ABSTRACT  Visual tracking is an open and exciting field of research. The researchers introduced great efforts to be close to the ideal state of stable tracking of objects regardless of different appearances or circumstances. Owing to the attractive advantages of generative adversarial networks (GANs), they have been a promising area of research in many fields. However, GAN network architecture has not been thoroughly investigated in the visual tracking research community. Inspired by visual tracking via adversarial learning (VITAL), we present a novel network to generate randomly initialized masks for building augmented feature maps using multilayer perceptron (MLP) generative models. To obtain more robust tracking these augmented masks can extract robust features that do not change over a long temporal span. Some models such as deep convolutional generative adversarial networks (DCGANs) have been proposed to obtain powerful generator architectures by eliminating or minimizing the use of fully connected layers. This study demonstrates that the use of MLP architecture for the generator is more robust and efficient than the convolution-only architecture. Also, to realize better performance, we used one-sided label smoothing to regularize the discriminator in the training stage and the label smoothing regularization (LSR) method to reduce the overfitting of the classifier in the online tracking stage. The experiments show that the proposed model is more robust than the DCGAN model and offers satisfactory performance compared with the state-of-the-art deep visual trackers on OTB-100, VOT2019 and LaSOT datasets.

INDEX TERMS  Deep learning, generative adversarial network, multilayer perceptron, visual tracking.

I. INTRODUCTION  Visual tracking is the estimation of the position and orientation of an unidentified target when only the first state of the target is available in a video frame. Visual tracking has many applications, such as autonomous robots, surgery, sports, self-driving cars, and augmented reality. As mentioned in the literature on traditional tracking algorithms, any growth in feature extraction methods directly affects the tracking performance. As a result, we consider that deep learning techniques significantly improve tracking performance, as they have shown powerful capabilities in feature extraction and object tracking.  As mentioned in [1], deep visual tracking has two main observation models: generative and discriminative models. Generative models (or one-stage regression frameworks) use a chosen similarity function to precisely match the target object template with a given search region. The discriminative models (or two-stage classification frameworks or tracking-by-detection frameworks) use binary classification conducted with the convolutional neural network (CNN) model to differentiate the tracked object from the surrounding background. The last model involves two stages: drawing candidate samples around the object and categorizing each sample as a target or background. The tracking-by-detection
model is the most familiar one, and there are a lot of efforts done to achieve high-performance tracking results by using it. This model was used in the present study. The appearance of an object differs according to its view, whether it is rotated, scaled, deformed, blurred, or occluded. These various appearances of the object throughout the video were considered the most difficult challenge faced by deep visual tracking algorithms, which depend on learning the network about the main features of the tracked object. The more robust the extracted features in the training process, the more the tracker can track the target, regardless of the surrounding conditions. We should consider that an object may change its appearance owing to movement, for example, changes in the perspective of hidden parts when changing the point of view. In such cases, the features to be tracked change continuously, and the tracker must deal with features that come in and out of the picture. Consider the frames in Fig. 1, which show the different appearances of the girl. Our goal is to develop a method that can track a girl’s face in any part of the image, from any angle of view, with any size throughout the video or under any other condition. As a first step, we apply feature extraction to our first image to extract the features required to track the girl throughout the video. However, the problem with the feature extractor is that it depends on the angle of view. In other words, we applied a feature extractor to the front angle of the face of the girl (Fig. 1(a)). Then, we can use these features to track the face as long as the front angle of view is present. It does not work when the angle changes from the front to the side angle (Fig. 1(c)), because the extracted features are limited to the front angle. The only way to handle these circumstances is to have large-scale training data that have most of the challenging attributes to be able to effectively model invariance. The existing trackers, especially CNN trackers, only use the positive samples in the first frames of the videos in the datasets for the training tasks. MDNet [2], a state-of-the-art tracking algorithm, and the winner in the VOT2015 challenge used a multi-domain layer for the first time to learn generic feature representations. However, MDNet extracts positive samples depending on the intersection over union (IoU) compared with the ground truth BBox, which does not provide adequate diversity of data that is required for a successful deep learning tracking model. One possible solution is to augment the training samples in the feature space by applying random appearances or geometric transformations to the annotated training samples via adversarial learning.

The goal of using adversarial learning is to augment these extracted features further to work at different angles (rotation invariance), object sizes (scale invariance), or occlusions. For example, if the system extracts features related to the two eyes for detection, it will not function from a side angle because the two eyes will not be visible from the side angle. Therefore, the adversarial learning model drops two eye-related features because it does not help the model track at different angles. Instead, it learns the color combination of the eyes preserved from the side and front angles. Many other visual attributes exist, such as illumination variation, background clutter, deformation, low resolution, fast motion, and motion blur. The tracker must track the target object in the face of all of these challenges. The VITAL network [3] is one of the most common visual trackers that applies adversarial learning. The generator used in the VITAL network is composed of only two fully connected layers, which is not sufficient to apply robust masks capable of augmenting the input feature maps by generating other versions with more generic features. Therefore, we introduced our network architecture which applies adversarial learning using a generator with five MLP layers.

The main contributions of the proposed work are summarized as follows:

1) We propose a new tracking framework using a 5-MLP generative model. This model generates masks using an adversarial learning process. These masks adaptively exclude the discriminative features that appear in individual frames and maintain the most robust features that continue for a long period.

2) We propose a new tracking framework using the deep convolutional generative adversarial network (DCGAN) [4] as a generative model for the purpose mentioned in the first point. However, the architecture of the generator is convolutional-only.
3) We integrated a cost-sensitive loss function into the Jensen-Shanon (JS) adversarial objective function to balance the training sample loss of the generator. This integration helps generate more powerful masks that can identify the most robust features.

4) We used one-sided label smoothing procedure during the training phase to reduce the overconfidence of the discriminator in the hard labels of real samples.

5) We used label smoothing regularization during the online fine-tuning to reduce the overfitting of the classifier to the hard labels of the classes of the input samples.

6) We proved that the MLP architecture is better than the convolutional architecture in generating more effective masks, which helps the tracker become more robust. We based our proof on the results for the OTB-100, VOT2019 and LaSOT datasets.

7) We evaluated both tracking models with the state-of-the-art visual trackers on the OTB-100, VOT2019 and LaSOT datasets. We found that our MLP-based tracker performed favorably compared to the other methods.

