Route and Track: Siamese Capsule Networks for Object tracking

Sai Ajay Modukuri, Chandu Dasari, Murali Nandan Nagarapu, Chandan Reddy Akiti
Department of Computer Science
Pennsylvania State University
{svm6277, cbd5416, mzn5322, cra5302}@psu.edu

1 Introduction

Siamese trackers proved useful in object tracking in the prior research work (Bertinetto et al., 2016), (Li et al., 2018). SiamFC uses a Convolutional Neural Network (CNN) to learn the similarity of exemplar image with search image proposals.

CNN is composed of convolution filters that are essential for matching patterns in the input. Convolution filters used in the SiamFC tracker provide a robust property of translation invariance. Sabour et al. (2017) hypothesized that convolutional filters, however, may fail to capture the spatial relationship between the features in the tracking object. Sabour et al. (2017) proposed Capsule Networks to address spatial relationships. The capsule networks attain high performance on the MNIST dataset. Moreover, Sabour et al. (2017) also proved that they provide high spatial correlation with overlapping MNIST digits in Multi-MNIST data. We hypothesize that Capsule Network in object trackers would help us learn the spatial relationship between the exemplar image and the search image, thus providing a better tracking performance. We call our proposed tracker as RAT tracker. The code can be found at https://github.com/chandan047/RAT-Tracker

2 Motivation

Object tracking is a highly sought out research area. Its applications are far-fetching in various areas like surveillance and security, traffic monitoring, video communication, robot vision, and animation.

The earlier methods for object tracking used an online approach by performing Stochastic Gradient Descent (SGD) on the video to track. This is not only painstakingly slow but also prone to errors. Siamese Networks address this issue by learning a general similarity in the training phase and applies this learned similarity to track objects in the offline phase.

Though fully convolutional Siamese networks have proven to be highly successful in object tracking, they fail to capture spatial relationships. This could hinder SiamFC from tracking spatial relationships of the object to be tracked. It may also make SiamFC prone to adversarial attacks and may also degrade performance. To address this, we propose a variation of SiamFC in which capsules follow convolutional filters. This variation is a first step towards building a mature capsule network in further research.

The motivation of the problem is to try to improve the performance or explainability of standard traditional signal object tracking like SiameseFC by using traditional computer vision single object tracking methods or any new methods. The end goal of the traditional SiameseFC is to calculate the similarity between the embedding from the image frame and the exemplary image.

We opt to find similarity between embeddings by using the dynamic routing algorithm presented in (Sabour et al., 2017). The dynamic routing algorithm provides a weighted mapping of capsules from a lower layer to a higher layer such that highly correlated capsules are weighted more.

3 Related Work

Object tracking started long back with the statistical models for bounding box predictions and object movement. Moreover, with the inception of big data and high computing powers, deep learning models have shown promising results in learning huge datasets with ease, and one among those is the one-shot learning introduced for signature verification (Koch, 2015). SiamFC (Bertinetto et al., 2016) was introduced for object tracking in the VOT'17 challenge to set benchmarks for object tracking. Consequently, much research was started with one-shot learning using deep learning models like Fast CNN's. For example, (El-Shafie et al., 2019) proposed the interpolation of deep features. (Li et al., 2019) devised methods to tackle when the object falls into some occlusion such that the bounding box would fall into uncertainty in the paper of re-detection.

Capsule Nets have increasingly gained momentum in object classification as an alternative to native CNN layers wherein convolution filters undermine the spatial relationship between the features, as explained in the first paper (Sabour et al., 2017).

Models involving capsule layers tend to comprehend better feature relations. (Amara, 2018) paper uses these capsule nets to detect traffic signs with high performance. (Wu et al., 2020) provides face recognition based on the encoding of capsule representations. (Hinton et al., 2018) further improved the dynamic routing algorithm using Expectation-Maximization (EM) Routing. (Ghofrani and Mahdian, 2019) provides digit recognition with EM Routing for better statistical routing.

Furthermore, (Ren et al., 2019) came up with adaptive routing between capsules where the dynamic routing of all capsules are further simplified to approximate the original dynamic routing proposed in (Sabour et al., 2017). DCNet (Phaye et al., 2018) introduced the dense capsule layers for diversified learning. Complete list of literature on capsule networks is available here. Our proposed method is an attempt to apply the capsule networks and measure the competency with the existing SiamFC baseline.

4 Approach

We plan to use Siamese Capsule Network (Neill, 2018) for similarity learning between the exemplar and search images.

