UNIVERSITY OF CALIFORNIA
RIVERSIDE

Increasing the Robustness of Deep Learning Models using Generative Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Rajkumar Theagarajan

June 2020

The Dissertation of Rajkumar Theagarajan is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I thank my dissertation chair, Dr. Bir Bhanu, who never gives up on motivating me to finish my degree. I would like to thank Dr. Yingbo Hua and Dr. Matthew Barth for being my committee members on my dissertation defense. I would also want to thank Dr. Salman Asif, Dr. Aaron Seitz and Dr. Ertem Tuncel for conferring my doctoral candidacy. I am grateful for the friends/colleagues who were part of my graduate life: Adam Witmer, Alex Woonggi Shin, Ankit Jain Rakesh Kumar, Asong Tambo, Hengyue Liu, Runze Li, Saisri Padmaja Jonnalagedda, Vincent On and Xiu Zhang. I also want to thank Dr. Ninad S. Thakoor, Dr. Federico Pala, Dr. Jing Zhang, and Dr. Ming Chen for their discussions as well as their support on my publications. I am extremely grateful to Benjamin X. Guan, Ashwini Shandilya, Eric Ebert and Don Ebert for helping us in collecting the data for the projects. My research was supported by NSF Grant 1552454 and 191197, ONR grant N00014-12-1-1026 and a gift from SEVAai. Inc. This dissertation was compiled from "DeephESC 2.0: Deep Generative Multi Adversarial Networks for improving the classification of hESC", "An Automated System for Generating Tactical Performance Statistics for Individual Soccer Players from Videos", and "ShieldNets: Defending Against Adversarial Attacks Using Probabilistic Adversarial Robustness" by R. Theagarajan *et al.* © 2019, 2020 the IEEE.

To my mother Manimekalai Theagarajan, I thank you for your support and understanding on my pursuit for higher education without her help, I would not have been here. I also thank both of my uncles Dr. Mathiazhagan Chakrawarthy, and Dr. Anbazhagan Chakrawarthy, for their encouragement and Meghan H. Keiser for all her love and support.

ABSTRACT OF THE DISSERTATION

Increasing the Robustness of Deep Learning Models using Generative Networks

by

Rajkumar Theagarajan

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2020
Dr. Bir Bhanu, Chairperson

Over the past few years deep learning has demonstrated impressive performance on many important practical problems such as image, video, and audio classification. This work develops three novel applications for automated stem cell classification, automated sports analytics, and a novel framework for defending deep learning models from white and black box adversarial attacks. In the field of stem cell classification, it is very expensive and time consuming to generate data which is very intrusive and not easy to obtain. This work leverages an ensemble of generative networks to create a large dataset of synthetic human embryonic stem cell (hESC) images which are exclusively used for training deep learning classifiers. In order to verify that the data distribution of the synthetic images is similar to that of the real-world images, the quality of the synthetic images are validated at a pixel level and high dimensional feature level with respect to the real-world data. Experimental results show that the classifiers trained on the synthetic dataset are able to achieve high performance when evaluated on real-world data and can be used as a tool for annotating more data saving hours of manual labor.

In the field of automated sports analytics, it is very important to analyze every minute detail in order to generate reliable statistics for every individual player. This work develops a novel framework for automatically generating the tactical statistics of soccer players directly from a video. The proposed approach empirically shows that high-level features learned from specific soccer matches do not necessarily generalize across all soccer matches and it is not feasible to obtain datasets for every single match. To solve this, the proposed approach develops a match-specific application that uses previously recorded videos of teams to learn fine-grained features that can generalize across other matches played by the same respective teams. Although generative networks have had huge success in augmenting existing datasets which improve the performance of deep learning classifiers, this work shows that they often overlook minute details when generating new data which is very important in sports analytics and can cause the performance of the classifiers to drop. This work proposes a novel generative architecture that learns to generate synthetic images with fine-grained structures which further improves our system to generate accurate tactical statistics for the players. Various ablation studies are performed to show the improvement in performance and significance of the results across different soccer matches.

Despite their outstanding performance, these models are vulnerable to adversarial manipulation of their input which could lead to poor performance. These adversarial manipulations are carefully crafted perturbations that are so subtle that a human observer does not even notice the modification at all, but can cause deep learning models to predict incorrect results. In order to address this vulnerability, this work proposes a novel white box defense algorithm that uses generative networks with Probabilistic Adversarial Robustness to neutralize adversarial examples by concentrating the sample probability to adversarial-free zones. Although, our proposed defense achieves state-of-the-art classification accuracy, this is not a reliable metric to determine if an image is "adversarial-free". This is a foundational problem for online image verification applications where the ground-truth of the input image is not known and hence we cannot validate the performance of the classifier or know if the image is "adversarial-free" or not. To address this problem, this work proposes a novel framework that uses an ensemble of individual defenses whose performance is continuously validated in a loop using Bayesian uncertainties and does not require any information about the black box classifier such as its architecture, parameters, or training dataset. Unlike existing defense mechanisms that requires knowing the ground-truth of the input data and modifying/re-training the black box classifier which is not feasible in online applications, our defense is designed in the first place to provide proactive protection to any existing deep learning based model. Evaluation on various public benchmark datasets including autonomous driving and face biometrics datasets shows that our defense can consistently detect adversarial examples and purify them against a variety of attacks with different ranges of perturbations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the field of stem cell classification, Human embryonic stem cells (hESC), derived from the blastocysts, provide unique cellular models for numerous potential applications. They have great promise in the treatment of diseases such as Parkinson's, Huntington's, diabetes mellitus, etc. hESC are a reliable developmental model for early embryonic growth because of their ability to divide indefinitely (pluripotency), and differentiate, or functionally change, into any adult cell type. Their adaptation to toxicological studies is particularly attractive as pluripotent stem cells can be used to model various stages of prenatal development. Automated detection and classification of human embryonic stem cell in videos is of great interest among biologists for quantified analysis of various states of hESC in experimental work. Currently video annotation is done by hand, a process which is very time consuming and exhaustive. To solve this problem, this thesis introduces DeephESC 2.0 an automated machine learning approach consisting of two parts: (a) Generative Multi Adversarial Networks (GMAN) for generating synthetic images of hESC, (b) a hierarchical classification system consisting of Convolution Neural Networks (CNN) and Triplet CNNs to classify phase contrast hESC images into six different classes namely: *Cell clusters*, *Debris*, *Unattached cells*, *Attached cells*, *Dynamically Blebbing cells* and *Apoptically Blebbing* *cells*. The approach is totally non-invasive and does not require any chemical or staining of hESC. DeephESC 2.0 is able to classify hESC images with an accuracy of 93.23% out performing state-of-the-art approaches by at least 20%. Furthermore, DeephESC 2.0 is able to generate large number of synthetic images which can be used for augmenting the dataset. Experimental results show that training DeephESC 2.0 exclusively on a large amount of synthetic images helps to improve the performance of the classifier on original images from 93.23% to 94.46%. This thesis also evaluates the quality of the generated synthetic images using the Structural SIMilarity (SSIM) index, Peak Signal to Noise ratio (PSNR) and statistical p-value metrics and compares them with state-of-the-art approaches for generating synthetic images. DeephESC 2.0 saves hundreds of hours of manual labor which would otherwise be spent on manually/semi-manually annotating more and more videos.

The world of sports intrinsically involves fast and complex events that are difficult for coaches, trainers and players to analyze, and also for audiences to follow. In fast paced team sports such as soccer, keeping track of all the players and analyzing their performance after every match are very challenging. Current scenarios for identifying the best talents in soccer involve word-of-mouth and coaches/recruiters scouring through hours of manually annotated videos. This is a very expensive and laborious process and also biased by the nature of the recruiters. To alleviate these problems, this paper proposes an automated system that can detect, track, classify the teams of multiple players and identify the player controlling the ball in a video. The system generates three very important tactical statistics for a player: 1) duration of ball possession, 2) number of successful passes and 3) number of successful steals. This is done by training Convolutional Neural Networks (CNNs) to (a) localize and track the players on the field, (b) classify the team of a detected player, (c) identify the player controlling the ball and (d) pooling all the information extracted from (a), (b), and (c) to generate the statistics of players. To overcome the problem that the features learned from specific soccer matches do not necessarily generalize across differ-

ent soccer matches, the thesis proposes minimal amount of match-specific annotation and data augmentation, using a variant of Deep Convolutional Generative Adversarial Networks (DCGAN) to improve the accuracy. Experimental results and ablation studies show that the proposed approach outperforms the state-of-the-art approaches in terms of accuracy and processing speed.

Deep learning models have achieved impressive results on various tasks but, these models have been shown to be vulnerable to adversarial attacks. Adversarial attacks are imperceptible perturbations added to an image such that the deep learning model misclassifies the image with high confidence. This thesis introduces Probabilistic adversarial robustness (PAR), as a theoretical framework to neutralize adversarial attacks by concentrating sample probability to adversarial-free zones. Distinct to most of the existing defense mechanisms that require modifying the architecture/training of the target classifier which is not feasible in the real-world scenario, e.g., when a model has already been deployed, PAR is designed in the first place to provide proactive protection to an existing fixed model. ShieldNet is implemented as a demonstration of PAR in this work by using PixelCNN. Experimental results show that this approach is generalizable, robust against adversarial transferability and resistant to a wide variety of attacks on the Fashion-MNIST and CIFAR10 datasets, respectively.

Existing adversarial defenses validate their performance using only the classification accuracy. However, classification accuracy by itself is not a reliable metric to determine if the resulting image is "adversarial-free". This is a foundational problem for online image recognition applications where the ground-truth of the incoming image is not known and hence we cannot compute the accuracy of the classifier or validate if the image is "adversarial-free" or not. This thesis proposes a novel framework for defending Black box classifiers from adversarial attacks using an ensemble of iterative adversarial image purifiers whose performance is continuously validated in a loop using Bayesian uncertainties.

The proposed approach can convert a single-step black box adversarial defense into an iterative defense and proposes three novel Knowledge Distillation approaches that use prior information from various datasets to mimic the performance of the Black box classifier. Additionally, this thesis also proves the existence of an optimal distribution for the purified image to reach a theoretical lower bound, beyond which the image can no longer be purified. Experimental results on six public benchmark datasets namely: 1) Fashion-MNIST, 2) CIFAR-10, 3) GTSRB, 4) MIO-TCD, 5) Tiny ImageNet, and 6) MS-Celeb show that the proposed approach can consistently detect adversarial examples and purify/reject them against a variety of adversarial attacks.

Chapter 2

DeephESC 2.0: Deep Generative Multi Adversarial Networks for improving the classification of hESC

Human embryonic stem cells (hESC) are derived from the inner cell mass of developing blastocysts and can be maintained indefinitely in vitro in a pluripotent state [1]. hESC have the ability to self-renew and differentiate into any cell type, thus providing a unique resource for regenerative medicine and toxicological testing of drugs [2, 3]. The biologists who study hESC have to manually analyze stem cell videos every day. On an average it takes 3-5 days for a biologist to manually analyze a single hESC video, taken over a period of 48 hours with a suitable sampling rate, and annotate its different stages of development.

To date, there are very limited automated tools [4, 5] for classifying hESC from videos making it a very laborious manual process. Video Bioinformatics [6–10] is an up-

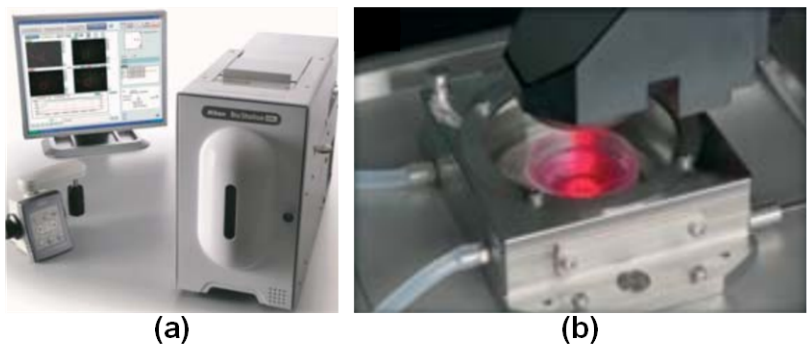

Fig. 1: The Nikon BioStation IM benchtop live cell imaging system. (a) External features include a incubation unit, joystick for controlling the position of the camera during sample selection, and a monitor. (b) Culture dish sitting inside the BioStation IM incubator unit.

coming field to help biologists use efficient and effective approaches to analyze expansive volumes of video data. In this study,the hESC videos were recorded using a Nikon BioStation IM [32] which has a phase contrast microscope. Each frame in the video can contain any number of the following six cell types: 1) *Cell clusters* ***(CC)***, 2) *Debris* ***(DEB)***, 3) *Unattached Cells* ***(UN)***, 4) *Attached Cells* ***(AT)***, 5) *Dynamically Blebbing Cells* ***(DYN)***, and 6) *Apoptotically Blebbing cells* ***(APO)***. Fig. 1 shows the Nikon BioStation IM and Fig. 2 shows the hESC phase contrast images that have been detected and cropped from full frame images for each class. **It should be noted that, our approach is totally non-invasive and does not require chemicals for staining the hESC.**

The *Unattached cells*, *Attached cells*, *Dynamically Blebbing cells* and *Apoptically Blebbing cells* are considered as the intrinsic cell types. *Cell clusters* are a colony of growing cells consisting of a group of two or more different intrinsic cell types that are packed close to each other. Blebbing cells are membrane protrusions that appear and disappear from the surface of cells. The changing area of the blebbing cells over time is important for understanding and evaluating the health of cells. *Dynamic blebs* indicate healthy cells and *Apoptotic blebs* indicate dying cells. The ability to analyze rates of bleb formation and

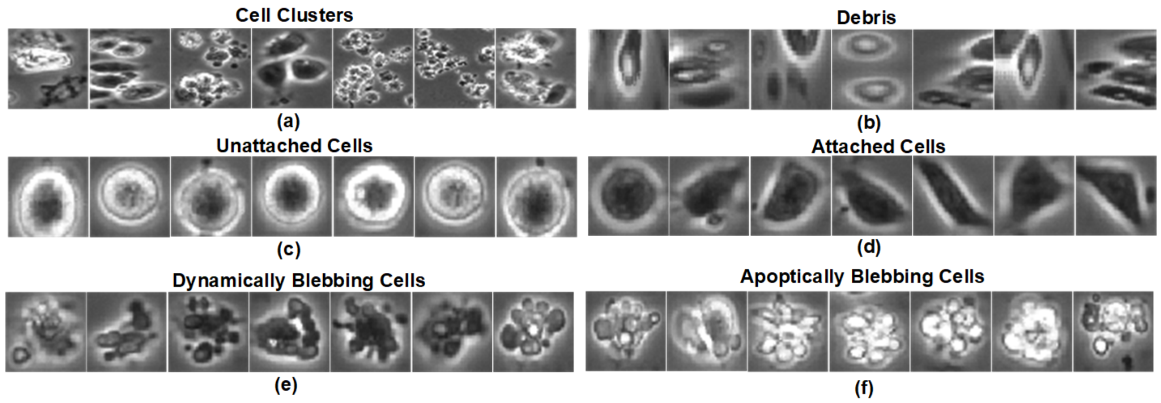

Fig. 2: Phase contrast images for the six different classes of hESC obtained from the Nikon BioStation IM.

retraction are important in the field of toxicology and could form the basis of an assay that depends on a functional cytoskeleton [33].

From Fig. 2, it can be observed that although certain classes such as *Debris* and *Unattached cells* look very discriminative compared to the remaining four classes. Certain classes like *Attached cells* and *Dynamically Blebbing cells* share very similar color intensities, similarly *Cell clusters* and *Apoptically Blebbing cells* share very similar texture making making it very challenging to classify these hESC classes.

Previous studies involving the classification of hESC have primarily used manual/ semi-manual detection and segmentation [34], hand-crafted feature extraction [4]. These manual methods, hand-crafted feature extraction approaches are prone to human bias and they are tedious and time-consuming processes when performed on a large volume of data. Therefore, it is advantageous to develop an image analysis software such as DeephESC 2.0 to automatically classify hESC images and also generate synthetic data to compensate for the lack of real data.

Recent years have witnessed the boom of CNNs in many computer vision and pattern recognition applications including object classification [28], object detection [29] and semantic segmentation [30]. In this chapter, we propose DeephESC 2.0, an automated machine learning based classification system for classifying hESC images using Convolution Neural Networks (CNN) and Triplet CNNs in a hierarchical system. The CNNs are trained on a very limited dataset consisting of phase contrast imagery of hESC to extract discriminative and robust features to automatically classify these images. This is not a straight forward task as some classes of hESC have very similar shape, intensity and texture. To solve this we trained triplet CNNs that help extract very fine-grained features and classify between two very similar but slightly distinctive classes of hESC. DeephESC 2.0 uses a CNN and two triplet CNNs fused together in a hierarchical manner to perform fine-grained classification on six different classes of hESC images. Previous studies have shown that augmenting the size and diversity of the dataset, results in improved classification accuracy [31].

The process of obtaining video recordings of hESC is a very long and tedious process, and to date there are no publicly available datasets. To compensate for the lack of data, DeephESC 2.0 uses Generative Multi Adversarial Networks (GMANs) to generate synthetic hESC images and augment the training dataset to further improve the classification accuracy. We compare different architectures of Generative Adversarial Networks (GANs) and the quality of the generated synthetic images using the Structural SIMilarity (SSIM) index and Peak Signal to Noise Ratio (PSNR). Furthermore, we trained DeephESC 2.0 using the synthetic images, evaluated it on the original hESC images obtained from biologists and verified the significance of our results using the p-value.

2.1 Related Work and Our Contributions

In the following we present the related work into the following three areas: detection of hESC in video, classification of hESC images and generation of synthetic hESC images

2.1.1 Detection of hESC in Video

Ambriz-Colin *et al.* [11] proposed two methods for cell region detection from phase contrast images: detection by pixel Intensity Variance (PIV) and detection by Gray Level Morphological Gradient (GLMG). The PIV method computes the variance of a pixel in a given neighborhood and based on a threshold classifies if the pixel belongs to a cell region or background. The GLMG approach converts the phase contrast image to a binary image and performs morphological dilation and erosion and based on a threshold separates the cell region and background. Li *et al.* [12] used a combination of morphological rolling-ball filtering and a Bayesian classifier to classify the pixels into either the cell regions or the background. The major drawback with these approaches is that they are very susceptible and would fail to classify the pixels even if there is slight change in pixel intensity or change in texture which normally occurs over time.

Eom *et al.* [13] used circular Hough transform to detect the shapes of cells in an image and classify them. This approach is very sensitive to the variation of shapes and appearance of cells. This approach is not viable for detection of hESC where blebbing is continuously altering the shape of the hESC. Miroslaw *et al.* [14] proposed to use correlation using template images for cell region detection. This approach requires pre-selection of exemplar template images which are not readily available in most cases. Moreover, this approach is most likely to fail in conditions where parts of two or more cells are overlapping in a single image.

The most commonly used algorithms for image segmentation are the K-means segmentation and mixture of Gaussians by Expectation-Maximization (EM) algorithm. Tatiraju *et al.* [15] used a variant of the K-means algorithm such that each pixel intensity is considered as an individual observation and the authors partition these observations into k clusters. This method does not consider the intensity distribution of its clusters. As a result the segmentation obtained lacks the connectivity within the neighborhood pixels. The mixture of Gaussians segmentation proposed by Farnoosh and Zarpak [16] depends heavily on the intensity distribution models to group the image data. The underlying assumption of their approach is that intensity distribution of the image can be represented by multiple Gaussians. However, it does not take into account the neighborhood information. As a result, the segmented regions lack connectivity with the pixels within their neighborhood.

DeephESC 2.0 detects the hESC regions using the approach proposed by Guan *et al.* [36]. The algorithm uses the intensity distributions of the foreground (hESC) and background (substrate) as well as the cell property for detection. The intensity distributions of the foreground and background are modeled as a mixture of two Gaussians and the cell property is translated into a local spatial information. The algorithm is optimized by parameters of the distributions and the cell regions evolve with the local cell property. The advantage of this approach is that, it not only uses information of the foreground and background, but it also uses cell properties resulting in fine-grained localization of the hESC even in the presence of background noise.

2.1.2 Classification of hESC Images

Lowry *et al.* [17] designed a texture based multi-stage Bayesian level set algorithm to segment pluripotent and trophectoderm colony images of hESC and their derivatives. The authors used an MR8 approach [18] for modeling the texture by convolving image patches with a filter bank containing Gaussian and Laplacian of Gaussian (LoG) filters at a fixed scale and edge bar filters at three different scales and several orientations. This results in a texton feature vector containing eight filter responses for every given pixel in the image. After extracting these texton features, the texturally inhomogeneous images are segmented using a multi stage Bayesian Level Set (BLS). The advantage of using BLS is that it produces smoother segmentation maps with regular borders and is much more tolerant to poor initial conditions.

Lowry *et al.* [19] combined set levels, multi resolution wavelet analysis and non-parametric estimation of the density functions of the wavelet coefficients to segment and classify stem cell nuclei. The authors also used an adjustable length window to deal with small size textures where the largest inscribed rectangular window may not contain a sufficient number of pixels for multiresolution analysis of elongated and irregularly shaped nuclei. Mangoubi *et al.* [20] classified hESC into differentiated and pluripotent cell colonies using a wavelet based texture decomposition. The authors used four visual features namely: *textural homegeneity*, *textural tightness*, *border sharpness* and *border circularity*. Based on these visual features, the authors achieved an accuracy of 96% in classifying colonies that were very distinct from each other and 86% in colonies with a mixed distribution. The authors suggest that a good pluripotent stem cell colony must exhibit a homogeneous, tight texture throughout, thus allowing a statistical analysis of the coefficients obtained from a wavelet based texture decomposition to discriminate between the colonies.

Desai *et al.* [21] classified fluorescent stem cell nucleus images into pluripotent and differentiated nucleus. The authors assume that the nucleus exhibits an onion layer texture where we may assume that within a layer the behavior is homogeneous, but may vary from layer to layer. The authors use a matrix edge function that adaptively modulates the shape, size, and orientation of neighborhoods over different regions of the texture, thus providing directional information on the texture that is not available in the more conventional scalar edge field based approaches.

Sammak *et al.* [22] classified differentiating cells into three classes namely: *Trophectoderm, Neurectoderm, and Progeny cells.* The authors showed that during differentiation the edges at the borders of the cell become more thin. The authors extract features, using wavelet decomposition and a matrix edge function, which are then given to a Support Vector Machine (SVM) for classification.

Niioka *et al.* [23] detect the cellular differentiation of myoblasts to myotubes using Convolutional Neural Networks. During the differentiation process, the cellular morphology changes from a round shape to an elongated tubular shape due to the fusion of cells. The authors trained their CNN using stained fluorescent images as input and were able to detect the differentiation with an accuracy of 91.3%.

Chang *et al.* [24] were able to classify human Induced Pluripotent Stem (iPS) cells in human cord blood CD34+ images using Convolutional Neural Networks. The authors used a 5 convolutional layer network to classify 256x256 patches of images with an accuracy of 91.8%. Xie *et al.* [25] performed cell counting in fluorescent images using a convolutional regression network. They trained a network to localize fluorescent labeled cell nuclei via down-convolutional feature extraction and symmetrically up-convolutional pixel-wise classification. They apply their network to a variety of datasets and manually annotated grayscale histology sections with an average error of 2.9% for their cell counting task. While their method is a successful implementation for training a neural network feature classifier for localizing cells, it is relatively easy to localize cells in a fluorescent dataset compared to more non-invasive complex datasets, such as the data with low contrast and high texture. A drawback of the works done by [23–25] is that they had to stain hESC in order to classify them making it an invasive approach, whereas, our approach is totally non-invasive.

Witmer *et al.* [26] developed an automated system to localize six cell colonies namely: *Debris*, *Dense*, *Spread*, *Differentiated*, *Partially spread*, and *Partially differentiated*. The authors extracted patches of size 224×224 using a sliding window from phase contrast images. These patches are then passed through an entropy filter that segments the cell colonies by exploiting the difference between the background and foreground of the images. The segmented patches are then passed through a CNN which classifies the patch into one of the 6 classes. The authors were able to achieve an accuracy of 89.35% in classifying the cell colonies, but a drawback of their approach is that they used a fixed window size of size 224×224 for localizing the cell colonies. This leads to smaller sized colonies to be overlooked leading to an incorrect segmentation.

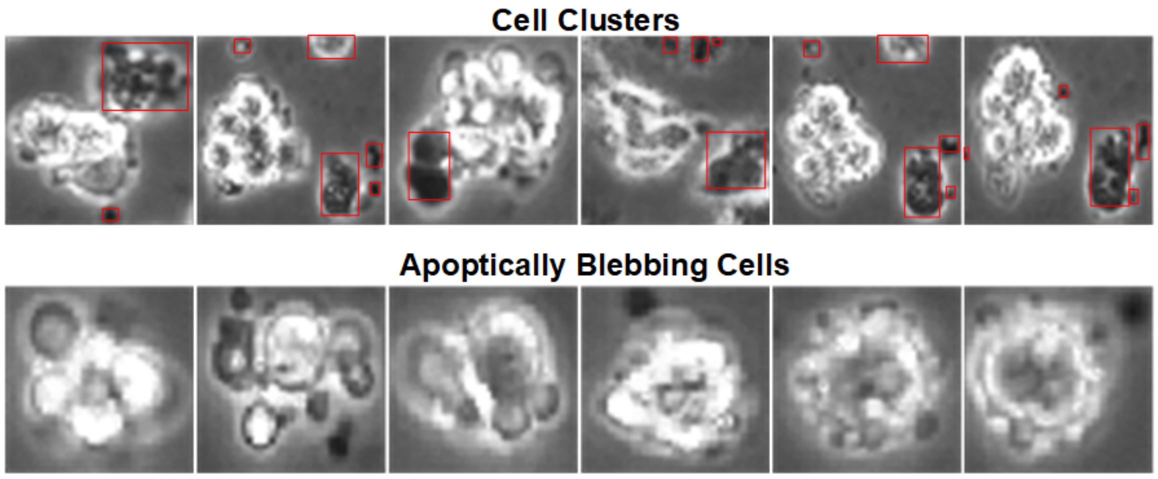

Fig. 3: Example images of *Cell clusters* and *Apoptically Blebbing cells*. The distinguishing features between *Cell clusters* and *Apoptically Blebbing cells* are the small cells in the *Cell clusters* packed close to each other.

In our previous work using DeephESC [5], we used a CNN and Triplet CNNs to classify hESC images into six different classes. DeephESC was able to classify hESC images with an accuracy of 91.71%, but a problem encountered in this approach is that images belonging to the class *Cell clusters* were misclassifed as *Apoptically Blebbing cells* with an error rate of 7.89% which was the highest error percentage between any two classes. The reason for this is that *Cell clusters* and *Apoptically Blebbing cells* have a very similar texture and intensity. Fig. 3 shows example images of *Cell clusters* and *Apoptically Blebbing cells*.

The main distinguishing factor between these two classes is the presence of smaller cells packed close to the *Cell clusters*.

In Fig. 3, the small cells in the manually annotated red bounding box are the factors that distinguish between a *Cell cluster* and an *Apoptically Blebbing cell*. Since these cells are very small, and as the image is passed forward through the convolution layers of CNN the dimensions of the feature maps progressively decrease and hence the receptive fields of the convolution filters are not able to detect these small cell bodies. To solve this DeephESC 2.0 skips connections between the initial and final convolution layers. The initial convolution layers learn a more coarse representation of the image where the receptive field of the filters are able to detect the small surrounding cells, whereas, the final layers learn a more fine-grained representation. By skipping intermediate layers and concatenating the feature maps of the initial and final convolution layers, DeephESC 2.0 is able to extract much more robust features that can detect these small surrounding cells which helps to improve the classification between these two classes.

2.1.3 Generation of Synthetic hESC Images

To the best of our knowledge, there is no published work that synthetically generates hESC images prior to our work in DeephESC [5] and this chapter [27]. In DeephESC we evaluated two different approaches for generating synthetic hESC images namely: Deep Convolution Generative Adversarial Networks (DCGAN) [42] and ensemble - Deep Convolution Generative Adversarial Networks (e-DCGAN) [5].

Generative adversarial nets were recently introduced as a novel way to train a network to generate synthetic images. They consists of two 'adversarial' models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data (real images) rather than the generator G (synthetic images). In order to learn the distribution $P_g(x)$ over data x, the generator builds a mapping function from a prior noise distribution $P_z(z)$ to data space as $G(z;\ \theta_g)$. The discriminator $D(x;\ \theta_d)$ outputs a single scalar representing the probability that x came from the training data rather than $P_g(x)$.

In this chapter, DeephESC 2.0 [27] uses a variant of the DCGAN architecture named Generative Multi Adversarial Network (GMAN) [37]. GMAN is different from DCGAN by the fact that instead of using a single discriminator, we use N multiple discriminators to train the generator. In practice, training against a single discriminator can impede the generator's learning. This is because if the generator is unlikely to generate any sample considered "realistic" by the discriminator's standards, the generator will receive negative feedback. This is problematic because the information contained in the gradient derived from negative feedback only dictates where to drive down $P_g(x)$, not specifically where to increase $P_g(x)$. Furthermore, driving down $P_g(x)$ necessarily increases $P_g(x)$ in other regions of X (to maintain $\int_X P_g(x) = 1$) which may or may not contain samples from the true dataset (whack-a-mole dilemma). In contrast, a generator is more likely to see positive feedback against an ensemble of discriminators (because the generator needs to fool only 1 of the N discriminators), which may better guide a generator towards amassing $P_g(x)$ in approximately correct regions of X.

2.1.4 Contributions of this Chapter

- An improved hierarchical classifier to classify hESC phase contrast image into six different classes with an accuracy of 93.23%
- Generating high quality synthetic hESC images using an ensemble of GMANs
- Exhaustive validation of the quality of the generated synthetic images using the Structural SIMilarity (SSIM) index, Peak Signal to Noise Ratio (PSNR) and statistical p-value tests.

- Training DeephESC 2.0 exclusively on a large amount of synthetic hESC images helps improve the classification accuracy of the classifier on the original hESC images from 93.23% to 94.46%.
- Comparison and visualization of the features learned using DeephESC [5] and DeephESC 2.0.

2.2 Data and Technical Approach

2.2.1 Data

The hESC were cultured in vitro using methods described in detail previously [35]. The videos were acquired using the Nikon BioStation IM with a 20x objective resulting in a resolution of 600x800. A dataset of 784 cropped images was obtained from nine hESC videos. The dataset had the following numbers of images for each class: 1) 122 *Cell Cluster* images; 2) 113 *Debris* images; 3) 135 *Unattached cell* images; 4) 132 *Attached cell* images; 5) 104 *Dynamically Blebbing cell* images; and 6) 178 *Apoptotically Blebbing cell* images. The ground-truth for the dataset was annotated manually by expert stem cell biologists. The annotation was done by observing the morphology of the cells in the image as well as how they change in the video.

2.2.2 Technical Approach for DeephESC 2.0

DeephESC 2.0 is designed in a modular manner with three parts: hESC detection, hESC classification and hESC generation. Fig. 4 shows the workflow of DeephESC 2.0. The source code was written and developed in PyTorch. The source code and supplied test data are available online at http://vislab.ucr.edu/SOFTWARE/software.php. To successfully run the source code requires the following softwares/libraries: python 3.5.2, pytorch 0.3.1, torchvision, PIL, numpy.

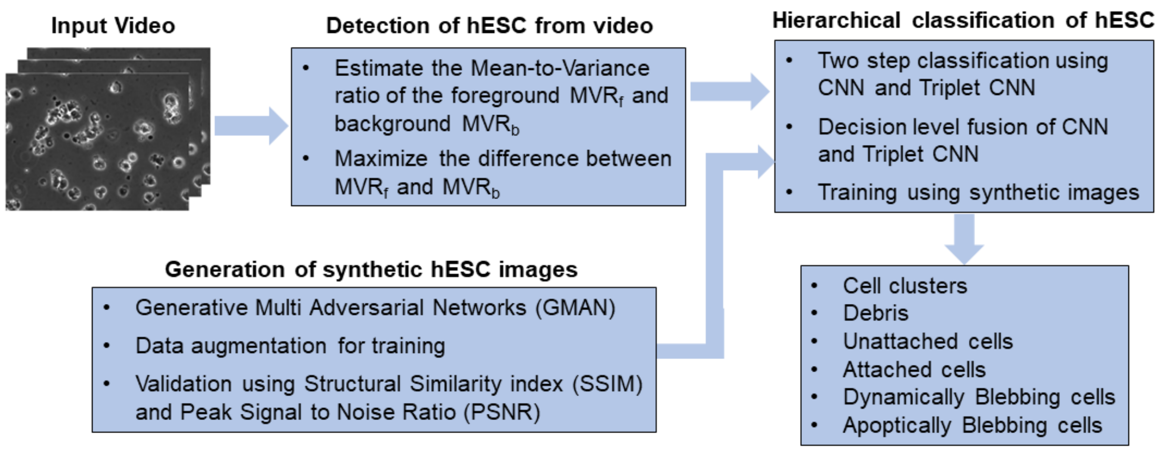
Input Video
Detection of hESC from video
Estimate the Mean-to-Variance ratio of the foreground MVR$_f$ and background MVR$_b$
Maximize the difference between MVR$_f$ and MVR$_b$
Hierarchical classification of hESC
Two step classification using CNN and Triplet CNN
Decision level fusion of CNN and Triplet CNN
Training using synthetic images
Generation of synthetic hESC images
Generative Multi Adversarial Networks (GMAN)
Data augmentation for training
Validation using Structural Similarity index (SSIM) and Peak Signal to Noise Ratio (PSNR)
Cell clusters
Debris
Unattached cells
Attached cells
Dynamically Blebbing cells
Apoptically Blebbing cells

Fig. 4: Workflow of DeephESC 2.0 is split into three modules namely: Detection of hESC from video, Generation of synthetic hESC images and hierarchical classification of the hESC images into six different classes.

(a) Detection of hESC from Videos using a mixture of Gaussians: We detected and cropped stem cells from video frames of size 600 x 800 using a method developed by Guan *et al.* [36]. In the following we provide a brief description of the method. The hESC are grown in culture dishes coated with a layer of substrate (Matrigel). The substrate becomes the background after the hESC are placed on its surface. Therefore, we model a hESC image with two regions of interest: foreground and background [36]. Fig. 5 shows examples of the cell (foreground) and the substrate (background) and their intensity distributions. Consequently we model the intensity distribution of foreground (cell region with a mean μ_f and variance ${\sigma_f}^2$) and background (substrate region with a mean μ_b and variance ${\sigma_b}^2$) as the mixture of two Gaussians.

With this model, we then want to maximize the absolute difference of mean-to-variance ratios of the foreground MVR_f and the mean-to-variance ratio of the background MVR_b; The MVRs of the foreground and background data sets are calculated by the following equations:

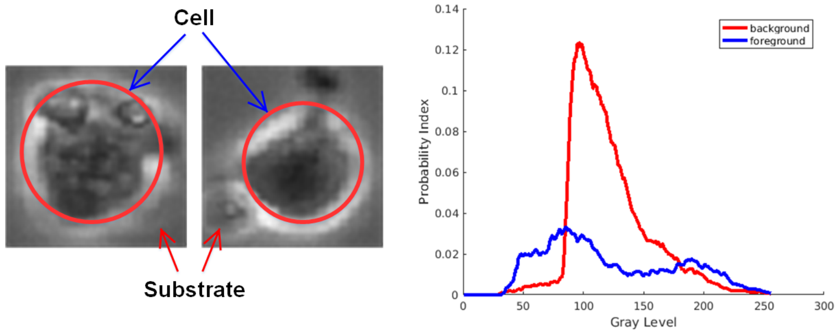

Fig. 5: Images of the cell body and the substrate and their corresponding intensity distribution.

$$MVR_f = \frac{\mu_f}{\sigma_f} \tag{1}$$

$$MVR_b = \frac{\mu_b}{\sigma_b} \tag{2}$$

where MVR_f and MVR_b are the MVRs for the foreground and background, respectively. Thus, the optimization metric M is formulated as:

$$M = |MVR_f - MVR_b| \tag{3}$$

substituting Eq (1) and Eq (2) into Eq (3), we get the following:

$$M = \left| \frac{\mu_f}{\sigma_f^2} - \frac{\mu_b}{\sigma_b^2} \right| \tag{4}$$

Eq (4) shows the metric that is used to determine how much the cell region data are different from the substrate region data. Since the algorithm is spatially evolving the foreground region from the initial high intensity variation region by a mean filter at each iteration, the foreground mean and variance are approaching to the background mean and variance. The limit of M is 0 as μ_f/σ_f^2 approaches to μ_b/σ_b^2. Therefore, our problem becomes finding M_{opt} which is the optimal value for metric M, given by:

$$M_{opt} = \max_{\mu_\mathrm{f}, {\sigma_\mathrm{f}}^2, \mu_\mathrm{b}, {\sigma_\mathrm{b}}^2} M(\mu_\mathrm{f}, {\sigma_\mathrm{f}}^2, \mu_\mathrm{b}, {\sigma_\mathrm{b}}^2) \tag{5}$$

M_{opt} finds the parameters that maximize the difference between the foreground and background pixels. Fig. 14 shows the detected components of a single frame. These detected components are then cropped and passed to the hierarchical classifier to be classified into one of the six aforementioned classes.