The remainder of this paper is organized as follows: Section 2 presents the related work to the proposed framework. Section 3 explains the proposed algorithm for both training and tracking states. The experimental results are presented and discussed in Section 4. Section 5 presents conclusions and directions for future research.

II. RELATED WORK

In this section, we present various tracking techniques based on various deep structures. The structures we focus on are convolutional neural networks (CNNs), Siamese neural networks (SNNs), and generative adversarial networks (GANs). Also, we present a brief explanation about DCGAN and MLP generator architectures which we use in our implementations.

A. CONVOLUTIONAL NEURAL NETWORK (CNN)

The CNN architecture has been used extensively in deep visual tracking methods in recent years. Many studies have introduced an effective target representation, which is the main advantage of using a CNN. Some approaches used offline training on large-scale datasets and designed specific deep CNNs such as MDNet [2], RT-MDNet [5], CODA [6], and DAT [7]. In contrast, others have constructed several target models to capture a variety of target appearances, such as TCNN [8] and STP [9]. In addition, CREST [10] and DeepSTRCF [11] incorporated spatial and temporal information to improve model generalization. To achieve robust and reliable visual tracking results, Li et al. presented a dual-regression architecture that fuses a discriminative fully convolutional module and fine-grained correlation filter component [12].

B. SIAMESE NEURAL NETWORK (SNN)

The SNN architecture has become one of the most attractive deep architectures for learning similarity knowledge and achieving real-time speed for efficient visual tracking. The common aim of using methods based on the SNN architecture is to solve the issues of pre-trained CNNs and exploit the benefits of using end-to-end learning for real-time applications. SiamDW-SiamRPN [13], SiamRPN++ [14], C-RPN [15], SiamMask [16], and DaSiamRPN [17] methods are based on the SiamRPN [18] method, which involves a Siamese subnetwork and region proposal subnetwork. Correlation feature maps and feature extraction force these subnetworks to deal with the visual tracking problem using a one-shot detection task. These methods are efficient and accurate estimations that incorporate the proposal choice and enhancement approaches into a Siamese network. The authors of [19] presented a self-supervised learning-based visual tracker that combines correlation filters and Siamese networks. To learn the feature extractor from neighboring video frames, they used a multi-cycle consistency loss as self-supervised information. The authors of [20] presented a multi-level similarity model for a thermal infrared object, one for global semantic similarity and the other for local structure similarity. The framework for this model is Siamese-based. Fan et al. incorporated alignment and aggregation modules in a Siamese-based network [21]. The feature-alignment module adjusts the search region to account for significant pose changes. To address the extreme appearance variations of the object, a shallow and high-level aggregation module is designed.

C. GENERATIVE ADVERSARIAL NETWORK (GAN)

It is common in the visual tracking community to use discriminative models (or classifiers) to distinguish between a target object and its background. In contrast, generative models learn to represent a target object and ignore the background. The aim of generative models is to model a distribution that approximates a given real dataset. More officially, given some data samples X and their corresponding labels Y, generative models identify the joint probability p(X, Y) (or p(X) if there are no labels). However, discriminative models identify the conditional probability p(Y|X). The three major methods that use generative models are variational autoencoders (VAEs), autoregressive density estimation (ADE), and generative adversarial networks (GANs).

Goodfellow et al. [22] proposed GANs that attracted deep-learning communities in 2014. The basic theory of GAN is using two convolutional neural networks, opposing one against the other (that’s why it’s known as “adversarial”), to generate new fake instances of data that can pass for real data. The two competitive networks are the generator G and discriminator D. G is trained to map the random vector z to generate fake samples, which should be as real as possible. By contrast, D can categorize images fed into real and fake (generated) images. Both networks attempt to optimize
opposing objective functions. Their losses are pushed against each other. Principally, the loss function of the GAN measures the similarity between the fake data distribution $p_g$ and the real sample distribution $p_r$ using the Jensen-Shannon (JS) divergence. The discriminator and generator play a mini-max opposing game in which we optimize the following loss function:

$$\min_{G} \max_{D} L(D, G) = E_{x \sim p_r(x)} \left[ \log D(x) \right] + E_{z \sim p_z(z)} \left[ \log(1 - D(G(z))) \right]$$

(1)

In this function:
- $D(x)$ is D’s estimation of the probability that real data instance $x$ is real,
- $E_{x \sim p_r(x)}$ is the expected value over all real data examples,
- $G(z)$ is the output of $G$, given noise $z$,
- $D(G(z))$ is the D’s estimation of the probability that a fake sample is real,
- $E_{z \sim p_z(z)}$ is the expected value of all random inputs to the generator.

On the one hand, the discriminator is expected to output a probability $D(G(z))$ near zero by maximizing $E_{z \sim p_z(z)} \left[ \log(1 - D(G(z))) \right]$. On the other hand, $G$ is trained to maximize the probability that $D$ produces a fake example, and consequently, to minimize $E_{z \sim p_z(z)} \left[ \log(1 - D(G(z))) \right]$.

Once the generator is trained to its optimum value ($p_g \approx p_r$), the discriminator’s output becomes 1/2, which is the Nash equilibrium point for the GANs.

In a standard GAN, the generator network uses a mixture of linear rectifier and sigmoid activation. Although the original GANs have achieved tremendous success, they face two significant problems: vanishing gradients and mode collapse, which are direct reasons for training instability. Many approaches have been proposed in the literature to solve standard GAN problems, either using different network architectures [4], [23], [24], [25] or objective functions [26], [27], [28]. GCGAN [29] is a successful update of GANs that is applied for generating high-resolution images and videos. The GCGAN model uses the enforcement of the geometry consistency constraint to solve the common mode-collapse problem that most GANs models face and to reduce the semantic distortions in the translation process.

GANs were not familiar in the visual tracking community for three reasons: 1) GANs were designed to generate synthetic images; however, visual trackers were designed for classification. Therefore, there is no relationship between the two targets. 2) Visual tracking often uses supervised learning with labeled training examples in the input. However, GANs are unsupervised learning algorithms based on random noise inputs. 3) Convergence is the most important target in learning deep visual trackers to explain the success of the training process, which is known to be very difficult to realize in GANs.