A capsule is a set of neurons that individually activate for various properties of a type of objects, such as position, size, and hue. Formally, a capsule is a set of neurons that collectively produce an activity vector with one element for each neuron to hold that neuron's instantiated value. Graphics programs use instantiation value to draw an object. Capsule networks attempt to derive these from their input. The probability of the entity's presence in a specific input is the vector's length, while the vector's orientation quantifies the capsule's properties.

To test our hypothesis, we add a capsule network on top of the AlexNet layers in the SiamFC model of (Bertinetto et al., 2016). Precisely, we direct the output of AlexNet to the capsule network layer. The capsule network outputs the capsule embedding of the exemplar image and search image. We hypothesize that introducing a capsule network in the network enables learning spatial information between the features extracted by the AlexNet (Krizhevsky et al., 2012). We conduct several experiments with mathematical and graphical intuitions to device a model. Further sections detail the model architecture and hyperparameters used.

5 Model

Our proposed model is built on top of the existing Siamese network, as shown in figure 2. At the core, it uses AlexNet (Krizhevsky et al., 2012) as the backbone, followed by a capsule network (Sabour et al., 2017). The Capsule network has a primary-caps layer followed by a pseudo-class-caps layer. The Capsule network outputs a map of scores in the form of vectors for pseudo-classes. We refer to them as pseudo-classes because there is no mapping of the capsule network output to a class set.

Figure 1: Capsule network output. This example figure shows the capsule network output of 5 capsules for 5-pseudo classes. In our architecture we output 16 capsules.

The search and the exemplar image sizes for the training phase are $(255 \times 255 \times 3)$ and $(127 \times 127 \times 3)$ respectively. The corresponding sizes of AlexNet outputs A_S and A_e are $(22 \times 22 \times 128)$ and $(6 \times 6 \times 128)$. We transform A_S by unfolding patches of (6×6), we represent this form by A_s and it is formed by 17×17 patches in total. The dimensions of A_s are thus $(289 \times 6 \times 6 \times 128)$.

The Capsule network is composed of two layers. We add first layer with 8 1×1 kernels of 2D-convolution layer on top of A_s and A_e. The output of this layer acts as the primary-layer $\hat{u}$ of the capsule network. We then add the dynamic routing layer on top of the primary-capsule layer that

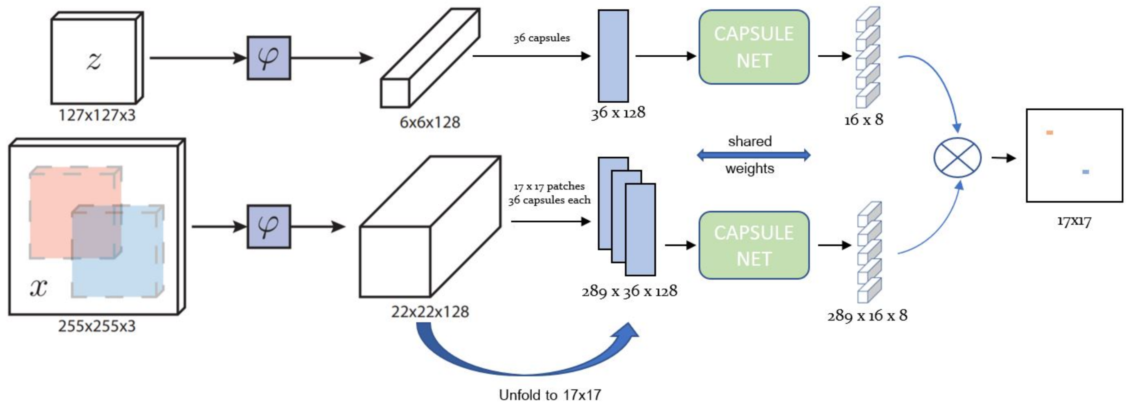

Figure 2: Route and Track: Siamese Capsule for Object Tracking architecture: Our model uses AlexNet as backbone, followed by a Capsule Network with 16 output capsules. All weights are shared between exemplar and search image.

outputs v, which we referred to as the pseudo-class layer. The dynamic routing layer is an iterative module that assigns weights to the capsules in the lower layer to route to the upper layers.

Ren et al. (2019) presented a simplified form of dynamic routing algorithm called adaptive routing that approximates the behavior of the model presented in Sabour et al. (2017). We thus used the adaptive routing algorithm to route the capsules in the primary-layer to the pseudo-class layer. The primary advantage is the reduction in the number of parameters to the otherwise large capsule network. The algorithm from Ren et al. (2019) is cited in figure 3.

