Deep Learning for Attribute Prediction and Small Sample Size Problems

Student Name: Akshay Sethi
Roll Number: 2014133

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Electronics & Communication Engineering
on April 17, 2018

BTP Track: Research Track

BTP Advisor
Dr. Mayank Vatsa
Dr. Richa Singh

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"Deep Learning for Attribute Prediction and Small Sample Size Problems"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Electronics & Communication Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under the guidance of **Dr. Mayank Vatsa and Dr. Richa Singh**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

.....
Place & Date: Indraprastha Institute of Information Technology, Delhi, 17/04/2018
Akshay Sethi

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.....
Place & Date: Indraprastha Institute of Information Technology, Delhi, 17/04/2018
Dr. Mayank Vatsa and Dr. Richa Singh

Abstract

The first part of this work focuses on facial attribute prediction using a novel deep learning formulation, termed as R-Codean autoencoder. The work presents Cosine similarity based loss function in an autoencoder which is then incorporated into the Euclidean distance based autoencoder to formulate R-Codean. The proposed loss function thus aims to incorporate both magnitude and direction of image vectors during feature learning. Inspired by the utility of shortcut connections in deep models to facilitate learning of optimal parameters, without incurring the problem of vanishing gradient, the proposed formulation is extended to incorporate shortcut connections in the architecture. The proposed R-Codean autoencoder is utilized in facial attribute prediction framework which incorporates patch-based weighting mechanism for assigning higher weights to relevant patches for each attribute. The experimental results on publicly available CelebA and LFWA datasets demonstrate the efficacy of the proposed approach in addressing this challenging problem.

The second part of this work focuses on understanding the limits of data augmentation by synthesized images by Generative Adversarial Networks. Using extensive experiments, we investigate the question of 'how much augmentation is good augmentation', in terms of the ultimate aim of classification. Experiments with multiple GAN architectures and classifiers across datasets further substantiate our findings.

In the final part,we present new architectural and training procedures for Generative Adversarial Networks(GAN)[16] .We first determine whether the probability distribution of data generated by GANs is equal(or close enough) to the probability distribution of the dataset. For this purpose we train Deep Convolutional Network Classifiers[32, 37] on the generated images and compare them with those trained on the dataset. We show that the probability distributions in both the cases differ.We conduct this analysis on the MNIST[36], CIFAR10[31] and Adience[12] datasets. Having shown the above, we suggest changes in the current GAN training algorithm using the subclass information present in datasets. We show improved results using the CIFAR100[31] and Adience(each of these datasets have the subclass information present) and follow the standard protocol defined on these datasets for training the GAN.

Keywords: ..(Autoencoders, Skip Connections, Facial Attributes, Generative Adversarial Networks, Convolutional Neural Network, Deep Neural Network, Sub-Class, Data Augmentation, Generative Modelling)...

Acknowledgments

I would like to express gratitude to my supervisors Dr. Mayank Vatsa and Dr. Richa Singh for their useful comments, remarks and engagement through the learning process of this work. Furthermore I would like to thank Maneet Singh,PhD Scholar IAB Lab for her mentorship in this project.
I would like to thank my loved ones, who have supported me throughout the entire process, both by keeping me harmonious and helping me put all the pieces together. I will be grateful forever to everyone who helped with this project.

Work Distribution

All the work done in this project was done by me.

Contents

Chapter 1

Introduction

Biometrics is the field of study that deals with identifying humans based on their physiological and behavioural traits[28]. The various modalities that are explored for biometric authentication include physiological traits such as face[65], fingerprint[40], iris[29], retina[24], palm-print[33], knuckle print[8], hand geometry[8], and ear[7] and behavioural traits such as gait[54], signature[47], and keystroke dynamics[44]. Among these, fingerprint, iris, face, voice and hand geometry are some of the most important biometric modalities in the real world.[51] Biometric systems can operate in two basic modes:

- 1:1 (called authentication or verification)
- 1: N (one-to-many, called matching or identification)

Biometrics was initially pushed forward by a need for robust security and surveillance applications, but its potential as a natural and effortless means of identification and authentication also paved the way for a host of smart applications that automatically identify the user and provide customized services. With increasing awareness of its psychological, privacy-related and ethical aspects, there is no doubt that biometrics will continue to contribute to many technological solutions of our daily lives. Biometrics relies on the input from a number of fields, starting with various kinds of sensors that are used to sample the biometric. Signal processing and pattern recognition methods are obviously relevant, as the acquired data need to be prepared for accurate and robust decisions. At its final stage, the system outputs a decision, which links the acquired and processed biometric trait to an identity. Algorithms and mathematical models developed by the machine learning community are frequently used in biometric systems to implement the decision function itself. Within the Machine learning community nowadays, Deep Learning has attained impressive performance in many fields such as image classification, semantic segmentation, and image compression. Some of the most successful deep learning methods involve artificial neural networks, such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Deep Belief Networks (DBN)[25] and Stacked AutoEncoder (SAE)[3]. The most important advantage of deep learning is replacing handcrafted features with efficient algorithms for unsupervised or semisupervised feature learning and hierarchical feature extraction. The performance of these deep neural networks has begun to exceed human performance in many tasks. Applications of deep learning includes Natural Language Processing (NLP), Computer Vision, Information Retrieval, Finance and Biometrics.

In particular, Biometric based recognition and authentication has been one of the most successful application of deep learning. Face recognition in particular has improved tremendously

Attributes			
5 o Clock Shadow	×	✓	×
Arched Eyebrows	✓	×	✓
Attractive	✓	✓	✓
Bags Under Eyes	×	✓	×
Bald	×	×	×
Bangs	×	×	✓
Big Lips	×	✓	✓
Big Nose	×	✓	×
Black Hair	×	✓	×
Blond Hair	×	×	×

Figure 1.1: Sample images from the CelebA dataset, along with a few attributes. Each attribute is a binary attribute, having value either a '1' or '0'.

when deep learning based models started to emerge.The recent advances in deep learning for face recognition have made it possible for a machine to outperform human for the first time in history. The Labelled Faces in the Wild (LFW) is a challenging dataset created to realistically assess the performance of face recognition. The best score human has archived in telling whether two faces belong to the same individual or not on LFW database was 97.5% whereas automated approaches have achieved above 99% on the same dataset[55].

In the present work we apply deep learning to various biometrics problems such as attribute prediction, small sample size problems and data augmentation for face recognition.The work has been divided into 3 parts comprising all the core components in deep learning such as using a custom loss function for training the deep network to investigating limits of learnt data augmentation using generative models such as Generative Adversarial Networks(GAN).

In the first part of the work we propose a novel Residual Cosine Similarity and Euclidean Distance based autoencoder, termed as R-Codean autoencoder. Unlike traditional autoencoders, the proposed formulation incorporates both direction and magnitude information at the time of feature extraction along with shortcut connections, thereby resulting in a residual network. To our knowledge this is the first work which incorporates the cosine distance based loss in an autoencoder which has traditionally been used in the information retrieval literature[42]. Furthermore, the R-Codean autoencoder is utilized to present a facial attribute prediction framework. The framework incorporates a patch-based weighting mechanism for providing higher weight to certain face patches for a given attribute. Experimental results on the Celeb Faces Attributes (CelebA) and Labeled Faces in the Wild Attributes (LFWA) datasets [38] illustrate the efficacy of the proposed model by achieving comparable results to existing deep Convolutional Neural Network (CNN) models.

The second part of this research focuses on understanding the limits of data augmentation using GANs, in order to answer the question *'How much augmentation is sufficient?'*. The data generated by various state of the art GAN models is augmented to the original training dataset. Ratio of the real data to the augmented data is varied to analyze the classification performance trends. The key contributions of this work are the Demonstration of the usefulness of data augmentation using generative adversarial networks in a variety of classification tasks, Evaluation of the limit of GAN based data augmentation for improving classification performance. And finally an evaluation of the generalizability of data augmentation across multiple GANs, datasets, and

classifiers.

In the final part of the thesis, we ask ourselves whether the data distribution learnt by GANs is close enough to the real dataset or not and if not in what ways could we improve this process. Guided by this question we claim the contributions of experimentally proving by training various state of the art GANs on datasets like CIFAR10 and classfication of the generated data by using CNNs like VGGNet, Resnet etc to show that true distribution varies significantly from generated data distribution. We then focus on a novel approaches for training Generative Adversarial Networks. The approaches used are the following :

- Changing the latent code of the generator to model each class of the GAN using separate latent code.
- Regularizing the Generator and Discriminator with class based L21 regularization
- Learning a Stack of GANs wherein the output from a previous GAN is given as input to the next GAN
- Instead of learning two layers as above we propose to learn a 3 layered network of GANs calling it DeepGANs

Chapter 2

Facial Attribute Analysis Using Residual Codean Autoencoder

2.1 Introduction

What Does Your Face Shape Say About You? [1]

Is Your Personality Written All Over Your Face? [2]

These are some common questions related to facial attributes. A face image can provide multitude of information such as identity, gender, race, apparent age, and expression. While gender, race, and age estimation from face images are well explored research problems, estimating other attributes such as hair color, nose shape, eye shape, and attractiveness is also getting attention. Predicting facial attributes has several unique applications ranging from focused digital marketing to law enforcement. Facial attribute prediction has an important application in the domain of online marketing, where targeted advertisements or products can be shown to a user based on his/her physical features and appearance. The increasing usage and access of computing devices (laptops, mobile phones) and Internet facilities has also led to the availability of abundant unlabeled data, which requires tagging in order to utilize facial attributes for meaningful tasks. Further, facial attributes can be used as ancillary (or soft) information for face recognition systems, and can help in reducing the identity search space. Existing research in the literature has shown substantial improvement upon incorporating ancillary information for the task of person identification [34, 53, 43]. Fig. 2.1 presents some sample images from the Celeb Faces Attributes (CelebA) Dataset [38]. Each row contains images corresponding to a single attribute. The challenging nature of the problem in terms of high intra-class variations can be observed from these sample images.

[1] http://tinyurl.com/k5t6rh7
[2] http://tinyurl.com/lpcz5jb

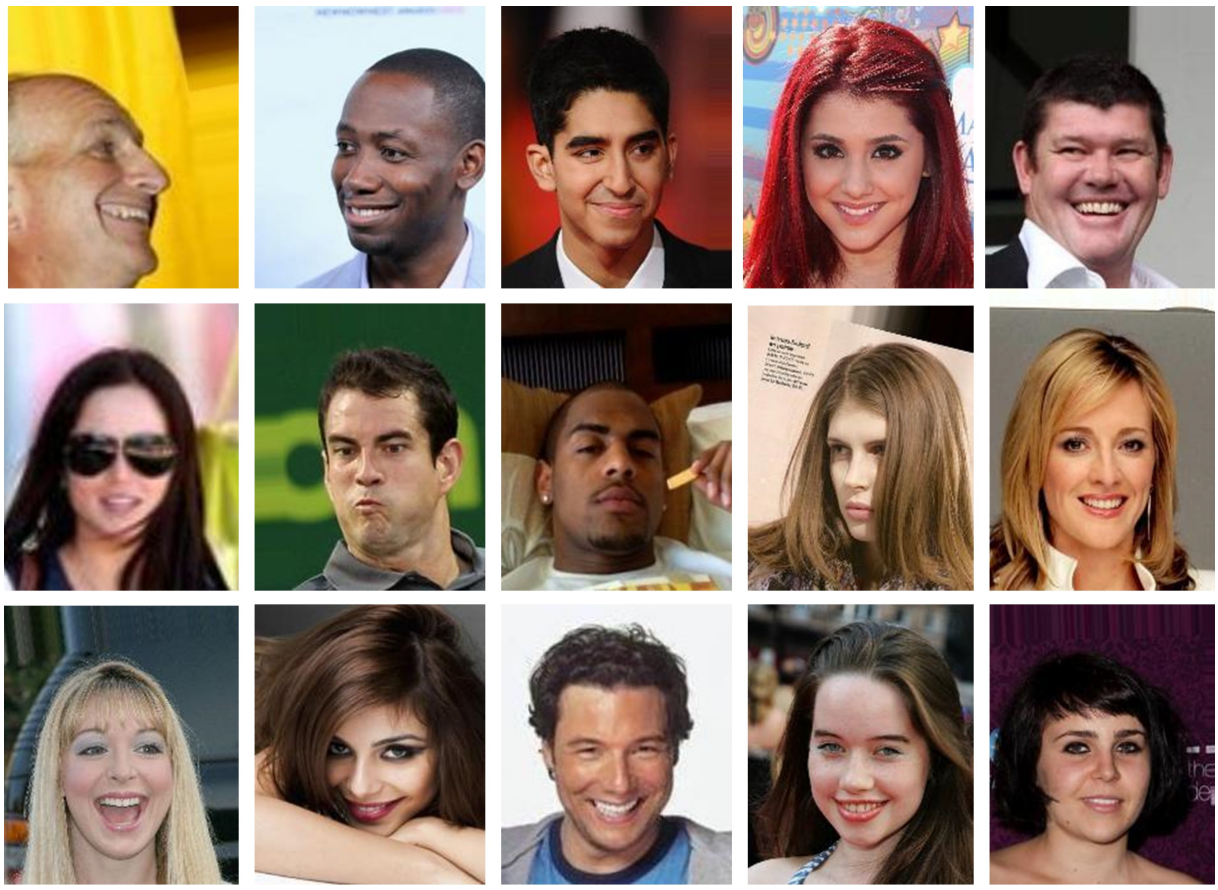

Figure 2.1: Sample images from the CelebA dataset [38] demonstrating intra-class variations for different attributes. The first row corresponds to the *smile*, second to *young*, and third to *attractive*.

Table 2.1: Literature review of facial attribute prediction algorithms.

Authors	Algorithm	Datasets Used	Classification Accuracy (Avg.)
Kumar et al.[34]	Hand crafted features and RBF SVMs as classifiers	LFW	83.62%
Chung et al.[10]	Supervised DBN for attribute classification	LFW	86.00%
Berg et al.[5]	Local part based features termed as POOF and linear SVMs	LFW	83.00%
Luo et al.[39]	Sum Product Network for prediction of attributes	LFW	87.90%
Liu et al.[38]	Image localization by deep CNN, followed by another deep CNN for prediction	CelebA, LFWA	87.00%, 84.00%
Huang et al.[26]	CNN based model to address the problem of class imbalance during training	CelebA	84.00%
Ehrlich et al. [11]	Multi Task RBMs for learning a joint feature representation for attribute classification	CelebA	87.00%
Wang et al.[61]	Siamese Network minimizing loss for face verification, followed by fine-tuning on CelebA dataset	CelebA, LFWA	88.00%, 87.00%
Zhong et al.[66]	Features extracted from off-the shelf deep CNN models and classified using linear SVMs	CelebA, LFWA	86.60%, 84.70%
Zhang et al.[67]	Features extracted from intermediate layers of deep CNN and classified using linear SVMs	CelebA, LFWA	89.80%,85.90%
Rosza et al.[49]	Treats attribute classification as a regression problem and uses MSE loss to train a VGG-16 topology CNN	CelebA	90.80%
Rudd et al.[50]	CNN based model to perform multi task optimization with class imbalanced training data	CelebA	90.94%
Hand et al.[21]	Multi-task deep CNN which also models attribute relationships.	CelebA, LFWA	Avg. accuracy not reported
Proposed (2018)	R-Codean AE: Incorporates Euclidean distance and Cosine similarity in the loss function, along with residual learning	CelebA, LFWA	90.14%, 84.80%

2.1.1 Related Work

[34], in one of the initial works on facial attribute analysis, extracted hand crafted features such as edge magnitudes and gradient directions from facial regions and used the feature vector as input to a Support Vector Machine for attribute classification. Later, [5] proposed to extract local part based features termed as POOF followed by linear SVMs for each attribute. In 2012, deep learning was explored by [10], where a deep attribute network was built over Deep Belief Networks trained in a supervised manner. Later, [38] proposed a two stage training approach for addressing the task of facial attribute prediction. Given an unconstrained face image, localization was performed using a deep Convolutional Neural Network (CNN) trained in a weakly supervised manner, followed by another CNN for learning feature representation and classification. In 2016, [61] proposed a Siamese network which aimed at minimizing the error for the task of face verification. Inspired by the advancements in deep learning models, [66, 67] also demonstrated that off-the-shelf features learned by CNN models pre-trained on massive facial identities, can effectively be adapted for attribute classification. The learned representations are provided as input to a binary linear SVM trained directly for all levels of representations to classify face attributes. The task of facial attribute prediction has also been posed as regression problem [49], where the authors adopt a 16 layer VGG topology while minimizing the mean squared error loss. [26] presented an approach for learning deep representation of class imbalanced data by incorporating triplet-header hinge loss in CNNs. Parallely, [50] proposed a custom loss function for multiple attributes using a single Deep Convolutional Neural Network. The authors utilized the VGG-16 topology and obtained state-of-the-art classification results. Recently, [21] also proposed deep multi task CNNs for performing attribute classification. Some existing techniques for facial attribute prediction have been summarized in Table 2.1.

