Literature Survey and Methodology Design

Object Tracking in Video with SiamFC model on MindSpore

Uzair Sipra

1498517

1 Introduction

Object tracking is a widely researched area in computer vision with a plethora of applications such as traffic monitoring, surveillance, robotics, and autonomous vehicle tracking [1]. Object tracking can be a specialized deep learning approach where an algorithm's main goal is to track the movement of an object, which entails detecting the specified target object in the first frame of a video and tracking the target object in the remainder of the video. Object detection, a necessary process for object tracking, is where an algorithm detects an object, classifies it by producing a bounding box (Bbox) around the object, and assigns an identification for every detected unique object [2].

Object tracking is a very challenging problem in the realm of computer vision for two main reasons. The first is the degree of variability in video quality and practical factors such as illumination, video resolution, occlusions, change in target position, and rapid movements. The second is that an object tracking process is usually only given the target object in the first frame of a video, limiting the algorithm when capturing the features of the target object [3]. Despite the challenges involved in object tracking, we can attempt to overcome these challenges by focusing on a few key features which will guide the design and development of a tracking algorithm to meet expectations [4].

Feature 1 Robustness: Robustness is the ability of a tracking algorithm to effectively track an object regardless of the video quality and practical factors affecting the object such as illumination, background clutter, and occlusion.

Feature 2 Adaptability: Adaptability is the ability of a tracking algorithm to manage the complex movement of an object by capturing and utilizing the past characteristic features of the object for future detection and tracking.

Feature 3 Real-time processing of information: A tracking system will typically be dealing with a video, which is just a sequence of images, which can require high amounts of computational power limiting the processing time of information, resulting in the need for highly efficient real-time processing tracking systems.

2 Background

The focus of this paper is the understanding and implementation of the Fully-Convolutional Siamese Architecture (SiamFC) for object tracking.

The intention of this background section is to provide detailed information regarding relevant topics that will ideally assist in understanding the SiamFC model. The background is broken into the various topics in a top-down manner where each subsequent topic is more specialized and closely related to the SiamFC model. The top-down layout allows for the flow of information, to begin with, general high-level topics and funnel down into focused lower-level topics.

2.1 Deep Learning Overview

Deep learning is a specific subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data. All machine learning models attempt to utilize mathematical and statistical techniques to enable machines to improve on tasks with experience. A deep learning task will always require these four key components to be successful.

Data: Input data that needs to be processed such as numerical data, images, videos, and audio.

Model: The architecture and specification of the deep learning model that will be used to process the input data.

Cost functions: Calculates the error difference between the predicted values from a model and the actual values from the input data.

Optimization: The optimization algorithm that will be used to update the weights in a deep learning model, typically consists of a variant of gradient descent.

There is a wide range of neural network designs and architectures that build upon each other, however, the building blocks of all these networks consist of the number of layers, the number of neurons/units per layer, the number of weights that connect pairs of units from each layer together, and the function of each layer within the network listed below. The starting layer of any neural network is called the input layer, any subsequent layers are called hidden layers, and the final layer is called the output layer. The function of a neural network layer can be reduced to 4 main types listed below [5].

Fully Connected Layer: Connects every unit from one layer in a neural network to the next layer. These layers are found in all types of networks, ranging from standard multilayer perceptrons (MLP) to convolutional neural networks (CNN).

Convolutional Layer: This layer is the core component of a CNN and is most commonly used to extract feature information from images. This layer uses a filter (kernel) of a certain size that has the same dimensions as the input and will "slide" across the entire image in segments equal to the size of the filter. The filter systematically multiplies the image section it is currently on with the filter weights to extract information and check if the image section is a feature, this systematic multiplication is called convolution.

Deconvolutional Layer: This layer essentially serves the opposite purpose of the convolutional layer. The layer uses transposed convolution to upsample images to a higher resolution.

Recurrent Layer: This layer is specialized such that connections between units form a directed circle, allowing for the layer to have an "internal state" that is updated to process sequential data. This layer is the core component of a recurrent neural network (RNN) which allows them to maintain their state across iterations.

