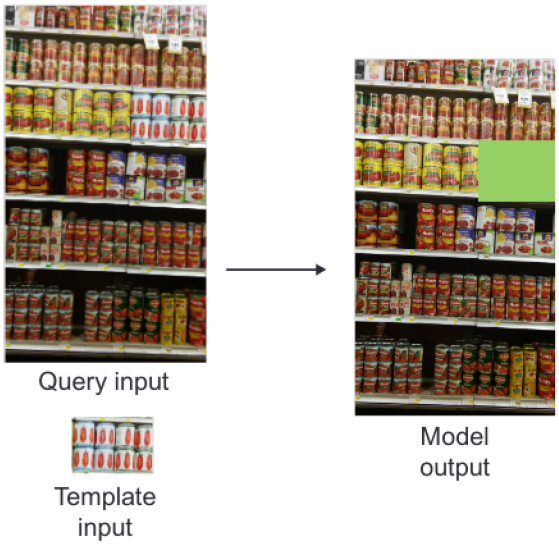

Figure 1. Template matching example. The algorithm localizes the template on the query image and output a binary heat map. The template image is padded to match VGG minimum input size.

In this paper, a first algorithm is trained to successfully perform the product finding task, which confirms our approach. Related Computer Vision and Deep Learning work is presented in the next section, followed by the description of the CNN built for the task. The CNN results are then compared with a SIFT algorithm and discussed.

2. Related work

Several approaches have been taken in Computer Vision to tackle the Template Matching Task. Hashemi [5] gives a non-exhaustive list of the different methods. Naive approaches, such as convolving the template over the query image, are limited by scaling, rotation and other transformations.

More recent techniques, such as Scale-Invariant Features Transform (SIFT) [8] uses key-points to match the template

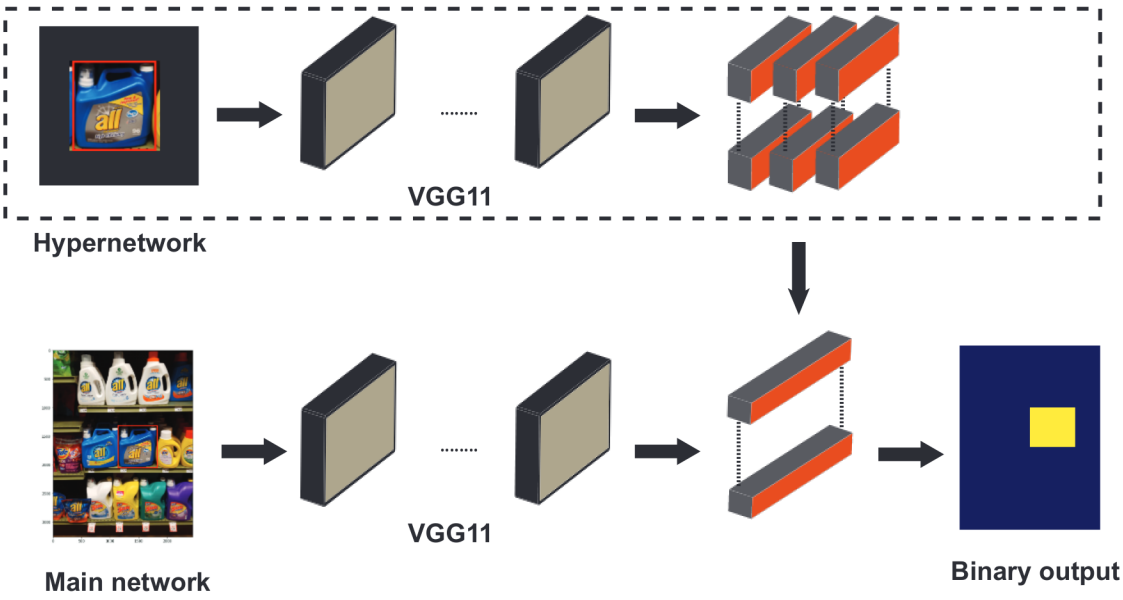

Figure 2. Architecture of the CNN implemented. The Hypernetwork take a padded template in order to maintain the input size constant. The output of the last convolution layer is reshaped and fed as weight for the last convolution layer of the main network. In this Figure, VGG-11 is used for both part of the network.

and the query image. Although keypointbased approaches have shown some success for template matching, 2D keypoints are unreliable for recognizing untextured objects or non-planar objects when the viewpoint is changed by more than 25 degrees [9].