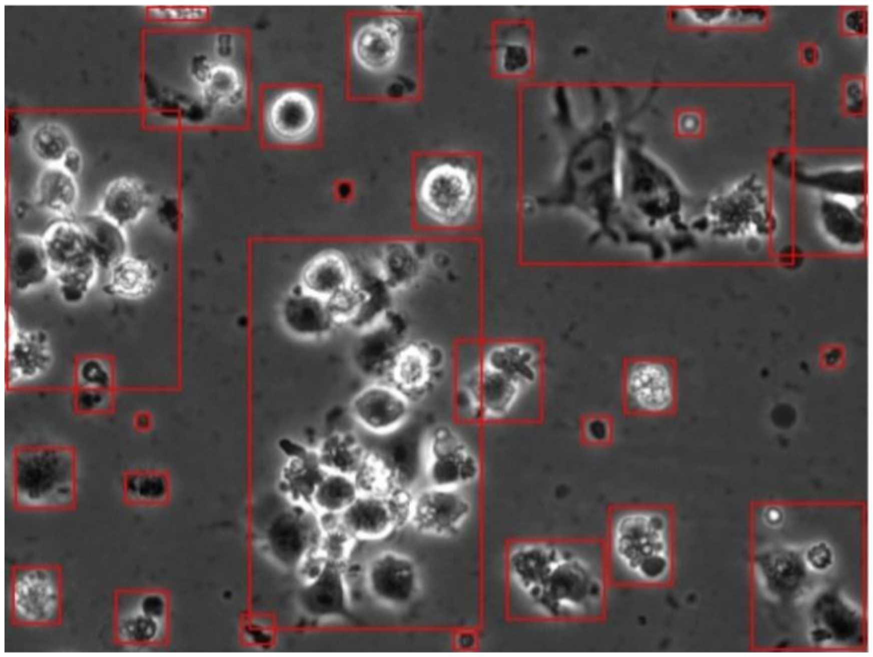

Fig. 6: Detected cell bodies of a single frame using the approach proposed by [36]. The detected cell bodies are then cropped and passed through the hierarchical classifier to be classified into one of the aforementioned six classes.

(b) Hierarchical Classification of hESC: In this section we explain in detail the architecture, training and parameters of the hierarchical classifier which includes the CNN and Triplet CNNs. Fig. 7 shows the work flow of the hierarchical classifier.

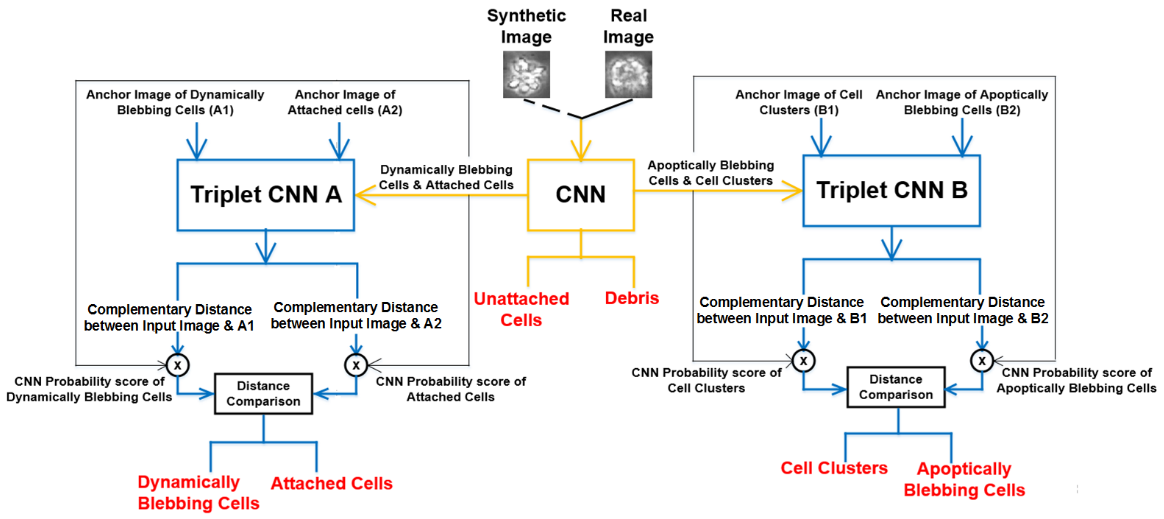
Synthetic Image
Real Image
Anchor Image of Dynamically Blebbing Cells (A1)
Anchor Image of Attached cells (A2)
Triplet CNN A
Dynamically Blebbing Cells & Attached Cells
CNN
Apoptically Blebbing Cells & Cell Clusters
Triplet CNN B
Anchor Image of Cell Clusters (B1)
Anchor Image of Apoptically Blebbing Cells (B2)
Complementary Distance between Input Image & A1
Complementary Distance between Input Image & A2
Unattached Cells
Debris
Complementary Distance between Input Image & B1
Complementary Distance between Input Image & B2
CNN Probability score of Dynamically Blebbing Cells
Distance Comparison
CNN Probability score of Attached Cells
CNN Probability score of Cell Clusters
Distance Comparison
CNN Probability score of Apoptically Blebbing Cells
Dynamically Blebbing Cells
Attached Cells
Cell Clusters
Apoptically Blebbing Cells

Fig. 7: Workflow of the hierarchical classifier. The input is either a real or synthetic image belonging to one of the six classes. The outputs of the CNN and Triplet CNNs are fused at the decision level using the product rule.

(c) Convolution Neural Networks: After detecting and cropping all the cell regions in a video, we resize all the hESC images to size 64x64. These images are then used for training the CNN. Table 1. shows the architecture details of our CNN. To train the CNN, we chose a mini batch size of 64. Since the size of our dataset is very limited, in order to prevent the CNN from over-fitting, we perform random affine transformations to the images and employ early stopping. Table. 2 shows the data augmentation performed for training the CNN. We perform early stopping by saving the model after every epoch, only if the validation accuracy increases compared to the previous epoch. If the validation accuracy has not increased after 3 consecutive epochs we stop the training.

We randomly chose 10 images from each class (60 images in total) as the validation dataset. The remaining of the dataset excluding the validation images, was divided into 5 folds for cross-validation. We did random hyper-parameter search for the CNN to obtain the best learning rate, momentum and weight decay. We chose random values for the

TABLE 1: Architecture of the Convolution Neural Network in the hierarchical classifier.

Input Dimension	**Output Dimension**	**Number of feature maps**	**Layer (Kernel dimension, stride, padding)**
64x64	32x32	64	Convolution (7, 2, 3)
32x32	16x16	64	Maxpool (3, 2, 1)
16x16	8x8	128	Convolution (5, 2, 2)
8x8	4x4	128	Maxpool (3, 2, 1)
2,048x1	6 classes	-	Fully connected layer

TABLE 2: Data augmentation performed to train the CNN.

Affine Transformation	**Parameters**
Image rotation	$-180°$ to $180°$
Image shearing	$0°$ to $30°$
Image zooming	70% to 140% of image size

learning rate, momentum and weight decay within a given range and step size and trained the network for three epochs. The combination of hyper-parameters that gave us the highest classification accuracy after three epochs are chosen as the best hyper-parameters for the network. The random hyper-parameter search was done by evaluating the CNN only on the validation dataset. Based on this we chose the best hyper-parameters as learning rate $= 1.2\text{x}10^{-2}$, momentum $= 0.9$ and weight decay $= 1\text{x}10^{-3}$ The network was optimized using the stochastic gradient descent algorithm with cross entropy loss.

We performed 5-fold cross validation and the results are shown in detail in Section 3.3. After evaluating the CNN we observed that the CNN was able to classify the classes *Debris* and *Unattached Cells* with high accuracy, but the classes *Cell clusters/Apoptically Blebbing cells* and *Dynamically Blebbing Cells/Attached Cells* were misclassified the most. The reason for this is that, the classes *Cell clusters/Apoptically Blebbing Cells* and *Dynamically Blebbing Cells/Attached Cells* have similar intensity and texture.

(d) Triplet Convolution Neural Network: To solve this misclassification, we train a Triplet CNN to perform fine-grained classification between *Cell clusters* and *Apoptically Blebbing Cells* and similarly, for *Dynamically Blebbing Cells* and *Attached Cells*. Fig. 8 shows the visual representation of the architecture for Triplet CNN A and Triplet CNN B from Fig. 7.

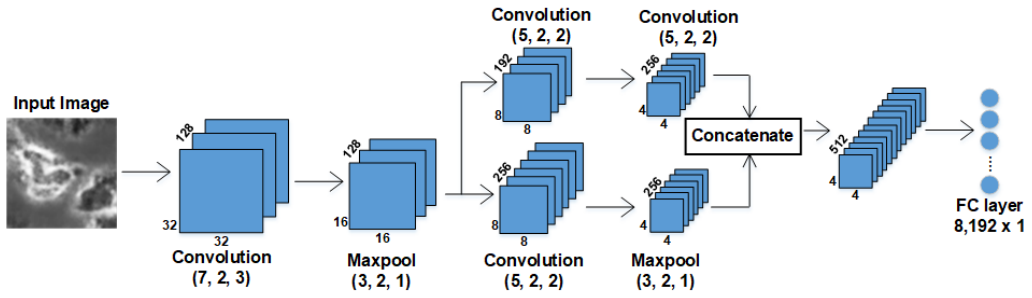

Fig. 8: Architecture of Triplet CNN A and Triplet CNN B in Fig. 7. The parameters within the parenthesis indicate the kernel dimension, stride and padding. By skipping intermediate layers and concatenating the feature maps of branched layers, DeephESC 2.0 is able to extract much more robust features, further improving the classification.

The Triplet CNN architecture in Fig. 8 is different from DeephESC by the fact that, DeephESC does not have any concatenation of feature maps between intermediate layers. By doing so, the initial convolution layers learn more coarse features while the final convolution layers are able to learn more fine-grained features. Concatenating the two branches together helps extract robust features and improves the classification accuracy compared to DeephESC [5]. Fig. 11 in Section 3.2.2b shows the visual comparison of features extracted between DeephESC and DeephESC 2.0 and it can be observed that DeephESC fails to extract robust features for a given image compared to DeephESC 2.0.

The Triplet CNN takes as input a query image and one anchor image from each class. The output of the Triplet CNN is the two pairwise distances between the extracted features for the query image and the two anchor images as shown in Fig. 7. For a correct classification, the pairwise distance between the query image and the anchor image belonging to the same class must be smaller (close to 0) compared to the distance between the query image and the anchor image belonging to the opposite class.

We used the same 10 validation images from each class used for validating the CNN, to validate the Triplet CNN. We randomly selected 5,000 triplet pairs for validation and 100,000 triplet pairs to train both Triplet CNN A and Triplet CNN B using 5-fold cross validation similar to how we trained the CNN. We chose a mini-batch size of 256 triplets and performed random hyper-parameter search and random affine transformation to the images as shown in Table 2 that was similarly done while training the CNN. Table 3. shows a summary for the best hyper-parameters for the CNN, Triplet CNN A and Triplet CNN B.

TABLE 3: Best hyper-parameters for training the networks in DeephESC 2.0.

Network	**Learning rate**	**Momentum**	**Weight decay**
CNN	1.2x10^{-2}	0.9	1x10^{-3}
Triplet CNN A	1.2x10^{-2}	0.8	1x10^{-3}
Triplet CNN B	2x10^{-2}	0.8	1x10^{-3}

The Triplet CNNs were optimized using the Stochastic Gradient Descent algorithm with the Ranked Marginal loss function given by Eq (6). In Eq (6), X_1 and X_2 are the two anchor images and $G(X)$ is the pairwise distance between the feature extracted by Triplet CNN for the query image and the anchor image. In Eq (6) if $Y = 1$ it indicates that the anchor image x_1 belongs to the same class as the query image, whereas, $Y = -1$ indicates that the anchor image x_2 belongs to the same class as the query image. For all of our experiments we set the value of the margin as 1.

$$Loss = Max(0, -Y * (G(X_1) - G(X_2)) + margin) \tag{6}$$

Upon evaluating the Triplet CNNs with 5-folds cross validation, Triplet CNN

A achieved an average classification accuracy of 95.24% and Triplet CNN B achieved an average classification accuracy of 95.83%.

(e) Decision level fusion of the CNN and Triplet CNNs: After training the CNN and the individual Triplet CNNs we combine them in a hierarchical system as shown in Fig. 7. The input hESC image is first passed into the CNN, the CNN is trained to classify the input image into one of the aforementioned six classes. If the predicted class is *Debris* or *Unattached cells*, we take the prediction of the CNN as the final prediction.

If the predicted class is *Attached cell* or *Dynamically Blebbing cells*, the input image is passed to Triplet CNN A, and we obtain the prediction of Triplet CNN A. Similarly, if the prediction of the CNN is *Cell cluster* or *Apoptically Blebbing cells*, the input image is passed to the Triplet CNN B and we obtain the prediction of Triplet CNN B.

The decision level fusion was done by taking the complementary pairwise distance (i.e. 1 - pairwise distance) measure outputs from the Triplet CNN and multiplying the corresponding probability score for that class from the CNN. For example in Fig. 7, in Triplet CNN B, the complementary pairwise distance measure between the input image and anchor image of *Cell clusters* is multiplied with the probability score for *Cell clusters* from the CNN. Similarly, the complementary pairwise distance measure between the input image and anchor image of *Apoptically Blebbing cells* is multiplied with the probability score for *Apoptically Blebbing cells* from the CNN, and so on for Triplet CNN A. The results obtained with and without the fusion are explained in detail in Section 3.2.2a.

(f) Generating Synthetic hESC Images using Generative Multi Adversarial Networks: The purpose of this section is to generate synthetic data and add more variability to the training dataset to help improve the classification performance of DeephESC 2.0. To achieve this we trained an ensemble of Generative Multi Adversarial Networks (GMAN) [37]

GMAN consists of a generator network G and N discriminator networks (D_1, D_2, ..., D_N). The generator takes a random noise vector z as input and returns an image $X_{gen} = G(z)$. On the other hand, the discriminator takes a real or a generated image, and outputs a probability distribution $P(S|X) = D(X)$ over the two image sources S. The discriminator is trained to maximize the log-likelihood of assigning the correct source while G tries to minimize it:

$$\begin{aligned}\min_G \max_D V(D,G) = \quad & \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \\ & + \mathbb{E}_{x\sim p_z(z)}\left[\log\left(1 - D(G(z))\right)\right]\end{aligned} \tag{7}$$

In GMAN since we have multiple discriminators, we combine the outputs of the N discriminators using the weighted geometric mean as shown in Eq (8).

$$GM(V,\lambda) = -exp(\sum_i^N w_i log(-V_i)) \tag{8}$$

where, $w_i = e^{\lambda V_i}/\sum_j e^{\lambda V_j}$, V_i is the output of the ith discriminator and λ is a constant such that $\lambda \geq 0$. The objective is that the generator network and the ensemble of discriminators converge to the Nash equilibrium so that $D_1, D_2, ..., D_N$ are maximally confused and G generates samples that resemble the training data. In our approach we trained six individual GMANs to generate images belonging to the corresponding six classes. Fig. 9 shows the architecture of a GMAN. In DeephESC 2.0, we chose to use three different discriminators in our GMAN architecture. Table 4. shows the architecture of the generator and the three discriminators.

We chose the learning rate for the generator to be 1x10^{-4} and learning rate of the three discriminators to be 1x10^{-5} and mini batch of size 32. All the networks were optimized using the Adam algorithm [38] with loss function as a combination of Binary Cross Entropy

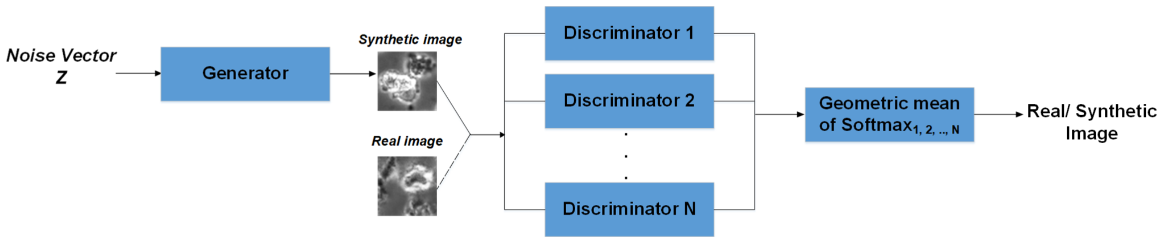

Fig. 9: Architecture of GMAN. The generator is trained to take as input a random noise vector and generate an image that resembles the training data. The task of the N discriminators are to predict if the input image to the discriminator is either a real or a synthetic image. In our architecture of GMAN the softmax outputs of the N discriminators are combined together by computing their geometric mean.

and Embedding loss as shown in equation (9).

$$Loss = \frac{-1}{n}\sum_{i=1}^{n} y_i * log(p_i) + (1-y_i) * log(1-p_i) + \alpha * \frac{1}{n}\sum_{i=1}^{n} ||X_i - X||^2 \tag{9}$$

In Eq. 9, the first term is the Binary Cross Entropy loss. y_i is the ground-truth label (real or synthetic image), p_i is the probability score being a real image. The second term is the Embedding loss, X_i is an image from the mini batch (either synthetic or real image) and X is a real image chosen randomly from the training dataset belonging to the same class as X_i. The Binary Cross Entropy loss ensures that the GMAN is able to extract accurate features to generate synthetic images resembling the images from the training dataset and the Embedding loss ensures that the generated images have a similar morphology as the images from the training dataset. α is an empirical value and was chosen to be 5x10^{-2}.

2.3 Experimental Results

2.3.1 Detection of hESC from Video

We evaluated the detection of hESC objects using the algorithm proposed by Guan *et al.* [36]. The metrics used for evaluating the detection are Jaccard similarity, Dice coefficient, Specificity and Sensitivity. The Sensitivity (SEN), measures the proportion of

TABLE 4: Architecture of the generator and the three discriminators used in our Generative Multi Adversarial Network.

Network	Input dimension	Output dimension	Number of feature maps	Layer (Kernel dimension, stride, padding)
Generator	100x1	8,192x1	-	Fully connected layer
	4x4	8x8	256	ConvolutionT* (6, 2, 2)
	8x8	16x16	128	ConvolutionT* (6, 2 , 2)
	16x16	32x32	64	ConvolutionT* (6, 2, 2)
	32x32	64x64	1	ConvolutionT* (6, 2, 2)
Discriminator 1	64x64	32x32	32	Convolution (5, 2, 2)
	32x32	16x16	64	Convolution (5, 2, 2)
	16x16	8x8	128	Convolution (5, 2, 2)
	8x8	4x4	256	Convolution (5, 2, 2)
	4,096x1	1	-	Fully connected layer
Discriminator 2	64x64	32x32	16	Convolution (5, 2, 2)
	32x32	16x16	32	Convolution (5, 2, 2)
	16x16	8x8	64	Convolution (5, 2, 2)
	8x8	4x4	128	Convolution(5, 2, 2)
	2,048x1	1	-	Fully connected layer
Discriminator 3	64x64	32x32	32	Convolution (5, 2, 2)
	32x32	16x16	64	Convolution (5, 2, 2)
	16x16	8x8	128	Convolution (5, 2, 2)
	8x8	4x4	256	Convolution (5, 2, 2)
	4x4	2x2	512	Convolution (5, 2, 2)
	2,048x1	1	-	Fully connected layer

* Note: ConvolutionT stands for the Convolution Transpose operation.

actual positives which are correctly detected:

$$SEN = \frac{TP}{(TP + FN)} \tag{10}$$

The Specificity (SPC), is the true negative rate which is given by:

$$SPC = \frac{TN}{(FP + TN)} \tag{11}$$

The Jaccard similarity (J), is a measure of similarity between the detected results and the ground-truth:

$$J = \frac{TP}{(TP + FP + FN)} \tag{12}$$

The Dice coefficient (DIC), measures the agreement between the detected results and the ground-truth:

$$DIC = \frac{2TP}{(2TP + FP + FN)} \tag{13}$$

The approach achieved a Jaccard similarity (J) of **0.754**, Dice coefficient (DIC) of **0.860**, Sensitivity (SEN) of **0.906** and Specificity (SPC) of **0.924**.

2.3.2 Measures for Classification Performance

We trained and evaluated the classifier using the K- fold cross validation. K-fold cross validation divides the dataset into K subsets. Each time, one of the K subsets is used as the testing set and the remaining K - 1 subsets are put together to form a training set. Then the average error across all K trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in the testing set exactly once, and gets to be in a training set K - 1 times. The variance of the resulting estimate is reduced as K is increased. In the following we evaluated the classification accuracy using the 5- fold cross validation ($K = 5$).

(a) Classification Results: Table 5 shows the average classification accuracy for the 5-fold cross validation using CNN, CNN-Triplet and Fused CNN-Triplet approach of Deep-

TABLE 5: Comparison of the average classification accuracy of the networks used in DeephESC and DeephESC 2.0.

Approach	Network	Average Classification Accuracy
ResNet18 [39]	CNN	70.44%
VGG19 [40]	CNN	72.57%
AlexNet [28]	CNN	71.91%
DeephESC [5]	CNN	86.14%
	CNN-Triplet	89.37%
	Fused CNN-Triplet	91.71%
DeephESC 2.0	CNN	**86.33%** $\pm$ 0.29
	CNN-Triplet	**90.88%** $\pm$ 0.26
	Fused CNN-Triplet	**93.23%** $\pm$ 0.24

TABLE 6: Confusion matrix for the classification of the 724 real hESC images using the CNN architecture of DeephESC 2.0.

Class	CC	DEB	UN	AT	DYN	APO
CC	**97**	3	0	0	1	11
DEB	0	**100**	1	1	1	0
UN	2	0	**121**	1	0	1
AT	1	2	0	**100**	16	3
DYN	2	0	1	10	**81**	0
APO	30	4	2	1	5	**126**

TABLE 7: Confusion matrix for the classification of the 724 real hESC images using the CNN-Triplet architecture of DeephESC 2.0.

Class	CC	DEB	UN	AT	DYN	APO
CC	**102**	3	0	0	1	6
DEB	0	**100**	1	1	1	0
UN	2	0	**121**	1	0	1
AT	1	2	0	**105**	11	3
DYN	2	0	1	4	**87**	0
APO	13	4	2	1	5	**143**

TABLE 8: Confusion matrix for the classification of the 724 real hESC images using the Fused CNN-Triplet architecture of DeephESC 2.0.

Class	**CC**	**DEB**	**UN**	**AT**	**DYN**	**APO**
CC	**105**	3	0	0	1	3
DEB	0	**100**	1	1	1	0
UN	2	0	**121**	1	0	1
AT	1	2	0	**110**	6	3
DYN	2	0	1	2	**89**	0
APO	6	4	2	1	5	**150**

The Abbreviations used in Table 6, 7 and 8 are as follows: **CC**: *Cell clusters*, **DEB**: *Debris*, **UN**: *Unattached cells*, **AT**: *Attached cells*, **DYN**: *Dynamically Blebbing cells*, **APO**: *Apoptically Blebbing cells.*

hESC 2.0 and Table 6, 7 and 8 show the confusion matrices for the CNN, CNN-Triplet and fused CNN-Triplet, respectively.

All the networks in Table 5 were trained and evaluated on the real hESC images. We compare the results obtained using DeephESC 2.0 with the results obtained using DeephESC. The dataset has a total of 784 real hESC images, 10 randomly chosen images from each class (60 in total) were used as the validation dataset. In order to maintain fairness in evaluation, these 60 validation images were not used for evaluating the performance of the networks. The remaining 724 hESC images are split into 5 folds for cross validation. Note that the results shown in Table 5 - Table 8 are for the 724 images used in the 5 fold cross validation.

Comparing Table 6 and Table 7 it can be observed that, the misclassification between the classes ***Cell clusters*** **(CC)** and ***Apoptically Blebbing cells*** **(APO)** has been reduced from 14.64% to 6.79% using the CNN-Triplet compared to just the CNN. Similarly, the misclassification of ***Attached cells*** **(AT)** and ***Dynamically Blebbing cells*** **(DYN)** has been reduced from 12.04% to 6.94%. Moreover, upon fusing the outputs of the CNN and the Triplet CNN we further reduced the misclassification of ***Cell clusters*** **(CC)** and ***Apoptically Blebbing cells*** **(APO)** to **3.21%**.

(b) Comparison of Features learned by DeephESC 2.0 and DeephESC: Fig. 10(a) and (b) shows the features extracted by the CNN used in DeephESC 2.0. for an *Apoptically Blebbing cell* and *Unattached cell* respectively. In Fig. 10, the first convolutional layer learns filters some of which look like edge detectors, filters for image blurring and image sharpening. These features become more sparse and localized as the data flows further through the layers of the CNN.

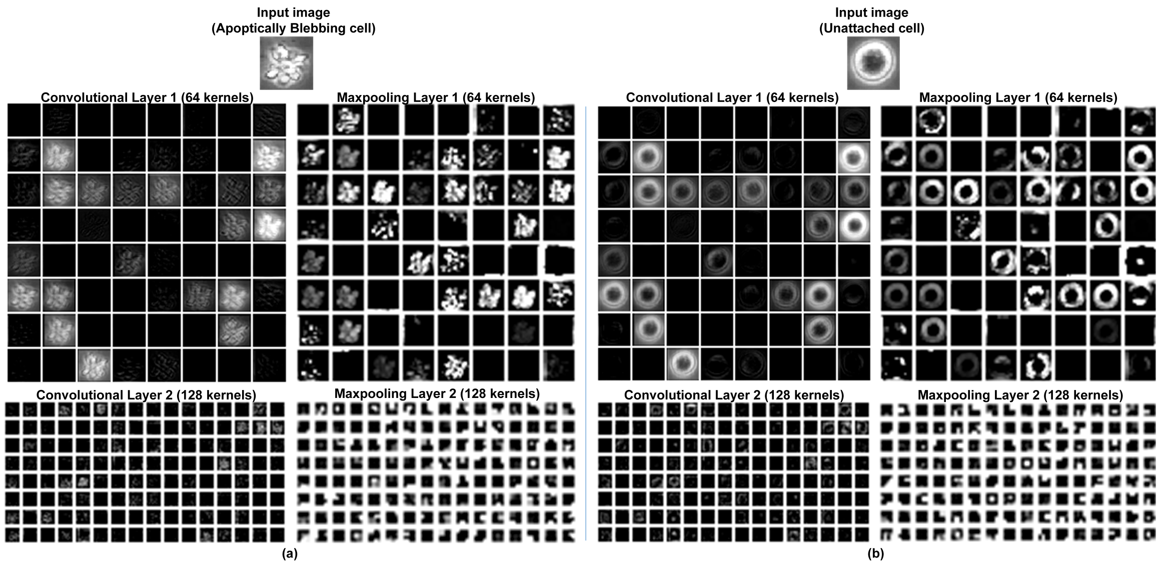
Input image
(Apoptically Blebbing cell)
Convolutional Layer 1 (64 kernels)
Maxpooling Layer 1 (64 kernels)
Convolutional Layer 2 (128 kernels)
Maxpooling Layer 2 (128 kernels)
(a)
Input image
(Unattached cell)
Convolutional Layer 1 (64 kernels)
Maxpooling Layer 1 (64 kernels)
Convolutional Layer 2 (128 kernels)
Maxpooling Layer 2 (128 kernels)
(b)

Fig. 10: Visualization of features extracted by the CNN in DeephESC 2.0 for (a) *Apoptically Blebbing cell* and (b) *Unattached cell*.

In order to compare the improvement in classification between DeephESC and DeephESC 2.0, we visualized the features learned by DeephESC 2.0. Fig. 11(a) shows an image of a *Cell cluster* (CC) that was correctly classified by DeephESC 2.0, but was incorrectly classified as *Apoptically Blebbing cell* (APO) by DeephESC.

We masked the area containing the surrounding small cells in Fig. 11(a) with a sliding window of size 5 x 5 with gray scale pixel value of 85 (pixel range is from 0 to 255) that matches the surrounding background as shown in Fig. 11(b). For visualization

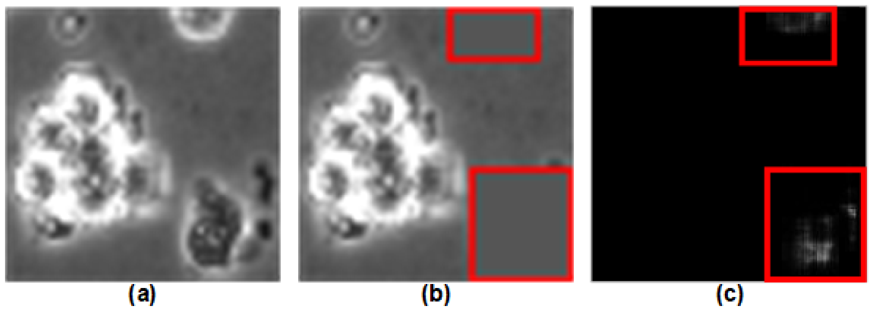

Fig. 11: Visualization of features learned by DeephESC 2.0. (a) Image of a *Cell cluster*. (b) Image after masking the surrounding small cells using a window. Red bounding boxes are drawn across the masked area only for visualization purposes. (c) Probability heat map for the class *Apoptically Blebbing cell*.

purposes we draw a red bounding box across the masked area in Fig. 11(b). The image in Fig. 11(b) is then passed through the hierarchical classifier for each position of the sliding window and the output probability score of the class *Apoptically Blebbing cell* (APO) for that center position of the sliding window is plotted in Fig. 11(c).

The inference that we get from Fig. 11(c) is that, the bright pixel locations indicate the locations that the classifier predicts as important features for the image being a *Cell cluster*. The reason for this is that, the 5 x 5 mask window centered around that area is masking the small cells as seen in Fig. 11(b), and since the network is unable to see these surrounding small cells, it predicts the image to be an *Apoptically Blebbing cell*. Hence, this means that the small cells in the image are considered as important features for the network to classify the image as a *Cell Cluster*.

2.3.3 Synthetic hESC from GMAN

Fig. 12(a) and 12(b) shows examples for visualizing the features learned by the generators in DeephESC 2.0 for generating an *Unattached cell* and *Attached cell*, respectively. In Fig. 12, the input to the respective generators is a 100x1 dimensional randomly sampled Gaussian noise vector. We can observe that the FC layer and the first convolutional layer learn features that are very sparse and localized. As these features progress through the layers of the generator, the features become more smooth and gradually start to resemble a hESC both in texture and shape.

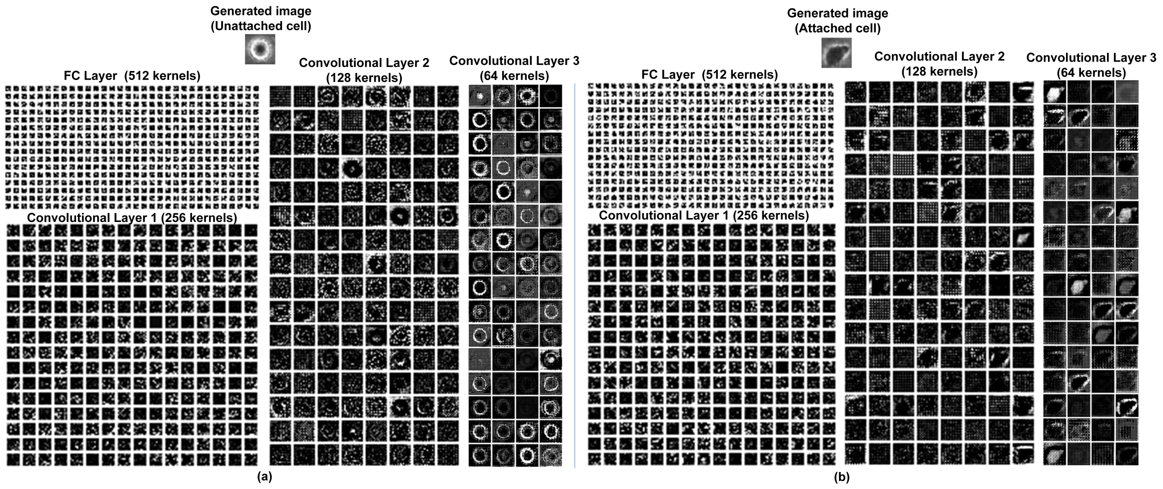
Generated image (Unattached cell)
FC Layer (512 kernels)
Convolutional Layer 1 (256 kernels)
Convolutional Layer 2 (128 kernels)
Convolutional Layer 3 (64 kernels)
(a)
Generated image (Attached cell)
FC Layer (512 kernels)
Convolutional Layer 1 (256 kernels)
Convolutional Layer 2 (128 kernels)
Convolutional Layer 3 (64 kernels)
(b)

Fig. 12: Visualization of features learned by the generators in DeephESC 2.0. (a) *Unattached cell* and (b) *Attached cell*.

(a) Evaluation of the Quality of the Generated Synthetic Images: In order to evaluate the quality of the synthetic images, we first generated 100 synthetic images for each of the six classes. Fig. 13 shows the 600 synthetic images that were generated for validating the quality. The average Structural Similarity (SSIM) score and average Peak-Signal-to-Noise Ratio (PSNR) score for a given synthetic image are computed by computing average the SSIM and PSNR between that given synthetic image and all the real images in the dataset for that given class. This is repeated for all the 100 synthetic images in each class and the average SSIM score and PSNR score is obtained. The structural similarity index between two images is calculated by:

$$SSIM(X,Y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{14}$$

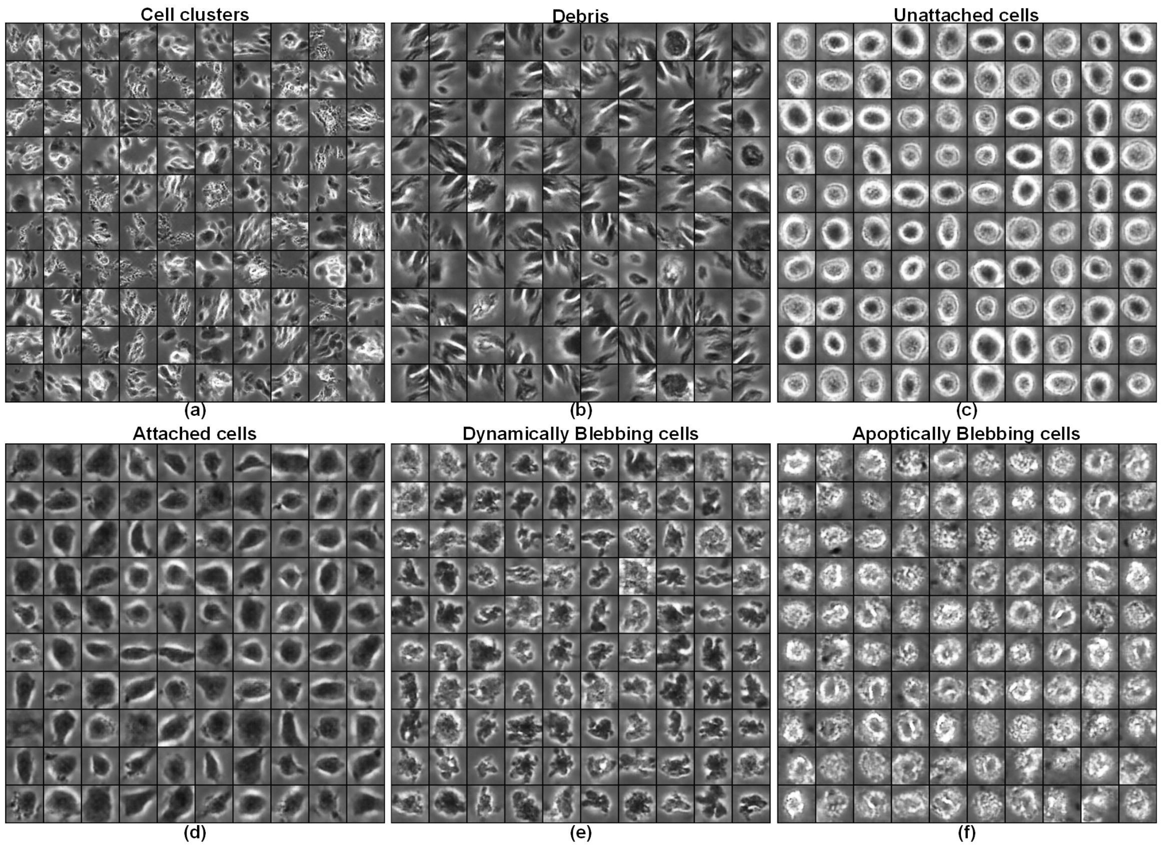

Fig. 13: The 600 synthetic images used for validating the quality in Table 9. (a) *Cell clusters*, (b) *Debris*, (c) *Unattached cells*, (d) *Attached cells*, (e) *Dynamically Blebbing cells*, (f) *Apoptically Blebbing cells*.