The multilayer perceptron (MLP) (Also called Feed-Forward Neural Network) can be said to be the most generic high-level neural network architecture and is the superclass of the convolutional neural network (CNN) and recurrent neural network (RNN). The "Feed-Forward" name arises from the fact that information flows one way from the input layer to subsequent hidden layers all the way to the output layer. In simple terms, the goal of a neural network is to approximate a target function by "learning" the values of trainable parameters that result in the best approximation. The neurons are processing units that receive inputs from other neurons and apply an activation function to the inputs, they are laid out in parallel within a layer. The width of a layer is equal to the number of neurons within that layer and the depth network is equal to the overall length of that network.

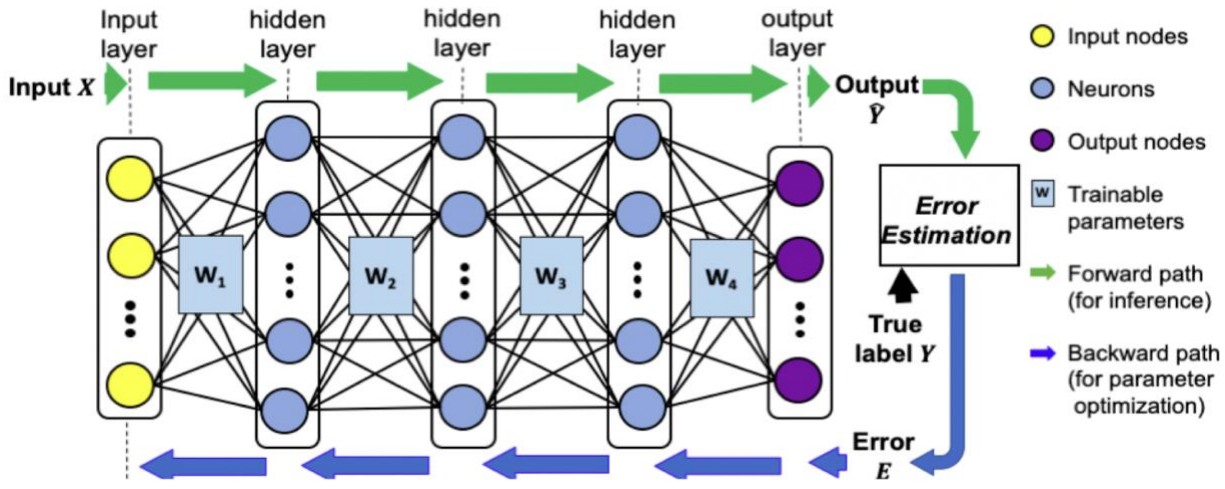

Figure 1: Multilayer Perceptron Architecture

2.2 Back Propagation/Gradient Descent Overview

Back-propagation is a generic approach to minimizing the loss function "R" via gradient descent. Backpropagation is a two-pass algorithm that allows for the weights associated with a neural network to be adjusted and optimized. The weights are optimized such that, for regression, the sum of squared error loss is minimized and for classification, the cross-entropy loss function is minimized. The two passes can be divided into the "forward" and "backward" passes.

This overview will only consider an SLP (single-layer perceptron), a neural network with a single hidden layer and the usual input and output layers, but the underlying fundamental process is the same for any neural network architecture.

The loss function "R" is a measure of how accurate the output values of a neural network are from the observed/actual values that were expected.

The sum of squared error loss function for a neural network with K outputs is:

$$R(\theta) = \sum_{i=1}^{N} R_i = \sum_{i=1}^{N} \sum_{k=1}^{K} \left(y_{ik} - f_k(x_i)\right)^2$$

In the forward pass, the current weights of the network are fixed, and the predicted values are calculated using:

$$f_k(x_i) = \beta_{0k} + g_k\left(\sum_{m=1}^{M} \beta_{mk}\, z_{mi}\right); where\ z_{mi} = \sigma\left(\alpha_{0m} + \sum_{l=1}^{p} \alpha_{lm}\, x_{li}\right)$$

z_{mi}: *Linear combinations of observations* x_i *squashed together with the* σ *activation function*

α_{lm}: *Weights from the input layer to the hidden layer*

β_{mk}: *Weights from the hidden layer to the output layer*

m: *number of units in the hidden layer,* $1 \dots M$

k: *number of output functions,* $1 \dots K$

i: *number of observations,* $1 \dots N$

l: *number of features,* $1 \dots p$

Note: In the regression setting, the output function is typically set to the identity function, which is what is assumed in this scenario.