Despite these reasons, some deep learning visual tracking methods augment training examples and target modelling. In [3] and [30], GANs augmented positive samples in a feature space that maintained the most robust features for a long temporal span. In addition, the TGGAN method [31] learns a broad appearance distribution to address the self-learning issue of visual tracking. In addition, the ADT approach [32] takes advantage of both regression and classification by combining regression and discriminative networks. SINT++ [33] proposed the generation of hard positive samples using adversarial learning for visual tracking. A novel sample-level GAN [34] augments the training data by producing a considerable number of samples that are similar to real-life situations and can exhibit a greater variety of appearances.

In the proposed framework, we used two models (MLP and DCGAN) proposed in [26] as two opposing generator structures. In [26], the effects of using these two architectures on the WGAN model performance were compared. Therefore, we investigated the effects of the same two generative networks on the visual tracking performance. In [26], the two models had four layers; however, in our proposed frameworks we added an additional layer to make them compatible with the tracking network. Ultimately, our comparison concentrated on the impact of using MLP and convolutional architectures of generators on the visual tracking performance while fixing the depth.

**D. DCGAN ARCHITECTURE**

The deep convolutional generative adversarial network (DCGAN) [4], extends the GAN architecture for deep convolutional-only networks. The DCGAN architecture includes four main modifications to the CNN structure.

- All pooling and fully connected layers are interchanged with convolutional layers to become an all-convolutional network.
- Apply batch normalization to the generator and discriminator, except for the generator’s output layer and discriminator’s input layer.
- The ReLU activation function is used in the generator for all layers except the output layer, which uses the tanh function.
- The LeakyReLU activation function is used in the discriminator for all layers except the output layer, which uses a sigmoid function.

Usually, existing enhancement methods use pure deep CNN layers without fully connected layers [4], [26], [27]. Such methods are mainly inspired by the DCGAN architecture, which removes all fully connected and pooling layers from the generator and discriminator. However, the FCC-GAN architecture [35] proves that using fully connected layers with convolutional layers is more effective than using convolutional-only architectures.

**E. MLP ARCHITECTURE**

A multilayer perceptron (MLP) is an artificial neural network (ANA). The architecture of an MLP network consists of at least three layers of nodes: input, hidden, and output layers. Each node is a neuron that applies a nonlinear activation function.
function, except for the input layers. We chose multilayer perceptron architecture in this study for two reasons. (1) MLP is compatible with convolutional neural networks, which are trained using back-propagation. (2) MLP can be a deep model consistent with the spirit of feature reuse.

In the proposed architecture, we are using 5-layer Leaky-RELU MLP for the generator. We performed a linear transformation at each hidden layer and passed it through a nonlinear Leaky-RELU with 0.2 slope activation function. Whereas convolutional layers are better at extracting spatial information and high-dimensional features in individual frames, MLP layers are better for extracting general and nonspatial information that lasts for a long temporal span [35], [36].

III. THE PROPOSED METHOD

In Section I, the GANs are rarely used for visual tracking. Only a few researchers have used GANs for visual tracking, and most of them are dependent on the same concept used in the VITAL method. The main idea was to generate masks (filters), which acted as the second part of the feature extraction stage. These masks enabled the classifier to recognize the most robust features of the target object that did not change over a long temporal span under different conditions. We replaced the GAN generator with the DCGAN architecture in one experiment and a 5-layer Leaky-RELU MLP in another experiment. Both generated masks to augment the input feature map.

Now, two tasks are in parallel: one is to track positive samples of the image, and the second is to ignore negative parts. To ignore the negative samples, we first need to detect them.

A. OFFLINE TRAINING

In general, the main problem that most deep visual trackers suffer from is the lack of training data. Most of them provided object information from the first frame as the training data. Therefore, we performed offline pre-training of our model based on positive and negative samples in the training data of the MDNet model [2]. After completing the training, the tracking was started. Throughout the video, tracking performs tests and evaluations on its own. However, it is impossible to preserve previously invisible object features from one frame to another. Moreover, the positive samples of the target object are highly spatially overlapped, and they cannot maintain rich variations in appearance. For this reason, we use an adversarial learning approach that trains weight masks to
FIGURE 3. Flowchart of the proposed framework in the training module.
capture robust features that last for a long temporal span from frame to frame.

The backbone CNN network on which we based our network was the VGG-M model [37]. As shown in Fig. 2, the architecture of our network in the training module consisted of three networks: feature extraction, adversarial learning, and classification. The details of the adversarial learning process sequence are shown in the flowchart in Fig. 3.

1) GENERATING POSITIVE AND NEGATIVE SAMPLES
The first frame was loaded with the ground truth coordinates. Then, random 500 positive samples were generated randomly close to the ground truth BBox (with IOU overlap ratio $\geq 0.7$ with ground-truth BBox). Also, random 5000 negative samples were drawn randomly far from the ground truth (with IOU overlap ratio $\leq 0.5$ with ground-truth BBox).

2) GENERATING POSITIVE AND NEGATIVE FEATURES
We used the first three convolutional layers in the VGG-M as a pre-trained feature extractor. Three filters of sizes 7, 5, and 3 were used. These convolutional layers are equipped with a rectified linear unit (ReLU) activation function. The feature extractor is pre-trained on the ImageNet dataset [38], its parameters are fixed, and only the parameters of the fully connected layers are updated during the training. Features (C) were extracted from positive and negative samples in the first frame. The output of these layers is the first feature map that contains both discriminative features in this frame and the most robust features.

3) OFFLINE-TRAINING OF D
The three fully connected layers (FC4, FC5, and FC6) used in the MDNet model were also used in the adversarial learning process as a discriminator (D), as shown in Fig. 4. The extracted features of the positive and negative samples were fed into D to train it once while fixing weights of G without training. The discriminator calculates the prediction scores for positive and negative samples using ground truth labels and supervised learning. The prediction scores were used to calculate the loss using the binary cross-entropy (BCE) function (2). This loss is optimized using the stochastic gradient descent (SGD) optimizer and the weights of D are updated.

$$L_{BCE} = -\sum_{j=1}^{2} q(j) \log(p(j))$$  \hspace{1cm} (2)

where $p$ and $q$ represent the distributions of the training samples and corresponding ground truth labels, respectively.