In Eq (14), μ_x and μ_y are the average pixel values of image X and Y respectively, σ_x^2 and σ_y^2 are the variance of the pixel values of image X and Y, respectively, σ_{xy} is the covariance between image X and Y. C_1 and C_2 are constants given by $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$, where, $L = 255$ is the maximum range of the pixel values and $K_1 = 0.01$ and $K_2 = 0.03$ are fixed constants. The PSNR between two images is calculated by:

$$PSNR(X,Y) = 10\,log_{10}\left(\frac{L^2}{MSE(X,Y)}\right) \tag{15}$$

In Eq (15), $L = 255$ is the maximum range of the pixel values, $MSE(X,Y)$ is computed by $MSE(X,Y) = \dfrac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[X(i,j) - Y(i,j)]^2$, m and n are the spatial dimensions of the synthetic image X and real image Y. Table 9 shows the average SSIM score and PSNR score obtained using the 100 synthetic images for each class shown in Fig. 13. The cells in Table 9 are formatted as x/y where x and y are the SSIM and PSNR values, respectively.

TABLE 9: Comparison of our GMAN architecture used in DeephESC 2.0 with e-DCGAN [5], DCGAN [42] and c-DCGAN [41] using the SSIM and PSNR metrics. SSIM has no units and PSNR is measured in decibels (dB)

Approach	**Cell Cluster**	**Debris**	**Unattached cell**	**Attached cell**	**Dynamically Blebbing cell**	**Apoptically Blebbing cell**
GMAN	**0.6312/ 19.71**	**0.6217/ 18.23**	**0.8347/ 26.27**	**0.6072/ 17.56**	**0.5921/16.25**	**0.5827/16.82**
e-DCGAN	0.6047/ 18.23	0.5931/ 15.77	0.7731/ 24.29	0.5730/ 16.28	0.5463/14.33	0.5498/14.24
DCGAN	0.5732/ 18.23	0.5931/ 15.77	0.7731/ 24.29	0.5730/ 16.28	0.5463/14.33	0.5498/14.24
c-DCGAN	0.5691/ 18.27	0.5722/ 15.28	0.7231/ 20.96	0.5897/ 15.29	0.5625/15.33	0.5411/15.09

The scale for SSIM is from 0 - 1 and has no unit, 0 indicates the images have no resemblance and 1 indicates they are the same images. The ideal range for SSIM score is between 0.5 - 0.85. The scale for PSNR is from 0 - ∞ and is measured in dB, 0 indicates the images have no similarity and ∞ indicates they are the same images. The ideal range for PSNR score is between 15dB - 30dB.

From Table 9, it can be observed that our GMAN architecture achieved the highest average SSIM and PSNR score for all the six classes. *Unattached cells* had the highest SSIM and PSNR score of 0.8347 and 26.27 dB, respectively as this class of hESC was the easiest to generate. The reason for this is that *Unattached cells* visually have the least complex structure compared to the other five classes. This is further supported by the observation that *Unattached cells* had a high correct classification accuracy of 96.80% because they are very easy to classify. It should also be noted that, the SSIM and PSNR for e-DCGAN and

DCGAN are the same except for the class *Cell clusters* because both of these approaches use the same architecture of generators and discriminators for all the classes except *Cell clusters.*

2.3.4 Augmenting the Dataset

Since SSIM and PSNR metrics tend to ignore the higher order characteristics of the image, we evaluated the quality of the synthetic images by training the classifier using different proportions of real and synthetic images. The assumption of this approach is that, if the synthetic images have similar higher order characteristics compared to the real images, then the features learned by the CNNs during the training on the synthetic images, should also be able to classify the real images.

To verify this assumption, we trained and evaluated our hierarchical classifier in two different data settings:

- Training on 100% real images.
- Training on 100% synthetic images.

Training on 100% real images is the same experiment as reported in Table 5. Table 10 shows the accuracy for each fold in the 5-fold cross validation using the 724 real hESC images. In the second data setting, we trained our fused CNN-Triplet classifier exclusively on the synthetic images and evaluate the performance on the real hESC images. Table 11 shows the accuracy after training the classifier using different amounts of synthetic images.

Observing the results in Table 11, it can be seen that training the classifier exclusively with the synthetic images resulted in an increase in the classification accuracy. This verifies our assumption that the generated synthetic images do have similar higher order characteristics as the real images and hence augmenting our dataset helps the classifier to generalize better resulting in an increase in classification accuracy.

TABLE 10: Accuracy and number of images of each fold for the 5-fold cross validation using the 724 real hESC images. The number in the brackets indicates the number of images per class for Cell clusters, Debris, Unattached cells, Attached cells, Dynamically blebbing cells, and Apoptically blebbing cells respectively.

Cross validation fold number	**Number of images for training**	**Number of images for testing**	**Classification Accuracy (%)**
Fold 1	580 (88, 80, 100, 100, 76, 136)	144 (24, 23, 25, 22, 18, 32)	93.18
Fold 2	579 (90, 83, 100, 97, 75, 134)	145 (22, 20, 25, 25, 19, 34)	93.02
Fold 3	579 (90, 83, 100, 97, 75, 134)	145 (22, 20, 25, 25, 19, 34)	93.65
Fold 4	579 (90, 83, 100, 97, 75, 134)	145 (22, 20, 25, 25, 19, 34)	93.21
Fold 5	579 (90, 83, 100, 97, 75, 134)	145 (22, 20, 25, 25, 19, 34)	93.10
Average	-	-	**93.23 ± 0.24**

TABLE 11: Comparison of using different data compositions of synthetic images for training the classifier and then testing it on the 724 real images

Number of synthetic hESC images per class used for training	**Classification Accuracy on the 724 real hESC images**
5,000	93.84%
10,000	94.26%
20,000	94.31%
30,000	94.43%
40,000	**94.46%**

We verified the significance of the accuracy in Table 11 using the statistical *p*-value test. The *p*-value is calculated using the one-way Analysis of Variance (one-way ANOVA). One-way ANOVA is a technique that can be used to compare means of two or more experiments using the F distribution. We assume the training using real images in Table 10 and the training using synthetic images in Table 11 to be two different experiments. Based on this setting, the one-way ANOVA yields a F score ratio of 33.18, which corresponds to a *p*-value of 4.24×10^{-4}. We set the significance threshold of the *p*-value as 0.01. Since, the p-value (4.24×10^{-4}) is lower than the threshold (0.01), our results are proved to be significant.

2.3.5 Discussion of Results

In this section we discuss about the improvement in classification accuracy, quality of the generated synthetic images and the reasons for misclassification.

(a) Improvement in Classification Accuracy: This sub-section explains the reasons for the improvement in classification accuracy compared to our prior work in DeephESC [5]. We show that by concatenating feature maps from the early and final stages of the CNN, the CNN learns a better feature representation and helps reduce the misclassification between visually similar classes.

It can be observed from Table 8 that *Debris* and *Unattached cells* had the highest classification accuracy of 97.08% and 96.80%, respectively. The reason for this is that these two classes are visually very distinctive compared to *Cell clusters/Apoptically Blebbing cells* and *Attached cells/Dynamically Blebbing cells*.

On the contrary, in comparison with DeephESC [5], *Cell clusters/ Apoptically Blebbing cells* had the highest misclassification rate of 7.89%. The reason for this is that the CNN was not able to detect the small neighboring cells which distinguish a *Cell cluster* from an *Apoptically Blebbing cell* as depicted in Fig. 3. In DeephESC 2.0 we solved this by concatenating features learned from the initial and final convolution layers which helps the CNN learn a more robust feature representation as shown in Fig. 11 which in turn reduces the misclassification rate from 7.89% to 3.21%. Similarly *Attached cells/Dynamically Blebbing cells* have very similar intensities and texture with the only difference being in their morphology. *Attached cells* have a more uniform and homogeneous morphology compared to *Dynamically Blebbing cells*. By concatenating the features from initial and final convolution layers we are able to reduce the misclassification rate from 5.26% to 3.70%.

(b) Quality of the Generated Synthetic Images: This sub-section explains why Unattached cells have higher SSIM and PSNR scores compared to the other five classes. We also explain the disadvantage of using SSIM and PSNR to validate the quality of the images and how we overcome this problem.

It is observed from Table 9 that the SSIM and PSNR for the five classes *Cell clusters*, *Debris*, *Attached cells*, *Dynamically* and *Apoptically Blebbing cells* were relatively lower compared to the SSIM and PSNR for Unattached cells. The reason for this is that the structure of these five classes are much more complex and diverse compared to Unattched cells as shown in Fig. 13. SSIM and PSNR metrics compare the similarity between two images at a pixel level ignoring the higher order characteristics (such as the overall structure and texture). Although our approach is able to generate synthetic images which visually look similar to the original images, due to the diverse variations in shape even a slight change in corresponding pixel values will result in a significantly low SSIM and PSNR value.

Since SSIM and PSNR tends to ignore higher order characteristics of the image, we evaluated the quality of higher order characteristics of the synthetic images by training our classifier exclusively on the synthetic image and tested its classification accuracy on the real hESC images as shown in Table 11. The assumption here is that, if the real hESC images and the generated synthetic images have similar higher order characteristics, then the features learned by the CNN trained on the synthetic images should be able to also classify the real hESC images. From Table 11, we can observe that our CNN trained exclusively on synthetic images is able to classify the real hESC images with an accuracy of 94.46%. This observation validates our assumption that the generated synthetic images do have similar higher order characteristics as the real hESC images.

(c) Saturation of Classification Accuracy: This sub-section shows how the accuracy of the classifier varies with increasing amounts of synthetic images as well as the trade-off

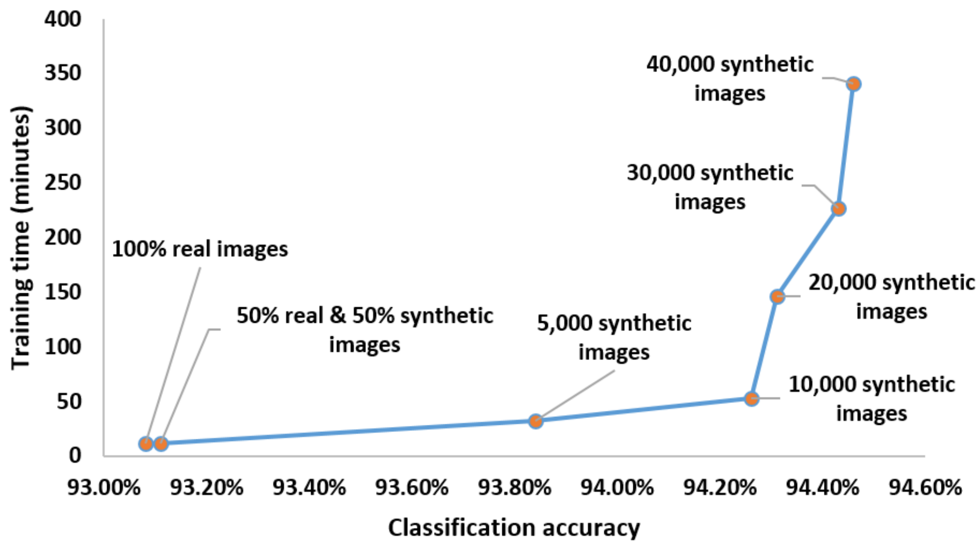

Fig. 14: Classification accuracy Vs training time trade-off.

between the number of images for training Vs the time taken for training. We also show some examples of hESC images that were predicted incorrectly by our classifier and explain the reason for the misclassification.

It can be observed from Table 11, the classification accuracy increases progressively as we generate more synthetic images, but after a certain amount of synthetic images (40,000 synthetic images per class) the classification accuracy does not significantly increase. In Table 11 we get an improvement in accuracy of only 0.03% from increasing the number of synthetic images from 30,000 to 40,000 per class but the time taken to train the classifier significantly increases. Hence, in order to balance the trade-off between the classification accuracy and the training time we limit the number of synthetic images per class to be 40,000. Fig. 14 shows the graph of the classification accuracy versus the training time trade-off.

A possible reason for the saturation in classification accuracy is that the ground-truth for certain images may have been labeled incorrectly by the biologists and the classifier is able to correctly classify these images even though the ground-truth is wrong. Fig. 15

shows examples of such images that were unintentionally labeled incorrectly by the biologist, but our classifier was still able to predict the correct class.

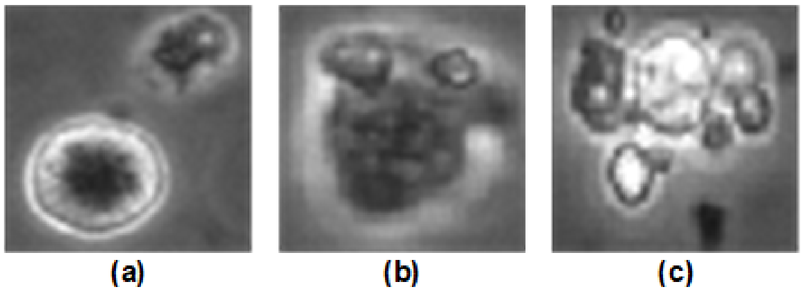

Fig. 15: Examples of images that were unintentionally labeled wrong by the biologist, but correctly classified by our classifier. (a) *Unattached cell* mislabeled as *Cell cluster*, (b) *Attached cell* mislabeled as *Dynamically Blebbing cell.* (c) *Apoptically Blebbing cell* mislabeled as *Cell cluster*.

Fig. 15(a) is an *Unattached cell*, but due to the presence of a growing *Dynamic Blebbing cell* near it, the biologist decided to label it as a *Cell cluster*. Fig. 15(b) is a *Dynamically Blebbing cell* that was mislabeled as an *Attached cell.* Since the morphology of these two classes are very similar, the biologist was not sure to which class the hESC belonged to. Fig. 15(c) is a *Cell cluster* mislabeled as *Apoptically Blebbing cell.* This is another example where the morphology of of two classes look very similar and the biologist was not sure as to which class the hESC belonged to.

Chapter 3

An Automated System for Generating Tactical Performance Statistics for Individual Soccer Players from Videos

In recent years, automatic interpretation of sports has gained a keen interest. It is a challenging task especially when it involves rapid changes and long-term dynamics. To date most of the applications for providing sports analysis and player training from video are carried out manually. This requires lots of hours spent watching videos and annotating them. Computer vision and machine learning play a key role in the world of sports in areas such as player detection, player tracking, action recognition and player analysis.

Soccer is one of the most popular sports played by high school students in the USA. According to a survey conducted by Ranker.com [43] and Statista.com [44] 846,844 (456,362 boys and 390,482 girls) high school students played soccer during the year 2017/18.

TABLE 12: Statistics of the number of male and female soccer players in high school, NCAA* and MLS* [44]

Year	**High school**		**NCAA***			**MLS***
	Male	**Female**	**Div 3**	**Div 2**	**Div1**	
2011/12	411,757	370,975	10,117	6,076	5,153	38
2012/13	410,982	371,532	11,097	6,165	4,832	38
2013/14	417,419	374,564	10,870	6,261	5,426	77
2014/15	432,569	375,681	11,679	6,489	5,623	84
2015/16	440,322	381,529	11,889	6,805	5,724	75
2016/17	450,234	388,339	11,256	6,754	5,953	81
2017/18	456,362	390,482	12,322	6,545	5,933	81

*NCAA stands for National Collegiate Athletic Association

*MLS stands for Major League Soccer

From this only 9% of the boys and 11.9% of the girls receive scholarship to go to college which makes it extremely competitive.

Table 12 shows the statistics of the number of male and female soccer players in high school, the National Collegiate Athletic Association (NCAA) and the Major League Soccer (MLS) for the years 2011-2018. After graduating from high school, less than 6% of soccer players get qualified for the NCAA. The NCAA consists of three tiers namely: Division 1, Division 2 and Division 3. Most of the Major League Soccer (MLS) recruiters seek out only the players in Division 1 and Division 2 and less than 100 of them qualify to become pros in the MLS every year. The most common reason for players not qualifying for the NCAA or other soccer clubs is that coaches do not have enough time to observe the performance of every player. To alleviate this problem, we propose an automated system that can detect, analyze all of the soccer players, identify the player controlling the ball in a video and generate the tactical statistics of each player.

In team-based sports, such as soccer, talent identification is a complex process due to the different qualities associated with performance; they include technique, tactics, fitness and psychological attributes [45].

Technique involves the player's style such as dribbling and how offensive/defensive is that player.

Tactics involve attributes such as how well is the player able to control the ball and play with the team-mates.

Fitness involves attributes such as the fatigue, stamina level of the player and history to injuries.

Psychological involves the emotional intelligence of the player and how the player deals with interpersonal and intrapersonal conflicts.

In this chapter, we focus on generating directly from the video, three very important tactical statistics for a soccer player namely: **(i)** duration of ball possession, **(ii)** number of successful passes and **(iii)** number of successful steals.

Importance of tactical statistics: Ball possession is a very important statistic (stat) for a player as it is has an influence on other statistics such as the number of successful shots at the goal and number of tackles won/lost [46]. The number of successful passes made by a player is very important because the number of overall attempted passes and number of successful passes are important factors in achieving better results leading to winning a match [47–49]. It has been shown that the accuracy of successful passes increase significantly five minutes before scoring [50]. The number of successful steals within the 6 yard area has been shown to increase the number of shots at the goal [51]. Interestingly, unsuccessful teams tend to play more in the pre-defensive area which increases the chances of the opposition to steal the ball which in turn also increase their chances of scoring [52].

In order to develop an automated system to compute the tactical statistics for players we collected a dataset that consists of 49,950 images of high school soccer players which are annotated into two classes namely: "*Players with the ball*" (12,585) images and "*Players without the ball*" (37,365) images. The first step in our system involves detecting the players using the YOLOv2 framework [53] and tracking them using the DeepSort algorithm [56]. Next, the detected players are passed through a Triplet-Convolutional Neural Network (CNN) that extracts fine-grained features which are used for predicting the team of the player and finding out if the player is controlling the ball. While trying to solve this problem, we also address two key issues: *Speed Vs. Performance* and *Generalizability.*

Speed Vs Performance: In the field of sports analytics, the speed at which algorithms perform without sacrificing accuracy is very important. For example, on the COCO dataset [57] the algorithms with the best performance are rather slow [58, 59], while the real-time algorithms have lesser accuracy [53, 60]. In this chapter, we experiment with different architectures of CNNs and show that during inference, our approach is able to perform relatively faster compared to the state-of-the-art approaches while not sacrificing accuracy. ***It should be noted that our system is not intended to run real-time but instead to be used as a tool for post-match analysis***.

Generalizability: A last problem can be the lack of generalizability, whose origin is at least two fold in sports video analysis: inter-sport variability, and intrasport variability. It is currently too ambitious to hope for a universal system that can perform accurate player analysis on any sports video, which underlines the need for developing sports-specific models. Besides, even within videos from a single sport, some play conditions may change from one match to the next, such as the outfits of the teams and environmental conditions in the case of outdoor sports such as soccer. Fast algorithms can be less robust to such variations, which might make them non-reusable from one match to the next.

Rather than trying to unify all of these conditions within a single network, it is more appropriate to re-train a specific network for every match in order to adjust to the conditions of that match [61]. To achieve this we experiment with different soccer matches played by different teams and find the least amount of images that need to be annotated in order to achieve robust performance. Another problem that arises is that how do we annotate images for a match that has not yet been played? To solve this we annotate images of matches previously played by the same teams and then re-train our models and evaluate it on the match that is to be played.

3.1 Related Work and Our Contributions

In this section, we describe various state-of-the-art approaches that have been used in sports for detecting and tracking players, identifying teams and events, and player analysis.

3.1.1 Player Tracking

Duh *et al.* [62] used a 2D Gaussian model for segmenting the soccer field followed by a histogram based method and a Spatial Similarity Matrix for detecting and tracking players. Liu *et al.* [63] performed an adaptive Gaussian background subtraction followed by clustering to detect soccer players. Finally the detected players are tracked using the Markov Chain Monte Carlo (MCMC) association. Chiang *et al.* [64] assume that the histogram of a moving player is narrower compared to the histogram of the entire frame. Based on this the mean shift algorithm is used for segmenting and tracking the player

Xing *et al.* [65] used a Bayesian inference approach that dynamically switches between an offline general model and an online dedicated model to track players under occlusions. D'Orazio *et al.* [66] track soccer players by first detecting the soccer players in the video using background subtraction. Bounding box features such as the position and velocity are extracted for each player. These features are then matched with the detected players in the next frame. Khatoonabadi and Rahmati [67] detected the soccer players using perspective transformations and tracked them using histogram based template matching

3.1.2 Player and Team Detection

Senocak *et al.* [68] used CNNs to extract holistic features from an image of a player and each part of his body which is encoded into a Fisher vector representation [69] and passed to a Support Vector Machine (SVM) for classification. Xu *et al.* [70] detected soccer players from 4K videos in multiple stationary cameras. The authors divided the video into non-overlapping regions of size 416×416 and used a YOLOv3 detector to detect the player in the regions where foreground movement was detected.

Liu and Bhanu [71]designed an automated system using a pose-guided R-CNN for detecting the jersey numbers of players from different sports and classify the digits. Istasse *et al.* [72] used a lightweight CNN to learn a pixel-wise embedding vector, that is similar for players from the same team and dissimilar for different teams. Theagarajan *et al.* [74] used histogram matching and a YOLO framework to detect and distinguish players from different teams. In their approach the authors manually cropped 10 template images of soccer players for each team.

3.1.3 Event Detection and Player Analysis

Cai *et al.* [75] detected the actions of ice hockey players using Part Affinity Fields (PAF) [76] and extracted temporal features using optical flow [77]. The pose and optical flow features are then fused to predict the action of the player. Piergiovanni and Ryoo [78] introduced the MLB-YouTube dataset that consists of 20 baseball games from the 2017 MLB post-season with over 42 hours of footage. The authors The authors experimented with various temporal feature pooling methods to classify the events in baseball. Cioppa *et al.* [79] used the contextual information from soccer broadcast videos such as the line marking and player position to predict how offensive/defensive the players were playing.

Tora *et al.* [80] classified the puck possession events in hockey using a player based CNN, a frame based CNN and a Long Short Term Memory (LSTM) recurrent neural network. The authors extract contextual and holistic features from the CNNswhich are given as input to train the LSTM. Li and Bhanu [81] used Mask-R-CNN [82] and OpenPose [83] to compute the Dribble Enery Image (DEI) and then classify the dribbling style of soccer players. Theagarajan *et al.* [74] designed an automated system to detect using the YOLO framework and VGG-16 CNN to and classify the soccer player controlling the ball videos recorded using a single un-calibrated camera.

3.1.4 Contributions of this chapter

In contrast to the state-of-the-art approaches described above, the contributions of this chapter are as follows:

- **Generating Tactical statistics**: To the best of our knowledge, this is the first approach in the field of computer vision, sports analysis and circuits and systems for video technology, that can automatically generate three quantifiable tactical statistics (duration of ball possession, number of successful passes, and steals) of individual soccer players from a video. In addition, an ablation study is carried out to show how different combinations of the individual modules affect the generation of tactical statistics at a match level and individual player level.

- **Team Identification**: Unlike previous approaches that use clustering based techniques such as [63, 67, 72] and ad hoc histogram based matching such as [73, 74, 85] which are susceptible to the player pose and environmental conditions, we evaluate three different approaches using Siamese and Triplet CNNs and show that our approach is more robust and outperforms the state-of-the-art approaches by 26%.

- **Player Analysis**: Prior work done by Theagarajan *et al.* [74] shows that regular CNNs have trouble in detecting minute details such as the soccer ball in low resolution images which is very important for differentiating between a player with and without the ball. To overcome this problem, we extract fine-grained features using a Triplet CNN trained on only 100 images per class and show that our approach outperforms the state-of-the-art by at least 14% and has significantly lesser number of parameters.

- **Generation of Fine-grained Synthetic Images**: This chapter shows that regular Generative Adversarial Networks (GANs) often overlook minute details such as the soccer ball when generating synthetic images. To overcome this problem, this chapter designs a Triplet CNN-DCGAN for generating fine-grained images of soccer players controlling the ball and performs an ablation study to show the improvement in generating tactical statistics with and without data augmentation.

3.2 Technical Approach

In this section, we explain the framework of our approach and its individual modules as shown in Fig. 16. The input video first passes through the player detection module where the soccer players are detected, tracked and cropped. Next, the cropped images are passed through a player classification module which consists of two Triplet CNNs that are trained to extract fine-grained features and 1) predict the team of the player and 2) identify the player controlling the ball. Next we pool together the outputs of the player detection and player classification modules for the entire video to generate the tactical statistics for all of the individual players.

3.2.1 Localization and Tracking

Localization of Soccer Players In our approach we detected the soccer players in a video using the YOLOv2 framework proposed by [53]. YOLOv2 consists of a CNN that predicts multiple bounding boxes for an image along with the respective class probabilities for each bounding box. YOLOv2 initially divides the input frame into a 11×11 grid. Each grid predicts B bounding boxes and the confidence score associated for each bounding box.

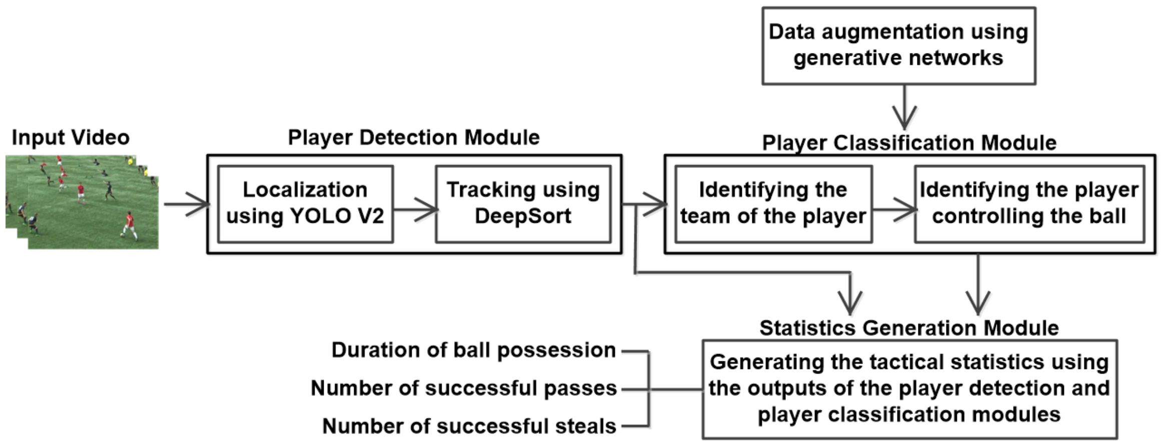
Data augmentation using generative networks
Input Video
Player Detection Module
Localization using YOLO V2
Tracking using DeepSort
Player Classification Module
Identifying the team of the player
Identifying the player controlling the ball
Statistics Generation Module
Duration of ball possession
Number of successful passes
Number of successful steals
Generating the tactical statistics using the outputs of the player detection and player classification modules

Fig. 16: Overall architecture of our approach.

Formally, the confidence is defined as Pr(object) * IOU, where Pr(object) is the probability of an object present and IOU is the Intersection Over Union between the predicted bounding box and the ground-truth bounding box. This probability is conditioned on the grid cell containing one object meaning that, if there is no object present on the grid cell, the loss function will not penalize the CNN for a wrong class prediction.

The network was trained on the COCO 2016 keypoints challenge dataset [57] as this dataset consists of diverse images for the class "*Person*" which also includes sports players. The images in these datasets have different scale variations, and occlusions which is similar to the scenario of a soccer field. For a given frame, the bounding boxes belonging to the class "*Person*" with probability greater than a given threshold are considered to be the locations of the soccer players for that frame. In our approach we set the threshold to be 0.5.

Tracking the Soccer Players After detecting the soccer players in consecutive frames, we use the DeepSort long-term tracking approach proposed by Wojke *et al.* [56] who reformulate the tracking association as the re-identification problem. In traditional broadcast videos of soccer league matches, the camera operator is located at least 100 feet away from the side lines of the soccer field at a reasonable height. This provides a large Field-of-View (FoV) for the camera operator and there is very small amount of gradual pan and tilt. In these kinds of videos when a player is moving on the field, the camera operator also pans the camera very gradually such that the cartesian coordinates of the soccer player in the video has minimal change. In high school soccer videos such as our dataset, due to the availability of limited space, the camera operator is located just 20 - 30 feet from the side lines of the soccer field at a height of 15 feet. This creates a very limited FoV for the camera operator which leads to large amount of pan, tilt and zoom even when the soccer players are not running very fast.

Based on these requirements we experimented with five state-of-the-art tracking algorithms (including DeepSORT and algorithms proposed in [86]) and found DeepSORT to perform the best given the current scenario. The reason for this is that, unlike the other mentioned algorithms that solely rely on the features extracted from object detectors, DeepSORT also uses an 8-dimensional state space vector which is given as input to a Kalman filter. This state space vector contains the information such as velocity and direction in which the player is moving relative to the camera. Assuming that the players usually move at a constant velocity relative to the camera that is located at a distance and the tracklets of a player always follow a linear model (a player cannot arbitrarily appear at different locations in consecutive frames) the Kalman filter is able to track the individual players more consistently. In the case of soccer videos if a group of players of the same team are close to each other, it would cause association problems if the tracking algorithm solely relied on the features extracted from an object detector. DeepSORT is able to handle these kind of situations with much ease compared to the other methods because it uses the combination of an object detector and a state space Kalman filter that assumes that players move linearly in a video.

In our approach the CNN used for detecting the players is YOLOv2 and the 8-dimensional state-space vector is represented by (u, v, γ, h, u', v', γ', h') where, (u, v) is the image coordinate of the center of the bounding box, γ is the aspect ratio, h is the height of the bounding box and (u', v', γ', h') are their respective velocities in the image coordinate. Finally, the features extracted by YOLOv2 and the state-space motion vector are concatenated and passed as the input to a Hungarian algorithm [87].

Moreover, it should be noted that our current system does not account for any camera calibration and the camera operator is allowed to freely pan, tilt and zoom the camera depending on where the action is happening on the soccer field. Due to this if a player moves out of the view of the camera and re-appears after a while, the algorithm associates the player as a new person. In this situation our approach will treat these players as new players and continue to generate the statistics. It is very challenging to track players under such conditions using state-of-the-art long-term tracking and re-identification algorithms and it is a problem of its own. In the case of soccer videos, it is more challenging because players belonging to the same team wear the same jersey and they visually look very similar from the camera's perspective.

3.2.2 Team Identification

In this sub-section we experiment three different approaches (*TI-1, TI-2, and TI-3*) for predicting the team of the players and compare their pros and cons individually.

TI-1: Cross Dataset Transfer Learning and Feature Matching using Siamese CNN: In soccer, since players belonging to the same team wear the same color of jersey, we can formulate the task of player team identification as a person re-identification problem. In this approach we train a Siamese CNN on the Town Center subset of the PETA dataset [88]

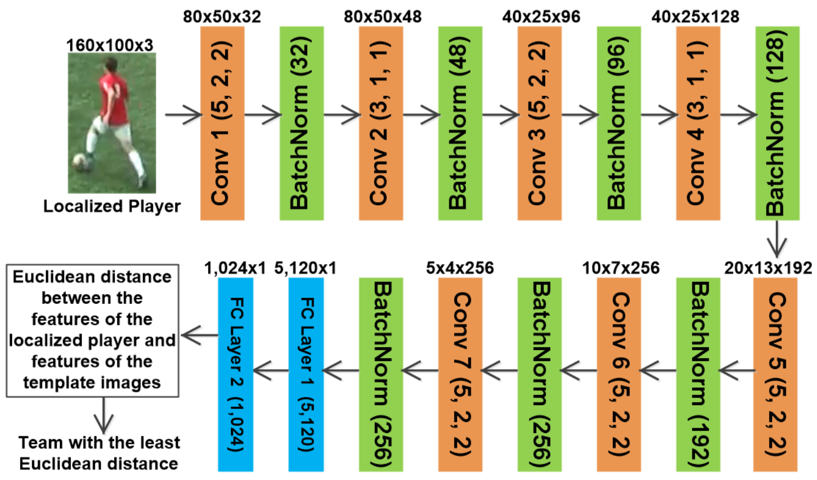

Fig. 17: Architecture of the Siamese CNN.

for the task of pedestrian re-identification with two output classes "Same person" and "Different person". Fig. 17 shows the architecture of the Siamese CNN. In Fig. 17 Conv(x, y, z) represents the dimension of the filter (x), stride of the filter (y), and padding (z). We used the Siamese loss function for training and it is given by:

$$Loss_{Siamese} = (1-Y)\frac{1}{2}D_W^2 + Y\frac{1}{2}\{max(0, m - D_W)\}^2 \tag{1}$$

In Eq. 1, D_W is the Euclidean distance between the outputs of the Siamese networks, m is the margin and is chosen as 1 and Y is either 0 or 1. If the inputs are from the same class, then the value of Y is 0, otherwise Y is 1.

After training the Siamese CNN on the PETA dataset, we evaluate the trained CNN on our dataset. Initially we select 10 template images for each team. Next, we pass the detected player through the Siamese CNN and we extract a feature vector X. Similarly, we also extract the feature vectors of the 10 template images of each team Y_i where $i = 1$ to 10. Next, we compute the average Euclidean distance between X and Y_i for both the teams and the team with the least average Euclidean distance is taken as the final prediction.

TI-2: Fine Tuning and Feature Matching using Siamese CNN: This approach is similar to that of **TI-1** except that, after pre-training on the PETA dataset we further fine tune the Siamese CNN with images from our dataset as well. This helps the CNN to learn features specific to the match being played which improves the performance of the prediction. During testing similar to **TI-1**, we select 10 template images from each team and compute the average Euclidean distance between the feature vector of the detected player X and the feature vector of the template images Y_i for both the teams and the team with the least average Euclidean distance is taken as the final prediction. Experimental results (see Table 17) show that this approach performs better than **TI-1** because in **TI-2** the CNN is able to learn features that are more specific to the soccer match.

TI-3: Fine-grained Feature Extraction using Triplet CNN: Triplet CNNs are known for extracting fine-grained features while maximizing the interclass variance and minimizing the intraclass variance at the same time [5, 27, 89, 90]. In this approach we use the same CNN architecture as in Fig. 17 and the only change is that we replaced the final fully connected layer (FC Layer 2) with two output nodes for "Team A" and "Team B". During testing, this approach does not require any template images for matching. We use both the Triplet loss (eq. 2) and binary cross entropy loss (eq. 3) for training.

$$Loss_{Trip} = Max(0, -Y * (G(X_1) - G(X_2)) + margin) \tag{2}$$

$$Loss_{BCE} = \frac{-1}{n} \sum_{i=1}^{n} y_i * log(p_i) + (1 - y_i) * log(1 - p_i) \tag{3}$$

$$Loss = \alpha_1 \times Loss_{Trip} + \alpha_2 \times Loss_{BCE} \tag{4}$$

In Eq. 2, *X_1* and *X_2* are the two anchor images and *$G(X)$* is the pairwise distance between the feature extracted by Triplet CNN for the localized player image and the anchor image. If $Y = 1$ it indicates that the anchor image X_1 belongs to the same class as the localized player image, whereas, Y = -1 indicates that the anchor image X_2 belongs to the same class as the localized player image.

In Eq. 3, y_i is the ground-truth label and p_i is the output probability score for the respective classes. In Eq. 2, α_1 and α_2 are constants and are chosen to be 0.5. The advantage of this approach compared to **TI-1** and **TI-2** is that, this approach does not require any template images during inference and gives us the highest accuracy.

3.2.3 Identifying the Player Controlling the Ball

To generate player statistics and visual analytics for soccer, we need to identify the player who is in control of the ball at any given point of time. To achieve this, we trained another Triplet CNN with the same architecture used in chapter 4.2.2 (TI-3) (the two CNNs do not share the same weights) to classify a given cropped image of the soccer player as either a "*player with the ball*" or "*player without the ball*". The cropped images of the soccer players are resized to size 160×100. We chose this size because the normal aspect ratio of a human body is between 0.6 - 0.7. We chose a mini-batch size of 256 Triplet pairs and during every epoch the training data is randomly shuffled and randomly horizontally flipped.