In the backwards pass, the errors from the current model at the output and hidden layers are computed through the gradient partial derivatives:

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi},$$

$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell}.$$

Where the errors for the output and hidden layers can be expressed as:

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi},$$

$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = s_{mi} x_{i\ell}.$$

δ_{ki}: *Error at the output layer*
s_{mi}: *Error at the hidden layer*

In the backwards pass, first, the errors at the output layer are computed and then they are backpropagated to compute the errors at the hidden layer via:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$$

$$\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}}$$

Where γ_r is the learning rate.

Once the updated weights have been computed, the entire backpropagation procedure is repeated over many iterations (one sweep through the entire training set is known as a training epoch), where the weights are continuously updated, until the decrease in the loss function is only marginally smaller than the last iteration or is completely the same.

When initially beginning the training of a neural network, random values near zero are chosen for the weights resulting in an almost linear model and as the weights are updated after each training epoch the model becomes more non-linear.

The loss function will likely be non-convex, meaning it will have multiple local minimums alongside the global minimum. Due to the non-convex nature, it is not guaranteed that back-propagation will arrive at the global minimum and rather converge to one of the local

minimums. To combat this, it is recommended to repeat the training of a neural network multiple times with different random starting initializations of the weights and biases so that different minima may be converged onto by back-propagation. The goal is not to arrive at the global minimum, because it is unlikely and difficult, but to arrive at a local minimum where the test error rate of the neural network is within an acceptable range.

2.3 Common Object Tracking Methods

The successful methods for object tracking are mainly divided into three categories. The first is object tracking based on correlation filters, the second is object tracking based on convolutional neural networks (CNN), and the final is object tracking based on siamese neural networks (SNN). The focus of this paper is the SNN method, but an overview of the other two methods is discussed [6] below with a more detailed explanation of the SNN tracking method later in this paper.

2.3.1 Correlation Filter Based Object Tracking

The correlation filter algorithm learns to distinguish the difference between foreground objects and the surrounding background by efficiently solving the ridge regression regularization problem.

This method is a very successful option for object tracking tasks mainly due to the replacement of the convolution operation with an element-by-element multiplication in a fast Fourier transform, allowing for a substantial decrease in time consumption.

The disadvantage of this method is that correlation filters are trained using data from a single video resulting in only simple models being learned.

2.3.2 Convolutional Neural Network Based Object Tracking

CNNs have made substantial progress in the realm of computer vision for image classification and object detection. They also allow for significant improvement in state-of-the-art methods for object tracking due to the powerful feature extraction abilities associated with CNNs.

There are two main ways that CNNs are used in object tracking tasks. The first is that CNNs are trained as feature extractors, providing rich feature representations for an object tracking model. The second is that CNNs can be trained as binary classifiers on a large image dataset where they can classify the target object and background in an image.

The disadvantage of CNN-based tracking is due to a large number of trainable parameters it is computationally expensive to update parameters during online training resulting in most CNN-based trackers running slower than in real-time.

2.4 Similarity Learning

Similarity learning is an area of machine learning where the goal is to quantify how similar two objects are to each other by training an algorithm to learn a similarity function. A high similarity score is returned when the two objects have a high degree of similarity, and the opposite is true for a low score [7].

Similarity learning typically follows a structured process where the objects of comparison are transformed into numerical vector representations which are used to calculate the difference between the vectors and classify if the objects are similar or dissimilar. It can be understood that it is much easier to empirically compare numbers to one another instead of arbitrary objects such as images or text. An overview of the similarity learning process is given below [8].

Transformation: The input data is fed into an encoder which extracts the features and transforms them into a vector. This encoder can take many forms depending on the type of input data but is typically a CNN or RNN. The feature vector is then fed into a fully-connected layer to classify the feature vector. There will be multiple feature vectors which are the latent space representations of the objects being compared.

Comparison: The feature vectors of the two objects can be compared to one another using a distance function, commonly will be Euclidean. Once the distance between the two vectors is known, a threshold needs to be set which will determine if the two objects are similar or dissimilar.