2.1.2 Research Contributions

In this research, we propose a novel deep learning formulation for facial attribute prediction in the wild. The key contributions of this research are as follows:

- A novel Residual Cosine Similarity and Euclidean Distance based autoencoder, termed as **R-Codean** autoencoder is proposed. Unlike traditional autoencoders, the proposed formulation incorporates both direction and magnitude information at the time of feature extraction along with shortcut connections, thereby resulting in a residual network,
- The proposed R-Codean autoencoder is utilized to present a facial attribute prediction framework. The framework incorporates a patch-based weighting mechanism for providing higher weight to certain face patches for a given attribute,
- Experimental results on the Celeb Faces Attributes (CelebA) and Labeled Faces in the Wild Attributes (LFWA) datasets [38] illustrate the efficacy of the proposed model by achieving comparable results to existing deep Convolutional Neural Network (CNN) models.

The remainder of this paper is organized as follows: the following section presents the proposed R-Codean autoencoder, followed by the proposed framework for facial attribute prediction in Section 3. Section 2.4 provides the details about the experimental protocol and evaluations, which is followed by the conclusions of this research.

2.2 Proposed Residual Cosine Euclidean Autoencoder

The proposed R-Codean autoencoder is built using a custom loss function which combines the traditionally used Euclidean distance measure with the Cosine similarity. The proposed loss function ensures that the model minimizes the loss between the input and the reconstructed sample not only in terms of *magnitude*, but also in terms of *direction*. To the best of our knowledge, this is the first work which incorporates cosine loss into the feature learning process of an autoencoder. The proposed model also incorporates residual shortcut connections [22] of two kinds (*symmetric* and *cross*) into an autoencoder.

2.2.1 Euclidean Distance based Autoencoder

The autoencoder model is an unsupervised deep learning architecture used for learning representations of the given data [60]. It is a special kind of neural network, where the input is also the target output. The objective of the model is to learn representations, such that the model is able to reconstruct the input from the learned representation. A single layer autoencoder consists of an encoding weight matrix ($\mathbf{W_e}$) and a decoding weight matrix ($\mathbf{W_d}$). For an input sample x, the loss function of an autoencoder is thus formulated as:

$$\mathbf{W_d}, \mathbf{W_e} \, \| x - \mathbf{W_d} \, \phi(\mathbf{W_e} x) \, \|_2^2 \tag{2.1}$$

where, ϕ corresponds to a non-linear activation function such as *sigmoid* or *tanh*. Here, $\mathbf{W_e}x$ corresponds to the learned representation for the input x. The decoding weight matrix $\mathbf{W_d}$ can optionally be constrained by $\mathbf{W_d} = \mathbf{W_e}^T$, in which case the autoencoder is said to have tied weights. The above loss function corresponds to the mean squared error, or the Euclidean distance based loss for the autoencoder. Based on the above loss function, the model aims to minimize the reconstruction error (i.e. the error between the input x and the reconstructed sample ($\mathbf{W_d}\phi(\mathbf{W_e}x)$) in order to learn meaningful representations. Fig. 2.2 presents a single layer autoencoder with x as input, z as the representation, and x' as the reconstructed sample. In an attempt to learn richer feature representations, Stacked Autoencoders [60] are proposed, where multiple autoencoders are stacked on top of each other. The representation learned by the first autoencoder is provided as input to the second autoencoder which further learns higher level features. Due to the large number of parameters, deep autoencoders or stacked autoencoders are often trained in a greedy layer-by-layer manner, where only a single autoencoder is trained at a time [4].

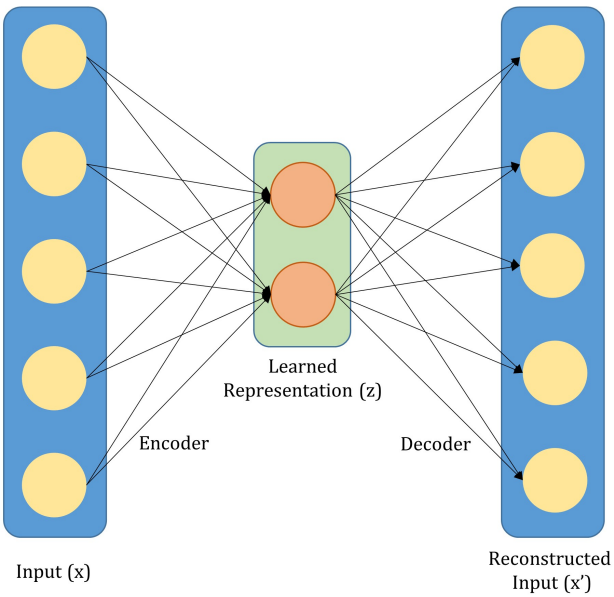

Figure 2.2: Diagrammatic representation of a single layer autoencoder with input x, learned representation z, and reconstructed input x'.

2.2.2 Cosine Similarity based Autoencoder

Cosine similarity based minimization techniques have extensively been used in document modeling and retrieval [18, 35]. Document retrieval techniques aim to model the underlying distribution of keyword occurrences, as opposed to the actual count of the words. A similar analogy can be drawn for structured images such as faces, where cosine based similarity measure can be used for modeling the underlying distribution of the pixel values, as opposed to calculating the distance between their actual pixel intensities. To further explain, Fig. 2.3 presents four face images of the same subject. We compute the corresponding Euclidean distance (D_e) and Cosine distance (D_c)[3] between the pixel values of image pairs. Since the values correspond to the distance scores, a smaller value represents more similar images. It can be observed that when the images are of the same intensity range and contain pose variations, Euclidean distance is able to model their similarity more effectively as compared to Cosine distance. On the other hand, when two images have variations in the intensity range, even with same pose, Euclidean distance is not able to encode the similarity. However, Cosine distance is able to model the similarity perfectly by producing a distance score of 0.00. Inspired by these observations, along with the properties of the Cosine similarity metric and its applicability in high dimensional feature space, we extend the formulation of traditional autoencoder to Cosine similarity based autoencoder. Cosine similarity based loss function helps the model to learn direction based features (built on the underlying distribution of pixel values) as opposed to the magnitude (pixel intensities). The loss function of a Cosine similarity based autoencoder is defined as:

[3]Cosine similarity is converted into a distance measure as $D_c = 1 - S_c$, where S_c is the Cosine similarity.

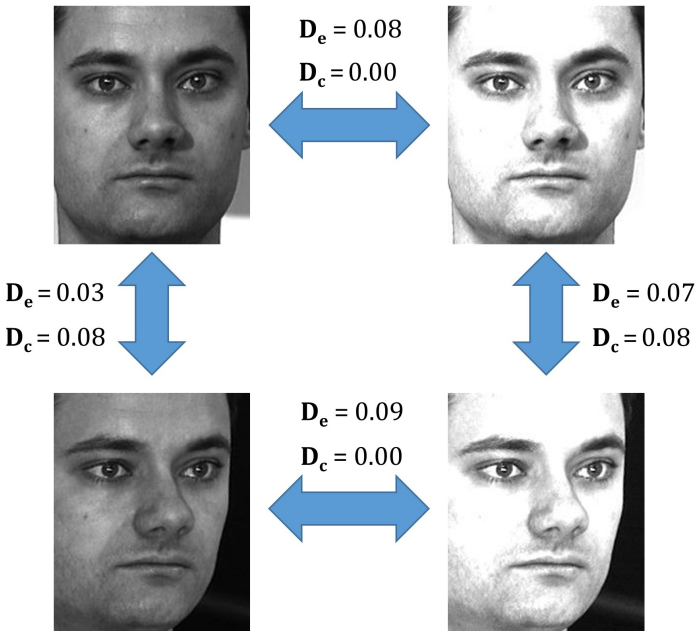

Figure 2.3: Sample images illustrating the effectiveness of Euclidean distance and Cosine similarity under varying conditions. D_e corresponds to the Euclidean distance between two images, while D_c corresponds to the cosine distance. Cosine distance is computed as $1 - Cosine\ Similarity$. It can be observed that for samples having slight pose variations, with constant illumination, the Euclidean distance is able to model the similarity well, however, in case of illumination variations, the Cosine similarity is able to encode the similarity better. A lower distance measure corresponds to a higher similarity.

$$\ell_{Cos} = -\frac{x \cdot (\mathbf{W_d}\phi(\mathbf{W_e}x))}{\|x\|_2^2 \times \ \|\mathbf{W_d}\phi(\mathbf{W_e}x)\|_2^2} \tag{2.2}$$

where, x is the input to the autoencoder and $\mathbf{W_d}\phi(\mathbf{W_e}x)$ is the reconstructed input learned by the autoencoder. As mentioned previously, $\mathbf{W_e}$ and $\mathbf{W_d}$ represent the encoding and decoding weights of the autoencoder, respectively. It is important to note that since the Cosine metric is a *similarity* metric, a negative sign has been incorporated in the loss function in order to decrease the overall distance, or reconstruction error of the model. The above equation minimizes the angle between the input and it's reconstruction by minimizing the cosine distance between the two. This model is especially useful for handling images with brightness or contrast variations.

2.2.3 Proposed R-Codean Autoencoder

In order to learn feature representations based on both direction and magnitude, we combine the Euclidean distance based loss function with the Cosine similarity based loss function, and a Cosine Euclidean (Codean) Autoencoder is proposed. The loss function of the model is formulated as:

$$\ell_{Codean} = \alpha \times \ell_{Euc} + \beta \times \ell_{Cos} + \ \lambda R \tag{2.3}$$

here, the first term corresponds to the Euclidean loss (Eq. 2.1), the second term refers to the Cosine loss (Eq. 2.2), and the third term corresponds to a regularization constraint. α, β,

and λ are the regularization parameters controlling the weight given to each individual term. The above equation helps the autoencoder learn a representation which minimizes both, the magnitude by the Euclidean loss and the direction by the Cosine similarity between the input and reconstructed sample. As explained previously (from Fig. 2.3), the Euclidean distance ensures that slight variations in pose in the reconstructed sample are handled by the autoencoder, while the Cosine distance handles illumination variations. For a single layer model, with input x, Eq. 2.3 with ℓ_1-norm regularization on the encoding weight matrix, can be expanded as follows:

$$\begin{aligned}\ell_{Codean} = \alpha \times \|x - \ (\mathbf{W_d}\phi(\mathbf{W_e}x))\|_2^2 - \beta \times \frac{x \cdot (\mathbf{W_d}\phi(\mathbf{W_e}x))}{\|x\|_2^2 \times \ \|\mathbf{W_d}\phi(\mathbf{W_e}x)\|_2^2} \\ +\lambda \times \|\mathbf{W_e}\|_1\end{aligned} \tag{2.4}$$

The ℓ_1-norm regularization helps learn sparse feature representations for the given input, thereby retaining meaningful latent variables during the feature learning process. The above equation depicts a single layer Codean autoencoder, which can easily be extended for k layers as follows:

$$\begin{aligned}\ell_{Codean} = \alpha \times \|x - g \circ f(x)\|_2^2 \ - \ \beta \times \frac{x \cdot (g \circ f(x))}{\|x\|_2^2 \times \ \|g \circ f(x)\|_2^2} \\ +\lambda \times \sum_{i=1}^{k} \left\|\mathbf{W_e^i}\right\|_1\end{aligned} \tag{2.5}$$

where, $f(x)$ is the encoder function, such that $f(x) = \mathbf{W_e^k}...\phi(\mathbf{W_e^2}(\phi(\mathbf{W_e^1}x)))$ and $g(x)$ is the decoder function, such that $g(x) = \mathbf{W_d^1}(...(\mathbf{W_d^{k-1}}(\mathbf{W_d^k}x)))$. Since the above loss function contains large number of parameters, training the entire model together results in the problem of vanishing gradients. Further, inspired by the analysis that adding shortcut connections facilitates learning of deeper networks, along with resulting in the network imitating the performance of multiple shallow networks [59], we next propose to incorporate shortcut connections in Codean to learn robust feature representations.

2.2.4 Residual Learning in Codean Autoencoder: R-Codean

Residual deep learning framework is introduced by [22] for Convolutional Neural Networks. The authors have observed that learning deeper models resulted in a higher error rate, as compared to their shallower counterpart. Since the aim of learning deeper models is to learn higher level features with each increasing layer, shortcut (or skip) connections are introduced in an attempt to reduce the overall error and facilitate better learning. The introduction of shortcut connections in deep networks thus results in creation of residual networks. Let $H(x)$ be the mapping to be learned by a model on an input x. Now, instead of learning $H(x)$, the residual network is made to learn the mapping $F(x)$, where $F(x)$ is given by:

$$F(x) = H(x) - x \tag{2.6}$$

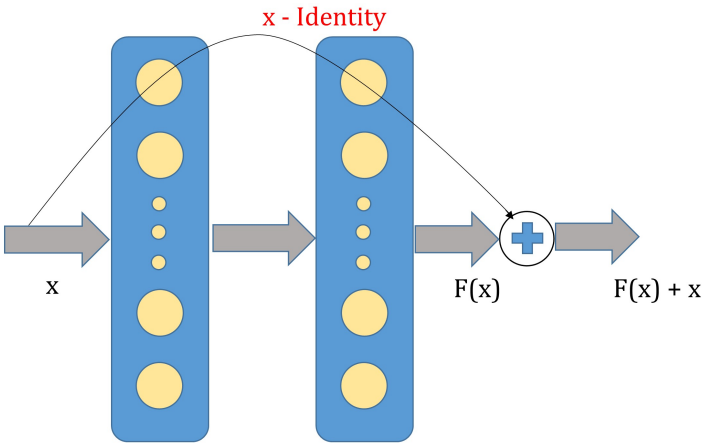

Figure 2.4: Shortcut connection between two layers, representing the concept of residual learning.

The input (x) is then added back to $F(x)$, effectively learning $H(x)$. Residual learning as described above helps to overcome the input degradation (or vanishing gradient) problem in deep networks. In the i^{th} layer of a neural network having weight matrix as $\mathbf{W}_i$, the residual learning framework can be incorporated with a building block defined as:

$$y = F(x, \mathbf{W}_i) + x \tag{2.7}$$

here, x and y are the input and output vectors of the i^{th} layer. The function $F(x, \mathbf{W}_i)$ represents the mapping to be learned. Fig. 2.4 presents a sample residual network of two layers.