We used a combination of both the Triplet loss and binary cross entropy loss as shown in Eq. 2 - 2 to train the Triplet CNN. Furthermore, we separated a part of our training data as the validation dataset for finding the best training hyper parameters. The validation dataset is used only for finding the best hyper parameters and it is never used for training. We performed random hyper parameter search to obtain the best learning rate, momentum and weight decay. This is done by training and validating the network with random values within a range for each hyper parameter for 5 epochs, and the combination of hyper parameters that resulted in the highest accuracy at the end of 5 epochs were chosen as the best. Based on this we chose the learning rate = 2 $\times 10^{-2}$, momentum = 0.7 and weight decay = 4 $\times 10^{-3}$. Finally, the networks were optimized using the stochastic gradient descent algorithm.

3.2.4 Data Augmentation using Triplet CNN-DCGAN

In this sub-section, we explain on how we performed data augmentation to our dataset. The purpose of data augmentation is to determine if adding more variability to the training dataset helps to improve the generation of tactical statistics. To achieve this we trained the Deep Convolutional Generative Adversial Network (DCGAN) proposed by [42]. It consists of two deep convolutional neural networks, a generator G and a discriminator D trained against each other. The generator takes a randomly sampled Gaussian noise vector, z, and returns an image, $X_{gen} = G(z)$. The discriminator takes a real or a generated image X, and outputs a log probability $P(S|X) = D(X)$ over the two image sources S. The optimization function V is given by:

$$\begin{aligned} \min_G \max_D V(D,G) = \quad & \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \\ & + \mathbb{E}_{x\sim p_z(z)}\left[\log\left(1 - D(G(z))\right)\right] \end{aligned} \tag{5}$$

In eq. 5, $log(D(x))$ is the log probability of the output of the discriminator and D is trained to maximize the probability of assigning the correct label to the image (*i.e.* is the image original or generated) while G is simultaneously trying to minimize it. The significance of eq. 5 is that by doing a minimax optimization, we are essentially pitting the generator against an adversary that can detect if an image is a counterfeit or not. This encourages G to learn the distribution of the dataset and hence generate images that resemble the dataset. The final objective is that the two networks converge to an equilibrium so that D is maximally confused and G generates samples that resemble the training data.

Fine-grained Synthetic Image Generation: After training the DCGAN, we observed that after the generator and discriminator have reached an equilibrium the generator was able to produce images of the soccer player but, most of these images did not have the soccer ball in it. Generating the soccer ball with the player is the most important feature for distinguishing a "*player with the ball*" from a "*player without the ball*". In our approach we solve this problem by introducing a regularizer CNN into the DCGAN architecture. The task of the regularizer CNN is to classify the images generated and if the image is classified as a "Player without the ball", the generator is penalized more. So now the task of the generator is not only to fool the discriminator but also generate images that resemble a "*player with the ball*". In our approach we used the Triplet CNN trained to identify the player controlling the ball as the regularizer CNN. Fig. 18 shows the Triplet CNN-DCGAN architecture.

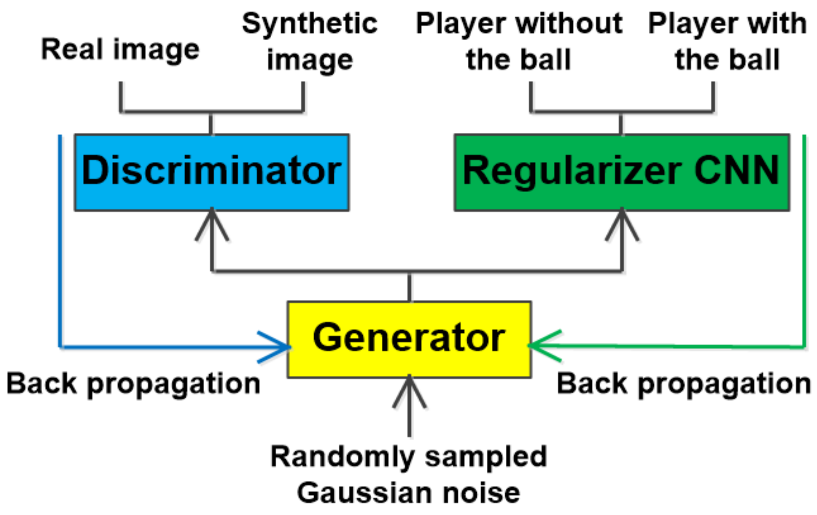

Fig. 18: Architecture of the Triplet CNN-DCGAN.

In Fig. 18, it should be noted that the regularizer CNN (i.e., Triplet CNN) is pre-trained to classify between the two classes "*Player with the ball*" and "*Player without the ball*" and its parameters are frozen after pre-training. This means that we do not update the weights of the regularizer CNN while training the generator. In this new architecture, we use the regular DCGAN loss as described in [42] along with the binary cross entropy loss of the Triplet CNN.

We included the regularizer CNN in the loop only after the generator and discriminator have already reached an equilibrium. The reason for this is that initially while training the DCGAN, the generator does not generate realistic images and if we pass these unrealistic images to the Triplet CNN, it would not be able to recognize the images, resulting in mis-classifications leading to an erroneous back propagation. Hence after the discriminator and generator have reached an equilibrium and the generator is able to produce realistic images of soccer players, we include the Triplet CNN in the loop. Fig. 19(a) shows images generated using the regular DCGAN approach [42] and Fig. 19(b) shows images generated using our fine-grained Triplet CNN-DCGAN approach. In Fig. 19(b) the red bounding box indicates the location of the soccer ball in the image.

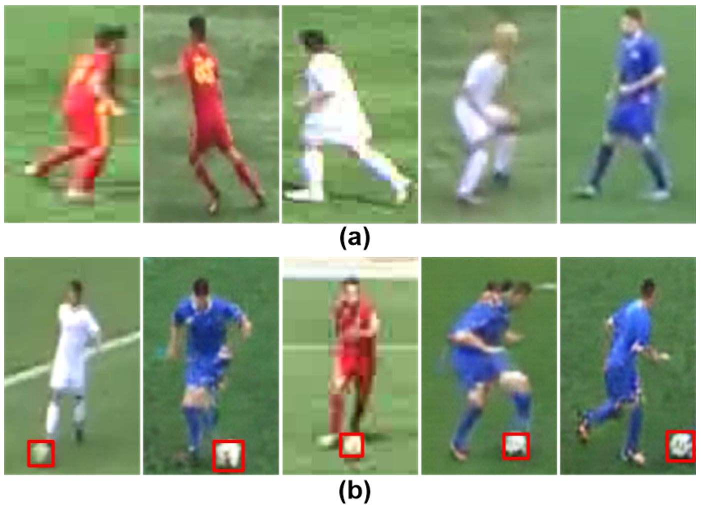

Fig. 19: Generated images of the class “Player with the ball” using (a) DCGAN [42] and (b) Triplet CNN-DCGAN.

3.2.5 Tactical Statistics Generation

In this sub-section we explain our algorithm for generating the tactical statistics of the soccer players in the video. We pool together the outputs of the player detection module and player classification modules shown in Fig. 16 for the entire video as inputs to our statistics generation module. The outputs of the statistics generation module are the duration of ball possession, number of successful passes and number of successful steals for every player in the video. The pseudo code for generating the tactical statistics is given below. In the pseudo code i is the frame number and j is the number of players detected in frame i. *ID[i][j]* contains the tracking IDs of all j players in frame i. This information is obtained as the output from the player detection module. *x[i]* contains the tracking ID of the player controlling the ball in frame i and *y[i]* contains the team name of the player controlling the ball in frame i. This information is obtained as the output from the Player classification module.

Pseudo code for generating the tactical statistics

Inputs: 1) List of tracking IDs (ID[i][j]), 2) Player with ball dictionary (x[i]), 3) Team dictionary (y[i]) where, i is the frame number of the video and j is the ID of the tracked player in frame i.

Outputs: Ball possession dictionary[], successful passes dictionary[], and successful steal dictionary[].

```
Initialize all the key-value pairs for the ball possession, successful passes and successful steal dictionary to 0
Initialize Players_tracked list to be empty
For i = 1 to N frames in the video
  do
      % This block keeps track of new incoming players
      For j = 1 to length (ID[i])
end
```

```
        if (ID[i][j] does not belong in Players_tracked):
          Append ID[i][j] to Players_tracked
          ball possession dictionary[ID[i][j]] = 0
          successful passes dictionary[ID[i][j]] = 0
          successful steal dictionary[ID[i][j]] = 0
  % This block computes the statistics
  while (i>1 and i<=N):
    % Is the ID of the player controlling the ball in frame i
     and frame i - 1 different?
    if (x[i] is not equal to x[i-1]): % Yes
      % Are the two players in the same team?
      if (y[i] is equal to y[i-1]): % Yes
        % It's a successful pass
        successful passes dictionary[x[i-1]] += 1
        ball possession dictionary[x[i]] += 1
      else: % No, they belong to different teams
        % It's a successful steal
        successful steal dictionary[x[i]] += 1
        ball possession dictionary[x[i]] += 1
    else: % No, it's the same player controlling the ball
     ball possession dictionary[x[i]] += 1
  end
```

3.3 Experimental Results

We trained and evaluated our approach on a dataset collected from three different high school soccer matches. The overall framework of our approach is implemented using 2 TITAN X GPUs with 7 TFlops of precision, 336.5GB/s of memory and 12 GB of RAM memory per GPU.

3.3.1 Dataset

We collected a dataset from three different soccer matches. The matches played by the teams were recorded using a single Canon XA10 video camera. The camera was installed at a height of 15 feet and 20 feet away from the horizontal baseline of the soccer field. The resolution of the recorded video is 1280×720. The camera operator was allowed to pan and zoom depending on where the action is happening on the soccer field in order to collect high resolution and good quality images with enough pixels on a player's body. To the best of our knowledge, the only other dataset that has annotations of soccer players in the video, was done by Pettersen *et al.* [91]. The authors used three stationary wide angle cameras installed inside the control room behind the audience in the Alfheim stadium in Norway. Since the control room is very far away from the soccer field (at least 100 feet) the resulting videos have very small number of pixels on the soccer player and even fewer pixels on the soccer ball making it difficult to distinguish the player controlling the ball. Hence we could not use this dataset.

Ground-truth for Training Data Our dataset consists of 49,950 images, and it is annotated into two classes namely: "*Players with the ball*" (12,585 images) and "*Players without the ball*" (37,365 images). The dataset consists of three teams whose jersey colors are white, red and blue. Out of the 49,950 images, the white team constitutes 13,960 images, the red team constitutes 17,390 images and the blue team constitutes 18,600 images of the dataset. Within the two classes, the white, red and blue team constitute 26.12%, 16.16% and 57.73% for the class "*Players with the ball*" and 28.58%, 41.26% and 30.16% for the class "*Players without the ball*", respectively. Table 13 shows the distribution of the two classes in our dataset. Fig. 20(a) and Fig. 20(b) shows example images of "*Players with the ball*" and "*Players without the ball*" from our dataset, respectively.

TABLE 13: Data distribution for the two classes with respect to the teams

Class	White team	Red team	Blue team
Player with the ball	3,348	2,071	7,400
Player without the ball	10,612	15,319	11,200

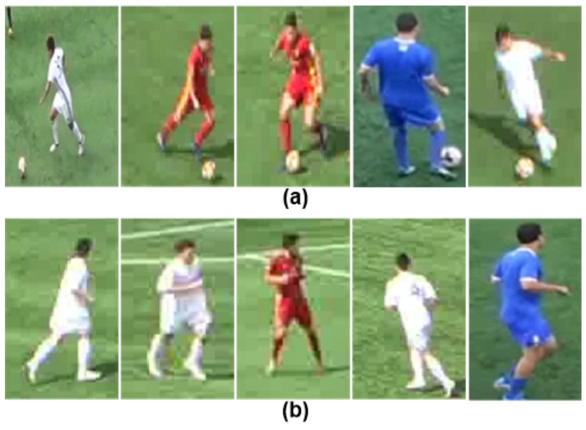

Fig. 20: Examples of players in our dataset for the class: (a) Player with the ball and (b) Player without the ball.

It should be noted that in our approach we are not generating tactical statistics for the goal keeper. The reason for this is that a goal keeper is evaluated based on the number of goal shots saved and since we are not generating that statistic we ignore the goal keeper and did not annotate any images of goal keepers in our dataset. From Table 13 it can be seen that the dataset is highly unbalanced which makes it challenging. The reason for this is that for every frame of the video only one player can control the ball which leaves 21 other players without the ball. But as the camera is being panned and zoomed not all 22 players are present in a single frame all the time, resulting in 25.66% of the data constituting for the class "*Players with the ball*" and 74.34% of the data constituting for the class "*Players without the ball*".

Clearly from Table 13 it can be seen that the dataset is highly unbalanced which makes it challenging. The reason for this is that for every frame of the video only one player can control the ball which leaves 21 other players without the ball. But as the camera is being panned and zoomed not all 22 players are present in a single frame all the time, resulting in 25.66% of the data constituting for the class "*players with the ball*" and 74.34% of the data constituting for the class "*players without the ball*".

Detailed Ground-truth Generation for Testing Videos of Varying Complexities: It should be noted that the dataset consisting of the 49,950 images does not have any annotations of player's positions in the video or their tactical statistics. Hence, in order to evaluate the performance of the player detection, tracking and tactical statistics generation module, we annotated seven highlight test videos for *evaluation/testing only.* Six of these highlight videos were extracted from matches played between the *Red jersey* Vs. *White jersey* categorized into three different complexities namely: ***Low*** (2 video clips), ***Moderate*** (2 video clips) and ***Severe*** (2 video clip). The duration of these six videos ranges from approximately 90 to 180 seconds. Since our approach uses only a single un-calibrated camera, we selected these six highlight videos such that the camera was not moving (pan, tilt, or zoom) faster than the players on the field. We also ensured that the FoVs of the camera in the highlight videos were large enough such that we do not have players entering/exiting the FoV of the camera for a long duration and then re-appearing elsewhere in the video. The 7th highlight video is publicly available on the internet and the match was played between a *white* and *blue* jersey team. The duration of this video is 31 seconds and it contains small segments of the *Low*, *Moderate*, and *Severe* complexity. This video contains segments where the players are entering/exiting the FoV of the camera and the tracking algorithms cannot associate them if the duration of entering/exiting is longer than 5 seconds. In order to evaluate the tracking algorithms in a fair manner, we consider the players who exit/enter after 5 seconds in the video to be new players. It should also be noted that these highlight videos were not used for training the CNNs.

We have carefully chosen these seven videos such that each video shows some level of complexity during different segments of the match. We chose the videos in the *Low* complexity when there are 4 - 5 players of the same team passing the ball among themselves and they are widely spread out. We can see this kind of play in a match when a team is trying to stall the opposing team. It is relatively easy to predict the statistics in this *Low* complexity case as compared to the *Moderate* and *Severe* complexity cases. In the *Moderate* complexity case we chose the videos where the mid-fielders are trying to penetrate the oppositions defense by passing the ball between their team mates while the opposition mid-fielders are trying to steal the ball. This scenario causes a lot of occlusion as there are usually 6 - 10 players in a small area as compared to the *Low* complexity case that makes it even more challenging to generate statistics. In the *Severe* complexity case, the offensive players try to aim for a shot at the goal while trying to avoid the opposition defenders and mid-fielders simultaneously. This scenario involves significant occlusion because usually there are several offensive players from the same team who pass the ball among themselves while there are at least 2 defenders and more than 2 mid-fielders from the opposition trying to steal the ball within a very narrow area of the field. This scenario causes the most occlusion leading to a significant drop in performance of generated statistics (see Table 24). In summary, these seven videos are representative and the results from our approach demonstrate how the overall system will perform under different segments of a match.

The ball possession for these highlight videos was annotated by identifying the player controlling the ball in all of the frames of all videos. The number of passes and steals made by each player in a video were annotated by observing the video and identifying the tracking ID of the players and the frame numbers of the video when the ball was passed/stolen.

3.3.2 Results for the Player Detection Module

In this sub-section we evaluate the performance of the player detection and tracking algorithms using the seven highlight test videos described above.

Localization Results and their Comparisons: We evaluated four state-of-the-art object detection algorithms namely: YOLOv2 [53], Single Shot Detection (SSD) [60], OpenPose [83], and Mask R-CNN [82] for detecting the soccer players in video. We used Intersection Over Union (IOU) between the ground truth and predicted bounding box and the speed of performance during inference in Frames Per Second (FPS) as metrics. Since we do not have any annotated training data with the soccer players position we trained all the above localization approaches on the COCO dataset [57] and then evaluated them on our highlight test videos. We chose the COCO dataset because it has a lot of diverse images for the class "*Person*". Table 14 shows the detection results on the seven highlight test videos.

TABLE 14: Performance metrics of different approaches for detecting the soccer players.

Approach	**Average IOU (%)**	**Average Speed (FPS)**
YOLOv2 [53]	84.81 ± 2.74	**17.2 ± 0.3**
SSD [60]	74.30 ± 2.03	15.8 ± 0.2
OpenPose [83]	69.45 ± 3.74	6.4 ± 0.2
Mask R-CNN [82]	**86.68 ± 2.58**	1.7 ± 0.4

All the approaches in Table 14 were evaluated using two TITAN X GPUs. From Table 14 it can be observed that YOLOv2 [53] had the highest average speed of performance with 17.2 FPS and average IOU accuracy of 84.68% while OpenPose had the least average IOU accuracy of 69.53%. SSD had a similar speed of performance compared to YOLOv2 but fell short in its average IOU accuracy. Mask R-CNN [82] had the highest average IOU of 86.47%, but had the least speed of performance of only 1.7 FPS making it impractical to use in our approach compared to YOLOv2.

Tracking Results and their Comparisons: We evaluated three deep learning based [54] - [56] and two hand-crafted features based [66, 67] long-term tracking algorithms. Table 15 shows the average Multi Object Tracking Accuracy (MOTA), Mostly Tracked (MT), and processing speed of each algorithm evaluated on our 7 highlight videos. MOTA is the accuracy of assigning the correct tracking ID to a player during all the frames the player is detected. MT is the accuracy of assigning the correct tracking ID to a player for at least 70% of their tracking duration. In Table 15 since we are only comparing the performance of tracking algorithms, for fair comparison we replaced the object detection in [66] and [67] with YOLOv2 [53] and used [54] and [55] as proposed in their paper.

TABLE 15: Performance comparison of tracking algorithms

Approach	**Average MOTA* (%)**	**Average MT* (%)**	**Average speed (FPS)**
Feichtenhofer *et al.* [54]	72.60 ± 7.37	60.88 ± 3.51	14.1 ± 0.4
Bertinetto *et al.* [55]	63.87 ± 6.55	49.02 ± 4.34	16.3 ± 0.3
D'Orazio *et al.* [66]	29.78 ± 3.22	15.63 ± 4.22	**20.6 ± 0.4**
Khatoonabadi and Rahmati [67]	23.35 ± 2.13	10.89 ± 2.27	20.4 ± 0.4
DeepSORT [56]	**76.59 ± 6.32**	**63.57 ± 3.85**	17.2 ± 0.4

*MOTA and MT refer to Multi Object Tracking Accuracy and Mostly Tracked, respectively.

From Table 15 it can be observed that DeepSORT had the best MOTA and MT with a speed of 17.2 FPS. The hand-crafted approaches of [66] and [67] had the worst MOTA and MT but had the best processing speed. Most of the errors for the 5 different approaches occurred in *Severe* complexity cases when multiple players overlap with each other, causing the detector to detect them as a single player. This kind of situation arises when a player approaches close to the opposition team aiming for a shot at the goal.

We currently do not account for any camera calibration and the players are tracked based on the Euclidean coordinates of the video frame and not the actual coordinates of the soccer field. As a result, when the camera changes its focus from one part of the field to another, some players do not appear in the video for a while and when they re-appear, they are detected and tracked as new players. It is very challenging to track re-appearing players if the duration between disappearing and re-appearing is very large. Possible solutions to eliminate this problem are:

Player re-identification: 1) We can a use a player re-identification framework to associate the players when they re-appear, but the challenge with this is that, the players are wearing the jersey of the same color which makes it difficult to re-identify them under all situations. 2) We can associate the soccer players by recognizing their jersey numbers as proposed by Liu *et al.* [71], but this is possible only when the soccer player is displaying the jersey number to the camera.

Additional hardware: 1) We can use multiple static cameras on opposite sides of the soccer field such that the collective FoV of the cameras spans the entire soccer field [84]. 2) We can use unique GPS trackers attached to the jersey of the soccer players along with camera calibration to get the physical location of the players on the soccer field.

3.3.3 Results for the Player Classification Module

Team Classification Results and their Comparisons: In this sub-section we evaluate and compare the performance of our three different team identification algorithms (***TI-1, TI-2, TI-3***) as described in Section 4.2.2. For the *TI-1* approach, we trained the Triplet CNN on the Town Center subset of the PETA dataset [88] and then evaluated the CNN on all of the 49,950 images from our dataset. For the *TI-2* and *TI-3* approaches, we randomly split our dataset consisting of 49,950 images evenly based on the number of images in our dataset for each team into 65% for training, 10% for validation and 25% for testing. The validation dataset was selected randomly and fixed for all of the experiments for team identification. We used the validation dataset only for finding the best hyper parameters. Table 16 shows the distribution of the training, validation and testing datasets and Table 17 shows the results and comparison of the four-fold cross validation for team identification.

TABLE 16: Data distribution for training, validation and testing datasets for team identification.

Team	**Training**	**Validation**	**Testing**
White Team	9,075	1,395	3,490
Red Team	11,303	1,739	4,348
Blue Team	12,090	1,860	4,650
Total Images	32,468	4,994	12,488

TABLE 17: Results and comparison for team identification.

Approach	**Average Accuracy (%)**	**Requires template?**
Theagarajan *et al.* [74]	71.28	yes
TI-1*	83.56	yes
TI-2*	93.17 ± 1.07	yes
Otsu [85]	64.07 ± 5.37	no
ResNet18 [39]	82.55 ± 1.56	no
ResNet34 [39]	89.40 ± 1.62	no
ResNet50 [39]	93.58 ± 1.40	no
AlexNet [28]	85.73 ± 1.44	no
VGG-16 [40]	82.07 ± 1.31	no
TI-3*	**97.46 ± 1.19**	no

*TI stands for Team Identification

In order to evaluate the approach of Theagarajan *et al.* [74], we randomly selected 10 template images from the dataset for each team and evaluated the performance on all of the remaining images from our dataset. We can observe from Table 17 that our approach (*TI-3*) outperforms all of the state-of-the-art methods. Otsu's method [85] had the least accuracy followed by Theagarajan *et al.* [74] because even within the same match if the pose of the detected player is in the profile view and the templates consists of images that are frontal/back view of the player, the histograms will look very different and it is not realistic to collect new templates dynamically during a match.

From Table 17 in the *TI-1* setting, we trained the Siamese CNN on the Town Center subset of the PETA dataset [88] for the task of pedestrian re-identification. After training the CNN, we evaluated it on our dataset by extracting features of the detected player and matching it with the features for the template images. This approach achieved an accuracy of 83.56% and did not require any knowledge of the soccer match making it generalizable to other soccer matches.

The *TI-2* setting is similar to the *TI-1* setting except that after pre-training on the subset of the PETA dataset [88], we further fine tune the Siamese CNN with our soccer dataset which helps to improve the accuracy to 93.17%.

In the *TI-3* setting, we train the Triplet CNN using both the Triplet loss and the binary cross entropy loss. The difference between *TI-1*, *TI-2* and *TI-3* is that *TI-3* does not require any template images for matching the features and it provides the highest accuracy of 97.46%. In *TI-3* we need to train the Triplet CNN for every single match, making it less generalizable compared to *TI-1*. Although the *TI-3* setting is less generalizable compared to the *TI-1* setting, in applications such as sports analytics, coaches and fans of the sport prefer to have a system that provides the highest accuracy compared to having a generalizable system that provides a significantly lower accuracy.

Results for Identifying the Player with the Ball and Comparison of Algorithms: In this sub-section we evaluate and compare the performance of our Triplet CNN for identifying the player controlling the ball using the prediction accuracy and speed of performance during inference as the performance metrics. For this purpose, we split the dataset evenly based on the number of images for the two classes from our dataset into 65% for training, 10% for validation and 25% for testing. Table 18 shows the data distribution for the training, validation and testing datasets and Table 19 shows the results and comparison of the four-fold cross validation for identifying the "*Player with the ball*". Similar to Table 16, the validation dataset was used only for finding the best hyper parameters and not for training.

TABLE 18: Data distribution for training, validation and testing for identifying the player controlling the ball.

Class	**Training**	**Validation**	**Testing**
Player with the ball	8,333	1,281	3,205
Player without the ball	24,135	3,713	9,283
Total Images	32,468	4,994	12,488

TABLE 19: Results of the four-fold cross validation for identifying the player controlling the ball.

Network	**Average accuracy (%)**	**Average speed (FPS)**	**Number of parameters**
ResNet18 [39]	81.70 ± 3.21	8.2 ± 0.7	111.7m
ResNet34 [39]	81.18 ± 3.58	6.3 ± 0.7	212.8m
ResNet50 [39]	82.51 ± 4.76	5.9 ± 0.6	235.1m
AlexNet [28]	79.51 ± 2.07	13.4 ± 0.8	61.1m
VGG-16 [40]	81.29 ± 3.22	7.9 ± 0.6	134.2m
Theagarajan *et al.* [74]	83.47 ± 4.02	7.5 ± 0.7	139.5m
Our Approach	**90.66 ± 2.46**	**26.7 ± 1.2**	**3.7m**

From Table 19 it can be observed that our approach had the highest accuracy and speed of performance compared to the state-of-the-art. Moreover, our CNN performs at least 2x faster with less than 16x the number of parameters than the state-of-the-art. The reason for this is that most of the state-of-the-art CNNs are built for more generalized tasks such as classifying the ImageNet dataset [92] which has more than 1,000 classes requiring more number of parameters and computation time. On the other hand, our approach for generating statistics of soccer players is for a specific and time-critical task which requires a Triplet CNN to extract fine-grained features for achieving higher accuracy with less number of parameters. Moreover, as shown in Theagarajan *et al.* [74] regular CNNs have trouble in detecting minute details such as the soccer ball in low resolution images which is the only feature that distinguishes between a player with and without the ball. By using a Triplet CNN, the CNN is able to learn fine-grained features that help in distinguishing a "*Player with the ball*" and further improves the accuracy.

3.3.4 Generalization Across Different Matches

Results on Generalizability and Comparison with other Algorithms: In this sub-section we evaluate and compare our approach for its generalizability across different matches. Generalizability is a very important metric for determining a classifier's robustness. Many studies have shown that a classifier that is generalizable across multiple domains does not necessarily have the best performance on all of the domains, similarly a classifier that has the best performance in one domain does not necessarily generalize across multiple domains [93, 94].

To evaluate the generalizability we trained the CNNs on all the training images (32,468 images) in our dataset as shown in Table 18 and evaluated them on soccer matches played by different teams that were never included in our dataset. We selected four high school soccer matches where two matches were played by *Pink jersey Vs Black jersey* and two matches were played by *Green jersey Vs Black jersey.* In order to validate the performance, we annotated 100 images per team per match for the two classes "*Player with the ball*" and "*Player without the ball*", resulting in a total of 800 images. Table 20 shows the results and comparison of different CNNs for their generalizability across different soccer matches.

TABLE 20: Results on the generalizability across different matches for identifying the player controlling the ball.

Network	**Accuracy (%)**	
	Pink Vs Black	***Green Vs Black***
ResNet18 [39]	59.31	54.22
ResNet34 [39]	57.60	61.23
ResNet50 [39]	60.44	60.76
AlexNet [28]	55.81	58.29
VGG-16 [40]	**63.36**	61.29
Theagarajan *et al.* [74]	58.35	61.07
Our Approach	60.26	**62.28**

From Table 20 it can be seen that all the CNNs fell short in their performance compared to Table 19. This indicates that the features learned from one match do not necessarily transfer over to another match played by two different teams. This is similar to the findings reported by the authors of [74, 79, 95]. Moreover, it is not feasible to collect data that resembles all the different conditions to train the network, hence, it is more appropriate to re-train the CNNs for every match with as minimal annotation as possible.

Match Specific Annotation for Robust Performance: In this sub-section we use a minimum number of images annotated for specific matches in varying proportions to fine tune the different CNNs and observe the performance. In order to validate the match specific performance, a problem that arises is that how do we annotate images for a match that has not yet been played? To solve this we annotate images of matches previously played by the same teams for training our models and evaluate it on the match that is to be played.

As mentioned in the previous sub-section, we annotated 100 images per team per class (*Player with/without the ball*) from four different matches, where two matches were played by *Pink jersey Vs Black jersey* and two matches were played by *Green jersey Vs Black jersey.* We took the two matches played by the same teams and used one match for training and the other match for testing. Based on this we perform two-fold cross validation. Table 21 and 22 shows the match specific performance for the different CNNs.

TABLE 21: Match specific performance of different CNNs for the game played between *Pink jersey Vs. Black jersey.*

Network	**Average accuracy (%)**		
	50 images per class	**75 images per class**	**100 images per class**
ResNet18 [39]	64.32 ± 2.56	64.88 ± 3.13	65.72 ± 2.01
ResNet34 [39]	65.60 ± 3.97	65.89 ± 3.42	67.22 ± 3.19
ResNet50 [39]	68.23 ± 4.15	68.42 ± 2.91	69.01 ± 2.88
AlexNet [28]	61.98 ± 4.23	62.34 ± 3.85	64.56 ± 3.76
VGG-16 [40]	66.39 ± 2.54	68.25 ± 4.31	69.14 ± 3.67
Theagarajan *et al.* [74]	65.78 ± 3.58	67.41 ± 3.19	70.22 ± 4.29
Our Approach	**84.05 ± 2.56**	**84.73 ± 3.02**	**84.81 ± 2.71**

In columns 2 and 3, we randomly selected 50 and 75 images, respectively, for training.

From Table 21 and 22 it can be seen that, as we fine tune the CNNs on images for a specific match, we can observe an increase in performance. Comparing Table 20 with Table 21 and 22 we can observe an increase in performance across all CNNs, but our approach significantly outperforms the state-of-the-art CNNs. One possible reason for this is that although the training dataset consists of only 100 images per class (*Player with/without the ball*), we can create more than 75,000 Triplet pairs and train the Triplet CNN to learn fine-grained features by increasing the inter-class variance and decreasing the intra-class

TABLE 22: Match specific performance of different CNNs for the game played between *Green jersey Vs. Black jersey.*

Network	**Average accuracy (%)**		
	50 images per class	**75 images per class**	**100 images per class**
ResNet18 [39]	63.02 ± 3.01	64.51 ± 2.78	64.43 ± 3.44
ResNet34 [39]	65.13 ± 3.44	66.58 ± 4.28	66.77 ± 2.95
ResNet50 [39]	66.08 ± 3.37	66.70 ± 3.56	68.35 ± 3.51
AlexNet [28]	63.28 ± 2.80	63.97 ± 3.89	64.89 ± 3.25
VGG-16 [40]	65.41 ± 4.05	66.98 ± 2.19	70.36 ± 2.71
Theagarajan *et al.* [74]	67.88 ± 4.69	67.13 ± 4.11	69.24 ± 3.53
Our Approach	**82.49 ± 3.76**	**84.17 ± 3.04**	**84.68 ± 3.32**

In columns 2 and 3, we randomly selected 50 and 75 images, respectively, for training.

variance. Furthermore, this finding is consistent with the works reported by the authors of [5, 27, 89, 90], wherein Triplet networks are able to outperform regular CNNs in the presence of very limited data.

3.3.5 Ablation Study for Generating the Tactical Statistics

Generating Match Level Tactical Statistics: In this sub-section we perform an ablation study to observe how using different combinations of player detectors and classifiers for identifying the team and player controlling the ball affects the generation of match level statistics. We do not need any tracking algorithm for generating match level statistics, hence we use only the outputs of the player detector and classifier for identifying the team and player controlling the ball. Table 23 and 24 shows the performance and comparison of our approach with the state-of-the-art approaches for generating match level statistics with and without data augmentation. Additionally, there is no other work that can directly provide the tactical statistics for the number of passes and steals from a video, hence, we cannot compare the state-of-the-art approaches in Tables 23 and 24 with our work for these statistics. The accuracy of the ball possession is calculated by identifying the correct player controlling the ball in all of the frames in the videos. Based on this we can observe from Tables 23 and 24 that using our approach for classification outperforms all other approaches for computing the match level ball possession accuracy. Additionally, it is observed that using Mask R-CNN [82] as the detector slightly improves the accuracy for ball possession compared to using YOLOv2 [53].

TABLE 23: Ablation study for comparing the performance of our system for generating the tactical statistics at a match level on the moderate complexity highlight videos.

Detector	**Classifier**	**Avg. ball possession acc. (w/ DA)/w/o DA (%)***	**No. of passes**	**No. of steals**
YOLOv2	**ResNet50** [39]	(73.56 ± 3.19)/72.17 ± 2.44	-	-
	AlexNet [39]	(71.92 ± 3.87)/70.46 ± 4.47	-	-
	Theagarajan *et al.* [74]	(74.82 ± 5.02)/71.17 ± 4.65	-	-
	Our Approach	**(84.19 ± 2.38)/81.83 ± 2.56**	**7/8**	**2/3**
Mask R-CNN	**ResNet50** [39]	(73.78 ± 4.24)/72.55 ± 3.13	-	-
	AlexNet [39]	(72.49 ± 3.17)/70.36 ± 3.20	-	-
	Theagarajan *et al.* [74]	(73.38 ± 3.86)/72.64 ± 3.41	-	-
	Our Approach	**(83.27 ± 2.56)/82.34 ± 2.15**	**7/8**	**2/3**

***w/ DA and w/o DA refer to with Data Augmentation and without Data Augmentation, respectively. Acc. refers to Accuracy.**

• ***Effect of Data Augmentation***: From, the data distribution shown in Table 13, we can observe that the class "*Player without the ball*" has 3x more training data than the class "*Player with the ball*". In order to observe the effect of data augmentation for generating the tactical statistics, we generated and augmented 20,000 synthetic images for the class "*Player with the ball*" to our dataset using our Triplet CNN-DCGAN approach explained

TABLE 24: Ablation study for comparing the performance of our system for generating the tactical statistics at a match level on the severe complexity highlight videos.

Detector	Classifier	Avg. ball possession acc. (w/ DA)/w/o DA (%)*	No. of passes	No. of steals
YOLOv2	**ResNet50** [39]	(68.24 ± 5.11)/65.57 ± 4.74	-	-
	AlexNet [28]	(63.18 ± 3.79)/61.53 ± 4.21	-	-
	Theagarajan *et al.* [74]	(64.28 ± 4.81)/63.17 ± 4.42	-	-
	Our Approach	**(76.65 ± 3.41)/74.06 ± 2.72**	**3/5**	**1/3**
Mask R-CNN	**ResNet50** [39]	(69.83 ± 4.76)/66.02 ± 4.31	-	-
	AlexNet [28]	(64.02 ± 4.16)/63.20 ± 3.77	-	-
	Theagarajan *et al.* [74]	(67.35 ± 4.53)/65.18 ± 4.39	-	-
	Our Approach	**(76.78 ± 3.82)/72.35 ± 2.56**	**3/5**	**1/3**

***w/ DA and w/o DA refer to with Data Augmentation and without Data Augmentation, respectively. Acc. refers to Accuracy.**

in Section 4.2.4. Next, we trained all the classifier approaches in Tables 23 and 24 using the augmented dataset and compared their results without any data augmentation. Based on this we can observe that performing data augmentation helped improve the performance of all approaches in Tables 23 and 24. We were able to improve the performance for our approach by 2.59% and 4.43% using YOLOv2 [53] and Mask R-CNN [82], respectively.

Our approach was able to successfully detect 7/8 passes and 2/3 steals in the *Moderate* complexity and 3/5 passes and 1/3 steals in the *Severe* complexity. There is a drop in performance in the *Severe* complexity because the players are too close to each other and since we are using only one camera, it causes a lot of occlusions. Hence, it is difficult for the network to identify which player is controlling the ball leading to a drop in performance.