Classification: The task of selecting a threshold can oftentimes be complex and instead can be assigned to another classifier. This classifier is given the distance difference between the two feature vectors and classifies it as similar or dissimilar. This classifier can be selected from a plethora of linear classifier options.

2.5 Siamese Neural Networks

A siamese neural network (SNN) is an architecture that consists of two identical neural networks which both take distinct inputs and the weights between them are tied together to ensure the networks remain identical during training. The reasoning behind identical neural networks with tied weights is to ensure that any distinct inputs are mapped to an identical feature space, guaranteeing that a sound empirical comparison between the distinct inputs is made. If the networks were not identical, it would not be possible to determine if two inputs classified as similar or dissimilar by an SNN is accurate because each network would have mapped either input to different feature spaces [9]. The outputs of the identical neural networks are joined with a shared similarity layer which calculates the distance between the two feature vector representations and then classifies the distance as “similar” or “dissimilar”. When training an SNN, the input data is formatted as pairs of inputs that are labelled as “genuine” for similar input pairs and “imposter” for dissimilar input pairs [10].

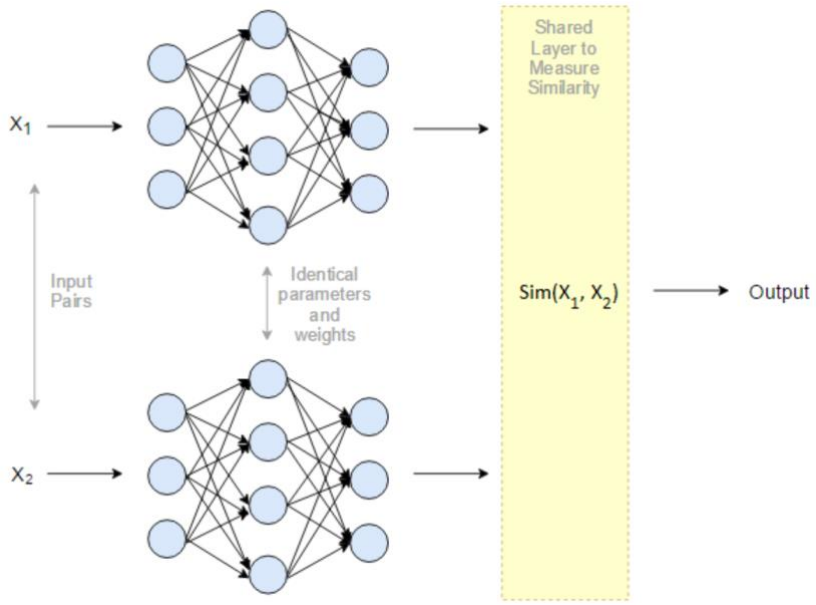

Figure 2: Siamese Neural Network Architecture
Note. Reused from [10]

SNNs make use of the "contrastive loss" function for the network architectures loss function. Contrastive loss can be expressed as the combination of two different loss functions, where one loss function deals with similar input pairs and the other loss function deals with dissimilar input pairs. When the input pairs are similar the distance between them is small, therefore we want a loss function that minimizes the distance between similar pairs of inputs i.e., L1 norm (Manhattan Distance) or L2 norm (Euclidean Distance) loss functions, depending on which distance function is utilized. The distance between dissimilar input pairs will be greater than some threshold value leading to the use of the hinge loss function. Combining these two loss functions results in the contrastive loss function which brings similar pairs together and pushes dissimilar pairs apart [11].

$$L2\ norm\ loss\colon L(A,B) = y||F(A) - F(B)||^2$$
$$LH\ (Hinge\ loss)\colon L(A,B) = (1-y)max(0, M^2 - ||F(A) - F(B)||^2)$$
$$Contrastive\ loss\colon L(A,B) = y||F(A) - F(B)||^2 + (1-y)max(0, M^2 - ||F(A) - F(B)||^2)$$
$$F(A), F(B)\colon Feature\ space\ values\ of\ input\ pairs$$
$$y\colon Binary\ variable, equal\ to\ 1\ for\ similar\ pairs\ and\ 0\ for\ disimlar\ pairs$$
$$M\colon Threshold\ or\ margin\ value\ seperating\ similar\ and\ dissimilar\ classification$$

3 Approach

The challenge we are trying to navigate is implementing the state of the art (SOTA) Fully Convolutional Siamese Network model (SiamFC) from the paper "Fully-Convolutional Siamese Networks for Object Tracking" [12] into MindSpore, which is Huawei's AI computing framework that has been fully developed from the ground up. MindSpore provides unified APIs

and end-to-end AI capabilities for model development, execution, and deployment in all scenarios making it the perfect framework to implement the SiamFC model.