In the proposed Codean autoencoder model, two types of shortcut (or skip) connections have been introduced in order to create the proposed Residual Codean (R-Codean) Autoencoder. The concept of residual learning has been incorporated in the autoencoder model by including *cross* and *symmetric* shortcut connections. As shown in Fig. 2.5, cross shortcut connections are overlapping connections which are made between the alternate layers of the network. Such connections help in preventing the input degradation problem in deep networks. Symmetric skip connections are non-overlapping connections created at a larger distance between the encoder and decoder of the same autoencoder. Symmetric shortcut connections help in passing the image details forward, thereby improving the reconstruction process of the autoencoder. Both these connections help in improving the gradient flow which further enables the model to converge to the optimal parameters. Incorporating these shortcut connections in the Codean autoencoder model described above results in the proposed R-Codean Autoencoder for learning robust feature representations.

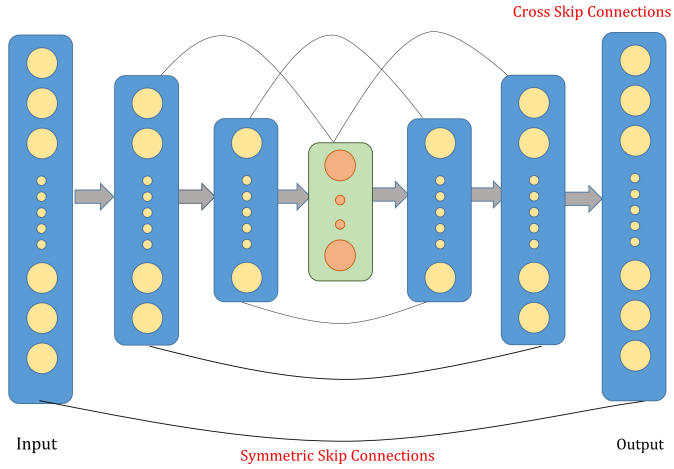

Figure 2.5: Cross and symmetric shortcut (skip) connections in the proposed residual autoencoder model.

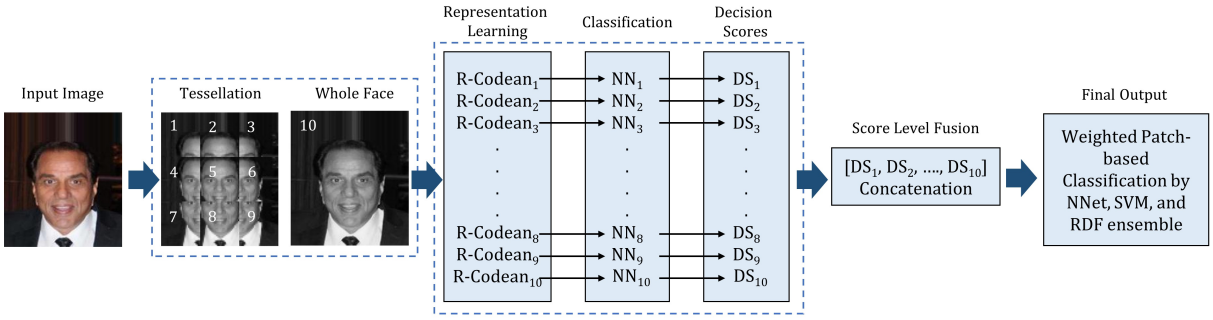

Figure 2.6: Illustrating the steps involved in the proposed attribute classification pipeline built upon the proposed Residual Cosine Euclidean (R-Codean) Autoencoder.

2.3 Proposed Facial Attribute Prediction Framework using Residual Codean Autoencoder

The proposed R-Codean autoencoder is utilized for facial attribute prediction. Fig. 2.6 presents the block diagram of the proposed facial attribute prediction framework. The entire pipeline can broadly be divided into the following components: (i) pre-processing, (ii) feature extraction using R-Codean autoencoder, and (iii) classification. Each step of the pipeline is explained in detail in the following subsections.

2.3.1 Pre-processing of Images

For a given input sample, the image is geometrically normalized and loose cropped to obtain the face region. The cropped image is down-sampled to 64×64 and converted to grayscale. In literature, it has often been observed that facial features are encoded both locally, and globally while performing face recognition [62, 6]. Inspired by these findings, a similar approach is followed for encoding facial attributes by tessellating the input image into nine equal overlapping patches (as shown in Fig. 2.6). The individual patches as well as the entire image are then used

for feature extraction.

2.3.2 Feature Extraction via Proposed R-Codean Autoencoder

The proposed R-Codean autoencoder is used for the task of feature extraction. One R-Codean autoencoder model is trained for each face image patch, and one for the entire image, thereby resulting in ten independent models (nine patches + 1 full face). Patch based feature learning enables the model to learn specific characteristic of a particular facial region. These local features can then be utilized for predicting face component-specific attributes. Coupled with the learned representation over the entire face image, the ten models provide a holistic (global) as well as local representation of the input sample.

2.3.3 Weighted Patch-based Classification

A two-step classification approach is followed in the proposed facial attribute prediction framework. In the first stage, for each R-Codean autoencoder trained on each patch location in the previous step, a two layer neural network classifier is learned using *sigmoid* activation function in the output layer. Therefore, ten neural networks are trained, one corresponding to each R-Codean autoencoder, i.e., one for each facial patch (9) and one for the full face. k probability scores are obtained from the output layer of each neural network, where k corresponds to the number of facial attributes in the dataset. In the second stage, score fusion is performed by concatenating the scores obtained from the ten neural networks to create a feature vector of length $10 \times k$. This feature vector is then provided as input to an ensemble of Neural Network, Random Decision Forest, and Support Vector Machine for obtaining the final attribute classification. In order to emulate the human tendency of focusing on specific regions for certain attributes, a weighted patch-based mechanism is also incorporated in the classification framework. Patch-based weights are learned for each attribute, such that relevant patches are given higher weights for a given attribute. For instance, in case of *wearing necktie* attribute, patches 7, 8, and 9 (shown in Figure 2.6) might contribute higher in the decision making process, as opposed to the other patches. Finally, max-voting is performed on the outputs of the three classifiers in order to obtain the final decision.

2.3.4 Implementation Details

For all experiments, detected and normalized face images are resized to 64×64 and converted to grayscale. Images are tessellated in 3×3 overlapping patches of dimension 32×32. 10 R-Codean autoencoders having three hidden layers with dimensions $[l, l, l]$, are trained on the face patches (and full face). k corresponds to the dimension of the input vector. ℓ_1-norm regularizer with $\lambda = 0.01$ is incorporated for learning a sparse representation of the data. Each R-Codean autoencoder also incorporates cross shortcut connections between the first and third encoding layers, second encoding and first decoding layer, third encoding and second decoding

layer. Symmetric shortcut connections are also introduced between the first encoding and third decoding layer, second encoding and decoding layer, third encoding and first decoding layer. R-Codean is optimized using the adam optimizer [30] with a decaying learning rate. The initial learning rate was set to 0.001, which decayed by a factor of ten whenever the training loss stagnated. For each R-Codean autoencoder, a two hidden layer neural network of dimension $[\frac{l}{2}, \frac{l}{4}]$ is learned in the first classification stage. The autoencoders and neural networks utilize *ReLU* activation function in their layers. The autoencoders are implemented in Python based Keras framework [9] and the classifiers in Scikit-learn library [46]. The source code will be made publicly available for the research community.

Table 2.2: Comparison with state-of-the-art and existing methods on CelebA and LFWA datasets. Accuracy corresponds to the mean accuracy obtained over all the attributes.

Architecture	**CelebA**	**LFWA**
Stacked Autoencoder + NNET	85.60%	76.22%
3-layer CNN + NNET	87.39%	82.37%
VGG-Face [45] + NNET	85.30%	81.04%
Fine-tuned VGG-Face [45] + NNET	87.45%	80.14%
ResNet [22] + NNET	84.03%	76.94%
Fine-tuned ResNet [22] + NNET	86.23%	78.98%
Zhong et al.[66]	86.80%	84.70%
Liu et al.[38]	87.00%	84.00%
Wang et al.[61]	88.00%	**87.00%**
Zhong et al.[67]	89.80%	**85.90%**
Rozsa et al.[49]	**90.80**%	-
Rudd et al.[50]	**90.94**%	-
Proposed	**90.14**%	**84.90%**

2.4 Experimental Results and Analysis

The proposed approach is evaluated on large-scale Celeb Faces Attributes (CelebA) and Labeled Faces in the Wild Attributes (LFWA) datasets [38]. The CelebA dataset contains 10,000 identities, each of which have twenty images. Therefore, the dataset contains a total of 2,00,000 images. LFWA contains 13,233 images pertaining to 5,749 subjects. Each image in both the datasets is annotated with forty binary face attributes such as Male, Young, Bangs, Gray Hair, and five facial landmark key points. Fig. 1.1 presents some sample images of CelebA dataset.

The efficacy of the proposed model has been evaluated on existing commonly used experimental protocols [38, 58, 20, 67]. For the CelebA dataset, the entire dataset is partitioned into

Table 2.3: Attribute classification accuracy (%) of all forty attributes of the CelebA dataset using the proposed facial attribute prediction framework.

Attribute	**Accuracy**	**Attribute**	**Accuracy**	**Attribute**	**Accuracy**
5'o' Clock Shadow	92.88	Arched Eyebrows	81.63	Attractive	79.67
Bags Under Eyes	83.15	Bald	99.52	Bangs	94.51
Big Lips	79.89	Big Nose	83.67	Black Hair	84.80
Blond Hair	94.97	Blurry	96.57	Brown Hair	82.97
Bushy Eyebrows	91.36	Chubby	95.51	Double Chin	96.45
Eyeglasses	98.18	Goatee	96.77	Gray Hair	97.92
Heavy Makeup	89.72	High Cheekbones	86.74	Male	95.87
Mouth S. Open	89.81	Mustache	96.31	Narrow Eyes	90.64
No Beard	94.64	Oval Face	76.55	Pale Skin	96.93
Pointy Nose	76.95	Receding Hairline	93.64	Rosy Cheeks	95.32
Sideburns	97.60	Smiling	92.83	Straight Hair	81.19
Wavy Hair	75.42	Wearing Earrings	82.65	Wearing Hat	97.93
Wearing Lipstick	91.96	Wearing Necklace	89.82	Wearing Necktie	95.88
Young	86.63	-	-	-	-

three parts, where the first 1,60,000 images are used to train the autoencoders, and next 20,000 images are used to train the ensemble classifier. The images of the remaining 1,000 identities (with 20,000 images) are used for testing. On the other hand, LFWA dataset is divided into two, one partition is used for training, while the second forms the test set.

2.4.1 Comparison with Other Deep Learning Models

The proposed facial attribute framework, built upon the proposed R-Codean autoencoder yields an overall mean classification accuracy of **90.14**% and **84.90%** over the 40 attributes of CelebA and LFWA datasets, respectively. Table 2.2 presents the mean accuracy of the proposed framework, along with other comparative architectures. In order to compare the performance of other deep model architectures, experiments are performed using a Stacked Autoencoder, Convolutional Neural Network (CNN), and existing state-of-the-art CNN models of ResNet [22] and VGG-Face [45], while using a neural network for classification. For VGG-Face and ResNet, comparison has been performed with both pre-trained models, as well as fine-tuned models. Fine-tuning is performed on the pre-trained models using the training partition of each dataset, while the Stacked Autoencoder model is trained from scratch with the available training data. It can be observed that the proposed R-Codean autoencoder based pipeline achieves an improvement of more than 4% on the CelebA and 7% on the LFWA database, as compared to stacked autoencoders. The proposed pipeline also presents improved results as compared to several existing state-of-the-art CNN models (with and without fine-tuning). Specifically, an improvement

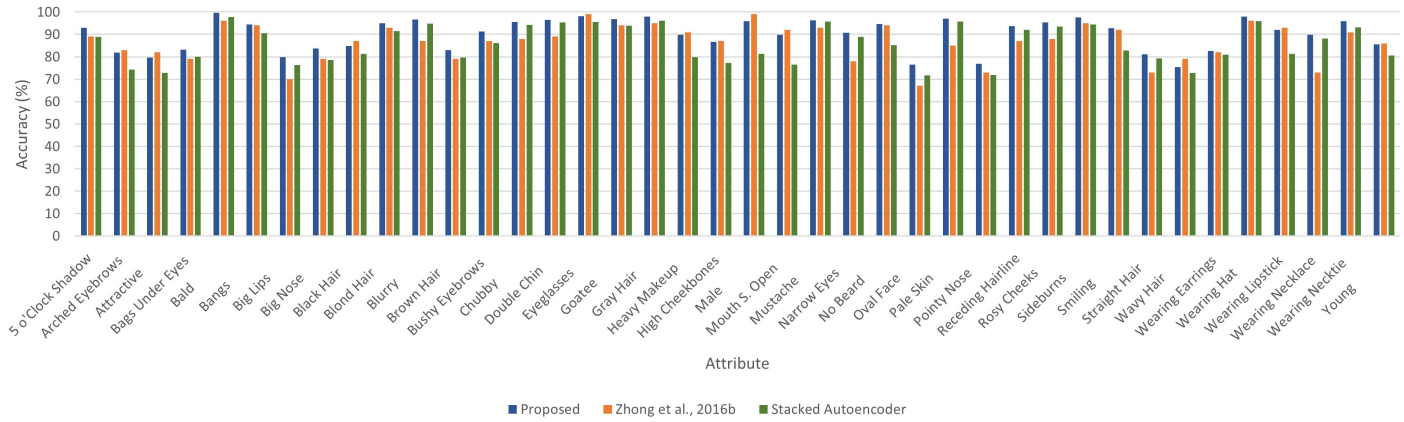

Figure 2.7: Bar graph illustrating the performance of the proposed framework in comparison with current state-of-the-art technique [67] and stacked autoencoder. Accuracies have been reported for all 40 attributes for the CelebA dataset.

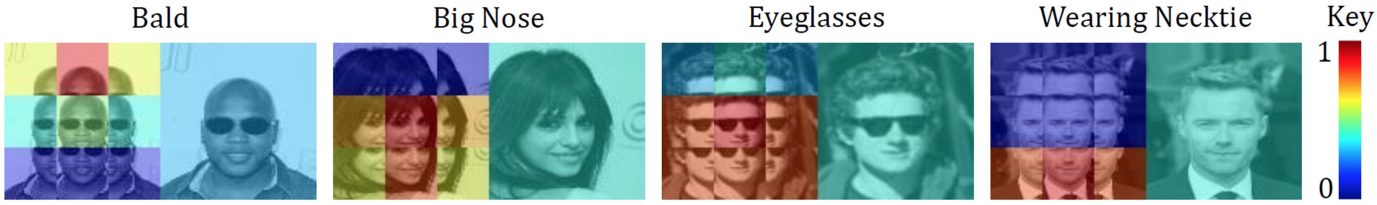

Figure 2.8: Samples illustrating the effect of weighted patch-based attribute classification. For certain attributes, specific regions are more focused and hence more weights are assigned (learned). For example, in case of *bald*, high weight is associated with patches containing the head region of the face image, while low weight is given to other parts of the image.

of more than 4% is observed in comparison to the ResNet model, along with similar results for VGG-Face features.