4) ADVERSARIAL LEARNING
After training D once in a supervised learning procedure, the unsupervised adversarial learning procedure was initiated. In this module, G and D are iteratively trained. The generator network (G) uses the extracted features of the positive samples to generate nine masks (G(C) or $M^*$) with the exact resolution of the input features (3 x 3). The masks were initially randomly set and then gradually modified. Each mask represents a different variation in appearance. Therefore, the nine masks were expected to cover most of the appearance variations. Predicted masks ($M^*$) were applied to the extracted features (C) to create an output feature map (C'). (C') is defined in (3), and this operation is termed the dropout operation. The input feature map (C) has nine elements arranged as a 3 x 3 matrix in each channel of 512 elements. Each channel is multiplied by the generated nine masks (arranged as a 3 x 3 matrix) to obtain the augmented feature map (C'), which has the same dimensions as the input feature map C (512@3 x 3). This is called a dropout operation because the elements that represent discriminative features in (C), which are multiplied by the low-
The architecture of the proposed MLP generator network.

All generated output features \((C^O)\) are fed into \(D\) to start adversarial learning. Output feature \(C^O_{ijk}\) with the highest prediction score is selected, and the corresponding \((M)\) is assigned as \((M)\), which is used to update \((M)\) in (4) and update \(G\) accordingly. It reduces the impact of discriminative features in individual frames by allowing the model to incline towards robust features over a long temporal span. The adversarial objective function (4) is derived from (1) in Section II. In (4), we consider that \((M.C)\) represents the real data and \((G(C).C)\) represents the noise data.

\[
L_{Adv} = \min_{G} \max_{D} E_{C(M) \sim P(C,M)} \left[ \log D(M.C) \right] + E_{C \sim P(C)} \left[ \log(1 - D(G(C).C)) \right] + \lambda E_{C(M) \sim P(C,M)} \left\| G(C) - M \right\|^2
\]

where \((M)\) is the actual mask selected to identify the most robust features, and \(\lambda\) is a trade-off factor set with a value of 0.05.

In each training iteration, \(D\) is updated once by ascending its stochastic gradients to maximize the first and second terms in (4). Meanwhile, \(G\) is trained many times in each iteration by descending its stochastic gradients to minimize the second term in (4). The \(G\) model is regularized using the mean square error loss (MSE) which is represented by the third term in (4).

As adversarial learning only adopts positive samples, labels of fake samples are not smoothed in the same way as the real ones not to encourage the model to generate incorrect (or fake) samples, as mentioned in [39] and [40].

\[D^\ast(M.C) = \frac{(1 - \alpha)P_{C(M)}}{P_{C(M)} + P_{C}}\]

where \(\alpha\) is a small number which is less than 1. In our case its value is 0.1. Using one-sided label smoothing improves the performance of the discriminator during the training phase.

As adversarial learning only adopts positive samples, we compute the cost-sensitive loss (6) to reduce the effect of a large number of easy negative samples on cross-entropy loss and reformulate our final loss as in (7). This final loss function receives the discriminator output prediction score as the input and computes the loss value that is optimized for the lowest. Based on the cost-sensitive loss equation, which is based on entropy loss,

\[L(p, y) = - (y.(1 - p) \cdot \log(p) + (1 - y) \cdot p \cdot \log(1 - p))\]

we reformulate the loss function in (4) as:

\[L_{Adv} = \min_{G} \max_{D} E_{C(M) \sim P(C,M)} \left[ K_1 \log D(M.C) \right] + E_{C \sim P(C)} \left[ K_2 \log(1 - D(G(C).C)) \right] + \lambda E_{C(M) \sim P(C,M)} \left\| G(C) - M \right\|^2\]
where $K_1 = 1 - D(M.C)$ and $K_2 = D(G(C).C)$ are the modulating factors that balance training sample loss.

5) TRAINING OF BOUNDING BOX REGRESSION MODEL
The proposed network has no ability to tightly localize the coordinates of the BBox which encloses the target object. Therefore, we used a bounding box regression model, which is commonly used for detection and localization problems [41]. We randomly generated 1000 positive samples with IOU overlap ratio \( \geq 0.6 \) with the ground truth BBox and then trained a linear regression model that has three convolutional layers to precisely localize the target using its features, as in [41]. The bounding box regressor is trained only once in the first frame, and its weights are then fixed during the tracking phase.

B. TRACKING AND ONLINE FINE-TUNING
At the time of tracking, we removed the generator network. As shown in Fig. 6, the network structure of the tracking module consists of only two networks: feature extraction and classification. The pre-trained feature extraction consisted of three convolutional layers. The classification network consists of three fully connected layers similar to that used in the training module. The sequence of online tracking starts from the 2nd frame in the video and is performed as shown in the flowchart in Fig. 7. Through the tracking phase, we fine-tuned the tracking model by applying short-term and long-term updates as shown in the following subsections.

1) TRACKING
From the 2nd frame till the end of the video, random positive and negative candidate samples in the input frame are generated around the BBox of the previous frame. 50 positive candidate samples with IOU overlap ratio \( \geq 0.7 \) and 200 negative candidate samples with IOU overlap ratio \( \leq 0.3 \) with the last BBox generated. The feature extraction network extracts the CNN features. The extracted features were then fed into the classifier network. The classifier assigned a probability score to each candidate. The candidate proposal with the highest score was considered as the tracked object (8). The target location was estimated by adjusting the previous location using the trained bounding box regression model.

\[
I_{HS} = \arg\min \frac{1}{N} \sum_{j=1}^{N} \left( -\sum_{j=1}^{2} q(j) \log(p(j)) \right) \quad (8)
\]

where $I_{HS}$ is the index of the positive sample with the highest prediction score, $N$ is the number of samples, and $p$ and $q$ represent the distributions of the candidate samples and corresponding labels of the previous frame, respectively.

2) ONLINE FINE-TUNING
The tracker performs two updates for online fine-tuning. It should be able to recognize new features of the target with new appearance variations. A short-term update occurs when the tracking goes wrong (IOU overlap with the previous BBox $\leq 0.5$). In contrast, long-term updates occurred at regular intervals (every 10 frames). The last positive and negative samples observed in the short-term were used in the fine-tuning phase. In the short-term update, only the weights of the D network are updated using BCE loss and supervised learning; however, in the long-term update, both weights of the G and D are updated because of the new features using adversarial learning.
FIGURE 7. Flowchart of the proposed framework in the tracking module.