3.1 PyTorch

With reference to a previous implementation of the SiamFC model in the PyTorch deep learning framework, an in-depth study of PyTorch will be conducted for a proper understanding of the previous implementation. PyTorch tutorials and documentation will be the core resources utilized.

3.2 MindSpore

An in-depth study of the MindSpore framework will be initiated in conjunction with PyTorch. MindSpore offers a detailed API mapping for PyTorch, which will undoubtedly be an extremely beneficial resource. The API mapping is provided by the community meaning it is always growing and being updated with improvements and even offers detailed explanations in areas that are difficult to map from PyTorch to MindSpore.

3.3 Atlas 800 AI Training Server

The development of the SiamFC model in MindSpore will be done on the Atlas 800 AI Training Server. The Atlas 800 is powered by 4 Kunpeng 920 processors and 8 Ascend 910 AI processors, giving it 2.56 PFLOPS in AI computing power. The Atlas 800 training server is an ideal candidate as it provides tremendous computing power remotely, allowing for quick and optimized training throughout the development of the SiamFC model. The jupyter notebooks programming environment is also offered through the Atlas 800 training server, making it an easy choice to use.

4 Methodology

Here we discuss the specifics in regards to the SiamFC model such as architecture explanation, training and testing of the model, and the evaluation metrics we will utilize in benchmarking the SiamFC model with other state-of-the-art models.

4.1 Fully-Convolutional Siamese Architecture (SiamFC)

The basis of the Fully-Convolutional Siamese Network concerning object tracking is to model the task of object tracking as a similarity learning problem. The SiamFC model builds upon the siamese neural network architecture, taking pairs of images as inputs and generating

feature maps via an embedding function. The pairs of input images are separated into two categories, one being exemplar images and the other being candidate images. The exemplar image is the very first frame in a video that contains a bounding box around the target tracking object. The candidate image corresponds to a new subsequent image frame of the same size as the exemplar. The candidate image acts as a patch within a larger search image of the current frame which is used to exhaustively test all possible locations of the target tracking object and return the candidate image that has the highest similarity to the target tracking object. The proposed learning function is intended to take the exemplar and candidate images as inputs, compare the two inputs, and return a similarity score for the two inputs, where a high score represents high similarity and vice versa for a low score [13].

The advantage of a fully convolutional network is that a larger size search image can be used as an input and the candidate image patch will translate over all possible sub-windows on a dense grid while calculating the similarity for each sub-window in a single evaluation. To implement this within the siamese network architecture we choose a convolutional network as the embedding function, which is essentially the output of the two identical branches within a siamese network. The convolutional embedding function will transform the input images into feature maps which will then be combined via a cross-correlation layer with added bias terms for each candidate image patch. The overall similarity function of the SiamFC model is shown below

$$f(z,x) = \varphi(z) * \varphi(x) + b$$
$$f(z,x): Similarity\ Function$$
$$\varphi(z), \varphi(x): Feature\ map\ outputs\ of\ the\ convolutional\ embedding\ branches$$
$$*: Cross-correlation\ operation$$
$$b: Bias\ due\ to\ candidate\ image\ patches$$

The final output of the SiamFC model is a score map containing the similarity scores of all candidate image patches to the target object [14].