2.4.2 Comparison with State-of-the-Art Techniques

Comparison has also been performed with existing state-of-the-art architectures present in the literature. Table 2.2 presents the accuracies of recently proposed architectures, taken directly from their publications. It can be observed that the proposed R-Codean autoencoder based approach achieves a comparable mean classification accuracy with respect to the current state-of-the-art approach [50] on the CelebA dataset. It is important to note that while the existing architectures incorporate Convolutional Neural Networks in their pipeline, this is the first work achieving comparable classification performance using autoencoders. Training R-Codean autoencoder requires only 20 seconds per epoch on a workstation powered with NVIDIA K20 GPU and 64GB RAM. Moreover, an unseen input sample takes less than a second for the entire attribute prediction pipeline. Table 2.3 provides the individual accuracy of each attribute obtained on the CelebA dataset. Fig. 2.9 presents the bar graph, depicting comparison with [67] and stacked autoencoder features over all 40 attributes. Since the attribute-specific accuracy is not provided by [49] or [50], attribute wise comparison is not possible.

Table 2.4: Evaluation of the proposed R-Codean based framework for facial attribute classification on CelebA dataset.

Architecture	**Accuracy**
Effect of Pre-Processing	
Learning a Full Face Model Only	86.86%
Learning Patch-based Models Only	88.16%
Effect of Shortcut Connections	
With Symmetric Connection only	87.90%
With Cross Connection Only	88.81%
With No Shortcut Connections	87.60%
Effect of Loss Function	
With Euclidean Distance (MSE) only	88.30%
With Cosine Distance only	85.10%
Effect of Patch-based Weighting	
Without Patch-based Weighting	89.42%
Effect of Classification Ensemble	
Support Vector Machine Only	88.81%
Neural Network Only	88.30%
Random Decision Forest Only	88.84%
Proposed Framework with R-Codean	**90.14%**

2.4.3 Analysis of the Proposed Facial Attribute Prediction Pipeline

In order to thoroughly evaluate the proposed R-Codean autoencoder based pipeline for facial attribute prediction, experiments have been performed on the CelebA dataset to evaluate each component of the same. Table 2.4 presents the mean classification accuracies for different variations of the proposed framework. To re-iterate, in the pre-processing component face tessellation is performed, and models are trained for both full face, as well as independent patches, in order to encode both local as well as global features. The framework is analyzed by learning features on only the full face *or* the tessellated patches, instead of both. As can be observed from Table 2.4, both the techniques independently yield lower results as compared to their combination, thereby strengthening our hypothesis of utilizing both local and global features for facial attribute prediction.

Further evaluations are performed on the R-Codean autoencoder model by understanding the effect of shortcut connections, and the combined loss function. As can be seen from Table 2.4, removing residual shortcut connections from the model results in a drop of around 3%. A similar drop in accuracy can be observed upon incorporating only a single kind of shortcut connection as opposed to both. In order to evaluate the efficacy of the loss function of R-Codean autoencoder, where the loss function is a combination of Euclidean distance (MSE), as well as the Cosine distance, comparison has been performed with models having only Cosine or Euclidean distance based loss function as well. An increase of at most 5% is observed upon incorporating the magnitude (Euclidean distance) as well as the direction (Cosine distance) in the feature learning process. At the classification level, the proposed framework utilizes an ensemble of Neural Network, Support Vector Machine, and Random Decision Forest. It can be observed from Table 2.4 that upon utilizing each of these classifiers independently, the framework does not yield optimal results. An improvement of close to 1.5% is observed upon using the proposed ensemble. Comparison has also been performed to understand the effect of training data size on the proposed framework. As can be observed from Fig. 2.9, even when only 50% of the entire training set is used, the proposed framework yields a classification accuracy of 87.83%, showcasing a drop of only 2.3%. This motivates the utility of the proposed R-Codean based framework for less training data as well.

To show the effect of learning weights in weighted patch-based classification, Fig. 2.8 illustrates some sample attributes, along with the associated weights. The values of weights lie between $0 - 1$ and the illustration is color coded, *red* corresponds to 1 and *blue* corresponds to 0. It can be observed that the learned weights are intuitive and the patches which relevant to the given attribute are selected. For instance, in case of *Wearing Necktie*, high weight is given to patches which contain the lower portion of the image. The opposite of this is observed for *Bald*, where high weight is observed for patches which contain the head, while very low weight is associated with the lower portion of the image.

Fig. 2.10 shows some sample images misclassified by the proposed facial attribute prediction pipeline. Fig. 2.10(a) refers to some sample false positives. These example show that large

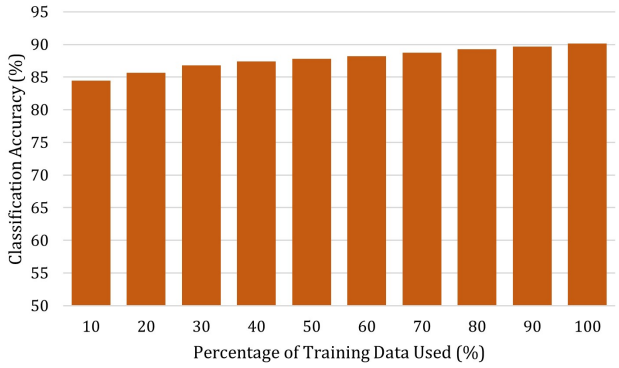

Figure 2.9: Bar graph representing the accuracy of the proposed framework with varying percentage of training data, for the CelebA dataset.

Figure 2.10: Sample mis-classifications of the proposed facial attribute prediction pipeline. (a) corresponds to the false positives, where the images in the first row are falsely predicted as attribute *male* and second row is predicted to have the attribute *wavy hair*. (b) refers to sample false negatives, where the first row corresponds to the attribute *attractive*, and second row corresponds to the attribute *receding hairline*.

variations in facial features make the task of facial attribute prediction challenging. Variations in hairstyles across males and females also lead to mis-classifications for attributes such as *male* and *wavy hair* (Fig. 2.10 (a)). Fig. 2.10(b) also presents some false negative samples of the proposed framework. It can be observed that subjective attributes such as *attractive* and *receding hairline* are difficult to encode in the trained model.

2.5 Conclusion

This research aims to address the problem of facial attribute prediction in the wild. A novel formulation for Residual Cosine Euclidean Autoencoder, termed as R-Codean has been presented for the same. The loss function of the proposed R-Codean model consists of a Euclidean distance based term, and a Cosine similarity based term. The model aims to learn representations of the input data, such that the error in direction and magnitude of the input and reconstructed sample is minimized. The Euclidean distance based component of the loss function handles slight variations across pose more efficiently, while the Cosine distance based component models

the illumination variations better. Shortcut connections in the R-Codean further ensure that optimal parameters are learned during the feature learning process. An entire framework has been presented for performing facial attribute analysis, which encodes local as well as global facial representations during feature learning process, and also incorporates weighted patch-based classification of attributes. Each component of the framework has been analyzed in order to create the optimal framework for facial attribute prediction. Experimental results on the CelebA and LFWA datasets, and comparison with existing state-of-the-art techniques demonstrate the efficacy of the proposed model.

Chapter 3

Evaluating Limit of Data Augmentation using Generative Adversarial Networks

3.1 Introduction

Generative models refer to a class of models that learn to represent an estimate of the data distribution of its training set. A model estimates the data distribution either explicitly by analyzing the training set, or implicitly by being able to generate samples similar to the given data. This field of generating data is termed as Generative Modeling, which is also thought of as the next frontier in Deep Learning [14]. Generative Adversarial Networks (GANs) [16] is an emerging technique for generative modeling using unsupervised or semi-supervised learning. They learn to estimate the data distribution implicitly by sampling from the learned distribution. The domain of generative modeling has achieved significant attention and has demonstrated promising results in the areas of vision [32, 23], language [2, 57], and speech [17] using different supervised learning techniques.

As shown in literature, these models posses impressive image generation capabilities [63], resulting in their usage for *data augmentation*. Training of deep models (high capacity models) often require large training data, which is not available in real world scenarios. A common method of addressing this is by creating more data, either via generative models, or image processing techniques (transformation) for increasing the training set [32, 56, 22]. Typically these transformations involve operations like image translation, horizontal reflections, and random cropping. Recently, GANs have been used for performing data augmentation resulting in increased classification performance [69, 13].

Generative Adversarial Networks consist 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 rather than G. In the form of a two play

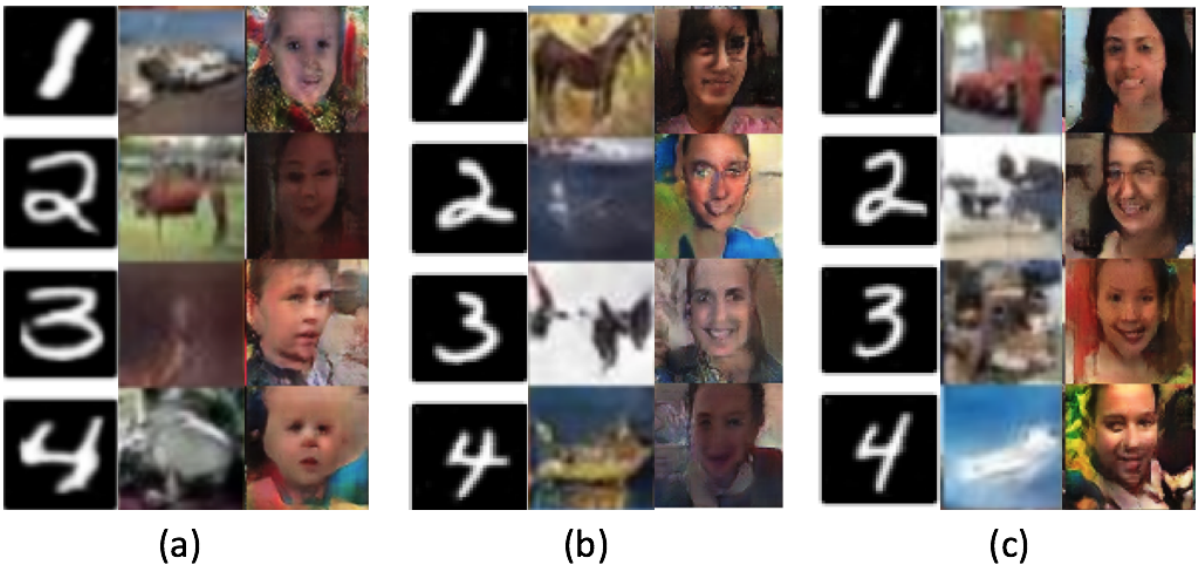

Figure 3.1: (a)shows images generated by DCGAN, (b) shows images generated by EBGAN, whereas (c) shows the images generated by WGAN on the datasets MNIST, CIFAR10 and Adience.

min-max game, GANs are trained such that the generator fools the discriminator into classifying a generated sample as a real sample. This research focuses on understanding the limits of data augmentation using GANs, in order to answer the question *'How much augmentation is sufficient?'*. The data generated by various state of the art GAN models is augmented to the original training dataset. Ratio of the real data to the augmented data is varied to analyze the classification performance trends. We observe that the classification accuracy approaches a limiting value as the ratio of the real data to generated data is increased. Analysis is performed on three datasets: MNIST [36], CIFAR10 [31] and Adience [12], using three widely used GAN architectures: Deep Convolutional Generative Adversarial Networks (DCGAN) [48], Energy Based Generative Adversarial Networks (EBGAN) [64] and Wasserstein Generative Adversarial Networks (WGAN) [1], along with three classifiers: 4-layered CNN, VGGNet [56], and ResNet [22]. The key contributions of this work are as follows:

1. Demonstrate the usefulness of data augmentation using generative adversarial networks in a variety of classification tasks,

2. Evaluate the limit of GAN based data augmentation for improving classification performance

3. Evaluate the generalizability of data augmentation across multiple GANs, datasets, and classifiers.

The remainder of the paper is organized as follows: Section 3.2 briefly describes the existing work on this problem, which is followed by some preliminaries for the proposed work. Section 3.4 presents details about the experimental setup and the results of the evaluation algorithm. Conclusion and future work are presented in Section 3.5.

3.2 Related Work

Data augmentation is a popular method to artificially enlarge the dataset using different kinds of label preserving transformations. This greatly helps to reduce overfitting in high capacity neural network models like Deep Convolutional Networks. Traditionally the transformations used for data augmentation involve simple operations like image translations, horizontal reflections and random cropping. In [32, 56, 22], the authors perform the above mentioned simple operations on the data to improve generalization and reduce overfitting of these high capacity models.

More recently there has been a lot of work which uses generative models, in particular Generative Adversarial Networks to generate data which are used for augmentation purposes for training deep networks. In [69], the authors use a cycleGAN based architecture for unpaired image to image translation to generate supplementary data for emotion classification task. This is done so as to balance the label distribution in the dataset using the augmentation technique. The authors report that the performance of classification model on the augmented dataset is significantly improved as compared to the baselines. In [13], the authors use a combination of both standard and GAN based data augmentation which improves the liver lesion classification performance. Here first the standard techniques are used to enlarge the training set and then further enlarges the dataset and diversity by using GAN based techniques for augmentation purposes.

3.3 Preliminaries: Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a novel framework for estimating generative models [16].They consist of two adversarial models: a generative model (G), and a discriminative model (D). The aim of the GAN framework is to train a discriminator and a generator such that the entire framework is able to generate data similar to a given training distribution. The generator generates a sample image which is provided as input to the discriminator. The discriminator classifies the input image as either *real* or *fake*. Therefore, the discriminator learns using supervised learning approaches, while the generator is trained to fool the discriminator. The loss function of a GAN model is given as:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.1}$$

where, x refers to the data sampled from the training set, and z refers to the latent code sampled from a prior distribution. The latent code is fed to generator, which yields the sample x drawn from the model's estimated probability distribution of the training data. On each iteration, two mini batches are sampled: a mini batch of x from the dataset and a mini batch of z from prior distribution over latent variables of the generator. Two gradient steps are made simultaneously: one updating the discriminators parameters to reduce its loss function and the other one updating the generators parameters to minimize its loss function. This research utilizes the DCGAN,

EBGAN, and WGAN models for evaluating the impact of data augmentation for the task of classification. Each of the models are explained in detail below:

Deep Convolutional Generative Adversarial Network (DCGAN) [48]: Proposed in 2015, DCGAN is built over some key guidelines in order to ensure that the model training is stabilized. Both the discriminator and the generator are convolutional neural networks trained with *ReLU* activation in the generator, and *Leaky ReLU* in the discriminator. This framework has heavily been used for applications which require generation between domains [27, 68].

Energy Based Generative Adversarial Network (EBGAN) [64]: The EBGAN model was proposed in 2016, in order to incorporate an energy-based framework in the GAN architecture. The model is designed such that the discriminator assigns a lower energy to data samples from the actual training set, and a higher energy to samples generated synthetically. Mathematically, the loss function can be expressed as:

$$\mathcal{L}_D(x, z) = D(x) + [m - D\big(G(z)\big)]^+ \tag{3.2}$$

$$\mathcal{L}_G(z) = D\big(G(z)\big) \tag{3.3}$$

where, $[a]^+ = max(0, a)$, and m is the margin value used for training the discriminator. Here x represents a data sample, $G(z)$ a generated sample, $\mathcal{L}_D$ the discriminator loss and $\mathcal{L}_G$ the generator loss.

Wasserstein Generative Adversarial Network (WGAN) [1]: Proposed in 2017, the WGAN architecture introduces the Wasserstein distance in the training process. Mathematically, it can be formulated as:

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} {}_{x\sim P_r}[f(x)] - {}_{x\sim P_\theta}\,[f(x)] \tag{3.4}$$

WGAN stabilizes the training of the GAN architecture, and has shown to approximate the Earth Mover (EM) distance well, resulting in improved image generation capabilities.