We regularized the discriminator by using one-sided label smoothing during the training phase to reduce the high confidence of labels of real samples. Here we apply label smoothing regularization (LSR) [42] to regularize the
classifier network during the short-term update to reduce the overfitting to the hard labels of class 1 and class 0. We modified the BCE loss into the formula in (11) to increase the robustness of the training during the short-term update.

\[
L_{LSR} = - (1 - \beta) \log(p(y)) - \beta \sum_{i=1}^{K} \log(p(i))
\]

where \(\beta\) is a small hyperparameter value between 0 and 1, \(i\) is the index of the samples, \(K\) is the number of classes (we have two classes only, object class (1) and background class (0)), \(p(i)\) represents the probability that the sample belongs to class \(i\), and \(p(y)\) represents the probability that the sample belongs to the ground-truth label. When \(\beta = 0\), the loss is calculated for the labeled training data, and when \(\beta = 1\), the loss is calculated for the unlabelled generated data. In this way, the synthesized data are integrated with the labeled training data to form a larger and more robust training set.

IV. EXPERIMENTAL RESULTS AND VALIDATION

In this section, we describe the implementation and evaluation of our proposed algorithms which are extensions to our preliminary models in [43]. We present an evaluation of the two proposed approaches, VITAL_DCGAN_LSR and VITAL_MLP_LSR, to extract the effects of each of the two architectures on the tracking performance. In the first approach (VITAL_DCGAN_LSR), we modified the original VITAL by using a different architecture for the generator network (5-layer DCGAN instead of the two fully connected layers) and applied LSR method to reduce overfitting. In the second approach (VITAL_MLP_LSR), we modified VITAL’s generator with 5-layer Leaky-ReLU MLP and also applied LSR method. The learning rates for training the G and D networks were 0.2 × 10⁻³ and 0.5 × 10⁻³, respectively. We compared our approaches with some of the state-of-the-art deep visual trackers: VITAL[3], MDNet[2], SORT[44], SSD[45], A3CT[46], A3CTD[47] and SiamRPN[48] and some recent visual trackers like SiamFC[49], CF[50] and USOT[51]. VITAL and MDNet algorithms are the top two competitors in the comprehensive survey paper on deep visual trackers [52], which was published in 2021. The evaluation metrics used in our experiments were the OTB-100 [53], VOT2019 [54] and LaSOT [55].

Hardware: All implementations were written in PyTorch and performed on a Linux-based cloud server using FluidStack. The cloud server had the following specifications: Ubuntu 18.04.5 LT, RAM: 114 GB, GPU: Nvidia RTX 2080, CPU: Intel(R) Xeon(R) Silver 4208 CPU @ 2.10G.

A. QUANTITATIVE EVALUATION

1) OTB-100 DATASET

We performed a comparison on some of the most well-known benchmark dataset for visual tracking OTB-100 [53], which has approximately 100 different videos. The videos were labelled using bounding box annotations for supervised learning. This dataset covers various challenging attributes, including visual tracking tasks (11 attributes), illumination variations, low resolution, scale variation, fast motion, background clutter, deformation, occlusion, out-of-view, motion blur, in-plane rotation, and out-of-plane rotation. As is commonly performed with the OTB dataset, the training samples are the first frames in all 100 sequences in the OTB dataset. Therefore, we used 100 images as the training set. The testing set contained all the remaining frames in the 100 sequences.

We followed the standard evaluation metrics followed by the OTB-100 benchmark dataset. We used the most common evaluation metric method, one-pass evaluation (OPE), which initializes tracking with the ground-truth state in the first frame and then reports the average precision and success rate in the remaining frames.

a) PRECISION

The Euclidean distance between the centers of the evaluated bounding box and manually labeled ground-truth bounding box was used to measure the performance of the trackers as shown in the following equation:

\[
\text{Euclidean distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}
\]

However, in cases where the target was lost, the distance was measured randomly. Therefore, it is better to measure the percentage of successful frames in which the distance between the evaluated bounding box and the ground-truth bounding box is within a given threshold (X-axis of the plot, in pixels).

b) SUCCESS

The precision metric reflects the pixel difference between the bounding boxes, but does not reflect the size and scale of the target object. Thus, in terms of the success rate, the evaluation depends on the overlap score. The overlap score was calculated using the following equation:

\[
S = \frac{|r_t \cap r_o|}{|r_t \cup r_o|}
\]

where \(r_t\) represents the tracked box, \(r_o\) represents the ground-truth box, \(\cap\) and \(\cup\) represent the intersection and union operators respectively, and \(|.|\) denotes the number of pixels in a given region. As the overlap score threshold varied between 0 and 1 on the x-axis, the success rate changed.

c) ABLATION STUDY

We introduce this analysis to demonstrate the effectiveness of using MLP layers instead of convolutional layers in the generator used to augment the input feature maps. Also, the analysis shows the impact of using label smoothing regularization method. Four models are used in this study. The first model is “random masks,” which does not use adversarial learning to generate masks during the training phase. The second model is “VITAL_DCGAN,” which uses
only convolutional layers in the generator network via adversarial learning. The third model is “VITAL-MLP,” which uses the MLP layers in the generator network via adversarial learning. The fourth model is “VITAL-MLP_LS” which uses one-sided label smoothing method during the offline training phase and the label smoothing regularization during the short-term update in the tracking phase. Fig. 8 shows the results for the OTB-100 dataset. From the results, we deduce that generating masks without adversarial learning causes a random selection of robust features. Therefore, sometimes robust features are selected, and sometimes discriminative features in individual frames are chosen. It is clear that the precision and success of using random masks are low compared with those of the other two models that use adversarial learning. In addition, when we replaced the convolutional layers with 5 MLP layers in the generator, the tracker performance improved in terms of both precision and success. Accordingly, this study emphasizes that using MLP layers for the generator is more effective than convolutional layers in extracting general non-spatial features that last for a long time. Using these general features makes the tracking procedure more invariant to appearance attributes and less susceptible to overfitting. Finally, adding label smoothing regularization to our method increased the generalization of the algorithm and decreased overfitting which achieved more robust tracking results.

d) COMPARISONS WITH STATE-OF-THE-ART ALGORITHMS
The results are shown in Fig. 9 and Table. 1. We used the average values of precision and success rates, and the area under the curve (AUC) to empirically study the performance of the proposed frameworks compared with state-of-the-art algorithms. Clearly, we can make some observations. First, the proposed VITAL_MLP_LSR tracker is a strong competitor to the MDNet tracker. Second, the VITAL_MLP_LSR model is less model-specific and can track objects with different appearance variations better than VITAL model and the other compared models. Third, the improvement in using...
the MLP architecture as a generator is more salient than
the improvement in using the DCGAN architecture in terms
of robustness and efficiency. As a result, we can conclude
that using MLP layers in the generator can extract more
robust nonspatial features than the convolutional-only layers,
which can extract discriminative spatial information in the
individual frames.