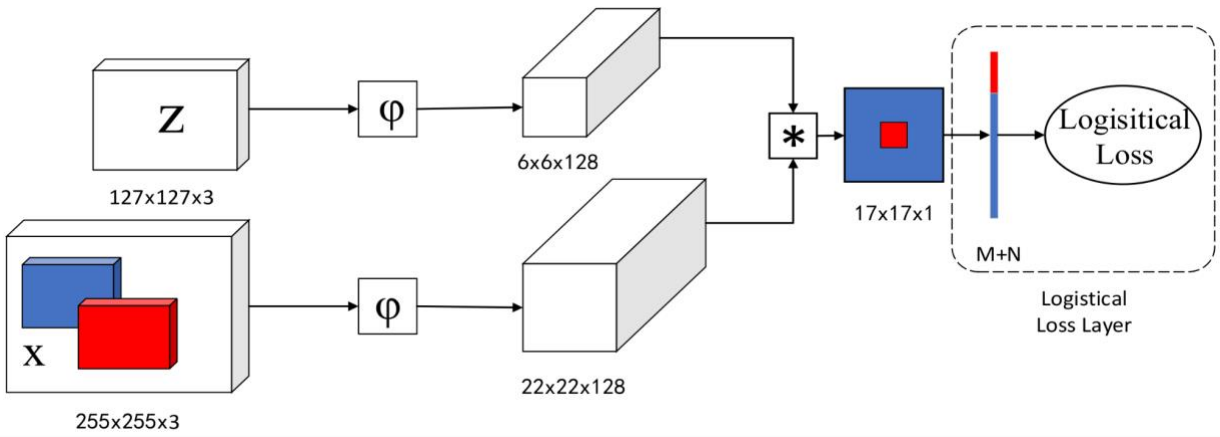

Figure 3: Architecture of the Fully-Convolutional Siamese Network for tracking
Note. Reused from [3], pp. 3

The logistic loss function is adopted to compute the individual losses of exemplar/candidate pairs

$$Logistic\ loss: l(y, v) = log(1 + exp(-yv))$$
$$v: Real\ valued\ score\ of\ a\ single\ exemplar/candidate\ pair$$
$$y: Ground\ truth\ label\ of\ a\ single\ exemplar/candidate\ pair\ \in \{+1, -1\}$$

The overall loss of the score map is the mean of the individual losses

$$L(y, v) = \frac{1}{|D|} \sum_{u \in D} l(y[u], v[u])$$
$$D: Size\ of\ the\ score\ map$$
$$u: Individual\ positions\ within\ the\ score\ map$$

The optimization of the convolutional network will be obtained via simple Stochastic Gradient Descent (SGD)

$$\arg\min_{\theta} E_{(z,x,y)}\, L(y, f(z, x; \theta))$$
$$\theta: Parameters\ of\ the\ convolutional\ network$$

4.2 Proposed Implementation Scope

4.2.1 Training Dataset: GOT-10K

The GOT-10k is a large tracking database that offers unprecedentedly wide coverage of common moving objects in the wild. GOT-10k populates the majority of over 560 classes of moving objects and 87 motion patterns. With GOT-10k offering over 10,000 video segments with more than 1.5 million manually labelled bounding boxes, it is the ideal dataset for training the proposed SiamFC model while enabling unified training and stable evaluation [15].

4.2.2 Tracking Dataset: VOT2018

The Visual Object Tracking Challenge is an annual tracker benchmarking activity organized by the VOT initiative [16]. In VOT2018, trackers were evaluated on the LTB35 dataset which contains 35 sequences, carefully selected to obtain a dataset with long sequences containing many target disappearances. Sequence resolutions range between 1280 x 720 and 290 x 217, lasting on average for 40 frames and the dataset contains a total of 14687 frames [17]. The carefully curated dataset from the VOT2018 is the selected choice for testing the SiamFC model.

4.2.3 Input Configurations

Input pairs will be obtained from the dataset of annotated videos by extracting the exemplar and search images centred on the target object. The class of the target object is not

considered during training and the scale of the target object within each image is normalized. Exemplar images will be contained to a size of 127 x 127 pixels and search images to a size of 255 x 255 pixels. The cross-correlation operation will result in a score map of size 17 x 17 x 1.

Figure 4: Exemplar and search image pairs extracted from the same videos. When a sub-window moves beyond the bounds of an image, the missing bounds are filled with the mean RGB value. Note. Reused from [12], pp. 5

4.3 Evaluation Metrics

The main evaluation metrics for an object tracking method are expected average overlap rate (EAO), robustness (R), and accuracy (A), as used by VOT. In the case of object tracking, accuracy is the average overlap between the predicted bounding boxes of a tracker and the ground truth bounding boxes, equivalent to the average IoU. Robustness measures how many times a tracker loses the target object during tracking. EAO is an estimator of the average overlap a tracker is expected to attain on a large collection of short-term sequences. Another evaluation metric that can be taken into consideration is the tracking speed, which is an important factor in practical applications of a tracker [18]. The tracking speed is essentially the FPS (frames per second) rate of a video that is still trackable by the tracking method.