Deep-Learning has been used in a similar context (retail stores product recognition) but focused on single instance recognition [7]. The algorithm, while outperforming SIFT, does not provide a full solution to the problem.

3. Method

3.1. Problem set up

CNN are made of several layers, with each neurons processing part of the input image. The spectrum of applications of CNN has grown exponentially over the years, from image captioning to identification[2]. Since CNN are so good at processing and extracting spatial information from images, two CNN have been implemented here. They are trained simultaneously, one is taking the template image as input and the other the query image. The last convolution layer of each network are connected using an Hypernetwork [4], a structure initially created to reduce the number of parameters of large CNN such as HyperNeat [11].

The Hypernetwork is set up to carry information from the template image to the query image. However, since the output of the Hypernetwork will be used for the weights of the last convolution layer of the main network, fully connected layers are necessary to reshape the tensors. This kernel is stored in a tensor $K \in \mathbb{R}^{N_{in} \times N_{out} \times f_{size} \times f_{size}}$ where $N_{in} \times N_{out}$ is the number of filter and f_{size} the size of the filer. For each channel output i of the Hypernetwork, the tensor is turned into a vector z_i. Follows a fully connected layer using weight $W_i \in \mathbb{R}^{d \times N_z}$ where d is an hyperparameter, the depth and N_z the length of the input vector z. Finally, the final layer takes a_i as an input and output $K_i \in \mathbb{R}^{N_{out} \times f_{size} \times f_{size}}$. The K_i are the concatenated to create the kernel K.

$$
\begin{aligned}
a_i &= W_i z_i + b_i && \forall i = 1, \cdots, N_{in} \\
K_i &= W_{out} a_i + B_{out} && \forall i = 1, \cdots, N_{in} \\
K &= \begin{pmatrix} K_1 & K_2 \cdots K_i \cdots K_{N_{in}} \end{pmatrix}
\end{aligned}
$$

Moreover, a computer vision algorithm using SIFT has been ran and its performance compared to the deep learning approach employed here.

3.2. Dataset

The data will be provided by Focal Systems, a company that applies Deep Learning and Computer Vision to Brick and Mortar retail. The training dataset consists in more than 86k labeled items on pictures of retail store shelves, such as presented Figure 1. Each item has the coordinates of its bounding box, its description and the url of the corresponding query images. The query images are hosted on

the cloud. Initially, the pictures were downloaded for each mini-batch, but it appeared that doing the data processing on the fly was making the algorithm too slow. The pictures were downloaded first and processed locally before loading them in the algorithm. Only images with the same aspect ratio have been selected and have all been resized to the shame shape. The template images have been padded in order to keep the dimension of the input of the Hypernetwork constant. For the rest of the paper, a train set of 8,000 images and a validation set of 2,000 images have been used. The inputs of the model are the query image, the template (2D slice of the query image using the item coordinates) and the "truth". The truth is a binary image, with ones where the template is located and zeros where it is not.

3.3. Network Details

The neural network created in this paper uses the PyTorch architecture. The network is presented Figure 2. In that case, the VGG-11 [10] neural network is used. Smaller networks made of five (convolution - ReLU - batchnorm) layer have also been used (for the overfitting experience especially). The template image is padded which has the double advantage of keeping N_z constant and not dropping the pooling layers of VGG. The model is either trained from scratch or, when VGG is used, the pretrained weights are used. The fully connected layers of VGG are removed and replace by the ones detailed previously. The entire network outputs two "images" of scores: for each pixel, the score of that pixel being a one or a zero. A Cross-Entropy Loss is employed. The accuracy is calculated by averaging over the class mismatch between the truth and the binary output for each pixel. An Adam optimizer with an initial learning of 0.001 and L2 regularization of 0.0005 have been selected.