Algorithm 1 Adaptive algorithm.

```
1: procedure ROUTING($\hat{u}_{j|i}$, $r$, $l$)
2:     capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$
3:     for $r$ iterations do
4:         $s_j \leftarrow \sum_i \hat{u}_{j|i}$
5:         $v_j \leftarrow \mathtt{squash}(s_j)$
6:         $\hat{u}_{j|i} \leftarrow v_j + \hat{u}_{j|i}$
7:     end for
8:     return $v_j$
9: end procedure
```

Figure 3: Adaptive routing algorithm from Ren et al. (2019). $\hat{u}_{j,i}$ is the capsule from lower layer and v_j is the output of adaptive routing algorithm. This algorithm is an approximation of dynamic routing algorithm with lesser parameters.

The output of the exemplar image v_e has the size of (16×8) and the output of the search image v_s has the dimension of $(289 \times 16 \times 8)$ as shown in figure 2.

The correlation of v_e and v_s is measured using the same methodology of convolution. That is, (16×8) maps of v_e and v_s correlate as the element-wise multiplication normalized to the size of the maps.

6 Loss functions

For our problem, we delved into the working of different loss functions that suit our architecture. Even though basic cross-entropy losses did work out better for vanilla SiamFC, the gradients computed showed little promise even when trained for longer epochs. Thus, a variety of loss functions have been experimented with our combined model with capsule networks. Understanding the intuition behind the working of loss functions is key for the experiments that are to be run in on the dataset. The capsule layer induces a large number of parameters into the model creating a spike in the degrees of freedom of the whole architecture.

6.1 Balanced Loss

The most commonly used cross-entropy loss has to give some more weighting to the class imbalance prior to incorporating in computing the gradients. We have known the cross-entropy loss defined as follows-

$$CE(x,p) = \begin{cases} -\log(p), & \text{if } y = 1 \\ -\log(1-p), & \text{otherwise} \end{cases}$$

This is now incorporated by a weighting factor $\alpha \in [0,1]$ for classes 1 and $1-\alpha$ corresponding to the class 0. Here α is treated as a hyper-parameter that is set by the cross-validation. The final loss now becomes-

$$CE(p_t) = -\alpha_t \log(p_t)$$

6.2 Focal Loss

When there is a high imbalance between foreground and background classes, focal loss performs a better gradient calculation and updates. This was first introduced in dense object detection(Lin et al., 2020). A major setback of cross-entropy loss when it comes to such scenarios is classes that can be easily classified do undergo a loss due to background in the image. And this non-triviality over many examples does sum up and can trip over in rare cases. Focal loss adds a modulating factor $(1 - p_t)^\gamma$ in addition to the cross-entropy where γ is a tunable hyper-parameter. The formal definition of focal loss is given by-

$$L(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

A couple of noteworthy aspects of this loss is that firstly, the loss is unaffected when the misclassification occurs and p_t is small. Secondly, the hyper-parameter adjusts smoothly as the model keeps down weighting the easy examples.

6.3 Margin Loss

Margin loss tries to push the outputs towards their consecutive polarities. Margin Loss proved quite useful in support vector machines. For our problem, the margin loss calculated on the map has worked quite effectively. The loss converges quickly when used with a capsule network.

$$L(p_t) = \frac{1}{N} \sum_t (y_t(m^+ - p_t) + (1 - y_t)(p_t - m^-))$$

where m^+, m^- are the margins.

We show how loss converging in our training when margin loss is used.

7 Experiments

We will use VOT 2018 dataset for the project. The baselines are available for various trackers on the VOT (vot). Evaluation Toolkit (eva) is available to run the tests. We will use two evaluation metrics popularized for VOT 2018 dataset.

- Expected Average Overlap (EAO)
- Accuracy
- Robustness

These metrics are provided by VOT 2018 Toolkit. We have conducted a handful of experiments with many variations to our base architecture (i.e., Siamese Network with AlexNet as backbone followed by Capsule Network). In this section, we will be explaining some of the main ideas in terms or architecture/approaches and the results and analysis related to them.