Generating Individual Level Tactical Statistics: In this sub-section we perform an ablation study to observe the performance in generating individual player level tactical statistics using different combinations of player detector and tracking algorithms and fixing our approach for predicting the team and player controlling the ball. For this purpose, we selected a 45 second clip from a video (recorded at 30 FPS) of moderate complexity where *Players ID # 6* and *10* belonging to the *white* team were passing the ball between them while *Player ID # 13* belonging to the *red* team was trying to steal the ball. Towards the end of the video *Player ID # 13* successfully stole the ball from *Player ID # 10*. Tables 25, 26, and 27 shows the performance and comparison of our approach with the state-of-the-art approaches for generating individual player statistics. In Tables 25, 26, and 27, the duration of ball possession is shown in frames, this can be converted into time by dividing it by the frame rate of the video. The cells in Duration of possession are formatted as x/y where x is the total number of frames the player was predicted to control the ball and y is the ground truth total number of frames the player was controlling the ball. The cells in the No. of pass/(No. of steal) are formatted as $p/q/(r/s)$ where p and q are the number of predicted and ground-truth passes, and r and s are the number of predicted and ground-truth steals, respectively.

TABLE 25: Ablation study for generating the tactical statistics on an individual level for Player ID: *#6 (White jersey)* from a 45 second video of moderate complexity.

Detector	**Tracker**	**Operating speed (FPS)**	**Duration of possession***	**No. of pass/ (No. of steal)**
YOLOv2	**DeepSort**	16.9	**498/533**	**2/2/(0/0)**
	D'Orazio	20.6	162/533	0/0/(0/0)
Mask R-CNN	**DeepSORT**	1.6	**498/533**	**2/2/(0/0)**
	D'Orazio	1.6	162/533	0/0/(0/0)
	Feichtenhofer	14.4	483/533	1/2/(1/0)

From Tables 25, 26, and 27, we can observe that using DeepSORT achieves better performance in generating the statistics with the highest processing speed using two TITAN X GPUs. Although, Mask R-CNN [82] outperformed YOLOv2 [53] by 3 frames in predicting the duration of ball possession for *Player ID #13*, there is no other significant change. On the contrary using YOLOv2 had the highest processing speed of 16.9 FPS which is a 10x improvement compared to Mask R-CNN. This is significant because, although our approach is offline it is unreasonable for a user to wait 10x longer to analyze a video using Mask R-CNN compared to using YOLOv2 for a very small trade off in accuracy. It can also be observed that the algorithm proposed by Feichtenhofer *et al.* [54] did not detect a pass of *Player ID # 6* and also had a false negative in predicting the steals of *Player ID #10.* The reason for this is that the algorithm had an identity flip for that player during which the pass was made leading to incorrect stats.

TABLE 26: Ablation study for generating the tactical statistics on an individual level for Player ID: *#10 (White jersey)* from a 45 second video of moderate complexity.

Detector	**Tracker**	**Operating speed (FPS)**	**Duration of possession***	**No. of pass/ (No. of steal)**
YOLOv2	**DeepSort**	16.9	**314/371**	**1/1/(0/0)**
	D'Orazio	20.6	107/371	0/0/(0/0)
Mask R-CNN	**DeepSORT**	1.6	**314/371**	**1/1/(0/0)**
	D'Orazio	1.6	106/371	0/0/(0/0)
	Feichtenhofer	14.4	314/371	1/1/(0/0)

3.3.6 Discussion of Results

In this sub-section we analyze the results and provide high level conclusions of the individual modules for generating the tactical statistics.

• ***Player detection and tracking***: In our approach we evaluated various player detection

TABLE 27: Ablation study for generating the tactical statistics on an individual level for Player ID: *#13 (Red jersey)* from a 45 second video of moderate complexity.

Detector	**Tracker**	**Operating speed (FPS)**	**Duration of possession***	**No. of pass/ (No. of steal)**
YOLOv2	**DeepSort**	16.9	**291/337**	**0/0/(1/1)**
	D'Orazio	20.6	82/337	0/0/(0/0)
Mask R-CNN	**DeepSORT**	1.6	**294/337**	**0/0/(1/1)**
	D'Orazio	1.6	85/337	0/0/(0/0)
	Feichtenhofer	14.4	274/337	0/0/(1/1)

and tracking algorithms and found the best combination for detecting and tracking players are YOLOv2 [53] and DeepSORT [56], respectively. Although the Mask R-CNN [82] approach was able to slightly outperform YOLOv2 in terms of IOU, YOLOv2 has a 10x improvement in processing speed. This is a very important trade-off in terms of processing speed. In terms of tracking we observed that deep learning based approaches proposed by [54–56] outperform some of the hand-crafted approaches described in [86]. The reason for this is that the approaches proposed in [86] are not very generalizable across different matches and do not handle player occlusions well.

• ***Team identification***: We proposed three different team identification algorithms (*TI-1*, *TI-2*, and *TI-3*) and found that *TI-3* outperformed all state-of-the-art approaches as shown in Table 17. A drawback of *TI-3* is that we require an annotated dataset for training, making it less generalizable. A solution for this problem is that since team jerseys do not often change, we can choose a match that was played in the past by the same teams and annotate those images for training the CNN. In cases where datasets are not available we can still use *TI-1* which is the most generalizable approach for a slight trade off in performance.

• ***Identifying the player controlling the ball***: We proposed to use a Triplet CNN for identifying the player controlling the ball throughout all the frames in a video. Prior work done by [74] showed that regular CNNs often overlook minute details such as soccer balls which is the most important feature for identifying the player controlling the ball. We empirically showed that by training Triplet CNNs to extract fine-grained features our approach outperforms the state-of-the-art classifiers for this task. A general drawback of all the approaches shown in Table 19 is that, they do not generalize to matches beyond the dataset. Our approach solves this problem by requiring only 100 annotated images per class (*Player with/without the ball*) per match in order to achieve a reasonable performance and it outperforms the other approaches shown in Tables 21 and 22.

3.3.7 Application to Internet of Things

Internet of Things (IOT) is an environment where individual devices sense and collect data which is shared through the internet where the data can be processed and interpreted in real time. This technology has been widely used in areas such remote monitoring, healthcare and recently in sports [96]. In our case the proposed approach can be integrated into a multi-camera system in order to generate more robust statistics and usually this would create a bottleneck problem in terms of processing speed. This kind of problem can be solved by moving heavy computations onto a cloud based IOT-environment as shown in [96]. Additionally, in order to make the tracking more robust, we can attach cheap GPS tracking sensors on the jerseys of the players which transmit the data to a cloud server where all of the data are being collectively processed in real-time.

Chapter 4

ShieldNets: Defending Against Adversarial Attacks Using Probabilistic Adversarial Robustness

Deep learning has demonstrated impressive performance on many important practical problems such as image [28], video [97], audio [98] and text classification [99]. Despite their outstanding performance, it has been recently shown that deep learning models are vulnerable to adversarial manipulation of their input which is intended to cause a mis-classification [100–102]. These attacks are carefully crafted perturbations that are so subtle that a human observer does not even notice the modification at all, but can cause deep learning models to mis-classify the input. Fig. 21(a) shows examples of original images from the CIFAR-10 [103] and Fashion-MNIST [104] testing datasets, (b) shows the adversarial perturbations crafted using attacks discussed in this chapter and (c) shows the adversarially perturbed images being mis-classified using a VGG [105] Convolutional Neural Network.

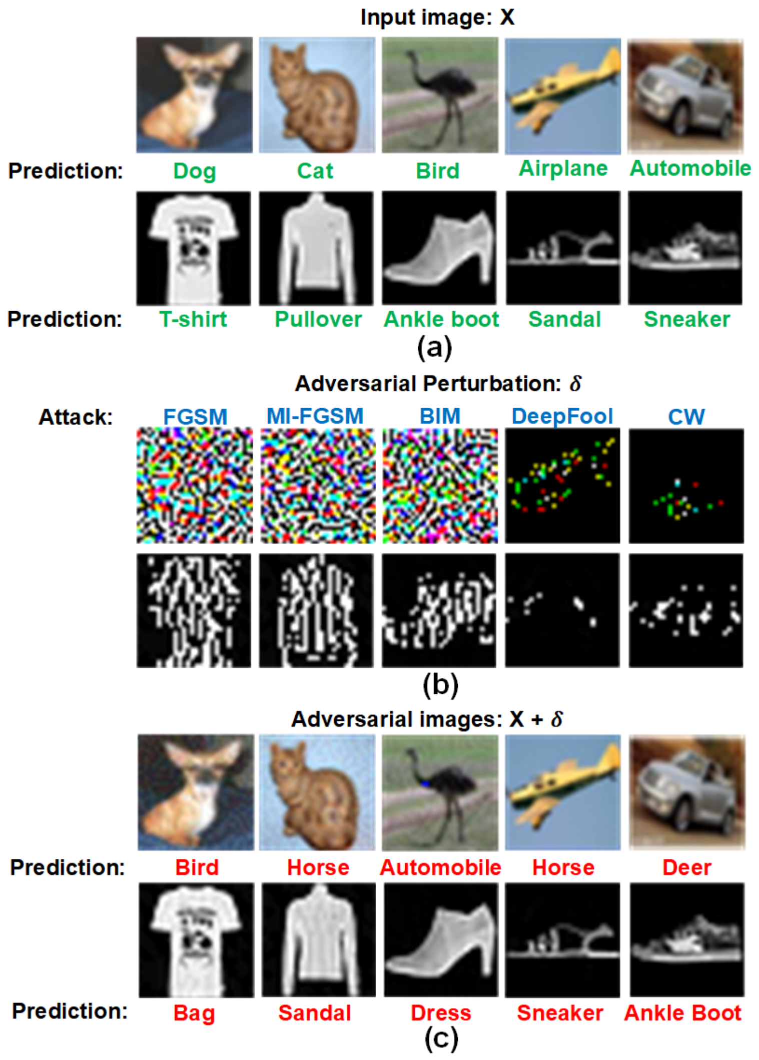

Fig. 21: (a) Examples of original images from the CIFAR-10 and fashion-MNIST datasets correctly classified by VGG, (b) generated perturbation for the corresponding images, (c) corresponding adversarial examples mis-classified by VGG.

Adversarial attacks can be achieved through black-box attacks and white-box attacks. In the black-box attack model [106], the attacker does not have any access to the parameters or architecture of the classification model, whereas in the white-box attack [107], the attacker has complete access to all the parameters and architecture of the classification model. Szegedy *et al.* [101] showed that an adversarial example that was designed to be mis-classified by a model *M1* can also be used to mis-classify a different model *M2*. This adversarial transferability helps bridge the gap between white-box attacks and black-box attacks. Furthermore, Kurakin *et al.* [100] showed that adversarial examples can also exist in the physical world [108]. The authors of [100] created an adversarial perturbation, printed the perturbed image, photographed the printed image and fed it back to the classifier. Their results show that the classifier mis-classified the photographed image, indicating that physical sensors are also prone to adversarial examples. These kind of attacks can provide disastrous results in safety-critical applications such as self-driving cars [109].

To this end we propose Probabilistic Adversarial Robustness (PAR) as a fundamental approach to neutralize adversarial attacks. The underlying concept of PAR is to utilize the application loss functions to guide a probabilistic model for projecting adversarial examples to the adversarial-free zones. In this chapter, we present the theory of PAR and its theoretical possibility to reliably prevent adversarial attacks in a compact region. Moreover, as a demonstration, we select PixelCNN [110] as a specific implementation of PAR's probabilistic model for its state-of-the-art performance in modeling image distributions and tractability of evaluating the data likelihood [111, 112]. The resulted defense network is named as ShieldNet. We train the ShieldNet to learn the adversarial-free zones around the input data distribution of the target CNN, and numerically show that the transformed image does belong closer to the training/testing manifold using statistical p-value tests.

4.1 Related Work on Adversarial Attacks and Defenses and Our Contributions

In this section we explain in detail the related works in two parts. First, we introduce and discuss different adversarial attack strategies employed in this literature. Second, we introduce and discuss existing defense mechanisms.

4.1.1 Adversarial Attacks

For an image-label pair (x, y), adversarial attacks try to find a small perturbation δ with $\parallel \delta \parallel_\infty \leq \epsilon$, such that a classifier $f(\cdot)$ gives $f(x + \delta) \neq y$. ϵ is a hyper-parameter that sets the perturbation limit for each pixel in x on the color scale.

Fast Gradient Sign Method (FGSM): This attack was proposed by Goodfellow *et al.* [113]. The authors generate a malicious perturbation given by

$$x^{adv} = x + \epsilon \cdot sign(\bigtriangledown_x L(x, y)) \tag{1}$$

where $\bigtriangledown_\mathbf{x} L(x, y)$ is the loss function used to train the model and y is the class label. This approach uses the sign of the gradients at every pixel to determine the direction with which to change the corresponding pixel value.

Basic Iterative Method (BIM): This attack was proposed by Kurakin *et al.* [100]. The authors implemented a variant of the FGSM attack by applying it multiple times with a smaller step size. The adversarial examples are formally computed as:

$$x^{adv}_{n+1} = Clip_\epsilon(x_n + \alpha \cdot sign(\bigtriangledown_x L(x_n, y))) \tag{2}$$

$Clip_\epsilon(\cdot)$ clips the resulting image to be within the ϵ-ball of x and α is the iterative step size. Similar to Kurakin *et al.* [100], we set $\alpha = 1$ and limit the number of iterations to be [min(ϵ + 4, 1.25 $\cdot$ ϵ)].

DeepFool: This attack was proposed by Moosavi-Dezfooli *et al.* [114]. The authors construct DeepFool by assuming that Neural Networks are linear, with a hyperplane separating each class. Based on this, they iteratively linearize the decision boundary and find the closest adversarial example. We clip the resulting image so that its perturbation is not larger than ϵ.

Carlini-Wagner (CW): Carlini and Wagner [107] proposed an efficient optimization objective for iteratively finding the adversarial examples with the smallest perturbation leading to high probability of mis-classification. We clip the resulting image so that its perturbation is not larger than ϵ.

Momentum Iterative-FGSM (MI-FGSM): Dong *et al.* [115] proposed integrating a momentum term into an existing iterative attack helps improve the success rate of the attack. The adversarial examples are formally computed as:

$$g_0 = 0,\ x_0^{adv} = x,\ g_{t+1} = \mu \cdot g_t + \frac{\bigtriangledown_x L(x_t^{adv}, y)}{\| \bigtriangledown_x L(x_t^{adv}, y) \|_1},\ x_{t+1}^{adv} = x_t^{adv} + \alpha \cdot sign(g_{t+1}) \quad (3)$$

where, μ is the momentum and $\alpha = \epsilon / T$.

4.1.2 Adversarial Defense

Current defenses against adversarial attacks can be classified into four approaches: 1) modifying the training data, 2) modifying the model, 3) using auxiliary tools, and 4) detecting and rejecting adversarial examples.

1) Modifying the Data:

Adversarial training: The adversarial samples are introduced into the training dataset to improve the robustness of the target. Szegedy *et al.* [101] injected the adversarial samples and modified its labels to make the model more robust in the face of the adversaries. Goodfellow *et al.* [113] reduced the misclassification rate on the MNIST dataset from 89.4% to 17.9% by using adversarial training. Huang *et al.* [128] increased the robustness of the model by punishing misclassified adversarial samples. Tramèr *et al.* [117] proposed ensemble adversarial training which can increase the diversity of adversarial samples. However, it is unrealistic to introduce all unknown attack samples into the adversarial training, which leads to the limitation of adversarial training. Zhang *et al.* [129] using the Pontryagin's Maximal Principle (PMP) showed that adversarial training mainly updates the weights of only the first few layers of the model. It was also shown that this approach was not scalable and an adversarially trained classifier can still be attacked with just slight modifications to the existing attacks [100, 130, 131].

Gradient hiding: A natural defense against gradient based attacks presented in [117] and attacks using adversarial crafting method such as FGSM. This method hides information about model gradient from the adversaries, i.e., if a model is non-differentiable (e.g., a decision tree or a nearest neighbor classifier), the gradient-based attack is invalid. However, by learning the proxy black-box model with gradient and using the adversarial samples generated by this model [106], the method can easily be fooled in this case.

Data compression: Dziugaite *et al.* [132] found that JPG compression method can improve a large number of network model recognition accuracy declined caused by FGSM attack disturbance. Das *et al.* [133] used a similar JPEG compression method to study a defense method against FGSM and DeepFool attacks. However, these image compression technologies still cannot serve as an effective defense against more powerful attacks, such as CW attack [134]. Similarly, the Display Compression Technology (DCT) compression method [133] used in the fight against universal disturbance attacks [**?**] has also been proved to be insufficient. The biggest limitation of these defense methods based on data compression is that a large amount of compression will lead to a decrease in the accuracy of original image classification, while a small amount of compression is often not enough to remove the impact of disturbance.

Data randomization: Xie *et al.* [135] demonstrated that the operation of random resizing adversarial samples can reduce the effectiveness of adversarial samples. Wang *et al.* [136] used a data conversion module separated from the network to eliminate the possible adversarial disturbance in the image.

Image transformations: Guo *et al.* [137] used a combination of random input transformations to counter adversarial examples. The authors used bit-depth reduction, JPEG compression, total variance minimization, and image quilting transformations. The authors reported that using total variance minimization and image quilting proved to be the most robust transformations. Raff *et al.* [138] improved the robustness against adversarial examples by combining a set of random hand crafted input transformations that are unknown to the adversary. They achieved the highest robustness when they transformed the image using ten different randomly chosen transformations. The drawback of these approaches is that, once the adversary knows what these transformations are, they can create adversary that specifically counter them.

Blocking the Transferability: Since the transferability attribute holds even if the classifiers have a different architecture or are trained on the disjoint dataset, the key to preventing the black-box attack is to prevent the transferability of adversarial samples. Hosseini *et al.* [139] proposed a three-step null Labeling method, in order to prevent the adversarial samples from one network to another network. Its main idea is adding a new null label to the dataset, and classify them to null label by training classifier to resist adversarial attacks. The three main steps are: initial training target classifier, computing null probabilities, and adversarial training.

2) Modifying the Model:

Defensive distillation: Papernot *et al.* [119] proposed a defensive distillation method to resist attacks on the basis of distillation technology [140]. Defensive distillation does not change the scale of the model, and produces a model with a smoother output surface and less sensitivity. However, the effectiveness of defense distillation cannot be guaranteed in black-box attacks.

Label Smoothing: This approach proposed by Warde-Farley and Goodfellow [118] converts one-hot labels to soft targets, where the correct class has a value 1 - T while the wrong classes have $T/(N$ - $1)$. Here T is a small constant and N is the number of classes. When the classifier is re-trained on these soft targets rather than one-hot labels it is more robust to adversarial examples. This method is similar to defensive distillation proposed by [119] but is shown to be computationally inexpensive.

Feature squeezing: Feature squeezing is a model enhancement technique [120], whose main idea is to reduce the complexity of the data representation, thereby reducing the adversarial interference due to low sensitivity. There are two heuristic methods, one is to reduce the color depth at the pixel level; the other is using a smooth filter on the image. Although this technique can effectively prevent adversarial attacks, it also reduces the accuracy of the classification of real samples.

Parseval networks: Cisse *et al.* [141] proposed a network called Parseval as a defensive method against adversarial attacks. This network adopts hierarchical regularization by controlling the global Lipschitz constant of the network. They proposed to control the spectral norm of the network weight matrix by parameterizing it through Parseval tight frames, so it was called "Parseval" network.

Thermometer encoding: Buckman *et al.* [123] discretized the input by replacing the pixel values with a binary vector using a thermometer encoding process. The idea here is that by encoding, the threshold effects of discretization makes it harder to find adversarial examples that only make small alterations of the image. A drawback of this approach is that it scales the input space dimension linearly with the number of discretization steps, leading to a significant increase in the number of parameters for the model.

Feature denoising: Xie *et al.* [142] included denosing blocks after the convolution layers while training the CNN models. The authors used combinations of non-local means, bilateral filter, mean filter, and median filter as their denoising blocks. Mustafa *et al.* [143] disentangled the class-wise intermediate feature representation by forcing the learned features to lie inside within a convex polytope that is maximally separated from the polytopes of other classes. The drawback of the these approaches is that they are highly dependent on the ϵ strength of the attack.

Mask Defense: Gao *et al.* [144] proposed to insert a mask layer before processing the classified network model. This mask layer trained the original images and corresponding adversarial samples and encoded the differences between these images and the output features of the previous network model layer. It is generally believed that the most important weight in the additional layer corresponds to the most sensitive feature in the network. Therefore, in the final classification, these features are masked by forcing the additional layers with a primary weight of zero. In this way, the deviation of classification results caused by adversarial samples can be shielded.

Architecture pruning: Madaan *et al.* [146] proposed to sparsify the latent features that are sensitive to adversarial perturbation. They proposed a Bayesian framework to prune features based on their contribution to both the original and adversarial loss. The authors also suggested that regularizing the features learned during training helps improve the robustness. Ye *et al.* [147] observed that adversarially trained models are much more sparse compared to the original models. They performed weight pruning on the adversarially trained model thus, achieving both model compression and more robustness against adversarial attacks.

3) Using Auxiliary Tools:

Defense-GAN: Samangouei *et al.* [122] used a Generative Adversarial Network (GAN) to project input images onto the range of the generator by minimizing the reconstruction error, prior to feeding the image to the classifier. As compared to the adversarial images, the clean images are closer to the range of the generator.

PixelDefend: This approach proposed by Song *et al.* [124] leverages pre-trained probabilistic generative networks to purify an adversarial example to resemble the distribution of the training dataset. Although their approach is model and attack agnostic, its performance decreases as the strength of the attack increases.

ShieldNets: Theagarajan *et al.* [145] extended [124] by using Probabilistic Adversarial Robustness (PAR). PAR is used to neutralize adversarial examples by concentrating the sample probability within adversarial-free zones. They showed the connection between the PAR loss and SGD loss, and prove the existence of an optimal distribution for the probabilistic transformation to reach a theoretical lower bound.

MagNet: This approach proposed by Meng *et al.* [121] uses an auto encoder to learn the distribution of the training data. During testing if the input image is from the real dataset the reconstruction loss will be minimum but if the input is an adversarial example, then the loss will be higher.

Deep Contractive Autoencoder: Gu and Rigazio [148] proposed a variant of autoencoder, to increase the robustness of neural networks. A denoising autoencoder network is trained to encode adversarial examples to the original images to remove adversarial perturbations.

High-Level Representation Guided Denoiser: Liao *et al.* [149] proposed High-Level Representation Guided Denoiser (HGD) as a defense for image classification. Standard denoisers such as [121, 148] suffer from the problem of error amplification, in which an already small adversarial noise is progressively further amplified. HGD is able to alleviate this by using a loss function defined as the difference between the target model's outputs activated by the clean image and corresponding denoised image. This advantage of this approach is that it can be trained on a small subset of the training images and it genaralizes well to other images.

4) Detecting and Rejecting Adversarial Examples:

SafetyNet: Lu *et al.* [150] introduced an approach called SceneProof which detects whether an image is a picture of a real scene or not using RGBD images. SceneProof uses a VGG architecture [105] and compares the RGB image and the depth map to see if the depth map is consistent with the RGB image. It relies on the relative difficulty of producing naturalistic depth maps for images in post processing.

Detection using branched networks: Metzen *et al.* [151] detected adversarial examples by introducing a branched subnetwork to an already trained classifier network. The original dataset and its corresponding adversarial data points are used for training the branched network and the output of the branched network is the probability that the image is from the original distribution or the adversarial distribution. The disadvantage of this approach is two folds: 1) the defense does not generalize well to different attacks and 2) the layer at which the branched network is attached specific for every dataset and different classification networks

Statistical testing: Grosse *et al.* [152] used the kernel based two sample test proposed by Gretton *et al.* [153] to distinguish adversarial examples from the training data. They reported that when the sample size of the adversarial examples is more than 100, the success rate of detecting them is more than 80%, but it drastically reduces for small sample size. To overcome this, they perform an outlier detection by adding an additional class to their model output and training the it to classify the adversarial examples to this class.

Bayesian uncertainty: Feinman *et al.* [154] used the model confidence on adversarial examples by looking at the Bayesian uncertainty metrics from dropout networks. After obtaining the Bayesian uncertainty metrics they perform a kernel density estimation in the feature space of the deeper layers of the network to see if the data points is close to the manifold corresponding to the real images. Rawat *et al.* [155] used three Bayesian uncertainty metrics namely: 1) Model Uncertainty as measured by Mutual Information (MUMMI), 2) Predicted entropy, and 3) Variation ratio for detecting adversarial examples. First, the authors converted regular CNNs into a corresponding Bayesian CNN using the Monte-Carlo dropout approach proposed by Yarin *et al.* [156]. From this they are able to create a Gaussian distribution for every parameter by passing the input multiple times through the network using different dropout ratios. Finally, for a given input they compute the three uncertainty metrics and observed that as the strength of the adversarial attack increases, the value of each uncertainty metric also correspondingly increases.

4.1.3 Contributions of this Chapter

In light of the state-of-the-art, our work is significantly different from the existing approaches.

- We introduce the theoretic framework of Probabilistic Adversarial Robustness (PAR) to neutralize adversarial attacks by concentrating sample probability to adversarial-free zones.

- We theoretically demonstrate the connection between PAR loss and the SGD loss, and prove the existence of an optimal distribution for the probabilistic transformation to reach a theoretical lower bound.
- We empirically show that our approach is generalizable and robust to adversarial transferability of attacks.
- Our approach is model and attack agnostic, and can be combined with other existing approaches which results in even more improved performance.

4.2 Probabilistic Adversarial Robustness (PAR)

In this chapter, we introduce the PAR to provide a theoretical foundation of possibly neutralizing adversarial attack (AA) samples in the compact regions near the "good" samples. The approach of PAR is to seek a random function via a probabilistic model to transform the AA samples to the adversarial-free regions. In the following, we establish the theory of PAR, and provide a demonstration of PAR implementation via PixelCNN [110].

4.2.1 Theory of PAR

For any given image $x \in \mathbb{R}^{M\times N}$ where $M \times N$ is the number of pixels in the image, an ϵ-bounded adversarial sample is denoted as $x + \delta$, where δ belongs to the l_p-bounded neighbourhood $\Delta = \{\delta \in \mathbb{R}^{M\times N} \mid \|\delta\|_p \leq \epsilon\}$ to x. The probabilistic generative model $\pi_\omega(x'|x+\delta)$ in PAR is expected to map the AA samples from adversarial regions back to safer space in Δ. Adversarial attack on any classification task with a loss function of $\mathscr{L}(x', y; \theta)$, where x' is sampled from $\pi_\omega(x'|x+\delta)$ transformation, can be achieved by optimizing,

$$\arg\max_{\delta\in\Delta} \int_\Delta \pi_\omega(x'|x+\delta)\mathscr{L}(x', y; \theta)\mathrm{d}x'. \tag{4}$$

The loss function of PAR can be expressed as the marginalized expectation,

$$\mathscr{L}_{PAR} = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} \mathbb{E}_{x\sim\pi_\omega(\cdot|x+\delta)} \big[\mathscr{L}(x', y; \theta)\big] p(\delta) \mathrm{d}\delta \tag{5}$$

where $p(\delta)$ represents the distribution of adversarial samples in Δ. The theoretical possibility of PAR to neutralize AA samples is supported by the following two theorems.

Theorem 1 *if $\pi_\omega(x'|x+\delta) = \delta_{Dirac}(x'-x)$ for $\forall\ x \sim D$, where $\delta_{Dirac}(\cdot)$ is the Dirac delta function, then $\mathscr{L}_{PAR}$ reduces to SGD loss.*

Proof of Theorem 1: Assume $p(\delta)$ is any distribution that only supports in Δ,

$$L_{PAR} = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} \mathbb{E}_{x'\sim\pi_\omega(x'|x+\delta)} \big[L(x', y; \theta)\big]\ p(\delta)\ \mathrm{d}\delta \tag{6a}$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta)\ \mathrm{d}\delta \int \mathrm{d}x'\ \pi_\omega(x' \mid x+\delta)\ L(x', y; \theta) \tag{6b}$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta)\ \mathrm{d}\delta \int \mathrm{d}x'\ \delta_{Dirac}(x'-x)\ L(x', y; \theta) \tag{6c}$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta)\ \mathrm{d}\delta\ L(x, y; \theta) \tag{6d}$$

$$= \mathop{\mathbb{E}}_{(x,y)\sim D} L(x, y; \theta) \tag{6e}$$

Theorem 2 *Assume $\mathscr{L}(x', y; \theta)$ is continuous in $x+\Delta$ and $\pi_\omega(x'|x+\delta)$ is supported on $x+\Delta$, there exists a lower bound for $\mathscr{L}_{PAR}$ in space Δ. If $\pi_\omega(x'|x+\delta) = \delta_{Dirac}(x'-x-\beta_0)$, $\mathscr{L}_{PAR}$ reaches the lower bound, where $\beta_0 = \arg\min_{\beta\in\Delta} \mathscr{L}(x+\beta, y; \theta)$.*

Proof of Theorem 2: Since $L(x+\beta, y; \theta)$ is continuous and Δ is compact, $\beta_0 =_{\beta\in\Delta} L(x+\beta, y; \theta)$ exists. Assume $p(\delta)$ is any distribution that only supports in Δ and π_ω

supports in $x + \Delta$,

$$\begin{aligned} L_{PAR} &= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} \mathbb{E}_{x' \sim \pi_\omega(x'|x+\delta)} \left[L(x', y; \theta) \right] p(\delta) \, \mathrm{d}\delta & \text{(7a)} \\ &= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta) \, \mathrm{d}\delta \int_{x+\Delta} \mathrm{d}x' \, \pi_\omega(x' \mid x+\delta) \, L(x', y; \theta) & \text{(7b)} \\ &\geq \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta) \, \mathrm{d}\delta \int_{x+\Delta} \mathrm{d}x' \, \pi_\omega(x' \mid x+\delta) \left(\min_{x' \in \Delta} L(x', y; \theta) \right) & \text{(7c)} \\ &= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta) \, \mathrm{d}\delta \int_{x+\Delta} \mathrm{d}x' \, \pi_\omega(x' \mid x+\delta) \, L(x+\beta_0, y; \theta) & \text{(7d)} \\ &= \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} p(\delta) \, \mathrm{d}\delta \, L(x+\beta_0, y; \theta) & \text{(7e)} \\ &= \mathop{\mathbb{E}}_{(x,y)\sim D} L(x+\beta_0, y; \theta) & \text{(7f)} \end{aligned}$$

The equality is satisfied when $\pi_\omega(x'|x+\delta) = \delta_{Dirac}(x' - x - \beta_0)$.

Corollary 3 *If $\mathscr{L}_{PAR}$ reaches the lower bound, adversarial perturbation exist only if $\delta \notin \Delta$.*

In practice, there is no guarantee that the lower bound of $\mathscr{L}_{PAR}$ can be realized. Although adversarial attacks through eq. (7) are possible, the optimization requires SGD and the convergence rate is in the order of $O(1/\lambda)$ [125], which is exponentially slower than the deterministic optimization, where λ is the convergence error.

4.2.2 PAR via PixelCNN

In this chapter we use PixelCNN as the probabilistic model for PAR. $\pi_\omega(x'|x+\delta)$ is a joint probability among all pixels, i.e.

$$Pcnn(x) = \prod_{i}^{M \times N} Pcnn(x_i | x_{1:(i-1)}) \ , \tag{8}$$

where x_i is the i-th pixel of the image. By adopting PixelCNN, it can be factorized into a product of conditional distributions.

$$\pi_\omega(x'|x+\delta) = \prod_{i=1}^{M \times N} p(x_i | [x_1, ..., x_{i-1}], x+\delta) \ . \tag{9}$$

Solving eq. (8) requires a proper definition of space Δ. As a practical solution, we introduce a regularization term $\gamma \cdot \mathrm{Reg}(x', x)$ to constraint how far x' can deviate from x, which implicitly confines the space Δ. Besides, as illustrated by Theorem 1, this regularization also acts as a restraint on $\mathscr{L}_{PAR}$ to be close to the SGD loss without adversarial perturbation. In this work, we use PixelCNN loss as $\mathrm{Reg}(x', x)$. The combined loss function is given by:

$$\mathscr{L}_{imp} = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{\Delta} \mathbb{E}_{x' \sim \pi_\omega(\cdot|x+\delta)} \left[\mathscr{L}(x', y; \theta) + \gamma \cdot \mathrm{Reg}(x', x) \right] \, p(\delta) \, \mathrm{d}\delta \tag{10}$$

where, ω is the parameters in the probabilistic model of PAR. We note that θ, which represents the parameters of protected model, is fixed during the learning of PAR. In all of our approaches for a given input image to the probabilistic model, we sample $n = 10$ number of transformations of x'.

For white-box scenario, the optimization of eq. (2) requires the estimation of $\partial \mathscr{L}_{imp} / \partial \omega$. We utilize PixelCNN++ [111] implementation, which employs mixture models of logistic distributions to represent pixel-wise conditional probability. Through variable transformation of $x' = x'(\omega)$[1] the gradients can be directly evaluated by chain rule.

4.2.3 ShieldNet Implementation

Fig. 22 shows the overview of the implementation of PAR as ShieldNet. ShieldNet consists of three major components: the probabilistic transformation model via PixelCNN, the target CNN classifier, and the optional average for logits. The inputs to ShieldNet are samples potentially with adversarial perturbations $x + \delta$. PixelCNN in Fig. 16 is a probabilistic model that generates n different neutralized samples $(x'^{n}_{i=1})$ for the provided AA sample. then given as input to the original target CNN classifier and the average of the n logits is taken for deciding the final prediction y.

[1]In this work, mean and standard deviation of logistic distribution with respect to ω are optimized.

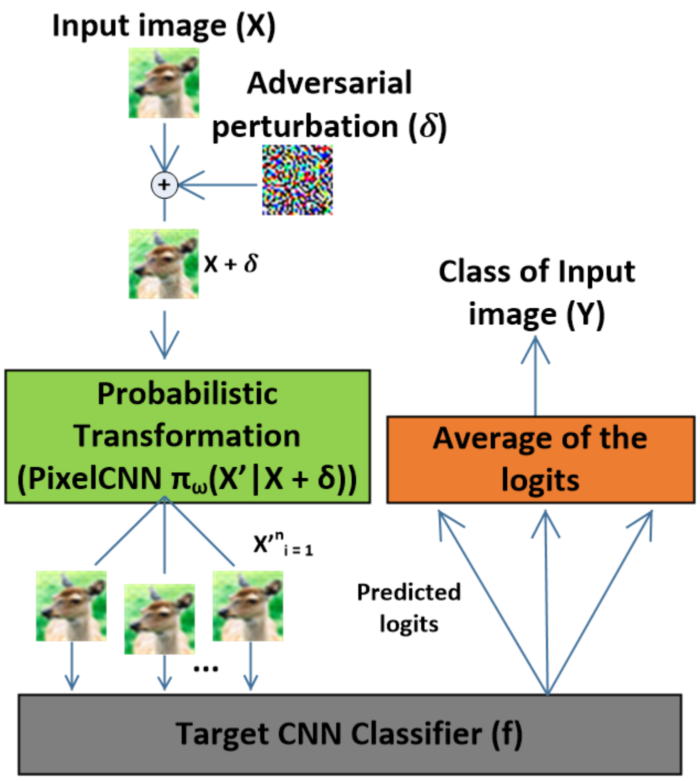
Input image (X)
Adversarial perturbation (δ)
X + δ
Class of Input image (Y)
Probabilistic Transformation (PixelCNN πω(X'|X + δ))
Average of the logits
X'ⁿᵢ₌₁
Predicted logits
...
Target CNN Classifier (f)

Fig. 22: Implementation of PAR: ShieldNet.

4.3 Experimental Results

4.3.1 Datasets and Target CNN Models

We evaluated our approach on two publicly available datasets namely: Fashion-MNIST [104] and CIFAR-10 [103]. Fashion-MNIST was designed to be a much more difficult and drop-in replacement for the MNIST dataset [126]. The dataset consists of 60,000 training and 10,000 testing gray-scale images of size 28x28 distributed evenly into 10 different classes. CIFAR-10 is another widely used dataset that consists of 50,000 training and 10,000 testing RGB images of size 32x32 distributed evenly into 10 different classes.

We evaluated our proposed approach on two state-of-the-art classifiers: ResNet and VGG. For fair comparison we use the same architectures used by [124]. Before training the agent, the two CNNs: ResNet and VGG, were pre-trained on the CIFAR-10 and Fashion-MNIST datasets, and after pre-training the parameters are fixed and not updated. In principle, we could train both the agent and the CNN jointly but, this is not our desired task, as our aim is to train an agent that can defend a CNN without changing the architecture or re-training the CNN. Table 16 shows the classification results of ResNet and VGG on the original Fashion-MNIST and CIFAR-10 testing datasets.

TABLE 16: Classification accuracy of ResNet and VGG on the Fashion-MNIST and CIFAR-10 testing datasets

Network	**Fashion-MNIST**	**CIFAR-10**
ResNet	93.51%	95.31%
VGG	93.05%	92.53%

4.3.2 Neutralizing Adversarial Examples

It has been shown by [124] that adversarial examples have lower probability densities compared to the original training/testing images. Most classifiers suffer from a covariate shift due to the lack of adversarial instances for training leading to mis-classifications.