Regarding speed, the SiamCorners, CF_ML and USOT
algorithms appear to be the fastest three trackers, as shown
in Table 1.

2) VOT2019 DATASET
In the OTB dataset, the tracker is initialized only by the
annotated bounding boxes in the first frames. Once a failure
occurred, the average overlap became zero until the end
of the video. Similar to all VOT datasets, VOT2019 [54]
applies a reset-based procedure by reinitializing the tracker
after five frames of each drift far from the target. It uses
three metrics to measure the tracker performance; robustness
(R), accuracy (A), and expected average overlap (EAO).
During successful tracking periods, accuracy is defined as
the average overlap between the predicted and ground truth
bounding boxes. The robustness of the tracker is measured by
the number of times it loses its target (fails) during tracking.
The expected average overlap (EAO) combines these two
metrics. VOT produces a state-of-the-art bound (SotA bound)
value over all its benchmarks to reduce the pressure of
fine-tuning to benchmarks by attempting to break the best
ranks. Any tracker that exceeds the SotA bound value (0.263)
is considered a state-of-the-art tracker. VOT2019 contains
four separate challenges; each one has a different dataset
with specific features. The challenge whose dataset had the
same features as our tracker was the VOT-ST2019 challenge.
It addresses short-term tracking in RGB images. In a short-
term setup, the target remains within the camera field of
vision throughout the sequence but may experience partial

| Tracker | VITAL-MLP/LSR | VITAL-DCGAN/LSR | VITAL [3] | MDNet [2] | SORT [44] | SSD [45] | A3CT [46] | Siam-Corners [49] | CF_ML [50] | USOT [51] |
|---------|---------------|-----------------|-----------|-----------|-----------|----------|------------|-----------------|------------|----------|
| Precision | 0.8578 | 0.8529 | 0.85237 | 0.85437 | 0.6856 | 0.5254 | 0.8365 | 0.832 | 0.853 | 0.813 |
| AUC | 0.75498 | 0.7415 | 0.7471 | 0.7497 | 0.5756 | 0.4967 | 0.7479 | 0.715 | 0.699 | 0.682 |
| FPS | 4.03803 | 3.9083 | 4.07215 | 4.61800 | 3.8746 | 4.6078 | 4.0364 | 44.3 | 58.9 | 37.7 |

| Tracker | VITAL-MLP/LSR | VITAL-DCGAN/LSR | VITAL [3] | MDNet [2] | SiamRPNpp [48] | SORT [44] | SSD [45] | A3CT [46] | Siam-Corners [49] | CF_ML [50] | USOT [51] |
|---------|---------------|-----------------|-----------|-----------|--------------|-----------|----------|------------|-----------------|------------|----------|
| Accuracy | 0.601 | 0.58 | 0.51 | 0.51 | 0.599 | 0.42 | 0.47 | 0.49 | 0.52 | 0.534 | 0.507 |
| Robustness | 82 | 93 | 85 | 99 | 100 | 145 | 142 | 100 | 84 | 97 | 101 |
| EAO | 0.304 | 0.281 | 0.294 | 0.274 | 0.285 | 0.2631 | 0.253 | 0.271 | 0.29 | 0.283 | 0.269 |

| Tracker | VITAL-MLP/LSR | VITAL-DCGAN/LSR | VITAL | MDNet | SiamRPNpp | A3CTD | Siam-Corners | CF_ML | USOT |
|---------|---------------|-----------------|-------|-------|----------|-------|--------------|-------|-------|
| Success | 0.487 | 0.325 | 0.390 | 0.397 | 0.495 | 0.415 | 0.480 | 0.305 | 0.358 |
| Precision | 0.49 | 0.291 | 0.360 | 0.373 | 0.493 | 0.378 | 0.406 | 0.287 | 0.340 |
| Normalized Precision | 0.569 | 0.352 | 0.453 | 0.460 | 0.570 | 0.454 | 0.555 | 0.341 | 0.398 |
FIGURE 10. Comparison between the form of the features in the feature maps and the generated ‘M’ masks in 10th and 50th frames.

FIGURE 11. Tracking results of the compared trackers on the ‘Basketball’ video (frames: 473 to 477).

short-term occlusions. TABLE 2 shows that the proposed algorithm has a favorable performance compared to state-of-the-art trackers. Our framework is the best in terms of accuracy, robustness and EAO. The strongest trackers on VOT2019 dataset after VITAL_MLP_LSR are: VITAL, SiamRPNpp and SiamCorners. All the compared models are considered state-of-the-art trackers because they exceed the SotA bound value except SSD.
FIGURE 12. Tracking results of the compared trackers on the 'BlurOwl' video (frames: 211 to 215).

FIGURE 13. Tracking results of the compared trackers on the 'Bolt2' video (frames: 200 to 204).
3) LaSOT DATASET
LaSOT [55] is a high-quality benchmark for large-scale single object tracking which has a large number of long-term videos. It contains 1400 videos that had between 1000 and 11397 frames per video. The test dataset comprises 280 videos and the training dataset contains 1120 videos. Each sequence includes a variety of difficulties drawn from the natural world, where the targets could disappear and then appear again in the view. We compared the two proposed methods with the state-of-the-art trackers. We compared our two trackers with the following trackers: VITAL, MDNet, SiamPRNpp, A3CTD, SiamCorners, CF_ML and USOT in terms of success, precision and normalized precision. As shown in Table 3, the proposed VITAL_MLP_LSR is a strong competitor to SiamPRNpp tracker despite of not using long-term strategies.