7.1 Experiment 1

The initial experiment setup consists of an AlexNet followed by a Capsule Network with two capsules. The output feature maps extracted by AlexNet are fed into the capsule network in which, one Capsule is responsible for routing similar and dissimilar patches of feature maps. More formally, both the search image and the exemplar images are passed into the AlexNet as follows-

$$A_x = \varphi(X)$$

$$A_z = \varphi(Z)$$

Where the AlexNet takes an input of search image X of shape (255 $\times$ 255 $\times$ 3) and an exemplar image Z of shape (127 $\times$ 127 $\times$ 3) separately and produces the corresponding feature maps. These extracted features of search and exemplar images from AlexNet are reshaped into (22$\times$22) and (6$\times$6) dimensions, respectively. Now, initially, the reshaped search image features are unfolded using a (6 $\times$ 6) patch to obtain 289 (6 $\times$ 6) patches. The unfolded feature patches both of search and exemplar are concatenated to give a (6 $\times$ 12) feature patches and consequently passed to our Capsule Network. The capsule network outputs a vector representation of feature space with shape of (289 $\times$ 8). The euclidean distance of each of the 8D vectors in the (289 $\times$ 8) representations are calculated, resulting in a vector of 289 dimensions. Each of the scores in the resultant matrix corresponds to a patch in the original image that was divided as (17 $\times$ 17) patch. Loss is calculated as Root Mean Squared Error (RMSE) between the resultant vector and the ground truth map of (17 $\times$ 17) calculated for the search image.

$$F_{xz} = Concat(A_x, A_z)$$

$$C_{xz} = \psi(F_{xz})$$

$$x = \|C_{xz}\|_2$$

$$Loss = \sqrt{(\frac{1}{289})\sum_{i=1}^{289}(y_i - x_i)^2}$$

The whole experiment is run for 50 epochs, wherein Stochastic Gradient Descent (SGD) is used as the optimizer (Bottou, 2010).

7.2 Experiment 2

From the analysis of results from Experiment 1, we hypothesized that feature concatenation is not an effective way to utilize capsule network features. To address those issues, in this experiment, the outputs from AlexNet A_x, A_z are passed to a Capsule Network separately. Instead of concatenating feature maps from AlexNet, we use Capsule Network as a feature extractor to produce C_x, C_z. The Capsule Network has been modified to have 16 output capsules to output 16 16×8 vectors.

$$A_x = \varphi(X)$$
$$A_z = \varphi(Z)$$

$$C_x = \psi_K(A_x)$$
$$C_z = \psi_L(A_z)$$

$$Loss = \|C_x - C_z\|_2$$

Loss function has been modified to an L2 norm between the capsule outputs of exemplar and search images i.em C_x and C_z. We have experimented with both SGD (Bottou, 2010) and Adam (Kingma and Ba, 2014) optimizers.

Although this experiment failed to converge after ten epochs, it gave us a clear direction and insights about the architecture to be used and also the behavior of various loss functions.

7.3 Experiment 3

In this experiment, we built and improved upon the architecture from Experiment 2. We follow the same network architecture as of Experiment 2, i.e, AlexNet followed by a Capsule Network to give C_x and C_z. However, a major distinction is the introduction of shared weights, even in the Capsule Network phase.

$$A_x = \varphi(X)$$
$$A_z = \varphi(Z)$$

$$C_x = \psi(A_x)$$
$$C_z = \psi(A_z)$$

The Capsule Network has 16 output capsules similar to Experiment 2 and the weights for both search and exemplar feature maps. This output from shared weight Capsule Network produces $C_x = [C_{x1}, ..., C_{x16}]$ and $C_z = [C_{z1}, ..., C_{z16}]$.

The final loss is computed as a mean of differences between the output feature map and the ground truth.

$$Loss = Mean(\|y - Correlation(C_{xi} - C_{zi})\|_2)$$

However, in the experiments, we found that the gradients were of the order of 10^{-7} at the highest layer of AlexNet. Thus it is necessary to backpropagate the loss to AlexNet layers through interlayer cross-correlation, which is explained in the next session.

7.4 Experiment 4

Having observed the problem with vanishing gradients, we modified the network to combine two different losses. The new loss is a function of loss at the end of AlexNet, i.e., convolutional head (loss at the end of AlexNet) and loss calculated at the end of the Capsule Network, i.e., capsule head.

$$A_x = \varphi(X)$$
$$A_z = \varphi(Z)$$

$$L1 = ||(y - Correlation(A_x, A_z))||_2$$

$$C_x = \psi_K(A_x)$$
$$C_z = \psi_L(A_z)$$

$$L2 = ||(y - Correlation(C_x, C_z))||_2$$

$$Loss = \alpha(L1) + \beta(L2)$$

The α, β values can be tuned such that their sum equals to one i.e, $\alpha + \beta = 1$. This method of combining and propagating loss at convolution head and Capsule heads solved the issue of vanishing gradients problem. The remaining part of the network is the same as the network in Experiment 3.