3.4 Experiments and Analysis

In order to understand the impact of data augmentation on the classification accuracies, experiments are performed with three GANs and three classifiers, on three datasets. For a given dataset, classifier, and GAN triplet, let D_{Real} correspond to the real training data, D_{Gen} refer to the data generated via a GAN, and let D_{Aug} correspond to the total augmented data used

Input: D_{real}
Output: Classification Accuracy
Train GAN on D_{Real}
Use GAN to generate D_{Gen}
Train C on D_{Aug} ($D_{Aug} = D_{Real} + D_{Gen}$)
Obtain classification accuracy of C on test set

Algorithm 1: Steps followed for evaluating the classification performance of classifier (C) with data generated via a GAN model (GAN).

for training the classifier. D_{Aug} is created as follows:

$$D_{Aug} = D_{Real} + D_{Gen} \tag{3.5}$$

here, the number of samples in D_{Gen} is decided as follows:

$$num(D_{Gen}) = \alpha \times num(D_{Real}) \tag{3.6}$$

α corresponds to the ratio of the number of generated samples to the number of real training samples. The classifier is trained with D_{Aug}, and tested on the real testing partition of the dataset. Algorithm 1 presents the above evaluation technique in a step-wise manner. This is repeated for multiple values of α, thereby enabling comparison of classification performance under different training data settings. GAN models of DCGAN, EBGAN, and WGAN are used for data generation, while a 4-layer CNN model, VGGNet, and ResNet are used as classifiers. Experiments are performed on the MNIST and CIFAR-10 datasets for the task of object classification, and Adience dataset for gender classification of face images. The ratio of generated to real samples (α) is varied as: $[0, 0.5, 2, 5]$). That is, experiments are performed with zero data augmentation, till five times augmentation of the real training set.

Implementation Details: The Convolutional Neural Networks (CNNs) used are implemented in the Keras framework whereas the GAN models are implemented in Pytorch. The CNNs are trained using the Adam optimization algorithm with a learning rate of 0.001 and the GAN models are trained with the learning rate and optimization algorithms described in their respective papers.

3.4.1 Results and Analysis

Table 1 through 9 show the comparison between the accuracy values obtained by training CNN classifiers on the various D_{aug} datasets. We see that there is a significant difference in accuracy values obtained on varying the ratios when adding D_{gen} to D_{real} to get the D_{aug} datasets.

From the above mentioned results we can make the following inferences,

1. As a better Generative Adversarial Network is taken, the accuracy due to data augmentation increases more.

As we can see comparing table 1, 2 and 3 (and all the tables with same datasets) we can see that as we pick an improved GAN for the dataset, the accuracy values increase. So there is an increase in the value when we go from 2 times the augmentation using DCGAN to EBGAN in the CIFAR10 dataset using ResNet as the classifier. The accuracy increases from 94.65 to 94.82. This same phenomenon can be observed with other datasets and other classifiers as well. For eg. in Table 8 and 9, we can see that while using Adience dataset, EBGAN is used with VGG as classifier we get accuracy of 92.31 whereas when using WGAN with VGG the accuracy increases to 92.44.

2. As a better Generative Adversarial Network is taken the accuracy value tends to saturate at a higher generated to real ratio.

   We can see that when we compare two different GANs on the same dataset, the better GAN tends to saturate at a high ratio. For Eg. if we compare DCGAN and EBGAN on CIFAR10 we see that the relative increase between ratios 0.5 and 2 is a lot less for DCGAN than EBGAN. This suggests that DCGAN tends to saturate well before the ratio of 2 whereas EBGAN tends to saturate near 2.

3. Using high capacity models on the dataset with GAN augmentation leads to higher accuracies than simpler models

   This fact is evident from table 1 through 9. With each GAN and Dataset we can see that as we take a better Classifier we tend to get a better accuracy. For Eg in Table 1 where we have MNIST with DCGAN, a 4 layer CNN gives us 99.48 accuracy whereas ResNet with the same amount of augmentation gives us an accuracy of 99.84. Similarly, if we look at Table 5, CIFAR10 with WGAN, as the classifier is varied from 4-Layer CNN to VGG to Resnet we see a continuous increment in accuracy values 93.20, 94.22 to 94.96 respectively.

4. Using the high capacity models we get very close to state of the art on a particular dataset without incorporating any domain knowledge in the training procedure or the classification model. We can very well observe from Table 3, Table 6 and Table 9, that when we tend to use ResNet with WGAN at the utmost limit of augmentation, we get results very close to the current state of the art on that particular dataset. For Eg. we achieve results of 99.84 on the MNIST dataset where as the state of the art is 99.86. Similarly we achieve an accuracy of 94.96 using CIFAR10 Whereas the state of the art is 97.00. Also we achieve around 92.44 on the adience gender classification task where the state of the art is 93.50.

3.4.2 Results with Small Sized Datasets

It is known widely that to train deep learning models a lot of data is required. Hence we use the above data augmentation algorithm to augment data with small sized datasets. We construct small sized datasets out of datasets like MNIST, NORB and CIFAR10 and also test out augmentation algorithm on the challenging NewBorn Face Recognition Dataset.

Table 3.1: MNIST with DCGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	99.20	99.36	99.48	99.48	97.40
VGG	99.68	99.74	99.82	99.82	98.00
ResNet	99.68	99.78	99.84	99.84	98.10

Table 3.2: MNIST with EBGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	99.20	99.38	99.55	99.55	97.65
VGG	99.68	99.74	99.82	99.82	98.28
ResNet	99.68	99.80	99.84	99.84	98.30

Table 3.3: MNIST with WGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	99.20	99.42	99.62	99.62	97.70
VGG	99.68	99.80	99.84	99.84	98.28
ResNet	99.68	99.80	99.84	99.84	98.34

Table 3.4: CIFAR10 with DCGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	91.40	91.92	92.20	92.20	88.20
VGG	92.84	93.36	93.72	93.72	88.92
ResNet	93.68	94.30	94.65	94.65	89.56

Table 3.5: CIFAR10 with EBGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	91.40	92.00	92.96	92.96	88.50
VGG	92.84	93.40	94.10	94.10	89.10
ResNet	93.68	94.42	94.82	94.82	89.85

Table 3.6: CIFAR10 with WGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	91.40	92.05	93.20	93.20	88.62
VGG	92.84	93.42	94.22	94.22	89.20
ResNet	93.68	94.42	94.96	94.96	89.90

Table 3.7: Adience with DCGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	86.20	86.72	87.38	87.38	83.80
VGG-Face	88.36	88.78	89.26	89.26	84.26
ResNet	91.52	92.00	92.23	92.23	85.10

Table 3.8: Adience with EBGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	86.20	86.90	87.52	87.52	84.00
VGG-Face	88.36	89.00	89.47	89.47	84.50
ResNet	91.52	92.08	92.31	92.31	85.32

Table 3.9: Adience with WGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	86.20	86.92	87.60	87.60	84.13
VGG-Face	88.36	89.12	89.64	89.64	84.80
ResNet	91.52	92.10	92.44	92.44	85.65

Table 3.10: MNIST with DCGAN on 100 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	78.24	78.47	79.02	79.02	74.78
VGG	80.46	81.00	81.24	81.24	76.53
ResNet	80.83	81.10	81.32	81.32	76.49

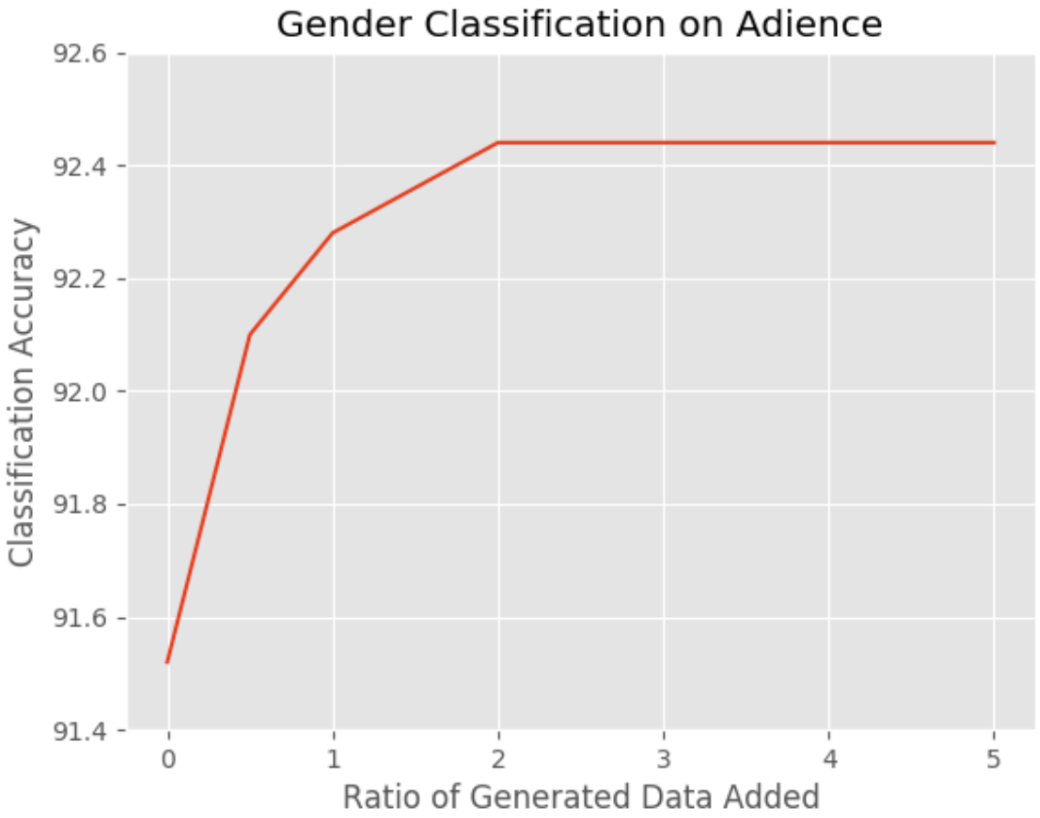

Figure 3.2: Graph showing accuracy values on Adience using data augmentation by WGAN

Table 3.11: MNIST with DCGAN on 1000 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	92.85	93.36	94.00	94.00	88.00
VGG	94.29	94.87	95.26	95.26	90.62
ResNet	95.12	96.00	96.38	96.38	91.00

Table 3.12: MNIST with DCGAN on 10000 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	98.10	98.52	99.10	99.10	94.8
VGG	98.62	99.00	99.40	99.40	95.86
ResNet	98.75	99.05	99.48	99.48	96.00

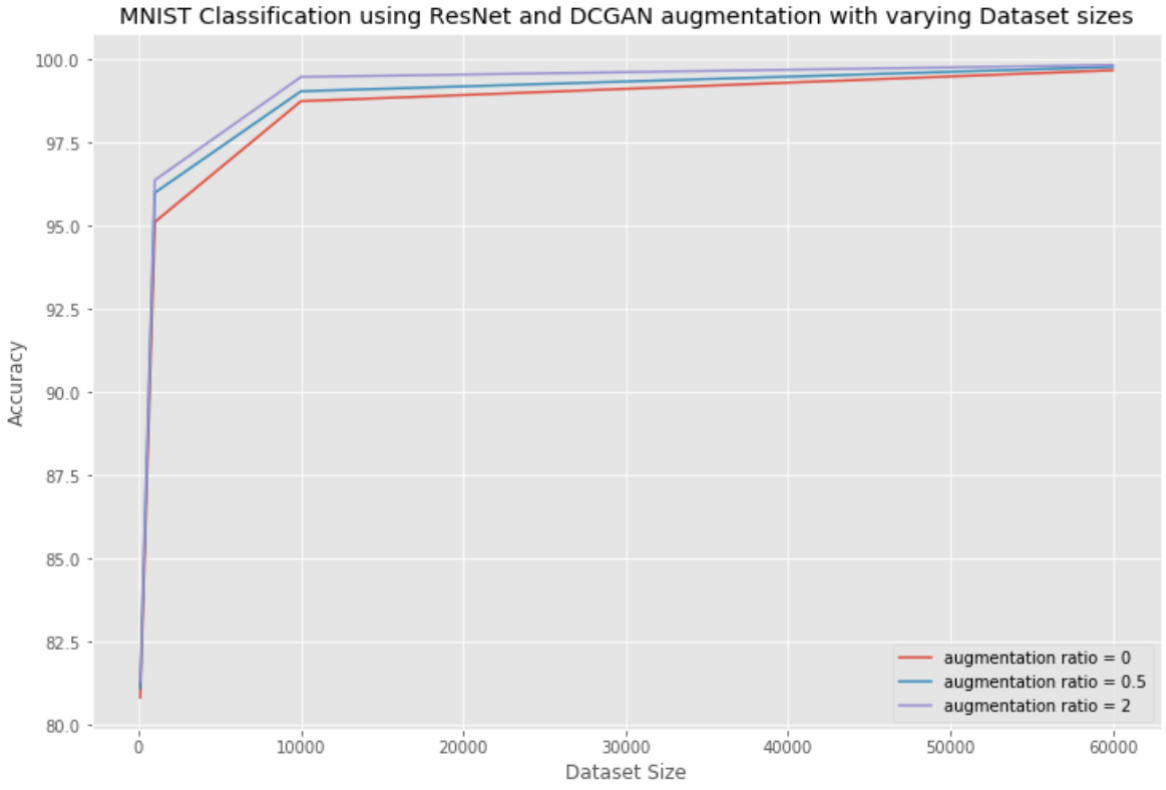

Figure 3.3: Graph showing accuracy values on MNIST using data augmentation by DCGAN

Table 3.13: CIFAR10 with DCGAN with 100 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	32.24	33.19	33.74	33.74	28.86
VGG	35.16	36.14	36.48	36.48	31.63
ResNet	35.84	36.49	36.75	36.75	32.00

Table 3.14: CIFAR10 with DCGAN with 1000 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	51.31	51.86	52.40	52.40	48.82
VGG	53.82	54.73	55.31	55.31	50.13
ResNet	55.35	56.23	56.48	56.48	51.20

Table 3.15: CIFAR10 with DCGAN with 10000 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	73.08	73.52	74.73	74.73	70.28
VGG	75.41	76.27	76.83	76.83	72.14
ResNet	76.25	76.79	77.36	77.36	72.45