4. Results

4.1. Overfit one image

In a first time, smaller CNNs have been built for the Hypernetwork and the main network. Two similar 15 layers CNN, made of five (convolution - batchnorm - ReLU) layers, has been used for both networks. Filter of size 3 have been used with a padding of 1. The model is trained from uniformly initialized weights on one example. The results are presented Figure 3.

As observed, the algorithm successfully locate the product, which proves the legitimacy of the approach chosen here. Even though the softmax output is presented here, one can get an intuition of the main interest of the CNN approach chosen here. Indeed, the CNN is creating heatmaps of class scores, which, using the softmax function, can be interpreted as probabilities. Therefore, an additional information is available, in opposition to Computer Vision approaches such as naive convolution.

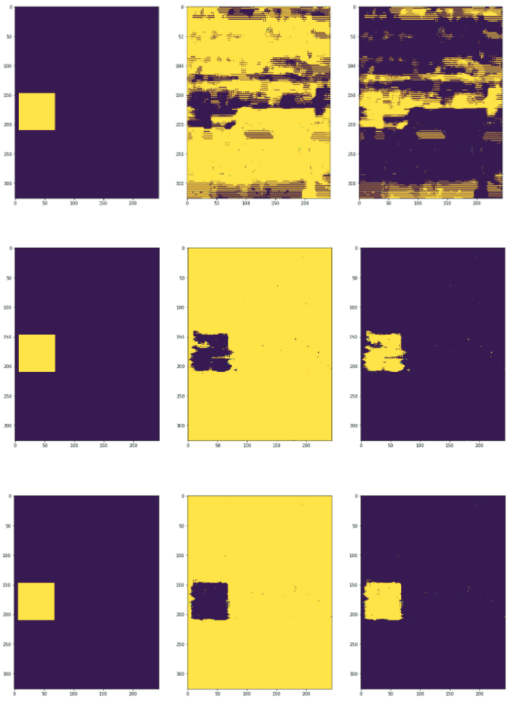

Figure 3. Example of overfit. The binary truth is located on the left. The scores maps (after softmax) are in the middle for class 0 and right for class 1. The three rows correspond to 100, 600 and 1,000 iterations.

4.2. Template matching using Deep Learning

In a second time, since the model is now trained on over 8,000 samples, the VGG-11 pretrained has been chosen for the Hypernetwork as well as the main network. The model is trained over five epochs, with a learning rate decay of 0.95 between each epoch. The loss function is presented Figure 4 and the train/validation curves are presented Figure 5.

A plateau is quickly reached (after two epochs) for the loss function and similar behavior is observed for the accuracy curves. The accuracy reached is above 95% and merely fluctuates for the next three epochs. The optimizer has been chosen according to the original Hypernetwork paper [4] but the shape of the loss curves shows that it might not be the optimal choice for this task. Therefore, other optimizer should be tested.

4.3. Template matching in Computer Vision

As mentioned earlier, the accuracy of the deep learning algorithm is compared to a more classic computer vision approach, the SIFT algorithm. The following algorithm has been implemented using the OpenCV library for Python.