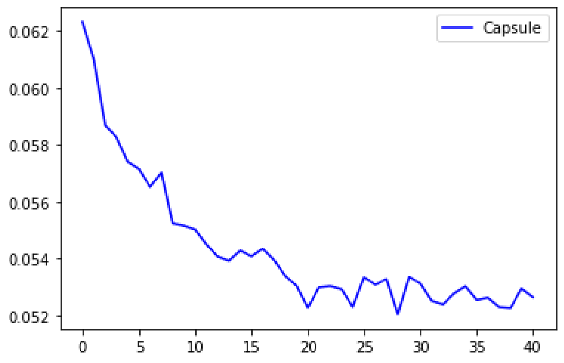

Figure 4: Loss at Capsule Head. The plot shows the loss at Capsule Head over 50 epochs

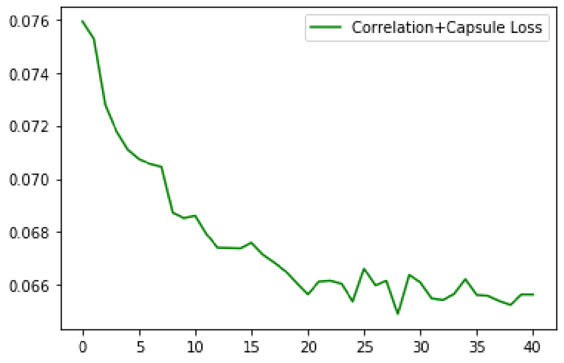

Figure 5: Loss at Convolutional Head. The plot shows the loss at Convolutional Head over 50 epochs

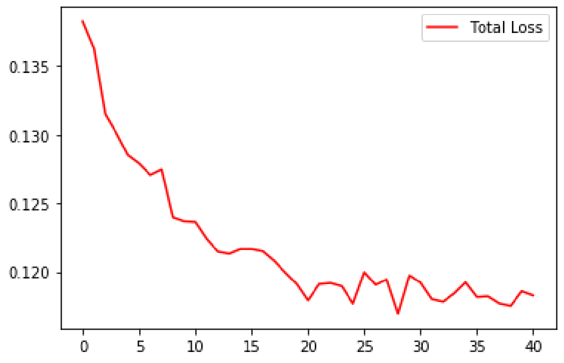

Figure 6: Total Loss. The plot shows the combined loss of Convulutioanl and Capsule heads over 50 epochs.

8 Analysis

We have trained on the GOT-10k dataset and tested the performance on the OTB-100 dataset. The GOT-10k dataset has 560+ moving object classes and 10k video segments. The OTB-100 dataset has 100 videos with 11 motion categories, one or more assigned to the videos.

Our model incorporated preliminary ideas of capsule network to object tracking. We have used a pseudo-class layer in capsule network with 16 output capsules, which inherently learns to represent a class of objects to be tracked. Even though our chosen dataset has 560+ different classes, the performance is still competitive to SiamFC in this setting. The reason for using 16 output capsules is to retain the computation efficiency required for a typical object tracker. Ideally, this layer needs to be trained with a loss corresponding to classification and reconstruction losses also.

	SiamFC	RAT
Success score	58.97	58.44
Precision score	79.20	78.88
Success rate	74.33	72.84
Speed (fps)	66.90	16.88

Table 1: Comparison of RAT tracker with SiamFC (Bertinetto et al., 2016) tracker.

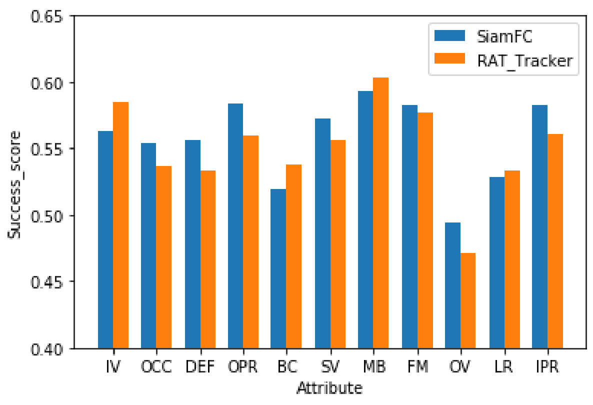

Figure 7: Comparison of success scores.