B. QUALITATIVE EVALUATION
This section analyzes how using adversarial learning and merging it in the middle of a CNN network achieves better results. As shown in Fig. 10, the feature maps in (b) and (e) extracted by the three convolutional layers are not sufficiently accurate to describe the main features of the tracked object. They contain both the main robust and discriminative features that appear in this frame. They have all features representing the appearance variations in this frame, such as scale variation, rotation, fast motion, and illumination variation, etc. However, the adversarially learned masks select the robust features to be picked and stored along the diagonals and the other discriminative features to be discarded, as shown in (c) and (f). The masks were transformed into $1 \times 4608$ flattened arrays for each sample. For example, a $3 \times 3$ patch was transformed to $512 \times 3 \times 3$ to apply masks to the extracted features. Therefore, we converted $512 \times 3 \times 3$ into a 2d array and displayed it as shown in Fig. 10. Each of the nine masks was assigned a probability score. We consider the mask with a higher probability of being stored as an M mask and dropping out other remaining masks. The better the robustness of the features in M, the better are the tracking results. The best contribution of the proposed network is that it allows M to store the robust features of the objects in the background to learn to avoid them during the tracking process.

For a qualitative comparison of the performances of the compared approaches, we selected five sequences from the OTB-100 dataset with different challenging attributes to show some samples of the tracking output. Fig. 11, 12, 13, 14, and 15 show the ability of each tracker to follow the target for each case of critical visual attributes. Unlike the other three adversarial learning methods, MDNet is trained to find discriminative features in individual frames, which may cause overfitting. So, it is not easy for it to deal with occlusion and out-of-plane-rotation attributes like in ‘Basketball’ (Fig. 11) and ‘Bird2’ (Fig. 15) sequences.
On the other hand, VITAL and VITAL_DCGAN_LSR methods do not perform well with scale variation and motion blur like in ‘BlurOwl’ (Fig. 12) and ‘Box’ (Fig. 14) sequences. The size of the generated masks was fixed. Therefore, they cannot handle scale variation attributes. Moreover, motion blur makes all the features, discriminative or robust, similar because they are all blurred. Therefore, using the generated masks to select the most robust features did not work in this case. In contrast, the figures show that using the VITAL method with MLP layers in the generator presents a good visual tracking competitor to state-of-the-art deep trackers and has a good ability to deal with most of the hard conditions and appearance variations.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduce two deep visual tracking architectures by designing new masks generated by the DCGAN architecture generator in the first approach and the MLP architecture in the second approach. These masks acted as the second stage in the feature extraction network used in the CNN classifier. In addition, using label smoothing regularization improved the performance as it only decreases the overfitting to the correct class without encouraging the model to choose an incorrect class in the training set. The proposed tracker can improve the robustness of the feature maps of positive samples of training data and preserve meaningful features under difficult circumstances. This shows that fully connected layers have a better ability to track the main features of the target, which lasts for a long temporal span. However, convolutional layers are better at tracking discriminative spatial features in individual frames. Our VITAL_MLP_LSR algorithm shows competitive results against the state-of-the-art deep visual trackers on the large visual tracking benchmarks OTB-100, VOT2019 and LaSOT.

Integrating GANs with CNNs is a rich area of research, and it is expected to improve over the next few years. Therefore, we expect that all these improvements will have a significant effect when used to enhance the quality of the feature extraction process in CNN by generating more efficient masks in the same way as in the proposed method. This field is open to try all variants of GANs for the same purpose. Besides, we are suggesting the application of the proposed algorithm with multi-class tracking algorithms because the label smoothing regularization has more effect in case of multi-object tracking.

REFERENCES

[1] P. Li, D. Wang, L. Wang, and H. Lu, “Deep visual tracking: Review and experimental comparison,” Pattern Recognit., vol. 76, pp. 323–338, Apr. 2018.

[2] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 4293–4302.
[3] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang, “VITAL: Visual tracking via adversarial learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8990–8999.

[4] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” CoRR, vol. abs/1511.06434, pp. 1–12, Nov. 2015.

[5] J. Jung, J. Son, M. Baek, and B. Han, “Real-time MDNet,” in Proc. ECCV, 2018, pp. 89–104.

[6] H. Li, X. Wang, F. Shen, Y. Li, F. Porikli, and M. Wang, “Real-time deep tracking via corrective domain adaptation,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 9, pp. 2600–2612, Sep. 2019.

[7] S. Pu, Y. Song, C. Ma, H. Zhang, and M. H. Yang, “Deep attentive tracking via reciprocative learning,” in Proc. NIPS, 2018, pp. 1931–1941.

[8] H. Nam, M. Baek, and B. Han, “Modeling and propagating CNNs in a tree structure for visual tracking,” 2016, arXiv:1608.07242.

[9] E. Burceanu and M. Leordeanu, “Learning a robust society of trackers,” in Proc. CVPR, 2010.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in Proc. NIPS, 2017.

[11] H. Zhao, G. Yang, D. Wang, and H. Lu, “Deep mutual learning for visual object tracking,” Pattern Recognit., vol. 52, pp. 847–857, Apr. 2020.

[12] J. Guo, T. Xu, S. Jiang, and Z. Chen, “Generating reliable online adaptive templates for visual tracking,” in Proc. 25th IEEE Int. Conf. Image Process. (ICIP), Oct. 2018, pp. 226–230.

[13] Y. Yin, D. Xu, W. Wang, and L. Zhang, “Adversarial feature sampling learning for efficient visual tracking,” IEEE Trans. Autom. Sci. Eng., vol. 17, no. 2, pp. 847–857, Apr. 2020.

[14] Z. Zhang and H. Peng, “Deeper and wider Siamese networks for real-time visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 4904–4913.

[15] X. Li, Q. Liu, N. Fan, Z. Zhou, Z. He, and X.-Y. Jing, “Dual-regression model for visual tracking,” Neural Netw., vol. 132, pp. 364–374, Dec. 2020.

[16] Z. Zhang and H. Peng, “Deeper and wider Siamese networks for real-time visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4591–4600.

[17] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, “SiamRPN++: Evolution of Siamese visual tracking with very deep networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4282–4291.

[18] H. Nam, M. Baek, and B. Han, “Modeling and propagating CNNs in a tree structure for visual tracking,” 2016, arXiv:1608.07242.

[19] E. R. AlBasiouny et al.: Multilayer Perceptron Generative Model via Adversarial Learning

[20] J. Guo, T. Xu, S. Jiang, and Z. Chen, “Generating reliable online adaptive templates for visual tracking,” in Proc. 25th IEEE Int. Conf. Image Process. (ICIP), Oct. 2018, pp. 226–230.

[21] F. Zhao, J. Wang, Y. Wu, and M. Tang, “Adversarial deep tracking,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 7, pp. 1998–2011, Jul. 2019.