Similar to [124], we empirically verify this hypothesis by training the PixelCNN model on the CIFAR-10 dataset and then use its log-likelihood estimate combined with a p-value test to detect if an input image is from the original distribution or from the low probability density adversarial space. Let us assume the adversarial input to the PixelCNN model belongs to a distribution $q(x)$ while the original images belong to $p(x)$.

Pseudo code for p-value estimation

Assumptions: If the adversarial distribution is same as the training/testing distribution, then the null hypothesis H_0 is given by $q(X) = p(X)$. The alternate hypothesis H_1 is given by $q(X) \neq p(X)$

Input to PixelCNN: perturbed image $\mathbf{X} + \delta$

Output: p-value of perturbed image

- Compute the output probability of the perturbed image as $\text{Pcnn}(\mathbf{X} + \delta)$
- Compute the output probabilities of the original images in the dataset as $\{\text{Pcnn}(\mathbf{X_1}), ..., \text{Pcnn}(\mathbf{X_N})\}$
- Compute the p-value P given by: $P = \frac{1}{N+1} \sum_{i=1}^{N} \mathbb{I}[Pcnn(\mathbf{X_i}) \leq Pcnn(\mathbf{X} + \delta)] + 1$ where, $\mathbb{I}[.] = 1$,

if the condition in the bracket is true, otherwise it is 0.

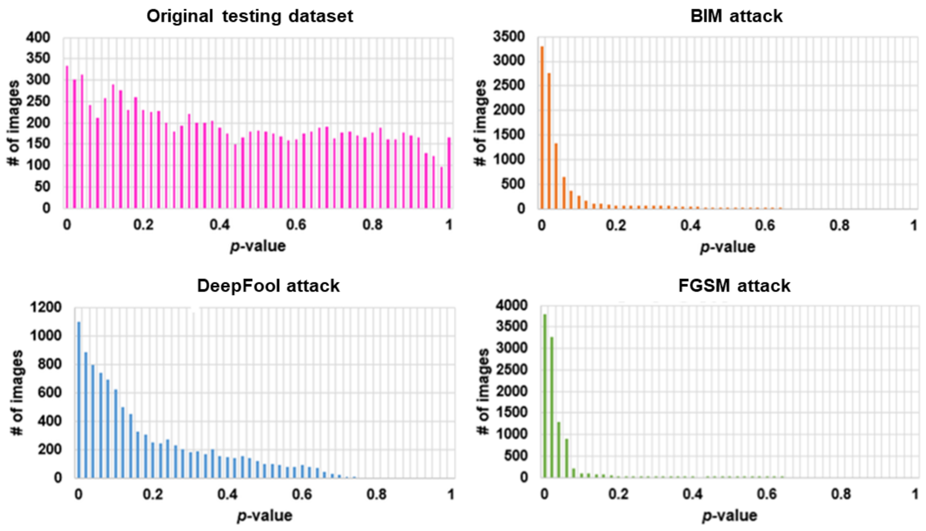

Fig. 23: p-values of the original testing dataset of CIFAR-10 and the adversarial attacks on the testing dataset of CIFAR-10 with $\epsilon = 8$.

Fig. 23 shows the p-values of the original testing dataset of CIFAR-10 and p-values of state-of-the-art adversarial attacks with $\epsilon = 8$. It can be observed that the original testing images have a more uniform p-value distribution compared to the adversarial attacks which significantly deviate from a uniform distribution. This proves that the distribution space

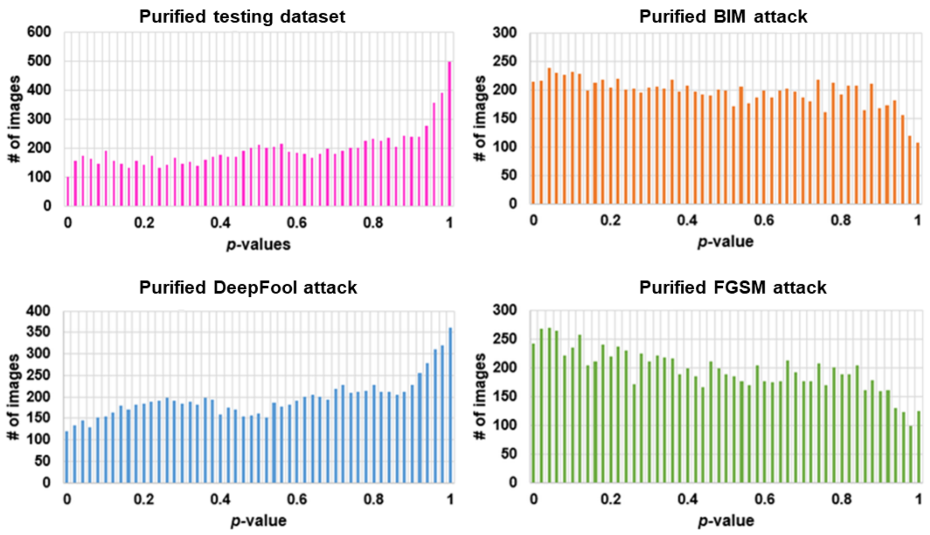

Fig. 24: p-values of the neutralized images after transformation using ShieldNet on the testing dataset of CIFAR-10 with $\epsilon = 8$.

of the original testing images is different from the adversarial distribution space proving the alternate hypothesis H_1. Fig. 24 shows the p-values of the corresponding images in Fig. 23 after being transformed by our approach. It can be observed that the transformed adversarial images have a much more uniform p-value distribution similar to the p-value distribution of the original testing dataset. In Fig. 24 after the neutralization, DeepFool and CW attacks have distributions very similar to the original testing images compared to FGSM, BIM and MI-FGSM. The reason for this is that DeepFool and CW attacks are designed to linearize the decision boundaries between classes which result in perturbations that are small enough just to fool the classifier compared to FGSM, BIM and MI-FGSM that create larger perturbations as shown in Fig. 21.

4.3.3 ShieldNet Defending Intra-attack

In this sub-section we evaluate the performance of individual ShieldNet defending against the same attacking schemes as the ones used in training. The evaluations cover the state-of-the-art attacking algorithms including FGSM, BIM, DeepFool, CW and MI-FGSM for ResNet and VGG on both Fashion-MNIST and CIFAR-10 datasets, as shown in Table 17 and 18. For fair comparison on both datasets, we use the same ϵ used by [124] for evaluating and comparing our approach. In Table 17 the evaluation on Fashion-MNIST dataset utilizes $\epsilon = 8$ and 25, where the CIFAR-10 experiments in Table 18 apply $\epsilon = 2$ and 16. The cells in Table 17 and 18, as well as all following tables in this chapter, are formatted as x/y where x and y are the accuracy for the smaller/larger ϵ.

Table 17 and 18 show that our approach in general outperforms other defending algorithms listed in the tables. For example, on Fashion-MNIST dataset, our approach outperforms PixelDefend in FGSM attack and achieves an accuracy of **89.04%** and **88.59%** against the strongest attack for ResNet and VGG respectively whereas, PixelDefend achieves 74% and 82% respectively. Although there is a drop in performance as the strength of the attack increases it does not significantly drop as compared to PixelDefend. On CIFAR-10 dataset, PixelDefend slightly outperforms our approach in defending VGG against FGSM and BIM when $\epsilon = 2$ but, as ϵ increases from 2 to 16, the performance of PixelDefend drops down drastically. When protecting ResNet against FGSM and BIM attacks with $\epsilon = 16$, PixelDefend only achieves 24% and 25% while our approach achieves an accuracy of **70.52%** and **68.86%** respectively. Moreover, by combining our approach with adversarial training using FGSM we observe overall increases in accuracy as shown in the bottom rows in Table 17 and 18.

It should be noted that in general as the strength of the attack increases the performance of defense algorithms tends to decrease, but these perturbations become more clearly visible even to a human observer and can easily be detected and filtered out using statistical p-value tests.

TABLE 17: Performance comparison of ShieldNet and other defense algorithms on the Fashion-MNIST testing dataset. The highest accuracy is indicated in bold + italic and the second highest accuracy is indicated in bold.

Fashion-MNIST ϵ = 8, 25						
Network	**Training Technique**	**FGSM**	**BIM**	**DeepFool**	**CW**	**MI-FGSM**
ResNet	**Label Smoothing**	64.23/ 36.81	9.76/ 0.00	22.42/ 3.37	20.77/ 4.61	4.25/ 0.00
	Adversarial FGSM	82.49/ 78.43	44.34/ 6.46	57.28/ 11.92	51.03/ 15.70	39.72/ 0.00
	PixelDefend	85.00/ 74.00	83.00/ 76.00	87.00/ 87.00	87.00/ 87.00	NA
	Our Approach	**91.59/ 89.04**	**91.17/ 89.74**	**92.62/ 90.28**	**92.66/ 90.78**	**90.63/ 90.47**
	Our Approach + Adversarial FGSM	***92.46/ 90.35***	***91.93/ 90.68***	***92.88/ 91.36***	***93.47/ 91.61***	***91.45/ 90.59***
VGG	**Adversarial FGSM**	84.55/ 76.21	56.39/ 22.74	37.48/ 18.71	30.69/ 12.52	28.72/ 10.11
	PixelDefend	87.00/ 82.00	85.00/ 83.00	88.00/ 88.00	88.00/ 88.00	NA
	Our Approach	**89.04/ 88.59**	**90.78/ 87.59**	**90.11/ 90.29**	**90.56/ 90.33**	**90.49/ 89.81**
	Our Approach + Adversarial FGSM	***91.55/ 88.72***	***91.37/ 90.15***	***91.02/ 90.77***	***91.27/ 90.76***	***90.95/ 90.56***

4.3.4 Generalization Across Different Attacks

In this sub-section we demonstrate the generalizability of ShieldNet against different attack schemes. It has been shown that adversarial training does not generalize across different attacking schemes. As an example, from Table 17 and 18, it can be observed that

TABLE 18: Performance comparison of ShieldNet and other defense algorithms on the CIFAR-10 testing dataset. The highest accuracy is indicated in bold + italic and the second highest accuracy is indicated in bold.

CIFAR-10 ϵ = 2, 16						
Network	**Training Technique**	**FGSM**	**BIM**	**DeepFool**	**CW**	**MI-FGSM**
ResNet	**Label Smoothing**	64.57/ 14.78	43.28/ 2.92	53.45/ 20.56	50.78/ 14.37	32.91/ 6.73
	Adversarial FGSM	***83.47/ 79.13***	34.58/ 6.73	39.22/ 8.76	28.47/ 5.38	26.94/ 2.33
	PixelDefend	73.00/ 24.00	71.00/ 25.00	80.00/ 80.00	78.00/ 78.00	NA
	Our Approach	76.57/ 70.52	**73.13/ 68.86**	**83.47/ 82.34**	**80.71/ 80.43**	**75.81/ 70.42**
	Our Approach + Adversarial FGSM	**81.29/ 72.61**	***75.59/ 69.84***	***84.73/ 84.08***	***82.91/ 80.86***	***78.44/ 71.27***
VGG	**Adversarial FGSM**	79.28/ 71.59	39.88/ 2.96	28.49/ 2.60	34.27/ 5.39	30.06/ 3.18
	PixelDefend	**80.00**/ 52.00	***80.00***/ 48.00	81.00/ 76.00	81.00/ 79.00	NA
	Our Approach	78.61/ **68.25**	75.32/ **67.34**	**83.19/ 76.20**	**83.26/ 79.11**	**73.92/ 70.43**
	Our Approach + Adversarial FGSM	***81.34/ 70.61***	**77.58/ 70.13**	***88.42/ 79.35***	***83.82/ 80.79***	***75.69/ 71.98***

adversarial training with FGSM examples is able to defend against the basic FGSM attacks, but fails to defend against other attacks. This finding is consistent with the results obtained by [124] and [127].

TABLE 19: Cross evaluation of adversarial attacks on the Fashion-MNIST dataset using ResNet.

Generalization of the ShieldNet + ResNet on Fashion-MNIST with ϵ = 8, 25					
Attack used for training	**FGSM**	**BIM**	**DeepFool**	**CW**	**MI-FGSM**
FGSM	***91.59/89.04***	**88.58/88.09**	86.43/**84.39**	84.22/83.05	83.79/**82.01**
BIM	**88.57/86.18**	***91.17/89.74***	**88.46**/83.99	**86.67/85.43**	**84.25**/81.22
DeepFool	81.39/78.21	83.47/81.26	***92.62/90.28***	85.91/84.04	83.19/81.48
CW	80.11/78.56	86.39/82.11	87.48/83.72	***92.66/90.78***	82.44/79.49
MI-FGSM	82.83/79.34	83.41/81.93	86.90/81.38	84.31/82.01	***90.63/90.47***

TABLE 20: Cross evaluation of adversarial attacks on the CIFAR-10 dataset using ResNet.

Generalization of ShieldNet + ResNet on CIFAR-10 with ϵ = 2, 16					
Attack used for training	**FGSM**	**BIM**	**DeepFool**	**CW**	**MI-FGSM**
FGSM	***76.57/70.52***	**71.56**/66.82	68.96/63.51	65.23/61.40	66.02/59.24
BIM	**70.44/68.52**	***73.13/68.86***	70.38/68.44	68.87/67.49	**71.37/68.94**
DeepFool	66.97/63.21	68.55/60.13	***83.47/82.34***	**78.37/77.40**	66.54/61.10
CW	70.91/64.11	65.22/60.98	**73.50/71.25**	***80.71/80.43***	63.17/61.45
MI-FGSM	66.10/61.32	68.90/**68.27**	71.44/66.71	71.77/70.84	***75.81/70.42***

Table 19 and 20 demonstrate that ShieldNet is able to generalize for both the training and other attacking schemes. As an example, ShieldNet trained with FGSM samples achieves an accuracy of 89.04%, 88.09%, 84.39%, 83.05%, 82.01% accuracy against FGSM, BIM, DeepFool, CW and MI-FGSM attacks respectively. We believe the reason that ShieldNet generalizes across different attacks is that, by using a PixelCNN model in PAR, the model learns to make small changes on the individual pixels that can move the perturbed image back to an adversarial-free zone around the training/testing data distribution. In Table 19 and 20 training our approach using the BIM attack had the best overall accuracy and generalization across different attacks. From Table 20 it should be noted that training on DeepFool is only able to successfully defend against CW attacks and *vice-versa* and falls short across the other attacks compared to BIM. This is because, DeepFool and CW attacks are designed to create higher order perturbations by directly linearizing the decision boundaries of the CNN, whereas, iterative attacks such as BIM create perturbations based on the sign of the gradient at every pixel irrespective of the decision boundary of the CNN. This means that DeepFool and CW attacks are highly dependent on the individual CNN indicating that they have less adversarial transferability.

4.3.5 Robustness against Adversarial Transferability

From a security perspective, an important property of adversarial examples is that they tend to transfer from one model to another, enabling an attacker to create adversarial examples from a source model $M1$ and then deploy those adversarial examples to fool a target model $M2$. To evaluate our approach against this property, we created adversarial examples that fooled the source CNN (ResNet/VGG) and used those adversarial examples to evaluate the performance on the target CNN (VGG/ResNet).

Table 21 and 22 shows the robustness of our approach against adversarial transferability. From Table 21 and 22, it can be observed that the adversarial transferability property of FGSM and MI-FGSM was the highest followed by BIM. DeepFool and CW attacks had the least adversarial transferability which is evident from the fact that, these attacks are designed to linearize decision boundaries of the CNN. Since ResNet and VGG were trained independent to each other, they do not have the same decision boundaries thus making the adversarial examples from DeepFool and CW less transferable. As the adversarial transferability of DeepFool and CW are relatively low, the accuracy of ShieldNet does not vary too much. However, for FGSM, BIM and MI-FGSM attacks which have relatively

TABLE 21: Evaluation of our approach against adversarial transferability on the CIFAR-10 testing dataset.

Adversarial transferability $\epsilon = 2, 16$						
Source CNN	**Target CNN**	**FGSM**	**BIM**	**DeepFool**	**CW**	**MI-FGSM**
ResNet	**VGG**	51.61/ 20.77	**72.45/** 41.56	83.24/ **82.56**	**81.02/ 77.29**	44.37/ 19.86
	ShieldNet + VGG	**78.62/ 72.60**	71.43/ **66.03**	**83.51**/ 79.95	79.91/ 76.53	**73.29/ 70.17**
VGG	**ResNet**	53.81/ 29.14	58.44/ 27.67	79.41/ **82.46**	81.93/ 80.85	51.45/ 18.26
	ShieldNet + ResNet	**79.14/ 70.82**	**70.43/ 68.18**	**80.94**/ 79.81	81.34/ **81.37**	**71.66/ 68.54**

TABLE 22: Evaluation of our approach against adversarial transferability on the Fashion-MNIST testing dataset.

Adversarial transferability $\epsilon = 8, 25$						
Source CNN	**Target CNN**	**FGSM**	**BIM**	**DeepFool**	**CW**	**MI-FGSM**
ResNet	**VGG**	66.41/ 26.37	70.39/ 37.25	86.41/ 87.26	81.92/ 82.43	45.22/ 22.19
	ShieldNet + VGG	**85.95/ 82.40**	**81.33/ 76.38**	**90.29/ 89.26**	**87.31/ 84.53**	**82.60/ 80.41**
VGG	**ResNet**	59.27/ 35.91	72.83/ 40.90	83.27/ 80.18	80.66/ 78.51	64.92/ 38.36
	ShieldNet + ResNet	**84.32/ 80.17**	**83.50/ 84.26**	**88.74/ 86.71**	**85.70/ 86.65**	**88.67/ 81.51**

high adversarial transferability, as shown in Table 21 and 22, our approach is able to defend against the adversarial transferability property by improving the accuracy from 20.77% to **72.60%** for ResNet and from 29.14% to **70.82%** for VGG against the strongest FGSM attack on the CIFAR-10 testing dataset. Similarly, the performance is improved from 26.37% to **82.40%** for ResNet and from 35.91% to **80.17%** for VGG against the strongest FGSM attack on the Fashion-MNIST testing dataset.

Chapter 5

Defending Black Box Classifiers Against On-line Adversarial Attacks

Although, deep learning has had astounding success on several image classification tasks, it has been shown to be vulnerable to adversarial attacks [101, 113, 131]. These attacks are carefully crafted perturbations, added to an image that are visually imperceptible to the human eye, and they can cause deep learning models to misclassify the image with high confidence. In the domain of adversarial attacks, there are two types of threat models: 1) white box, and 2) black box attacks. In the white box setting, the attacker has full knowledge about the classification model's parameters and architecture, whereas in the black box the attack does not have this knowledge. In this chapter we focus only on the black box based attacks. The drawbacks of the approaches discusses in the related works in Section 4.1.2 are that, they do not quantify how much adversarial component is left in the resulting purified image and these approaches are single-step defenses. Here, single-step defense refers to defenses that purify the image only once and assume that the purified image is void of all adversarial components. These defenses are trained on some annotated datasets and after achieving reasonable performance on the annotated dataset, they are deployed to defend real world applications. Once deployed it is assumed that all of the purified images are no longer adversarial, but in reality this is not the case. This is a foundational problem for online/safety-critical applications as shown in [187] - [191] where it is not possible to know the annotations of all of the incoming input images, and these defenses do not have the ability to determine by themselves whether if the purified image is adversarial or not which could further cause disastrous results. To address this problem, this paper proposes a general framework for defending black box classifiers from adversarial attacks using an ensemble of iterative adversarial defenses whose performance is continuously validated in a loop using Bayesian uncertainties. This paper also proposes three novel knowledge distillation approaches for transferring the functionality of the black box classifier into our defense and experimental results on six different datasets shows that our defense can be applied to defend various black box applications ranging from the general Fashion-MNIST [104] and CIFAR-10 [103] datasets to face biometrics and classification for autonomous driving. In summary, the contributions of this paper are as follows:

- To the best of our knowledge prior to [192], this is the first approach that defends against adversarial attacks using an ensemble of iterative adversarial defenses and can convert any single-step black box adversarial defense into an iterative adversarial defense.
- We theoretically and empirically prove that, there exist a lower bound on the amount of purification done to an image, beyond which an image can no longer be purified.
- This paper proposes three novel knowledge distillation approaches that exploit prior meta-information of the training dataset in order to transfer the functionality of the black box classifier into our defense and does not require any information such as the logits probabilities or *Teacher model* architecture.

- Exhaustive evaluation on six public benchmark datasets shows that are approach is able to consistently purify/reject adversarial examples and ablation studies show that it is computationally expensive to break the defense of our framework compared to stand-alone defenses.

5.1 Related Works

In this section we describe state-of-the-art black box adversarial defenses and knowledge distillation approaches and contrast them with our approach. Table 23 shows a summary of the related work.

5.2 Technical Approach

Fig. 24 shows the overall framework of our approach. The input image (X) first passes through the Bayesian CNN and if the image is classified as an adversarial image, it is purified by the ensemble of independently trained iterative adversarial defenses. The purified image of each defense is averaged resulting in the average purified image (X'_i). Here i refers to the current iteration number. Next, X'_i is passed as input back to the Bayesian CNN. If X'_i is not adversarial, it is passed as input to the Black box classifier for final classification, else, it is again passed as input to the ensemble of adversarial defenses and this continues for M iterations. After M iterations, if X'_M is still adversarial then the image is rejected. In Fig. 24, we chose to take the average of the purified image of each individual defense as this further helps in removing high frequency adversarial perturbations [197, 198] that may have not been purified by some of the individual defenses. Additionally, the number of iterations of purification M is chosen empirically depending on the training dataset used (see Section III D1) and we prove the existence of a theoretical lower bound beyond which an image cannot be further purified (see Section III D2).

TABLE 23: Summary of the related work for black box adversarial defenses and knowledge distillation

Authors	**Def/KD***	**Comments**
Das *et al.* [133]	Def	Used JPG compression to suppress adversarial perturbations
Raff *et al.* [138]	Def	Performed N number of random transformations
Goswami *et al.* [157]	Def	Performed selective dropout on convolutional filters
Samangouei *et al.* [122]	Def	Defense-GAN: Used Generative Adversarial Networks to map the adversarial image to the closest image within the latent space
Song *et al.* [124]	Def	PixelDefend: Used PixelCNN to remap adversarial images to the distribution of the training dataset
Meng *et al.* [121]	Def	MagNet: Used the reconstruction loss of autoencoder to detect and purify adversarial examples
Theagarajan *et al.* [145]	Def	ShieldNets: Used Probabilistic Adversarial Robustness (PAR) to purify adversarial images
Grosse *et al.* [152]	Def	Used kernel based two sample test to distinguish between adversarial and original images
Rawat *et al.* [155]	Def	Used Bayesian CNN to detect adversarial attacks
Orekondy *et al.* [180]	KD	Used the softmax probability scores and cross dataset images to distill knowledge
Furlanello *et al.* [193]	KD	Decomposed the predictions of the teacher model into incorrect prediction and ground-truth information
Frosst and Hinton [194]	KD	Distilled deep networks to decision trees

5.2.1 Assumptions and Target Applications of our Defense

• ***Assumptions of our Defense***: **1)** The output of the Black box model is the top predicted class without any probabilities. **2)** The architecture, parameters and *entire* training dataset of the Black box are not known to both the adversary and the defense algorithm.

Ba and Caruana [195]	KD	Distilled knowledge using the l_2 norm between the logits of the Teacher and Student model
Shin *et al.* [196]	KD	Used KD to minimize forgetting in continuous learning
This paper	**Def**	**This paper proposes a novel defense framework that uses an ensemble of adversarial defenses to iteratively purify adversarial images and shows the relationship between the adversarial image and its corresponding purified image and proves the existence of a theoretical lower bound beyond which the image cannot be further purified**
	KD	**This paper proposes three knowledge distillation approaches that exploit the prior meta-information of the training datasets and shows that knowledge distillation can still be performed even when we do not have any knowledge such as the logit probabilities or information about the black box classifier**

*Def and KD refer to black box adversarial Defense and Knowledge Distillation, respectively

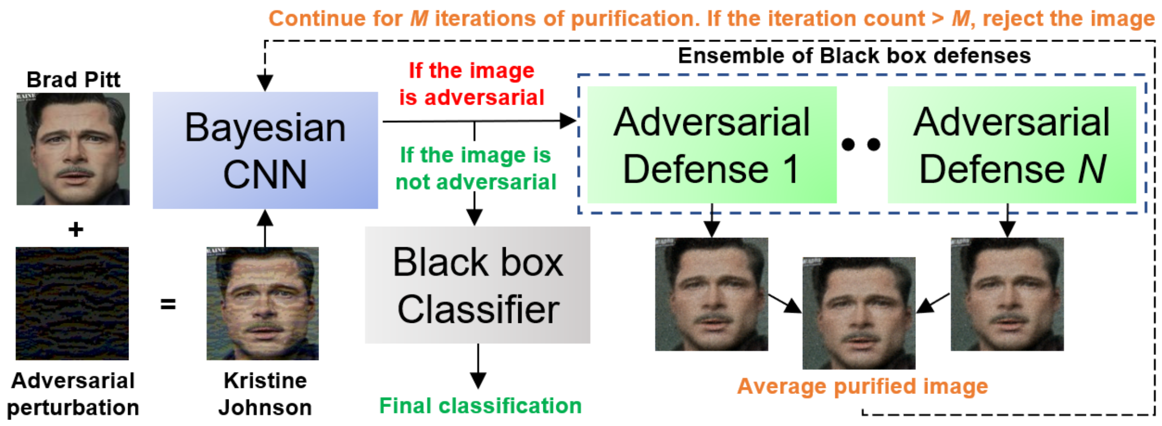
Continue for *M* iterations of purification. If the iteration count > *M*, reject the image
Ensemble of Black box defenses
Brad Pitt
+
Adversarial perturbation
=
Kristine Johnson
Bayesian CNN
If the image is adversarial
If the image is not adversarial
Black box Classifier
Final classification
Adversarial Defense 1
Adversarial Defense *N*
Average purified image

Fig. 24: Overall framework of our approach.

3) The outputs of the Bayesian framework and ensemble of adversarial image purifiers are not shared with the adversary (i.e., the adversary can only see the final classification of the

Black box). Although this is a strong assumption, we perform ablation studies to observe the computational complexity required to break the defense, when the adversary knows partial information of our defense (see Section IV G).

• ***Target Applications for our Defense***: Our defense is suited for Black box applications that contain sensitive personnel information such as human biometrics, remote monitoring, autonomous driving, etc. These are critical applications that require security against adversarial attacks and preserve user privacy and do not want to give up sensitive information.

5.2.2 Functionality Transfer Using Knowledge Distillation

According to our assumptions described in Section III A, we do not know any information about the target black box model's architecture, parameters, and training dataset, hence we cannot directly use our defense algorithm. To solve this problem, we transfer the functionality of the black box model to a substitute model using Knowledge Distillation [178], [193],- [180]. In this paper we assume that the target black box classifier is deterministic (weights are fixed after training) and ignore approaches that continuously update the weights of the CNN (i.e., incremental [199] and reinforcement learning [200]).

Based on this we propose three novel Knowledge Distillation (KD) approaches namely: KD-1, KD-2, and KD-3 that exploit the meta-information of the training datasets. Unlike the approaches proposed in Table 23 our knowledge distribution approach is different because the approaches in Table 23 assume that they have prior knowledge about the training dataset or the logit probability outputs/architecture of the target black box classifier. Whereas, in our approach we assume a strict black box classifier where we have no knowledge about the architecture/logit probabilities of the black box classifier and the only observable output is the single top predicted class.

Assumptions of our KD approaches:

• ***KD-1***: We know the name, total number of the classes in the training dataset and $X\%$ of images from each class.

• ***KD-2***: We know only the name, total number of classes in the training dataset and the domain the dataset belongs to.

• ***KD-3***: We know only the total number of classes in the training dataset and the domain the dataset belongs to.

In KD-2 and KD-3, domain refers to the application of the black box classifier, e.g., classifiers trained on the MS-Celeb [173] and MIO-TCD [201] datasets generally belong to the domain of face recognition and vehicle classification.

KD-1: Here we assume that we have access to $X\%$ amount of images of each class from the training dataset used to train the black box classifier. For simplicity we refer to the black box classifier as the *Teacher model* and our substitute classifier as the *Student model*. Next, we randomly select $X\%$ of images from each class of the dataset and use these images to train a Deep Convolutional Generative Adversarial Network (DCGAN) [42]. After training the DCGAN, we probe the *Teacher model* with the generated images and label them as the predicted class. We then augment these labeled images to the $X\%$ amount of original images to create a pseudo-labeled dataset. In order to have an equal data distribution between the original and pseudo-labeled dataset, we made the number of images per class in the pseudo-labeled dataset to be the same as the original dataset used for training the *Teacher model*. Finally, the pseudo-labeled dataset is used for training the *Student model* (i.e., our substitute classifier).

In KD-1, since we randomly select $X\%$ of images from each class to train the DCGAN and create the pseudo-labeled dataset, there is no guarantee that the selected images statistically represent an accurate distribution of each class. To address this issue we perform k-fold cross validation by selecting different folds of $X\%$ amount of images (see

Section IV D). In our approach we set the value of X to be 25% and 50% and use the Fashion-MNIST [104], CIFAR-10 [103], and GTSRB [202] datasets to evaluate KD-1.

KD-2: In KD-2, we do not have any knowledge about the images or their data distribution, but we know the names and total number of classes in the dataset. Hence, we search for images belonging to those classes from publicly available datasets and the internet. First, we create a pseudo-labeled dataset by probing the *Teacher model* with the images from the public domain and label these images with the predicted class. It should be noted that since we already know the names of the class the image belongs to, we do not need to further probe the *Teacher model* and re-label the images, but prior work done by [203] - [205] shows that when we train the *Student model* with images that were manually annotated by humans, the classification accuracy with respect to the *Teacher model* is lower compared to training the *Student model* with images that were annotated entirely by the *Teacher model*. The reason for this is that images that are misclassified by the *Teacher model* add a regularizing effect while training the *Student model* thus resulting in efficient functionality transfer [205].

Finally, after creating the pseudo-labeled dataset, we use it for training the *Student model*. In this paper we use the MIO-TCD [201] and Tiny ImageNet [206] dataset to evaluate the KD-2 approach.

• *MIO-TCD dataset*: This dataset consists of two parts: 1) classification dataset, and 2) localization dataset. We trained the *Teacher model* using the MIO-TCD classification dataset and used the localization dataset to create our pseudo-labeled dataset for training the *Student model*. It should be noted that in the MIO-TCD classification dataset, we ignored the class "Background" because this class does not belong in the localization dataset [**?**]. Based on this we probed the *Teacher model* and created the pseudo-labeled dataset such that each class had at least 1,713 images.

• *Tiny ImageNet dataset*: The Tiny ImageNet dataset consists of 200 classes and this dataset is used for training the *Teacher model* and the pseudo-labeled dataset is created using the ImageNet dataset [92]. It should be noted that we use only the 200 classes in the ImageNet dataset to create the pseudo-labeled dataset. Based on this we probed the *Teacher model* and created the pseudo-labeled dataset such that each class had at least 487 images. In the MIO-TCD and MS-Celeb datasets we chose the number of augmented images in the pseudo-labeled dataset to be at least 1,713 and 487 images per class respectively, because this is the maximum amount of images possible for the class with the least amount of images.

KD-3: In this setting we know only the total number of classes in the training dataset and do not have any knowledge of the images in the dataset nor we do know the names of the classes either. This scenario occurs in large scale re-identification applications such as face recognition and pedestrian re-identification. We evaluate the KD-3 approach using the MS-Celeb dataset [173]. The MS-Celeb dataset consists of approximately 9.5M images for 99,892 celebrities. It has been shown that this dataset is extremely noisy with many incorrect annotations [207] - [209]. In order to reduce the noise due to incorrect annotation, we followed the community detection algorithm [210] approach of Jin *et al.* [211]. Based on this method, the authors provided a list of correctly annotated images and showed that approximately 97.3% of images in the dataset are correctly labeled. This results in a total of approximately 6.5M images for 94,682 celebrities.

In order to train the *Teacher model*, we manually selected 100 celebrities that had at least 100 images after discarding images that had extremely skewed poses and celebrities wearing sunglasses. We denote this dataset as $Q_{1:100}$ (Q_i is the identity of the celebrity) and it is used for training *Teacher model*. In order to train the *Student model* we first create a pseudo-labeled dataset by probing the *Teacher model* with images of celebrities that do not belong in $Q_{1:100}$ and labeled the images with the predicted class. We denote this pseudo-labeled dataset as $Q_{101:\infty}$ and it should be noted that the dataset $Q_{1:100}$ and $Q_{101:\infty}$ contain images of different celebrities and their data distributions do not overlap (ignoring the noise due to incorrect annotations). Based on this we pseudo-labeled 3,000 images per class in the $Q_{101:\infty}$ dataset.

Although the *Student model* is trained on a dataset that is entirely different from the dataset used for training the *Teacher model*, we are still able to distill some of the learned features from the *Teacher model*. The reason for this is that the *Teacher model* is assumed to be a deterministic model meaning that, after training, the features learned are fixed and do not change over time. Hence, when we probe a given image x, the resulting prediction $f(x) = y$ will never change and with a considerably large and diverse pseudo-labeled dataset, the *student model* model is able to distill the learned features of the *Teacher model* and achieve good classification accuracy on the testing dataset belonging to $Q_{1:100}$. *Prior to [212], this observation had not been addressed in the fields of face biometrics* and is very advantageous because of the abundance of unrestricted images available in the public domain that can effectively be used for distilling the knowledge of Black box facial recognition classifiers.

5.2.3 Uncertainty Prediction via Bayesian Learning

In the domain of adversarial defense, it is very important to know the amount of adversarial perturbation that still remains in the output of any defense algorithm. Bayesian methods offer a principled way to represent these uncertainties in a model and can be utilized to quantify a model's confidence in its prediction [155]. Deep learning models $f(\cdot)$ consists of a set of weights w that are optimized on a labeled dataset $D = \{x_i, y_i\}_{i=1}^{N}$, where x_i and y_i are the input data and corresponding ground-truth, respectively. Bayesian inference involves learning a posterior distribution over the weights $p(w|D)$ which is used for predicting unseen observations:

$$p(y|x, D) = \int p(y|x, w)\, p(w|D) dw \tag{1}$$

The above integral is intractable because of the sheer number of parameters in deep learning models. To overcome this, in our approach we design the Bayesian CNN using *Bayes by Backprop* [182]. *Bayes by Backprop* is a variational inference to Bayesian neural networks where the posterior is assumed to be a diagonal Gaussian distribution which assumes independence between variables. The Gaussian posterior $q_\theta(\omega|D)$ is defined to be as similar as possible to the true posterior $p(\omega|D)$ when measured by the KL divergence [184]. Based on this the optimal parameters are defined as:

$$\theta_{opt} = \arg\min_{\theta} KL(q_\theta(w|D)||p(w)) \; - \; \mathbb{E}_{q(w|\theta)}(log\; p(D|w)) \; + \; log\; p(D) \tag{2}$$

After learning the approximate posterior distribution we compute two uncertainty metrics namely: 1) Aleatoric, and 2) Epistemic uncertainties [181] which are given by:

$$Aleatoric_{\;Uncertainty} \;= \frac{1}{T}\sum_{t=1}^{T} diag(\hat{g}_t) \; - \; \hat{g}_t\, \hat{g}_t^T \tag{3}$$

$$Epistemic_{\;Uncertainty} \;= \frac{1}{T}\sum_{t=1}^{T} (\hat{g}_t \; - \; \tilde{g})(\hat{g}_t \; - \; \tilde{g})^T \tag{4}$$

where, T is the number of samples drawn from the posterior distribution, $\tilde{g} = \frac{1}{T}\sum_{t=1}^{T} \hat{g}_t$ and $\hat{g}_t = f_{w_t}(x)$. It should be noted that we trained the Bayesian CNN using the same pseudo-labeled dataset used for training the *Substitute* model described in the previous sub-section.

Aleatoric Uncertainty: is a measure for the variation of data. This value increases if certain classes are heavily unbalanced/lack of data. Adversarial images have been shown to lie in the high frequency and low probability density regions [124, 145], which is similar to highly imbalanced datasets and long-tailed datasets. This is also a reason why adversarial training [113] is an effective white box defense.

Epistemic Uncertainty: is caused by the model itself. It is the ability of the model to learn robust and representative features which depends on its architecture and parameters. This value increases in the presence of adversarial attacks.