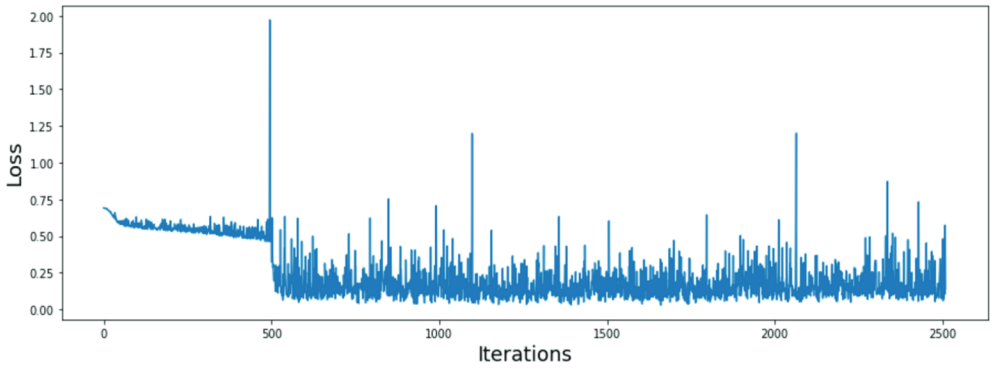

Figure 4. Cross-Entropy loss function for the CNN. An Adam optimizer has been used.

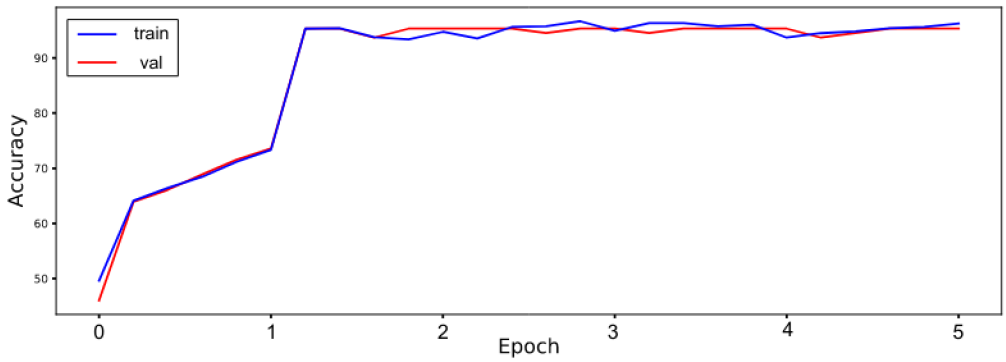

Figure 5. Training and Validation accuracy along five epochs. A plateau of 95% accuracy is reached after two epochs. 50% accuracy would be obtained with uniformly chosen class.

Using a SIFT detector, the keypoints and descriptors of the template and the query image have been obtained. Similar to Lowe's work, a ratio test using the two nearest neighbor has been implemented. The ratio between the closest and second closest neighbor allows to reject false matches while barely affecting correct matches [8]. Figure 6 shows an example of template matching using SIFT and a distance ratio greater than 0.5. We observe that, despite many similarities between the products on the shelves, the algorithm successfully identify the template.

Using the "good" matches of keypoints between the template image and the query image, bounding box can be created and compared with the binary truth. The mean accuracy is calculated on the validation set. The accuracy of the algorithm in function of the ratio is presented Figure 7. We observe that the accuracy dramatically drops when the ratio increases above 0.5. In his paper, Lowe rejected all the ratio over 0.8.

For ratio below 0.4, an accuracy of 95% is observed.

Figure 6. Template Matching using SIFT. A ratio of 0.5 has been used here.

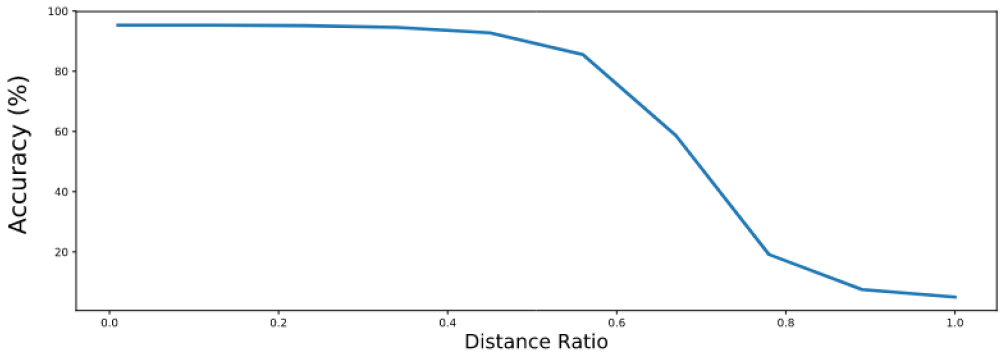

Figure 7. SIFT accuracy as a function of the test ratio (distance ratio of the two closest neighbor)