As observed in figure 7, 8, 9, our vanilla model has better performance on illumination variation (IV), background clutters (BC), low resolution (LR) and motion blur (MB) videos. Performance on fast motion (FM) is competitive. Interestingly, our model handles these classes pretty well. Even though we expected the capsule network to perform well on occlusion (OCC) videos, the above observation is quite interesting and is worth exploring further.

The model performs worse on In-plane (IPR), out-plane rotations (OPR), scale variations (SV), deformation (DEF), and significantly worsen on out-of-view (OV) videos.

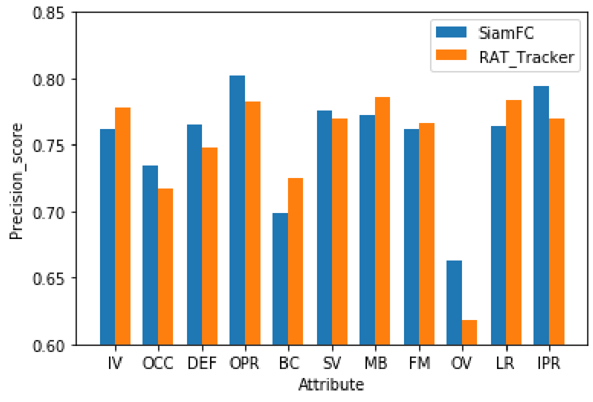

Figure 8: Comparison of precision scores.

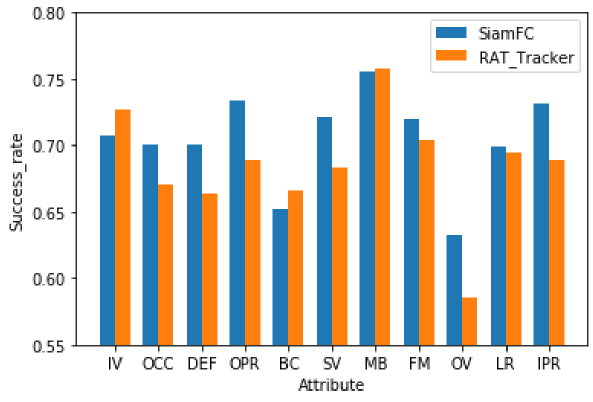

Figure 9: Comparison of precision scores.

9 Conclusion and Future Work

In this report, we present an enhancement of object tracker proposed by Bertinetto et al. (2016). We added capsule layers with the intention to better capture the spatial relationship between features of the image. We have achieved better performance on videos with attributes illumination variation (IV), background clutters (BC), low resolution (LR), motion blur (MB), and fast motion (FM). These results show that capsule networks are better at handling ambiguities in images under the above conditions. Contrary to this advantage, there is a performance drop on other attributes like in-plane (IPR), out-plane rotations (OPR), scale variations (SV), deformation (DEF), and out-of-view (OV). This proves our hypothesis that Capsule Networks can improve object tracking by leveraging spatial relationships. However, we have implemented a vanilla capsule architecture for this tracker. And, there is significant scope for more research in areas like pseudo-class capsule layer and dynamic routing algorithm to futhur accelerate the task of object tracking.

References

Evaluation Toolkit VOT18.

VOT Challenge 2018.

Dinesh Amara. 2018. Novel deep learning model for traffic sign detection using capsule networks.

Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. 2016. Fully-convolutional siamese networks for object tracking.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Al-Hussein A. El-Shafie, Mohamed Zaki, and S. E. D. Habib. 2019. Fast cnn-based object tracking using localization layers and deep features interpolation. *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*.

Ali Ghofrani and Rahil Mahdian. 2019. Capsule-based persian/arabic robust handwritten digit recognition using em routing.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *International Conference on Learning Representations*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gregory R. Koch. 2015. Siamese neural networks for one-shot image recognition.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. 2018. High performance visual tracking with siamese region proposal network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8971–8980.

Daqun Li, Yi Yu, and Xiaolin Chen. 2019. Object tracking framework with siamese network and re-detection mechanism. *EURASIP Journal on Wireless Communications and Networking*, 2019:1–14.

T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. 2020. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327.

James O' Neill. 2018. Siamese capsule networks.

Sai Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti Bathula. 2018. Dense and diverse capsule networks: Making the capsules learn better.

Qiang Ren, Shaohua Shang, and Lianghua He. 2019. Adaptive routing between capsules.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules.

Yujia Wu, Jing Li, Jia Wu, and Jun Chang. 2020. Siamese capsule networks with global and local features for text classification. *Neurocomputing*.