5.2.4 When is an image adversarial?

After learning the posterior distribution $q_\theta(\omega|D)$, in order to find the minimum uncertainty to classify an input image as adversarial. For this we generated adversarial images with the smallest perturbation (i.e. $\epsilon = 1/255$) for the *Substitute* model using three well known attacks: IFGSM [113], BIM [100], and PGD [127]. Next, these adversarial images are transferred to the Bayesian CNN and we compute the average (μ) and standard deviation (σ) of the Epistemic and Aleatoric uncertainties. Finally, we set two thresholds T_1 and T_2 given by:

$$T_1 \; = \; \mu(Aleatoric) \; - 3\sigma(Aleatoric) \tag{5}$$

$$T_2 \; = \; \mu(Epistemic) \; - 3\sigma(Epistemic) \tag{6}$$

For a given image if at least one uncertainty is greater than its corresponding threshold, we classify it as an adversarial image and pass it as input to our ensemble of iterative defenses. We chose the threshold values for T_1 and T_2 to be as shown in Eq. 5 and 6 because, although μ - σ is the least amount of uncertainty corresponding to the annotated training dataset D, in practice there could be unseen attacks in the real-world where the uncertainty value is below μ - σ. To accommodate these unseen adversarial images we set T_1 and T_2 to be two standard deviations lesser than μ - σ as shown in Eq. 5 and 6.

5.2.5 Ensemble of Iterative Adversarial Defenses

The adversarial defenses used in this paper are auxiliary generative networks that can be used in conjunction with any classifier as a pre-processing step without modifying the structure of the classifier. These approaches do not assume any classifier model and are model agnostic. In this paper we chose to use MagNet [121], PixelDefend [124], ShieldNets [145], and Defense-GAN [122] in our ensemble because they achieve state-of-the-art results for white/black box defense. It should be noted that the above mentioned defenses are all single-step defenses and cannot quantify if a purified image is adversarial or not. However, using our defense framework, we are able to convert these single-step defenses into iterative defenses and quantify the amount of adversarial component remaining after each iteration of purification. Additionally, each of these individual defenses are trained independently and this provides flexibility to alter the structure of the ensemble (i.e., add/remove individual defenses without affecting entire framework. See Section IV E).

5.2.6 Determining the Number of Iterations of Purification

In Fig. 24, M is the maximum number of iterations an image can be purified before being (a) passed as input to the black box CNN or (b) rejected. The reason for this is that after each iteration, the amount of purification done decreases and after M iterations the ensemble is not able to further purify the image. This situation arises when the adversarial perturbation (ϵ) is very high causing the adversarial noise to dominate the image, which makes it very difficult to purify the image. This is a potential threat an adversary could use to lock our defense in a state of infinite loops of purification, thus crashing the defense framework. To eliminate this threat, we set a threshold (M) on the maximum number of iterations of purification before rejecting an image. In order to empirically determine the value of M, we attacked the *Substitute* CNN using the IFGSM [113], BIM [100], and PGD [127] attacks with $\epsilon = 0.05$, 0.1, and 0.2. We chose the values of ϵ within the range of 0.05 - 0.2 because this is the range an adversarial attack is likely to fool a human observer, and adversarial images with $\epsilon > 0.2$ makes the resulting images more discernible to the human eye [145, 192]. The resulting adversarial images are then passed as input to our ensemble of image purifiers for six iterations of purification. From this we quantify the amount of purification done by measuring the l_2 distance between the input and output at every iteration. Fig. 25 shows the plots for the amount of purification, Aleatoric, and Epistemic uncertainties VS. the number of iterations of purification for the Fashion-

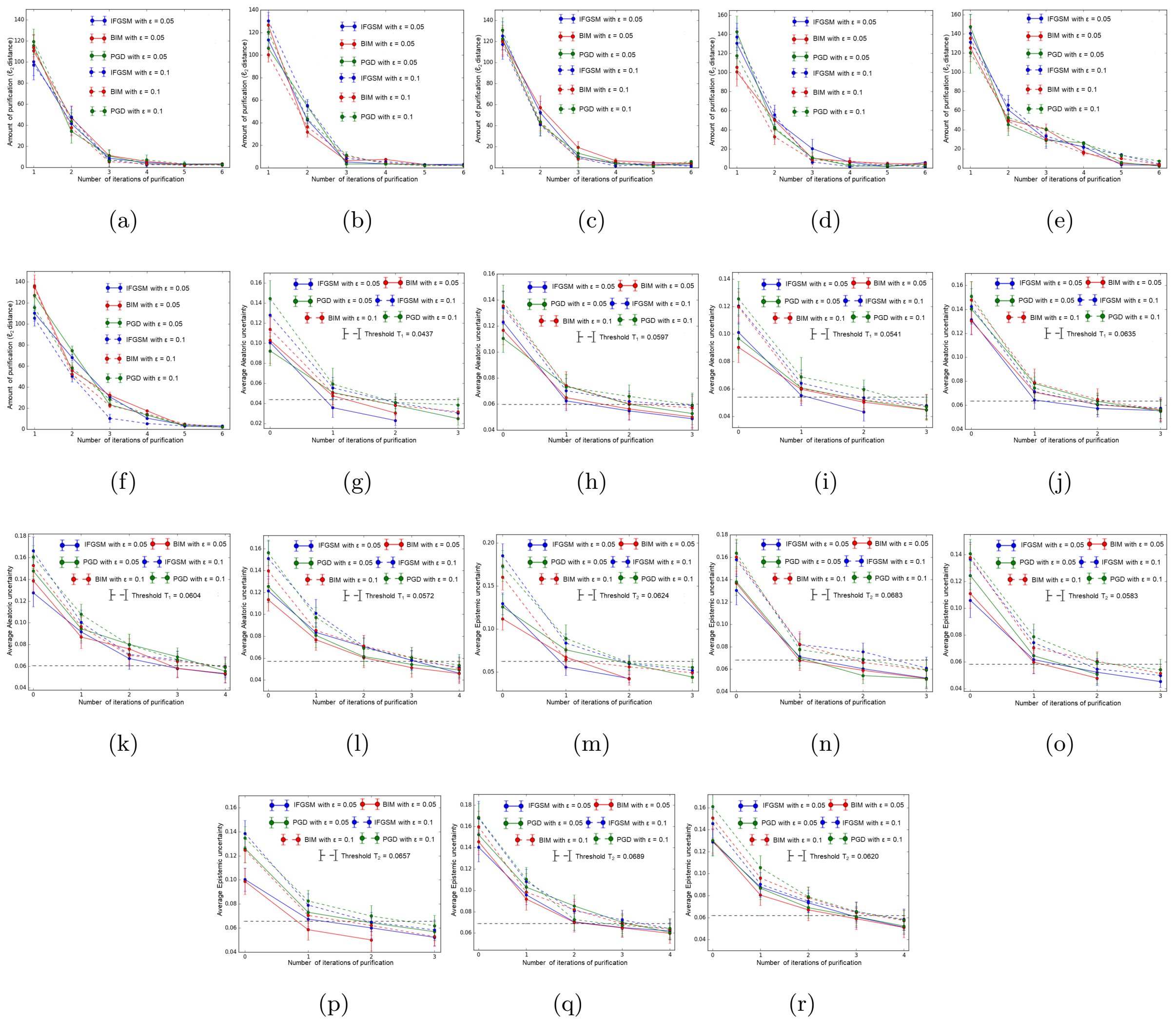

Fig. 25: (a) - (f) shows the average amount of purification VS. the number of iterations of purification, (g) - (l) shows the average Aleatoric uncertainties and (m) - (r) shows the average Epistemic uncertainties for the Fashion-MNIST, CIFAR-10, GTSRB, Tiny ImageNet, MIO-TCD, and MS-Celeb datasets, respectively.

MNIST [104], CIFAR-10 [103], GTSRB [202], MIO-TCD [201], Tiny ImageNet [206], and MS-Celeb [173] datasets with $\epsilon = 0.05$ and 0.1. From Fig. 25(a) - (f) we can see that after 3 iterations, the amount of purification does not significantly change for the Fashion-MNIST, CIFAR-10, GTSRB, and Tiny ImageNet datasets. Hence, we set the vale of $M = 3$ for these datasets. Similarly, for the MIO-TCD and MS-Celeb datasets we set $M = 4$. It is also interesting to note that, as the dimension of the input space increases (see Table 25), the value of M also increases.

5.2.7 Theoretical Lower Bound on the Amount of Purification

For any given image X $\in \mathbb{R}^{P\times Q}$, where $P\times Q$ is the number of pixels in the image, an ϵ-bounded adversarial sample is denoted as $\mathbf{X}+\delta$, where δ belongs to the l_p bounded neighborhood $\Delta = \{\delta \in \mathbb{R}^{P\times Q} \mid \|\delta\|_p \leq \epsilon\}$ to $\mathbf{X}$. The individual adversarial defenses $\pi_\omega(\mathbf{X}'|\mathbf{X}+\delta)$ are expected to map the adversarial images from adversarial regions back to a safer space within Δ, where ω are the trainable parameters of the defense and X' is the output of the defense. Adversarial attacks on any classification task with a loss function of $\mathscr{L}(\mathbf{X}',\mathbf{Y};\theta)$, where, θ are the parameters of the black box classifier, can be achieved by optimizing,

$$\arg\max_{\delta\in\Delta} \int_\Delta \pi_\omega(\mathbf{X}'|\mathbf{X}+\delta)\mathscr{L}(\mathbf{X}',\mathbf{Y};\theta)\mathrm{d}\mathbf{X}'. \tag{7}$$

The loss function of the adversarial defense can be expressed as the marginalized expectation:

$$\mathscr{L}_{Def} = \mathop{\mathbb{E}}_{(\mathbf{X},\mathbf{Y})\sim D} \int_\Delta \mathbb{E}_{\mathbf{X}'\sim\pi_\omega(\cdot|\mathbf{X}+\delta)}\big[\mathscr{L}(\mathbf{X}',\mathbf{Y};\theta)\big]p(\delta)\mathrm{d}\delta \tag{8}$$

where $p(\delta)$ represents the distribution of adversarial samples in Δ. The theoretical possibility of the adversarial defense to neutralize the adversarial images is supported by the following theorem:

Theorem 4 *Assume $\mathscr{L}(\mathbf{X}',\mathbf{Y};\theta)$ is continuous in $\mathbf{X}+\Delta$ and $\pi_\omega(\mathbf{X}'|\mathbf{X}+\delta)$ is supported on $\mathbf{X}+\Delta$, there exists a lower bound for $\mathscr{L}_{PAR}$ in space Δ. If $\pi_\omega(\mathbf{X}'|\mathbf{X}+\delta) = \delta_{Dirac}(\mathbf{X}'-\mathbf{X}-\beta_0)$, $\mathscr{L}_{PAR}$ reaches the lower bound, where $\beta_0 = \arg\min_{\beta\in\Delta}\mathscr{L}(\mathbf{X}+\beta,\mathbf{Y};\theta)$.*

Proof of Theorem 4: See Theorem 2 in Chapter 4.2.1. Theorem 4 is further empirically supported by Fig. 25, in Fig. 25(a) - (f) it can be seen that after M iterations the amount of purification significantly decreases and the image can no longer be purified. This also means that after M iterations, the adversarial image X_{adv} is transformed/purified to X' $\in \beta_0$, beyond which it cannot be further purified. Additionally, from Fig. 25 it can be seen that as the amount of purification decreases after each iteration (Fig. 25(a) - (f)), the aleatoric and epistemic uncertainties also gradually decrease (Fig. 25(g) - (r)) and after M iterations they do not significantly vary because the the amount of purification does not change. This further emphasizes that the resulting purified image X' has been mapped into the lower bound space $\beta_0 \in \Delta$ and cannot be further purified.

5.3 Experimental Results

5.3.1 CNN Architectures and Datasets

Table 24 shows the architectures of the CNN used in our defense. For fair comparison, the CNN architectures in Table 24 are the same as reported in [145] and [192]. We evaluated our black box defense by creating an adversarial *Substitute* CNN and transferred the adversarial images generated for the adversary's *Substitute* as input to our defense [106, 185]. It should be noted that in Table 24 we used the same CNN architecture and training data for our defense's *Substitute* as well as the adversary's *Substitute.* By doing so we are giving the adversary equal knowledge as to our *Substitute* model in order to have a fair evaluation of our defense.

TABLE 24: CNN architectures

CNN Architectures	
Target Black Box CNN	ResNet
Defense *Substitute* CNN	VGG
Bayesian CNN	Bayesian VGG
Adversary's *Substitute* CNN	VGG

We evaluate our defense on six public benchmark datasets namely: Fashion-MNIST [104], CIFAR-10 [103], GTSRB [202], MIO-TCD [201], Tiny ImageNet [206], and MS-Celeb [173]. Fig. 26 shows examples of images and Table 25 shows a brief summary of the datasets used in this paper.

Fig. 26: Example of images from the (a) GTSRB, (b) MIO-TCD, and (c) MS-Celeb datasets

TABLE 25: Summary of the datasets used in this paper.

Dataset	**Image size**	**Grayscale/ RGB**	**Number of classes**	**Balanced classes? †**	**Training data**	**Testing data**
Fashion-MNIST	28 x 28	Grayscale	10	Yes (6,000)	60,000	10,000
CIFAR-10	32 x 32	RGB	10	Yes (5,000)	50,000	10,000
GTSRB	64 x 64*	RGB	43	No	39,252	12,630
Tiny ImageNet	64 x 64	RGB	200	Yes (500)	100,000	10,000
MS-Celeb	128 x 128*	RGB	100	No	8,933	2,177
MIO-TCD	224 x 224**	RGB	10	No	359,164	129,796

*images are resized to the specified size. ** shorter side of the image is resized 256 maintaining the aspect ratio, and center cropped to size of 224 x 224. † Number in brackets indicates the number of images per class.

5.3.2 Threat Models

In this sub-section we define the adversarial attacks used for evaluating our defense. For a given test image-label pair (x, y), adversarial attacks find a perturbation δ with $||\delta||_\infty \leq \epsilon$ such that a deep learning classifier $f(\cdot)$ results in $f(x + \delta) \neq y$. ϵ is a hyper-parameter that sets the perturbation limit for each pixel in x on the color scale.

• **Iterative Fast Gradient Sign Method (IFGSM)** [113]: This attack uses the sign of the gradients at every pixel to determine the direction of perturbation.

$$x_{n+1}^{adv} = x_n \; + \epsilon \cdot sign(\bigtriangledown_x L(x, y)) \tag{9}$$

• **Basic Iterative Method (BIM)** [100]: This attack extends the FGSM attack [113] by iterating it multiple times with a small step size.

$$x_{n+1}^{adv} = Clip_\epsilon(x_n + \alpha \cdot sign(\bigtriangledown_x L(x_n, y))) \tag{10}$$

• **Projected Gradient Descent** [127]: This attack computes the gradient in the direction of the highest loss and projects it back to the l_p norm around the sample.

$$x_{n+1}^{adv} = \prod^{\epsilon}(x_n + \alpha \cdot sign(\bigtriangledown_x L(x_n, y))) \tag{11}$$

In eq. (9) - (11), $\bigtriangledown_x L(x, y)$ is the loss function used to train the CNN, α is the iterative step size, Clip $(\cdot)$ and $\prod$ $(\cdot)$ are the clipping and projection functions, respectively.

5.3.3 Performance Evaluation of the Proposed KD Approaches

Table 26 shows the baseline performance comparison between the *Student model* (defense substitute classifier) and the *Teacher model* (black box classifier). In Table 26 although the *Teacher model* outperforms the *Student model*, it can be seen that as the overlap between the *Teacher model's* training dataset and the pseudo-labeled dataset increases, the performance of the *Student model* also increases.

TABLE 26: Performance evaluation and comparison of our KD approaches with respect to the Teacher (black box) classifier.

Training dataset (Teacher classifier)	**Teacher accuracy (%)**	**Pseudo-labeled dataset (substitute)**	**Student accuracy (%)**
Fashion-MNIST training dataset	93.51	X + GAN	89.67 ± 1.32
		X + GAN	91.63 ± 0.76
CIFAR-10 training dataset	95.31	X + GAN	85.79 ± 1.78
		X + GAN	88.42 ± 1.26
GTSRB training dataset	96.45	X + GAN	90.34 ± 1.04
		X + GAN	91.79 ± 0.68
MIO-TCD classification training dataset	94.68	MIO-TCD localization	84.51
		MIO-TCD localization + 50% MIO-TCD classification	89.04 ± 0.83
Tiny ImageNet training dataset	Top 1: 46.79 Top 5: 72.30	200 classes from ImageNet training dataset	Top 1: 40.85 Top 5: 64.37
		200 classes from ImageNet + 50% Tiny ImageNet	Top 1: 41.33 ± 0.57 Top 5: 65.59 ± 1.08
MS-Celeb (100 celebrities) 75% of 100 celebrities for training	90.32 ± 1.56	Every other celeb than the 100	74.58 ± 3.72
		Every other celebrity than the 100 + 50% of 100 celebrities	76.40 ± 2.47

5.3.4 Performance Evaluation and Comparison of our Defense Against Adversarial Attacks

Tables 27 - 34 show the performance and comparison of our approach against the state-of-the-art on the six public benchmark dataset described in Section IV A. Note that in Tables 27 - 34 all of the single-step adversarial defenses (MagNet [121], PixelDefend [124], ShieldNets [145], and Defense-GAN [122]) are converted into iterative defenses using our

TABLE 27: Performance Comparison of our Defense on the Fashion-MNIST dataset using the KD-1 approach with X = 25%.

		ϵ = 0.1 (26/255)		
			T1 = 0.0437	**T2 = 0.0624**
Attack	**Defense**	**Classification Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	37.51	-	uncertainty
	MagNet	80.37 ± 4.56	0.0496 ± 0.0095	0.0683 ± 0.0098
	ShieldNets	83.56 ± 4.78	0.0480 ± 0.0087	0.0655 ±0.0099
	Defense-GAN	83.70 ± 5.09	0.0349 ± 0.0085	0.0441 ± 0.0088
	PixelDefend	81.24 ± 4.17	0.0487 ± 0.0092	0.0673 ± 0.0096
	Ensemble	87.03 ± 4.40	0.0302 ± 0.0089	0.0418 ± 0.0082
BIM	**No Defense**	35.66	-	-
	MagNet	83.66 ± 5.11	0.0479 ± 0.0102	0.0661 ± 0.0094
	ShieldNets	84.51 ± 4.38	0.0455 ± 0.0094	0.0637 ± 0.0091
	Defense-GAN	83.02 ± 4.18	0.0327 ± 0.0090	0.0430 ± 0.0093
	PixelDefend	81.08 ± 4.22	0.0482 ± 0.0107	0.0647 ± 0.0091
	Ensemble	86.75 ± 3.98	0.0315 ± 0.0091	0.0391 ± 0.0095
PGD	**No Defense**	31.83	-	-
	MagNet	80.74 ± 3.68	0.0522 ± 0.0090	0.0701 ± 0.0109
	ShieldNets	81.63 ± 4.87	0.0505 ± 0.0094	0.0684 ± 0.0103
	Defense-GAN	82.80 ± 3.79	0.0376 ± 0.0080	0.0456 ± 0.0093
	PixelDefend	78.52 ± 5.56	0.0589 ± 0.0096	0.0733 ± 0.0104
	Ensemble	84.55 ± 4.12	0.0364 ± 0.0087	0.0433 ± 0.0098

framework. From Tables 27 - 34 it can be seen that our ensemble of defenses outperforms all the stand-alone defenses. Although, as the perturbation limit (ϵ) of the adversarial attack increases, the performance of all the approaches in Tables 27 - 34 gradually decrease, but there is also a gradual increase in the Bayesian uncertainty metrics. This means that even if an adversary tries to break our defense by significantly increasing the value of (ϵ), the resulting adversarial image would still be rejected because the uncertainty values of the image are beyond the threshold limits T_1 and T_2. Although increasing the value of (ϵ) in

TABLE 28: Performance comparison of our Defense on the Fashion-MNIST dataset using the KD-1 approach with X = 50%.

		= 0.1 (26/255)		
			T1 = 0.0426	**T2 = 0.0654**
Attack	**Defense**	**Accuracy**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	37.51	-	-
	MagNet	81.96 ± 2.90	0.0445 ± 0.0077	0.0644 ± 0.0080
	ShieldNets	83.92 ± 2.68	0.0412 ± 0.0098	0.0603 ±0.0089
	Defense-GAN	84.11 ± 2.88	0.0358 ± 0.0089	0.0424 ± 0.0090
	PixelDefend	82.05 ± 2.59	0.0464 ± 0.0094	0.0632 ± 0.0091
	Ensemble	**88.29 ± 2.40**	**0.0330 ± 0.0084**	**0.0406 ± 0.0087**
BIM	**No Defense**	35.66	-	-
	MagNet	84.87 ± 1.86	0.0435 ± 0.0097	0.0618 ± 0.0090
	ShieldNets	84.75 ± 2.46	0.0433 ± 0.0091	0.0602 ± 0.0086
	Defense-GAN	84.44 ± 2.95	0.0380 ± 0.0096	0.0452 ± 0.0081
	PixelDefend	81.76 ± 2.03	0.0466 ± 0.0097	0.0627 ± 0.0093
	Ensemble	**87.08 ± 2.92**	**0.0347 ± 0.0086**	**0.0411 ± 0.0087**
PGD	**No Defense**	31.83	-	-
	MagNet	81.27 ± 3.24	0.0462 ± 0.0098	0.0629 ± 0.0095
	ShieldNets	82.94 ± 2.76	0.0455 ± 0.0084	0.0621 ± 0.0087
	Defense-GAN	83.06 ± 1.93	0.0345 ± 0.0091	0.0437 ± 0.0086
	PixelDefend	80.47 ± 1.86	0.0491 ± 0.0074	0.0635 ± 0.0089
	Ensemble	**86.13 ± 2.38**	**0.0319 ± 0.0094**	**0.0446 ± 0.0091**

order to break a defense may seem very trivial, it is still a foundational problem in online applications where there is no human in the loop [190, 191]. In Table 32, we report the Top 5 classification accuracy for the Tiny ImageNet as this is the metric used for evaluating the dataset. Additionally, the adversarial images used in Table 32 were generated such that the top 5 predictions for an adversarial image do not contain the ground-truth label (y), i.e., $y \notin \{y'_1, y'_2, y'_3, y'_4, y'_5\}$.

TABLE 29: Performance comparison of our Defense on the CIFAR-10 dataset using the KD-1 approach with X = 25%.

		= 0.1 (26/255)		
			T1 = 0.0597	**T2 = 0.0688**
Attack	**Defense**	**Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	15.61	-	-
	MagNet	61.74 ± 4.83	0.0626 ± 0.0088	0.0635 ± 0.0091
	ShieldNets	62.36 ± 3.78	0.0608 ± 0.0094	0.0619 ± 0.0076
	PixelDefend	58.03 ± 3.17	0.0645 ± 0.0097	0.0657 ± 0.0084
	Ensemble	**67.91 ± 3.55**	**0.0587 ± 0.0086**	**0.0610 ± 0.0098**
BIM	**No Defense**	12.78	-	-
	MagNet	64.70 ± 3.67	0.0588 ± 0.0091	0.0611 ± 0.0085
	ShieldNets	64.33 ± 3.91	0.0593 ± 0.0095	0.0598 ± 0.0094
	PixelDefend	61.26 ± 3.40	0.0613 ± 0.0088	0.0604 ± 0.0092
	Ensemble	**67.45 ± 3.73**	**0.0571 ± 0.0079**	**0.0582 ± 0.0083**
PGD	**No Defense**	12.56	-	-
	MagNet	61.35 ± 3.18	0.0633 ± 0.0089	0.0625 ± 0.0081
	ShieldNets	60.87 ± 3.05	0.0631 ± 0.0096	0.0642 ± 0.0090
	PixelDefend	59.87 ± 3.11	0.0622 ± 0.0080	0.0628 ± 0.0084
	Ensemble	**65.09 ± 3.37**	**0.0593 ± 0.0084**	**0.0595 ± 0.0092**

5.3.5 Ablation Study for Evaluating the Different Combinations of Ensembles of Iterative Defenses

In this subsection we perform an ablation study to evaluate different combinations of ensembles of adversarial defenses and compare their performance. For this purpose we chose to use the Fashion-MNIST [104] and CIFAR-10 [103] datsets and the following adversarial defenses: MagNet [121], PixelDefend [124], ShieldNets [145], and Defense-GAN [122]. Table 35 shows the comparisons of different ensembles. From Table 35 it can be seen that using the ensemble MPSD achieves the best overall performance followed my MSD.

TABLE 30: Performance comparison of our Defense with on the CIFAR-10 dataset using the KD-1 approach with X = 50%.

		ϵ = 0.1 (26/255)		
			T1 = 0.0614	**T2 = 0.0672**
Attack	**Defense**	**Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	15.61	-	-
	MagNet	62.47 ± 3.97	0.0597 ± 0.0082	0.0608 ± 0.0086
	ShieldNets	62.94 ± 3.55	0.0571 ± 0.0090	0.0588 ± 0.0082
	PixelDefend	60.44 ± 3.48	0.0609 ± 0.0080	0.0634 ± 0.0077
	Ensemble	**68.52 ± 3.14**	**0.0579 ± 0.0082**	**0.0598 ± 0.0095**
BIM	**No Defense**	12.78	-	-
	MagNet	66.10 ± 4.09	0.0582 ± 0.0079	0.0591 ± 0.0082
	ShieldNets	64.96 ± 3.51	0.0578 ± 0.0096	0.0605 ± 0.0091
	PixelDefend	62.89 ± 3.62	0.0601 ± 0.0082	0.0585 ± 0.0087
	Ensemble	**68.01 ± 3.58**	**0.0549 ± 0.0083**	**0.0561 ± 0.0084**
PGD	**No Defense**	12.56	-	-
	MagNet	63.11 ± 3.21	0.0590 ± 0.0077	0.0615 ± 0.0096
	ShieldNets	64.26 ± 3.30	0.0605 ± 0.0079	0.0591 ± 0.0082
	PixelDefend	61.90 ± 3.20	0.0609 ± 0.0083	0.0610 ± 0.0094
	Ensemble	**66.28 ± 2.89**	**0.0564 ± 0.0070**	**0.0573 ± 0.0095**

5.4 Robustness of our Defense - an Adversary's Point of View

In Section III A, we assumed that the adversary has no knowledge about our defense and can only see the input and final classification output of the black box classifier. Although this may seem to be a strong assumption, in this sub-section we relax this assumption by allowing the adversary to have partial amounts of information about our defense. We chose to use the CIFAR-10 dataset and IFGSM attack with $\epsilon = 0.1$. In order to quantify the robustness of our approach, we measure the time taken for the adversary to create 50 successful adversarial attacks against our defense using 2 TITAN X GPUs. Table 36 shows the time complexity required to break our defense when the adversary has

TABLE 31: Performance comparison of our Defense on the GTSRB dataset using the KD-1 approach with X = 50% and $\epsilon = 0.1$.

		GTSRB KD 1 (X = 50%)		
			T1 = 0.0525	**T2 = 0.0570**
Attack	**Defense**	**Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	28.93	-	-
	MagNet	84.01 ± 3.12	0.0497 ± 0.0091	0.0515 ± 0.0087
	ShieldNets	86.89 ± 3.35	0.0477 ± 0.0080	0.0483 ± 0.0077
	PixelDefend	79.43 ± 2.81	0.0537 ± 0.0084	0.0534 ± 0.0088
	Ensemble	**89.94 ± 2.74**	**0.0467 ± 0.0090**	**0.0471 ± 0.0081**
BIM	**No Defense**	22.05	-	-
	MagNet	81.33 ± 3.76	0.0539 ± 0.0074	0.0548 ± 0.0079
	ShieldNets	86.07 ± 3.16	0.0459 ± 0.0081	0.0502 ± 0.0078
	PixelDefend	80.76 ± 2.91	0.0532 ± 0.0085	0.0537 ± 0.0081
	Ensemble	**86.97 ± 3.03**	**0.0482 ± 0.0083**	**0.0460 ± 0.0095**
PGD	**No Defense**	24.51	-	-
	MagNet	83.44 ± 3.17	0.0518 ± 0.0070	0.0510 ± 0.0073
	ShieldNets	86.95 ± 3.47	0.0472 ± 0.0093	0.0496 ± 0.0078
	PixelDefend	82.04 ± 2.93	0.0503 ± 0.0088	0.0508 ± 0.0080
	Ensemble	**89.40 ± 3.53**	**0.0451 ± 0.0090**	**0.0457 ± 0.0074**

varying amounts of information about our defense framework. In Table 36 we attack the defense framework by creating adversarial attacks against the adversary's substitute CNN and transfer the attacks to our defense framework [106, 185]. From Table 36 we can see that when the adversary has no knowledge about our defense framework, it takes approximately 38 hours to create 50 successful adversarial attacks. To put this into perspective, it took 10 hours for Song *et al.* [124] to create 100 attacks against their defense using 1 TITAN X GPU and 27 hours for Theagarajan *et al.* [145] to create 50 attacks against their defense using 2 TITAN X GPUs.

TABLE 32: Performance comparison of our Defense on the Tiny ImageNet dataset using the KD-2 approach with $\epsilon = 0.1$.

		Black box training data = Tiny ImageNet **Defense training data = ImageNet + 50% ofTiny ImageNet**		
			T1 = 0.0598	**T2 = 0.0647**
Attack	**Defense**	**Top 5 Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	12.76	-	-
	MagNet	44.57 ± 1.24	0.0523 ± 0.0089	0.0607 ± 0.0095
	ShieldNets	47.17 ± 1.06	0.0527 ± 0.0082	0.0571 ± 0.0088
	PixelDefend	40.07 ± 0.97	0.0591 ± 0.0099	0.0605 ± 0.0106
	Ensemble	**50.34 ± 1.08**	**0.0528 ± 0.0080**	**0.0526 ± 0.0078**
BIM	**No Defense**	14.97	-	-
	MagNet	45.80 ± 0.87	0.0521 ± 0.0090	0.0569 ± 0.083
	ShieldNets	49.27 ± 1.11	0.0508 ± 0.0079	0.0518 ± 0.0086
	PixelDefend	42.39 ± 0.72	0.0560 ± 0.0108	0.0627 ± 0.0094
	Ensemble	**50.86 ± 1.04**	**0.0483 ± 0.0091**	**0.0502 ± 0.0077**
PGD	**No Defense**	8.24	-	-
	MagNet	41.85 ± 1.37	0.0587 ± 0.0110	0.0611 ± 0.0087
	ShieldNets	43.18 ± 1.30	0.0571 ± 0.0103	0.0604 ± 0.0093
	PixelDefend	37.90 ± 1.45	0.0635 ± 0.0097	0.0667 ± 0.0094
	Ensemble	**44.01 ± 1.08**	**0.0534 ± 0.0080**	**0.0579 ± 0.0089**

Attacking the Bayesian CNN: In our defense the Bayesian CNN decides if an incoming image is adversarial or not before (i) passing it to the Black box classifier or (ii) rejecting the image. Hence, a natural target for an adversary to beat our defense would be to adversarially attack the Bayesian CNN. The optimization function for creating adversarial

TABLE 33: Performance comparison of our Defense on the MIO-TCD classification dataset using the KD-2 approach with no overlap between the black box and pseudo-labeled dataset.

		Black box training data = MIO-TCD Classification Defense training data = MIO-TCD Localization		
		ϵ = 0.1 (26/255)		
			T1 = 0.0604	**T2 = 0.0689**
Attack	**Defense**	**Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	13.52	-	-
	MagNet	73.28	0.0588 ± 0.0094	0.0637 ± 0.0102
	ShieldNets	75.89	0.0579 ± 0.0089	0.0618 ± 0.0096
	PixelDefend	69.57	0.0632 ± 0.0085	0.0683 ± 0.0094
	Ensemble	**77.04**	**0.0547 ± 0.0082**	**0.0607 ± 0.0079**
BIM	**No Defense**	14.21	-	-
	MagNet	74.01	0.0576 ± 0.0091	0.0621 ± 0.0084
	ShieldNets	72.60	0.0589 ±0.0083	0.0635 ± 0.0105
	PixelDefend	67.83	0.0645 ± 0.0095	0.0702 ± 0.0090
	Ensemble	**74.58**	**0.0559 ± 0.0084**	**0.0613 ± 0.097**
PGD	**No Defense**	12.76	-	-
	MagNet	73.96	0.0569 ± 0.0104	0.0602 ± 0.0094
	ShieldNets	75.07	0.0572 ± 0.0087	0.0626 ± 0.0097
	PixelDefend	65.72	0.0652 ± 0.0086	0.0733 ± 0.0081
	Ensemble	**75.78**	**0.0560 ± 0.0091**	**0.0623 ± 0.0084**

examples against the Bayesian CNN is shown in Eq. 12.

$$\arg\max \ \ ||\delta|| \ < \ \epsilon \tag{12a}$$

$$\textbf{S.T.} \ \ y' \ \neq \ y \tag{12b}$$

$$\textit{Aleatoric uncertainty} \ < \ T_1 \tag{12c}$$

$$\textit{Epistemic uncertainty} \ < \ T_2 \tag{12d}$$

From Table 36 we can see that, attacking the Bayesian CNN is by far the weakest point

TABLE 34: Performance comparison of our Defense on the MS-Celeb dataset using the KD-3 approach with no overlap between the black box and pseudo-labeled training dataset.

		Black box training dataset = 75% of the 100 celebrities Defense training dataset = All other celebrities		
		ϵ = 0.1 (26/255)		
			T1 = 0.0572	**T2 = 0.0620**
Attack	**Defense**	**Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
IFGSM	**No Defense**	18.77 ± 4.29	-	-
	MagNet	61.38 ± 2.91	0.0541 ± 0.0068	0.0585 ± 0.0077
	ShieldNets	62.02 ± 2.75	0.0539 ± 0.0079	0.0546 ± 0.0087
	PixelDefend	58.15 ± 3.04	0.0551 ± 0.0080	0.0613 ± 0.0092
	Ensemble	**63. 44 ± 2.70**	**0.0506 ± 0.0065**	**0.0542 ± 0.0071**
BIM	**No Defense**	22.43 ± 3.97	-	-
	MagNet	61.97 ± 3.24	0.0522 ± 0.0089	0.0576 ± 0.0082
	ShieldNets	60.55 ± 2.74	0.0567 ± 0.0068	0.0566 ± 0.0072
	PixelDefend	56.38 ± 3.39	0.0583 ± 0.0083	0.0637 ± 0.0090
	Ensemble	**62.84 ± 2.13**	**0.0517 ± 0.0069**	**0.0528 ± 0.0074**
PGD	**No Defense**	17.94 ± 4.25	-	-
	MagNet	60.37 ± 2.61	0.0553 ± 0.0079	0.0557 ± 0.0082
	ShieldNets	62.58 ± 2.84	0.0534 ± 0.0082	0.0540 ± 0.0073
	PixelDefend	55.93 ± 3.07	0.0606 ± 0.0097	0.0618 ± 0.0103
	Ensemble	**62.97 ± 2.71**	**0.0535 ± 0.0068**	**0.0529 ± 0.0076**

in our defense which can be exploited by an adversary provided the adversary has the information. But, optimizing Eq. 12 makes the adversarial images to be very close to the boundary of the original images thus resulting in a weakly perturbed adversarial image. We noticed that when we pass these weakly perturbed adversarial images through our ensemble of defense the resulting image is purified in just 1 iteration of purification. Hence, in Table 36 when we force all input images (regardless if they are adversarial or not) to at least 1 iteration of purification, these weakly perturbed adversarial examples are no longer

TABLE 35: Ablation study for comparing different combinations of ensembles of iterative defenses.

		IFGSM $\epsilon = 0.05$ (13/255)		
			T1 = 0.0437	**T2 = 0.0624**
Dataset	**Ensemble**	**Accuracy (%)**	**Avg. Aleatoric uncertainty**	**Avg. Epistemic uncertainty**
Fashion-MNIST	**MPD**	84.33 ± 4.01	0.0314 ± 0.0079	0.0347 ± 0.0080
	MSD	86.83 ± 4.44	0.0276 ± 0.0085	0.0336 ± 0.0086
	PSD	85.09 ± 5.07	0.0304 ± 0.0077	0.0352 ± 0.0078
	MPS	87.35 ± 3.24	0.0309 ± 0.0083	0.0335 ± 0.0081
	MPSD	**87.94 ± 4.51**	**0.0288 ± 0.0089**	**0.0324 ± 0.0091**
			T1 = 0.0597	T2 = 0.0683
CIFAR-10	**MPD**	75.08 ± 4.49	0.0513 ± 0.0078	0.0546 ± 0.0083
	MSD	77.23 ± 4.05	**0.0472 ± 0.0086**	0.0539 ± 0.092
	PSD	74.37 ± 3.64	0.0533 ± 0.0077	0.0560 ± 0.0088
	MPS	75.96 ± 3.74	0.0509 ± 0.0085	**0.0514 ± 0.0074**
	MPSD	**77.86 ± 3.87**	0.0487 ± 0.0095	0.0521 ± 0.0081

*M, P, S, and D refers to MagNet, PixelDefend, ShieldNets, and Defense-GAN, respectively.

adversarial and are correctly classified by the black box classifier. By doing so, even if the adversary has full knowledge about the Bayesian CNN but no information about our ensemble of defenses, it takes approximately 34 hours to break our defense compared to just 17 minutes when there is no forced purification.