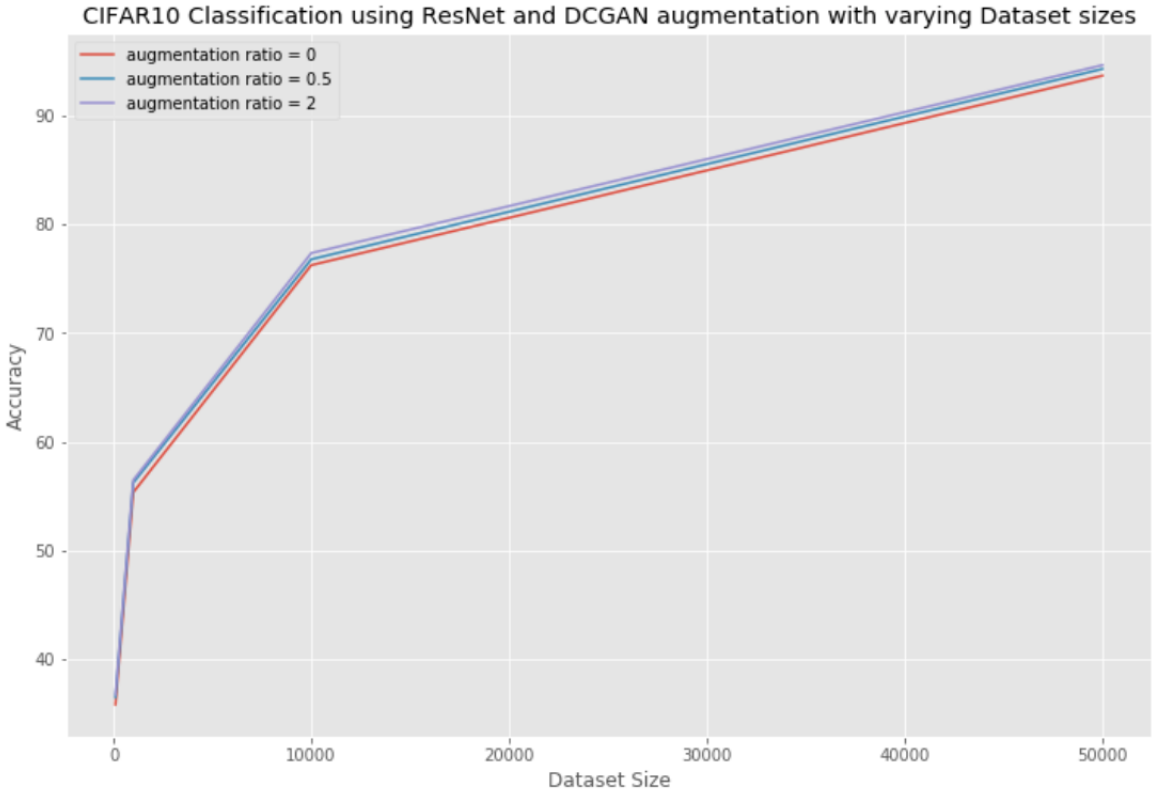

Figure 3.4: Graph showing accuracy values on CIFAR10 using data augmentation by DCGAN

Table 3.16: NORB with DCGAN on 100 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	53.2	54.2	55.0	55.0	49.8
VGG-Face	52.8	55.0	56.0	56.0	49.5
ResNet	54.0	56.2	56.8	56.8	51.0

Table 3.17: NORB with DCGAN on 1000 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	62.4	64.0	65.5	65.5	60.1
VGG-Face	66.0	67.9	69.4	69.4	63.5
ResNet	68.0	70.5	72.0	72.0	64.0

Table 3.18: NORB with DCGAN on 10000 samples

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	73.73	75.2	77.0	77.0	70.2
VGG-Face	78.4	79.0	80.0	80.0	74.4
ResNet	78.5	79.2	80.1	80.1	74.8

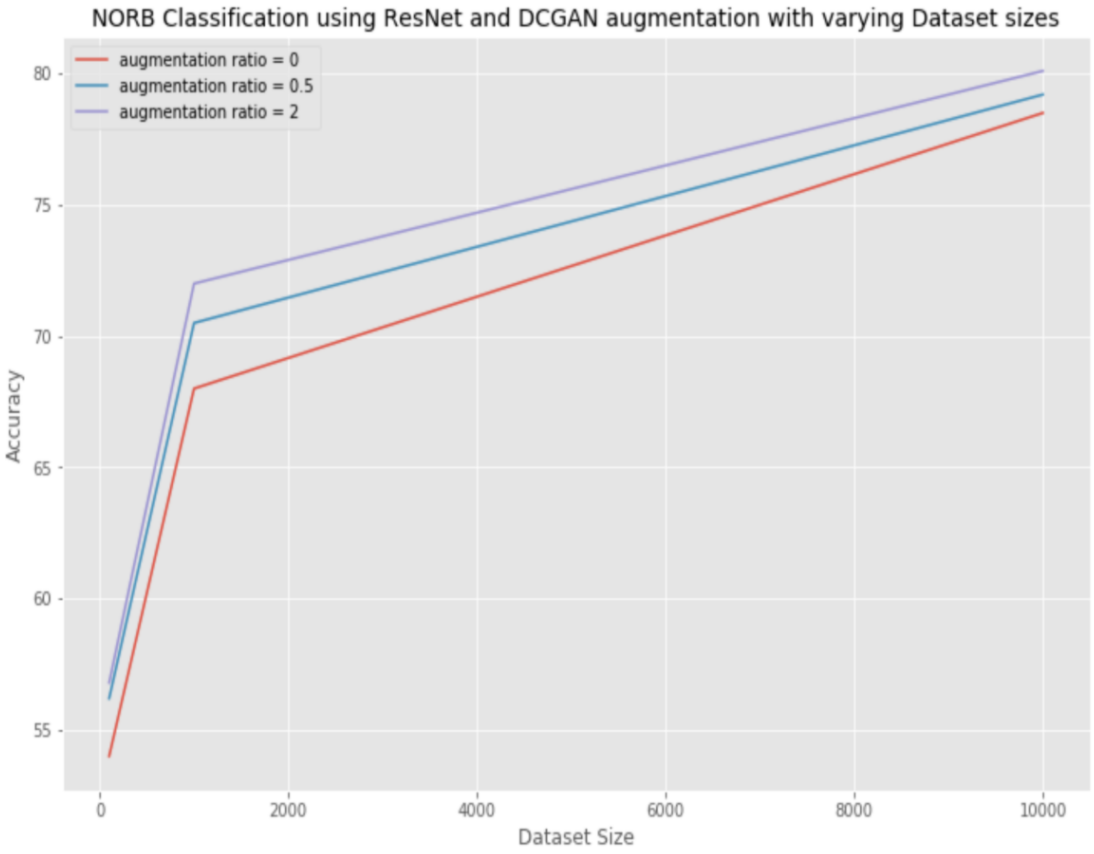

Figure 3.5: Graph showing accuracy values on NORB using data augmentation by DCGAN

Table 3.19: NewBornFaceRec with DCGAN

	Ratio of D_{Gen} to D_{Real} in Training				
	0	0.5	2	5	*Syn.*
4 Layer CNN	53.5	57.9	58.44	58.44	50.6
VGG-Face	56.2	61.80	64.17	64.17	52.9
ResNet	78.12	79.00	79.60	79.60	75.86

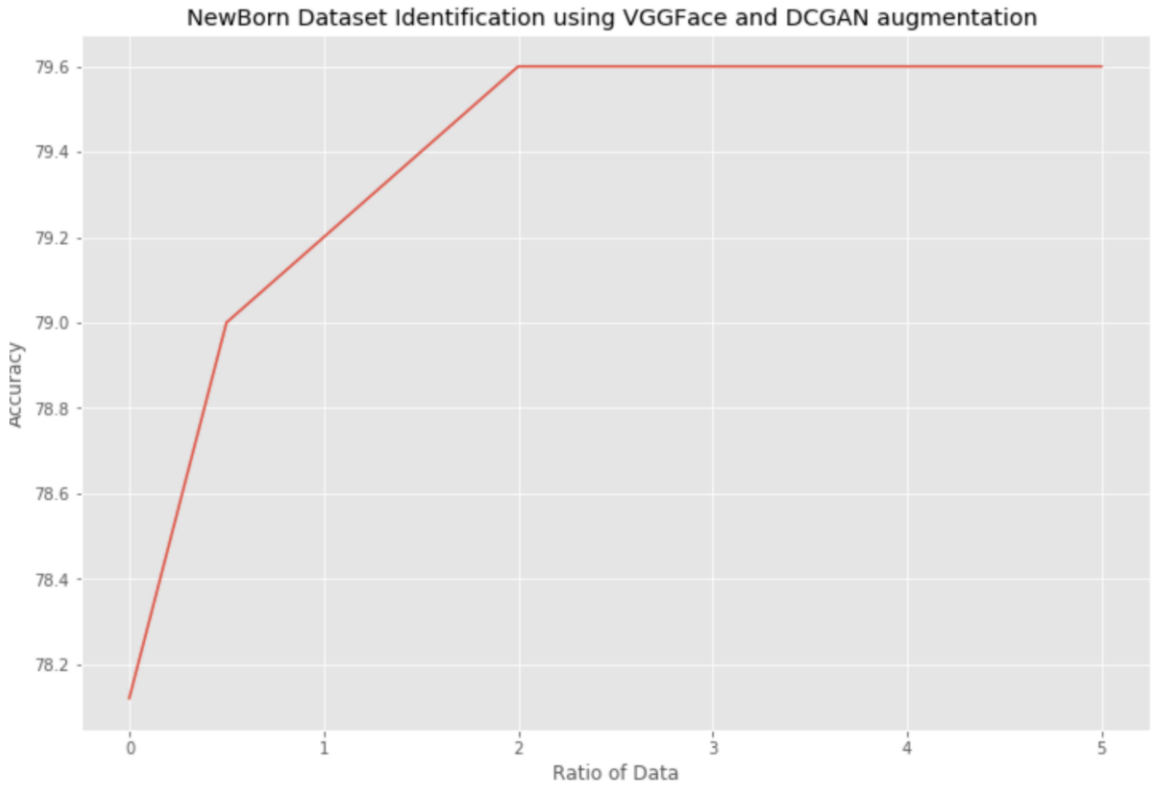

Figure 3.6: Graph showing accuracy values on NewBornFaceRec using data augmentation by DCGAN

3.5 Conclusions

In this research we present an evaluation method for finding the limit of data augmentation using generative adversarial networks. We do so on various datasets for a variety of classification tasks. Furthermore we show that classification accuracies tend to approach a limiting value as the ratio of generated data to real data is increased. Another interesting result from the research is that we get near state of the art results on various classification tasks without using any domain knowledge either in the classification model or in the training process.

Chapter 4

Sub-Class Generative Adversarial Networks

4.1 Introduction

A generative model is defined as any model that takes a training set and learns to represent an estimate of that distribution. For some classes of models, the model estimates the data distribution explicitly whereas in some cases it is only able to do so implicitly by being able to generate samples from the learnt data distribution by the model.

Generative Adversarial Networks[16, 15] are an example of generative models.They fall within the latter class of models wherein they are able to implicitly sample from the learnt model distribution to generate the image samples.In this work we focus on a novel approach for training Generative Adversarial Networks.Generative Adversarial Networks are a class of Generative models which learn to estimate the data distribution implicitly by sampling from the learnt distribution.These models posses impressive image generation capabilities.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 rather than G. Both G and D could be a non-linear mapping function, such as a multi-layer perceptron or a Convolutional Neural Network.

Generative Adversarial Networks are known to be very unstable during the training process and loss curves of these Networks do not have any interpretable meaning.State of the art research on image generation using Generative Adversarial Networks focuses on alleviating this problem in Generative Adversarial Networks and also on improving the quality of the generated images.Another major issue was that for comparing the quality of image generation by GANs, researchers had to undergo a visual turing test to determine how good the samples are which made the comparison process highly subjective on the observer. Recently Salimans et al. in [52, 15] came up with a metric known as the Inception score to quantity the generated images.This score is now widely used in the GAN literature as it correlates well with human evaluation.

Salimans et al. in [52] show some improved ways of training a generative adversarial network such as one-sided label smoothing, virtual batch normalization etc. which helps in convergence of the GAN training process.More recently LeCun et al. [64] introduced the Energy Based Generative Adversarial Network which views the discriminator as an energy function that assigns low energies to regions near the data manifold and high energies to other regions. Looking at the discriminator as an energy function allows to use a variety of loss functions in addition to the usual binary classifier with logistic output.

Arjovsky et al. in [1] introduced the notion of the Wasserstein distance in the GAN training process.Instead of using Binary cross entropy loss for training they use the standard mean squared error for the training. Another change is that since the output activation is no longer sigmoid for the discriminator it is considered to be a critic where as the Generator is termed to be an Actor.Hence the GAN here is referred to as the Actor-Critic Model rather than the Generator and Discriminator.Also the weights of both the generator and the discriminator are clipped within a certain range.All these changes make the GAN training process much more stable as compared to DCGAN, EBGAN etc. and the quality of the generated images measured in terms of the Inception Score is significantly higher than the other models.More recently Gulrajani et al. in [19] show that regularizing the loss of Wasserstein GAN with the norm of the gradient helps to stabilize the training process even further.Similar gradient penalizing approaches have been followed by Hays et al. in [41] and in Maximum Mean Discrepency GANs.

In our proposed architecture and training procedures for GAN we utilize the subclass information present in many datasets like CIFAR100 and Adience. We first ask ourselves whether the data distribution learnt by GANs is close enough to the real dataset or not and if not in what ways could we improve this process.Guided by this question we claim the following contributions

1. Experimental proof by training various state of the art GANs on datasets like CIFAR10 and classification of the generated data by using CNNs like VGGNet, Resnet etc to show that true distribution varies significantly from generated data distribution.
2. Changing the latent code of the generator to model each class of the GAN using separate latent code.
3. Regularizing the Generator and Discriminator with class based L21 regularization
4. Learning a Stack of GANs wherein the output from a previous GAN is given as input to the next GAN
5. Instead of learning two layers as above we propose to learn a 3 layered network of GANs calling it DeepGANs

4.1.1 Generative Models

A generative model is defined as any model that takes a training set and learns to represent an estimate of that distribution.For some classes of models, the model estimates the data

distribution explicitly whereas in some cases it is only able to do so implicitly by being able to generate samples from the data distribution learnt by the model.Generative Adversarial Networks (the focus of this work) are an example of generative models.They fall within the latter class of models wherein they are able to implicitly sample from the learnt model distribution to generate the image samples.

4.1.2 Importance of Generative Models

- Generative modeling and density estimation is an important area in machine learning.
- Training and sampling from generative models is an excellent test of our ability to represent high-dimensional probability distributions.
- Generative models can be trained on missing data and can further provide predictions on the missing samples.
- Deep learning models are known to require lots and lots of labelled data to train on, these generative models can be used to sample from the learnt distribution of the data to generate more such examples which can both improve the performance of the existing machine learning classifiers as well as enable training of deep models on limited datasets.

4.2 Problem Statements

4.2.1 Do GANs learn the data distribution ?

This first part of the work tries to measure the closeness between the real data distribution and images generated by Generative Adversarial Networks.We pose the following question : If the real data distribution could be approximated by the images generated by GANs then discriminative classifiers trained on both the datasets should yield similar accuracies.Running the above experiment would prove or disprove experimentally whether the data distributions are close to each other or not.

4.2.2 Improving Image Generation in GANs

If the above posed question is true i.e GANs are not able to learn the data distribution then the GAN training framework should be changed so that the generated data distribution is able to approximate the true distribution.For the evaluation purpose of the new training frameworks the approach suggested above(training discriminative classifiers on the generated datasets) should be used as a metric for evaluation.

Further more the Inception score(introduced by Salimans et al. in [52]) is a widely used metric for evaluation of GANs.The changed training process and architecture should also be evaluated using this metric.

4.2.3 Sub-Class Generative Adversarial Networks

There are several datasets which have subclass information present,for Eg.the Adience dataset, the CIFAR100 dataset etc.Can such information be utilized to improve the image generation process and quality in Generative Adversarial Networks.

4.3 Background Information

In this work, we propose new architectures and training techniques for Generative Adversarial Networks. In this section we briefly review neural networks, Convolutional Neural Networks and Generative Adversarial Networks.