Qualitative metrics to be considered are visual confirmation of tracker operation on video samples and tracker performance comparison graphs with different object tracking models to the SiamFC architecture.

5 References

[1] Z. Soleimanitaleb, M. Ali Keyvanrad, A. Jafari, "Object Tracking Methods: A Review", in *9th International Conference on Computer and Knowledge Engineering (ICCKE 2019)*, October 24-25 2019, pp. 282

[2] N. Barla, V7labs March 19 2022, "The Complete Guide to Object Tracking [+V7 Tutorial], <https://www.v7labs.com/blog/object-tracking-guide>

[3] H. Xu, Y, Zhu, "Real-time object tracking based on improved fully-convolutional siamese network", in *Computers & Electrical Engineering (Elsevier)*, September 2020, pp.1

[4] Z. Soleimanitaleb, M. Ali Keyvanrad, A. Jafari, "Object Tracking Methods: A Review", in *9th International Conference on Computer and Knowledge Engineering (ICCKE 2019)*, October 24-25 2019, pp. 283

[5] M. Isaksson, Towards Data Science June 6 2020, "Four Common Types of Neural Network Layers", <https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c>

[6] H. Xu, Y, Zhu, "Real-time object tracking based on improved fully-convolutional siamese network", in *Computers & Electrical Engineering (Elsevier)*, September 2020, pp.1-2

[7] Great Learning December 17 2020, "Similarity Learning with Siamese Networks", <https://www.mygreatlearning.com/blog/siamese-networks/#sh2>

[8] T. Martino, Towards Data Science June 1 2020, "An introduction to Deep Similarity Learning for sequences", <https://towardsdatascience.com/introduction-to-deep-similarity-learning-for-sequences-89d9c26f8392>

[9] G. Koch, R. Zemel, R. Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition", pp. 3

[10] MARTIN, K., WIRATUNGA, N., SANI, S., MASSIE, S. and CLOS, J. 2017. A convolutional Siamese network for developing similarity knowledge in the SelfBACK dataset. In *Sanchez-Ruiz, A.A. and Kofod-Petersen, A. (eds.) Workshop proceedings of the 25th International conference on case-based reasoning (ICCBR 2017)*, 26-29 June 2017, Trondheim, Norway. CEUR workshop proceedings, 2028. Aachen: CEUR-WS [online], session 2: case-based reasoning and deep learning workshop (CBRDL-2017), pages 85-94. Available from: http://ceur-ws.org/Vol-2028/paper8.pdf

[11] Great Learning December 17 2020, "Similarity Learning with Siamese Networks", <https://www.mygreatlearning.com/blog/siamese-networks/#sh2>

[12] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking", *from Department of Engineering Science, University of Oxford*

[13] H. Xu, Y, Zhu, "Real-time object tracking based on improved fully-convolutional siamese network", in *Computers & Electrical Engineering (Elsevier)*, September 2020, pp. 2-4

[14] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking", *from Department of Engineering Science, University of Oxford*, pp. 3-4

[15] GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild.
L. Huang*, X. Zhao*, and K. Huang. *(*Equal contribution)*
IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI).

[16] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pfugfelder, Luka Čehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernandez, and et al, "The sixth Visual Object Tracking VOT2018 challenge results", in *VOT2018 workshop, ECCV2018*, 2018, pp. 3

[17] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pfugfelder, Luka Čehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernandez, and et al, "The sixth Visual Object Tracking VOT2018 challenge results", in *VOT2018 workshop, ECCV2018*, 2018, pp. 10

[18] M.Kristan,J.Matas,A.Leonardis,M.Felsberg,L.Cˇehovin, G. Ferna´ndez, T. Vojir, G. Ha¨ger, G. Nebehay, R. Pflugfelder, and et al. The visual object tracking vot2015 challenge re- sults. In *ICCV2015 Workshops, Workshop on visual object tracking challenge*, 2015, pp. 6-7