[22] X. Li, Q. Liu, N. Fan, Z. Zhou, Z. He, and X.-Y. Jing, “Dual-regression model for visual tracking,” Neural Netw., vol. 132, pp. 364–374, Dec. 2020.

[23] Z. Zhang and H. Peng, “Deeper and wider Siamese networks for real-time visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4591–4600.

[24] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, “SiamRPN++: Evolution of Siamese visual tracking with very deep networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4282–4291.

[25] H. Nam, M. Baek, and B. Han, “Modeling and propagating CNNs in a tree structure for visual tracking,” 2016, arXiv:1608.07242.

[26] E. R. AlBasiouny et al.: Multilayer Perceptron Generative Model via Adversarial Learning

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in Proc. NIPS, 2017.

[28] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, “Improving the improved training of Wasserstein GANs: A consistency term and its dual effect,” in Proc. ICLR, 2018.

[29] H. Pu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, and D. Tao, “Geometry-consistent generative adversarial networks for one-sided unsupervised domain adaptation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2427–2436.

[30] Y. Yin, D. Xu, W. Wang, and L. Zhang, “Adversarial feature sampling learning for efficient visual tracking,” IEEE Trans. Autom. Sci. Eng., vol. 17, no. 2, pp. 847–857, Apr. 2020.

[31] J. Guo, T. Xu, S. Jiang, and Z. Chen, “Generating reliable online adaptive templates for visual tracking,” in Proc. 25th IEEE Int. Conf. Image Process. (ICIP), Oct. 2018, pp. 226–230.

[32] F. Zhao, J. Wang, Y. Wu, and M. Tang, “Adversarial deep tracking,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 7, pp. 1998–2011, Jul. 2019.

[33] X. Li, Q. Liu, N. Fan, Z. Zhou, Z. He, and X.-Y. Jing, “Dual-regression model for visual tracking,” Neural Netw., vol. 132, pp. 364–374, Dec. 2020.

[34] Z. Zhang and H. Peng, “Deeper and wider Siamese networks for real-time visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4591–4600.

[35] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, “SiamRPN++: Evolution of Siamese visual tracking with very deep networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4282–4291.

[36] H. Nam, M. Baek, and B. Han, “Modeling and propagating CNNs in a tree structure for visual tracking,” 2016, arXiv:1608.07242.

[37] E. R. AlBasiouny et al.: Multilayer Perceptron Generative Model via Adversarial Learning

[38] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of Wasserstein GANs,” in Proc. NIPS, 2017.

[39] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, “Improving the improved training of Wasserstein GANs: A consistency term and its dual effect,” in Proc. ICLR, 2018.

[40] H. Pu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, and D. Tao, “Geometry-consistent generative adversarial networks for one-sided unsupervised domain adaptation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2427–2436.

[41] Y. Yin, D. Xu, W. Wang, and L. Zhang, “Adversarial feature sampling learning for efficient visual tracking,” IEEE Trans. Autom. Sci. Eng., vol. 17, no. 2, pp. 847–857, Apr. 2020.
[53] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[54] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. C. Zajc, O. Drbohlav, A. Lukeziec, A. Berg, and A. Eldesokey, “The seventh visual object tracking VOT2019 challenge results,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2206–2241.

[55] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, “LaSOT: A high-quality benchmark for large-scale single object tracking,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5374–5383.

**EMAN R. ALBASIOUNY** received the B.Sc. degree from the Computer Science and Engineering Department, Kafrelsheikh University, Kafrelsheikh, Egypt, in 2008, and the M.Sc. degree from Tanta University, Tanta, Egypt, in 2016. She is currently an Assistant Lecturer with Kafrelsheikh University. Her research interests include visual tracking, deep learning, and augmented reality.

**ABDEL-FATTAH ATTIA HELIEL** received the Graduate degree, in 1991, the M.Sc. degree from Ain Shams University, Cairo, Egypt, in 1997, and the Ph.D. degree in control engineering and robotics from Czech Technical University in Prague, Czech Republic, in 2002. He is currently the Dean of the Faculty of Engineering, Kafrelsheikh University, and he is the Founder and the Head of the Intelligent Systems Research Group (ISRG). The group has expertise in other techniques, such as neuro-modeling control and heuristic optimization techniques. These algorithms can be applied in many applications, such as power systems and renewable energy. Group members have several publications in the field of power energy. He has more than 45 scientific research articles published in prestigious international journals. He co-supervised nine graduate students (four Ph.D. and five M.Sc. students) at Ain Shams University. His research interests include the development, modeling, identification of nonlinear systems, fuzzy models, neural networks, neuro-fuzzy, and genetic algorithms. He is a Referee for many international journals, such as *Electric Power Components and Systems*, *Journal of Electrical Engineering*, *Applied Soft Computing*, *Soft Computing*, and *IEEE Transactions on Systems Science and Cybernetics*.

**HOSSAM E. ABDEL-MUNIM** received the B.Sc. and M.S. degrees in electrical engineering from Ain Shams University, Cairo, Egypt, in 1995 and 2000, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Louisville, Louisville, KY. He joined the Computer Vision and Image Processing Laboratory (CVIP Laboratory), University of Louisville, in June 2002, where he has been involved in the applications of image processing and computer vision for medical image analysis. He is currently a Full Professor at the Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University. His current research interests include image modeling, image segmentation, and deep learning, about which he has authored or coauthored more than 75 technical articles, including *ICIP, MICCAI, ICCV, CVPR, BMVC, DICTA, IEEE Transactions on Image Processing, IEEE Transactions on Pattern Analysis and Machine Intelligence*, and *JRTIP*.

**HAZEM M. ABBAS** (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Ain Shams University, Cairo, in 1983 and 1988, respectively, and the Ph.D. degree from Queen’s University, Canada, in 1993. He is currently a Professor in computer and systems engineering. He has served as the Chairperson of the Department of Computer and Systems Engineering, Ain Shams University, and the Dean of the Faculty of Media Engineering and Technology, German University, Cairo. He worked at the Royal Military College, Kingston, Canada, the IBM Toronto Laboratory, and Mentor Graphics. He chaired the IEEE Signal Processing Chapter in Cairo. His research interests include machine learning and computational intelligence.

---

**EMAN R. ALBASIOUNY**

**ABDEL-FATTAH ATTIA HELIEL**

**HOSSAM E. ABDEL-MUNIM**

**HAZEM M. ABBAS**