4.3.1 Neural Network

Artificial neural networks(ANN) refer to mathematical models which consists of processing nodes with multiple connections. These connections have modifiable parameters which modify the signals which pass through them. Most networks may be split into two important types: Supervised and Unsupervised. One of the most important type of supervised neural network is the feedforward multilayer perceptron. Feedforward refers to the fact that there is a definite input and output and flow of data in one direction. It consists of an Input Layer several hidden layers and an output layer. Each of the input nodes connects to every node in the next layer of nodes and each of these connections has a weight associated with it. A node in hidden layer forms a weighted sum of inputs and passes them through a non linear function.

$$a_j = \sigma(\sum_i w_{i,j} x_i) \tag{4.1}$$

where a_j is the activation of the hidden unit j

4.3.2 Convolutional Neural Network

A Convolutional Neural Network(CNN) consists of one or more Convolutional layers followed by one or more fully connected layers.The CNN takes advantage of 2D structure of input image is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. CNNs are easier to train and fewer parameters than fully connected networks with the same number of hidden units.

Architecture

A CNN contains a number of convolutional and sub-sampling layers optionally followed by fully connected layers.The input to a Convolutional Network is a $m \times m \times r$ where m is the height and width of image and r is the number of channels.The convolutional layer has k filters of size $n \times n \times r$ where n is smaller than dimension of image.The filters give rise to a locally connected structure which are each convolved with image to produce k feature maps.Each map is then subsampled.

Layers in a Convolutional Neural Network

Input layer $[n \times n \times r]$ holds the raw pixel values of the image and with color channels of the image.
Convolutional layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.
Pool layer performs a downsampling operation along the spatial dimensions (width, height)
Fully-connected layer computes the class scores, resulting in volume of size where each of the numbers correspond to a class score. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Parameter Sharing

Parameter sharing scheme is used in Convolutional Layers to control the number of parameters.Parameter sharing can dramatically reduce the number of parameters by assuming that if one feature is useful to compute at some spatial position $(x1, y1)$, then it should also be useful to compute at a different position $(x2, y2)$.
During backpropagation, every neuron in the volume will compute the gradient for its weights, but these gradients will be added up across each depth slice and only update a single set of weights per slice.

Fully Convolutional Neural Network

These networks have been a recent spurt of popular innovations in the field of deep learning. They involve a combination of downsampling convolutions on an image and upsampling the tiny resolution feature maps back to the pixel label array size. To perform efficient upsampling also ensuring that the blurring of feature maps does not take place, it is important to introduce convolutions in the upsampling process too, building up sparse arrays which are convoluted with kernels for a smooth upsampling. The resulting network architecture involves a set of encoder stages (downsampling convolutions) and decoder stages (

upsampling convolutions).

4.3.3 Generative Adversarial Networks

Generative adversarial nets are a novel approach to train a generative model.These models posses impressive image generation capabilities.
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 rather than G. Both G and D could be a non-linear mapping function, such as a multi-layer perceptron or a Convolutional Neural Network.

The Discriminator

The discriminator learns using supervised learning approaches, dividing inputs into two classes (real or fake). The generator is trained to fool the discriminator.Both the generator and discriminator have cost functions that are defined in terms of both their parameters which they try to minimize.

The Generator

The generator is a simple differentiable function.When the latent code(z) is sampled from a prior distribution and fed to Generator, it yields a sample x drawn from models estimated probability distribution of the training data.

Training Process

There are many ways to train a Generative Adversarial Network, but the most frequently used is simultaneous SGD on the Generator and the discriminator.On each iteration, two mini batches are sampled : a mini batch of x from the dataset and a mini batch of z from prior distribution over latent variables of the generator.Two gradient steps are made simultaneously one : one updating the discriminators parameters to reduce its loss function and the other one updating the generators parameters to minimize its loss function.

The cost used for the discriminator is:

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2} \underset{x \sim pdata}{E} \log D(x) - \frac{1}{2} \log \left(1 - D\left(G(z)\right)\right). \tag{4.2}$$

This is the cross entropy cost which is minimized when training a binary classifier with a sigmoidal output.The only difference being, here the classifier is trained on two mini batches of data, one from the dataset and the other one from the output of the generator.

The cost used for the Generator is :

$$J^{(G)} = -J^{(D)}. \tag{4.3}$$

GAN variants

The Generative Adversarial Network proposed by Goodfellow et al. in [16] has many limitations relating to the quality of the generated image and the stability of the training process.Many Architectures and training techniques have been suggested for overcoming the same.

DCGAN - Deep Convolutional Generative Adversarial Network

Radford et al. in [48] gave some key guidelines regarding the Generative Adversarial Network Architectures such that the model training is stabilizes.The architecture guidelines are :

- Replacement of any pooling layers with strided convolutions in the discriminator and use of fractional-strided convolutions in the generator network.
- Use of batch-normalization in both the generator and the discriminator networks.
- Use of Fully Convolutional Architectures for both generator and discriminator(Avoiding use Fully Connected Layer in Deeper Architectures).
- Use of ReLU activation in generator.
- Use LeakyReLU activation in the discriminator.

4.4 State of the Art

After the introduction of GAN by Goodfellow et al. in [16], many ways have been suggested to improve the quality as well as the stability of generative adversarial network.One of the major problem though is the visual Turing test which the researcher has to do to evaluate the quality of the generated samples,thus making the comparison of different GAN training algorithms very difficult.Very recently Salimans et al. in [52] proposed the use of Inception score as a metric for evaluating the quality of the generated images.This Inception score is now being used in the GAN literature as it correlates very well with human evaluation.

In this section we describe some state of the art Generative Adversarial Network architectures and training methods which give the highest inception scores on datasets like CIFAR10 and MNIST.

(a) Improved Techniques for Training Deep Convolutional GANs

(b) Energy Based GANs

(c) Wasserstein GANs

(d) Wasserstein GAN with gradient penalty

4.4.1 Improved Techniques for Training Deep Convolutional GANs

Salimans et al. in [52] showed various techniques for stabilizing the GAN training process and generating images of improved quality as compared to DCGANs(described in previous section).

They advocate the use of techniques such as Virtual Batch Normalization, Feature Matching and One-Sided Label smoothing. They are described as follows :

Feature Matching

Feature matching addresses the instability of GANs by specifying a new objective for the generator.Instead of trying to directly maximize the output of the generator it makes the generator to generate data that matches the statistics of the real data.

Minibatch Discrimination

To avoid the problem of mode collapse in GANs, the authors propose the use of mini batch discrimination where the discriminator network that looks at multiple examples in combination, rather than in isolation.

One sided Label Smoothing

The positive labels are smoothed to values like 0.9 ,0.95 etc and negative labels are left at 0.This helps the networks train more effectively.

Virtual Batch Normalization

In Virtual Batch Normalization(VBN) each example x is normalized based on the statistics collected on a reference batch of examples that are chosen once and fixed at the start of training.

4.4.2 Energy Based GAN

The Energy based Generative Adversarial Network model (EBGAN) views the discriminator as an energy function which attributes low energies to the regions near the data manifold and higher energies to other regions. The generator is seen as being trained to produce samples with minimal energies, while the discriminator is trained to assign high energies to these generated samples. Viewing the discriminator as an energy function allows to use a wide variety of architectures and loss functions which greatly helps in stabilizing the training process.

4.4.3 Wasserstein GAN

The authors introduce a new training algorithm Wasserstein GAN, an alternative to traditional GAN training. In this new model, they show that stability of training is improved which alleviates the model of problems like mode collapse and provides meaningful loss curves.
They introduce the notion of the Wasserstein distance in the GAN training process.Instead of using Binary cross entropy loss for training they use the Standard Mean squared error for the training. Another change is that since the output activation is no longer sigmoid for the discriminator it is considered to be a critic where as the Generator is termed to be an Actor.Hence the GAN here is referred to as the Actor-Critic Model rather than the Generator and Discriminator.Also the weights of both the generator and the discriminator are clipped within a certain range.These changes make the GAN training process much more stable as compared to DCGAN, EBGAN etc. and the quality of the generated images measured in terms of the Inception Score is significantly higher than the other models.

4.4.4 Wasserstein GAN with Gradient penalty

The above mentioned WGAN is very stable during the training process but can still generate low-quality images or fail to converge in some settings.The authors show that these problems are due to use of weight clipping in WGAN to enforce a Lipschitz constraint

on the critic.The authors propose a alternative strategy : penalization of the norm of gradient of the critic with respect to the input.

4.5 Proposed Algorithms

In this section we describe our proposed GAN training architectures and training algorithms.

4.5.1 Do GANs model the data distribution ?

We ask ourselves the question whether it is the case in Generative Adversarial Networks.If this were to be case, then discriminative classifiers trained on generated data by GANs should have the same accuracy as the classifiers trained on the real data. We run a large scale experiment wherein we choose three state of the art GAN architectures namely, WGAN, DCGAN and EBGAN.We train these GANs on three datasets(CIFAR10, MNIST and Adience).Then we try to compare the performance of the CNN classifiers like ResNet, VGGNet etc. trained on natural datasets to data generated by these GAN architectures.

4.5.2 Improving Quality of Generated Images in GANs

After proving experimentally that the data distribution learnt by GANs is not same as the true data distribution, we propose new training procedures and architectures for GAN applicable mainly to datasets which have subclass information present.

Changing the Latent Code

The latent Code in a Generative Adversarial Network can have any distribution as the prior. Researchers mostly tend to use unit normal Gaussian for the same.We experiment with assigning the latent code with different distributions such that every class(or subclass) of data can be assigned a different prior on the latent code.We show improved results in comparison to Vanilla GAN.

Using Class Based regularization

To enable our Generator and Discriminators to learn effective class based sparse representation we regularize the Generator and Discriminator networks with L21 regularization.We show improved results using this technique in comparison to vanilla GANs.

Learning a Stack of GANs

For effectively utilizing the subclass information present in datasets, we have proposed a stacked GAN architecture wherein the output from previous GAN is given as input into the next GAN.In learning such a stack of GANs, except for the first layer, the input is an image which is enhanced at that level. So self regularization can be applied to GAN between the input and the output images of the GAN.

Such a hierarchical GAN training process enables us to generate images which have significantly better quality as compared to standard GANs like DCGAN.

DeepGANs

Here we propose to take the StackGAN experiment to an extreme where we propose a stack of 3 layers of GANs.Within this architecture the first layer of GANs generate data from noise and the rest layers generate data taking the output of the previous layer's images as input.Here again the output images can be self regularized with respect to the input. There is no fixed architecture for DeepGANs are the architecture itself is dataset dependent.For Eg. if we consider the CIFAR100 dataset, the first layer consists of 100 vanilla DCGANs(one corresponding to each class), the next layer consists of 20 DCGANs(one corresponding to each super class in the dataset and each DCGAN here is conditioned on the data of the 5 classes it generates), and finally we have a single GAN in the last layer.

Such a deep hierarchical GAN architecture enables us to generate images which have significantly better quality as compared to standard GANs like DCGAN.

4.6 Dataset Description

In this section we describe the datasets we have used for the purpose of evaluating our GAN training algorithms.

4.6.1 MNIST Dataset

The MNIST database[36] is a large database of handwritten digits that is commonly used for training various image processing systems.The database consists of 60,000 training images and 10,000 testing images of size 28x28.

4.6.2 CIFAR10 Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.The dataset is divided into five training batches and one test batch, each with 10000 images.

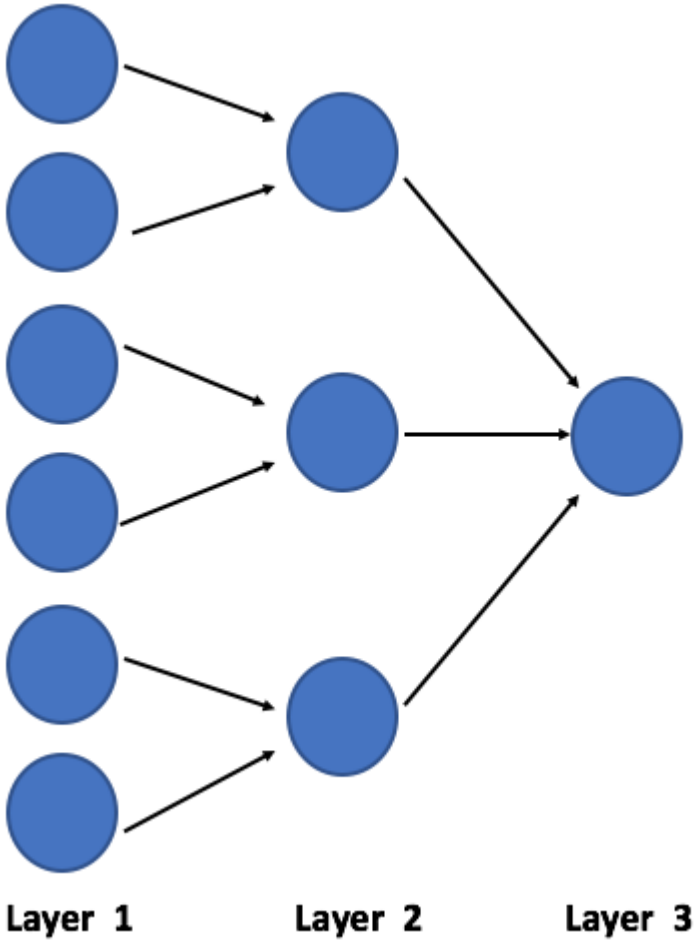

Figure 4.1: DeepGAN architecture. Each blue circle represents a DCGAN. The data flow is also visualized here. Layer 1 represents one GAN for each subclass, whereas Layer 2 represents one GAN for each class and Layer 3 the final output GAN. The dataset here is assumed to have 6 subclasses and 3 classes. Data generated by each of the subclass GAN is given as input to the corresponding class GAN which is inturn given as output to the output GAN.

4.6.3 CIFAR100 Dataset

The CIFAR100 dataset consists of 60000 training examples and 10000 testing examples.This dataset has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses.The size of each image is 32x32.

4.6.4 Adience Dataset

The Adience dataset[12] consists of 26,580 images of 2,284 subjects.The photos are collected from Flikr.com and have annotations on Gender and Age.The dataset has 4 training-testing folds defined.

4.6.5 CelebA Dataset

The Celebfaces Attributes Dataset (CelebA) is a face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations.The dataset contains over 10,000 identities, more than 200K celebrity images and 40 annotated binary attributes for each image.

4.7 System Requirements

In this section we list the hardware requirements which are required to run the experiments in this work.

4.7.1 Hardware Requirements

All experiments we performed were run on a Nvidia 1080i GPU with 11GB memory.Furthermore the system had 32GB of CPU RAM, 1TB of HDD space as well as 128GB of SSD memory.Training of a 4 Layered Convolutional architecture(Generator Description) on CIFAR10 typically took 2 hours whereas training of SubClass GANs and DeepGANs took more than 24 hours on datasets like Adience and CIFAR100.

We also experimented with the latent code of the generator by changing the gaussian parameters as well as the trying to model the code with a Gaussian mixture model.These experiments took an average of 4 hours per GAN model training.

4.7.2 Software Requirements

We have used OpenCV in Python for data preprocessing like resizing the images to feed as input to the GAN. We have used Keras and Pytorch for the implementation of the mentioned GAN and CNN architectures. Being well-documented and having a variety of pre-existing CNN and GAN implementations made our task relatively easy.

4.8 Experiments and Results

In this section we list the results of all our performed experiments.

4.8.1 Analysis of learnt data distribution by GANs

We trained all the state of the art GANs on Datasets like CIFAR10, MNIST and Adience. Furthermore we trained Convolutional Networks on the real and simulated datasets to check whether the classification accuracies were comparable.The results of the experiments are given below.