However, it drops to 86% for a ratio of 0.6 and down to 20% for a ratio of 0.8. It appears that the algorithm finds numerous similar keypoints, therefore the constraints must be increased to maintain accuracy.

The Hypernetwork framework performs similarly to SIFT when the right ratio is selected and outperforms it when the wrong ratio is picked.

5. Discussion and conclusion

Template matching has been a recurrent challenge in computer vision over the last decades and still remained unsolved. With the emergence of Convolutional Neural Networks, a new way of tackling the template matching task has been developed. The approach chosen in this paper is using an Hypernetwork to learn spatial information about the template. In this paper, the legitimacy of such approach has been proven by successfully identifying template on query images and matching/beating the accuracy of classic computer vision approaches. The impact of the depth and width of the networks or of some hyperparameters should be studied next. For example, the depth d of the first fully connected layer of the Hypernetwork greatly affects the amount of information transmitted from the template image to the query image. The impact of depth and width of the CNN on the accuracy should also be studied and models as VGG-16 or 19 should be implemented and compared. Moreover, the Deep Learning has the advantages of creating "heatmaps" of scores for each class. These heatmaps could be used to relax the algorithm in further work. For example, instead of precisely matching a "Pinot Gris 2015", the algorithm could identify all wines bottles on the picture. This could be achieved by adding regularization parameters to the model. Finally, this work should be combined with the work from [6] in order to create a Deep Learning framework that could match rotated, scaled or partially visible products.

Acknowledgements

The author thanks Focal Systems for providing the dataset as well as the GPU necessary to run the neural networks. Whereas this project has been single-handedly coded, the author thanks the people at Focal Systems for the always meaningful discussion on Deep Learning and CNNs.

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[2] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. *Nature*, 521:436–444, 2015.

[3] M. S. Choi and W. Y. Kim. A novel two stage template matching method for rotation and illumination invariance. *Pattern Recognition*, 35(1):119–129, 2002.

[4] D. Ha, A. Dai, and Q. V. Le. HyperNetworks. 2016.

[5] N. S. Hashemi, R. B. Aghdam, A. S. B. Ghiasi, and P. Fatemi. Template matching advances and applications in image analysis. *arXiv preprint arXiv:1610.07231*, 2016.

[6] D. Held, S. Thrun, and S. Savarese. Deep Learning for Single-View Instance Recognition. *arXiv preprint*, page 16, 2015.

[7] D. Held, S. Thrun, and S. Savarese. Robust Single-View Instance Recognition. *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2152–2159, 2016.

[8] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[9] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. *International journal of computer vision*, 73(3):263–284, 2007.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

A. Data processing

A significant amount of work has been done in preprocessing the images for the CNN. Prior to working on the images, a json file containing information about each item has been created from a Mongo Database. Then the preprocessing of the images was made of several steps. First of all, since the input are minibatches of constant sizes, the images had to filtered by aspect ratio. Only the images with an aspect ratio of $\frac{4}{3}$ (height over width) have been used. All the images have been rescaled to a size of $326 \times 244 \times 3$. The templates have the additional challenges that their shape is not constant and varies too much along the data set. Since the smallest input dimensions for VGG are $224 \times 224 \times 3$, this size has been chosen. The templates which sizes did not match these dimensions have padded with zeros as shown Figure 2. Finally, due to the nature of the task, a data loader class had to be written. Instead of outputing an image and the corresponding target class, the data loader outputs three images: the query, the template and the binary truth.