Attacking the Ensemble of Defenses: From Table 36 we can see that when the adversary has partial information about our ensemble of defenses, the time taken to break the defense ranges from 28 to 6.65 hours. The reason for this is that, with the inclusion of probabilistic generative networks such as PixelDefend [124] and ShieldNets [145], the ensemble changes from being a deterministic system to a probabilistic system and in order to attack a probabilistic system, one needs to solve the stochastic gradient descent. The convergence rate for solving this is in the order of $O(1/\lambda)$, where λ is the convergence error and This

TABLE 36: Time complexity required for breaking our defense framework on the CIFAR-10 dataset using the IFGSM attack with $\epsilon = 0.1$.

Bayesian CNN	**MagNet**	**PixelDefend**	**ShieldNets**	**Time to create 50 attacks**
✗	✗	✗	✗	38 hours
✓	✗	✗	✗	17 minutes
✓*	✗	✗	✗	34 hours
✗	✓	✗	✗	19 hours
✗	✗	✓	✗	28 hours
✗	✗	✗	✓	14 hours
✗	✓	✓	✗	9.55 hours
✗	✗	✓	✓	8.70 hours
✗	✓	✗	✓	6.65 hours

*we force all input images to have 1 forced iteration of purification

is exponentially slower than the deterministic case. Additionally, the individual defenses are all trained independently and have independent parameters, hence a perturbation in a certain direction leading to a misclassification against a particular defense does not necessarily lead to a similar perturbation in the same direction for the other defenses within the ensemble. Moreover, breaking our defense requires a significant amount of probing and querying from the adversary's side and this can be limited by setting a threshold beyond which the adversary cannot probe the defense for a certain amount of time [186], hence further increasing the computation overhead.

Chapter 6

Conclusions

This thesis proposed three novel applications for automated stem cell classification, sports analytics, and a novel framework for defending deep learning models from white and black box models. This dissertation showed how crucial generative networks are for improving the robustness of deep learning models and defending them against adversarial attacks. In the field of stem cell classification, we proposed DeephESC 2.0 an automated system for detecting and classifying hESC images. DeephESC 2.0 outperforms the state-of-the-art in both the classification and generation of synthetic hESC images. We observed that the certain classes such as *Cell clusters/Apoptically Blebbing cells* and *Attached cells/Dynamically Blebbing cells* have similar texture and intensity and they are only different in their morphology. To exploit this difference we designed Triplet CNN architectures with branched convolution layers that can detect these minute changes in morphology and perform fine-grained classification for further improving the classification accuracy of these classes. Moreover, by fusing the outputs of the CNN and Triplet CNNs using the product rule we were able to further improve the classification accuracy to 93.23%.

We designed individual GMANs for each class to generate synthetic hESC images. We evaluated the quality of the generated images using the SSIM, PSNR and statistical

p- value metrics and our approach outperformed state-of-the-art approaches for generating synthetic hESC images. Furthermore, we trained the classifier of DeephESC 2.0 exclusively on 40,000 synthetic images per class and evaluated the classifier on the real hESC images and achieved further improved classification accuracy of 94.46%. We discussed the possible reasons for misclassification and observed that some images were unintentionally mislabeled by the biologists and our approach was able to predict their correct class. This shows that our approach is robust even in the presence of noisy data.

In sports analytics, we proposed and designed a system for analyzing the performance of soccer players and generating three tactical statistics of each player from a video. We collected a dataset consisting of 49,950 images from high school soccer matches and performed exhaustive evaluation and comparison of algorithms on the dataset and our approach achieved the best performance in terms of accuracy and computation time. Moreover, we observed that although our approach achieves the best performance on matches played between teams in our training dataset, the features learned do no generalize well across matches played by teams that are not in our dataset. To solve this we employed a minimum amount of match specific annotations using a novel Triplet CNN-DCGAN architecture and showed that by fine tuning the network with only 100 annotated images per class (*Player with/without the ball*) we can obtain robust performance. Finally, we performed an ablation study that showed how individual modules of our proposed approach and data augmentation affect the generation of tactical statistics at a match level and individual player level. The Future work will include using multiple wide lens stationary cameras and GPS trackers in an IOT based cloud environment which will provide real-time performance. Additionally more actions can be integrated into our system such as shots on the goal, dribbling detection and player style classification which can be used for generating a more comprehensive performance characterization of an individual soccer player.

Although, these applications achieve state-of-the-art performance, this dissertation showed how these approaches are vulnerable to adversarial attacks. To solve this, this dissertation proposed Probabilistic Adversarial Robustness (PAR) and implemented it via adopting PixelCNN as the probabilistic transformation model to defend target CNNs against adversarial attacks. We theoretically derived the connection between PAR loss and the SGD loss, and the existence of a theoretical lower bound of PAR loss representing the optimal mapping of the adversarial examples to the adversarial-free zones. We numerically demonstrated that ShieldNet can greatly improve the defending accuracy for intra-attack and generalize well across different attacking methods. Moreover, experimental results demonstrated the generality of our approach to adversarial transferability with respect to different CNN models and its resistance to existing attacks. We additionally proposed a novel framework for defending black box classifiers from adversarial attacks. The proposed framework uses an ensemble of defenses and has the ability to convert existing single-step black box defenses into an iterative defense and experimental results show that using an ensemble of defenses outperforms the corresponding single-step defenses. We demonstrated the relationship between an adversarial image and its corresponding purified image and proved the existence of a lower bound space beyond which an image cannot be further purified. This paper also proposed three novel knowledge distillation approaches that exploit prior meta-information of the training datasets. Furthermore, the results show that crowd sourced images that are available in the public domain can be used to effectively distill the knowledge from the Black box classifier and still achieve reasonable performance in defending against adversarial attacks.

Bibliography

[1] J. A. Thomson, J. Itskovitz-Eldor, S. S. Shapiro, M. A. Waknitz, J. J. Swiergiel, V. S. Marshall, and J. M. Jones, "Embryonic stem cell lines derived from human blastocysts", *Science*, 282(5395), pp. 1145-1147, 1998.

[2] Z. Zhu, and D. Huangfu, "Human pluripotent stem cells: an emerging model in developmental biology", *Development*, 140, pp. 705-717, 2013.

[3] P. Talbot, and S. Lin, "Mouse and human embryonic stem cells: can they improve human health by preventing disease?", *Current Topics in Medicinal Chemistry*, 11(13), pp. 1638-1652, 2011.

[4] B. X. Guan, B. Bhanu, B, P. Talbot, S. Lin, and N. Weng, "Comparison of texture features for human embryonic stem cells with bio-inspired multi-class support vector machine", *IEEE International Conference in Image Processing*, pp. 4102-4106, 2014.

[5] R. Theagarajan, B. X. Guan, and B. Bhanu, "DeephESC: An automated system for generating and classification of human embryonic stem cells", *IEEE International Conference on Pattern Recognition*, 2018.

[6] B. Bhanu, and P. Talbot (Eds.), "Video Bioinformatics: From Live Imaging to Knowledge", *Springer*, 2015.

[7] S. Lin, S. Fonteno, S. Satish, B. Bhanu, and P. Talbot, "Video bioinformatics analysis of human embryonic stem cell colony growth", *Journal of visualized experiments*, 2010.

[8] P. Talbot, N. Zur Nieden, S. Lin, I. Martinez, B. X. Guan, and B. Bhanu, "Use of video bioinformatics tools in stem cell toxicology", *Handbook of Nanotoxicology, Nanomedicine and Stem Cell Use in Toxicology*, 2014.

[9] R. Sakamoto, M. M. Rahman, M. Shimomura, M. Itoh, T. and Nakatsura, "Time-lapse imaging assay using the BioStation CT: A sensitive drug-screening method for three-dimensional cell culture", *Cancer science*, 106(6), pp. 757-765, 2015.

[10] A. Zahedi, V. On, S. C. Lin, B. C. Bays, E. Omaiye, B. Bhanu, and P. Talbot, "Evaluating cell processes, quality, and biomarkers in pluripotent stem cells using video bioinformatics", *PLoS One*, 11(2), 2016.

[11] F. Ambriz-Colin, M. Torres-Cisneros, J. Avina-Cervantes, J. Saavedra-Martinez, O. Debeir, and J. Sanchez-Mondragon, "Detection of biological cells in phase-contrast microscopy images", *Mexican International Conference on Artificial Intelligence*, pp. 68-77, 2006.

[12] K. Li, M. Chen, and T. Kanade, "Cell population tracking and lineage construction with spatiotemporal context", *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 295-302, 2007.

[13] S. Eom, R. Bise, and T. Kanade, "Detection of hematopoietic stem cells in microscopy images using a bank of ring filters", *IEEE International Symposium on Biomedical Imaging*, pp. 137-140, 2010.

[14] L. Miroslaw, A. Chorazyczewski, F. Buchholz, and R. Kittler, "Correlation-based method for automatic mitotic cell detection in phase contrast microscopy", *Advances in Intelligent and Soft Computing*, pp. 627-634, 2005.

[15] S. Tatiraju, and A. Mehta. "Image segmentation using k-means clustering, EM and normalized cuts", *UC Irvine*, 2008.

[16] R. Farnoosh, and B. Zarpak. "Image segmentation using Gaussian mixture model", *International Journal on Engineering and Science*, 19, 29-32, 2008.

[17] N. Lowry, R. Mangoubi, M. Desai, Y. Marzouk, and P. Sammak, "Texton-based segmentation and classification of human embryonic stem cell colonies using multi-stage Bayesian level sets", *IEEE International Symposium on Biomedical Imaging*, pp. 194-197, 2012.

[18] M. Varma, and A. Zisserman, "A statistical approach to texture classification from single images", *International Journal of Computer Vision*, 62(1-2), pp. 61-81, 2005.

[19] N. Lowry, R. Mangoubi, M. Desai, and P. Sammak, "Nonparametric segmentation and classification of small size irregularly shaped stem cell nuclei using adjustable windowing", *IEEE International Symposium on Biomedical Imaging*, pp. 141-144, 2010.

[20] R. Mangoubi, C. Jeffreys, A. Copeland, M. Desai, and P. Sammak. "Non-invasive image based support vector machine classification of human embryonic stem cells", *IEEE International Symposium on Biomedical Imaging*, pp. 284-287, 2007.

[21] M. Desai, R. Mangoubi, and P. Sammak. "Noise adaptive matrix edge field analysis of small sized heterogeneous onion layered textures for characterizing human embryonic stem cell nuclei", *IEEE International Symposium on Biomedical Imaging*, pp. 1386-1389, 2009.

[22] P. J. Sammak, R. Mangoubi, T. M. Erb, S. Mucko, and M. Desai. "Methods of generating trophectoderm and neurectoderm from human embryonic stem cells", *U.S. Patent 9,607,202*, 2017.

[23] H. Niioka, S. Asatani, A. Yoshimura, H. Ohigashi, S. Tagawa, and J. Miyake, "Classification of C2C12 cells at differentiation by convolutional neural network of deep learning using phase contrast images", *Human cell*, 31(1), pp. 87-93, 2018.

[24] Y. H. Chang, K. Abe, H. Yokota, K. Sudo, Y. Nakamura, C. Y. Li, and M. D. Tsai. "Human induced pluripotent stem cell region recognition in microscopy images using convolutional neural networks", *IEEE International Conference on Engineering in Medicine and Biology Society*, pp. 4058-4061, 2017.

[25] W. Xie, J. A. Noble, and A. Zisserman. "Microscopy cell counting and detection with fully convolutional regression networks", *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6(3), pp. 283-292, 2018.

[26] A. Witmer, and B. Bhanu. "Multi-label Classification of Stem Cell Microscopy Images Using Deep Learning", *IEEE International Conference on Pattern Recognition*, 2018.

[27] R. Theagarajan, and B. Bhanu, "DeephESC 2.0: Deep Generative Multi Adversarial Networks for improving the classification of hESC", *PloS one*, 14(3), 2019.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolution neural networks", *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[29] J. Redmon, and A. Farhadi, "YOLO9000: better, faster, stronger", *arXiv preprint*, 2017.

[30] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolution encoder-decoder architecture for image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), pp. 2481-2495, 2017.

[31] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?", *Digital Image Computing: Techniques and Applications*, pp. 1-6, 2016.

[32] Nikon Biostation-IM. http://www.nikoninstruments.com/Products/Cell-IncubatorObservation/BioStation-IM.

[33] S. Lin, S. Fonteno, J.H. Weng, and P. Talbot, "Comparison of the toxicity of smoke from conventional and harm reduction cigarettes using human embryonic stem cells", *Toxicology Science*, 118, pp. 202-212, 2010.

[34] Nikon. *CL-Quant*, http://www.nikoninstruments.com/News/US-News/Nikon-Instruments-Introduces-CL-Quant-Automated-Image-Analysis-Software, 2013.

[35] S. Lin, and P. Talbot, "Methods for culturing mouse and human embryonic stem cells", *Embryonic Stem Cell Therapy for Osteo-Degenerative Diseases. Humana Press*, pp. 31-56, 2011.

[36] B. X. Guan, B. Bhanu, P. Talbot, and S. Lin, "Bio-driven cell region detection in human embryonic stem cell assay", *IEEE Transactions on Computational Biology and Bioinformatics*, 11(3), pp. 604-611, 2014.

[37] I. Durugkar, I. Gemp, and S. Mahadevan, "Generative multi-adversarial networks", *arXiv preprint*, arXiv:1611.01673, 2016.

[38] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint*, arXiv:1412.6980, 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

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

[41] M. Mirza, and S. Osindero, "Conditional generative adversarial nets", *arXiv preprint*, arXiv:1411.1784, 2014.

[42] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolution generative adversarial networks", *arXiv preprint*, arXiv:1511.06434, 2017.

[43] https://www.ranker.com/crowdranked-list/most-popular-american-sports

[44] https://www.statista.com/statistics/267963

[45] https://www.footballscience.net/special-topics/performance-analysis/

[46] C. Lago-Peñas and A. Dellal, "Ball possession strategies in elite soccer according to the evolution of the match-score: the influence of situational variables," *Journal of Human Kinetics*, 25, pp.93-100, 2010.

[47] K. Saito, and M. Yoshimura, "Pass appearance time and pass attempts by teams qualifying for the second stage of FIFA world cup 2014 in Brazil," *Journal of Sports Science*, 4, pp.156-162, 2016.

[48] K. Saito, M. Yoshimura, and T. Ogiwara, "Pass appearance time and pass attempts by teams qualifying for the second stage of FIFA World Cup 2010 in South Africa," *Football Science*, 10, pp.65-69, 2013.

[49] A. Janković, B. Leontijević, M. Pašić, and V. Jelušić, "Influence of certain tactical attacking patterns on the result achieved by the teams participants of the 2010 FIFA World Cup in South Africa," *Physical Culture*, 65(1), pp.34-45, 2011.

[50] A. Redwood-Brown, "Passing patterns before and after goal scoring in FA Premier League Soccer," *International Journal of Performance Analysis in Sport*, 8(3), pp.172-182, 2008.

[51] M. A. Gómez, M. Gómez-Lopez, C. Lago, and J. Sampaio, “Effects of game location and final outcome on game-related statistics in each zone of the pitch in professional football,” *European Journal of Sport Science*, 12(5), pp.393-398, 2012.

[52] A. Scoulding, N. James, and J. Taylor, “Passing in the Soccer World Cup 2002,” *International Journal of Performance Analysis in Sport*, 4(2), pp.36-41, 2004.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788, 2016.

[54] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Detect to track and track to detect,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3038-3046, 2017.

[55] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, “Fully-convolutional siamese networks for object tracking,” *European conference on computer vision*, pp. 850-865, 2016.

[56] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” *IEEE International Conference on Image Processing*, pp. 3645-3649, 2017.

[57] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” *European Conference on Computer Vision*, pp. 740-755, 2014.

[58] Z. Cai, and N. Vasconcelos, “Cascade r-cnn: Delving into high quality object detection,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6154-6162, 2018.

[59] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” *IEEE Conference on Computer Vision*, pp. 2961-2969, 2017.

[60] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *European Conference on Computer Vision*, pp. 21-37, 2016.

[61] P. Parisot, and C. De Vleeschouwer, “Scene-specific classifier for effective and efficient team sport players detection from a single calibrated camera,” *Computer Vision and Image Understanding*, pp.74-88, 2017.

[62] D. J. Duh, S. Y. Chang, S. Y. Chen, and C. C. Kan, “Automatic broadcast soccer video analysis, player detection, and tracking based on color histogram,” *Intelligent Technologies and Engineering Systems*, pp. 123-130, 2013.

[63] J. Liu, X. Tong, W. Li, T. Wang, Y. Zhang, and H. Wang, “Automatic player detection, labeling and tracking in broadcast soccer video,” *Pattern Recognition Letters*, 30(2), pp. 103-113, 2009.

[64] T. K. Chiang, J. J. Leou, and C. S. Lin, “An improved mean shift algorithm based tracking system for soccer game analysis,” *Asia-Pacific Signal and Information Processing Association*, pp. 380-385, 2009.

[65] J. Xing, H. Ai, L. Liu, and S. Lao, “Multiple player tracking in sports video: A dual-mode two-way Bayesian inference approach with progressive observation modeling,” *IEEE Transactions on Image Processing*, 20(6), pp. 1652-1667, 2010.

[66] T. D’Orazio, M. Leo, P. Spagnolo, P. L. Mazzeo, N. Mosca, M. Nitti, and A. Distanter, “An investigation into the feasibility of real-time soccer offside detection from a multiple camera system,” *IEEE Transactions on Circuits and Systems for Video Technology*, 19(12), pp. 1804-1818, 2009.

[67] S. H. Khatoonabadi, and M. Rahmati, “Automatic soccer players tracking in goal scenes by camera motion elimination,” *Image and Vision Computing*, 27(4), pp. 469-479, 2009.

[68] A. Senocak, T. H. Oh, J. Kim, and I. So Kweon, “Part-based player identification using deep convolutional representation and multi-scale pooling,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1732-1739, 2018.

[69] T. Jaakkola, and D. Haussler, “Exploiting generative models in discriminative classifiers,” *Advances in Neural Information Processing Systems*, pp. 487-493, 1999.

[70] J. Xu, L. Kanokphan, and K. Tasaka, “Fast and accurate object detection using image cropping/resizing in multi-view 4K sports videos,” *ACM International Workshop on Multimedia Content Analysis in Sports*, pp. 97-103, 2018.

[71] H. Liu, and B. Bhanu, “Pose-guided R-CNN for jersey number recognition in sports,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[72] M. Istasse, J. Moreau, and C. De Vleeschouwer, “Associative embedding for team discrimination,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[73] W. L. Lu, J. A. Ting, K. P. Murphy, and J. J. Little,“Identifying players in broadcast sports videos using conditional random fields,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3248 - 3256, 2011.

[74] R. Theagarajan, F. Pala, X. Zhang, and B. Bhanu, “Soccer: Who has the ball? Generating visual analytics and player statistics,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1749-1757, 2018.

[75] Z. Cai, H. Neher, K. Vats, D. Clausi, and J. Zelek, “Temporal hockey action recognition via pose and optical flows,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[76] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation using part affinity fields,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291-7299, 2017.

[77] T. W. Hui, X. Tang, and C. Change Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[78] A. J. Piergiovanni, and M. S. Ryoo, 2018. "Fine-grained activity recognition in baseball videos," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1740-1748, 2018.

[79] A. Cioppa, A. Deliège, and M. Van Droogenbroeck, "A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1765-1774, 2018.

[80] M. R. Tora, J. Chen, and J. J. Little, "Classification of puck possession events in ice hockey," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 147-154, 2017.

[81] R. Li, and B. Bhanu, "Fine-grained visual dribbling style analysis for soccer videos with augmented dribble energy image," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[82] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *IEEE International Conference on Computer Vision*, pp. 2961-2969, 2017.

[83] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291-7299, 2017.

[84] B. Bhanu, C. V. Ravishankar, A. K. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos (Eds.), "Distributed video sensor networks", *Springer Science Business Media*, 2011.

[85] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 1979.

[86] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "Survey on player tracking in soccer videos", *Computer Vision and Image Understanding*, 159, pp. 19-46, 2017.

[87] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, 2(1-2), pp. 83-97, 1955.

[88] Y. Deng, P. Luo, C. C. Loy, and X. Tang, "Pedestrian attribute recognition at far distance," *ACM international conference on Multimedia*, pp. 789-792, 2014.

[89] Z. Han, B. Wei, Y. Zheng, Y. Yin, K. Li, and S. Li, "Breast cancer multi-classification from histopathological images with structured deep learning model," *Scientific Reports*, 7(1), 2017.

[90] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, 13(11), pp.2884-2896, 2018.

[91] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen, "Soccer video and player position dataset," *International Conference on Multimedia Systems*, pp. 18-23, 2014.

[92] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248-255, 2009.

[93] R. Wiehagen, and C. H. Smith, "Generalization versus classification," *Journal of Experimental & Theoretical Artificial Intelligence*, 7(2), pp. 163-174, 2007.

[94] O. O. Koyejo, N. Natarajan, P. K. Ravikumar, and I. S. Dhillon, "Consistent binary classification with generalized performance metrics," *Advances in Neural Information Processing Systems*, pp. 2744-2752, 2014.

[95] A. Cioppa, A. Deliége, M. Istasse, C. De Vleeschouwer, and M. Van Droogenbroeck, "ARTHuS: Adaptive real-time human segmentation in sports through online distillation," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[96] https://aws.amazon.com/solutions/case-studies/hudl/

[97] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal processing magazine*, 29(6), pp. 82-97, 2012.

[98] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725-1732, 2014.

[99] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014.

[100] A. Kurakin, I. J. Goodfellow, and S. Bengio. "Adversarial examples in the physical world", *arXiv preprint arXiv:1607.02533*, 2016.

[101] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neuralnetworks", *arXiv preprint arXiv:1312.6199*, 2013.

[102] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time", *European conference on machine learning and knowledge discovery in databases*, pp. 387-402, 2013.

[103] A. Krizhevsky, and G. Hinton, "Learning multiple layers of features from tiny images", *Technical report, Citeseer*, 2009.

[104] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms", *arXiv cs.LG/1708.07747*, 2017.

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

[106] N. Papernot, P. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning", *Asia Conference on Computer and Communications Security*, pp. 506-519, 2017.

[107] N. Carlini, and D. Wagner, "Towards evaluating the robustnessof neural networks", *arXiv preprint arXiv:1608.04644*, 2016.

[108] https://www.youtube.com/watch?v=zQ_uMenoBCk&feature=youtu.be

[109] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety", *arXivpreprint arXiv:1606.06565*, 2016.

[110] A. V. D. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks", *arXiv preprint arXiv:1601.06759*, 2016.

[111] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications", *arXiv preprint arXiv:1701.05517*, 2017.

[112] A. V. D. Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, "Conditional image generation with pixelcnn decoders", *Neural Information Processing Systems*, pp. 4790-4798, 2016.

[113] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *arXiv preprint arXiv:1412.6572*, 2014.

[114] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural net-works", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574-2582, 2016.

[115] Y. Dong, F. Liao, T. Pang, H. Su, X. Hu, J. Li, and J. Zhu, "Boosting adversarial attacks with momentum", *arXiv preprint arXiv:1710.06081*, 2017.

[116] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy", *arXiv preprint arXiv:1805.12152*, 2018.

[117] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses", *arXiv preprint arXiv:1705.07204*, 2017.

[118] D. Warde-Farley, and I. J. Goodfellow, "Adversarial perturbations of deep neural networks", *Perturbations, Optimization,and Statistics*, 2016.

[119] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks", *arXiv preprint arXiv:1511.04508*, 2015.

[120] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks”, *arXiv preprint arXiv:1704.01155*, 2017.

[121] D. Meng, and H. Chen, “MagNet: a two-pronged defense against adversarial examples”, *Conference on Computer and Communications Security*, pp. 135-147, 2017.

[122] P. Samangouei, M. Kabkab, and R. Chellappa, “Defense-GAN: Protecting classifiers against adversarial attacks using generative models”, *arXiv preprint arXiv:1805.06605*, 2018.

[123] J. Buckman, A. Roy, C. Raffel, and I. J. Goodfellow. “Thermometer encoding: One hot way to resist adversarial examples”, *International Conference on Learning Representations*, 2018.

[124] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, “PixelDefend: Leveraging generative models to understandand defend against adversarial examples”, *International Conference on Learning Representations*, 2018.

[125] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming”, *SIAM Journal on Optimization*, 19(4) pp. 1574-1609, 2009.

[126] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, 86(11), pp. 2278-2324, 1998.

[127] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks”, *arXiv preprint arXiv:1706.06083*, 2017.

[128] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, “Learning with a strong adversary”, *arXiv preprint arXiv:1511.03034*, 2015.

[129] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, ”You only propagate once: Accelerating adversarialtraining via maximal principle”, *arXiv preprint arXiv:1905.00877*, 2019.

[130] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, “There is no free lunch in adversarialrobustness (but there are unexpected benefits)”, *arXiv preprint arXiv:1805.12152*, 2018.

[131] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale”, *arXiv preprint arXiv:1611.01236*, 2016.

[132] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, “A study of the effect of jpg compression on adversarialimages”, *arXiv preprint arXiv:1608.00853*, 2016.

[133] N. Das, M. Shanbhogue, S. T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, “Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression”, *arXiv preprint arXiv:1705.02900*, 2017.

[134] N. Carlini, and D. Wagner, "Towards evaluating the robustness of neural networks", *IEEE Symposium on Security and Privacy*, pp. 39-57, 2017.
satc r8S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1765-1773, 2018.

[135] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic seg-mentation and object detection", *IEEE International Conference on Computer Vision*, pp. 1369–1378, 2017.

[136] Q. Wang, W. Guo, K. Zhang, I. Ororbia, G. Alexander, X. Xing, X. Liu, and C. L. Giles, "Learning adversary-resistant deep neural networks", *arXiv preprint arXiv:1612.01401*, 2016.

[137] C. Guo, M. Rana, M. Cisse, and L. V. D. Maaten, "Countering adversarial images using inputtransformations", *arXiv preprint arXiv:1711.00117*, 2017.

[138] E. Raff, J. Sylvester, S. Forsyth, and M. McLean, "Barrage of random transforms for adversariallyrobust defense", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6528–6537, 2019.

[139] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems", *arXiv preprint arXiv:1703.04318*, 2017.

[140] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network", *arXiv preprint arXiv:1503.02531*, 2015.

[141] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models", *arXiv preprint arXiv:1707.05373*, 2017.

[142] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

[143] A. Mustafa, S. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Adversarial defense by restricting the hidden space of deep neural networks", *IEEE International Conference on Computer Vision*, 2019.

[144] J. Gao, B. Wang, Z. Lin, W. Xu, and Y. Qi, "Deepcloak: Masking deep neural network models for robustness against adversarial samples", *IEEE International Conference on Learning Representations*, 2017.

[145] R. Theagarajan, M. Chen, B. Bhanu, J. Zhang, "ShieldNets: Defending Against Adversarial Attacks Using Probabilistic Adversarial Robustness", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6988-6996, 2019.

[146] D. Madaan, and S. J. Hwang, "Adversarial neural pruning", *arXiv preprint arXiv:1908.04355*, 2019.

[147] S. Ye, K. Xu, S. Liu, H. Cheng, J. H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, and X. Lin, “Second rethinking of network pruning in the adversarial setting”, *arXiv preprint arXiv:1903.12561*, 2019.

[148] S. Gu, and L. Rigazio, “Towards deep neural network architectures robust to adversarial examples”, *arXiv preprint arXiv:1412.5068*, 2014.

[149] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser”, *IEEE Conference on Computer Vision and Pattern Recognition* pp. 1778-1787), 2018.

[150] J. Lu, T. Issaranon, and D. Forsyth, “Safetynet: Detecting and rejecting adversarial examples robustly”, *IEEE International Conference on Computer Vision*, pp. 446–454, 2017.

[151] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, “On detecting adversarial perturbations”, *IEEE International Conference on Learning Representations*, 2017.

[152] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, “On the (statistical) detection ofadversarial examples”, *arXiv preprint arXiv:1702.06280*, 2017.

[153] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test”. *Journal of Machine Learning Research*, 13, pp. 723–773, 2012.

[154] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, “Detecting adversarial samples from artifacts”, *arXiv preprint arXiv:1703.00410*, 2017.

[155] A. Rawat, M. Wistuba, and M. I. Nicolae, “Adversarial phenomenon in the eyes of Bayesian deep learning”, *arXiv preprint arXiv:1711.08244*, 2017.

[156] Y. Gal, and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”, *IEEE International Conference on Machine Learning*, pp. 1050–1059, 2016.

[157] G. Goswami, A. Agarwal, N. Ratha, R. Singh, and M. Vatsa, “Detecting and mitigating adversarial perturbations for robust face recognition”, *International Journal of Computer Vision*, 127(6-7), pp. 719–742, 2019.

[158] A. J. Bose, and P. Aarabi, “Adversarial attacks on face detectors using Neural Net based constrained optimization”, *IEEE International Workshop on Multimedia Signal Processing*, pp. 1–6, 2018.

[159] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang, and J. Zhu, “Efficient decision-based black-box adversarial attacks on face recognition”, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7714–7722, 2019.

[160] J. Lu, H. Sibai, and E. Fabry, “Adversarial examples that fool detectors”, *arXiv preprint arXiv:1712.02494*, 2017.

[161] M. A. M. Milton, “Evaluation of momentum diverse input iterative fast gradient sign method (M-DI2-FGSM) based attack method on MCS 2018 adversarial attacks on black box face recognition system”, *arXiv preprint arXiv:1806.08970*, 2018.

[162] Z. Zhou, D. Tang, X. Wang, W. Han, X. Liu, and K. Zhang, “Invisible mask: Practical attacks on face recognition with infrared”, *arXiv preprint arXiv:1803.04683*, 2018.

[163] D. Deb, J. Zhang, and A. K. Jain, “Advfaces: Adversarial face synthesis”, *arXiv preprint arXiv:1908.05008*, 2019.

[164] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets”, *Neural Information Processing Systems*, pp. 2672–2680,2014.

[165] T. Wadhwa, and N. Dhillon, “Defending against attacks on biometrics-based authentication”, *Technical Report*, 2018.

[166] A. Agarwal, R. Singh, and M. Vatsa, “Face anti-spoofing using Haralick features”, *IEEE International Conference on Biometrics Theory, Applications and Systems*, pp. 1–6, 2016.

[167] G. Goswami, N. Ratha, A. Agarwal, R. Singh, and M. Vatsa, “Unravelling robustness of deep learning based face recognition against adversarial attacks”, *AAAI Conference on Artificial Intelligence*, 2018.

[168] A. Agarwal, A. Sehwag, R. Singh, and M. Vatsa, “Deceiving face presentation attack detection via image transforms”, *IEEE International Conference on Multimedia Big Data*, pp. 373–382, 2019.

[169] G. Tao, S. Ma, Y. Liu, and X. Zhang, “Attacks meet interpretability: Attribute-steered detection of adversarial samples”, *Neural Information Processing Systems*, pp. 7717–7728, 2018.

[170] A. Agarwal, R. Singh, M. Vatsa, and N. Ratha, “Are image-agnostic universal adversarial perturbations for face recognition difficult to detect?”, *IEEE International Conference on Biometrics Theory, Applications and Systems*, pp. 1–7, 2018

[171] Y. Zhang, D. Zhao, J. Sun, G. Zou, and W. Li, 11Adaptive Convolutional Neural Network and its application in face recognition”, *Neural Processing Letters*, 43(2), pp. 389–399, 2016.

[172] Y. Rao, J. Lu, and J. Zhou, “Attention-aware deep reinforcement learning for video face recognition”, *IEEE International Conference on Computer Vision*, pp. 3931–3940, 2017.

[173] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, “MS-Celeb-1m: A dataset and benchmark for large-scale face recognition”, *European Conference on Computer Vision*, pp. 87–102, 2016.

[174] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 212–220, 2017.

[175] S. Fortunato, "Community detection in graphs", *Physics reports*, 486(3-5), pp. 75–174, 2010.

[176] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer, "Ensemble methods as a defense to adversarial perturbations against deep Neural Networks", *arXiv preprint arXiv:1709.03423*, 2017.

[177] Z. H. Zhou, "Ensemble methods: Foundations and algorithms", *CRC press*, 2012.

[178] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation", *Neural Information Processing Systems*, pp. 742-751, 2017.

[179] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks", *International Conference on Machine Learning*, 2018.

[180] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing functionality of black-box models", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4954-4963, 2019.

[181] K. Shridhar, F. Laumann, and M. Liwicki, "Uncertainty estimations by softplus normalization in Bayesian convolutional neural networks with variational inference", *arXiv preprint arXiv:1806.05978*, 2018.

[182] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks", *International Conference on Machine Learning*, 2015.

[183] J. M. Hernández-Lobato, and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks", *International Conference on Machine Learning*, pp. 1861-1869, 2015.

[184] S. Kullback, and R. A. Leible, "On information and sufficiency", *The annals of mathematical statistics*, 22(1), pp.79-86, 1951.

[185] N. Narodytska, and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks", *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1310-1318, 2017.

[186] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information", *International Conference on Machine Learning*, 2018.

[187] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: Adversarial patches to attack person detection," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[188] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P. Chen, Y. Wang, and X. Lin, "Evading real-time person detectors by adversarial t-shirt," *arXiv preprint arXiv:1910.11099*, 2019.

[189] S. T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector," *European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 52-68, 2018.

[190] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *International Conference on Learning Representations*, 2017.

[191] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems* 30(9), pp. 2805-2824, 2019.

[192] R. Theagarajan, and B. Bhanu, "Defending Black Box Facial Recognition Classifiers Against Adversarial Attacks," *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[193] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again Neural Networks," *International Conference on Machine Learning*, 2018.

[194] N. Frosst, and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[195] J. Ba, and R. Caruana, "Do deep nets really need to be deep?," *Neural Information Processing Systems*, pp. 2654–2662, 2014.

[196] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Neural Information Processing Systems*, pp. 2994–3003, 2017.

[197] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 23(5), pp.828-841, 2019.

[198] V. Khrulkov, and I. Oseledets, "Art of singular vectors and universal adversarial perturbations," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8562-8570, 2018.

[199] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "ICARL: Incremental classifier and representation learning," *IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001-2010, 2017.

[200] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, 518(7540), pp.529-533, 2015.

[201] Z. Luo, F. Branchaud-Charron, C. Lemaire, J. Konrad, S. Li, A. Mishra, A. Achkar, J. Eichel, and P. M. Jodoin, "MIO-TCD: A new benchmark dataset for vehicle classification and localization," *IEEE Transactions on Image Processing*, 27(10), pp.5129-5141, 2018.

[202] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” *Neural Networks 32* pp. 323-332, 2012.

[203] M. Ren, W. Zeng, B. Yang, and R. Urtasun, “Learning to reweight examples for robust deep learning,” *International Conference on Machine Learning*, 2018.

[204] S. Jenni, and P. Favaro, “Deep bilevel learning,” *European Conference on Computer Vision*, pp. 618-633, 2018.

[205] J. M. Köhler, M. Autenrieth, and W. H. Beluch, “Uncertainty based detection and relabeling of noisy image labels,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 33-37, 2019.

[206] L. Yao, and J. Miller, “Tiny imagenet classification with convolutional neural networks,” *CS 231N*, 2(5), pp. 8, 2015.

[207] J. Deng, Y. Zhou, and S. Zafeiriou, “Marginal loss for deep face recognition,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 60–68, 2017.

[208] F. Wang, L. Chen, C. Li, S. Huang, Y. Chen, C. Qian, and C. C. Loy, “The devil of face recognition is in the noise,” *European Conference on Computer Vision*, pp. 765–780, 2018.

[209] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” *European Conference on Computer Vision*, pp. 499–515, 2016.

[210] S. Fortunato, “Community detection in graphs,” *Physics reports*, 486(3-5), pp. 75–174, 2010

[211] C. Jin, R. Jin, K. Chen, and Y. Dou, “A community detection approach to cleaning extremely large face database,” *Computational Intelligence and Neuroscience*, 2018.

[212] R. Theagarajan, and B. Bhanu, “Defending Black Box Facial Recognition Classifiers Against Adversarial Attacks,” *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.