Dataset	MNIST(real , simulated)	CIFAR10(real , simulated)	Adience(real , simulated)
DCGAN	99.2 , 97.32	91.2 , 87.8	86.2 ,81.6
EBGAN	99.1 , 97.3	91.4 , 88.1	86.5 , 82.4
WGAN	99.2 , 97.54	91.4 , 88.2	86.2 , 82.4

Table 4.1: Classification accuracies for real and synthetic data on a 4 layer CNN classifier.Note the first element corresponds to the accuracy on real data and second on simulated data

Dataset	MNIST(real , simulated)	CIFAR10(real , simulated)	Adience(real , simulated)
DCGAN	98.8 , 97.4	91.8 , 87.6	86.4 , 81.6
EBGAN	98.8 , 97.4	91.8 , 88.0	86.4 , 82.4
WGAN	98.8 , 97.6	91.8 , 88.1	86.4 , 82.6

Table 4.2: Classification accuracies for real and synthetic data on VGGnet[56] classifier.Note the first element corresponds to the accuracy on real data and second on simulated data

Dataset	MNIST(real , simulated)	CIFAR10(real , simulated)	Adience(real , simulated)
DCGAN	98.8 , 97.4	91.6 , 87.6	86.2 , 81.7
EBGAN	98.8 , 97.3	91.6 , 88.2	86.2 , 82.6
WGAN	98.8 , 97.7	91.6 , 88.2	86.2 , 82.6

Table 4.3: Classification accuracies for real and synthetic data on ResNet50[22] classifier.Note the first element corresponds to the accuracy on real data and second on simulated data

4.8.2 Improvement in Image Generation process

The results in above sections suggest that the distribution of real data differs from the distribution learnt by the model.Here we show results on various proposed GAN training algorithms.

Changing the latent Code and L21 regularization

In the below set of experiments we change the latent code of the generator with a different gaussian prior for each subclass.We use the CIFAR100 and Adience datasets for these experiments.

Subclass-GANs and DeepGANs

In the below section we give results on Subclass Generative Adversarial Networks and Deep GAN architectures.

Conclusion : Our results show that the data distribution learnt by GANs is not very close to that of the true data distribution.Furthermore of all our proposed approaches for improving the GAN training framework we find that subclass CGAN with self regularization gives the best results on the CIFAR100 and Adience.

Experiment	Adience(Simulated Data)	Adience(Real Data)
DCGAN	81.6	86.2
Fit Gaussian on each subclass of data	82.8	86.2
Random Gaussians on each subclass of data	81.8	86.2
Using L21 on Generator	81.8	86.2
Using L21 on Generator and Discriminator	81.6	86.2

Table 4.4: Classification accuracies for real and synthetic data on 4 layered Convolutional classifier for the Adience Dataset.

Experiment	CIFAR100(Simulated Data)	CIFAR100(Real Data)
DCGAN	68.3	73.6
Fit Gaussian on each subclass of data	70.2	73.6
Random Gaussians on each subclass of data	69.8	73.6
Using L21 on Generator	69.3	73.6
Using L21 on Generator and Discriminator	70.1	73.6

Table 4.5: Classification accuracies for real and synthetic data on 4 layered Convolutional classifier for the CIFAR100 Dataset.

Experiment	Adience(Simulated Data)	Adience(Real Data)
DeepGAN	84.9	86.2
Subclass CGAN with self regularization	85.4	86.2
CGAN with self regularization	85.1	86.2
Subclass CGAN	84.8	86.2
CGAN	84.1	86.2

Table 4.6: Classification accuracies for real and synthetic data on 4 layered Convolutional classifier for the Adience Dataset.

Experiment	CIFAR100(Simulated Data)	CIFAR100(Real Data)
DeepGAN	72.5	73.6
Subclass CGAN with self regularization	72.9	73.6
CGAN with self regularization	72.5	73.6
Subclass CGAN	72.5	73.6
CGAN	72.3	73.6

Table 4.7: Classification accuracies for real and synthetic data on 4 layered Convolutional classifier for the CIFAR100 Dataset.

Figure 4.2: Sample of generated images after 200 epochs on CIFAR100 dataset

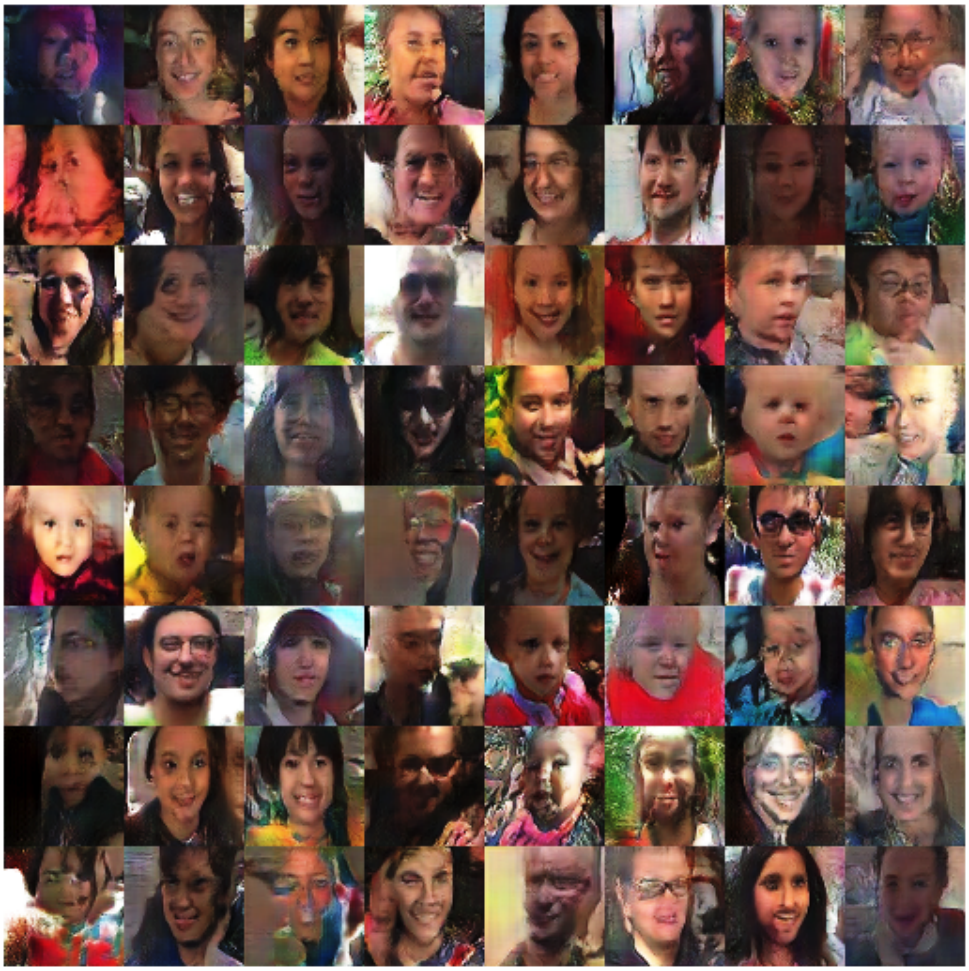

Figure 4.3: Sample of generated images after 200 epochs on Adience dataset

4.9 Discussion

In this study, we have firstly analyzed how close the data distribution learnt by GANs is to the real data distribution. After determining that they differ by some margin we propose new training techniques and architectures for the GAN framework. The techniques proposed rely on using the subclass information present in the various datasets. We show improved results on the quality of images generated by our proposed methods evaluated by inception score and classification accuracy of the discriminative classifiers.

We also propose the use of hierarchical GAN learning in the form of Stacked GAN and DeepGAN where the output of one GAN is given as input to another.These techniques also show improved accuracy as compared to vanilla GANs.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[3] Pierre Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. 2012, pp. 37–49.

[4] Yoshua Bengio et al. "Greedy Layer-wise Training of Deep Networks". In: *International Conference on Neural Information Processing Systems*. NIPS'06. Canada, 2006, pp. 153–160.

[5] Thomas Berg and Peter Belhumeur. "Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 955–962.

[6] Samarth Bharadwaj et al. "Domain Specific Learning for Newborn Face Recognition". In: *IEEE Transactions on Information Forensics and Security* 11.7 (2016), pp. 1630–1641.

[7] Zhimin Chen and Yuguang Tong. "Face super-resolution through wasserstein gans". In: *arXiv preprint arXiv:1705.02438* (2017).

[8] KamYuen Cheng and Ajay Kumar. "Contactless finger knuckle identification using smartphones". In: *Biometrics Special Interest Group (BIOSIG), 2012 BIOSIG-Proceedings of the International Conference of the*. IEEE. 2012, pp. 1–6.

[9] Franfffdfffdois Chollet. *Keras*. `https://github.com/fchollet/keras`. 2015.

[10] Junyoung Chung et al. "Deep attribute networks". In: *Deep Learning and Unsupervised Feature Learning NIPS Workshop*. 2012.

[11] Max Ehrlich et al. "Facial attributes classification using multi-task representation learning". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 47–55.

[12] Eran Eidinger, Roee Enbar, and Tal Hassner. "Age and gender estimation of unfiltered faces". In: *IEEE Transactions on Information Forensics and Security* 9.12 (2014), pp. 2170–2179.

[13] M. Frid-Adar et al. "Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification". In: *ArXiv e-prints* (Jan. 2018). arXiv: `1801.02385 [cs.CV]`.

[14] Ian Goodfellow. "NIPS 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

[15] Ian Goodfellow. "NIPS 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

[16] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[17] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pp. 6645–6649.

[18] David A Grossman and Ophir Frieder. *Information retrieval: Algorithms and heuristics*. Vol. 15. Springer Science & Business Media, 2012.

[19] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *arXiv preprint arXiv:1704.00028* (2017).

[20] Manuel Günther, Andras Rozsa, and Terrance E Boult. "AFFACT-Alignment Free Facial Attribute Classification Technique". In: *arXiv preprint arXiv:1611.06158* (2016).

[21] Emily Hand and Rama Chellappa. "Attributes for Improved Attributes: A Multi-Task Network Utilizing Implicit and Explicit Relationships for Facial Attribute Classification". In: *AAAI Conference on Artificial Intelligence*. 2017, pp. 4068–4074.

[22] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[23] Kaiming He et al. "Mask r-cnn". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2980–2988.

[24] Robert fffdfffdfffdBuzzfffdfffdfffd Hill. "Retina identification". In: *Biometrics: Personal Identification in Networked Society* (1996), pp. 123–141.

[25] Geoffrey Hinton. "Deep belief nets". In: *Encyclopedia of Machine Learning*. Springer, 2011, pp. 267–269.

[26] Chen Huang et al. "Learning deep representation for imbalanced classification". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5375–5384.

[27] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: ().

[28] Anil K Jain, Patrick Flynn, and Arun A Ross. *Handbook of biometrics*. Springer Science & Business Media, 2007.

[29] J Daugman Jan. *How Iris Recognition works, IEEE Transactions on Circuits and systems for video Technology*. 2004.

[30] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: `http://arxiv.org/abs/1412.6980`.

[31] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images". In: (2009).

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[33] Ajay Kumar et al. "Personal verification using palmprint and hand geometry biometric". In: *International Conference on Audio-and Video-Based Biometric Person Authentication*. Springer. 2003, pp. 668–678.

[34] Neeraj Kumar et al. "Attribute and simile classifiers for face verification". In: *IEEE International Conference on Computer Vision*. 2009, pp. 365–372.

[35] John Lafferty and Chengxiang Zhai. "Document Language Models, Query Models, and Risk Minimization for Information Retrieval". In: *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2001, pp. 111–119.

[36] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* ().

[37] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[38] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *IEEE International Conference on Computer Vision*. 2015, pp. 3730–3738.

[39] Ping Luo, Xiaogang Wang, and Xiaoou Tang. "A deep sum-product architecture for robust facial attributes analysis". In: *IEEE International Conference on Computer Vision*. 2013, pp. 2864–2871.

[40] Davide Maltoni et al. *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[41] Xudong Mao et al. "Least squares generative adversarial networks". In: *arXiv preprint ArXiv:1611.04076* (2016).

[42] Bhaskar Mitra and Nick Craswell. "Neural Models for Information Retrieval". In: *arXiv preprint arXiv:1705.01509* (2017).

[43] Paritosh Mittal et al. "Composite sketch recognition using saliency and attribute feedback". In: *Information Fusion* 33 (2017), pp. 86–99.

[44] Fabian Monrose and Aviel D Rubin. "Keystroke dynamics as a biometric for authentication". In: *Future Generation computer systems* 16.4 (2000), pp. 351–359.

[45] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition." In: *British Machine Vision Conference*. Vol. 1. 3. 2015.

[46] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435.

[47] Réjean Plamondon and Sargur N Srihari. "Online and off-line handwriting recognition: a comprehensive survey". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.1 (2000), pp. 63–84.

[48] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[49] Andras Rozsa, Ethan M Rudd, and Terrance E Boult. "Adversarial diversity and hard positive generation". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 25–32.

[50] Ethan M Rudd, Manuel Günther, and Terrance E Boult. "Moon: A mixed objective optimization network for the recognition of facial attributes". In: *European Conference on Computer Vision*. 2016, pp. 19–35.

[51] Albert Ali Salah. "Machine learning for biometrics". In: *Machine Learning: Concepts, Methodologies, Tools and Applications*. IGI Global, 2012, pp. 704–723.

[52] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.

[53] Pouya Samangouei, Vishal M Patel, and Rama Chellappa. "Attribute-based continuous user authentication on mobile devices". In: *IEEE International Conference on Biometrics Theory, Applications and Systems*. 2015, pp. 1–8.

[54] Sudeep Sarkar et al. "The humanid gait challenge problem: Data sets, performance, and analysis". In: *IEEE transactions on pattern analysis and machine intelligence* 27.2 (2005), pp. 162–177.

[55] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[56] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[57] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[58] L. Tan, Z. Li, and Q. Yu. "Deep face attributes recognition using spatial transformer network". In: *IEEE International Conference on Information and Automation*. 2016, pp. 1928–1932.

[59] Andreas Veit, Michael J Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 550–558.

[60] Pascal Vincent et al. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *Journal of Machine Learning Research* 11 (2010), pp. 3371–3408. ISSN: 1532-4435.

[61] J. Wang, Y. Cheng, and R. S. Feris. "Walk and Learn: Facial Attribute Representation Learning from Egocentric Video and Contextual Data". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2295–2304.

[62] Y. Wong et al. "Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition". In: *Computer Vision and Pattern Recognition Workshops*. 2011, pp. 74–81.

[63] Xian Wu, Kun Xu, and Peter Hall. "A survey of image synthesis and editing with generative adversarial networks". In: *Tsinghua Science and Technology* 22.6 (2017), pp. 660–674.

[64] Junbo Zhao, Michael Mathieu, and Yann LeCun. "Energy-based generative adversarial network". In: *arXiv preprint arXiv:1609.03126* (2016).

[65] Wenyi Zhao et al. "Face recognition: A literature survey". In: *ACM computing surveys (CSUR)* 35.4 (2003), pp. 399–458.

[66] Yang Zhong, Josephine Sullivan, and Haibo Li. "Face attribute prediction using off-the-shelf CNN features". In: *IEEE International Conference on Biometrics*. 2016, pp. 1–7.

[67] Yang Zhong, Josephine Sullivan, and Haibo Li. "Leveraging mid-level deep representations for predicting face attributes in the wild". In: *IEEE International Conference on Image Processing*. 2016, pp. 3239–3243.

[68] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: ().

[69] X. Zhu et al. "Data Augmentation in Emotion Classification Using Generative Adversarial Networks". In: *ArXiv e-prints* (Nov. 2017). arXiv: `1711.00648 [cs.CV]`.